

L4Re Operating System Framework

Generated on Sun May 17 2026 11:56:04 for L4Re Operating System Framework by Doxygen
1.15.0

Sun May 17 2026 11:56:04

| | |
|--|----------|
| 1 Overview | 1 |
| 2 Introduction | 3 |
| 2.1 L4Re Microkernel | 3 |
| 2.1.1 Communication | 3 |
| 2.1.2 Kernel Objects | 4 |
| 2.2 L4Re System Structure | 4 |
| 2.3 L4Re Runtime Environment | 6 |
| 3 Programming for L4Re | 7 |
| 3.1 L4 Inter-Process Communication (IPC) | 7 |
| 3.1.1 IPC mechanism | 8 |
| 3.1.1.1 IPC Flags | 8 |
| 3.1.1.2 Partner capability selector | 9 |
| 3.1.1.3 IPC Label | 9 |
| 3.1.1.4 IPC Message Tag | 9 |
| 3.1.1.5 IPC Timeouts | 10 |
| 3.1.1.6 User-level Thread Control Block | 11 |
| 3.1.1.7 Transfer of Typed Send Items | 13 |
| 3.1.2 Examples | 13 |
| 3.1.2.1 User Thread to Kernel Object | 14 |
| 3.1.2.2 User Thread to User Thread | 14 |
| 3.1.2.3 User Thread to User Object | 16 |
| 3.2 Kernel ABI | 16 |
| 3.2.1 Capability selector and flags | 17 |
| 3.2.2 Label | 17 |
| 3.2.3 Message tag | 17 |
| 3.2.4 Timeouts | 18 |
| 3.2.5 User-level thread control block (UTCB) | 19 |
| 3.2.5.1 Buffer descriptor register | 19 |
| 3.2.6 Typed message items | 20 |
| 3.2.6.1 Flexpages | 20 |
| 3.2.6.2 Send items | 21 |
| 3.2.6.3 Receive items | 21 |
| 3.2.6.4 Return items | 22 |
| 3.3 Capabilities and Naming | 23 |
| 3.4 Spaces and Mappings | 24 |
| 3.5 Initial Environment and Application Bootstrapping | 25 |
| 3.5.1 Configuring an application before startup | 26 |
| 3.5.2 Connecting clients and servers | 26 |
| 3.6 Memory management - Data Spaces and the Region Map | 27 |
| 3.6.1 User-level paging | 27 |
| 3.6.1.1 Data spaces | 27 |

| | |
|---|-----------|
| 3.6.1.2 Virtual Memory Handling | 27 |
| 3.6.1.3 Memory Allocation | 27 |
| 3.7 Program Input and Output | 28 |
| 3.8 Initial Memory Allocator and Factory | 28 |
| 3.9 Application and Server Building Blocks | 28 |
| 3.9.1 Creating Additional Application Threads | 28 |
| 3.9.2 Providing a Service | 29 |
| 3.10 Pthread Support | 29 |
| 3.11 Interface Definition Language | 30 |
| 3.11.1 Parameter types for RPC | 31 |
| 3.11.2 Server Side Interface | 32 |
| 3.11.3 RPC Return Types | 32 |
| 3.11.4 RPC Method Declaration | 33 |
| 3.12 L4Re Build System | 33 |
| 3.12.1 Building L4Re | 34 |
| 3.12.2 Writing BID Make Files | 35 |
| 3.12.3 prog.mk - Application Role | 35 |
| 3.12.4 include.mk - Header File Role | 38 |
| 3.12.5 test.mk - Test Application Role | 39 |
| 3.13 Kernel Factory | 46 |
| 3.13.1 Passing parameters for the create stream | 46 |
| 4 L4Re Servers | 49 |
| 4.1 Sigma0, the Root-Pager | 50 |
| 4.1.1 Factory | 51 |
| 4.2 Moe | 51 |
| 4.2.1 Moe objects | 51 |
| 4.2.1.1 Factory | 51 |
| 4.3 Ned, the Init Process | 56 |
| 4.3.1 Lua Bindings for L4Re | 57 |
| 4.3.1.1 Tutorial | 57 |
| 4.3.1.2 Capabilities in Lua | 57 |
| 4.3.1.3 Access to L4Re::Env Capabilities | 57 |
| 4.3.1.4 Constants | 57 |
| 4.3.1.5 Application Startup Details | 58 |
| 4.3.1.6 Reacting on task termination | 60 |
| 4.3.1.7 Control scheduling | 60 |
| 4.3.1.8 Access to the kernel debugger | 60 |
| 4.3.1.9 Using the interactive ned prompt | 61 |
| 4.3.2 Command Line Options | 61 |
| 4.4 Io, the Io Server | 61 |
| 4.5 Virtio Net P2P, a virtual network point-to-point link | 67 |

| | |
|---|------------|
| 4.6 Virtio Net Switch, a virtual network switch | 69 |
| 4.7 Uvmm, the virtual machine monitor | 72 |
| 4.7.1 RAM configuration | 81 |
| 4.8 RTC driver | 82 |
| 4.9 Mag, the GUI Multiplexer | 82 |
| 4.10 Cons, the Console Multiplexer | 85 |
| 5 uvmm_dtg The device tree generator for Uvmm | 89 |
| 6 Bootstrap, the L4 kernel bootstrapper | 91 |
| 7 Deprecated List | 95 |
| 8 Topic Index | 97 |
| 8.1 Topics | 97 |
| 9 Namespace Index | 101 |
| 9.1 Namespace List | 101 |
| 10 Hierarchical Index | 103 |
| 10.1 Class Hierarchy | 103 |
| 11 Data Structure Index | 117 |
| 11.1 Data Structures | 117 |
| 12 File Index | 135 |
| 12.1 File List | 135 |
| 13 Topic Documentation | 149 |
| 13.1 Base API | 149 |
| 13.1.1 Detailed Description | 152 |
| 13.1.2 Basic Macros | 152 |
| 13.1.2.1 Detailed Description | 154 |
| 13.1.2.2 Macro Definition Documentation | 154 |
| 13.1.2.3 Function Documentation | 156 |
| 13.1.3 Fiasco extensions | 157 |
| 13.1.3.1 Detailed Description | 158 |
| 13.1.3.2 Function Documentation | 159 |
| 13.1.3.3 Kernel Debugger | 161 |
| 13.1.3.4 Kernel Information Dump | 169 |
| 13.1.3.5 Kernel Tracing | 170 |
| 13.1.4 Flexpages | 175 |
| 13.1.4.1 Detailed Description | 177 |
| 13.1.4.2 Enumeration Type Documentation | 177 |
| 13.1.4.3 Function Documentation | 181 |
| 13.1.5 C++ IPC Interface Definition | 193 |

| | |
|---|-----|
| 13.1.5.1 Detailed Description | 193 |
| 13.1.5.2 Internal Helpers | 194 |
| 13.1.6 Cache Consistency | 194 |
| 13.1.6.1 Detailed Description | 195 |
| 13.1.6.2 Function Documentation | 195 |
| 13.1.7 Memory related | 198 |
| 13.1.7.1 Detailed Description | 199 |
| 13.1.7.2 Macro Definition Documentation | 200 |
| 13.1.7.3 Enumeration Type Documentation | 203 |
| 13.1.7.4 Function Documentation | 203 |
| 13.1.8 Error codes | 208 |
| 13.1.8.1 Detailed Description | 209 |
| 13.1.8.2 Enumeration Type Documentation | 209 |
| 13.1.9 Object Invocation | 210 |
| 13.1.9.1 Detailed Description | 212 |
| 13.1.9.2 Timeouts during IPC | 212 |
| 13.1.9.3 Macro Definition Documentation | 212 |
| 13.1.9.4 Function Documentation | 213 |
| 13.1.9.5 Message Items | 229 |
| 13.1.9.6 Timeouts | 236 |
| 13.1.9.7 Error Handling | 245 |
| 13.1.9.8 Realtime API | 251 |
| 13.1.9.9 Message Tag | 251 |
| 13.1.9.10 Virtual Registers (UTCBS) | 264 |
| 13.1.10 Kernel Objects | 278 |
| 13.1.10.1 Detailed Description | 280 |
| 13.1.10.2 IPC-Gate API | 281 |
| 13.1.10.3 DMA space | 286 |
| 13.1.10.4 L4 kernel object type information | 286 |
| 13.1.10.5 Factory | 288 |
| 13.1.10.6 Virtual Machines | 300 |
| 13.1.10.7 Interrupt controller | 321 |
| 13.1.10.8 IRQs | 338 |
| 13.1.10.9 Platform Control C API | 353 |
| 13.1.10.10 Scheduler | 359 |
| 13.1.10.11 Kernel-provided semaphore | 367 |
| 13.1.10.12 Task | 369 |
| 13.1.10.13 Thread | 382 |
| 13.1.10.14 Thread groups | 416 |
| 13.1.10.15 Virtual Console | 418 |
| 13.1.11 Kernel Interface Page | 433 |
| 13.1.11.1 Detailed Description | 435 |

| | |
|--|-----|
| 13.1.11.2 Typedef Documentation | 435 |
| 13.1.11.3 Enumeration Type Documentation | 435 |
| 13.1.11.4 Function Documentation | 436 |
| 13.1.11.5 Memory descriptors (C version) | 440 |
| 13.1.12 Capabilities | 445 |
| 13.1.12.1 Detailed Description | 446 |
| 13.1.12.2 Typedef Documentation | 446 |
| 13.1.12.3 Enumeration Type Documentation | 447 |
| 13.1.12.4 Function Documentation | 447 |
| 13.1.13 Memory Operations | 449 |
| 13.1.13.1 Detailed Description | 450 |
| 13.1.13.2 Enumeration Type Documentation | 450 |
| 13.1.13.3 Function Documentation | 451 |
| 13.1.14 Integer Types | 452 |
| 13.1.14.1 Detailed Description | 454 |
| 13.2 EDID parsing functionality | 454 |
| 13.2.1 Detailed Description | 455 |
| 13.2.2 Enumeration Type Documentation | 455 |
| 13.2.2.1 Libedid_consts | 455 |
| 13.2.3 Function Documentation | 455 |
| 13.2.3.1 libedid_check_header() | 455 |
| 13.2.3.2 libedid_checksum() | 455 |
| 13.2.3.3 libedid_dump() | 456 |
| 13.2.3.4 libedid_dump_standard_timings() | 456 |
| 13.2.3.5 libedid_num_ext_blocks() | 456 |
| 13.2.3.6 libedid_pnp_id() | 457 |
| 13.2.3.7 libedid_prefered_resolution() | 457 |
| 13.2.3.8 libedid_revision() | 457 |
| 13.2.3.9 libedid_version() | 458 |
| 13.3 IO interface | 458 |
| 13.3.1 Detailed Description | 459 |
| 13.3.2 Typedef Documentation | 459 |
| 13.3.2.1 l4io_resource_t | 459 |
| 13.3.3 Enumeration Type Documentation | 459 |
| 13.3.3.1 l4io_device_types_t | 459 |
| 13.3.3.2 l4io_iomem_flags_t | 460 |
| 13.3.3.3 l4io_resource_types_t | 460 |
| 13.3.4 Function Documentation | 461 |
| 13.3.4.1 l4io_has_resource() | 461 |
| 13.3.4.2 l4io_lookup_device() | 461 |
| 13.3.4.3 l4io_lookup_resource() | 462 |
| 13.3.4.4 l4io_release_iomem() | 462 |

| | |
|---|-----|
| 13.3.4.5 l4io_release_ioport() | 463 |
| 13.3.4.6 l4io_request_iomem() | 463 |
| 13.3.4.7 l4io_request_iomem_region() | 464 |
| 13.3.4.8 l4io_request_ioport() | 465 |
| 13.3.4.9 l4io_request_resource_iomem() | 465 |
| 13.4 IPC Helpers | 466 |
| 13.4.1 Detailed Description | 466 |
| 13.4.2 Function Documentation | 466 |
| 13.4.2.1 throw_ipc_exception() [1/2] | 466 |
| 13.4.2.2 throw_ipc_exception() [2/2] | 467 |
| 13.5 IRQ handling library | 467 |
| 13.5.1 Detailed Description | 468 |
| 13.5.2 Interface using direct functionality. | 468 |
| 13.5.2.1 Detailed Description | 469 |
| 13.5.2.2 Function Documentation | 469 |
| 13.5.2.3 Interface using direct functionality. | 473 |
| 13.5.3 Interface for asynchronous ISR handlers. | 475 |
| 13.5.3.1 Detailed Description | 476 |
| 13.5.3.2 Function Documentation | 476 |
| 13.5.3.3 Interface for asynchronous ISR handlers with a given IRQ capability. | 477 |
| 13.6 L4 IPC Opcodes | 478 |
| 13.6.1 Detailed Description | 479 |
| 13.6.2 Enumeration Type Documentation | 479 |
| 13.6.2.1 L4_icu_opcode | 479 |
| 13.6.2.2 L4_ipc_gate_ops | 480 |
| 13.6.2.3 L4_platform_ctl_ops | 481 |
| 13.6.2.4 L4_task_ops | 481 |
| 13.6.2.5 L4_thread_ops | 481 |
| 13.6.2.6 L4_vcon_ops | 482 |
| 13.7 L4 VIRTIO Interface | 483 |
| 13.7.1 Detailed Description | 483 |
| 13.7.2 L4 VIRTIO Transport Layer | 483 |
| 13.7.2.1 Detailed Description | 485 |
| 13.7.2.2 Typedef Documentation | 485 |
| 13.7.2.3 Enumeration Type Documentation | 486 |
| 13.7.2.4 Function Documentation | 488 |
| 13.7.3 L4 VIRTIO Block Device | 492 |
| 13.7.3.1 Detailed Description | 493 |
| 13.7.3.2 Enumeration Type Documentation | 493 |
| 13.7.4 L4 VIRTIO Input Device | 494 |
| 13.7.4.1 Detailed Description | 494 |
| 13.7.5 L4 VIRTIO Network Device | 494 |

| | |
|--|-----|
| 13.7.5.1 Detailed Description | 495 |
| 13.8 L4 Vbus functions | 495 |
| 13.8.1 Detailed Description | 496 |
| 13.8.2 Enumeration Type Documentation | 497 |
| 13.8.2.1 L4vbus_dma_domain_assign_flags | 497 |
| 13.8.3 Function Documentation | 497 |
| 13.8.3.1 l4vbus_assign_dma_domain() | 497 |
| 13.8.3.2 l4vbus_get_adr() | 498 |
| 13.8.3.3 l4vbus_get_device() | 498 |
| 13.8.3.4 l4vbus_get_device_by_hid() | 499 |
| 13.8.3.5 l4vbus_get_hid() | 500 |
| 13.8.3.6 l4vbus_get_next_device() | 501 |
| 13.8.3.7 l4vbus_get_resource() | 502 |
| 13.8.3.8 l4vbus_is_compatible() | 502 |
| 13.8.3.9 l4vbus_release_ioport() | 503 |
| 13.8.3.10 l4vbus_request_ioport() | 504 |
| 13.8.3.11 l4vbus_vicu_get_cap() | 505 |
| 13.8.4 L4vbus GPIO functions | 505 |
| 13.8.4.1 Detailed Description | 506 |
| 13.8.4.2 Enumeration Type Documentation | 506 |
| 13.8.4.3 Function Documentation | 507 |
| 13.8.5 L4vbus PCI functions | 516 |
| 13.8.5.1 Detailed Description | 516 |
| 13.8.5.2 Function Documentation | 516 |
| 13.8.6 L4vbus power management functions | 522 |
| 13.8.6.1 Detailed Description | 522 |
| 13.8.6.2 Function Documentation | 522 |
| 13.9 L4Re C Interface | 524 |
| 13.9.1 Detailed Description | 525 |
| 13.9.2 L4Re Util C Interface | 526 |
| 13.9.3 Dataspace interface | 526 |
| 13.9.3.1 Detailed Description | 527 |
| 13.9.3.2 Enumeration Type Documentation | 527 |
| 13.9.3.3 Function Documentation | 528 |
| 13.9.4 Debug interface | 532 |
| 13.9.4.1 Detailed Description | 532 |
| 13.9.4.2 Function Documentation | 532 |
| 13.9.5 DMA Space Interface | 533 |
| 13.9.5.1 Detailed Description | 533 |
| 13.9.5.2 Typedef Documentation | 534 |
| 13.9.5.3 Function Documentation | 534 |
| 13.9.6 Event interface | 537 |

| | |
|--|-----|
| 13.9.6.1 Detailed Description | 537 |
| 13.9.6.2 Function Documentation | 538 |
| 13.9.7 Log interface | 540 |
| 13.9.7.1 Detailed Description | 541 |
| 13.9.7.2 Function Documentation | 541 |
| 13.9.8 Memory allocator | 543 |
| 13.9.8.1 Detailed Description | 544 |
| 13.9.8.2 Enumeration Type Documentation | 544 |
| 13.9.8.3 Function Documentation | 544 |
| 13.9.9 Namespace interface | 548 |
| 13.9.9.1 Detailed Description | 549 |
| 13.9.9.2 Enumeration Type Documentation | 549 |
| 13.9.9.3 Function Documentation | 549 |
| 13.9.10 Parent interface | 552 |
| 13.9.11 Region map interface | 552 |
| 13.9.11.1 Detailed Description | 553 |
| 13.9.11.2 Enumeration Type Documentation | 554 |
| 13.9.11.3 Function Documentation | 554 |
| 13.9.12 Capability allocator | 570 |
| 13.9.12.1 Detailed Description | 571 |
| 13.9.12.2 Function Documentation | 571 |
| 13.9.13 Kumem allocator utility | 571 |
| 13.9.14 Video API | 572 |
| 13.9.14.1 Detailed Description | 573 |
| 13.9.14.2 Typedef Documentation | 573 |
| 13.9.14.3 Enumeration Type Documentation | 574 |
| 13.9.14.4 Function Documentation | 574 |
| 13.9.15 Initial Environment | 580 |
| 13.9.15.1 Detailed Description | 581 |
| 13.9.15.2 Function Documentation | 581 |
| 13.10 L4Re C++ Interface | 585 |
| 13.10.1 Detailed Description | 587 |
| 13.10.2 L4Re Util C++ Interface | 587 |
| 13.10.2.1 Detailed Description | 588 |
| 13.10.2.2 L4Re Capability API | 588 |
| 13.10.2.3 Kumem utilities | 591 |
| 13.10.3 Console API | 592 |
| 13.10.3.1 Detailed Description | 592 |
| 13.10.4 Debugging API | 593 |
| 13.10.4.1 Detailed Description | 593 |
| 13.10.5 L4Re ELF Auxiliary Information | 593 |
| 13.10.5.1 Detailed Description | 594 |

| | |
|--|-----|
| 13.10.5.2 Macro Definition Documentation | 594 |
| 13.10.5.3 Enumeration Type Documentation | 595 |
| 13.10.6 Event API | 596 |
| 13.10.6.1 Detailed Description | 596 |
| 13.10.7 Auxiliary data | 596 |
| 13.10.7.1 Detailed Description | 597 |
| 13.10.8 Logging interface | 597 |
| 13.10.8.1 Detailed Description | 597 |
| 13.10.9 Name-space API | 598 |
| 13.10.9.1 Detailed Description | 598 |
| 13.10.10 Parent API | 598 |
| 13.10.10.1 Detailed Description | 599 |
| 13.10.11 L4Re Protocol identifiers | 599 |
| 13.10.11.1 Detailed Description | 600 |
| 13.10.11.2 Enumeration Type Documentation | 600 |
| 13.10.12 Region map API | 600 |
| 13.10.12.1 Detailed Description | 601 |
| 13.10.13 Video API | 601 |
| 13.10.13.1 Detailed Description | 602 |
| 13.10.14 C++ Exceptions | 602 |
| 13.10.14.1 Detailed Description | 603 |
| 13.10.15 Vbus API | 603 |
| 13.10.15.1 Detailed Description | 604 |
| 13.11 L4SHM-based ring buffer implementation | 604 |
| 13.11.1 Detailed Description | 605 |
| 13.11.2 Sender | 605 |
| 13.11.3 Receiver | 605 |
| 13.11.4 Internal | 605 |
| 13.11.4.1 Detailed Description | 606 |
| 13.11.4.2 Macro Definition Documentation | 606 |
| 13.12 Shared Memory Library | 607 |
| 13.12.1 Detailed Description | 608 |
| 13.12.2 Function Documentation | 609 |
| 13.12.2.1 l4shmc_area_overhead() | 609 |
| 13.12.2.2 l4shmc_area_size() | 609 |
| 13.12.2.3 l4shmc_area_size_free() | 609 |
| 13.12.2.4 l4shmc_attach() | 610 |
| 13.12.2.5 l4shmc_chunk_overhead() | 610 |
| 13.12.2.6 l4shmc_connect_chunk_signal() | 611 |
| 13.12.2.7 l4shmc_create() | 611 |
| 13.12.2.8 l4shmc_get_client_nr() | 612 |
| 13.12.2.9 l4shmc_get_initialized_clients() | 612 |

| | |
|---|-----|
| 13.12.2.10 l4shmc_mark_client_initialized() | 613 |
| 13.12.3 Chunks | 613 |
| 13.12.3.1 Detailed Description | 614 |
| 13.12.3.2 Function Documentation | 614 |
| 13.12.3.3 Producer | 618 |
| 13.12.3.4 Consumer | 621 |
| 13.12.4 Signals | 626 |
| 13.12.4.1 Detailed Description | 626 |
| 13.12.4.2 Function Documentation | 626 |
| 13.12.4.3 Producer | 629 |
| 13.12.4.4 Consumer | 630 |
| 13.13 Sigma0 API | 634 |
| 13.13.1 Detailed Description | 635 |
| 13.13.2 Enumeration Type Documentation | 635 |
| 13.13.2.1 l4sigma0_return_flags_t | 635 |
| 13.13.3 Function Documentation | 636 |
| 13.13.3.1 l4sigma0_debug_dump() | 636 |
| 13.13.3.2 l4sigma0_map_anypage() | 636 |
| 13.13.3.3 l4sigma0_map_errstr() | 637 |
| 13.13.3.4 l4sigma0_map_iomem() | 637 |
| 13.13.3.5 l4sigma0_map_kip() | 638 |
| 13.13.3.6 l4sigma0_map_mem() | 638 |
| 13.13.4 Internal constants | 639 |
| 13.13.4.1 Detailed Description | 640 |
| 13.14 Small C++ Template Library | 640 |
| 13.14.1 Detailed Description | 641 |
| 13.14.2 Function Documentation | 641 |
| 13.14.2.1 clamp() | 641 |
| 13.14.2.2 max() [1/2] | 642 |
| 13.14.2.3 max() [2/2] | 642 |
| 13.14.2.4 min() [1/2] | 643 |
| 13.14.2.5 min() [2/2] | 643 |
| 13.14.2.6 operator new() | 643 |
| 13.15 The L4Re IPC Framework | 644 |
| 13.15.1 Detailed Description | 644 |
| 13.15.2 Server-Side IPC framework | 644 |
| 13.15.2.1 Detailed Description | 645 |
| 13.15.2.2 Enumeration Type Documentation | 646 |
| 13.16 Utility Functions | 646 |
| 13.16.1 Detailed Description | 648 |
| 13.16.2 Function Documentation | 649 |
| 13.16.2.1 l4_sleep() | 649 |

| | | |
|------------|--|-----|
| 13.16.2.2 | l4_touch_ro() | 649 |
| 13.16.2.3 | l4_touch_rw() | 650 |
| 13.16.2.4 | l4_usleep() | 651 |
| 13.16.2.5 | l4util_micros2l4to() | 651 |
| 13.16.2.6 | l4util_splitlog2_hdl() | 652 |
| 13.16.2.7 | l4util_splitlog2_size() | 652 |
| 13.16.3 | CPU related functions | 653 |
| 13.16.3.1 | Detailed Description | 654 |
| 13.16.3.2 | Function Documentation | 654 |
| 13.16.4 | IA32 Port I/O API | 656 |
| 13.16.4.1 | Detailed Description | 657 |
| 13.16.4.2 | Function Documentation | 657 |
| 13.16.5 | Timestamp Counter | 662 |
| 13.16.5.1 | Detailed Description | 663 |
| 13.16.5.2 | Function Documentation | 663 |
| 13.16.6 | Atomic Instructions | 669 |
| 13.16.6.1 | Detailed Description | 671 |
| 13.16.6.2 | Function Documentation | 671 |
| 13.16.7 | Internal functions | 687 |
| 13.16.7.1 | Detailed Description | 687 |
| 13.16.8 | Bit Manipulation | 687 |
| 13.16.8.1 | Detailed Description | 688 |
| 13.16.8.2 | Function Documentation | 688 |
| 13.16.9 | ELF binary format | 694 |
| 13.16.9.1 | Detailed Description | 700 |
| 13.16.9.2 | Macro Definition Documentation | 700 |
| 13.16.9.3 | Enumeration Type Documentation | 702 |
| 13.16.10 | Kernel Interface Page API | 719 |
| 13.16.10.1 | Detailed Description | 719 |
| 13.16.10.2 | Macro Definition Documentation | 719 |
| 13.16.10.3 | Function Documentation | 720 |
| 13.16.11 | Comfortable Command Line Parsing | 721 |
| 13.16.11.1 | Detailed Description | 721 |
| 13.16.11.2 | Function Documentation | 721 |
| 13.16.12 | Random number support | 723 |
| 13.16.12.1 | Detailed Description | 723 |
| 13.16.12.2 | Function Documentation | 723 |
| 13.16.13 | Low-Level Thread Functions | 724 |
| 13.17 | Virtio Net Switch | 724 |
| 13.17.1 | Detailed Description | 725 |
| 13.18 | vCPU Support Library | 725 |
| 13.18.1 | Detailed Description | 726 |

| | |
|---|------------|
| 13.18.2 Function Documentation | 726 |
| 13.18.2.1 l4vcpu_irq_disable() | 726 |
| 13.18.2.2 l4vcpu_irq_disable_save() | 727 |
| 13.18.2.3 l4vcpu_irq_enable() | 728 |
| 13.18.2.4 l4vcpu_irq_restore() | 729 |
| 13.18.2.5 l4vcpu_is_irq_entry() | 730 |
| 13.18.2.6 l4vcpu_is_page_fault_entry() | 730 |
| 13.18.2.7 l4vcpu_print_state() | 731 |
| 13.18.2.8 l4vcpu_wait_for_event() | 731 |
| 13.18.3 Extended vCPU support | 732 |
| 13.18.3.1 Detailed Description | 733 |
| 13.18.3.2 Function Documentation | 733 |
| 14 Namespace Documentation | 735 |
| 14.1 cxx Namespace Reference | 735 |
| 14.1.1 Detailed Description | 737 |
| 14.1.2 Function Documentation | 738 |
| 14.1.2.1 access_once() | 738 |
| 14.1.2.2 gcd() | 739 |
| 14.1.2.3 lcm() | 740 |
| 14.1.2.4 write_now() | 741 |
| 14.2 cxx::Bits Namespace Reference | 741 |
| 14.2.1 Detailed Description | 742 |
| 14.3 Enum_bitops Namespace Reference | 742 |
| 14.3.1 Detailed Description | 742 |
| 14.4 Enum_bitops_impl Namespace Reference | 742 |
| 14.4.1 Detailed Description | 743 |
| 14.5 L4 Namespace Reference | 743 |
| 14.5.1 Detailed Description | 747 |
| 14.5.2 Enumeration Type Documentation | 748 |
| 14.5.2.1 anonymous enum | 748 |
| 14.5.3 Function Documentation | 748 |
| 14.5.3.1 cap_cast() [1/2] | 748 |
| 14.5.3.2 cap_cast() [2/2] | 750 |
| 14.5.3.3 cap_dynamic_cast() | 750 |
| 14.5.3.4 cap_reinterpret_cast() [1/2] | 752 |
| 14.5.3.5 cap_reinterpret_cast() [2/2] | 754 |
| 14.5.3.6 round_order() | 754 |
| 14.5.3.7 trunc_order() | 755 |
| 14.6 L4::lpc Namespace Reference | 756 |
| 14.6.1 Detailed Description | 758 |
| 14.6.2 Function Documentation | 758 |

| | |
|---------------------------------------|-----|
| 14.6.2.1 buf_cp_in() | 758 |
| 14.6.2.2 buf_cp_out() | 759 |
| 14.6.2.3 buf_in() | 759 |
| 14.6.2.4 make_cap() | 760 |
| 14.6.2.5 make_cap_full() | 760 |
| 14.6.2.6 make_cap_grant() | 761 |
| 14.6.2.7 make_cap_rw() | 762 |
| 14.6.2.8 make_cap_rws() | 763 |
| 14.6.2.9 msg_ptr() | 763 |
| 14.6.2.10 read() | 763 |
| 14.6.2.11 str_cp_in() | 764 |
| 14.7 L4::lpc::Msg Namespace Reference | 764 |
| 14.7.1 Detailed Description | 765 |
| 14.7.2 Enumeration Type Documentation | 766 |
| 14.7.2.1 anonymous enum | 766 |
| 14.7.3 Function Documentation | 766 |
| 14.7.3.1 align_to() [1/2] | 766 |
| 14.7.3.2 align_to() [2/2] | 767 |
| 14.7.3.3 check_size() [1/2] | 768 |
| 14.7.3.4 check_size() [2/2] | 768 |
| 14.7.3.5 msg_add() | 769 |
| 14.7.3.6 msg_get() | 770 |
| 14.8 L4::lpc_svr Namespace Reference | 771 |
| 14.8.1 Detailed Description | 772 |
| 14.9 L4::Typeid Namespace Reference | 772 |
| 14.9.1 Detailed Description | 772 |
| 14.10 L4::Types Namespace Reference | 772 |
| 14.10.1 Detailed Description | 774 |
| 14.10.2 Function Documentation | 774 |
| 14.10.2.1 declval() | 774 |
| 14.11 L4Re Namespace Reference | 774 |
| 14.11.1 Detailed Description | 777 |
| 14.11.2 Typedef Documentation | 777 |
| 14.11.2.1 Shared_cap | 777 |
| 14.11.2.2 Shared_del_cap | 778 |
| 14.11.2.3 Unique_cap | 778 |
| 14.11.2.4 Unique_del_cap | 779 |
| 14.11.3 Function Documentation | 779 |
| 14.11.3.1 chkcap() | 779 |
| 14.11.3.2 chkipc() | 781 |
| 14.11.3.3 chksys() [1/3] | 782 |
| 14.11.3.4 chksys() [2/3] | 782 |

| | |
|--|-----|
| 14.11.3.5 chksys() [3/3] | 783 |
| 14.11.3.6 make_shared_cap() | 785 |
| 14.11.3.7 make_shared_del_cap() | 786 |
| 14.11.3.8 make_unique_cap() | 787 |
| 14.11.3.9 make_unique_del_cap() | 788 |
| 14.11.3.10 shared_cap_cast() [1/2] | 788 |
| 14.11.3.11 shared_cap_cast() [2/2] | 789 |
| 14.11.3.12 shared_cap_dynamic_cast() [1/2] | 790 |
| 14.11.3.13 shared_cap_dynamic_cast() [2/2] | 791 |
| 14.11.3.14 shared_cap_reinterpret_cast() [1/2] | 792 |
| 14.11.3.15 shared_cap_reinterpret_cast() [2/2] | 792 |
| 14.11.3.16 shared_del_cap_cast() [1/2] | 793 |
| 14.11.3.17 shared_del_cap_cast() [2/2] | 794 |
| 14.11.3.18 shared_del_cap_dynamic_cast() [1/2] | 795 |
| 14.11.3.19 shared_del_cap_dynamic_cast() [2/2] | 796 |
| 14.11.3.20 shared_del_cap_reinterpret_cast() [1/2] | 796 |
| 14.11.3.21 shared_del_cap_reinterpret_cast() [2/2] | 797 |
| 14.11.3.22 throw_error() | 798 |
| 14.12 L4Re::Util Namespace Reference | 799 |
| 14.12.1 Detailed Description | 802 |
| 14.12.2 Typedef Documentation | 802 |
| 14.12.2.1 Shared_cap | 802 |
| 14.12.2.2 Shared_del_cap | 803 |
| 14.12.2.3 Unique_cap | 804 |
| 14.12.2.4 Unique_del_cap | 804 |
| 14.12.3 Function Documentation | 805 |
| 14.12.3.1 make_shared_cap() | 805 |
| 14.12.3.2 make_shared_del_cap() | 805 |
| 14.12.3.3 make_unique_cap() | 806 |
| 14.12.3.4 make_unique_del_cap() | 806 |
| 14.12.3.5 shared_cap_cast() [1/2] | 807 |
| 14.12.3.6 shared_cap_cast() [2/2] | 807 |
| 14.12.3.7 shared_cap_dynamic_cast() [1/2] | 808 |
| 14.12.3.8 shared_cap_dynamic_cast() [2/2] | 809 |
| 14.12.3.9 shared_cap_reinterpret_cast() [1/2] | 810 |
| 14.12.3.10 shared_cap_reinterpret_cast() [2/2] | 810 |
| 14.12.3.11 shared_del_cap_cast() [1/2] | 811 |
| 14.12.3.12 shared_del_cap_cast() [2/2] | 812 |
| 14.12.3.13 shared_del_cap_dynamic_cast() [1/2] | 813 |
| 14.12.3.14 shared_del_cap_dynamic_cast() [2/2] | 814 |
| 14.12.3.15 shared_del_cap_reinterpret_cast() [1/2] | 814 |
| 14.12.3.16 shared_del_cap_reinterpret_cast() [2/2] | 815 |

| | |
|--|------------|
| 14.13 L4Re::Vfs Namespace Reference | 816 |
| 14.13.1 Detailed Description | 817 |
| 14.14 L4vbus Namespace Reference | 817 |
| 14.14.1 Detailed Description | 817 |
| 14.15 L4virtio Namespace Reference | 818 |
| 14.15.1 Detailed Description | 818 |
| 15 Data Structure Documentation | 819 |
| 15.1 Buffer Struct Reference | 819 |
| 15.1.1 Detailed Description | 821 |
| 15.2 cxx::arith::Ld< V > Struct Template Reference | 821 |
| 15.2.1 Detailed Description | 821 |
| 15.3 cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC > Class Template Reference | 822 |
| 15.3.1 Detailed Description | 825 |
| 15.3.2 Constructor & Destructor Documentation | 826 |
| 15.3.2.1 Avl_map() | 826 |
| 15.3.3 Member Function Documentation | 826 |
| 15.3.3.1 emplace() | 826 |
| 15.3.3.2 insert() | 827 |
| 15.3.3.3 operator[]() [1/2] | 827 |
| 15.3.3.4 operator[]() [2/2] | 828 |
| 15.4 cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC > Class Template Reference | 828 |
| 15.4.1 Detailed Description | 832 |
| 15.5 cxx::Avl_tree< Node, Get_key, Compare > Class Template Reference | 832 |
| 15.5.1 Detailed Description | 837 |
| 15.5.2 Member Typedef Documentation | 838 |
| 15.5.2.1 Iterator | 838 |
| 15.5.3 Member Function Documentation | 838 |
| 15.5.3.1 insert() | 838 |
| 15.5.3.2 remove() | 839 |
| 15.6 cxx::Avl_tree_node Class Reference | 840 |
| 15.6.1 Detailed Description | 842 |
| 15.7 cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc > Class Template Reference | 842 |
| 15.7.1 Detailed Description | 844 |
| 15.7.2 Member Enumeration Documentation | 845 |
| 15.7.2.1 anonymous enum | 845 |
| 15.7.3 Member Function Documentation | 845 |
| 15.7.3.1 alloc() | 845 |
| 15.7.3.2 free() | 845 |
| 15.7.3.3 free_objects() | 846 |
| 15.7.3.4 total_objects() | 846 |
| 15.8 cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::Slab_i Struct Reference | 846 |

| | |
|--|-----|
| 15.8.1 Detailed Description | 847 |
| 15.9 <code>cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc ></code> Class Template Reference | 847 |
| 15.9.1 Detailed Description | 849 |
| 15.9.2 Member Enumeration Documentation | 850 |
| 15.9.2.1 anonymous enum | 850 |
| 15.9.3 Member Function Documentation | 850 |
| 15.9.3.1 <code>alloc()</code> | 850 |
| 15.9.3.2 <code>free()</code> | 851 |
| 15.9.3.3 <code>free_objects()</code> | 851 |
| 15.9.3.4 <code>total_objects()</code> | 851 |
| 15.10 <code>cxx::Bitfield< T, LSB, MSB ></code> Class Template Reference | 852 |
| 15.10.1 Detailed Description | 854 |
| 15.10.2 Member Typedef Documentation | 854 |
| 15.10.2.1 <code>Bits_type</code> | 854 |
| 15.10.2.2 <code>Shift_type</code> | 854 |
| 15.10.3 Member Enumeration Documentation | 855 |
| 15.10.3.1 anonymous enum | 855 |
| 15.10.4 Member Function Documentation | 855 |
| 15.10.4.1 <code>get()</code> | 855 |
| 15.10.4.2 <code>get_unshifted()</code> | 855 |
| 15.10.4.3 <code>set()</code> | 856 |
| 15.10.4.4 <code>set_dirty()</code> | 856 |
| 15.10.4.5 <code>set_unshifted()</code> | 857 |
| 15.10.4.6 <code>set_unshifted_dirty()</code> | 857 |
| 15.10.4.7 <code>val()</code> | 858 |
| 15.10.4.8 <code>val_dirty()</code> | 859 |
| 15.10.4.9 <code>val_unshifted()</code> | 860 |
| 15.11 <code>cxx::Bitfield< T, LSB, MSB >::Value< TT ></code> Class Template Reference | 860 |
| 15.11.1 Detailed Description | 861 |
| 15.12 <code>cxx::Bitfield< T, LSB, MSB >::Value_base< TT ></code> Class Template Reference | 861 |
| 15.12.1 Detailed Description | 862 |
| 15.13 <code>cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT ></code> Class Template Reference | 862 |
| 15.13.1 Detailed Description | 863 |
| 15.14 <code>cxx::Bitmap< BITS ></code> Class Template Reference | 864 |
| 15.14.1 Detailed Description | 867 |
| 15.14.2 Member Function Documentation | 867 |
| 15.14.2.1 <code>scan_zero()</code> | 867 |
| 15.15 <code>cxx::Bitmap_base</code> Class Reference | 868 |
| 15.15.1 Detailed Description | 870 |
| 15.15.2 Member Enumeration Documentation | 871 |
| 15.15.2.1 anonymous enum | 871 |
| 15.15.3 Member Function Documentation | 871 |

| | | |
|------------|--|-----|
| 15.15.3.1 | atomic_clear_bit() | 871 |
| 15.15.3.2 | atomic_get_and_clear() | 872 |
| 15.15.3.3 | atomic_get_and_set() | 872 |
| 15.15.3.4 | atomic_set_bit() | 873 |
| 15.15.3.5 | bit() [1/2] | 873 |
| 15.15.3.6 | bit() [2/2] | 874 |
| 15.15.3.7 | bit_index() | 875 |
| 15.15.3.8 | clear_bit() | 876 |
| 15.15.3.9 | operator[]() [1/2] | 877 |
| 15.15.3.10 | operator[]() [2/2] | 877 |
| 15.15.3.11 | scan_zero() | 878 |
| 15.15.3.12 | set_bit() | 879 |
| 15.15.3.13 | word_index() | 879 |
| 15.16 | cxx::Bitmap_base::Bit Class Reference | 881 |
| 15.16.1 | Detailed Description | 881 |
| 15.17 | cxx::Bitmap_base::Char< BITS > Class Template Reference | 881 |
| 15.17.1 | Detailed Description | 882 |
| 15.18 | cxx::Bitmap_base::Word< BITS > Class Template Reference | 882 |
| 15.18.1 | Detailed Description | 883 |
| 15.19 | cxx::Bits::Avl_map_get_key< KEY_TYPE > Struct Template Reference | 883 |
| 15.19.1 | Detailed Description | 883 |
| 15.20 | cxx::Bits::Avl_set_get_key< KEY_TYPE > Struct Template Reference | 884 |
| 15.20.1 | Detailed Description | 884 |
| 15.21 | cxx::Bits::Avl_set_iter< Node, Key, Node_op > Class Template Reference | 884 |
| 15.21.1 | Detailed Description | 886 |
| 15.21.2 | Constructor & Destructor Documentation | 886 |
| 15.21.2.1 | Avl_set_iter() [1/2] | 886 |
| 15.21.2.2 | Avl_set_iter() [2/2] | 887 |
| 15.21.3 | Member Function Documentation | 887 |
| 15.21.3.1 | operator*() | 887 |
| 15.21.3.2 | operator->() | 887 |
| 15.22 | cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY > Class Template Reference | 888 |
| 15.22.1 | Detailed Description | 891 |
| 15.22.2 | Member Enumeration Documentation | 891 |
| 15.22.2.1 | anonymous enum | 891 |
| 15.22.3 | Constructor & Destructor Documentation | 891 |
| 15.22.3.1 | Base_avl_set() [1/2] | 891 |
| 15.22.3.2 | Base_avl_set() [2/2] | 892 |
| 15.22.4 | Member Function Documentation | 892 |
| 15.22.4.1 | begin() [1/2] | 892 |
| 15.22.4.2 | begin() [2/2] | 892 |
| 15.22.4.3 | emplace() | 893 |

| | |
|---|-----|
| 15.22.4.4 end() [1/2] | 893 |
| 15.22.4.5 end() [2/2] | 893 |
| 15.22.4.6 erase() | 894 |
| 15.22.4.7 find_node() | 894 |
| 15.22.4.8 insert() | 895 |
| 15.22.4.9 lower_bound_node() | 895 |
| 15.22.4.10 operator=() | 896 |
| 15.22.4.11 rbegin() [1/2] | 896 |
| 15.22.4.12 rbegin() [2/2] | 896 |
| 15.22.4.13 remove() | 897 |
| 15.22.4.14 rend() [1/2] | 897 |
| 15.22.4.15 rend() [2/2] | 898 |
| 15.23 cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node Class Reference | 898 |
| 15.23.1 Detailed Description | 899 |
| 15.23.2 Member Function Documentation | 899 |
| 15.23.2.1 operator*() | 899 |
| 15.23.2.2 operator->() | 899 |
| 15.23.2.3 valid() | 900 |
| 15.24 cxx::Bits::Basic_list< POLICY > Class Template Reference | 900 |
| 15.24.1 Detailed Description | 902 |
| 15.24.2 Member Function Documentation | 902 |
| 15.24.2.1 clear() | 902 |
| 15.24.2.2 iter() | 902 |
| 15.25 cxx::Bits::Bst< Node, Get_key, Compare > Class Template Reference | 903 |
| 15.25.1 Detailed Description | 906 |
| 15.25.2 Member Function Documentation | 906 |
| 15.25.2.1 begin() [1/2] | 906 |
| 15.25.2.2 begin() [2/2] | 907 |
| 15.25.2.3 dir() [1/2] | 908 |
| 15.25.2.4 dir() [2/2] | 908 |
| 15.25.2.5 end() [1/2] | 909 |
| 15.25.2.6 end() [2/2] | 910 |
| 15.25.2.7 find() | 910 |
| 15.25.2.8 find_node() | 911 |
| 15.25.2.9 lower_bound_node() | 911 |
| 15.25.2.10 rbegin() [1/2] | 912 |
| 15.25.2.11 rbegin() [2/2] | 913 |
| 15.25.2.12 remove_all() | 913 |
| 15.25.2.13 remove_tree() | 914 |
| 15.25.2.14 rend() [1/2] | 915 |
| 15.25.2.15 rend() [2/2] | 915 |
| 15.26 cxx::Bits::Bst_node Class Reference | 916 |

| | |
|---|-----|
| 15.26.1 Detailed Description | 918 |
| 15.27 <code>cx::Bits::Direction</code> Struct Reference | 918 |
| 15.27.1 Detailed Description | 919 |
| 15.27.2 Member Enumeration Documentation | 919 |
| 15.27.2.1 <code>Direction_e</code> | 919 |
| 15.27.3 Member Function Documentation | 920 |
| 15.27.3.1 <code>operator!()</code> | 920 |
| 15.28 <code>cx::Bits::Smart_ptr_list< ITEM ></code> Class Template Reference | 920 |
| 15.28.1 Detailed Description | 922 |
| 15.28.2 Member Function Documentation | 923 |
| 15.28.2.1 <code>pop_front()</code> | 923 |
| 15.29 <code>cx::Bits::Smart_ptr_list_item< T, STORE_T ></code> Class Template Reference | 923 |
| 15.29.1 Detailed Description | 924 |
| 15.30 <code>cx::Elide_dtor< T ></code> Class Template Reference | 925 |
| 15.30.1 Detailed Description | 925 |
| 15.31 <code>cx::Error</code> Class Reference | 926 |
| 15.31.1 Detailed Description | 926 |
| 15.32 <code>cx::H_list< T, POLICY ></code> Class Template Reference | 926 |
| 15.32.1 Detailed Description | 930 |
| 15.32.2 Member Function Documentation | 930 |
| 15.32.2.1 <code>erase()</code> | 930 |
| 15.32.2.2 <code>insert()</code> | 930 |
| 15.32.2.3 <code>insert_after()</code> | 931 |
| 15.32.2.4 <code>insert_before()</code> | 931 |
| 15.32.2.5 <code>iter()</code> | 932 |
| 15.32.2.6 <code>pop_front()</code> | 933 |
| 15.32.2.7 <code>remove()</code> | 933 |
| 15.32.2.8 <code>replace()</code> | 934 |
| 15.33 <code>cx::H_list_item_t< ELEM_TYPE ></code> Class Template Reference | 934 |
| 15.33.1 Detailed Description | 936 |
| 15.33.2 Constructor & Destructor Documentation | 936 |
| 15.33.2.1 <code>H_list_item_t()</code> | 936 |
| 15.33.2.2 <code>~H_list_item_t()</code> | 937 |
| 15.34 <code>cx::H_list_t< T ></code> Struct Template Reference | 937 |
| 15.34.1 Detailed Description | 941 |
| 15.35 <code>cx::List< D, Alloc ></code> Class Template Reference | 941 |
| 15.35.1 Detailed Description | 942 |
| 15.35.2 Member Function Documentation | 942 |
| 15.35.2.1 <code>operator[]()</code> [1/2] | 942 |
| 15.35.2.2 <code>operator[]()</code> [2/2] | 943 |
| 15.36 <code>cx::List< D, Alloc >::Iter</code> Class Reference | 943 |
| 15.36.1 Detailed Description | 943 |

| | |
|--|-----|
| 15.37 cxx::List_alloc Class Reference | 944 |
| 15.37.1 Detailed Description | 944 |
| 15.37.2 Constructor & Destructor Documentation | 945 |
| 15.37.2.1 List_alloc() | 945 |
| 15.37.3 Member Function Documentation | 945 |
| 15.37.3.1 alloc() | 945 |
| 15.37.3.2 alloc_max() | 946 |
| 15.37.3.3 avail() | 947 |
| 15.37.3.4 free() | 947 |
| 15.38 cxx::List_item Class Reference | 948 |
| 15.38.1 Detailed Description | 949 |
| 15.38.2 Member Function Documentation | 949 |
| 15.38.2.1 push_back() | 949 |
| 15.38.2.2 push_front() | 950 |
| 15.38.2.3 remove() | 950 |
| 15.39 cxx::List_item::Iter Class Reference | 951 |
| 15.39.1 Detailed Description | 953 |
| 15.40 cxx::List_item::T_iter< T, Poly > Class Template Reference | 953 |
| 15.40.1 Detailed Description | 954 |
| 15.41 cxx::Lt_functor< Obj > Struct Template Reference | 955 |
| 15.41.1 Detailed Description | 955 |
| 15.42 cxx::New_allocator< _Type > Class Template Reference | 955 |
| 15.42.1 Detailed Description | 956 |
| 15.43 cxx::Nothrow Class Reference | 957 |
| 15.43.1 Detailed Description | 957 |
| 15.44 cxx::Pair< First, Second > Struct Template Reference | 957 |
| 15.44.1 Detailed Description | 959 |
| 15.44.2 Constructor & Destructor Documentation | 960 |
| 15.44.2.1 Pair() [1/2] | 960 |
| 15.44.2.2 Pair() [2/2] | 960 |
| 15.45 cxx::Pair_first_compare< Cmp, Typ > Class Template Reference | 960 |
| 15.45.1 Detailed Description | 961 |
| 15.45.2 Constructor & Destructor Documentation | 961 |
| 15.45.2.1 Pair_first_compare() | 961 |
| 15.45.3 Member Function Documentation | 962 |
| 15.45.3.1 operator>() | 962 |
| 15.46 cxx::Ref_obj_list_item< T > Struct Template Reference | 962 |
| 15.46.1 Detailed Description | 964 |
| 15.47 cxx::Ref_ptr< T, CNT > Class Template Reference | 964 |
| 15.47.1 Detailed Description | 966 |
| 15.47.2 Constructor & Destructor Documentation | 967 |
| 15.47.2.1 Ref_ptr() [1/3] | 967 |

| | |
|---|------|
| 15.47.2.2 Ref_ptr() [2/3] | 967 |
| 15.47.2.3 Ref_ptr() [3/3] | 968 |
| 15.47.3 Member Function Documentation | 968 |
| 15.47.3.1 get() | 968 |
| 15.47.3.2 ptr() | 968 |
| 15.47.3.3 release() | 968 |
| 15.48 cxx::Result< T > Class Template Reference | 969 |
| 15.48.1 Detailed Description | 969 |
| 15.48.2 Constructor & Destructor Documentation | 969 |
| 15.48.2.1 Result() | 969 |
| 15.49 cxx::S_list< T, POLICY > Class Template Reference | 970 |
| 15.49.1 Detailed Description | 972 |
| 15.49.2 Member Function Documentation | 973 |
| 15.49.2.1 pop_front() | 973 |
| 15.50 cxx::Slab< Type, Slab_size, Max_free, Alloc > Class Template Reference | 973 |
| 15.50.1 Detailed Description | 976 |
| 15.50.2 Member Function Documentation | 976 |
| 15.50.2.1 alloc() | 976 |
| 15.50.2.2 free() | 977 |
| 15.51 cxx::Slab_static< Type, Slab_size, Max_free, Alloc > Class Template Reference | 978 |
| 15.51.1 Detailed Description | 981 |
| 15.51.2 Member Function Documentation | 981 |
| 15.51.2.1 alloc() | 981 |
| 15.52 cxx::static_vector< T, IDX > Class Template Reference | 982 |
| 15.52.1 Detailed Description | 983 |
| 15.53 cxx::String Class Reference | 983 |
| 15.53.1 Detailed Description | 987 |
| 15.53.2 Constructor & Destructor Documentation | 987 |
| 15.53.2.1 String() | 987 |
| 15.53.3 Member Function Documentation | 987 |
| 15.53.3.1 find() | 987 |
| 15.53.3.2 from_dec() | 988 |
| 15.53.3.3 from_hex() | 989 |
| 15.53.3.4 starts_with() | 990 |
| 15.54 cxx::Weak_ref< T > Class Template Reference | 991 |
| 15.54.1 Detailed Description | 993 |
| 15.55 cxx::Weak_ref_base Class Reference | 993 |
| 15.55.1 Detailed Description | 996 |
| 15.56 cxx::Weak_ref_base::List Struct Reference | 996 |
| 15.56.1 Detailed Description | 1000 |
| 15.57 Elf32_Auxv Struct Reference | 1000 |
| 15.57.1 Detailed Description | 1000 |

| | |
|-----------------------------------|------|
| 15.57.2 Field Documentation | 1001 |
| 15.57.2.1 atype | 1001 |
| 15.58 Elf32_Dyn Struct Reference | 1001 |
| 15.58.1 Detailed Description | 1001 |
| 15.58.2 Field Documentation | 1002 |
| 15.58.2.1 d_tag | 1002 |
| 15.59 Elf32_Ehdr Struct Reference | 1002 |
| 15.59.1 Detailed Description | 1003 |
| 15.59.2 Field Documentation | 1003 |
| 15.59.2.1 e_flags | 1003 |
| 15.59.2.2 e_machine | 1004 |
| 15.59.2.3 e_type | 1004 |
| 15.59.2.4 e_version | 1004 |
| 15.60 Elf32_Phdr Struct Reference | 1005 |
| 15.60.1 Detailed Description | 1005 |
| 15.60.2 Field Documentation | 1006 |
| 15.60.2.1 p_flags | 1006 |
| 15.60.2.2 p_type | 1006 |
| 15.61 Elf32_Rel Struct Reference | 1006 |
| 15.61.1 Detailed Description | 1007 |
| 15.62 Elf32_Rela Struct Reference | 1007 |
| 15.62.1 Detailed Description | 1007 |
| 15.63 Elf32_Shdr Struct Reference | 1008 |
| 15.63.1 Detailed Description | 1009 |
| 15.63.2 Field Documentation | 1009 |
| 15.63.2.1 sh_flags | 1009 |
| 15.63.2.2 sh_type | 1009 |
| 15.64 Elf32_Sym Struct Reference | 1010 |
| 15.64.1 Detailed Description | 1010 |
| 15.65 Elf64_Auxv Struct Reference | 1011 |
| 15.65.1 Detailed Description | 1011 |
| 15.65.2 Field Documentation | 1011 |
| 15.65.2.1 atype | 1011 |
| 15.66 Elf64_Dyn Struct Reference | 1012 |
| 15.66.1 Detailed Description | 1012 |
| 15.66.2 Field Documentation | 1012 |
| 15.66.2.1 d_tag | 1012 |
| 15.67 Elf64_Ehdr Struct Reference | 1013 |
| 15.67.1 Detailed Description | 1014 |
| 15.67.2 Field Documentation | 1014 |
| 15.67.2.1 e_flags | 1014 |
| 15.67.2.2 e_machine | 1014 |

| | |
|---|------|
| 15.67.2.3 e_type | 1015 |
| 15.67.2.4 e_version | 1015 |
| 15.68 Elf64_Phdr Struct Reference | 1015 |
| 15.68.1 Detailed Description | 1016 |
| 15.68.2 Field Documentation | 1016 |
| 15.68.2.1 p_flags | 1016 |
| 15.68.2.2 p_type | 1016 |
| 15.69 Elf64_Rel Struct Reference | 1017 |
| 15.69.1 Detailed Description | 1017 |
| 15.70 Elf64_Rela Struct Reference | 1017 |
| 15.70.1 Detailed Description | 1018 |
| 15.71 Elf64_Shdr Struct Reference | 1018 |
| 15.71.1 Detailed Description | 1019 |
| 15.71.2 Field Documentation | 1019 |
| 15.71.2.1 sh_flags | 1019 |
| 15.71.2.2 sh_type | 1019 |
| 15.72 Elf64_Sym Struct Reference | 1020 |
| 15.72.1 Detailed Description | 1020 |
| 15.73 Enum_bitops::Enable< T > Struct Template Reference | 1021 |
| 15.73.1 Detailed Description | 1022 |
| 15.74 Enum_bitops::Has_marker< typename, typename > Struct Template Reference | 1022 |
| 15.74.1 Detailed Description | 1024 |
| 15.75 Enum_bitops::Has_marker< T, Void< decltype(enum_bitops_enable(declval< T >()))> > Struct Template Reference | 1025 |
| 15.75.1 Detailed Description | 1027 |
| 15.76 L4::Alloc_list Class Reference | 1027 |
| 15.76.1 Detailed Description | 1027 |
| 15.77 L4::Arm_smccc Class Reference | 1028 |
| 15.77.1 Detailed Description | 1028 |
| 15.77.2 Member Function Documentation | 1028 |
| 15.77.2.1 call() | 1028 |
| 15.78 L4::Base_exception Class Reference | 1030 |
| 15.78.1 Detailed Description | 1032 |
| 15.79 L4::Basic_registry Class Reference | 1032 |
| 15.79.1 Detailed Description | 1033 |
| 15.79.2 Member Function Documentation | 1033 |
| 15.79.2.1 dispatch() | 1033 |
| 15.79.2.2 find() | 1034 |
| 15.80 L4::Bounds_error Class Reference | 1035 |
| 15.80.1 Detailed Description | 1037 |
| 15.81 L4::Cap< T > Class Template Reference | 1038 |
| 15.81.1 Detailed Description | 1041 |

| | |
|--|------|
| 15.81.2 Constructor & Destructor Documentation | 1042 |
| 15.81.2.1 Cap() [1/4] | 1042 |
| 15.81.2.2 Cap() [2/4] | 1042 |
| 15.81.2.3 Cap() [3/4] | 1042 |
| 15.81.2.4 Cap() [4/4] | 1043 |
| 15.81.3 Member Function Documentation | 1043 |
| 15.81.3.1 check_castable_from() | 1043 |
| 15.81.3.2 check_convertible_from() | 1043 |
| 15.81.3.3 copy() | 1044 |
| 15.81.3.4 move() | 1044 |
| 15.82 L4::Cap_base Class Reference | 1044 |
| 15.82.1 Detailed Description | 1046 |
| 15.82.2 Member Enumeration Documentation | 1047 |
| 15.82.2.1 Cap_type | 1047 |
| 15.82.2.2 No_init_type | 1047 |
| 15.82.3 Constructor & Destructor Documentation | 1047 |
| 15.82.3.1 Cap_base() [1/2] | 1047 |
| 15.82.3.2 Cap_base() [2/2] | 1048 |
| 15.82.4 Member Function Documentation | 1049 |
| 15.82.4.1 cap() | 1049 |
| 15.82.4.2 copy() | 1050 |
| 15.82.4.3 fpage() | 1051 |
| 15.82.4.4 is_valid() | 1052 |
| 15.82.4.5 move() | 1053 |
| 15.82.4.6 snd_base() | 1054 |
| 15.82.4.7 validate() [1/2] | 1055 |
| 15.82.4.8 validate() [2/2] | 1056 |
| 15.83 L4::Com_error Class Reference | 1057 |
| 15.83.1 Detailed Description | 1059 |
| 15.83.2 Constructor & Destructor Documentation | 1060 |
| 15.83.2.1 Com_error() | 1060 |
| 15.84 L4::Debugger Class Reference | 1060 |
| 15.84.1 Detailed Description | 1064 |
| 15.84.2 Member Function Documentation | 1064 |
| 15.84.2.1 add_image_info() | 1064 |
| 15.84.2.2 get_object_name() | 1065 |
| 15.84.2.3 global_id() | 1065 |
| 15.84.2.4 kobj_to_id() | 1066 |
| 15.84.2.5 query_log_name() | 1067 |
| 15.84.2.6 query_log_typeid() | 1067 |
| 15.84.2.7 query_object_name() | 1068 |
| 15.84.2.8 set_object_name() | 1069 |

| | |
|---|------|
| 15.84.2.9 switch_log() | 1070 |
| 15.85 L4::Element_already_exists Class Reference | 1070 |
| 15.85.1 Detailed Description | 1073 |
| 15.86 L4::Element_not_found Class Reference | 1074 |
| 15.86.1 Detailed Description | 1076 |
| 15.87 L4::Epiface Struct Reference | 1077 |
| 15.87.1 Detailed Description | 1078 |
| 15.87.2 Member Function Documentation | 1078 |
| 15.87.2.1 dispatch() | 1078 |
| 15.87.2.2 get_buffer_demand() | 1079 |
| 15.87.2.3 obj_cap() | 1080 |
| 15.87.2.4 server_iface() | 1080 |
| 15.87.2.5 set_server() | 1081 |
| 15.88 L4::Epiface_t< Derived, IFACE, BASE, bool > Struct Template Reference | 1082 |
| 15.88.1 Detailed Description | 1084 |
| 15.88.2 Member Function Documentation | 1085 |
| 15.88.2.1 dispatch() | 1085 |
| 15.89 L4::Epiface_t0< RPC_IFACE, BASE > Struct Template Reference | 1085 |
| 15.89.1 Detailed Description | 1087 |
| 15.89.2 Member Function Documentation | 1088 |
| 15.89.2.1 obj_cap() | 1088 |
| 15.90 L4::Exception Class Reference | 1088 |
| 15.90.1 Detailed Description | 1089 |
| 15.90.2 Member Function Documentation | 1089 |
| 15.90.2.1 exception() | 1089 |
| 15.91 L4::Exception_tracer Class Reference | 1090 |
| 15.91.1 Detailed Description | 1091 |
| 15.92 L4::Factory Class Reference | 1091 |
| 15.92.1 Detailed Description | 1095 |
| 15.92.2 Member Function Documentation | 1095 |
| 15.92.2.1 create() [1/2] | 1095 |
| 15.92.2.2 create() [2/2] | 1096 |
| 15.92.2.3 create_factory() | 1098 |
| 15.92.2.4 create_gate() | 1099 |
| 15.92.2.5 create_task() | 1100 |
| 15.92.2.6 create_thread_group() | 1101 |
| 15.93 L4::Factory::Lstr Struct Reference | 1102 |
| 15.93.1 Detailed Description | 1103 |
| 15.93.2 Constructor & Destructor Documentation | 1103 |
| 15.93.2.1 Lstr() | 1103 |
| 15.94 L4::Factory::Nil Struct Reference | 1104 |
| 15.94.1 Detailed Description | 1104 |

| | |
|--|------|
| 15.95 L4::Factory::S Class Reference | 1104 |
| 15.95.1 Detailed Description | 1106 |
| 15.95.2 Constructor & Destructor Documentation | 1106 |
| 15.95.2.1 S() [1/2] | 1106 |
| 15.95.2.2 S() [2/2] | 1106 |
| 15.95.2.3 ~S() | 1107 |
| 15.95.3 Member Function Documentation | 1107 |
| 15.95.3.1 operator l4_msgtag_t() | 1107 |
| 15.95.3.2 operator<<() [1/2] | 1108 |
| 15.95.3.3 operator<<() [2/2] | 1108 |
| 15.95.3.4 put() [1/5] | 1109 |
| 15.95.3.5 put() [2/5] | 1109 |
| 15.95.3.6 put() [3/5] | 1110 |
| 15.95.3.7 put() [4/5] | 1110 |
| 15.95.3.8 put() [5/5] | 1111 |
| 15.96 L4::lcu Class Reference | 1111 |
| 15.96.1 Detailed Description | 1115 |
| 15.96.2 Member Function Documentation | 1115 |
| 15.96.2.1 bind() | 1115 |
| 15.96.2.2 info() | 1116 |
| 15.96.2.3 mask() | 1117 |
| 15.96.2.4 msi_info() | 1118 |
| 15.96.2.5 set_mode() | 1119 |
| 15.96.2.6 unbind() | 1120 |
| 15.97 L4::lcu::Info Class Reference | 1121 |
| 15.97.1 Detailed Description | 1122 |
| 15.98 L4::Invalid_capability Class Reference | 1123 |
| 15.98.1 Detailed Description | 1125 |
| 15.98.2 Constructor & Destructor Documentation | 1125 |
| 15.98.2.1 Invalid_capability() | 1125 |
| 15.98.3 Member Function Documentation | 1126 |
| 15.98.3.1 cap() | 1126 |
| 15.99 L4::io_pager Class Reference | 1126 |
| 15.99.1 Detailed Description | 1128 |
| 15.99.2 Member Function Documentation | 1129 |
| 15.99.2.1 io_page_fault() | 1129 |
| 15.100 L4::lommu Class Reference | 1129 |
| 15.100.1 Detailed Description | 1131 |
| 15.100.2 Member Function Documentation | 1132 |
| 15.100.2.1 bind() | 1132 |
| 15.100.2.2 unbind() | 1133 |
| 15.101 L4::IOModifier Class Reference | 1134 |

| | |
|--|------|
| 15.101.1 Detailed Description | 1134 |
| 15.102 L4::lpc::Array< ELEM_TYPE, LEN_TYPE > Struct Template Reference | 1134 |
| 15.102.1 Detailed Description | 1136 |
| 15.103 L4::lpc::Array_in_buf< ELEM_TYPE, LEN_TYPE, MAX > Struct Template Reference | 1137 |
| 15.103.1 Detailed Description | 1139 |
| 15.104 L4::lpc::Array_ref< ELEM_TYPE, LEN_TYPE > Struct Template Reference | 1140 |
| 15.104.1 Detailed Description | 1140 |
| 15.105 L4::lpc::As_value< T > Struct Template Reference | 1141 |
| 15.105.1 Detailed Description | 1141 |
| 15.106 L4::lpc::Call Struct Reference | 1142 |
| 15.106.1 Detailed Description | 1142 |
| 15.107 L4::lpc::Call_t< RIGHTS > Struct Template Reference | 1143 |
| 15.107.1 Detailed Description | 1144 |
| 15.108 L4::lpc::Call_zero_send_timeout Struct Reference | 1144 |
| 15.108.1 Detailed Description | 1145 |
| 15.109 L4::lpc::Cap< T > Class Template Reference | 1145 |
| 15.109.1 Detailed Description | 1148 |
| 15.109.2 Member Enumeration Documentation | 1148 |
| 15.109.2.1 anonymous enum | 1148 |
| 15.109.2.2 Map_type | 1149 |
| 15.109.3 Constructor & Destructor Documentation | 1149 |
| 15.109.3.1 Cap() [1/2] | 1149 |
| 15.109.3.2 Cap() [2/2] | 1150 |
| 15.109.4 Member Function Documentation | 1150 |
| 15.109.4.1 from_ci() | 1150 |
| 15.110 L4::lpc::Gen_fpage Class Reference | 1150 |
| 15.110.1 Detailed Description | 1152 |
| 15.110.2 Member Enumeration Documentation | 1152 |
| 15.110.2.1 Type | 1152 |
| 15.111 L4::lpc::In_out< T > Struct Template Reference | 1153 |
| 15.111.1 Detailed Description | 1153 |
| 15.112 L4::lpc::lostream Class Reference | 1154 |
| 15.112.1 Detailed Description | 1157 |
| 15.112.2 Constructor & Destructor Documentation | 1158 |
| 15.112.2.1 lostream() | 1158 |
| 15.112.3 Member Function Documentation | 1158 |
| 15.112.3.1 call() | 1158 |
| 15.112.3.2 get() [1/3] | 1159 |
| 15.112.3.3 get() [2/3] | 1160 |
| 15.112.3.4 get() [3/3] | 1160 |
| 15.112.3.5 put() [1/2] | 1161 |
| 15.112.3.6 put() [2/2] | 1161 |

| | |
|--|------|
| 15.112.3.7 reply_and_wait() [1/2] | 1161 |
| 15.112.3.8 reply_and_wait() [2/2] | 1162 |
| 15.112.3.9 reset() | 1163 |
| 15.113 L4::lpc::Istream Class Reference | 1164 |
| 15.113.1 Detailed Description | 1166 |
| 15.113.2 Constructor & Destructor Documentation | 1167 |
| 15.113.2.1 Istream() | 1167 |
| 15.113.3 Member Function Documentation | 1168 |
| 15.113.3.1 get() [1/3] | 1168 |
| 15.113.3.2 get() [2/3] | 1168 |
| 15.113.3.3 get() [3/3] | 1169 |
| 15.113.3.4 receive() | 1170 |
| 15.113.3.5 reset() | 1171 |
| 15.113.3.6 skip() | 1172 |
| 15.113.3.7 tag() [1/2] | 1172 |
| 15.113.3.8 tag() [2/2] | 1173 |
| 15.113.3.9 wait() [1/2] | 1173 |
| 15.113.3.10 wait() [2/2] | 1174 |
| 15.114 L4::lpc::Msg::Cnt_val_ops< MTYPE, DIR, CLASS > Struct Template Reference | 1175 |
| 15.114.1 Detailed Description | 1176 |
| 15.115 L4::lpc::Msg::Cls_buffer Struct Reference | 1176 |
| 15.115.1 Detailed Description | 1177 |
| 15.116 L4::lpc::Msg::Cls_data Struct Reference | 1178 |
| 15.116.1 Detailed Description | 1178 |
| 15.117 L4::lpc::Msg::Cls_item Struct Reference | 1179 |
| 15.117.1 Detailed Description | 1179 |
| 15.118 L4::lpc::Msg::Dir_in Struct Reference | 1180 |
| 15.118.1 Detailed Description | 1180 |
| 15.119 L4::lpc::Msg::Dir_out Struct Reference | 1181 |
| 15.119.1 Detailed Description | 1181 |
| 15.120 L4::lpc::Msg::Do_in_data Struct Reference | 1182 |
| 15.120.1 Detailed Description | 1182 |
| 15.121 L4::lpc::Msg::Do_in_items Struct Reference | 1183 |
| 15.121.1 Detailed Description | 1184 |
| 15.122 L4::lpc::Msg::Do_out_data Struct Reference | 1184 |
| 15.122.1 Detailed Description | 1185 |
| 15.123 L4::lpc::Msg::Do_out_items Struct Reference | 1185 |
| 15.123.1 Detailed Description | 1186 |
| 15.124 L4::lpc::Msg::Do_rcv_buffers Struct Reference | 1187 |
| 15.124.1 Detailed Description | 1188 |
| 15.125 L4::lpc::Msg::Elem< Array< A, LEN > & > Struct Template Reference | 1188 |
| 15.125.1 Detailed Description | 1188 |

| | |
|---|------|
| 15.126 L4::lpc::Msg::Elem< Array< A, LEN > > Struct Template Reference | 1189 |
| 15.126.1 Detailed Description | 1189 |
| 15.127 L4::lpc::Msg::Elem< Array_ref< A, LEN > & > Struct Template Reference | 1190 |
| 15.127.1 Detailed Description | 1190 |
| 15.128 L4::lpc::Msg::False Struct Reference | 1191 |
| 15.128.1 Detailed Description | 1192 |
| 15.129 L4::lpc::Msg::Is_valid_rpc_type< T > Struct Template Reference | 1193 |
| 15.129.1 Detailed Description | 1194 |
| 15.130 L4::lpc::Msg::Svr_arg_pack< IPC_TYPE > Struct Template Reference | 1195 |
| 15.130.1 Detailed Description | 1195 |
| 15.131 L4::lpc::Msg::Svr_val_ops< MTYPE, DIR, CLASS > Struct Template Reference | 1195 |
| 15.131.1 Detailed Description | 1196 |
| 15.132 L4::lpc::Msg::True Struct Reference | 1196 |
| 15.132.1 Detailed Description | 1198 |
| 15.133 L4::lpc::Msg_ptr< T > Class Template Reference | 1199 |
| 15.133.1 Detailed Description | 1199 |
| 15.133.2 Constructor & Destructor Documentation | 1200 |
| 15.133.2.1 Msg_ptr() | 1200 |
| 15.134 L4::lpc::Opt< T > Struct Template Reference | 1200 |
| 15.134.1 Detailed Description | 1202 |
| 15.135 L4::lpc::Ostream Class Reference | 1202 |
| 15.135.1 Detailed Description | 1205 |
| 15.135.2 Member Function Documentation | 1205 |
| 15.135.2.1 put() [1/2] | 1205 |
| 15.135.2.2 put() [2/2] | 1206 |
| 15.135.2.3 send() | 1206 |
| 15.135.2.4 tag() [1/2] | 1207 |
| 15.135.2.5 tag() [2/2] | 1207 |
| 15.136 L4::lpc::Out< T > Struct Template Reference | 1208 |
| 15.136.1 Detailed Description | 1209 |
| 15.137 L4::lpc::Rcv_fpage Class Reference | 1209 |
| 15.137.1 Detailed Description | 1212 |
| 15.137.2 Constructor & Destructor Documentation | 1212 |
| 15.137.2.1 Rcv_fpage() | 1212 |
| 15.137.3 Member Function Documentation | 1213 |
| 15.137.3.1 io() | 1213 |
| 15.137.3.2 mem() | 1214 |
| 15.137.3.3 obj() | 1214 |
| 15.137.3.4 rcv_task() | 1215 |
| 15.138 L4::lpc::Ret_array< T > Struct Template Reference | 1216 |
| 15.138.1 Detailed Description | 1217 |
| 15.139 L4::lpc::Send_only Struct Reference | 1217 |

| | |
|---|------|
| 15.139.1 Detailed Description | 1217 |
| 15.140 L4::lpc::Small_buf Class Reference | 1218 |
| 15.140.1 Detailed Description | 1218 |
| 15.140.2 Constructor & Destructor Documentation | 1219 |
| 15.140.2.1 Small_buf() [1/2] | 1219 |
| 15.140.2.2 Small_buf() [2/2] | 1219 |
| 15.141 L4::lpc::Snd_fpage Class Reference | 1220 |
| 15.141.1 Detailed Description | 1223 |
| 15.141.2 Member Enumeration Documentation | 1223 |
| 15.141.2.1 Cacheopt | 1223 |
| 15.141.2.2 Continue | 1224 |
| 15.141.2.3 Map_type | 1224 |
| 15.141.3 Constructor & Destructor Documentation | 1224 |
| 15.141.3.1 Snd_fpage() [1/2] | 1224 |
| 15.141.3.2 Snd_fpage() [2/2] | 1225 |
| 15.141.4 Member Function Documentation | 1226 |
| 15.141.4.1 cap_received() | 1226 |
| 15.141.4.2 id_received() | 1226 |
| 15.141.4.3 io() | 1226 |
| 15.141.4.4 is_compound() | 1227 |
| 15.141.4.5 local_id_received() | 1227 |
| 15.141.4.6 mem() | 1228 |
| 15.141.4.7 obj() | 1229 |
| 15.142 L4::lpc::Str_cp_in< T > Class Template Reference | 1230 |
| 15.142.1 Detailed Description | 1230 |
| 15.142.2 Constructor & Destructor Documentation | 1231 |
| 15.142.2.1 Str_cp_in() | 1231 |
| 15.143 L4::lpc::Varg Class Reference | 1231 |
| 15.143.1 Detailed Description | 1232 |
| 15.143.2 Member Function Documentation | 1233 |
| 15.143.2.1 data() | 1233 |
| 15.143.2.2 get_value() | 1233 |
| 15.143.2.3 is_nil() | 1234 |
| 15.143.2.4 is_of() | 1234 |
| 15.143.2.5 is_of_int() | 1235 |
| 15.143.2.6 length() | 1236 |
| 15.143.2.7 tag() | 1236 |
| 15.143.2.8 type() | 1236 |
| 15.143.2.9 value() | 1237 |
| 15.144 L4::lpc::Varg_list< MAX > Class Template Reference | 1237 |
| 15.144.1 Detailed Description | 1240 |
| 15.145 L4::lpc::Varg_list_ref Class Reference | 1240 |

| | |
|---|------|
| 15.145.1 Detailed Description | 1241 |
| 15.145.2 Constructor & Destructor Documentation | 1242 |
| 15.145.2.1 Varg_list_ref() | 1242 |
| 15.146 L4::lpc::Varg_list_ref::Iterator Class Reference | 1242 |
| 15.146.1 Detailed Description | 1243 |
| 15.147 L4::lpc_gate Class Reference | 1244 |
| 15.147.1 Detailed Description | 1247 |
| 15.147.2 Member Function Documentation | 1248 |
| 15.147.2.1 get_infos() | 1248 |
| 15.148 L4::lpc_svr::Br_manager_no_buffers Class Reference | 1249 |
| 15.148.1 Detailed Description | 1252 |
| 15.148.2 Member Function Documentation | 1253 |
| 15.148.2.1 alloc_buffer_demand() | 1253 |
| 15.149 L4::lpc_svr::Compound_reply Struct Reference | 1254 |
| 15.149.1 Detailed Description | 1255 |
| 15.150 L4::lpc_svr::Dbg_dispatch< R, Exc, Printer > Struct Template Reference | 1255 |
| 15.150.1 Detailed Description | 1257 |
| 15.151 L4::lpc_svr::Default_loop_hooks Struct Reference | 1257 |
| 15.151.1 Detailed Description | 1260 |
| 15.152 L4::lpc_svr::Default_setup_wait Struct Reference | 1260 |
| 15.152.1 Detailed Description | 1261 |
| 15.153 L4::lpc_svr::Default_timeout Struct Reference | 1261 |
| 15.153.1 Detailed Description | 1263 |
| 15.154 L4::lpc_svr::Direct_dispatch< R > Struct Template Reference | 1263 |
| 15.154.1 Detailed Description | 1264 |
| 15.155 L4::lpc_svr::Direct_dispatch< R * > Struct Template Reference | 1265 |
| 15.155.1 Detailed Description | 1266 |
| 15.156 L4::lpc_svr::Exc_dispatch< R, Exc > Struct Template Reference | 1267 |
| 15.156.1 Detailed Description | 1268 |
| 15.157 L4::lpc_svr::Ignore_errors Struct Reference | 1269 |
| 15.157.1 Detailed Description | 1270 |
| 15.158 L4::lpc_svr::Server_iface Class Reference | 1270 |
| 15.158.1 Detailed Description | 1273 |
| 15.158.2 Member Function Documentation | 1273 |
| 15.158.2.1 add_timeout() | 1273 |
| 15.158.2.2 alloc_buffer_demand() | 1274 |
| 15.158.2.3 get_rcv_cap() | 1274 |
| 15.158.2.4 get_rcv_mem() | 1275 |
| 15.158.2.5 rcv_cap() [1/2] | 1275 |
| 15.158.2.6 rcv_cap() [2/2] | 1276 |
| 15.158.2.7 realloc_rcv_cap() | 1277 |
| 15.158.2.8 remove_timeout() | 1277 |

| | |
|---|------|
| 15.158.2.9 take_reply_cap() | 1277 |
| 15.159 L4::lpc_svr::Server_iface::Mem_window Class Reference | 1278 |
| 15.159.1 Detailed Description | 1278 |
| 15.160 L4::lpc_svr::Timeout Class Reference | 1278 |
| 15.160.1 Detailed Description | 1281 |
| 15.160.2 Member Function Documentation | 1281 |
| 15.160.2.1 expired() | 1281 |
| 15.160.2.2 timeout() | 1281 |
| 15.161 L4::lpc_svr::Timeout_queue Class Reference | 1282 |
| 15.161.1 Detailed Description | 1282 |
| 15.161.2 Member Function Documentation | 1283 |
| 15.161.2.1 add() | 1283 |
| 15.161.2.2 handle_expired_timeouts() | 1283 |
| 15.161.2.3 next_timeout() | 1284 |
| 15.161.2.4 remove() | 1284 |
| 15.161.2.5 timeout_expired() | 1285 |
| 15.162 L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN > Class Template Reference | 1285 |
| 15.162.1 Detailed Description | 1289 |
| 15.162.2 Member Function Documentation | 1289 |
| 15.162.2.1 add_timeout() | 1289 |
| 15.162.2.2 remove_timeout() | 1290 |
| 15.163 L4::lrq Class Reference | 1290 |
| 15.163.1 Detailed Description | 1295 |
| 15.163.2 Member Function Documentation | 1296 |
| 15.163.2.1 bind_vcpu() | 1296 |
| 15.163.2.2 detach() | 1297 |
| 15.163.2.3 receive() | 1298 |
| 15.163.2.4 unmask() | 1299 |
| 15.163.2.5 wait() | 1300 |
| 15.164 L4::lrq_eoi Class Reference | 1300 |
| 15.164.1 Detailed Description | 1301 |
| 15.164.2 Member Function Documentation | 1302 |
| 15.164.2.1 unmask() | 1302 |
| 15.165 L4::lrq_handler_object Struct Reference | 1303 |
| 15.165.1 Detailed Description | 1306 |
| 15.166 L4::lrqep_t< Derived, BASE, bool > Struct Template Reference | 1307 |
| 15.166.1 Detailed Description | 1309 |
| 15.166.2 Member Function Documentation | 1310 |
| 15.166.2.1 dispatch() | 1310 |
| 15.166.2.2 obj_cap() | 1311 |
| 15.167 L4::Kip::Mem_desc Class Reference | 1311 |
| 15.167.1 Detailed Description | 1313 |

| | |
|--|------|
| 15.167.2 Member Enumeration Documentation | 1313 |
| 15.167.2.1 Arch_sub_type_common | 1313 |
| 15.167.2.2 Info_sub_type | 1314 |
| 15.167.2.3 Mem_type | 1314 |
| 15.167.3 Constructor & Destructor Documentation | 1315 |
| 15.167.3.1 Mem_desc() | 1315 |
| 15.167.4 Member Function Documentation | 1316 |
| 15.167.4.1 all() [1/2] | 1316 |
| 15.167.4.2 all() [2/2] | 1316 |
| 15.167.4.3 count() [1/2] | 1317 |
| 15.167.4.4 count() [2/2] | 1317 |
| 15.167.4.5 end() | 1318 |
| 15.167.4.6 first() | 1319 |
| 15.167.4.7 is_virtual() | 1320 |
| 15.167.4.8 set() | 1320 |
| 15.167.4.9 size() | 1321 |
| 15.167.4.10 start() | 1321 |
| 15.167.4.11 sub_type() | 1322 |
| 15.167.4.12 type() | 1322 |
| 15.168 L4::Kobject Class Reference | 1322 |
| 15.168.1 Detailed Description | 1323 |
| 15.168.2 Member Function Documentation | 1323 |
| 15.168.2.1 cap() | 1323 |
| 15.168.2.2 dec_refcnt() | 1325 |
| 15.169 L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND > Class Template Reference | 1326 |
| 15.169.1 Detailed Description | 1328 |
| 15.169.2 Member Typedef Documentation | 1328 |
| 15.169.2.1 __lface | 1328 |
| 15.169.2.2 __lface_list | 1329 |
| 15.169.2.3 Class | 1329 |
| 15.169.3 Member Function Documentation | 1329 |
| 15.169.3.1 __check_protocols__() | 1329 |
| 15.169.3.2 c() | 1329 |
| 15.170 L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND > Struct Template Reference | 1330 |
| 15.170.1 Detailed Description | 1331 |
| 15.170.2 Member Typedef Documentation | 1332 |
| 15.170.2.1 __lface | 1332 |
| 15.170.2.2 __lface_list | 1332 |
| 15.170.2.3 Class | 1332 |
| 15.170.3 Member Function Documentation | 1333 |
| 15.170.3.1 __check_protocols__() | 1333 |
| 15.170.3.2 c() | 1333 |

| | |
|---|------|
| 15.171 L4::Kobject_demand< T > Struct Template Reference | 1333 |
| 15.171.1 Detailed Description | 1334 |
| 15.172 L4::Kobject_t< Derived, Base, PROTO, S_DEMAND > Class Template Reference | 1334 |
| 15.172.1 Detailed Description | 1335 |
| 15.173 L4::Kobject_typeid< T > Struct Template Reference | 1336 |
| 15.173.1 Detailed Description | 1337 |
| 15.173.2 Member Typedef Documentation | 1337 |
| 15.173.2.1 Demand | 1337 |
| 15.173.3 Member Function Documentation | 1338 |
| 15.173.3.1 demand() | 1338 |
| 15.173.3.2 id() | 1338 |
| 15.173.3.3 proto_dispatch() | 1339 |
| 15.174 L4::Kobject_typeid< void > Struct Reference | 1340 |
| 15.174.1 Detailed Description | 1341 |
| 15.174.2 Member Function Documentation | 1341 |
| 15.174.2.1 demand() | 1341 |
| 15.174.2.2 id() | 1342 |
| 15.174.2.3 proto_dispatch() | 1342 |
| 15.175 L4::Kobject_x< Derived, ARGS > Struct Template Reference | 1343 |
| 15.175.1 Detailed Description | 1344 |
| 15.176 L4::Lock_guard Class Reference | 1344 |
| 15.176.1 Detailed Description | 1345 |
| 15.176.2 Constructor & Destructor Documentation | 1345 |
| 15.176.2.1 Lock_guard() [1/2] | 1345 |
| 15.176.2.2 Lock_guard() [2/2] | 1346 |
| 15.176.2.3 ~Lock_guard() | 1346 |
| 15.176.3 Member Function Documentation | 1346 |
| 15.176.3.1 operator=() | 1346 |
| 15.176.3.2 status() | 1347 |
| 15.177 L4::Meta Class Reference | 1347 |
| 15.177.1 Detailed Description | 1350 |
| 15.177.2 Member Function Documentation | 1351 |
| 15.177.2.1 interface() | 1351 |
| 15.177.2.2 num_interfaces() | 1351 |
| 15.177.2.3 supports() | 1352 |
| 15.178 L4::Out_of_memory Class Reference | 1353 |
| 15.178.1 Detailed Description | 1356 |
| 15.179 L4::Pager Class Reference | 1357 |
| 15.179.1 Detailed Description | 1359 |
| 15.179.2 Member Function Documentation | 1359 |
| 15.179.2.1 page_fault() | 1359 |
| 15.180 L4::Platform_control Class Reference | 1360 |

| | |
|---|------|
| 15.180.1 Detailed Description | 1363 |
| 15.180.2 Member Function Documentation | 1364 |
| 15.180.2.1 <code>cpu_allow_shutdown()</code> | 1364 |
| 15.180.2.2 <code>cpu_disable()</code> | 1364 |
| 15.180.2.3 <code>cpu_enable()</code> | 1365 |
| 15.180.2.4 <code>system_shutdown()</code> | 1366 |
| 15.180.2.5 <code>system_suspend()</code> | 1367 |
| 15.181 L4::Poll_timeout_counter Class Reference | 1368 |
| 15.181.1 Detailed Description | 1369 |
| 15.181.2 Constructor & Destructor Documentation | 1370 |
| 15.181.2.1 <code>Poll_timeout_counter()</code> | 1370 |
| 15.181.3 Member Function Documentation | 1370 |
| 15.181.3.1 <code>set()</code> | 1370 |
| 15.181.3.2 <code>timed_out()</code> | 1371 |
| 15.182 L4::Poll_timeout_kipclock Class Reference | 1371 |
| 15.182.1 Detailed Description | 1372 |
| 15.182.2 Constructor & Destructor Documentation | 1372 |
| 15.182.2.1 <code>Poll_timeout_kipclock()</code> | 1372 |
| 15.182.3 Member Function Documentation | 1373 |
| 15.182.3.1 <code>set()</code> | 1373 |
| 15.182.3.2 <code>test()</code> | 1373 |
| 15.182.3.3 <code>timed_out()</code> | 1374 |
| 15.183 L4::Proto_t< P > Struct Template Reference | 1375 |
| 15.183.1 Detailed Description | 1375 |
| 15.184 L4::Rcv_endpoint Class Reference | 1375 |
| 15.184.1 Detailed Description | 1378 |
| 15.184.2 Member Function Documentation | 1379 |
| 15.184.2.1 <code>bind_snd_destination()</code> | 1379 |
| 15.184.2.2 <code>bind_thread()</code> | 1380 |
| 15.185 L4::Registry_iface Class Reference | 1381 |
| 15.185.1 Detailed Description | 1383 |
| 15.185.2 Member Function Documentation | 1383 |
| 15.185.2.1 <code>register_irq_obj()</code> | 1383 |
| 15.185.2.2 <code>register_obj()</code> [1/3] | 1384 |
| 15.185.2.3 <code>register_obj()</code> [2/3] | 1384 |
| 15.185.2.4 <code>register_obj()</code> [3/3] | 1385 |
| 15.185.2.5 <code>unregister_obj()</code> | 1385 |
| 15.186 L4::Reply_cap Class Reference | 1386 |
| 15.186.1 Detailed Description | 1386 |
| 15.186.2 Constructor & Destructor Documentation | 1387 |
| 15.186.2.1 <code>Reply_cap()</code> | 1387 |
| 15.186.2.2 <code>~Reply_cap()</code> | 1387 |

| | |
|--|------|
| 15.186.3 Member Function Documentation | 1388 |
| 15.186.3.1 get() | 1388 |
| 15.186.3.2 reply() | 1388 |
| 15.186.3.3 reset() | 1389 |
| 15.187 L4::Reply_cap_alloc Class Reference | 1390 |
| 15.187.1 Detailed Description | 1390 |
| 15.187.2 Member Function Documentation | 1390 |
| 15.187.2.1 alloc() | 1390 |
| 15.187.2.2 free() | 1391 |
| 15.188 L4::Reply_cap_idx Class Reference | 1392 |
| 15.188.1 Detailed Description | 1392 |
| 15.189 L4::Runtime_error Class Reference | 1393 |
| 15.189.1 Detailed Description | 1395 |
| 15.189.2 Constructor & Destructor Documentation | 1395 |
| 15.189.2.1 Runtime_error() | 1395 |
| 15.189.3 Member Function Documentation | 1396 |
| 15.189.3.1 err_no() | 1396 |
| 15.189.3.2 extra_str() | 1397 |
| 15.190 L4::Scheduler Class Reference | 1397 |
| 15.190.1 Detailed Description | 1401 |
| 15.190.2 Member Function Documentation | 1402 |
| 15.190.2.1 idle_time() | 1402 |
| 15.190.2.2 info() | 1403 |
| 15.190.2.3 is_online() | 1403 |
| 15.190.2.4 run_thread() | 1404 |
| 15.191 L4::Semaphore Struct Reference | 1406 |
| 15.191.1 Detailed Description | 1409 |
| 15.191.2 Member Function Documentation | 1409 |
| 15.191.2.1 down() | 1409 |
| 15.191.2.2 up() | 1410 |
| 15.192 L4::Server< LOOP_HOOKS > Class Template Reference | 1411 |
| 15.192.1 Detailed Description | 1415 |
| 15.192.2 Constructor & Destructor Documentation | 1415 |
| 15.192.2.1 Server() | 1415 |
| 15.192.3 Member Function Documentation | 1416 |
| 15.192.3.1 internal_loop() | 1416 |
| 15.192.3.2 loop() | 1417 |
| 15.192.3.3 loop_dbg() | 1417 |
| 15.193 L4::Server_object Class Reference | 1417 |
| 15.193.1 Detailed Description | 1420 |
| 15.193.2 Member Function Documentation | 1420 |
| 15.193.2.1 dispatch() [1/2] | 1420 |

| | |
|--|------|
| 15.193.2.2 dispatch() [2/2] | 1421 |
| 15.194 L4::Server_object_t< IFACE, BASE > Struct Template Reference | 1422 |
| 15.194.1 Detailed Description | 1425 |
| 15.194.2 Member Function Documentation | 1426 |
| 15.194.2.1 dispatch_meta_request() | 1426 |
| 15.194.2.2 get_buffer_demand() | 1426 |
| 15.194.2.3 proto_dispatch() | 1426 |
| 15.195 L4::Server_object_x< Derived, IFACE, BASE > Struct Template Reference | 1427 |
| 15.195.1 Detailed Description | 1431 |
| 15.196 L4::Smart_cap< T, SMART > Class Template Reference | 1431 |
| 15.196.1 Detailed Description | 1435 |
| 15.196.2 Constructor & Destructor Documentation | 1435 |
| 15.196.2.1 Smart_cap() | 1435 |
| 15.197 L4::String Class Reference | 1435 |
| 15.197.1 Detailed Description | 1436 |
| 15.198 L4::Task Class Reference | 1436 |
| 15.198.1 Detailed Description | 1440 |
| 15.198.2 Member Function Documentation | 1440 |
| 15.198.2.1 add_ku_mem() | 1440 |
| 15.198.2.2 cap_equal() | 1441 |
| 15.198.2.3 cap_valid() | 1442 |
| 15.198.2.4 delete_obj() | 1443 |
| 15.198.2.5 map() | 1444 |
| 15.198.2.6 release_cap() | 1445 |
| 15.198.2.7 unmap() | 1446 |
| 15.198.2.8 unmap_batch() | 1448 |
| 15.199 L4::Thread Class Reference | 1449 |
| 15.199.1 Detailed Description | 1452 |
| 15.199.2 Member Function Documentation | 1452 |
| 15.199.2.1 control() | 1452 |
| 15.199.2.2 ex_regs() [1/2] | 1453 |
| 15.199.2.3 ex_regs() [2/2] | 1454 |
| 15.199.2.4 modify_senders() | 1455 |
| 15.199.2.5 register_del_irq() | 1456 |
| 15.199.2.6 register_doorbell_irq() | 1458 |
| 15.199.2.7 stats_time() | 1459 |
| 15.199.2.8 switch_to() | 1459 |
| 15.199.2.9 vcpu_control() | 1460 |
| 15.199.2.10 vcpu_control_ext() | 1461 |
| 15.199.2.11 vcpu_resume_commit() | 1462 |
| 15.199.2.12 vcpu_resume_start() | 1463 |
| 15.200 L4::Thread::Attr Class Reference | 1464 |

| | |
|---|------|
| 15.200.1 Detailed Description | 1465 |
| 15.200.2 Constructor & Destructor Documentation | 1466 |
| 15.200.2.1 Attr() | 1466 |
| 15.200.3 Member Function Documentation | 1466 |
| 15.200.3.1 alien() | 1466 |
| 15.200.3.2 bind() | 1467 |
| 15.200.3.3 exc_handler() [1/2] | 1467 |
| 15.200.3.4 exc_handler() [2/2] | 1468 |
| 15.200.3.5 pager() [1/2] | 1468 |
| 15.200.3.6 pager() [2/2] | 1469 |
| 15.201 L4::Thread::Modify_senders Class Reference | 1470 |
| 15.201.1 Detailed Description | 1470 |
| 15.201.2 Member Function Documentation | 1471 |
| 15.201.2.1 add() | 1471 |
| 15.202 L4::Thread_group Class Reference | 1472 |
| 15.202.1 Detailed Description | 1474 |
| 15.202.2 Member Function Documentation | 1474 |
| 15.202.2.1 add() | 1474 |
| 15.202.2.2 remove() | 1475 |
| 15.203 L4::Triggerable Struct Reference | 1476 |
| 15.203.1 Detailed Description | 1479 |
| 15.203.2 Member Function Documentation | 1479 |
| 15.203.2.1 trigger() | 1479 |
| 15.204 L4::Type_info Struct Reference | 1480 |
| 15.204.1 Detailed Description | 1481 |
| 15.205 L4::Type_info::Demand Class Reference | 1481 |
| 15.205.1 Detailed Description | 1483 |
| 15.205.2 Constructor & Destructor Documentation | 1484 |
| 15.205.2.1 Demand() | 1484 |
| 15.205.3 Member Function Documentation | 1484 |
| 15.205.3.1 no_demand() | 1484 |
| 15.206 L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS > Struct Template Reference | 1485 |
| 15.206.1 Detailed Description | 1488 |
| 15.206.2 Member Enumeration Documentation | 1488 |
| 15.206.2.1 anonymous enum | 1488 |
| 15.207 L4::Type_info::Demand_union_t< D1, D2 > Struct Template Reference | 1488 |
| 15.207.1 Detailed Description | 1491 |
| 15.208 L4::Typeid::Detail::_Rpc< OPCODE, O, X > Struct Template Reference | 1492 |
| 15.208.1 Detailed Description | 1492 |
| 15.209 L4::Typeid::Detail::_Rpc< OPCODE, O, Default_op< R > >::Rpc< Y > Struct Template Reference | 1493 |
| 15.209.1 Detailed Description | 1493 |
| 15.210 L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... > Struct Template Reference | 1493 |

| | |
|---|------|
| 15.210.1 Detailed Description | 1496 |
| 15.211 L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... >::Rpc< Y > Struct Template Reference | 1496 |
| 15.211.1 Detailed Description | 1496 |
| 15.212 L4::Typeid::Detail::_Rpc_end Struct Reference | 1497 |
| 15.212.1 Detailed Description | 1497 |
| 15.213 L4::Typeid::P_dispatch< LIST > Struct Template Reference | 1497 |
| 15.213.1 Detailed Description | 1498 |
| 15.214 L4::Typeid::Raw_ipc< CLASS > Struct Template Reference | 1498 |
| 15.214.1 Detailed Description | 1499 |
| 15.215 L4::Typeid::_Rpc_nocode< OPERATION > Struct Template Reference | 1500 |
| 15.215.1 Detailed Description | 1501 |
| 15.216 L4::Typeid::_Rpc< RPCS > Struct Template Reference | 1502 |
| 15.216.1 Detailed Description | 1503 |
| 15.217 L4::Typeid::_Rpc_code< OPCODE_TYPE > Struct Template Reference | 1504 |
| 15.217.1 Detailed Description | 1505 |
| 15.218 L4::Typeid::_Rpc_code< OPCODE_TYPE >::F< RPCS > Struct Template Reference | 1505 |
| 15.218.1 Detailed Description | 1507 |
| 15.219 L4::Typeid::_Rpc_sys< ARG > Struct Template Reference | 1508 |
| 15.219.1 Detailed Description | 1509 |
| 15.220 L4::Types::_Add_rvalue_reference_helper< T, typename > Struct Template Reference | 1510 |
| 15.220.1 Detailed Description | 1511 |
| 15.221 L4::Types::_Add_rvalue_reference_helper< T, Void< T && > > Struct Template Reference | 1511 |
| 15.221.1 Detailed Description | 1513 |
| 15.222 L4::Types::_Underlying_type_helper< T, bool > Struct Template Reference | 1513 |
| 15.222.1 Detailed Description | 1514 |
| 15.223 L4::Types::_Underlying_type_helper< T, false > Struct Template Reference | 1514 |
| 15.223.1 Detailed Description | 1516 |
| 15.224 L4::Types::Add_rvalue_reference< T > Struct Template Reference | 1516 |
| 15.224.1 Detailed Description | 1516 |
| 15.225 L4::Types::Bool< V > Struct Template Reference | 1517 |
| 15.225.1 Detailed Description | 1517 |
| 15.226 L4::Types::False Struct Reference | 1518 |
| 15.226.1 Detailed Description | 1519 |
| 15.227 L4::Types::Flags< BITS_ENUM, UNDERLYING > Class Template Reference | 1520 |
| 15.227.1 Detailed Description | 1522 |
| 15.227.2 Member Enumeration Documentation | 1523 |
| 15.227.2.1 None_type | 1523 |
| 15.227.3 Constructor & Destructor Documentation | 1523 |
| 15.227.3.1 Flags() [1/2] | 1523 |
| 15.227.3.2 Flags() [2/2] | 1523 |
| 15.227.4 Member Function Documentation | 1524 |
| 15.227.4.1 clear() | 1524 |

| | |
|---|------|
| 15.227.4.2 from_raw() | 1524 |
| 15.228 L4::Types::Flags_ops_t< DT > Struct Template Reference | 1524 |
| 15.228.1 Detailed Description | 1527 |
| 15.229 L4::Types::Flags_t< DT, T > Struct Template Reference | 1527 |
| 15.229.1 Detailed Description | 1530 |
| 15.230 L4::Types::Int_for_size< SIZE, bool > Struct Template Reference | 1531 |
| 15.230.1 Detailed Description | 1531 |
| 15.231 L4::Types::Int_for_type< T > Struct Template Reference | 1531 |
| 15.231.1 Detailed Description | 1532 |
| 15.232 L4::Types::Integral_constant< T, Value > Struct Template Reference | 1532 |
| 15.232.1 Detailed Description | 1533 |
| 15.233 L4::Types::Is_enum< T > Struct Template Reference | 1534 |
| 15.233.1 Detailed Description | 1535 |
| 15.234 L4::Types::Same< A, B > Struct Template Reference | 1536 |
| 15.234.1 Detailed Description | 1537 |
| 15.235 L4::Types::Same_template< T, Template > Struct Template Reference | 1538 |
| 15.235.1 Detailed Description | 1540 |
| 15.236 L4::Types::True Struct Reference | 1541 |
| 15.236.1 Detailed Description | 1542 |
| 15.237 L4::Types::Underlying_type< T > Struct Template Reference | 1543 |
| 15.237.1 Detailed Description | 1544 |
| 15.238 L4::Uart Class Reference | 1544 |
| 15.238.1 Detailed Description | 1547 |
| 15.238.2 Member Function Documentation | 1547 |
| 15.238.2.1 change_mode() | 1547 |
| 15.238.2.2 char_avail() | 1547 |
| 15.238.2.3 enable_rx_irq() | 1548 |
| 15.238.2.4 generic_write() | 1548 |
| 15.238.2.5 get_char() | 1549 |
| 15.238.2.6 mode() | 1549 |
| 15.238.2.7 rate() | 1549 |
| 15.238.2.8 reg_shift() | 1550 |
| 15.238.2.9 shutdown() | 1550 |
| 15.238.2.10 startup() | 1550 |
| 15.238.2.11 write() | 1551 |
| 15.239 L4::Uart_apb Class Reference | 1551 |
| 15.239.1 Detailed Description | 1554 |
| 15.239.2 Member Function Documentation | 1555 |
| 15.239.2.1 change_mode() | 1555 |
| 15.239.2.2 char_avail() | 1555 |
| 15.239.2.3 enable_rx_irq() | 1556 |
| 15.239.2.4 get_char() | 1556 |

| | |
|---|------|
| 15.239.2.5 shutdown() | 1556 |
| 15.239.2.6 startup() | 1557 |
| 15.239.2.7 write() | 1557 |
| 15.240 L4::Unknown_error Class Reference | 1558 |
| 15.240.1 Detailed Description | 1560 |
| 15.241 L4::Vcon Class Reference | 1560 |
| 15.241.1 Detailed Description | 1564 |
| 15.241.2 Member Function Documentation | 1565 |
| 15.241.2.1 get_attr() | 1565 |
| 15.241.2.2 read() | 1565 |
| 15.241.2.3 read_with_flags() | 1566 |
| 15.241.2.4 send() | 1567 |
| 15.241.2.5 set_attr() | 1568 |
| 15.241.2.6 write() | 1569 |
| 15.242 L4::Vm Class Reference | 1570 |
| 15.242.1 Detailed Description | 1574 |
| 15.243 l4_buf_regs_t Struct Reference | 1574 |
| 15.243.1 Detailed Description | 1575 |
| 15.243.2 Field Documentation | 1575 |
| 15.243.2.1 bdr | 1575 |
| 15.244 l4_exc_regs_t Struct Reference | 1575 |
| 15.244.1 Detailed Description | 1578 |
| 15.244.2 Field Documentation | 1578 |
| 15.244.2.1 flags | 1578 |
| 15.244.2.2 ss | 1578 |
| 15.245 l4_fpage_t Union Reference | 1579 |
| 15.245.1 Detailed Description | 1579 |
| 15.246 l4_icu_info_t Struct Reference | 1579 |
| 15.246.1 Detailed Description | 1581 |
| 15.246.2 Field Documentation | 1581 |
| 15.246.2.1 features | 1581 |
| 15.247 l4_icu_msi_info_t Struct Reference | 1581 |
| 15.247.1 Detailed Description | 1582 |
| 15.248 l4_kernel_info_mem_desc_t Struct Reference | 1582 |
| 15.248.1 Detailed Description | 1582 |
| 15.249 l4_kernel_info_t Struct Reference | 1583 |
| 15.249.1 Detailed Description | 1584 |
| 15.250 l4_msg_regs_t Union Reference | 1584 |
| 15.250.1 Detailed Description | 1585 |
| 15.251 l4_msgtag_t Struct Reference | 1585 |
| 15.251.1 Detailed Description | 1586 |
| 15.251.2 Member Function Documentation | 1587 |

| | |
|--|------|
| 15.251.2.1 flags() | 1587 |
| 15.251.2.2 has_error() | 1587 |
| 15.252 l4_sched_cpu_set_t Struct Reference | 1588 |
| 15.252.1 Detailed Description | 1589 |
| 15.252.2 Member Function Documentation | 1589 |
| 15.252.2.1 granularity() | 1589 |
| 15.252.2.2 offset() | 1590 |
| 15.252.2.3 set() | 1590 |
| 15.252.3 Field Documentation | 1591 |
| 15.252.3.1 gran_offset | 1591 |
| 15.253 l4_sched_param_t Struct Reference | 1592 |
| 15.253.1 Detailed Description | 1592 |
| 15.253.2 Field Documentation | 1593 |
| 15.253.2.1 prio | 1593 |
| 15.254 l4_snd_fpage_t Struct Reference | 1593 |
| 15.254.1 Detailed Description | 1594 |
| 15.255 l4_thread_regs_t Struct Reference | 1594 |
| 15.255.1 Detailed Description | 1594 |
| 15.255.2 Field Documentation | 1595 |
| 15.255.2.1 error | 1595 |
| 15.255.2.2 free_marker | 1595 |
| 15.256 l4_timeout_s Struct Reference | 1595 |
| 15.256.1 Detailed Description | 1596 |
| 15.257 l4_timeout_t Union Reference | 1596 |
| 15.257.1 Detailed Description | 1597 |
| 15.258 l4_vcon_attr_t Struct Reference | 1597 |
| 15.258.1 Detailed Description | 1598 |
| 15.258.2 Member Function Documentation | 1598 |
| 15.258.2.1 set_raw() | 1598 |
| 15.259 l4_vcpu_arch_state_t Struct Reference | 1599 |
| 15.259.1 Detailed Description | 1599 |
| 15.260 l4_vcpu_ipc_regs_t Struct Reference | 1599 |
| 15.260.1 Detailed Description | 1600 |
| 15.261 l4_vcpu_regs_t Struct Reference | 1601 |
| 15.261.1 Detailed Description | 1602 |
| 15.261.2 Field Documentation | 1603 |
| 15.261.2.1 ax | 1603 |
| 15.261.2.2 bp | 1603 |
| 15.261.2.3 bx | 1603 |
| 15.261.2.4 cx | 1603 |
| 15.261.2.5 di | 1603 |
| 15.261.2.6 dx | 1604 |

| | |
|---|------|
| 15.261.2.7 si | 1604 |
| 15.262 l4_vcpu_state_t Struct Reference | 1604 |
| 15.262.1 Detailed Description | 1607 |
| 15.262.2 Field Documentation | 1607 |
| 15.262.2.1 version | 1607 |
| 15.263 l4_vm_state_t Struct Reference | 1608 |
| 15.263.1 Detailed Description | 1608 |
| 15.264 l4_vm_svm_vmcb_control_area Struct Reference | 1608 |
| 15.264.1 Detailed Description | 1609 |
| 15.265 l4_vm_svm_vmcb_state_save_area Struct Reference | 1609 |
| 15.265.1 Detailed Description | 1610 |
| 15.266 l4_vm_svm_vmcb_state_save_area_seg Struct Reference | 1610 |
| 15.266.1 Detailed Description | 1610 |
| 15.267 l4_vm_svm_vmcb_t Struct Reference | 1611 |
| 15.267.1 Detailed Description | 1611 |
| 15.268 l4_vm_tz_state Struct Reference | 1612 |
| 15.268.1 Detailed Description | 1612 |
| 15.269 l4_vm_vmx_vcpu_infos_t Struct Reference | 1612 |
| 15.269.1 Detailed Description | 1613 |
| 15.269.2 Field Documentation | 1613 |
| 15.269.2.1 df1 | 1613 |
| 15.270 l4_vm_vmx_vcpu_state_t Struct Reference | 1613 |
| 15.270.1 Detailed Description | 1614 |
| 15.271 l4_vm_vmx_vcpu_vmcs_t Struct Reference | 1615 |
| 15.271.1 Detailed Description | 1616 |
| 15.272 l4_vmx_offset_table_t Struct Reference | 1616 |
| 15.272.1 Detailed Description | 1617 |
| 15.273 L4drivers::Mmio_register_block< MAX_BITS > Struct Template Reference | 1618 |
| 15.273.1 Detailed Description | 1619 |
| 15.274 L4drivers::Register_block< MAX_BITS, BLOCK > Class Template Reference | 1620 |
| 15.274.1 Detailed Description | 1620 |
| 15.274.2 Member Function Documentation | 1621 |
| 15.274.2.1 operator[]() [1/2] | 1621 |
| 15.274.2.2 operator[]() [2/2] | 1622 |
| 15.274.2.3 r() [1/2] | 1623 |
| 15.274.2.4 r() [2/2] | 1623 |
| 15.275 L4drivers::Register_block_base< MAX_BITS > Struct Template Reference | 1624 |
| 15.275.1 Detailed Description | 1625 |
| 15.276 L4drivers::Register_block_impl< BASE, MAX_BITS > Struct Template Reference | 1625 |
| 15.276.1 Detailed Description | 1627 |
| 15.277 L4drivers::Register_block_tmpl< BLOCK > Class Template Reference | 1627 |
| 15.277.1 Detailed Description | 1628 |

| | |
|---|------|
| 15.278 L4drivers::Register_tmpl< BITS, BLOCK > Class Template Reference | 1628 |
| 15.278.1 Detailed Description | 1631 |
| 15.278.2 Member Function Documentation | 1631 |
| 15.278.2.1 clear() | 1631 |
| 15.278.2.2 modify() | 1632 |
| 15.278.2.3 operator=() | 1632 |
| 15.278.2.4 set() | 1633 |
| 15.278.2.5 write() | 1633 |
| 15.279 L4drivers::Ro_register_block< MAX_BITS, BLOCK > Class Template Reference | 1634 |
| 15.279.1 Detailed Description | 1635 |
| 15.279.2 Member Function Documentation | 1635 |
| 15.279.2.1 operator[]() | 1635 |
| 15.279.2.2 r() | 1636 |
| 15.280 L4drivers::Ro_register_tmpl< BITS, BLOCK > Class Template Reference | 1636 |
| 15.280.1 Detailed Description | 1638 |
| 15.280.2 Member Function Documentation | 1638 |
| 15.280.2.1 operator value_type() | 1638 |
| 15.280.2.2 read() | 1638 |
| 15.281 L4Re::Cap_alloc Class Reference | 1639 |
| 15.281.1 Detailed Description | 1640 |
| 15.281.2 Member Function Documentation | 1640 |
| 15.281.2.1 alloc() [1/2] | 1640 |
| 15.281.2.2 alloc() [2/2] | 1641 |
| 15.281.2.3 free() | 1642 |
| 15.282 L4Re::Console Class Reference | 1642 |
| 15.282.1 Detailed Description | 1645 |
| 15.283 L4Re::Core::Ref_ptr< T, CNT > Class Template Reference | 1645 |
| 15.283.1 Detailed Description | 1647 |
| 15.283.2 Constructor & Destructor Documentation | 1648 |
| 15.283.2.1 Ref_ptr() [1/3] | 1648 |
| 15.283.2.2 Ref_ptr() [2/3] | 1648 |
| 15.283.2.3 Ref_ptr() [3/3] | 1649 |
| 15.283.3 Member Function Documentation | 1649 |
| 15.283.3.1 get() | 1649 |
| 15.283.3.2 ptr() | 1649 |
| 15.283.3.3 release() | 1649 |
| 15.284 L4Re::Dataspace Class Reference | 1650 |
| 15.284.1 Detailed Description | 1653 |
| 15.284.2 Member Function Documentation | 1653 |
| 15.284.2.1 allocate() | 1653 |
| 15.284.2.2 clear() | 1654 |
| 15.284.2.3 copy_in() | 1655 |

| | |
|---|------|
| 15.284.2.4 flags() | 1656 |
| 15.284.2.5 info() | 1657 |
| 15.284.2.6 map() | 1658 |
| 15.284.2.7 map_info() | 1660 |
| 15.284.2.8 map_region() | 1661 |
| 15.284.2.9 size() | 1662 |
| 15.285 L4Re::Dataspace::F Struct Reference | 1663 |
| 15.285.1 Detailed Description | 1663 |
| 15.285.2 Member Enumeration Documentation | 1663 |
| 15.285.2.1 anonymous enum | 1663 |
| 15.285.2.2 Flags | 1664 |
| 15.286 L4Re::Dataspace::Stats Struct Reference | 1664 |
| 15.286.1 Detailed Description | 1665 |
| 15.287 L4Re::Debug_obj Class Reference | 1665 |
| 15.287.1 Detailed Description | 1668 |
| 15.287.2 Member Function Documentation | 1668 |
| 15.287.2.1 debug() | 1668 |
| 15.288 L4Re::Default_event_payload Struct Reference | 1669 |
| 15.288.1 Detailed Description | 1670 |
| 15.289 L4Re::Dma_space Class Reference | 1670 |
| 15.289.1 Detailed Description | 1672 |
| 15.289.2 Member Typedef Documentation | 1672 |
| 15.289.2.1 Attributes | 1672 |
| 15.289.3 Member Enumeration Documentation | 1672 |
| 15.289.3.1 Attribute | 1672 |
| 15.289.3.2 Direction | 1673 |
| 15.289.3.3 Space_attrib | 1673 |
| 15.289.4 Member Function Documentation | 1674 |
| 15.289.4.1 associate() | 1674 |
| 15.289.4.2 disassociate() | 1675 |
| 15.289.4.3 map() | 1675 |
| 15.289.4.4 unmap() | 1676 |
| 15.290 L4Re::Env Class Reference | 1677 |
| 15.290.1 Detailed Description | 1680 |
| 15.290.2 Member Function Documentation | 1681 |
| 15.290.2.1 dbg_events() [1/2] | 1681 |
| 15.290.2.2 dbg_events() [2/2] | 1681 |
| 15.290.2.3 env() | 1682 |
| 15.290.2.4 factory() [1/2] | 1683 |
| 15.290.2.5 factory() [2/2] | 1683 |
| 15.290.2.6 first_free_cap() [1/2] | 1683 |
| 15.290.2.7 first_free_cap() [2/2] | 1684 |

| | |
|---|------|
| 15.290.2.8 first_free_reply_cap() [1/2] | 1684 |
| 15.290.2.9 first_free_reply_cap() [2/2] | 1684 |
| 15.290.2.10 first_free_utcb() [1/2] | 1684 |
| 15.290.2.11 first_free_utcb() [2/2] | 1685 |
| 15.290.2.12 get() | 1685 |
| 15.290.2.13 get_cap() [1/2] | 1686 |
| 15.290.2.14 get_cap() [2/2] | 1686 |
| 15.290.2.15 initial_caps() [1/2] | 1687 |
| 15.290.2.16 initial_caps() [2/2] | 1688 |
| 15.290.2.17 itas() [1/2] | 1688 |
| 15.290.2.18 itas() [2/2] | 1688 |
| 15.290.2.19 log() [1/2] | 1688 |
| 15.290.2.20 log() [2/2] | 1689 |
| 15.290.2.21 main_thread() [1/2] | 1689 |
| 15.290.2.22 main_thread() [2/2] | 1689 |
| 15.290.2.23 mem_alloc() [1/2] | 1689 |
| 15.290.2.24 mem_alloc() [2/2] | 1690 |
| 15.290.2.25 parent() [1/2] | 1690 |
| 15.290.2.26 parent() [2/2] | 1690 |
| 15.290.2.27 rm() [1/2] | 1691 |
| 15.290.2.28 rm() [2/2] | 1691 |
| 15.290.2.29 scheduler() [1/2] | 1692 |
| 15.290.2.30 scheduler() [2/2] | 1692 |
| 15.290.2.31 task() | 1692 |
| 15.290.2.32 utcb_area() [1/2] | 1693 |
| 15.290.2.33 utcb_area() [2/2] | 1693 |
| 15.291 L4Re::Event Class Reference | 1693 |
| 15.291.1 Detailed Description | 1697 |
| 15.291.2 Member Function Documentation | 1697 |
| 15.291.2.1 get_axis_info() | 1697 |
| 15.291.2.2 get_buffer() | 1699 |
| 15.291.2.3 get_num_streams() | 1700 |
| 15.291.2.4 get_stream_info() | 1700 |
| 15.291.2.5 get_stream_info_for_id() | 1701 |
| 15.291.2.6 get_stream_state_for_id() | 1702 |
| 15.292 L4Re::Event_buffer_t< PAYLOAD > Class Template Reference | 1703 |
| 15.292.1 Detailed Description | 1705 |
| 15.292.2 Constructor & Destructor Documentation | 1706 |
| 15.292.2.1 Event_buffer_t() | 1706 |
| 15.292.3 Member Function Documentation | 1706 |
| 15.292.3.1 next() | 1706 |
| 15.292.3.2 put() | 1706 |

| | |
|--|------|
| 15.293 L4Re::Event_buffer_t< PAYLOAD >::Event Struct Reference | 1707 |
| 15.293.1 Detailed Description | 1707 |
| 15.294 L4Re::Inhibitor Class Reference | 1708 |
| 15.294.1 Detailed Description | 1710 |
| 15.294.2 Member Enumeration Documentation | 1711 |
| 15.294.2.1 anonymous enum | 1711 |
| 15.294.3 Member Function Documentation | 1711 |
| 15.294.3.1 acquire() | 1711 |
| 15.294.3.2 next_lock_info() | 1712 |
| 15.294.3.3 release() | 1713 |
| 15.295 L4Re::Itas Class Reference | 1714 |
| 15.295.1 Detailed Description | 1717 |
| 15.295.2 Member Enumeration Documentation | 1717 |
| 15.295.2.1 anonymous enum | 1717 |
| 15.295.3 Member Function Documentation | 1717 |
| 15.295.3.1 getitimer() | 1717 |
| 15.295.3.2 raise() | 1718 |
| 15.295.3.3 register_thread() | 1719 |
| 15.295.3.4 setitimer() | 1719 |
| 15.295.3.5 sigaction() | 1720 |
| 15.295.3.6 sigaltstack() | 1721 |
| 15.295.3.7 sigpending() | 1722 |
| 15.295.3.8 sigprocmask() | 1722 |
| 15.295.3.9 unregister_thread() | 1723 |
| 15.296 L4Re::Log Class Reference | 1724 |
| 15.296.1 Detailed Description | 1729 |
| 15.296.2 Member Function Documentation | 1729 |
| 15.296.2.1 print() | 1729 |
| 15.296.2.2 printn() | 1729 |
| 15.297 L4Re::Mem_alloc Class Reference | 1730 |
| 15.297.1 Detailed Description | 1733 |
| 15.297.2 Member Enumeration Documentation | 1734 |
| 15.297.2.1 Mem_alloc_flags | 1734 |
| 15.297.3 Member Function Documentation | 1734 |
| 15.297.3.1 alloc() | 1734 |
| 15.297.3.2 info() | 1735 |
| 15.298 L4Re::Mem_alloc::Stats Struct Reference | 1736 |
| 15.298.1 Detailed Description | 1737 |
| 15.298.2 Field Documentation | 1737 |
| 15.298.2.1 mem_free | 1737 |
| 15.298.2.2 mem_limit | 1737 |
| 15.298.2.3 mem_used | 1738 |

| | |
|---|------|
| 15.298.2.4 quota | 1738 |
| 15.298.2.5 quota_used | 1738 |
| 15.299 L4Re::Mmio_space Struct Reference | 1739 |
| 15.299.1 Detailed Description | 1741 |
| 15.299.2 Member Enumeration Documentation | 1742 |
| 15.299.2.1 Access_width | 1742 |
| 15.299.3 Member Function Documentation | 1742 |
| 15.299.3.1 mmio_read() | 1742 |
| 15.299.3.2 mmio_write() | 1743 |
| 15.300 L4Re::Namespace Class Reference | 1744 |
| 15.300.1 Detailed Description | 1748 |
| 15.300.2 Member Enumeration Documentation | 1748 |
| 15.300.2.1 Query_result_flags | 1748 |
| 15.300.2.2 Query_timeout | 1748 |
| 15.300.2.3 Register_flags | 1749 |
| 15.300.3 Member Function Documentation | 1749 |
| 15.300.3.1 query() [1/2] | 1749 |
| 15.300.3.2 query() [2/2] | 1750 |
| 15.300.3.3 register_obj() | 1751 |
| 15.300.3.4 unlink() | 1753 |
| 15.301 L4Re::Ned::Cmd_control Class Reference | 1754 |
| 15.301.1 Detailed Description | 1754 |
| 15.301.2 Member Function Documentation | 1755 |
| 15.301.2.1 execute() [1/2] | 1755 |
| 15.301.2.2 execute() [2/2] | 1755 |
| 15.302 L4Re::Parent Class Reference | 1756 |
| 15.302.1 Detailed Description | 1759 |
| 15.302.2 Member Function Documentation | 1759 |
| 15.302.2.1 signal() | 1759 |
| 15.303 L4Re::Random Struct Reference | 1761 |
| 15.303.1 Detailed Description | 1764 |
| 15.303.2 Member Function Documentation | 1764 |
| 15.303.2.1 get_random() | 1764 |
| 15.304 L4Re::Rm Class Reference | 1765 |
| 15.304.1 Detailed Description | 1770 |
| 15.304.2 Member Enumeration Documentation | 1770 |
| 15.304.2.1 Detach_flags | 1770 |
| 15.304.2.2 Detach_result | 1771 |
| 15.304.2.3 Region_flag_shifts | 1771 |
| 15.304.3 Member Function Documentation | 1771 |
| 15.304.3.1 attach() [1/2] | 1771 |
| 15.304.3.2 attach() [2/2] | 1773 |

| | |
|--|------|
| 15.304.3.3 detach() [1/3] | 1775 |
| 15.304.3.4 detach() [2/3] | 1776 |
| 15.304.3.5 detach() [3/3] | 1777 |
| 15.304.3.6 find() | 1778 |
| 15.304.3.7 free_area() | 1779 |
| 15.304.3.8 get_areas() | 1780 |
| 15.304.3.9 get_info() | 1781 |
| 15.304.3.10 get_regions() | 1782 |
| 15.304.3.11 reserve_area() [1/2] | 1784 |
| 15.304.3.12 reserve_area() [2/2] | 1785 |
| 15.305 L4Re::Rm::Area Struct Reference | 1786 |
| 15.305.1 Detailed Description | 1786 |
| 15.306 L4Re::Rm::F Struct Reference | 1786 |
| 15.306.1 Detailed Description | 1787 |
| 15.306.2 Member Enumeration Documentation | 1787 |
| 15.306.2.1 Attach_flags | 1787 |
| 15.306.2.2 Region_flags | 1788 |
| 15.307 L4Re::Rm::Region Struct Reference | 1788 |
| 15.307.1 Detailed Description | 1789 |
| 15.308 L4Re::Rm::Unique_region< T > Class Template Reference | 1789 |
| 15.308.1 Detailed Description | 1791 |
| 15.308.2 Constructor & Destructor Documentation | 1791 |
| 15.308.2.1 Unique_region() [1/3] | 1791 |
| 15.308.2.2 Unique_region() [2/3] | 1792 |
| 15.308.2.3 Unique_region() [3/3] | 1792 |
| 15.308.2.4 ~Unique_region() | 1792 |
| 15.308.3 Member Function Documentation | 1793 |
| 15.308.3.1 get() | 1793 |
| 15.308.3.2 is_valid() | 1793 |
| 15.308.3.3 operator=() | 1794 |
| 15.308.3.4 release() | 1794 |
| 15.308.3.5 reset() | 1794 |
| 15.309 L4Re::Smart_cap_auto< Unmap_flags > Class Template Reference | 1795 |
| 15.309.1 Detailed Description | 1795 |
| 15.310 L4Re::Smart_count_cap< Unmap_flags > Class Template Reference | 1795 |
| 15.310.1 Detailed Description | 1796 |
| 15.311 L4Re::Util::_Cap_alloc Class Reference | 1797 |
| 15.311.1 Detailed Description | 1798 |
| 15.311.2 Member Function Documentation | 1799 |
| 15.311.2.1 alloc() [1/2] | 1799 |
| 15.311.2.2 alloc() [2/2] | 1799 |
| 15.311.2.3 free() | 1800 |

| | |
|---|------|
| 15.312 L4Re::Util::Bitmap< BITS > Class Template Reference | 1800 |
| 15.312.1 Detailed Description | 1804 |
| 15.312.2 Member Function Documentation | 1804 |
| 15.312.2.1 scan_zero() | 1804 |
| 15.313 L4Re::Util::Bitmap_base Class Reference | 1805 |
| 15.313.1 Detailed Description | 1807 |
| 15.313.2 Member Enumeration Documentation | 1808 |
| 15.313.2.1 anonymous enum | 1808 |
| 15.313.3 Member Function Documentation | 1808 |
| 15.313.3.1 atomic_clear_bit() | 1808 |
| 15.313.3.2 atomic_get_and_clear() | 1808 |
| 15.313.3.3 atomic_get_and_set() | 1809 |
| 15.313.3.4 atomic_set_bit() | 1809 |
| 15.313.3.5 bit() [1/2] | 1809 |
| 15.313.3.6 bit() [2/2] | 1810 |
| 15.313.3.7 bit_index() | 1810 |
| 15.313.3.8 clear_bit() | 1810 |
| 15.313.3.9 operator[]() [1/2] | 1811 |
| 15.313.3.10 operator[]() [2/2] | 1811 |
| 15.313.3.11 scan_zero() | 1811 |
| 15.313.3.12 set_bit() | 1812 |
| 15.313.3.13 word_index() | 1812 |
| 15.314 L4Re::Util::Bitmap_base::Bit Class Reference | 1813 |
| 15.314.1 Detailed Description | 1813 |
| 15.315 L4Re::Util::Bitmap_base::Char< BITS > Class Template Reference | 1813 |
| 15.315.1 Detailed Description | 1814 |
| 15.316 L4Re::Util::Bitmap_base::Word< BITS > Class Template Reference | 1814 |
| 15.316.1 Detailed Description | 1815 |
| 15.317 L4Re::Util::Br_manager Class Reference | 1815 |
| 15.317.1 Detailed Description | 1818 |
| 15.317.2 Member Function Documentation | 1819 |
| 15.317.2.1 alloc_buffer_demand() | 1819 |
| 15.317.2.2 get_rcv_cap() | 1820 |
| 15.317.2.3 get_rcv_mem() | 1820 |
| 15.317.2.4 realloc_rcv_cap() | 1821 |
| 15.317.2.5 set_rcv_cap_flags() | 1822 |
| 15.318 L4Re::Util::Br_manager_hooks Struct Reference | 1823 |
| 15.318.1 Detailed Description | 1825 |
| 15.319 L4Re::Util::Br_manager_timeout_hooks Struct Reference | 1826 |
| 15.319.1 Detailed Description | 1828 |
| 15.320 L4Re::Util::Cap_alloc_base Class Reference | 1829 |
| 15.320.1 Detailed Description | 1829 |

| | |
|--|------|
| 15.321 L4Re::Util::Counter< COUNTER > Struct Template Reference | 1830 |
| 15.321.1 Detailed Description | 1830 |
| 15.321.2 Member Function Documentation | 1830 |
| 15.321.2.1 dec() | 1830 |
| 15.321.2.2 inc() | 1831 |
| 15.322 L4Re::Util::Counter_atomic< COUNTER > Struct Template Reference | 1831 |
| 15.322.1 Detailed Description | 1832 |
| 15.322.2 Member Function Documentation | 1832 |
| 15.322.2.1 dec() | 1832 |
| 15.322.2.2 inc() | 1832 |
| 15.323 L4Re::Util::Counting_cap_alloc< COUNTERTYPE, Dbg > Class Template Reference | 1833 |
| 15.323.1 Detailed Description | 1834 |
| 15.323.2 Constructor & Destructor Documentation | 1834 |
| 15.323.2.1 Counting_cap_alloc() | 1834 |
| 15.323.3 Member Function Documentation | 1835 |
| 15.323.3.1 alloc() [1/2] | 1835 |
| 15.323.3.2 alloc() [2/2] | 1835 |
| 15.323.3.3 free() | 1836 |
| 15.323.3.4 release() | 1837 |
| 15.323.3.5 setup() | 1838 |
| 15.323.3.6 take() | 1838 |
| 15.324 L4Re::Util::Dataspace_svr Class Reference | 1839 |
| 15.324.1 Detailed Description | 1840 |
| 15.324.2 Member Function Documentation | 1840 |
| 15.324.2.1 allocate() | 1840 |
| 15.324.2.2 clear() | 1841 |
| 15.324.2.3 copy() | 1841 |
| 15.324.2.4 is_static() | 1842 |
| 15.324.2.5 map() | 1842 |
| 15.324.2.6 map_hook() | 1843 |
| 15.324.2.7 map_info() | 1844 |
| 15.324.2.8 page_shift() | 1845 |
| 15.324.2.9 release() | 1845 |
| 15.324.2.10 take() | 1845 |
| 15.325 L4Re::Util::Event_buffer_consumer_t< PAYLOAD > Class Template Reference | 1846 |
| 15.325.1 Detailed Description | 1848 |
| 15.325.2 Member Function Documentation | 1848 |
| 15.325.2.1 foreach_available_event() | 1848 |
| 15.325.2.2 process() | 1849 |
| 15.326 L4Re::Util::Event_buffer_t< PAYLOAD > Class Template Reference | 1849 |
| 15.326.1 Detailed Description | 1852 |
| 15.326.2 Member Function Documentation | 1852 |

| | |
|--|------|
| 15.326.2.1 attach() | 1852 |
| 15.326.2.2 buf() | 1852 |
| 15.326.2.3 detach() | 1853 |
| 15.327 L4Re::Util::Event_svr< SVR > Class Template Reference | 1853 |
| 15.327.1 Detailed Description | 1855 |
| 15.328 L4Re::Util::Event_t< PAYLOAD > Class Template Reference | 1855 |
| 15.328.1 Detailed Description | 1856 |
| 15.328.2 Member Enumeration Documentation | 1857 |
| 15.328.2.1 Mode | 1857 |
| 15.328.3 Member Function Documentation | 1857 |
| 15.328.3.1 buffer() | 1857 |
| 15.328.3.2 init() | 1857 |
| 15.328.3.3 init_poll() | 1858 |
| 15.328.3.4 irq() | 1858 |
| 15.329 L4Re::Util::Item_alloc_base Class Reference | 1859 |
| 15.329.1 Detailed Description | 1859 |
| 15.330 L4Re::Util::Names::Name Class Reference | 1860 |
| 15.330.1 Detailed Description | 1863 |
| 15.331 L4Re::Util::Names::Name_space Class Reference | 1863 |
| 15.331.1 Detailed Description | 1864 |
| 15.331.2 Member Function Documentation | 1864 |
| 15.331.2.1 alloc_dynamic_entry() | 1864 |
| 15.331.2.2 copy_receive_cap() | 1865 |
| 15.331.2.3 free_capability() | 1865 |
| 15.331.2.4 free_dynamic_entry() | 1866 |
| 15.331.2.5 free_epiface() | 1866 |
| 15.331.2.6 get_epiface() | 1867 |
| 15.332 L4Re::Util::Object_registry Class Reference | 1868 |
| 15.332.1 Detailed Description | 1871 |
| 15.332.2 Constructor & Destructor Documentation | 1871 |
| 15.332.2.1 Object_registry() [1/2] | 1871 |
| 15.332.2.2 Object_registry() [2/2] | 1872 |
| 15.332.3 Member Function Documentation | 1872 |
| 15.332.3.1 register_irq_obj() | 1872 |
| 15.332.3.2 register_obj() [1/3] | 1873 |
| 15.332.3.3 register_obj() [2/3] | 1873 |
| 15.332.3.4 register_obj() [3/3] | 1874 |
| 15.332.3.5 unregister_obj() | 1874 |
| 15.333 L4Re::Util::Ref_cap< T > Struct Template Reference | 1875 |
| 15.333.1 Detailed Description | 1876 |
| 15.334 L4Re::Util::Ref_del_cap< T > Struct Template Reference | 1877 |
| 15.334.1 Detailed Description | 1877 |

| | |
|--|------|
| 15.335 L4Re::Util::Registry_server< LOOP_HOOKS > Class Template Reference | 1878 |
| 15.335.1 Detailed Description | 1881 |
| 15.335.2 Constructor & Destructor Documentation | 1881 |
| 15.335.2.1 Registry_server() [1/3] | 1881 |
| 15.335.2.2 Registry_server() [2/3] | 1882 |
| 15.335.2.3 Registry_server() [3/3] | 1882 |
| 15.335.3 Member Function Documentation | 1882 |
| 15.335.3.1 loop() | 1882 |
| 15.335.3.2 loop_dbg() | 1883 |
| 15.336 L4Re::Util::Smart_cap_auto< Unmap_flags > Class Template Reference | 1884 |
| 15.336.1 Detailed Description | 1885 |
| 15.337 L4Re::Util::Smart_count_cap< Unmap_flags > Class Template Reference | 1885 |
| 15.337.1 Detailed Description | 1886 |
| 15.338 L4Re::Util::Vcon_svr< SVR > Class Template Reference | 1886 |
| 15.338.1 Detailed Description | 1887 |
| 15.339 L4Re::Util::Video::Goos_svr Class Reference | 1887 |
| 15.339.1 Detailed Description | 1889 |
| 15.339.2 Member Function Documentation | 1889 |
| 15.339.2.1 get_fb() | 1889 |
| 15.339.2.2 init_infos() | 1889 |
| 15.339.2.3 refresh() | 1890 |
| 15.339.2.4 screen_info() | 1890 |
| 15.339.2.5 view_info() | 1890 |
| 15.340 L4Re::Vfs::Be_file Class Reference | 1891 |
| 15.340.1 Detailed Description | 1894 |
| 15.340.2 Member Function Documentation | 1894 |
| 15.340.2.1 check_ready() | 1894 |
| 15.340.2.2 data_space() | 1894 |
| 15.340.2.3 fstat() | 1895 |
| 15.340.2.4 unlock_all_locks() | 1895 |
| 15.341 L4Re::Vfs::Be_file_system Class Reference | 1896 |
| 15.341.1 Detailed Description | 1897 |
| 15.341.2 Constructor & Destructor Documentation | 1898 |
| 15.341.2.1 Be_file_system() | 1898 |
| 15.341.2.2 ~Be_file_system() | 1898 |
| 15.341.3 Member Function Documentation | 1898 |
| 15.341.3.1 type() | 1898 |
| 15.342 L4Re::Vfs::Directory Class Reference | 1899 |
| 15.342.1 Detailed Description | 1900 |
| 15.342.2 Member Function Documentation | 1901 |
| 15.342.2.1 faccessat() | 1901 |
| 15.342.2.2 link() | 1901 |

| | |
|--|------|
| 15.342.2.3 mkdir() | 1902 |
| 15.342.2.4 rename() | 1902 |
| 15.342.2.5 rmdir() | 1903 |
| 15.342.2.6 symlink() | 1903 |
| 15.342.2.7 unlink() | 1904 |
| 15.343 L4Re::Vfs::File Class Reference | 1904 |
| 15.343.1 Detailed Description | 1907 |
| 15.344 L4Re::Vfs::File_system Class Reference | 1908 |
| 15.344.1 Detailed Description | 1909 |
| 15.344.2 Member Function Documentation | 1909 |
| 15.344.2.1 mount() | 1909 |
| 15.344.2.2 type() | 1910 |
| 15.345 L4Re::Vfs::Fs Class Reference | 1910 |
| 15.345.1 Detailed Description | 1913 |
| 15.345.2 Member Function Documentation | 1913 |
| 15.345.2.1 alloc_fd() | 1913 |
| 15.345.2.2 free_fd() | 1913 |
| 15.345.2.3 get_file() | 1914 |
| 15.345.2.4 mount() | 1914 |
| 15.345.2.5 set_fd() | 1915 |
| 15.346 L4Re::Vfs::Generic_file Class Reference | 1915 |
| 15.346.1 Detailed Description | 1917 |
| 15.346.2 Member Enumeration Documentation | 1918 |
| 15.346.2.1 Ready_type | 1918 |
| 15.346.3 Member Function Documentation | 1918 |
| 15.346.3.1 check_ready() | 1918 |
| 15.346.3.2 fchmod() | 1919 |
| 15.346.3.3 fstat() | 1920 |
| 15.346.3.4 get_status_flags() | 1921 |
| 15.346.3.5 set_status_flags() | 1922 |
| 15.346.3.6 unlock_all_locks() | 1923 |
| 15.347 L4Re::Vfs::Mman Class Reference | 1924 |
| 15.347.1 Detailed Description | 1925 |
| 15.348 L4Re::Vfs::Ops Class Reference | 1925 |
| 15.348.1 Detailed Description | 1928 |
| 15.349 L4Re::Vfs::Regular_file Class Reference | 1928 |
| 15.349.1 Detailed Description | 1930 |
| 15.349.2 Member Function Documentation | 1931 |
| 15.349.2.1 data_space() | 1931 |
| 15.349.2.2 fdatasync() | 1931 |
| 15.349.2.3 fsync() | 1932 |
| 15.349.2.4 ftruncate() | 1932 |

| | | |
|------------|--|------|
| 15.349.2.5 | get_lock() | 1933 |
| 15.349.2.6 | lseek() | 1934 |
| 15.349.2.7 | readv() | 1935 |
| 15.349.2.8 | set_lock() | 1936 |
| 15.349.2.9 | writev() | 1937 |
| 15.350 | L4Re::Vfs::Special_file Class Reference | 1938 |
| 15.350.1 | Detailed Description | 1940 |
| 15.350.2 | Member Function Documentation | 1940 |
| 15.350.2.1 | ioctl() | 1940 |
| 15.351 | L4Re::Video::Color_component Class Reference | 1941 |
| 15.351.1 | Detailed Description | 1942 |
| 15.351.2 | Constructor & Destructor Documentation | 1942 |
| 15.351.2.1 | Color_component() | 1942 |
| 15.351.3 | Member Function Documentation | 1942 |
| 15.351.3.1 | dump() | 1942 |
| 15.351.3.2 | get() | 1943 |
| 15.351.3.3 | operator==(()) | 1943 |
| 15.351.3.4 | set() | 1944 |
| 15.351.3.5 | shift() | 1944 |
| 15.351.3.6 | size() | 1945 |
| 15.352 | L4Re::Video::Goos Class Reference | 1946 |
| 15.352.1 | Detailed Description | 1949 |
| 15.352.2 | Member Enumeration Documentation | 1949 |
| 15.352.2.1 | Flags | 1949 |
| 15.352.3 | Member Function Documentation | 1950 |
| 15.352.3.1 | create_buffer() | 1950 |
| 15.352.3.2 | create_view() | 1950 |
| 15.352.3.3 | delete_buffer() | 1951 |
| 15.352.3.4 | delete_view() | 1952 |
| 15.352.3.5 | get_static_buffer() | 1953 |
| 15.352.3.6 | info() | 1954 |
| 15.352.3.7 | view() | 1955 |
| 15.353 | L4Re::Video::Goos::Info Struct Reference | 1956 |
| 15.353.1 | Detailed Description | 1958 |
| 15.354 | L4Re::Video::Pixel_info Class Reference | 1958 |
| 15.354.1 | Detailed Description | 1959 |
| 15.354.2 | Constructor & Destructor Documentation | 1960 |
| 15.354.2.1 | Pixel_info() [1/2] | 1960 |
| 15.354.2.2 | Pixel_info() [2/2] | 1961 |
| 15.354.3 | Member Function Documentation | 1962 |
| 15.354.3.1 | a() [1/2] | 1962 |
| 15.354.3.2 | a() [2/2] | 1962 |

| | |
|---|------|
| 15.354.3.3 b() [1/2] | 1963 |
| 15.354.3.4 b() [2/2] | 1963 |
| 15.354.3.5 bits_per_pixel() | 1964 |
| 15.354.3.6 bytes_per_pixel() [1/2] | 1964 |
| 15.354.3.7 bytes_per_pixel() [2/2] | 1965 |
| 15.354.3.8 dump() | 1965 |
| 15.354.3.9 g() [1/2] | 1966 |
| 15.354.3.10 g() [2/2] | 1966 |
| 15.354.3.11 has_alpha() | 1966 |
| 15.354.3.12 operator==() | 1967 |
| 15.354.3.13 padding() | 1967 |
| 15.354.3.14 r() [1/2] | 1968 |
| 15.354.3.15 r() [2/2] | 1968 |
| 15.355 L4Re::Video::View Class Reference | 1969 |
| 15.355.1 Detailed Description | 1970 |
| 15.355.2 Member Enumeration Documentation | 1970 |
| 15.355.2.1 Flags | 1970 |
| 15.355.2.2 V_flags | 1971 |
| 15.355.3 Member Function Documentation | 1971 |
| 15.355.3.1 info() | 1971 |
| 15.355.3.2 refresh() | 1972 |
| 15.355.3.3 set_info() | 1973 |
| 15.355.3.4 set_viewport() | 1974 |
| 15.355.3.5 stack() | 1975 |
| 15.356 L4Re::Video::View::Info Struct Reference | 1976 |
| 15.356.1 Detailed Description | 1977 |
| 15.357 l4re_aux_t Struct Reference | 1978 |
| 15.357.1 Detailed Description | 1978 |
| 15.358 l4re_ds_stats_t Struct Reference | 1979 |
| 15.358.1 Detailed Description | 1979 |
| 15.359 l4re_elf_aux_mword_t Struct Reference | 1979 |
| 15.359.1 Detailed Description | 1980 |
| 15.360 l4re_elf_aux_t Struct Reference | 1980 |
| 15.360.1 Detailed Description | 1980 |
| 15.361 l4re_elf_aux_vma_t Struct Reference | 1981 |
| 15.361.1 Detailed Description | 1981 |
| 15.362 l4re_env_cap_entry_t Struct Reference | 1981 |
| 15.362.1 Detailed Description | 1982 |
| 15.362.2 Constructor & Destructor Documentation | 1982 |
| 15.362.2.1 l4re_env_cap_entry_t() | 1982 |
| 15.362.3 Field Documentation | 1983 |
| 15.362.3.1 flags | 1983 |

| | |
|--|------|
| 15.363 l4re_env_t Struct Reference | 1983 |
| 15.363.1 Detailed Description | 1985 |
| 15.363.2 Field Documentation | 1985 |
| 15.363.2.1 caps | 1985 |
| 15.364 l4re_event_t Struct Reference | 1986 |
| 15.364.1 Detailed Description | 1986 |
| 15.365 l4re_video_color_component_t Struct Reference | 1987 |
| 15.365.1 Detailed Description | 1987 |
| 15.366 l4re_video_goos_info_t Struct Reference | 1987 |
| 15.366.1 Detailed Description | 1989 |
| 15.367 l4re_video_pixel_info_t Struct Reference | 1989 |
| 15.367.1 Detailed Description | 1990 |
| 15.368 l4re_video_view_info_t Struct Reference | 1990 |
| 15.368.1 Detailed Description | 1992 |
| 15.369 l4re_video_view_t Struct Reference | 1992 |
| 15.369.1 Detailed Description | 1992 |
| 15.370 l4shmc_ringbuf_head_t Struct Reference | 1993 |
| 15.370.1 Detailed Description | 1993 |
| 15.371 l4shmc_ringbuf_t Struct Reference | 1994 |
| 15.371.1 Detailed Description | 1995 |
| 15.372 l4util_l4mod_info Struct Reference | 1995 |
| 15.372.1 Detailed Description | 1996 |
| 15.372.2 Field Documentation | 1996 |
| 15.372.2.1 vbe_ctrl_info | 1996 |
| 15.373 l4util_l4mod_mod Struct Reference | 1997 |
| 15.373.1 Detailed Description | 1997 |
| 15.374 l4util_mb_addr_range_t Struct Reference | 1998 |
| 15.374.1 Detailed Description | 1998 |
| 15.375 l4util_mb_apm_t Struct Reference | 1999 |
| 15.375.1 Detailed Description | 1999 |
| 15.376 l4util_mb_drive_t Struct Reference | 1999 |
| 15.376.1 Detailed Description | 2000 |
| 15.377 l4util_mb_info_t Struct Reference | 2000 |
| 15.377.1 Detailed Description | 2001 |
| 15.378 l4util_mb_mod_t Struct Reference | 2002 |
| 15.378.1 Detailed Description | 2002 |
| 15.379 l4util_mb_vbe_ctrl_t Struct Reference | 2003 |
| 15.379.1 Detailed Description | 2003 |
| 15.380 l4util_mb_vbe_mode_t Struct Reference | 2003 |
| 15.380.1 Detailed Description | 2006 |
| 15.381 L4vbus::Device Class Reference | 2007 |
| 15.381.1 Detailed Description | 2009 |

| | |
|--|------|
| 15.381.2 Constructor & Destructor Documentation | 2010 |
| 15.381.2.1 Device() | 2010 |
| 15.381.3 Member Function Documentation | 2010 |
| 15.381.3.1 bus_cap() | 2010 |
| 15.381.3.2 dev_handle() | 2011 |
| 15.381.3.3 device() | 2012 |
| 15.381.3.4 device_by_hid() | 2012 |
| 15.381.3.5 get_resource() | 2013 |
| 15.381.3.6 is_compatible() | 2014 |
| 15.381.3.7 next_device() | 2015 |
| 15.381.3.8 operator!=(()) | 2016 |
| 15.381.3.9 operator==(()) | 2016 |
| 15.382 L4vbus::Gpio_module Class Reference | 2017 |
| 15.382.1 Detailed Description | 2020 |
| 15.382.2 Member Function Documentation | 2020 |
| 15.382.2.1 config_pad() | 2020 |
| 15.382.2.2 get() | 2021 |
| 15.382.2.3 pin() | 2021 |
| 15.382.2.4 set() | 2022 |
| 15.382.2.5 setup() | 2023 |
| 15.383 L4vbus::Gpio_module::Pin_slice Struct Reference | 2024 |
| 15.383.1 Detailed Description | 2024 |
| 15.384 L4vbus::Gpio_pin Class Reference | 2024 |
| 15.384.1 Detailed Description | 2028 |
| 15.384.2 Member Function Documentation | 2028 |
| 15.384.2.1 config_get() | 2028 |
| 15.384.2.2 config_pad() | 2029 |
| 15.384.2.3 config_pull() | 2029 |
| 15.384.2.4 get() | 2030 |
| 15.384.2.5 pin() | 2030 |
| 15.384.2.6 set() | 2031 |
| 15.384.2.7 setup() | 2031 |
| 15.384.2.8 to_irq() | 2032 |
| 15.385 L4vbus::Icu Class Reference | 2033 |
| 15.385.1 Detailed Description | 2036 |
| 15.385.2 Member Enumeration Documentation | 2036 |
| 15.385.2.1 Src_types | 2036 |
| 15.385.3 Member Function Documentation | 2036 |
| 15.385.3.1 vicu() | 2036 |
| 15.386 L4vbus::Pci_dev Class Reference | 2037 |
| 15.386.1 Detailed Description | 2040 |
| 15.386.2 Member Function Documentation | 2041 |

| | | |
|------------|--|------|
| 15.386.2.1 | cfg_read() | 2041 |
| 15.386.2.2 | cfg_write() | 2041 |
| 15.386.2.3 | irq_enable() | 2042 |
| 15.387 | L4vbus::Pci_host_bridge Class Reference | 2043 |
| 15.387.1 | Detailed Description | 2047 |
| 15.387.2 | Member Function Documentation | 2047 |
| 15.387.2.1 | cfg_read() | 2047 |
| 15.387.2.2 | cfg_write() | 2048 |
| 15.387.2.3 | irq_enable() | 2048 |
| 15.388 | L4vbus::Pm< DEC > Class Template Reference | 2049 |
| 15.388.1 | Detailed Description | 2051 |
| 15.388.2 | Member Function Documentation | 2051 |
| 15.388.2.1 | pm_resume() | 2051 |
| 15.388.2.2 | pm_suspend() | 2052 |
| 15.389 | L4vbus::Vbus Class Reference | 2052 |
| 15.389.1 | Detailed Description | 2059 |
| 15.389.2 | Member Function Documentation | 2059 |
| 15.389.2.1 | assign_dma_domain() [1/2] | 2059 |
| 15.389.2.2 | assign_dma_domain() [2/2] | 2060 |
| 15.389.2.3 | release_ioport() | 2061 |
| 15.389.2.4 | request_ioport() | 2062 |
| 15.389.2.5 | root() | 2062 |
| 15.390 | l4vbus_device_t Struct Reference | 2063 |
| 15.390.1 | Detailed Description | 2064 |
| 15.391 | l4vbus_resource_t Struct Reference | 2064 |
| 15.391.1 | Detailed Description | 2065 |
| 15.392 | L4vcpu::State Class Reference | 2065 |
| 15.392.1 | Detailed Description | 2066 |
| 15.392.2 | Constructor & Destructor Documentation | 2066 |
| 15.392.2.1 | State() | 2066 |
| 15.392.3 | Member Function Documentation | 2066 |
| 15.392.3.1 | add() | 2066 |
| 15.392.3.2 | clear() | 2067 |
| 15.392.3.3 | set() | 2067 |
| 15.393 | L4vcpu::Vcpu Class Reference | 2067 |
| 15.393.1 | Detailed Description | 2071 |
| 15.393.2 | Member Function Documentation | 2071 |
| 15.393.2.1 | cast() [1/2] | 2071 |
| 15.393.2.2 | cast() [2/2] | 2071 |
| 15.393.2.3 | entry_ip() | 2072 |
| 15.393.2.4 | entry_sp() | 2073 |
| 15.393.2.5 | ext_alloc() | 2073 |

| | |
|---|------|
| 15.393.2.6 i() [1/2] | 2074 |
| 15.393.2.7 i() [2/2] | 2074 |
| 15.393.2.8 irq_disable_save() | 2074 |
| 15.393.2.9 irq_enable() | 2075 |
| 15.393.2.10 irq_restore() | 2075 |
| 15.393.2.11 is_irq_entry() | 2076 |
| 15.393.2.12 is_page_fault_entry() | 2076 |
| 15.393.2.13 r() [1/2] | 2077 |
| 15.393.2.14 r() [2/2] | 2077 |
| 15.393.2.15 saved_state() [1/2] | 2077 |
| 15.393.2.16 saved_state() [2/2] | 2078 |
| 15.393.2.17 state() [1/2] | 2078 |
| 15.393.2.18 state() [2/2] | 2078 |
| 15.393.2.19 task() | 2079 |
| 15.393.2.20 wait_for_event() | 2079 |
| 15.394 L4virtio::Device Class Reference | 2080 |
| 15.394.1 Detailed Description | 2084 |
| 15.394.2 Member Function Documentation | 2085 |
| 15.394.2.1 config_queue() | 2085 |
| 15.394.2.2 device_config() | 2086 |
| 15.394.2.3 device_notification_irq() | 2086 |
| 15.394.2.4 register_ds() | 2087 |
| 15.394.2.5 set_status() | 2088 |
| 15.395 L4virtio::Driver::Block_device Class Reference | 2090 |
| 15.395.1 Detailed Description | 2093 |
| 15.395.2 Member Function Documentation | 2093 |
| 15.395.2.1 add_block() | 2093 |
| 15.395.2.2 process_request() | 2094 |
| 15.395.2.3 process_used_queue() | 2094 |
| 15.395.2.4 send_request() | 2095 |
| 15.395.2.5 setup_device() | 2096 |
| 15.395.2.6 start_request() | 2097 |
| 15.396 L4virtio::Driver::Block_device::Handle Class Reference | 2098 |
| 15.396.1 Detailed Description | 2098 |
| 15.397 L4virtio::Driver::Device Class Reference | 2098 |
| 15.397.1 Detailed Description | 2101 |
| 15.397.2 Member Function Documentation | 2101 |
| 15.397.2.1 bind_notification_irq() | 2101 |
| 15.397.2.2 config_queue() | 2102 |
| 15.397.2.3 driver_acknowledge() | 2103 |
| 15.397.2.4 driver_connect() | 2103 |
| 15.397.2.5 feature_negotiated() | 2105 |

| | |
|---|------|
| 15.397.2.6 max_queue_size() | 2106 |
| 15.397.2.7 register_ds() | 2106 |
| 15.397.2.8 send() | 2107 |
| 15.397.2.9 send_and_wait() | 2108 |
| 15.397.2.10 wait() | 2109 |
| 15.397.2.11 wait_for_next_used() | 2110 |
| 15.398 L4virtio::Driver::Virtio_net_device Class Reference | 2111 |
| 15.398.1 Detailed Description | 2115 |
| 15.398.2 Member Function Documentation | 2115 |
| 15.398.2.1 bind_rx_notification_irq() | 2115 |
| 15.398.2.2 finish_rx() | 2115 |
| 15.398.2.3 rx_pkt() | 2116 |
| 15.398.2.4 rx_queue_size() | 2116 |
| 15.398.2.5 setup_device() | 2117 |
| 15.398.2.6 tx() | 2118 |
| 15.398.2.7 tx_queue_size() | 2118 |
| 15.398.2.8 wait_rx() | 2119 |
| 15.399 L4virtio::Driver::Virtio_net_device::Packet Struct Reference | 2120 |
| 15.399.1 Detailed Description | 2120 |
| 15.400 L4virtio::Driver::Virtqueue Class Reference | 2121 |
| 15.400.1 Detailed Description | 2124 |
| 15.400.2 Member Function Documentation | 2125 |
| 15.400.2.1 alloc_descriptor() | 2125 |
| 15.400.2.2 desc() | 2125 |
| 15.400.2.3 enqueue_descriptor() | 2126 |
| 15.400.2.4 find_next_used() | 2126 |
| 15.400.2.5 free_descriptor() | 2127 |
| 15.400.2.6 init_queue() [1/2] | 2127 |
| 15.400.2.7 init_queue() [2/2] | 2128 |
| 15.400.2.8 initialize_rings() | 2128 |
| 15.401 L4virtio::Ptr< T > Class Template Reference | 2130 |
| 15.401.1 Detailed Description | 2131 |
| 15.401.2 Member Enumeration Documentation | 2131 |
| 15.401.2.1 Invalid_type | 2131 |
| 15.401.3 Member Function Documentation | 2132 |
| 15.401.3.1 get() | 2132 |
| 15.401.3.2 is_valid() | 2132 |
| 15.402 L4virtio::Svr::Bad_descriptor Struct Reference | 2133 |
| 15.402.1 Detailed Description | 2134 |
| 15.402.2 Member Enumeration Documentation | 2134 |
| 15.402.2.1 Error | 2134 |
| 15.402.3 Constructor & Destructor Documentation | 2134 |

| | |
|--|------|
| 15.402.3.1 Bad_descriptor() | 2134 |
| 15.402.4 Member Function Documentation | 2135 |
| 15.402.4.1 message() | 2135 |
| 15.403 L4virtio::Svr::Block_dev_base< Ds_data > Class Template Reference | 2135 |
| 15.403.1 Detailed Description | 2139 |
| 15.403.2 Constructor & Destructor Documentation | 2139 |
| 15.403.2.1 Block_dev_base() | 2139 |
| 15.403.3 Member Function Documentation | 2140 |
| 15.403.3.1 finalize_request() | 2140 |
| 15.403.3.2 get_writeback() | 2141 |
| 15.403.3.3 set_blk_size() | 2141 |
| 15.403.3.4 set_config_wce() | 2142 |
| 15.403.3.5 set_discard() | 2142 |
| 15.403.3.6 set_size_max() | 2142 |
| 15.403.3.7 set_topology() | 2143 |
| 15.403.3.8 set_write_zeroes() | 2143 |
| 15.404 L4virtio::Svr::Block_request< Ds_data > Class Template Reference | 2144 |
| 15.404.1 Detailed Description | 2144 |
| 15.404.2 Member Function Documentation | 2145 |
| 15.404.2.1 data_size() | 2145 |
| 15.404.2.2 next_block() | 2145 |
| 15.405 L4virtio::Svr::Console::Control_message Struct Reference | 2146 |
| 15.405.1 Detailed Description | 2147 |
| 15.405.2 Member Enumeration Documentation | 2147 |
| 15.405.2.1 Events | 2147 |
| 15.406 L4virtio::Svr::Console::Control_request Struct Reference | 2148 |
| 15.406.1 Detailed Description | 2149 |
| 15.407 L4virtio::Svr::Console::Device Class Reference | 2149 |
| 15.407.1 Detailed Description | 2154 |
| 15.407.2 Constructor & Destructor Documentation | 2154 |
| 15.407.2.1 Device() [1/3] | 2154 |
| 15.407.2.2 Device() [2/3] | 2155 |
| 15.407.2.3 Device() [3/3] | 2156 |
| 15.407.3 Member Function Documentation | 2156 |
| 15.407.3.1 notify_queue() | 2156 |
| 15.407.3.2 port() | 2157 |
| 15.407.3.3 port_read() | 2158 |
| 15.407.3.4 port_write() | 2159 |
| 15.407.3.5 process_device_ready() | 2161 |
| 15.407.3.6 process_port_open() | 2162 |
| 15.407.3.7 process_port_ready() | 2162 |
| 15.408 L4virtio::Svr::Console::Device_port Struct Reference | 2163 |

| | |
|--|------|
| 15.408.1 Detailed Description | 2167 |
| 15.409 L4virtio::Svr::Console::Features Struct Reference | 2167 |
| 15.409.1 Detailed Description | 2170 |
| 15.409.2 Member Typedef Documentation | 2171 |
| 15.409.2.1 console_multiport_bfm_t | 2171 |
| 15.409.2.2 console_size_bfm_t | 2171 |
| 15.409.2.3 emerg_write_bfm_t | 2171 |
| 15.410 L4virtio::Svr::Console::Port Struct Reference | 2172 |
| 15.410.1 Detailed Description | 2175 |
| 15.410.2 Member Enumeration Documentation | 2175 |
| 15.410.2.1 Port_status | 2175 |
| 15.410.3 Field Documentation | 2176 |
| 15.410.3.1 state_transitions | 2176 |
| 15.411 L4virtio::Svr::Console::Port::Transition Struct Reference | 2176 |
| 15.411.1 Detailed Description | 2177 |
| 15.412 L4virtio::Svr::Console::Virtio_con Class Reference | 2177 |
| 15.412.1 Detailed Description | 2181 |
| 15.412.2 Constructor & Destructor Documentation | 2182 |
| 15.412.2.1 Virtio_con() | 2182 |
| 15.412.3 Member Function Documentation | 2183 |
| 15.412.3.1 handle_control_message() | 2183 |
| 15.412.3.2 notify_queue() | 2184 |
| 15.412.3.3 port() | 2184 |
| 15.412.3.4 port_add() | 2185 |
| 15.412.3.5 port_name() | 2186 |
| 15.412.3.6 port_open() | 2187 |
| 15.412.3.7 port_remove() | 2188 |
| 15.412.3.8 process_device_ready() | 2189 |
| 15.412.3.9 process_port_open() | 2190 |
| 15.412.3.10 process_port_ready() | 2191 |
| 15.412.3.11 reset_device() | 2192 |
| 15.412.3.12 send_control_message() | 2192 |
| 15.413 L4virtio::Svr::Data_buffer Struct Reference | 2194 |
| 15.413.1 Detailed Description | 2195 |
| 15.413.2 Constructor & Destructor Documentation | 2195 |
| 15.413.2.1 Data_buffer() | 2195 |
| 15.413.3 Member Function Documentation | 2196 |
| 15.413.3.1 copy_to() | 2196 |
| 15.413.3.2 done() | 2197 |
| 15.413.3.3 set() | 2197 |
| 15.413.3.4 skip() | 2197 |
| 15.414 L4virtio::Svr::Dev_config Class Reference | 2198 |

| | |
|--|------|
| 15.414.1 Detailed Description | 2200 |
| 15.414.2 Constructor & Destructor Documentation | 2200 |
| 15.414.2.1 Dev_config() [1/2] | 2200 |
| 15.414.2.2 Dev_config() [2/2] | 2201 |
| 15.414.3 Member Function Documentation | 2202 |
| 15.414.3.1 add_irq_status() | 2202 |
| 15.414.3.2 change_queue_config() | 2203 |
| 15.414.3.3 ds() | 2203 |
| 15.414.3.4 get_cmd() | 2204 |
| 15.414.3.5 guest_features() | 2204 |
| 15.414.3.6 hdr() | 2205 |
| 15.414.3.7 negotiated_features() | 2205 |
| 15.414.3.8 qconfig() | 2206 |
| 15.414.3.9 reset_cmd() | 2207 |
| 15.414.3.10 reset_queue() | 2207 |
| 15.414.3.11 set_device_needs_reset() | 2208 |
| 15.414.3.12 set_device_notify_index() | 2209 |
| 15.414.3.13 set_status() | 2209 |
| 15.414.3.14 status() | 2210 |
| 15.415 L4virtio::Svr::Dev_features Struct Reference | 2210 |
| 15.415.1 Detailed Description | 2213 |
| 15.416 L4virtio::Svr::Dev_status Struct Reference | 2213 |
| 15.416.1 Detailed Description | 2216 |
| 15.416.2 Member Function Documentation | 2216 |
| 15.416.2.1 running() | 2216 |
| 15.417 L4virtio::Svr::Device_t< DATA > Class Template Reference | 2216 |
| 15.417.1 Detailed Description | 2219 |
| 15.417.2 Member Function Documentation | 2220 |
| 15.417.2.1 add_trusted_dataspaces() | 2220 |
| 15.417.2.2 device_error() | 2220 |
| 15.417.2.3 device_notify_irq() | 2220 |
| 15.417.2.4 handle_mem_cmd_write() | 2221 |
| 15.417.2.5 init_mem_info() | 2221 |
| 15.417.2.6 register_driver_irq() | 2221 |
| 15.417.2.7 reset_queue_config() | 2222 |
| 15.417.2.8 setup_queue() | 2222 |
| 15.418 L4virtio::Svr::Driver_mem_list_t< DATA > Class Template Reference | 2223 |
| 15.418.1 Detailed Description | 2225 |
| 15.418.2 Member Function Documentation | 2225 |
| 15.418.2.1 add() | 2225 |
| 15.418.2.2 find() | 2226 |
| 15.418.2.3 full() | 2226 |

| | | |
|-------------|---|------|
| 15.418.2.4 | init() | 2227 |
| 15.418.2.5 | load_desc() [1/3] | 2227 |
| 15.418.2.6 | load_desc() [2/3] | 2228 |
| 15.418.2.7 | load_desc() [3/3] | 2228 |
| 15.418.2.8 | remove() | 2229 |
| 15.419 | L4virtio::Svr::Driver_mem_region_t< DATA > Class Template Reference | 2229 |
| 15.419.1 | Detailed Description | 2232 |
| 15.419.2 | Constructor & Destructor Documentation | 2232 |
| 15.419.2.1 | Driver_mem_region_t() | 2232 |
| 15.419.3 | Member Function Documentation | 2232 |
| 15.419.3.1 | contains() | 2232 |
| 15.419.3.2 | drv_base() | 2233 |
| 15.419.3.3 | ds() | 2233 |
| 15.419.3.4 | ds_offset() | 2233 |
| 15.419.3.5 | empty() | 2234 |
| 15.419.3.6 | flags() | 2234 |
| 15.419.3.7 | is_writable() | 2234 |
| 15.419.3.8 | local() | 2234 |
| 15.419.3.9 | local_base() | 2235 |
| 15.419.3.10 | size() | 2235 |
| 15.420 | L4virtio::Svr::Request_processor Class Reference | 2236 |
| 15.420.1 | Detailed Description | 2237 |
| 15.420.2 | Member Function Documentation | 2237 |
| 15.420.2.1 | current_flags() | 2237 |
| 15.420.2.2 | has_more() | 2238 |
| 15.420.2.3 | next() | 2239 |
| 15.420.2.4 | start() [1/2] | 2240 |
| 15.420.2.5 | start() [2/2] | 2242 |
| 15.421 | L4virtio::Svr::Scmi::Base_attr_t Struct Reference | 2243 |
| 15.421.1 | Detailed Description | 2244 |
| 15.422 | L4virtio::Svr::Scmi::Base_proto Class Reference | 2244 |
| 15.422.1 | Detailed Description | 2246 |
| 15.423 | L4virtio::Svr::Scmi::Perf_proto Class Reference | 2246 |
| 15.423.1 | Detailed Description | 2248 |
| 15.424 | L4virtio::Svr::Scmi::Performance_attr_t Struct Reference | 2249 |
| 15.424.1 | Detailed Description | 2249 |
| 15.425 | L4virtio::Svr::Scmi::Performance_describe_level_t Struct Reference | 2250 |
| 15.425.1 | Detailed Description | 2250 |
| 15.426 | L4virtio::Svr::Scmi::Performance_describe_levels_n_t Struct Reference | 2250 |
| 15.426.1 | Detailed Description | 2251 |
| 15.427 | L4virtio::Svr::Scmi::Performance_domain_attr_t Struct Reference | 2252 |
| 15.427.1 | Detailed Description | 2253 |

| | |
|---|------|
| 15.428 L4virtio::Svr::Scmi::Proto< OBSERV > Struct Template Reference | 2254 |
| 15.428.1 Detailed Description | 2255 |
| 15.429 L4virtio::Svr::Scmi::Scmi_dev Class Reference | 2255 |
| 15.429.1 Detailed Description | 2258 |
| 15.430 L4virtio::Svr::Scmi::Scmi_hdr_t Struct Reference | 2259 |
| 15.430.1 Detailed Description | 2260 |
| 15.431 L4virtio::Svr::Virtio_gpio< Request_handler, Epiface > Class Template Reference | 2261 |
| 15.431.1 Detailed Description | 2265 |
| 15.432 L4virtio::Svr::Virtio_gpio< Request_handler, Epiface >::Host_irq Struct Reference | 2265 |
| 15.432.1 Detailed Description | 2268 |
| 15.433 L4virtio::Svr::Virtio_gpio< Request_handler, Epiface >::Irq_handler Struct Reference | 2269 |
| 15.433.1 Detailed Description | 2269 |
| 15.434 L4virtio::Svr::Virtio_gpio< Request_handler, Epiface >::Request_processor Struct Reference | 2270 |
| 15.434.1 Detailed Description | 2272 |
| 15.434.2 Member Function Documentation | 2272 |
| 15.434.2.1 get_request() | 2272 |
| 15.435 L4virtio::Svr::Virtio_i2c< Request_handler, Epiface > Class Template Reference | 2273 |
| 15.435.1 Detailed Description | 2277 |
| 15.436 L4virtio::Svr::Virtio_i2c< Request_handler, Epiface >::Host_irq Class Reference | 2277 |
| 15.436.1 Detailed Description | 2280 |
| 15.437 L4virtio::Svr::Virtio_i2c< Request_handler, Epiface >::Request_processor Class Reference | 2281 |
| 15.437.1 Detailed Description | 2283 |
| 15.437.2 Member Function Documentation | 2283 |
| 15.437.2.1 get_request() | 2283 |
| 15.438 L4virtio::Svr::Virtio_rng< Rnd_state, Epiface > Class Template Reference | 2284 |
| 15.438.1 Detailed Description | 2288 |
| 15.439 L4virtio::Svr::Virtio_rng< Rnd_state, Epiface >::Host_irq Class Reference | 2288 |
| 15.439.1 Detailed Description | 2291 |
| 15.440 L4virtio::Svr::Virtio_rng< Rnd_state, Epiface >::Request_processor Class Reference | 2292 |
| 15.440.1 Detailed Description | 2294 |
| 15.441 L4virtio::Svr::Virtio_spi< Spi_request_handler, Epiface > Class Template Reference | 2294 |
| 15.441.1 Detailed Description | 2299 |
| 15.442 L4virtio::Svr::Virtio_spi< Spi_request_handler, Epiface >::Host_irq Class Reference | 2299 |
| 15.442.1 Detailed Description | 2302 |
| 15.443 L4virtio::Svr::Virtio_spi< Spi_request_handler, Epiface >::Request_processor Class Reference | 2303 |
| 15.443.1 Detailed Description | 2305 |
| 15.443.2 Member Function Documentation | 2305 |
| 15.443.2.1 get_request() | 2305 |
| 15.444 L4virtio::Svr::Virtqueue Class Reference | 2306 |
| 15.444.1 Detailed Description | 2309 |
| 15.444.2 Member Function Documentation | 2310 |
| 15.444.2.1 consumed() [1/2] | 2310 |

| | |
|--|------|
| 15.444.2.2 consumed() [2/2] | 2311 |
| 15.444.2.3 desc() | 2311 |
| 15.444.2.4 desc_avail() | 2312 |
| 15.444.2.5 disable_notify() | 2313 |
| 15.444.2.6 enable_notify() | 2314 |
| 15.444.2.7 finish() [1/2] | 2314 |
| 15.444.2.8 finish() [2/2] | 2316 |
| 15.444.2.9 next_avail() | 2317 |
| 15.444.2.10 rewind_avail() | 2318 |
| 15.445 L4virtio::Svr::Virtqueue::Head_desc Class Reference | 2318 |
| 15.445.1 Detailed Description | 2319 |
| 15.445.2 Member Function Documentation | 2319 |
| 15.445.2.1 desc() | 2319 |
| 15.445.2.2 operator bool() | 2320 |
| 15.445.2.3 valid() | 2320 |
| 15.446 L4virtio::Virtqueue Class Reference | 2321 |
| 15.446.1 Detailed Description | 2324 |
| 15.446.2 Member Function Documentation | 2324 |
| 15.446.2.1 avail_align() | 2324 |
| 15.446.2.2 avail_size() | 2325 |
| 15.446.2.3 desc_align() | 2326 |
| 15.446.2.4 desc_size() | 2326 |
| 15.446.2.5 disable() | 2327 |
| 15.446.2.6 dump() | 2328 |
| 15.446.2.7 get_avail_idx() | 2329 |
| 15.446.2.8 get_tail_avail_idx() | 2329 |
| 15.446.2.9 no_notify_guest() | 2329 |
| 15.446.2.10 no_notify_host() [1/2] | 2330 |
| 15.446.2.11 no_notify_host() [2/2] | 2330 |
| 15.446.2.12 num() | 2331 |
| 15.446.2.13 ready() | 2332 |
| 15.446.2.14 setup() | 2332 |
| 15.446.2.15 setup_simple() | 2334 |
| 15.446.2.16 total_size() [1/2] | 2335 |
| 15.446.2.17 total_size() [2/2] | 2335 |
| 15.446.2.18 used_align() | 2336 |
| 15.446.2.19 used_size() | 2336 |
| 15.447 L4virtio::Virtqueue::Avail Class Reference | 2338 |
| 15.447.1 Detailed Description | 2339 |
| 15.448 L4virtio::Virtqueue::Avail::Flags Struct Reference | 2339 |
| 15.448.1 Detailed Description | 2340 |
| 15.448.2 Member Typedef Documentation | 2340 |

| | |
|--|------|
| 15.448.2.1 no_irq_bfm_t | 2340 |
| 15.449 L4virtio::Virtqueue::Desc Class Reference | 2340 |
| 15.449.1 Detailed Description | 2342 |
| 15.450 L4virtio::Virtqueue::Desc::Flags Struct Reference | 2342 |
| 15.450.1 Detailed Description | 2343 |
| 15.450.2 Member Typedef Documentation | 2343 |
| 15.450.2.1 indirect_bfm_t | 2343 |
| 15.450.2.2 next_bfm_t | 2344 |
| 15.450.2.3 write_bfm_t | 2344 |
| 15.451 L4virtio::Virtqueue::Used Class Reference | 2344 |
| 15.451.1 Detailed Description | 2345 |
| 15.452 L4virtio::Virtqueue::Used::Flags Struct Reference | 2345 |
| 15.452.1 Detailed Description | 2346 |
| 15.452.2 Member Typedef Documentation | 2346 |
| 15.452.2.1 no_notify_bfm_t | 2346 |
| 15.453 L4virtio::Virtqueue::Used_elem Struct Reference | 2347 |
| 15.453.1 Detailed Description | 2347 |
| 15.453.2 Constructor & Destructor Documentation | 2348 |
| 15.453.2.1 Used_elem() | 2348 |
| 15.454 l4virtio_block_config_t Struct Reference | 2348 |
| 15.454.1 Detailed Description | 2349 |
| 15.455 l4virtio_block_discard_t Struct Reference | 2349 |
| 15.455.1 Detailed Description | 2349 |
| 15.456 l4virtio_block_header_t Struct Reference | 2350 |
| 15.456.1 Detailed Description | 2350 |
| 15.457 l4virtio_config_hdr_t Struct Reference | 2351 |
| 15.457.1 Detailed Description | 2351 |
| 15.458 l4virtio_config_queue_t Struct Reference | 2352 |
| 15.458.1 Detailed Description | 2353 |
| 15.459 l4virtio_input_absinfo_t Struct Reference | 2353 |
| 15.459.1 Detailed Description | 2353 |
| 15.460 l4virtio_input_config_t Struct Reference | 2354 |
| 15.460.1 Detailed Description | 2354 |
| 15.461 l4virtio_input_devids_t Struct Reference | 2354 |
| 15.461.1 Detailed Description | 2355 |
| 15.462 l4virtio_input_event_t Struct Reference | 2355 |
| 15.462.1 Detailed Description | 2355 |
| 15.463 l4virtio_net_config_t Struct Reference | 2356 |
| 15.463.1 Detailed Description | 2356 |
| 15.464 l4virtio_net_header_t Struct Reference | 2356 |
| 15.464.1 Detailed Description | 2357 |
| 15.465 L4virtio_port Class Reference | 2357 |

| | |
|---|------|
| 15.465.1 Detailed Description | 2362 |
| 15.465.2 Member Function Documentation | 2362 |
| 15.465.2.1 drop_requests() | 2362 |
| 15.466 Mac_addr Class Reference | 2363 |
| 15.466.1 Detailed Description | 2363 |
| 15.467 Mac_table< Size > Class Template Reference | 2363 |
| 15.467.1 Detailed Description | 2364 |
| 15.467.2 Member Function Documentation | 2365 |
| 15.467.2.1 flush() | 2365 |
| 15.467.2.2 learn() | 2365 |
| 15.467.2.3 lookup() | 2366 |
| 15.468 Net_transfer Class Reference | 2366 |
| 15.468.1 Detailed Description | 2368 |
| 15.468.2 Member Function Documentation | 2368 |
| 15.468.2.1 cur_buf() | 2368 |
| 15.468.2.2 done() | 2368 |
| 15.469 Rm Class Reference | 2369 |
| 15.469.1 Detailed Description | 2373 |
| 15.469.2 Member Enumeration Documentation | 2373 |
| 15.469.2.1 Detach_flags | 2373 |
| 15.469.2.2 Detach_result | 2374 |
| 15.469.2.3 Region_flag_shifts | 2374 |
| 15.469.3 Member Function Documentation | 2374 |
| 15.469.3.1 attach() [1/2] | 2374 |
| 15.469.3.2 attach() [2/2] | 2375 |
| 15.469.3.3 detach() [1/3] | 2377 |
| 15.469.3.4 detach() [2/3] | 2377 |
| 15.469.3.5 detach() [3/3] | 2378 |
| 15.469.3.6 find() | 2379 |
| 15.469.3.7 free_area() | 2380 |
| 15.469.3.8 get_areas() | 2380 |
| 15.469.3.9 get_info() | 2381 |
| 15.469.3.10 get_regions() | 2381 |
| 15.469.3.11 reserve_area() [1/2] | 2382 |
| 15.469.3.12 reserve_area() [2/2] | 2382 |
| 15.470 Rm::Area Struct Reference | 2383 |
| 15.470.1 Detailed Description | 2384 |
| 15.471 Rm::F Struct Reference | 2384 |
| 15.471.1 Detailed Description | 2385 |
| 15.471.2 Member Enumeration Documentation | 2385 |
| 15.471.2.1 Attach_flags | 2385 |
| 15.471.2.2 Region_flags | 2385 |

| | |
|---|------|
| 15.472 Rm::Region Struct Reference | 2386 |
| 15.472.1 Detailed Description | 2386 |
| 15.473 Rm::Unique_region< T > Class Template Reference | 2387 |
| 15.473.1 Detailed Description | 2389 |
| 15.473.2 Constructor & Destructor Documentation | 2389 |
| 15.473.2.1 Unique_region() [1/3] | 2389 |
| 15.473.2.2 Unique_region() [2/3] | 2390 |
| 15.473.2.3 Unique_region() [3/3] | 2390 |
| 15.473.2.4 ~Unique_region() | 2390 |
| 15.473.3 Member Function Documentation | 2390 |
| 15.473.3.1 get() | 2390 |
| 15.473.3.2 is_valid() | 2391 |
| 15.473.3.3 operator=() | 2391 |
| 15.473.3.4 release() | 2391 |
| 15.473.3.5 reset() | 2392 |
| 15.474 Switch_factory Class Reference | 2392 |
| 15.474.1 Detailed Description | 2395 |
| 15.474.2 Member Function Documentation | 2396 |
| 15.474.2.1 op_create() | 2396 |
| 15.475 utrace::Ring_buffer< SEQUENCE_TYPE > Class Template Reference | 2396 |
| 15.475.1 Detailed Description | 2398 |
| 15.475.2 Member Function Documentation | 2398 |
| 15.475.2.1 check_items() | 2398 |
| 15.475.2.2 check_mask() | 2399 |
| 15.476 utrace::Ring_buffer_consumer< SEQUENCE_TYPE, ITEM_TYPE, GENERATION_PTR > Class Template Reference | 2400 |
| 15.476.1 Detailed Description | 2403 |
| 15.476.2 Constructor & Destructor Documentation | 2404 |
| 15.476.2.1 Ring_buffer_consumer() | 2404 |
| 15.476.3 Member Function Documentation | 2404 |
| 15.476.3.1 dequeue() | 2404 |
| 15.476.3.2 peek() | 2405 |
| 15.477 utrace::Ring_buffer_consumer_raw< SEQUENCE_TYPE, ITEM_TYPE, GENERATION_PTR > Class Template Reference | 2406 |
| 15.477.1 Detailed Description | 2409 |
| 15.477.2 Member Enumeration Documentation | 2409 |
| 15.477.2.1 Drop_policy | 2409 |
| 15.477.3 Constructor & Destructor Documentation | 2410 |
| 15.477.3.1 Ring_buffer_consumer_raw() | 2410 |
| 15.477.4 Member Function Documentation | 2410 |
| 15.477.4.1 peek() | 2410 |
| 15.478 utrace::Ring_buffer_producer< SEQUENCE_TYPE, ITEM_TYPE, GENERATION_PTR > Class Template Reference | 2411 |

| | |
|--|------|
| 15.478.1 Detailed Description | 2413 |
| 15.478.2 Constructor & Destructor Documentation | 2413 |
| 15.478.2.1 Ring_buffer_producer() | 2413 |
| 15.478.3 Member Function Documentation | 2414 |
| 15.478.3.1 enqueue() | 2414 |
| 15.479 utrace::Ring_slot< ITEM_TYPE, GENERATION_PTR > Class Template Reference | 2415 |
| 15.479.1 Detailed Description | 2416 |
| 15.479.2 Member Function Documentation | 2416 |
| 15.479.2.1 size() | 2416 |
| 15.480 utrace::Ring_status< SEQUENCE_TYPE > Class Template Reference | 2417 |
| 15.480.1 Detailed Description | 2418 |
| 15.480.2 Member Function Documentation | 2419 |
| 15.480.2.1 alignment() | 2419 |
| 15.480.2.2 version() | 2419 |
| 15.481 utrace::Tracebuffer Class Reference | 2420 |
| 15.481.1 Detailed Description | 2421 |
| 15.481.2 Member Function Documentation | 2421 |
| 15.481.2.1 dequeue() | 2421 |
| 15.481.2.2 index() | 2422 |
| 15.481.2.3 indexes() | 2422 |
| 15.481.2.4 validate() | 2422 |
| 15.482 utrace::Tracebuffer::Index_desc Struct Reference | 2423 |
| 15.482.1 Detailed Description | 2423 |
| 15.483 Virtio_net Class Reference | 2423 |
| 15.483.1 Detailed Description | 2428 |
| 15.483.2 Member Function Documentation | 2428 |
| 15.483.2.1 notify_queue() | 2428 |
| 15.484 Virtio_net_request Class Reference | 2428 |
| 15.484.1 Detailed Description | 2429 |
| 15.484.2 Member Function Documentation | 2429 |
| 15.484.2.1 drop_requests() | 2429 |
| 15.484.2.2 get_request() | 2430 |
| 15.485 Virtio_switch Class Reference | 2431 |
| 15.485.1 Detailed Description | 2432 |
| 15.485.2 Constructor & Destructor Documentation | 2433 |
| 15.485.2.1 Virtio_switch() | 2433 |
| 15.485.3 Member Function Documentation | 2433 |
| 15.485.3.1 add_monitor_port() | 2433 |
| 15.485.3.2 add_port() | 2433 |
| 15.485.3.3 check_ports() | 2434 |
| 15.485.3.4 handle_l4virtio_port_tx() | 2434 |
| 15.485.3.5 port_available() | 2435 |

| | |
|---|-------------|
| 15.486 Virtio_vlan_mangle Class Reference | 2436 |
| 15.486.1 Detailed Description | 2436 |
| 15.486.2 Constructor & Destructor Documentation | 2437 |
| 15.486.2.1 Virtio_vlan_mangle() | 2437 |
| 15.486.3 Member Function Documentation | 2437 |
| 15.486.3.1 add() | 2437 |
| 15.486.3.2 copy_pkt() | 2438 |
| 15.486.3.3 remove() | 2438 |
| 15.486.3.4 rewrite_hdr() | 2439 |
| 16 File Documentation | 2441 |
| 16.1 asm_access.h | 2441 |
| 16.2 asm_access.h | 2442 |
| 16.3 asm_access.h | 2443 |
| 16.4 asm_access.h | 2444 |
| 16.5 asm_access.h | 2444 |
| 16.6 asm_access.h | 2444 |
| 16.7 asm_access.h | 2445 |
| 16.8 asm_access.h | 2446 |
| 16.9 asm_access_gen.h | 2446 |
| 16.10 hw_mmio_register_block | 2447 |
| 16.11 hw_register_block | 2448 |
| 16.12 io_regblock.h | 2451 |
| 16.13 io_regblock_port.h | 2453 |
| 16.14 poll_timeout_counter.h | 2454 |
| 16.15 device.h | 2455 |
| 16.16 uart_16550.h | 2456 |
| 16.17 uart_16550_dw.h | 2457 |
| 16.18 uart_apb.h | 2458 |
| 16.19 uart_base.h | 2458 |
| 16.20 uart_bcm2835.h | 2460 |
| 16.21 uart_cadence.h | 2460 |
| 16.22 uart_dcc-v6.h | 2461 |
| 16.23 uart_dm.h | 2461 |
| 16.24 uart_dummy.h | 2462 |
| 16.25 uart_geni.h | 2462 |
| 16.26 uart_imx.h | 2463 |
| 16.27 uart_leon3.h | 2464 |
| 16.28 uart_linflex.h | 2465 |
| 16.29 uart_lpuart.h | 2465 |
| 16.30 uart_mvebu.h | 2466 |
| 16.31 uart_of.h | 2466 |

| | | |
|-----------|---|------|
| 16.32 | uart_omap35x.h | 2467 |
| 16.33 | uart_pl011.h | 2467 |
| 16.34 | uart_s3c2410.h | 2468 |
| 16.35 | uart_sa1000.h | 2469 |
| 16.36 | uart_sbi.h | 2469 |
| 16.37 | uart_sh.h | 2470 |
| 16.38 | uart_sifive.h | 2470 |
| 16.39 | uart_tegra-tcu.h | 2471 |
| 16.40 | tutorial.lua | 2471 |
| 16.41 | cmd_control | 2474 |
| 16.42 | pkg/uvmm/configs/vmm.lua File Reference | 2475 |
| 16.42.1 | Function Documentation | 2475 |
| 16.42.1.1 | new_sched() | 2475 |
| 16.42.1.2 | set_sched() | 2475 |
| 16.42.1.3 | start_io() | 2476 |
| 16.42.1.4 | start_virtio_switch() | 2476 |
| 16.42.1.5 | start_virtio_switch_tbl() | 2477 |
| 16.42.1.6 | start_vm() | 2478 |
| 16.43 | vmm.lua | 2478 |
| 16.44 | debug.h | 2483 |
| 16.45 | l4/re/c/debug.h File Reference | 2484 |
| 16.45.1 | Detailed Description | 2485 |
| 16.46 | debug.h | 2485 |
| 16.47 | filter.cc | 2485 |
| 16.48 | filter.h | 2486 |
| 16.49 | mac_addr.h | 2486 |
| 16.50 | mac_table.h | 2487 |
| 16.51 | main.cc | 2489 |
| 16.52 | Makefile | 2497 |
| 16.53 | Makefile | 2497 |
| 16.54 | Makefile | 2497 |
| 16.55 | Makefile | 2498 |
| 16.56 | options.cc | 2498 |
| 16.57 | options.h | 2500 |
| 16.58 | port.h | 2501 |
| 16.59 | port_ixl.h | 2503 |
| 16.60 | port_l4virtio.h | 2505 |
| 16.61 | request.h | 2509 |
| 16.62 | request_ixl.h | 2510 |
| 16.63 | request_l4virtio.h | 2512 |
| 16.64 | stats.h | 2515 |
| 16.65 | switch.cc | 2516 |

| | |
|--|------|
| 16.66 switch.h | 2519 |
| 16.67 virtio_net.h | 2520 |
| 16.68 virtio_net.h | 2523 |
| 16.69 virtio_net_buffer.h | 2524 |
| 16.70 vlan.h | 2524 |
| 16.71 amd64/l4/sys/segment.h File Reference | 2525 |
| 16.71.1 Detailed Description | 2527 |
| 16.71.2 Enumeration Type Documentation | 2527 |
| 16.71.2.1 L4_sys_segment | 2527 |
| 16.71.2.2 L4_task_ldt_x86_consts | 2527 |
| 16.71.3 Function Documentation | 2527 |
| 16.71.3.1 fiasco_amd64_segment_info() | 2527 |
| 16.71.3.2 fiasco_amd64_set_fs() | 2528 |
| 16.71.3.3 fiasco_amd64_set_segment_base() | 2529 |
| 16.72 segment.h | 2530 |
| 16.73 x86/l4/sys/segment.h File Reference | 2532 |
| 16.73.1 Detailed Description | 2533 |
| 16.73.2 Enumeration Type Documentation | 2533 |
| 16.73.2.1 L4_task_ldt_x86_consts | 2533 |
| 16.74 segment.h | 2533 |
| 16.75 amd64/l4/util/perform.h File Reference | 2534 |
| 16.75.1 Detailed Description | 2535 |
| 16.76 perform.h | 2535 |
| 16.77 x86/l4/util/perform.h File Reference | 2540 |
| 16.77.1 Detailed Description | 2541 |
| 16.78 perform.h | 2541 |
| 16.79 amd64/l4/util/port_io.h File Reference | 2546 |
| 16.79.1 Detailed Description | 2548 |
| 16.80 port_io.h | 2548 |
| 16.81 x86/l4/util/port_io.h File Reference | 2550 |
| 16.81.1 Detailed Description | 2552 |
| 16.82 port_io.h | 2552 |
| 16.83 amd64/l4/util/rdtsc.h File Reference | 2554 |
| 16.83.1 Detailed Description | 2556 |
| 16.84 rdtsc.h | 2556 |
| 16.85 x86/l4/util/rdtsc.h File Reference | 2558 |
| 16.85.1 Detailed Description | 2560 |
| 16.86 rdtsc.h | 2560 |
| 16.87 amd64/l4/util/spin.h File Reference | 2563 |
| 16.87.1 Detailed Description | 2563 |
| 16.88 spin.h | 2564 |
| 16.89 x86/l4/util/spin.h File Reference | 2564 |

| | |
|--|------|
| 16.89.1 Detailed Description | 2565 |
| 16.90 spin.h | 2565 |
| 16.91 __kip-arch.h | 2565 |
| 16.92 __kip-arch.h | 2565 |
| 16.93 __kip-arch.h | 2566 |
| 16.94 __kip-arch.h | 2566 |
| 16.95 __kip-arch.h | 2567 |
| 16.96 amd64/l4/sys/__vcpu-arch.h File Reference | 2567 |
| 16.96.1 Detailed Description | 2568 |
| 16.96.2 Enumeration Type Documentation | 2568 |
| 16.96.2.1 anonymous enum | 2568 |
| 16.97 __vcpu-arch.h | 2568 |
| 16.98 arm/l4/sys/__vcpu-arch.h File Reference | 2569 |
| 16.98.1 Detailed Description | 2571 |
| 16.98.2 Enumeration Type Documentation | 2571 |
| 16.98.2.1 anonymous enum | 2571 |
| 16.98.2.2 L4_vcpu_e_field_ids | 2571 |
| 16.99 __vcpu-arch.h | 2572 |
| 16.100 arm64/l4/sys/__vcpu-arch.h File Reference | 2573 |
| 16.100.1 Detailed Description | 2574 |
| 16.100.2 Enumeration Type Documentation | 2574 |
| 16.100.2.1 anonymous enum | 2574 |
| 16.100.2.2 L4_vcpu_e_field_ids | 2574 |
| 16.101 __vcpu-arch.h | 2575 |
| 16.102 riscv/l4/sys/__vcpu-arch.h File Reference | 2576 |
| 16.102.1 Detailed Description | 2577 |
| 16.102.2 Enumeration Type Documentation | 2577 |
| 16.102.2.1 anonymous enum | 2577 |
| 16.103 __vcpu-arch.h | 2577 |
| 16.104 x86/l4/sys/__vcpu-arch.h File Reference | 2578 |
| 16.104.1 Detailed Description | 2579 |
| 16.104.2 Enumeration Type Documentation | 2579 |
| 16.104.2.1 anonymous enum | 2579 |
| 16.105 __vcpu-arch.h | 2579 |
| 16.106 ktrace_events.h | 2580 |
| 16.107 ktrace_events.h | 2583 |
| 16.108 ktrace_events.h | 2586 |
| 16.109 ktrace_events.h | 2589 |
| 16.110 ktrace_events.h | 2595 |
| 16.111 amd64/l4/sys/linkage.h File Reference | 2598 |
| 16.111.1 Detailed Description | 2598 |
| 16.112 linkage.h | 2599 |

| | |
|--|------|
| 16.113 arm/l4/sys/linkage.h File Reference | 2599 |
| 16.113.1 Detailed Description | 2599 |
| 16.114 linkage.h | 2599 |
| 16.115 linkage.h | 2600 |
| 16.116 riscv/l4/sys/linkage.h File Reference | 2600 |
| 16.116.1 Detailed Description | 2600 |
| 16.117 linkage.h | 2600 |
| 16.118 x86/l4/sys/linkage.h File Reference | 2601 |
| 16.118.1 Detailed Description | 2601 |
| 16.119 linkage.h | 2601 |
| 16.120 arm/l4/sys/mem_op.h File Reference | 2602 |
| 16.120.1 Detailed Description | 2602 |
| 16.121 mem_op.h | 2603 |
| 16.122 arm/l4/sys/syscall_defs.h File Reference | 2604 |
| 16.122.1 Detailed Description | 2604 |
| 16.123 syscall_defs.h | 2604 |
| 16.124 vm.h | 2605 |
| 16.125 arm/l4/sys/vm.h File Reference | 2605 |
| 16.125.1 Detailed Description | 2605 |
| 16.126 vm.h | 2606 |
| 16.127 vm.h | 2607 |
| 16.128 vm.h | 2607 |
| 16.129 vm.h | 2608 |
| 16.130 amd64/l4/util/arch/l4_macros.h File Reference | 2608 |
| 16.130.1 Detailed Description | 2608 |
| 16.131 l4_macros.h | 2608 |
| 16.132 arm/l4/util/arch/l4_macros.h File Reference | 2609 |
| 16.132.1 Detailed Description | 2609 |
| 16.133 l4_macros.h | 2609 |
| 16.134 l4/util/l4_macros.h File Reference | 2609 |
| 16.134.1 Detailed Description | 2610 |
| 16.135 l4_macros.h | 2610 |
| 16.136 x86/l4/util/arch/l4_macros.h File Reference | 2610 |
| 16.136.1 Detailed Description | 2610 |
| 16.137 l4_macros.h | 2611 |
| 16.138 thread.h | 2611 |
| 16.139 thread.h | 2611 |
| 16.140 arm/l4/sys/arch/thread.h File Reference | 2611 |
| 16.140.1 Detailed Description | 2612 |
| 16.141 thread.h | 2612 |
| 16.142 thread.h | 2612 |
| 16.143 arm64/l4/sys/arch/thread.h File Reference | 2612 |

| | |
|---|------|
| 16.143.1 Detailed Description | 2613 |
| 16.144 thread.h | 2613 |
| 16.145 l4/sys/thread.h File Reference | 2614 |
| 16.145.1 Detailed Description | 2616 |
| 16.146 thread.h | 2616 |
| 16.147 l4/util/thread.h File Reference | 2623 |
| 16.147.1 Detailed Description | 2624 |
| 16.147.2 Macro Definition Documentation | 2624 |
| 16.147.2.1 __L4UTIL_THREAD_FUNC | 2624 |
| 16.148 thread.h | 2625 |
| 16.149 thread.h | 2625 |
| 16.150 thread.h | 2625 |
| 16.151 thread.h | 2625 |
| 16.152 amd64/l4/util/bitops_arch.h File Reference | 2626 |
| 16.152.1 Detailed Description | 2626 |
| 16.153 bitops_arch.h | 2626 |
| 16.154 arm/l4/util/bitops_arch.h File Reference | 2629 |
| 16.154.1 Detailed Description | 2630 |
| 16.155 bitops_arch.h | 2630 |
| 16.156 x86/l4/util/bitops_arch.h File Reference | 2630 |
| 16.156.1 Detailed Description | 2630 |
| 16.157 bitops_arch.h | 2630 |
| 16.158 amd64/l4/util/cpu.h File Reference | 2633 |
| 16.158.1 Detailed Description | 2634 |
| 16.159 cpu.h | 2635 |
| 16.160 arm/l4/util/cpu.h File Reference | 2636 |
| 16.160.1 Detailed Description | 2636 |
| 16.161 cpu.h | 2636 |
| 16.162 x86/l4/util/cpu.h File Reference | 2636 |
| 16.162.1 Detailed Description | 2637 |
| 16.163 cpu.h | 2637 |
| 16.164 amd64/l4/util/mbi_argv.h File Reference | 2638 |
| 16.164.1 Detailed Description | 2639 |
| 16.165 mbi_argv.h | 2639 |
| 16.166 arm/l4/util/mbi_argv.h File Reference | 2640 |
| 16.166.1 Detailed Description | 2640 |
| 16.167 mbi_argv.h | 2641 |
| 16.168 x86/l4/util/mbi_argv.h File Reference | 2641 |
| 16.168.1 Detailed Description | 2642 |
| 16.169 mbi_argv.h | 2642 |
| 16.170 contrib/libio-io/l4/io/io.h File Reference | 2643 |
| 16.170.1 Function Documentation | 2644 |

| | |
|--|------|
| 16.170.1.1 l4io_get_root_device() | 2644 |
| 16.170.1.2 l4io_iterate_devices() | 2644 |
| 16.170.1.3 l4io_request_all_ioports() | 2645 |
| 16.170.1.4 l4io_request_icu() | 2645 |
| 16.171 io.h | 2645 |
| 16.172 l4/cxx/alloc.h File Reference | 2646 |
| 16.172.1 Detailed Description | 2647 |
| 16.173 alloc.h | 2647 |
| 16.174 arith | 2647 |
| 16.175 l4/cxx/avl_map File Reference | 2648 |
| 16.175.1 Detailed Description | 2650 |
| 16.176 avl_map | 2650 |
| 16.177 l4/cxx/avl_set File Reference | 2651 |
| 16.177.1 Detailed Description | 2652 |
| 16.178 avl_set | 2653 |
| 16.179 l4/cxx/avl_tree File Reference | 2656 |
| 16.179.1 Detailed Description | 2658 |
| 16.180 avl_tree | 2658 |
| 16.181 l4/cxx/basic_ostream File Reference | 2662 |
| 16.181.1 Detailed Description | 2663 |
| 16.182 basic_ostream | 2663 |
| 16.183 l4/cxx/basic_vector.h File Reference | 2666 |
| 16.183.1 Detailed Description | 2666 |
| 16.184 basic_vector.h | 2667 |
| 16.185 bitfield | 2667 |
| 16.186 bitmap | 2669 |
| 16.187 l4/cxx/bits/bst.h File Reference | 2672 |
| 16.187.1 Detailed Description | 2674 |
| 16.188 bst.h | 2674 |
| 16.189 l4/cxx/bits/bst_base.h File Reference | 2676 |
| 16.189.1 Detailed Description | 2678 |
| 16.190 bst_base.h | 2678 |
| 16.191 l4/cxx/bits/bst_iter.h File Reference | 2679 |
| 16.191.1 Detailed Description | 2680 |
| 16.192 bst_iter.h | 2681 |
| 16.193 list_basics.h | 2682 |
| 16.194 l4/cxx/bits/smart_ptr_list.h File Reference | 2684 |
| 16.194.1 Detailed Description | 2685 |
| 16.195 smart_ptr_list.h | 2685 |
| 16.196 type_traits.h | 2687 |
| 16.197 dlist | 2690 |
| 16.198 elide_dtor | 2695 |

| | |
|---|------|
| 16.199 l4/cxx/exceptions File Reference | 2696 |
| 16.199.1 Detailed Description | 2698 |
| 16.200 exceptions | 2698 |
| 16.201 hlist | 2700 |
| 16.202 l4/cxx/iostream File Reference | 2702 |
| 16.202.1 Detailed Description | 2703 |
| 16.203 iostream | 2703 |
| 16.204 l4/cxx/ipc_helper File Reference | 2703 |
| 16.204.1 Detailed Description | 2704 |
| 16.205 ipc_helper | 2705 |
| 16.206 l4/cxx/ipc_stream File Reference | 2705 |
| 16.206.1 Detailed Description | 2708 |
| 16.206.2 Function Documentation | 2708 |
| 16.206.2.1 operator<<() [1/4] | 2708 |
| 16.206.2.2 operator<<() [2/4] | 2709 |
| 16.206.2.3 operator<<() [3/4] | 2709 |
| 16.206.2.4 operator<<() [4/4] | 2710 |
| 16.206.2.5 operator>>() [1/6] | 2711 |
| 16.206.2.6 operator>>() [2/6] | 2711 |
| 16.206.2.7 operator>>() [3/6] | 2712 |
| 16.206.2.8 operator>>() [4/6] | 2713 |
| 16.206.2.9 operator>>() [5/6] | 2714 |
| 16.206.2.10 operator>>() [6/6] | 2714 |
| 16.207 ipc_stream | 2715 |
| 16.208 ipc_timeout_queue | 2723 |
| 16.209 l4/cxx/l4iostream File Reference | 2725 |
| 16.209.1 Detailed Description | 2725 |
| 16.210 l4iostream | 2726 |
| 16.211 l4/cxx/l4types.h File Reference | 2726 |
| 16.211.1 Detailed Description | 2727 |
| 16.212 l4types.h | 2727 |
| 16.213 list | 2727 |
| 16.214 list_alloc | 2731 |
| 16.215 l4/cxx/lock_guard.h File Reference | 2737 |
| 16.215.1 Detailed Description | 2737 |
| 16.216 lock_guard.h | 2737 |
| 16.217 l4/cxx/main_thread File Reference | 2738 |
| 16.217.1 Detailed Description | 2739 |
| 16.218 main_thread | 2739 |
| 16.219 minmax | 2740 |
| 16.220 numeric | 2741 |
| 16.221 observer | 2741 |

| | |
|--|------|
| 16.222 l4/cxx/pair File Reference | 2742 |
| 16.222.1 Detailed Description | 2743 |
| 16.223 pair | 2743 |
| 16.224 ref_ptr | 2744 |
| 16.225 l4/cxx/ref_ptr_list File Reference | 2747 |
| 16.225.1 Detailed Description | 2748 |
| 16.226 ref_ptr_list | 2748 |
| 16.227 result | 2749 |
| 16.228 slab_alloc | 2751 |
| 16.229 slist | 2754 |
| 16.230 static_container | 2757 |
| 16.231 static_vector | 2758 |
| 16.232 std_alloc | 2758 |
| 16.233 std_ops | 2759 |
| 16.234 string | 2759 |
| 16.235 l4/cxx/string.h File Reference | 2762 |
| 16.235.1 Detailed Description | 2763 |
| 16.236 string.h | 2763 |
| 16.237 type_list | 2764 |
| 16.238 type_traits | 2764 |
| 16.239 unique_ptr | 2769 |
| 16.240 l4/cxx/unique_ptr_list File Reference | 2770 |
| 16.240.1 Detailed Description | 2771 |
| 16.241 unique_ptr_list | 2772 |
| 16.242 utils | 2772 |
| 16.243 weak_ref | 2772 |
| 16.244 amd64/l4/util/irq.h File Reference | 2774 |
| 16.244.1 Detailed Description | 2774 |
| 16.245 irq.h | 2775 |
| 16.246 arm/l4/util/irq.h File Reference | 2775 |
| 16.246.1 Detailed Description | 2776 |
| 16.247 irq.h | 2776 |
| 16.248 l4/irq/irq.h File Reference | 2777 |
| 16.248.1 Detailed Description | 2778 |
| 16.249 irq.h | 2778 |
| 16.250 l4/sys/irq.h File Reference | 2779 |
| 16.250.1 Detailed Description | 2781 |
| 16.251 irq.h | 2781 |
| 16.252 x86/l4/util/irq.h File Reference | 2783 |
| 16.252.1 Detailed Description | 2784 |
| 16.253 irq.h | 2784 |
| 16.254 backend | 2785 |

| | |
|---|------|
| 16.255 default_ops_impl.h | 2788 |
| 16.256 fd_store.h | 2789 |
| 16.257 fd_store_impl.h | 2790 |
| 16.258 ns_fs.h | 2790 |
| 16.259 ns_fs_impl.h | 2791 |
| 16.260 ro_file.h | 2796 |
| 16.261 ro_file_impl.h | 2796 |
| 16.262 vcon_stream.h | 2798 |
| 16.263 vcon_stream_impl.h | 2798 |
| 16.264 vfs_impl.h | 2801 |
| 16.265 vfs.h | 2813 |
| 16.266 virtio-net | 2822 |
| 16.267 l4virtio | 2825 |
| 16.268 l4virtio | 2827 |
| 16.269 l4virtio | 2829 |
| 16.270 virtio | 2840 |
| 16.271 virtio-block | 2844 |
| 16.272 virtio-block | 2848 |
| 16.273 virtio-console | 2854 |
| 16.274 virtio-console-device | 2860 |
| 16.275 virtio-gpio-device | 2864 |
| 16.276 virtio-i2c-device | 2869 |
| 16.277 virtio-rng-device | 2874 |
| 16.278 virtio-scmi-device | 2876 |
| 16.279 virtio-spi-device | 2885 |
| 16.280 virtio.h | 2890 |
| 16.281 virtio_block.h | 2893 |
| 16.282 virtio_input.h | 2894 |
| 16.283 virtqueue | 2894 |
| 16.284 l4/libedid/edid.h File Reference | 2899 |
| 16.285 edid.h | 2900 |
| 16.286 l4/re/c/dataspace.h File Reference | 2900 |
| 16.286.1 Detailed Description | 2902 |
| 16.287 dataspace.h | 2902 |
| 16.288 l4/re/c/dma_space.h File Reference | 2903 |
| 16.288.1 Detailed Description | 2905 |
| 16.288.2 Enumeration Type Documentation | 2905 |
| 16.288.2.1 l4re_dma_space_direction | 2905 |
| 16.288.2.2 l4re_dma_space_space_attrbs | 2905 |
| 16.289 dma_space.h | 2906 |
| 16.290 l4/re/c/event.h File Reference | 2906 |
| 16.290.1 Detailed Description | 2908 |

| | |
|--|------|
| 16.291 event.h | 2908 |
| 16.292 l4/re/event.h File Reference | 2909 |
| 16.292.1 Detailed Description | 2909 |
| 16.293 event.h | 2910 |
| 16.294 event_buffer.h | 2910 |
| 16.295 l4/re/c/inhibitor.h File Reference | 2911 |
| 16.295.1 Detailed Description | 2912 |
| 16.295.2 Function Documentation | 2912 |
| 16.295.2.1 l4re_inhibitor_acquire() | 2912 |
| 16.295.2.2 l4re_inhibitor_next_lock_info() | 2913 |
| 16.295.2.3 l4re_inhibitor_release() | 2913 |
| 16.296 inhibitor.h | 2914 |
| 16.297 l4/re/c/log.h File Reference | 2914 |
| 16.297.1 Detailed Description | 2915 |
| 16.298 log.h | 2915 |
| 16.299 l4/re/c/mem_alloc.h File Reference | 2915 |
| 16.299.1 Detailed Description | 2916 |
| 16.300 mem_alloc.h | 2916 |
| 16.301 l4/re/c/namespace.h File Reference | 2917 |
| 16.301.1 Detailed Description | 2918 |
| 16.302 namespace.h | 2918 |
| 16.303 l4/re/c/parent.h File Reference | 2919 |
| 16.303.1 Detailed Description | 2919 |
| 16.303.2 Function Documentation | 2920 |
| 16.303.2.1 l4re_parent_signal() | 2920 |
| 16.304 parent.h | 2920 |
| 16.305 l4/re/c/rm.h File Reference | 2921 |
| 16.305.1 Detailed Description | 2922 |
| 16.306 rm.h | 2922 |
| 16.307 l4/re/c/util/cap_alloc.h File Reference | 2925 |
| 16.307.1 Detailed Description | 2926 |
| 16.308 cap_alloc.h | 2926 |
| 16.309 l4/re/c/util/kumem_alloc.h File Reference | 2927 |
| 16.309.1 Detailed Description | 2927 |
| 16.309.2 Function Documentation | 2928 |
| 16.309.2.1 l4re_util_kumem_alloc() | 2928 |
| 16.310 kumem_alloc.h | 2928 |
| 16.311 l4/re/c/util/video/goos_fb.h File Reference | 2929 |
| 16.311.1 Detailed Description | 2929 |
| 16.312 goos_fb.h | 2929 |
| 16.313 l4/re/c/video/colors.h File Reference | 2930 |
| 16.313.1 Detailed Description | 2931 |

| | |
|---|------|
| 16.314 colors.h | 2932 |
| 16.315 l4/re/c/video/goos.h File Reference | 2932 |
| 16.315.1 Detailed Description | 2934 |
| 16.316 goos.h | 2934 |
| 16.317 l4/re/c/video/view.h File Reference | 2935 |
| 16.317.1 Detailed Description | 2936 |
| 16.318 view.h | 2937 |
| 16.319 console | 2938 |
| 16.320 l4/re/dataspace File Reference | 2938 |
| 16.320.1 Detailed Description | 2939 |
| 16.321 dataspace | 2939 |
| 16.322 l4/re/dataspace-sys.h File Reference | 2941 |
| 16.322.1 Detailed Description | 2941 |
| 16.323 dataspace-sys.h | 2942 |
| 16.324 dbg_events | 2942 |
| 16.325 l4/re/dma_space File Reference | 2942 |
| 16.326 dma_space | 2944 |
| 16.327 l4/re/elf_aux.h File Reference | 2945 |
| 16.327.1 Detailed Description | 2946 |
| 16.328 elf_aux.h | 2946 |
| 16.329 l4/re/env File Reference | 2947 |
| 16.329.1 Detailed Description | 2948 |
| 16.330 env | 2948 |
| 16.331 l4/re/env.h File Reference | 2949 |
| 16.331.1 Detailed Description | 2951 |
| 16.331.2 Typedef Documentation | 2951 |
| 16.331.2.1 l4re_env_t | 2951 |
| 16.332 env.h | 2951 |
| 16.333 l4/re/error_helper File Reference | 2953 |
| 16.333.1 Detailed Description | 2954 |
| 16.334 error_helper | 2955 |
| 16.335 event-sys.h | 2956 |
| 16.336 event_enums.h | 2956 |
| 16.337 l4/re/impl/dataspace_impl.h File Reference | 2963 |
| 16.337.1 Detailed Description | 2963 |
| 16.338 dataspace_impl.h | 2964 |
| 16.339 l4/re/impl/mem_alloc_impl.h File Reference | 2965 |
| 16.339.1 Detailed Description | 2966 |
| 16.340 mem_alloc_impl.h | 2966 |
| 16.341 l4/re/impl/namespace_impl.h File Reference | 2966 |
| 16.341.1 Detailed Description | 2967 |
| 16.342 namespace_impl.h | 2967 |

| | |
|---|------|
| 16.343 I4/re/impl/rm_impl.h File Reference | 2969 |
| 16.343.1 Detailed Description | 2969 |
| 16.344 rm_impl.h | 2970 |
| 16.345 inhibitor | 2971 |
| 16.346 inhibitor-sys.h | 2971 |
| 16.347 itas | 2972 |
| 16.348 I4/re/I4aux.h File Reference | 2973 |
| 16.348.1 Detailed Description | 2973 |
| 16.349 I4aux.h | 2974 |
| 16.350 I4/re/log File Reference | 2974 |
| 16.350.1 Detailed Description | 2975 |
| 16.351 log | 2975 |
| 16.352 I4/re/log-sys.h File Reference | 2976 |
| 16.352.1 Detailed Description | 2976 |
| 16.353 log-sys.h | 2976 |
| 16.354 I4/re/mem_alloc File Reference | 2976 |
| 16.354.1 Detailed Description | 2978 |
| 16.355 mem_alloc | 2978 |
| 16.356 I4/re/mem_alloc-sys.h File Reference | 2979 |
| 16.356.1 Detailed Description | 2979 |
| 16.357 mem_alloc-sys.h | 2979 |
| 16.358 I4/re/mmio_space File Reference | 2980 |
| 16.358.1 Detailed Description | 2980 |
| 16.359 mmio_space | 2981 |
| 16.360 I4/re/namespace File Reference | 2981 |
| 16.360.1 Detailed Description | 2983 |
| 16.361 namespace | 2983 |
| 16.362 I4/re/namespace-sys.h File Reference | 2984 |
| 16.362.1 Detailed Description | 2984 |
| 16.363 namespace-sys.h | 2985 |
| 16.364 I4/re/parent File Reference | 2985 |
| 16.364.1 Detailed Description | 2986 |
| 16.365 parent | 2986 |
| 16.366 I4/re/parent-sys.h File Reference | 2987 |
| 16.366.1 Detailed Description | 2987 |
| 16.367 parent-sys.h | 2987 |
| 16.368 I4/re/protocols.h File Reference | 2987 |
| 16.368.1 Detailed Description | 2988 |
| 16.369 protocols.h | 2988 |
| 16.370 I4/re/random File Reference | 2988 |
| 16.370.1 Detailed Description | 2989 |
| 16.371 random | 2990 |

| | |
|---|------|
| 16.372 remote_access | 2990 |
| 16.373 l4/re/rm File Reference | 2991 |
| 16.373.1 Detailed Description | 2992 |
| 16.374 rm | 2992 |
| 16.375 l4/re/rm-sys.h File Reference | 2996 |
| 16.375.1 Detailed Description | 2996 |
| 16.376 rm-sys.h | 2997 |
| 16.377 l4/re/util/bitmap_cap_alloc File Reference | 2997 |
| 16.377.1 Detailed Description | 2998 |
| 16.378 bitmap_cap_alloc | 2998 |
| 16.379 br_manager | 2999 |
| 16.380 l4/re/util/cap File Reference | 3002 |
| 16.380.1 Detailed Description | 3003 |
| 16.381 cap | 3003 |
| 16.382 l4/re/cap_alloc File Reference | 3004 |
| 16.382.1 Detailed Description | 3005 |
| 16.383 cap_alloc | 3005 |
| 16.384 l4/re/util/cap_alloc File Reference | 3007 |
| 16.384.1 Detailed Description | 3009 |
| 16.385 cap_alloc | 3009 |
| 16.386 l4/re/util/cap_alloc_impl.h File Reference | 3010 |
| 16.386.1 Detailed Description | 3012 |
| 16.387 cap_alloc_impl.h | 3012 |
| 16.388 l4/re/util/counting_cap_alloc File Reference | 3013 |
| 16.388.1 Detailed Description | 3014 |
| 16.389 counting_cap_alloc | 3014 |
| 16.390 dataspace_svr | 3018 |
| 16.391 l4/re/debug File Reference | 3021 |
| 16.391.1 Detailed Description | 3022 |
| 16.392 debug | 3022 |
| 16.393 debug | 3022 |
| 16.394 env_ns | 3024 |
| 16.395 event | 3025 |
| 16.396 l4/re/util/event File Reference | 3028 |
| 16.397 event | 3028 |
| 16.398 event_buffer | 3030 |
| 16.399 event_svr | 3031 |
| 16.400 icu_svr | 3032 |
| 16.401 l4/re/util/item_alloc File Reference | 3035 |
| 16.401.1 Detailed Description | 3036 |
| 16.402 item_alloc | 3036 |
| 16.403 l4/re/util/kumem_alloc File Reference | 3037 |

| | |
|---|------|
| 16.403.1 Detailed Description | 3038 |
| 16.404 kumem_alloc | 3038 |
| 16.405 name_space_svr | 3038 |
| 16.406 object_registry | 3044 |
| 16.407 poll_timeout_kipclock | 3046 |
| 16.408 l4/re/util/region_mapping File Reference | 3047 |
| 16.408.1 Detailed Description | 3048 |
| 16.409 region_mapping | 3048 |
| 16.410 region_mapping_svr | 3053 |
| 16.411 reply_cap_hooks | 3056 |
| 16.412 l4/re/shared_cap File Reference | 3057 |
| 16.412.1 Detailed Description | 3059 |
| 16.413 shared_cap | 3059 |
| 16.414 l4/re/util/shared_cap File Reference | 3061 |
| 16.414.1 Detailed Description | 3063 |
| 16.415 shared_cap | 3063 |
| 16.416 l4/re/unique_cap File Reference | 3065 |
| 16.416.1 Detailed Description | 3066 |
| 16.417 unique_cap | 3067 |
| 16.418 l4/re/util/unique_cap File Reference | 3067 |
| 16.418.1 Detailed Description | 3068 |
| 16.419 unique_cap | 3069 |
| 16.420 vcon_svr | 3069 |
| 16.421 goos_fb | 3070 |
| 16.422 goos_svr | 3072 |
| 16.423 colors | 3074 |
| 16.424 goos | 3075 |
| 16.425 l4/re/video/goos-sys.h File Reference | 3078 |
| 16.425.1 Detailed Description | 3079 |
| 16.426 goos-sys.h | 3079 |
| 16.427 view | 3080 |
| 16.428 l4/shmc/ringbuf.h File Reference | 3080 |
| 16.428.1 Function Documentation | 3081 |
| 16.428.1.1 l4shmc_rb_attach_receiver() | 3081 |
| 16.428.1.2 l4shmc_rb_attach_sender() | 3082 |
| 16.428.1.3 l4shmc_rb_deinit_buffer() | 3082 |
| 16.428.1.4 l4shmc_rb_init_buffer() | 3083 |
| 16.428.1.5 l4shmc_rb_init_receiver() | 3083 |
| 16.428.1.6 l4shmc_rb_receiver_copy_out() | 3084 |
| 16.428.1.7 l4shmc_rb_receiver_notify_done() | 3084 |
| 16.428.1.8 l4shmc_rb_receiver_read_next_size() | 3085 |
| 16.428.1.9 l4shmc_rb_receiver_wait_for_data() | 3085 |

| | |
|---|------|
| 16.428.1.10 l4shmc_rb_sender_alloc_packet() | 3086 |
| 16.428.1.11 l4shmc_rb_sender_commit_packet() | 3086 |
| 16.428.1.12 l4shmc_rb_sender_next_copy_in() | 3086 |
| 16.428.1.13 l4shmc_rb_sender_put_data() | 3087 |
| 16.429 ringbuf.h | 3087 |
| 16.430 l4/shmc/shmc.h File Reference | 3089 |
| 16.430.1 Detailed Description | 3092 |
| 16.431 shmc.h | 3092 |
| 16.432 l4/sigma0/sigma0.h File Reference | 3094 |
| 16.432.1 Detailed Description | 3096 |
| 16.433 sigma0.h | 3096 |
| 16.434 __kernel_object_impl.h | 3098 |
| 16.435 l4/sys/__ktrace-impl.h File Reference | 3098 |
| 16.435.1 Detailed Description | 3100 |
| 16.436 __ktrace-impl.h | 3100 |
| 16.437 __l4_fpage.h | 3101 |
| 16.438 __platform_control-arm.h | 3105 |
| 16.439 __task-arm.h | 3106 |
| 16.440 __timeout.h | 3106 |
| 16.441 l4/sys/__typeinfo.h File Reference | 3109 |
| 16.441.1 Detailed Description | 3111 |
| 16.442 __typeinfo.h | 3111 |
| 16.443 __vcpu-arm.h | 3121 |
| 16.444 __vm-svm.h | 3122 |
| 16.445 __vm-vmx.h | 3124 |
| 16.446 l4/sys/arm_smccc File Reference | 3130 |
| 16.446.1 Detailed Description | 3131 |
| 16.447 arm_smccc | 3132 |
| 16.448 l4/sys/arm_smccc.h File Reference | 3132 |
| 16.448.1 Detailed Description | 3133 |
| 16.448.2 Function Documentation | 3133 |
| 16.448.2.1 l4_arm_smccc_call() | 3133 |
| 16.449 arm_smccc.h | 3134 |
| 16.450 amd64/l4/sys/arch/cache.h File Reference | 3135 |
| 16.450.1 Detailed Description | 3136 |
| 16.451 cache.h | 3136 |
| 16.452 arm/l4/sys/arch/cache.h File Reference | 3137 |
| 16.452.1 Detailed Description | 3137 |
| 16.453 cache.h | 3138 |
| 16.454 arm64/l4/sys/arch/cache.h File Reference | 3139 |
| 16.454.1 Detailed Description | 3140 |
| 16.455 cache.h | 3140 |

| | |
|--|------|
| 16.456 l4/sys/cache.h File Reference | 3141 |
| 16.456.1 Detailed Description | 3142 |
| 16.457 cache.h | 3142 |
| 16.458 riscv/l4/sys/arch/cache.h File Reference | 3143 |
| 16.458.1 Detailed Description | 3144 |
| 16.459 cache.h | 3144 |
| 16.460 x86/l4/sys/arch/cache.h File Reference | 3145 |
| 16.460.1 Detailed Description | 3145 |
| 16.461 cache.h | 3145 |
| 16.462 l4/sys/capability File Reference | 3146 |
| 16.462.1 Detailed Description | 3147 |
| 16.462.2 Macro Definition Documentation | 3147 |
| 16.462.2.1 L4_DISABLE_COPY | 3147 |
| 16.463 capability | 3148 |
| 16.464 l4/sys/compiler.h File Reference | 3149 |
| 16.464.1 Detailed Description | 3151 |
| 16.465 compiler.h | 3151 |
| 16.466 amd64/l4/sys/arch/consts.h File Reference | 3153 |
| 16.466.1 Detailed Description | 3154 |
| 16.467 consts.h | 3154 |
| 16.468 arm/l4/sys/arch/consts.h File Reference | 3154 |
| 16.468.1 Detailed Description | 3155 |
| 16.469 consts.h | 3155 |
| 16.470 arm64/l4/sys/arch/consts.h File Reference | 3155 |
| 16.470.1 Detailed Description | 3156 |
| 16.471 consts.h | 3156 |
| 16.472 l4/re/consts.h File Reference | 3156 |
| 16.472.1 Detailed Description | 3157 |
| 16.472.2 Enumeration Type Documentation | 3158 |
| 16.472.2.1 anonymous enum | 3158 |
| 16.473 consts.h | 3158 |
| 16.474 l4/sys/consts.h File Reference | 3158 |
| 16.474.1 Detailed Description | 3160 |
| 16.474.2 Macro Definition Documentation | 3160 |
| 16.474.2.1 L4_CAP_SIZE | 3160 |
| 16.474.2.2 L4_REPLY_CAP_BIT | 3161 |
| 16.474.2.3 L4_SYSF_CALL | 3161 |
| 16.474.2.4 L4_SYSF_OPEN_WAIT | 3161 |
| 16.474.2.5 L4_SYSF_RECV | 3161 |
| 16.474.2.6 L4_SYSF_REPLY | 3162 |
| 16.474.2.7 L4_SYSF_REPLY_AND_WAIT | 3162 |
| 16.474.2.8 L4_SYSF_SEND | 3162 |

| | |
|--|------|
| 16.474.2.9 L4_SYSF_SEND_AND_WAIT | 3162 |
| 16.474.2.10 L4_SYSF_WAIT | 3163 |
| 16.475 consts.h | 3163 |
| 16.476 riscv/l4/sys/arch/consts.h File Reference | 3165 |
| 16.476.1 Detailed Description | 3165 |
| 16.477 consts.h | 3166 |
| 16.478 x86/l4/sys/arch/consts.h File Reference | 3166 |
| 16.478.1 Detailed Description | 3166 |
| 16.479 consts.h | 3166 |
| 16.480 capability.h | 3167 |
| 16.481 l4/re/consts File Reference | 3171 |
| 16.481.1 Detailed Description | 3171 |
| 16.482 consts | 3172 |
| 16.483 consts | 3172 |
| 16.484 ipc_array | 3173 |
| 16.485 ipc_basics | 3176 |
| 16.486 l4/sys/cxx/ipc_client File Reference | 3180 |
| 16.486.1 Macro Definition Documentation | 3182 |
| 16.486.1.1 L4_RPC_DEF | 3182 |
| 16.487 ipc_client | 3182 |
| 16.488 ipc_epiface | 3183 |
| 16.489 l4/sys/cxx/ipc_iface File Reference | 3188 |
| 16.489.1 Detailed Description | 3190 |
| 16.489.2 Macro Definition Documentation | 3191 |
| 16.489.2.1 L4_INLINE_RPC | 3191 |
| 16.489.2.2 L4_INLINE_RPC_NF | 3191 |
| 16.489.2.3 L4_INLINE_RPC_NF_OP | 3192 |
| 16.489.2.4 L4_INLINE_RPC_OP | 3192 |
| 16.489.2.5 L4_RPC | 3193 |
| 16.489.2.6 L4_RPC_NF | 3193 |
| 16.489.2.7 L4_RPC_NF_OP | 3194 |
| 16.489.2.8 L4_RPC_OP | 3194 |
| 16.490 ipc_iface | 3195 |
| 16.491 ipc_legacy | 3200 |
| 16.492 ipc_ret_array | 3200 |
| 16.493 l4/cxx/ipc_server File Reference | 3202 |
| 16.493.1 Detailed Description | 3202 |
| 16.494 ipc_server | 3203 |
| 16.495 ipc_server | 3203 |
| 16.496 ipc_server_loop | 3208 |
| 16.497 ipc_string | 3212 |
| 16.498 l4/sys/cxx/ipc_types File Reference | 3213 |

| | |
|--|------|
| 16.499 ipc_types | 3215 |
| 16.500 ipc_varg | 3222 |
| 16.501 limits | 3227 |
| 16.502 l4/sys/cxx/smart_capability_1x File Reference | 3229 |
| 16.503 smart_capability_1x | 3230 |
| 16.504 l4/sys/cxx/types File Reference | 3232 |
| 16.505 types | 3234 |
| 16.506 l4/sys/debugger File Reference | 3238 |
| 16.506.1 Detailed Description | 3240 |
| 16.507 debugger | 3240 |
| 16.508 l4/sys/debugger.h File Reference | 3241 |
| 16.508.1 Detailed Description | 3243 |
| 16.508.2 Enumeration Type Documentation | 3243 |
| 16.508.2.1 anonymous enum | 3243 |
| 16.509 debugger.h | 3243 |
| 16.510 l4/sys/err.h File Reference | 3247 |
| 16.510.1 Detailed Description | 3248 |
| 16.511 err.h | 3248 |
| 16.512 l4/sys/exception File Reference | 3249 |
| 16.512.1 Detailed Description | 3250 |
| 16.513 exception | 3250 |
| 16.514 l4/sys/factory File Reference | 3250 |
| 16.514.1 Detailed Description | 3252 |
| 16.515 factory | 3252 |
| 16.516 l4/sys/factory.h File Reference | 3254 |
| 16.516.1 Detailed Description | 3255 |
| 16.517 factory.h | 3256 |
| 16.518 l4/sys/icu File Reference | 3260 |
| 16.518.1 Detailed Description | 3261 |
| 16.519 icu | 3261 |
| 16.520 l4/sys/icu.h File Reference | 3262 |
| 16.520.1 Detailed Description | 3264 |
| 16.521 icu.h | 3265 |
| 16.522 iommu | 3268 |
| 16.523 ipc.h | 3268 |
| 16.524 ipc.h | 3269 |
| 16.525 ipc.h | 3269 |
| 16.526 l4/sys/ipc.h File Reference | 3270 |
| 16.526.1 Detailed Description | 3272 |
| 16.526.2 Function Documentation | 3273 |
| 16.526.2.1 l4_ipc_to_errno() | 3273 |
| 16.527 ipc.h | 3273 |

| | |
|--|------|
| 16.528 ipc.h | 3277 |
| 16.529 x86/I4/sys/arch/ipc.h File Reference | 3277 |
| 16.529.1 Detailed Description | 3278 |
| 16.530 ipc.h | 3278 |
| 16.531 I4/sys/ipc_gate File Reference | 3279 |
| 16.531.1 Detailed Description | 3280 |
| 16.532 ipc_gate | 3281 |
| 16.533 I4/sys/ipc_gate.h File Reference | 3281 |
| 16.533.1 Detailed Description | 3282 |
| 16.534 ipc_gate.h | 3283 |
| 16.535 I4/sys/irq File Reference | 3284 |
| 16.535.1 Detailed Description | 3286 |
| 16.536 irq | 3286 |
| 16.537 I4/sys/kdebug.h File Reference | 3288 |
| 16.537.1 Detailed Description | 3289 |
| 16.537.2 Enumeration Type Documentation | 3290 |
| 16.537.2.1 I4_kdebug_ops_t | 3290 |
| 16.537.3 Function Documentation | 3290 |
| 16.537.3.1 __I4_kdebug_3_text() | 3290 |
| 16.537.3.2 __I4_kdebug_op() | 3291 |
| 16.537.3.3 __I4_kdebug_op_1() | 3292 |
| 16.537.3.4 __I4_kdebug_text() | 3294 |
| 16.537.3.5 I4_kd_enter() | 3296 |
| 16.537.3.6 I4_kd_outchar() | 3296 |
| 16.537.3.7 I4_kd_outdec() | 3297 |
| 16.537.3.8 I4_kd_outhex12() | 3298 |
| 16.537.3.9 I4_kd_outhex16() | 3298 |
| 16.537.3.10 I4_kd_outhex20() | 3299 |
| 16.537.3.11 I4_kd_outhex32() | 3299 |
| 16.537.3.12 I4_kd_outhex64() | 3300 |
| 16.537.3.13 I4_kd_outhex8() | 3300 |
| 16.537.3.14 I4_kd_outnstring() | 3301 |
| 16.537.3.15 I4_kd_outstring() | 3302 |
| 16.537.3.16 I4_kd_outumword() | 3302 |
| 16.538 kdebug.h | 3303 |
| 16.539 I4/sys/kdump.h File Reference | 3305 |
| 16.539.1 Detailed Description | 3306 |
| 16.539.2 Function Documentation | 3306 |
| 16.539.2.1 fiasco_dump_kmem_stats() | 3306 |
| 16.540 kdump.h | 3307 |
| 16.541 I4/sys/kernel_object.h File Reference | 3307 |
| 16.541.1 Detailed Description | 3308 |

| | |
|---|------|
| 16.542 kernel_object.h | 3308 |
| 16.543 l4/sys/kip File Reference | 3309 |
| 16.543.1 Detailed Description | 3310 |
| 16.544 kip | 3310 |
| 16.545 kobject | 3312 |
| 16.546 l4/sys/ktrace.h File Reference | 3312 |
| 16.546.1 Detailed Description | 3313 |
| 16.547 ktrace.h | 3314 |
| 16.548 amd64/l4/sys/arch/l4int.h File Reference | 3314 |
| 16.548.1 Detailed Description | 3315 |
| 16.549 l4int.h | 3315 |
| 16.550 arm/l4/sys/arch/l4int.h File Reference | 3315 |
| 16.550.1 Detailed Description | 3315 |
| 16.551 l4int.h | 3316 |
| 16.552 arm64/l4/sys/arch/l4int.h File Reference | 3316 |
| 16.552.1 Detailed Description | 3316 |
| 16.553 l4int.h | 3316 |
| 16.554 l4/sys/l4int.h File Reference | 3317 |
| 16.554.1 Detailed Description | 3318 |
| 16.555 l4int.h | 3318 |
| 16.556 riscv/l4/sys/arch/l4int.h File Reference | 3318 |
| 16.556.1 Detailed Description | 3319 |
| 16.557 l4int.h | 3319 |
| 16.558 x86/l4/sys/arch/l4int.h File Reference | 3319 |
| 16.558.1 Detailed Description | 3320 |
| 16.559 l4int.h | 3320 |
| 16.560 l4/sys/memdesc.h File Reference | 3320 |
| 16.560.1 Detailed Description | 3321 |
| 16.561 memdesc.h | 3322 |
| 16.562 meta | 3323 |
| 16.563 l4/sys/meta File Reference | 3324 |
| 16.563.1 Detailed Description | 3325 |
| 16.564 meta | 3325 |
| 16.565 l4/sys/obj_info.h File Reference | 3325 |
| 16.565.1 Detailed Description | 3326 |
| 16.565.2 Function Documentation | 3327 |
| 16.565.2.1 l4_debugger_query_obj_infos() | 3327 |
| 16.566 obj_info.h | 3328 |
| 16.567 l4/sys/pager File Reference | 3329 |
| 16.567.1 Detailed Description | 3331 |
| 16.568 pager | 3331 |
| 16.569 l4/sys/platform_control File Reference | 3331 |

| | |
|---|------|
| 16.569.1 Detailed Description | 3332 |
| 16.570 platform_control | 3333 |
| 16.571 platform_control.h | 3333 |
| 16.572 platform_control.h | 3333 |
| 16.573 platform_control.h | 3334 |
| 16.574 I4/sys/platform_control.h File Reference | 3334 |
| 16.574.1 Detailed Description | 3335 |
| 16.575 platform_control.h | 3335 |
| 16.576 platform_control.h | 3337 |
| 16.577 platform_control.h | 3338 |
| 16.578 I4/sys/rcv_endpoint File Reference | 3338 |
| 16.578.1 Detailed Description | 3339 |
| 16.579 rcv_endpoint | 3339 |
| 16.580 I4/sys/rcv_endpoint.h File Reference | 3340 |
| 16.580.1 Detailed Description | 3341 |
| 16.580.2 Enumeration Type Documentation | 3341 |
| 16.580.2.1 L4_rcv_ep_ops | 3341 |
| 16.581 rcv_endpoint.h | 3341 |
| 16.582 I4/sys/scheduler File Reference | 3342 |
| 16.582.1 Detailed Description | 3343 |
| 16.583 scheduler | 3343 |
| 16.584 I4/sys/scheduler.h File Reference | 3344 |
| 16.584.1 Detailed Description | 3346 |
| 16.585 scheduler.h | 3347 |
| 16.586 I4/sys/semaphore File Reference | 3349 |
| 16.586.1 Detailed Description | 3351 |
| 16.587 semaphore | 3351 |
| 16.588 I4/sys/semaphore.h File Reference | 3351 |
| 16.588.1 Detailed Description | 3353 |
| 16.589 semaphore.h | 3353 |
| 16.590 I4/sys/smart_capability File Reference | 3354 |
| 16.590.1 Detailed Description | 3355 |
| 16.591 smart_capability | 3355 |
| 16.592 I4/sys/snd_destination File Reference | 3357 |
| 16.592.1 Detailed Description | 3358 |
| 16.593 snd_destination | 3359 |
| 16.594 I4/sys/snd_destination.h File Reference | 3359 |
| 16.594.1 Detailed Description | 3360 |
| 16.595 snd_destination.h | 3360 |
| 16.596 I4/sys/task File Reference | 3360 |
| 16.596.1 Detailed Description | 3362 |
| 16.597 task | 3362 |

| | |
|--|------|
| 16.598 task.h | 3363 |
| 16.599 task.h | 3363 |
| 16.600 task.h | 3363 |
| 16.601 l4/sys/task.h File Reference | 3363 |
| 16.601.1 Detailed Description | 3364 |
| 16.602 task.h | 3365 |
| 16.603 task.h | 3367 |
| 16.604 task.h | 3368 |
| 16.605 l4/cxx/thread File Reference | 3368 |
| 16.605.1 Detailed Description | 3369 |
| 16.606 thread | 3369 |
| 16.607 l4/sys/thread File Reference | 3370 |
| 16.607.1 Detailed Description | 3371 |
| 16.608 thread | 3371 |
| 16.609 l4/sys/thread_group File Reference | 3373 |
| 16.610 thread_group | 3373 |
| 16.611 l4/sys/thread_group.h File Reference | 3374 |
| 16.612 thread_group.h | 3375 |
| 16.613 l4/sys/typeinfo_svr File Reference | 3376 |
| 16.613.1 Detailed Description | 3377 |
| 16.614 typeinfo_svr | 3377 |
| 16.615 types.h | 3378 |
| 16.616 l4/sys/types.h File Reference | 3378 |
| 16.616.1 Detailed Description | 3380 |
| 16.616.2 Typedef Documentation | 3380 |
| 16.616.2.1 l4_proto_t | 3380 |
| 16.616.2.2 l4_ret_t | 3381 |
| 16.616.3 Function Documentation | 3381 |
| 16.616.3.1 l4_capability_next() | 3381 |
| 16.617 types.h | 3381 |
| 16.618 amd64/l4/sys/arch/utcb.h File Reference | 3384 |
| 16.618.1 Detailed Description | 3385 |
| 16.619 utcb.h | 3385 |
| 16.620 arm/l4/sys/arch/utcb.h File Reference | 3386 |
| 16.620.1 Detailed Description | 3387 |
| 16.621 utcb.h | 3388 |
| 16.622 arm64/l4/sys/arch/utcb.h File Reference | 3389 |
| 16.622.1 Detailed Description | 3390 |
| 16.623 utcb.h | 3390 |
| 16.624 l4/sys/utcb.h File Reference | 3391 |
| 16.624.1 Detailed Description | 3392 |
| 16.625 utcb.h | 3393 |

| | |
|--|------|
| 16.626 riscv/l4/sys/arch/utcb.h File Reference | 3395 |
| 16.626.1 Detailed Description | 3396 |
| 16.627 utcb.h | 3396 |
| 16.628 x86/l4/sys/arch/utcb.h File Reference | 3397 |
| 16.628.1 Detailed Description | 3399 |
| 16.629 utcb.h | 3399 |
| 16.630 l4/sys/vcon File Reference | 3400 |
| 16.630.1 Detailed Description | 3402 |
| 16.631 vcon | 3402 |
| 16.632 l4/sys/vcon.h File Reference | 3402 |
| 16.632.1 Detailed Description | 3405 |
| 16.632.2 Enumeration Type Documentation | 3405 |
| 16.632.2.1 L4_vcon_read_flags | 3405 |
| 16.633 vcon.h | 3405 |
| 16.634 l4/sys/vcpu_context File Reference | 3408 |
| 16.634.1 Detailed Description | 3409 |
| 16.635 vcpu_context | 3409 |
| 16.636 vcpu_context.h | 3409 |
| 16.637 l4/sys/vm File Reference | 3409 |
| 16.637.1 Detailed Description | 3410 |
| 16.638 vm | 3411 |
| 16.639 l4/umalloc/umalloc.h File Reference | 3411 |
| 16.639.1 Detailed Description | 3412 |
| 16.639.2 Function Documentation | 3412 |
| 16.639.2.1 umalloc_area_create() | 3412 |
| 16.639.3 Variable Documentation | 3413 |
| 16.639.3.1 umalloc_area_granularity | 3413 |
| 16.640 umalloc.h | 3413 |
| 16.641 l4/sys/assert.h File Reference | 3413 |
| 16.641.1 Detailed Description | 3414 |
| 16.641.2 Macro Definition Documentation | 3415 |
| 16.641.2.1 l4_assert | 3415 |
| 16.642 assert.h | 3415 |
| 16.643 l4/util/assert.h File Reference | 3416 |
| 16.643.1 Detailed Description | 3417 |
| 16.644 assert.h | 3417 |
| 16.645 atomic.h | 3419 |
| 16.646 l4/cxx/atomic.h File Reference | 3419 |
| 16.646.1 Detailed Description | 3420 |
| 16.647 atomic.h | 3420 |
| 16.648 l4/util/atomic.h File Reference | 3421 |
| 16.648.1 Detailed Description | 3423 |

| | |
|---|------|
| 16.649 atomic.h | 3424 |
| 16.650 l4/util/backtrace.h File Reference | 3428 |
| 16.650.1 Detailed Description | 3429 |
| 16.650.2 Function Documentation | 3429 |
| 16.650.2.1 l4util_backtrace() | 3429 |
| 16.651 backtrace.h | 3430 |
| 16.652 l4/util/base64.h File Reference | 3430 |
| 16.652.1 Detailed Description | 3431 |
| 16.653 base64.h | 3432 |
| 16.654 l4/util/bitops.h File Reference | 3432 |
| 16.654.1 Detailed Description | 3434 |
| 16.655 bitops.h | 3434 |
| 16.656 l4/util/elf.h File Reference | 3437 |
| 16.656.1 Detailed Description | 3443 |
| 16.657 elf.h | 3444 |
| 16.658 l4/util/getopt.h File Reference | 3454 |
| 16.658.1 Detailed Description | 3455 |
| 16.659 getopt.h | 3455 |
| 16.660 l4/util/keymap.h File Reference | 3456 |
| 16.660.1 Detailed Description | 3457 |
| 16.661 keymap.h | 3457 |
| 16.662 kip.h | 3457 |
| 16.663 kip.h | 3458 |
| 16.664 kip.h | 3458 |
| 16.665 l4/sys/kip.h File Reference | 3458 |
| 16.665.1 Detailed Description | 3459 |
| 16.665.2 Macro Definition Documentation | 3460 |
| 16.665.2.1 l4_kip_for_each_feature | 3460 |
| 16.665.3 Function Documentation | 3460 |
| 16.665.3.1 l4_kip_kernel_has_feature() | 3460 |
| 16.666 kip.h | 3461 |
| 16.667 l4/util/kip.h File Reference | 3463 |
| 16.668 kip.h | 3463 |
| 16.669 kip.h | 3464 |
| 16.670 kip.h | 3464 |
| 16.671 l4/util/kprintf.h File Reference | 3464 |
| 16.671.1 Detailed Description | 3464 |
| 16.672 kprintf.h | 3465 |
| 16.673 l4/util/l4mod.h File Reference | 3465 |
| 16.673.1 Detailed Description | 3466 |
| 16.673.2 Enumeration Type Documentation | 3466 |
| 16.673.2.1 l4util_l4mod_mod_info_flag | 3466 |

| | |
|---|------|
| 16.674 l4mod.h | 3467 |
| 16.675 l4/util/list_alloc.h File Reference | 3467 |
| 16.675.1 Detailed Description | 3468 |
| 16.675.2 Function Documentation | 3468 |
| 16.675.2.1 l4la_alloc() | 3468 |
| 16.675.2.2 l4la_avail() | 3469 |
| 16.675.2.3 l4la_dump() | 3469 |
| 16.675.2.4 l4la_free() | 3469 |
| 16.675.2.5 l4la_init() | 3470 |
| 16.676 list_alloc.h | 3470 |
| 16.677 l4/util/lock.h File Reference | 3470 |
| 16.677.1 Detailed Description | 3471 |
| 16.678 lock.h | 3471 |
| 16.679 l4/util/mb_info.h File Reference | 3472 |
| 16.679.1 Detailed Description | 3475 |
| 16.679.2 Macro Definition Documentation | 3475 |
| 16.679.2.1 l4util_mb_for_each_mmap_entry | 3475 |
| 16.679.2.2 L4UTIL_MB_MEMORY | 3475 |
| 16.679.2.3 MB_ART_MEMORY | 3475 |
| 16.680 mb_info.h | 3476 |
| 16.681 l4/util/parse_cmd.h File Reference | 3480 |
| 16.681.1 Detailed Description | 3481 |
| 16.682 parse_cmd.h | 3481 |
| 16.683 printf_helpers.h | 3482 |
| 16.684 l4/util/rand.h File Reference | 3482 |
| 16.684.1 Detailed Description | 3483 |
| 16.685 rand.h | 3483 |
| 16.686 l4/util/splitlog2.h File Reference | 3483 |
| 16.686.1 Detailed Description | 3484 |
| 16.687 splitlog2.h | 3484 |
| 16.688 util.h | 3485 |
| 16.689 l4/utrace/ring_buffer File Reference | 3486 |
| 16.689.1 Detailed Description | 3487 |
| 16.690 ring_buffer | 3487 |
| 16.691 l4/utrace/utrace File Reference | 3491 |
| 16.691.1 Detailed Description | 3492 |
| 16.692 utrace | 3492 |
| 16.693 vbus | 3494 |
| 16.694 l4/vbus/vbus.h File Reference | 3495 |
| 16.694.1 Detailed Description | 3497 |
| 16.694.2 Enumeration Type Documentation | 3497 |
| 16.694.2.1 anonymous enum | 3497 |

| | |
|---|------|
| 16.694.2.2 l4vbus_icu_src_types | 3498 |
| 16.695 vbus.h | 3498 |
| 16.696 vbus_generic | 3499 |
| 16.697 vbus_gpio | 3499 |
| 16.698 vbus_gpio-ops.h | 3501 |
| 16.699 vbus_gpio.h | 3501 |
| 16.700 vbus_i2c.h | 3502 |
| 16.701 vbus_inhibitor.h | 3502 |
| 16.702 l4/vbus/vbus_interfaces.h File Reference | 3502 |
| 16.702.1 Detailed Description | 3504 |
| 16.702.2 Typedef Documentation | 3504 |
| 16.702.2.1 l4vbus_iface_type_t | 3504 |
| 16.702.3 Enumeration Type Documentation | 3504 |
| 16.702.3.1 anonymous enum | 3504 |
| 16.702.3.2 l4vbus_iface_type_t | 3505 |
| 16.702.4 Function Documentation | 3505 |
| 16.702.4.1 l4vbus_subinterface_supported() | 3505 |
| 16.703 vbus_interfaces.h | 3506 |
| 16.704 vbus_mcspi.h | 3506 |
| 16.705 vbus_pci | 3506 |
| 16.706 vbus_pci-ops.h | 3507 |
| 16.707 vbus_pci.h | 3508 |
| 16.708 vbus_pm-ops.h | 3508 |
| 16.709 vbus_pm.h | 3509 |
| 16.710 l4/vbus/vbus_types.h File Reference | 3509 |
| 16.710.1 Detailed Description | 3510 |
| 16.710.2 Enumeration Type Documentation | 3511 |
| 16.710.2.1 l4vbus_device_flags_t | 3511 |
| 16.710.2.2 l4vbus_resource_flags_t | 3511 |
| 16.710.2.3 l4vbus_resource_type_t | 3511 |
| 16.711 vbus_types.h | 3512 |
| 16.712 vdevice-ops.h | 3513 |
| 16.713 l4/vcpu/vcpu File Reference | 3513 |
| 16.713.1 Detailed Description | 3514 |
| 16.714 vcpu | 3514 |
| 16.715 l4/sys/vcpu.h File Reference | 3515 |
| 16.715.1 Detailed Description | 3517 |
| 16.715.2 Function Documentation | 3517 |
| 16.715.2.1 l4_vcpu_check_version() | 3517 |
| 16.716 vcpu.h | 3518 |
| 16.717 l4/vcpu/vcpu.h File Reference | 3518 |
| 16.717.1 Detailed Description | 3520 |

| | |
|--|-------------|
| 16.718 vcpu.h | 3520 |
| 17 Examples | 3523 |
| 17.1 hello/server/src/main.c | 3523 |
| 17.2 examples/sys/ipc/ipc_example.c | 3523 |
| 17.3 examples/sys/ipc/ipc.cfg | 3524 |
| 17.4 examples/sys/start-with-exc/main.c | 3525 |
| 17.5 examples/sys/singlestep/main.c | 3527 |
| 17.6 examples/sys/aliens/main.c | 3529 |
| 17.7 examples/sys/utcb-ipc/main.c | 3532 |
| 17.8 examples/sys/isr/main.c | 3534 |
| 17.9 examples/clntsrv/src/server.cc | 3536 |
| 17.10 examples/clntsrv/src/client.cc | 3537 |
| 17.11 examples/clntsrv/src/shared.h | 3538 |
| 17.12 examples/clntsrv/configs/clntsrv.cfg | 3538 |
| 17.13 examples/libs/l4re/c/ma+rm.c | 3538 |
| 17.14 examples/libs/l4re/c++/mem_alloc/ma+rm.cc | 3539 |
| 17.15 examples/libs/l4re/c++/shared_ds/ds_clnt.cc | 3540 |
| 17.16 examples/libs/l4re/c++/shared_ds/ds_srv.cc | 3542 |
| 17.17 examples/libs/l4re/c++/shared_ds/shared_ds.cfg | 3544 |
| 17.18 examples/libs/l4re/streammap/server.cc | 3545 |
| 17.19 examples/libs/l4re/streammap/client.cc | 3546 |
| 17.20 examples/libs/l4re/streammap/streammap.cfg | 3546 |
| 17.21 examples/libs/libirq/loop.c | 3547 |
| 17.22 examples/libs/libirq/async_isr.c | 3548 |
| 17.23 examples/sys/migrate/thread_migrate.cc | 3548 |
| 17.24 examples/sys/migrate/thread_migrate.cfg | 3550 |
| 17.25 tmpfs/lib/src/fs.cc | 3550 |
| 17.26 examples/libs/shmc/prodcons.c | 3557 |
| Index | 3561 |

Chapter 1

Overview

Welcome to the documentation of the L4Re Operating System Framework, or L4Re for short. There are two parts to this documentation: a user manual, which provides a birds eye view of L4Re and its environment, and a reference section which documents the complete programming API.

User Manual

1. [Introduction](#) shortly explains the concept of microkernels and introduces the basic terminology.
2. [Programming for L4Re](#) explains in detail the most important programming concepts.
3. [L4Re Servers](#) provides a quick overview over standard services running on the L4Re operating system.

Reference

The second part provides the complete reference of all classes and functions of the L4Re Operating System Framework as well as a list of example code.

Chapter 2

Introduction

The intention of this section is to provide a short overview about the [L4Re](#) Operating System Framework.

The general structure of a microkernel-based system will be introduced and the principal functionality of the servers in the basic environment outlined.

2.1 L4Re Microkernel

The [L4Re](#) Microkernel is the lowest-level component of software running in an L4Re-based system. The microkernel is the only component that runs in privileged processor mode. It does not include complex services such as program loading, device drivers, or file systems; those are implemented in user-level programs on top of it (a basic set of these services and abstractions is provided by the [L4](#) Runtime Environment).

Microkernel services are implemented in kernel objects. Tasks hold references to kernel objects in their respective *"object space"*, which is a kernel-protected table. These references are called *capabilities*. System calls to the microkernel are function invocations on kernel objects through the corresponding capabilities. These can be thought of as function invocations on object references in an object-oriented programming environment. Furthermore, if a task owns a capability, it may grant other tasks the same (or fewer) rights on this object by passing the capability from its own to the other task's object space.

From a design perspective, capabilities are a concept that enables flexibility in the system structure. A thread that invokes an object through a capability does not need to care about where this object is implemented. In fact, it is possible to implement all objects either in the kernel or in a user-level server and replace one implementation with the other transparently for clients.

2.1.1 Communication

The basic communication mechanism in L4-based systems is called *"Inter Process Communication (IPC)"*. It is always synchronous, i.e. both communication partners need to actively rendezvous for IPC. In addition to transmitting arbitrary data between threads, IPC is also used to resolve hardware exceptions, faults and for virtual memory management.

2.1.2 Kernel Objects

The following list gives a short overview of the kernel objects provided by the [L4Re](#) Microkernel:

- **Task** A task comprises a memory address space (represented by the task's page table), an object space (holding the kernel protected capabilities), and on x86 an IO-port address space.
- **Thread** A thread is bound to a task and executes code. Multiple threads can coexist in one task and are scheduled by the microkernel's scheduler.
- **Factory** A factory is used by applications to create new kernel objects. Access to a factory is required to create any new kernel object. Factories can control and restrict object creation.
- **IPC Gate** An IPC gate is used to create a secure communication channel between different tasks. It embeds a label (kernel protected payload) that securely identifies the gate through which a message is received. The gate label is not visible to and cannot be altered by the sender.
- **IRQ** IRQ objects provide access to hardware interrupts. Additionally, programs can create new virtual interrupt objects and trigger them. This allows to implement a signaling mechanism. The receiver cannot decide whether the interrupt is a physical or virtual one.
- **Vcon** Provides access to the in-kernel debugging console (input and output). There is only one such object in the kernel and it is only available, if the kernel is built with debugging enabled. This object is typically interposed through a user-level service or without debugging in the kernel can be completely based on user-level services.
- **Scheduler** Implements scheduling policy and assignment of threads to CPUs, including CPU statistics.

2.2 L4Re System Structure

The system has a multi-tier architecture consisting of the following layers depicted in the figure below:

- **Microkernel** The microkernel is the component at the lowest level of the software stack. It is the only piece of software that is running in the privileged mode of the processor.
- **Tasks** Tasks are the basic containers (address spaces) in which system services and applications are executed. They run in the processor's deprivileged user mode.

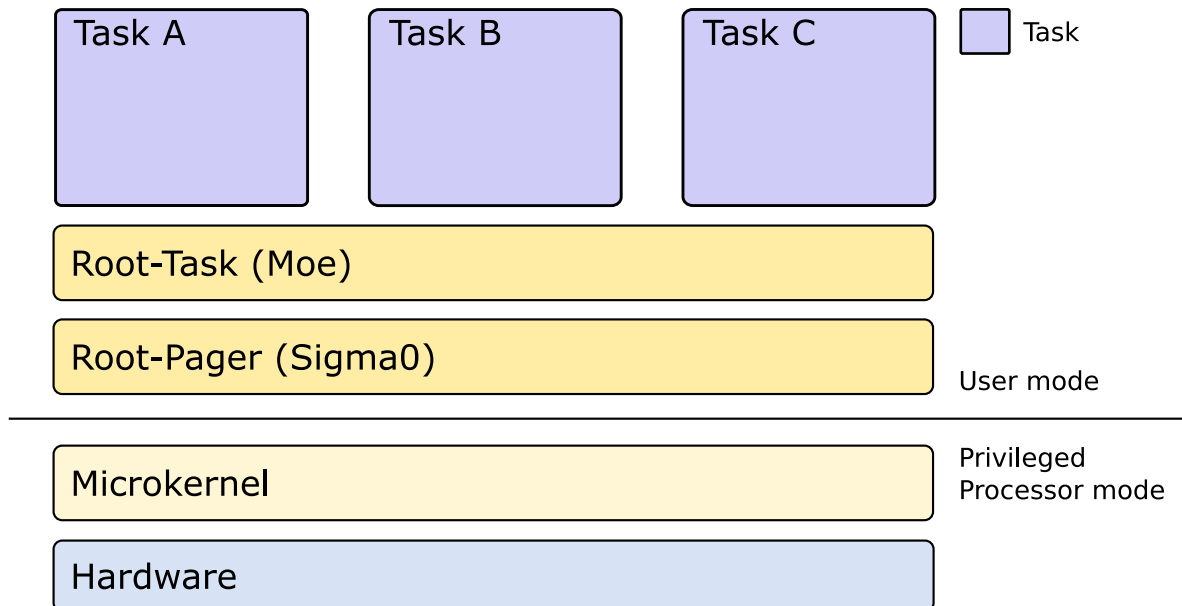


Figure 2.1 Basic Structure of an L4Re based system

In terms of functionality, the system is structured as follows:

- Microkernel** The kernel provides primitives to execute programs in tasks, to enforce isolation among them, and to provide means of secure communication in order to let them cooperate. As the kernel is the most privileged, security-critical software component in the system, it is a general design goal to make it as small as possible in order to reduce its attack surface. It provides only a minimal set of mechanisms that are necessary to support applications.
- Runtime Environment** The small kernel offers a concise set of interfaces, but these are not necessarily suited for building applications directly on top of it. The [L4Re](#) Runtime Environment aims at providing more convenient abstractions for application development. It comprises low-level software components that interface directly with the microkernel. The root pager *sigma0* and the root task *Moe* are the most basic components of the [L4Re](#) Runtime Environment. Other services (e.g., for device enumeration) use interfaces provided by them.
- Applications** Applications run on top of the system and use services provided by the runtime environment – or by other applications. There may be several types of applications in the system and even virtual machine monitors and device drivers are considered applications in the terminology used in this document. They are running alongside other applications on the system.

Lending terminology from the distributed systems area, applications offering services to other applications are usually called *servers*, whereas applications using those services are named *clients*. Being in both roles is also common, for instance, a file system server may be viewed as a server with respect to clients using the file system, while the server itself may also act as a client of a hard disk driver.

2.3 L4Re Runtime Environment

The [L4Re](#) Runtime Environment provides a basic set of services and abstractions, which are useful to implement and run user-level applications on top of the [L4Re](#) Microkernel. They form the [L4Re](#) Operating System Framework.

The [L4Re](#) Operating System Framework consists of a set of libraries and servers. [L4Re](#) follows an object-oriented design. Server interfaces are object-oriented, and the implementation is also object-oriented.

A minimal L4Re-based application needs 3 components to be booted beforehand: the [L4Re](#) Microkernel, the root pager (Sigma0), and the root task (Moe). The Sigma0 root pager initially owns all system resources, but is usually used only to resolve page faults for the Moe root task. Moe provides the essential services to normal user applications such as an initial program loader, a region-map service for virtual memory management, and a memory (data space) allocator.

Chapter 3

Programming for L4Re

This part of the documentation discusses the concept of microkernel-based programming in more detail.

You should already have a basic understanding of the [L4Re](https://l4re.org/) programming environment from the tutorials available on l4re.org(<https://l4re.org/>).

- [L4 Inter-Process Communication \(IPC\)](#)
- [Kernel ABI](#)
- [Capabilities and Naming](#)
- [Spaces and Mappings](#)
- [Initial Environment and Application Bootstrapping](#)
- [Memory management - Data Spaces and the Region Map](#)
- [Program Input and Output](#)
- [Initial Memory Allocator and Factory](#)
- [Application and Server Building Blocks](#)
- [Pthread Support](#)
- tasks and threads
- communication channels
- server loops
- [Interface Definition Language](#)
- hardware access
- [L4Re Build System](#)
- [Kernel Factory](#)

3.1 L4 Inter-Process Communication (IPC)

Inter-process communication (IPC) is the fundamental communication mechanism in the [L4Re](#) Microkernel.

Basically IPC invokes a subroutine in a different context using input and output parameters. It is used to communicate between userland threads and, as a special case, to communicate between a userland thread and a kernel object. IPC provides the only (non-debugging) way of doing system calls.

3.1.1 IPC mechanism

When using this API, an IPC operation can be conducted using the `l4_ipc()` function (or one of its related [helpers](#)). In general it causes a method to be invoked on the called kernel object. An IPC operation consists of a send and receive phase, but either of them can be skipped, that is, an IPC operation can consist of only either a send or a receive phase. IPC is always synchronous and blocking and can be given a timeout. Timeouts can be specified separately for each phase. Invoking the IPC syscall without any phase is deprecated.

On the lowest abstraction level, IPC operations need the following arguments:

- [flags](#) describing the IPC mode,
- the [capability selector](#) of the communication partner,
- a [label](#),
- a [message tag](#), and
- a pair of [timeout](#) values.

During an IPC operation the kernel accesses the UTCB of the current thread to read parameters which are not passed as direct arguments.

As result of an IPC operation the kernel returns a message tag and a label and the kernel also changes UTCB content. For the detailed call signature, refer to `l4_ipc()`. Furthermore, depending on the IPC parameters, the kernel might have transferred the FPU state and capabilities (memory, I/O ports, and/or object capabilities) from the sender to the receiving thread.

The transition between the IPC send phase and the IPC receive phase is atomic, that is, as soon as the send phase has finished, the thread receive phase starts. A relative receive timeout does not start before the send phase has finished (see also below) and a thread which received an IPC call from another thread can assume that the other thread is ready to receive the reply message and the replying thread can therefore reply with a timeout of zero, see [IPC Timeouts](#).

For performance optimization and under certain conditions, the kernel may perform a context switch from the IPC sender to the IPC receiver without consulting the scheduler after the send phase finished. For instance, a client performing an IPC call to a server has to wait for the server anyway. Hence, after the client request was sent to the server, the kernel switches directly to the server thread. This behavior can be disabled by setting the `L4_MSGTAG_SCHEDULE` flag in the sender message tag (see below).

3.1.1.1 IPC Flags

The capability selector *flags* (`L4_SYSF_SEND`, `L4_SYSF_RECV`, `L4_SYSF_OPEN_WAIT`, `L4_SYSF_REPLY`) are used by the invoking thread to define the intended IPC operation. The variants of `l4_ipc()` (see [Object Invocation](#)) use the flags

- to request the IPC phases (send-only IPC, receive-only IPC, IPC with send and receive phase), and
- to decide, if the reply capability (see [below](#)) should be used instead of the capability of a dedicated kernel object as target for the send phase (*reply*), and
- to decide, if receiving should wait for an incoming message from any possible sender (*open wait*) instead of a message from a dedicated sender (*closed wait*).

3.1.1.2 Partner capability selector

The *partner capability selector* defines a kernel object as partner of the IPC operation. Some kernel objects forward IPC to a userland thread.

Basically an object capability is represented by `l4_cap_idx_t` where the bits starting from `L4_CAP_SHIFT` are used as index into the local capability table of the current address space.

Specifying `L4_INVALID_CAP` as target for an IPC operation is equivalent to specifying a thread capability of the current thread with full permissions. As a result, the userland thread either communicates with its corresponding kernel thread object (if `L4_PROTO_THREAD` is specified as protocol value, see the description of the message tag below) or the IPC target is the current userland thread. In the latter case, no IPC will be performed and the send phase and the receive phase will block until the corresponding timeout has expired, see below.

A special partner is defined by the *reply capability*. Since it would be impractical (and a security flaw) to always pass an explicit object capability to reply to for each IPC, the kernel generates an implicit one that can be used for just that purpose if the IPC contains an **open wait** phase. The reply capability is valid after a receive phase and points to the kernel object that sent the IPC just received. It can be used only once. The reply capability is selected by setting the `L4_SYSF_REPLY` flag, see above.

3.1.1.3 IPC Label

The IPC label is a machine word which is transferred unchanged from the IPC sender to the IPC receiver when directly sending to a userland thread. However, the primary purpose of the label is to create a relationship between an `L4::Rcv_endpoint` (`L4::lpc_gate` or `L4::lrq`) and the bound thread.

During `L4::Rcv_endpoint::bind_thread()`, a label is specified. If the thread receives an IPC message through the endpoint, that label is delivered to the receiving thread as output parameter of `l4_ipc()` instead of the label specified during the corresponding IPC send operation (see the detailed description of `L4::lpc_gate` for more details on the label with IPC gates). The label is usually used by the receiving thread to invoke the object which is responsible for handling the IPC request of the corresponding endpoint. This mechanism is used by the `L4::Epiface` mechanism for server loops.

3.1.1.4 IPC Message Tag

The *message tag* (`l4_msgtag_t`) describes the payload of the IPC and can also be used to enable certain features. It contains:

- a *protocol value* (cf. `l4_msgtag_t::label()`, also called the tag's *label*),
- the number of items in *UTCB message registers* to transfer (cg. `l4_msgtag_t::words()` and `l4_msgtag_t::items()`), and
- flags (cf. `l4_msgtag_t::flags()` and `L4_msgtag_flags`, may be 0).

The information from the message tag is required by the kernel to transfer the message. The IPC system call returns a message tag as result of the IPC operation. On success, a copy of the message tag specified by the sender is returned if there is a receive phase. Without receive phase, a successful IPC syscall returns the message tag specified as input parameter.

Failures during IPC are signalled using the `L4_MSGTAG_ERROR` flag in the message tag. If this bit is set by the kernel, the content of the returned message tag apart from that bit is undefined and the kernel wrote the actual error code into the `l4_thread_regs_t::error` register of the UTCB (see also *IPC Thread Control Registers*). When an IPC error occurs after the rendezvous of the IPC partners, both partners observe the same error information. If, for

instance, the IPC was aborted using `L4::Thread::ex_regs()`, the sender gets an `L4_IPC_SECANCELED` error while the receiver gets an `L4_IPC_RECANCELED` error. The function `L4Re::chkipc()` can be used to verify that an IPC operation finished successfully: It throws an error if the IPC failed.

The *protocol value* is usually used to distinguish between different protocols of the same interface. Certain protocol IDs are pre-defined when talking to kernel objects, see `L4_msgtag_protocol`. From IPC point of view, the protocol value is just payload that is transferred from sender to receiver and hence doesn't have a dedicated meaning.

By convention, during IPC calls, the protocol value is used for return values, where negative values signify errors. See the [section](#) about L4 RPC return types for further information. The function `L4Re::chksys()` can be used to verify that an RPC call using IPC was successful: It throws an error if the IPC failed or if the returned protocol value is negative.

3.1.1.5 IPC Timeouts

As written above, IPC *timeouts* are specified separately for the send phase (IPC send timeout) and the receive phase (IPC receive timeout). The timeout of one phase is encapsulated by `l4_timeout_s`. Both timeouts are combined into a single `l4_timeout_t` parameter. Timeouts are either relative to the current time of the invoking thread or absolute. In the latter case, the absolute time of the deadline of the respective phase is written into a UTCB buffer register (see `l4_timeout_abs()`). The relative timeout of the receive phase starts immediately after the send phase has finished.

Two specific timeout values are sufficient for most IPC operations:

- `L4_IPC_TIMEOUT_NEVER` is used if the IPC partner might not yet be ready. Usually a client trusting a server will perform an IPC call with an infinite timeout for both phases. The send phase will take some time if the IPC receiver is currently not ready. The receive phase will take some time as the server needs time to serve the request. Also, a server will usually wait with an infinite receive timeout for the next request (see below for a possible exception).
- `L4_IPC_TIMEOUT_0` is used when it is either certain that the IPC receiver is currently ready or if the IPC sender doesn't want to wait if the IPC receiver is not ready. The former case applies to a thread which was called with an IPC call, for example a server got a client request. The reply to the IPC call should use a timeout of zero to ensure that a client doesn't block a server (server could not deliver the result of the request). See also `L4::ipc_srv::Default_timeout`. Another case is an IPC send operation for waking up another thread from an IPC receive operation. If the other thread is not ready to receive, then it might be already woken up and it does not make sense to wait any longer. Also triggering an IRQ is usually done with a send timeout of 0, see `L4::Triggerable::trigger()`.

In certain cases it also makes sense to specify an IPC timeout different from "never" or "zero":

- A server might leave the server loop after some time to perform idle work (see `L4::ipc_srv::Server_iface::add_timeout()`).
- A thread does not want to wait for an infinite time if the partner is not ready. This could be also some safety measure.
- A thread wants to block for a certain amount of time without consuming CPU time. The `l4_ipc_sleep()` function specifies the `L4_INVALID_CAP` as target for an IPC receive operation and specifies the intended relative waiting period as IPC timeout. Waiting for an absolute timeout would be possible with similar code.

Note

The kernel IPC path is optimized for the two special cases using `L4_IPC_TIMEOUT_NEVER` and `L4_IPC_TIMEOUT_0`. Specifying a different timeout causes more maintenance effort for the kernel.

Special care is required if a finite timeout for the receive phase of an IPC call is specified: The IPC receive operation could abort before the partner was able to send the reply message. Under certain circumstances the partner may still have the temporary reply capability to the calling thread and may use this capability to reply to the caller at a later, unexpected time specifying an arbitrary IPC label. This case is relevant for servers which call another, possibly untrusted, server while serving a client request.

3.1.1.6 User-level Thread Control Block

The **UTCB** is located on a power-of-2-sized and power-of-2-aligned memory area shared between userland and the kernel (`L4::Task::add_ku_mem()`). The UTCB is bound to a thread during the `L4::Thread::control()` operation with the `L4::Thread::Attr` parameters set up using `L4::Thread::Attr::bind()`. The UTCB is used for IPC-related data exchange and is set up before invoking `l4_ipc()`. To access certain parts of the UTCB, the corresponding functions have to be used (there is no data type `L4_utcb` or similar). The UTCB consists of:

- the **message registers** `MR[0], MR[1], ..., MR[n-1]` with $n = \text{L4_UTCB_GENERIC_DATA_SIZE}$ (access using `l4_utcb_mr()`),
- the **buffer descriptor register** `BDR` (access using `l4_utcb_br()`, see `l4_buf_regs_t::bdr`),
- the **buffer registers** `BR[0], BR[1], ..., BR[m-1]` with $m = \text{L4_UTCB_GENERIC_BUFFERS_SIZE}$ (access using `l4_utcb_br()`),
- the **thread control registers** (access using `l4_utcb_tcr()`, includes the IPC error code), and
- in case of an exception IPC, the register state of the thread which triggered the exception (access using `l4_utcb_exc()`).

IPC to a kernel object requires the setup of the UTCB of the corresponding userlevel thread. IPC between userlevel threads requires the setup of UTCBs of both partners.

The kernel changes only the following UTCB content:

- The message registers of the UTCB of the receiver of an IPC, and
- the IPC error field of the thread invoking `l4_ipc()` if there was an error during IPC.

3.1.1.6.1 IPC Message registers

The *message registers* contain *untyped items* and *typed items*. The sender's typed items are interpreted by the kernel in conjunction with the receiver's *receive items*. Each typed send item occupies two message registers. The untyped items, on the other hand, are free to be used by the communication partners to exchange data: The content of all message registers used for untyped items (`l4_msgtag_t::words()`) is copied from the UTCB of the IPC sender to the UTCB of the IPC receiver.

A typed send item consists of a *flexpage* (see `l4_fpage()`, `l4_obj_fpage()`, and `l4_iofpage()`) of the to be transferred capabilities (*flexpage word*) and a *message word*. There are two types of send items: *map items* and *void items*. For a void item, the message word is all zero. For a map item, the message word contains:

- the *compound bit* allowing to use the same receive buffer for the subsequent typed send item (scatter-gather behavior, see `L4_ITEM_CONT` of `l4_msg_item_consts_t`),
- the *type bit* defining this typed send item as a *map item*,
- the *grant flag* for delegating the access to the flexpage from the sender to the receiver and atomically removing the rights from the sender (see `l4_msg_item_consts_t`),
- *attributes* with semantics depending on the item type; for memory mappings, they contain cacheability information (see `l4_fpage_cacheability_opt_t`); for objects, they contain additional rights (see [Attributes and additional permissions for object send items](#)),
- the *send base* (also called *hot spot*) defining what is actually mapped when send and receive flexpages have a different size.

A typed send item can be created using `l4_sndfpage_add()`. This function sets the compound bit unconditionally. Alternatively, the functions `l4_map_control()` and `l4_map_obj_control()` can be used to set up the message word of a map item for a memory flexpage respective for objects.

See [below](#) for a description how typed items are transferred.

3.1.1.6.2 IPC Buffer Registers

The *buffer registers* and *buffer descriptor register* are interpreted by the kernel during the receive phase (if any). [Buffer](#) registers define *receive items* which are required to receive typed send items (memory, I/O ports or object capabilities). To specify a receive item, up to three buffer registers are required:

- A *small receive item* ([L4::lpc::Small_buf](#)) occupying one buffer register is sufficient to receive one object capability.
- A *receive item* ([L4::lpc::Rcv_fpage](#)) occupying two or three buffer registers (*message word*, a *flexpage*, and an optional destination task capability index) is required to receive memory flexpages, I/O ports, or multiple object capabilities.

3.1.1.6.3 IPC Buffer Descriptor Register

The buffer descriptor register defines indices of buffer registers used to receive dedicated types of send items and also contains a flag:

- 5 bits starting at [L4_BDR_MEM_SHIFT](#) define the index of the first receive item used for memory flexpages.
- 5 bits starting at [L4_BDR_IO_SHIFT](#) define the index of the first receive item used for I/O flexpages.
- 5 bits starting at [L4_BDR_OBJ_SHIFT](#) define the index of the first receive item used for object flexpages.
- The [L4_UTCB_INHERIT_FPU](#) can be set using [l4_utcb_inherit_fpu\(\)](#) and allows to receive the FPU state from the IPC sender. This is only relevant if the sender set [L4_MSGTAG_TRANSFER_FPU](#) in the message tag.

For most use cases, a BDR value of zero is sufficient. In that case, if `BR[0]` contains a void item, no capabilities are received. Otherwise, only one type of capabilities (memory, I/O ports or objects) can be received even if there are several receive items. For more complex setups that require receiving different types of capabilities in one receive operation, other BDR values are necessary.

The BDR of the receiving thread is only used by the kernel if at least one typed item is transferred during the IPC or if [L4_MSGTAG_TRANSFER_FPU](#) is set in the UTCB of the sending thread.

3.1.1.6.4 IPC Thread Control Registers

The [l4_thread_regs_t::error](#) register contains the IPC error code in case the [L4_MSGTAG_ERROR](#) flag is set in the message tag returned by an IPC syscall. Otherwise this register is not touched by the kernel. See [l4_ipc_tcr_error_t](#) for a detailed enumeration of all possible error codes during an IPC operation.

The [l4_thread_regs_t::free_marker](#) is set by the kernel to zero immediately before a thread is destroyed. This value indicates that the kernel does not longer use the UTCB and it can be re-used by other threads.

The other members of [l4_thread_regs_t](#) are never touched by the kernel.

3.1.1.7 Transfer of Typed Send Items

The kernel interprets all typed items in the sender UTCB ([l4_msgtag_t::items\(\)](#)) and performs the following operations while modifying the corresponding typed items in the receiver UTCB:

- If the message item of the sender is void, this item is ignored and the message word of the corresponding typed item in the receiver UTCB is set to zero (making it a void item). The flexpage word of this item is not changed.
- Otherwise, if the type bit of the message item of the sender is not set, the IPC is aborted with [L4_IPC_SEMSGCUT](#) / [L4_IPC_REMSGCUT](#).
- Otherwise, if there is a receive item corresponding to the flexpage type of the send item available (see [IPC Buffer Descriptor Register](#)), information described by the flexpage is transferred to the receiver.

In the last case, the message word of the typed item in the receiver UTCB is modified to contain the send base, the type and the size of the transferred flexpage, as well as a copy of the compound bit and the type bit of the send item. If the receiver ordered a local ID in the corresponding receive item, the kernel attempts to apply special rules, see [L4_RCV_ITEM_LOCAL_ID](#). Otherwise, regular mappings as described by the flexpage of the send item are created in the receiver space.

A receive item forms a *receive window* of a specific address and size in the receiver space. Each typed send item that is a map item requires a corresponding receive item. By default, there is a one-to-one mapping (one receive item per typed send item) but it is also possible to use one receive item to receive several typed send items: The compound bit (see [l4_msg_item_consts_t](#)) of a send item defines if the following typed send item shall use the same receive item as the current one for receiving the flexpage. If the compound bit is set, proper values of the send base shall be used to prevent overlapping of addresses in the receiver space.

The send base is relevant when the size of the receive flexpage differs from the size of the transferred resource. As a typical example, triggering a memory page fault opens a receive window covering the entire memory address space of the faulting thread. The pager will usually reply a memory flexpage smaller than the entire address space of the faulting thread. Hence, the pager has to specify a proper base which is used as offset of the sent object in the receive flexpage in the *receiver space*. If the sender flexpage is bigger than the receive window, a flexpage of the size of the receive window starting at the send base in the *sender space* is transferred to the receiver.

The kernel will stop transmission of typed items before [l4_msgtag_t::items\(\)](#) is reached either if it finds a void item as receive buffer or if the flexpage type of the send item does not match the flexpage type of the corresponding receive item. Under both conditions, the IPC is aborted with [L4_IPC_SEMSGCUT](#) / [L4_IPC_REMSGCUT](#).

Note

The kernel ignores the flexpage access rights of the receive items. The actual rights for a transferred resource in the target address space are defined by the access rights to that resource of the IPC sender address space and the flexpage access rights in the typed send item. Additionally, when transferring object capabilities, the transferred rights also depend on the sender's rights on the capability invoked for IPC.

The kernel does not unmap capabilities in the receive window when there is no capability present at the corresponding index at the sender. Further, the receiver cannot reliably detect at which capability indices it received capabilities in its receive windows. Therefore, before receiving from an untrusted source, all receive windows should be cleared. Otherwise the receiver may erroneously associate a capability in one of its receive windows with his last IPC partner although it was actually received in an earlier IPC.

However, the kernel indicates if at least one object capability was received or not, see [L4::lpc::Snd_fpage::cap_received\(\)](#).

3.1.2 Examples

A number of examples show the interplay of the concepts introduced so far.

3.1.2.1 User Thread to Kernel Object

The `L4::Scheduler` kernel object has a method `L4::Scheduler.idle_time()`. It takes a set of CPUs to query, represented by two machine words.

In user space, the function `L4::Cap<L4::Scheduler>->idle_time()` is called, which does the following:

- Fill `MR[0]` with a constant identifying the method being called (`L4_SCHEDULER_IDLE_TIME_OP`).
- Fill `MR[1]` and `MR[2]` with the two words making up the CPU set.
- Set up the message tag with the protocol value (`L4_PROTO_SCHEDULER`), the number of untyped and typed items (3 and 0), and some flags.
- Call `l4_ipc()` with the partner capability ID, the tag, the pointer to the filled UTCB, infinite timeouts, and with flags indicating a send and receive phase. (The label does not matter in this case.)

This function traps into kernel space using standard platform specific mechanisms. The kernel reads the protocol value on the message tag, checks that the partner capability ID refers to a valid object that speaks that protocol, and dispatches the message to the appropriate handler. The handler fills the first 64 bits of the message registers with the computed time value. This will cover `MR[0]` on 64-bit architectures and `MR[0]` and `MR[1]` on 32-bit architectures. It then sets up the return message tag:

- The number of untyped items is 1 or 2.
- The number of typed items is 0.
- The protocol value contains the return value and is set to 0.
- An error would be signalled as a negative protocol value. Under certain conditions (e.g. wrong kernel object capability specified), the error is signalled as IPC error (see `L4_MSGTAG_ERROR` in the description of the `IPC Message Tag`).
- (The return label is as irrelevant in this case as the send label.)

This reply is received by the receive phase of the original `l4_ipc()` call from userland. Finally the `l4_ipc()` function copies the time value out of the message registers and forwards it with a possible error from the message tag flags to its caller.

3.1.2.2 User Thread to User Thread

When the target kernel object is of type `L4::Thread` (or `L4::ipc_gate`, but we will cover this in the example below) and the message tag's protocol value is not `L4_PROTO_THREAD`, the kernel will forward the IPC message to the userland thread represented by the kernel object. There it can be received by a call to `l4_ipc()`. The restriction of the protocol number is necessary because one also wants to invoke `L4::Thread`'s control methods such as `L4::Thread.switch_to()` or `L4::Thread.ex_regs()`. Besides this restriction, the interpretation of all the IPC parameters and the untyped items of the UTCB is up to the communication partners.

3.1.2.2.1 Simple Messages

A simple example is a client calling a server to have a computation performed on a value: Similar to IPC from a userland thread to a kernel object, the client writes the value to `MR[0]`. It sets up the message tag with some agreed upon protocol value (which, as explained above, must be different from `L4_PROTO_THREAD`), number of untyped items to 1, typed items to 0, and no flags set. The client may want to pass a label that identifies itself, as many clients can use the server. In this context, the identifier might also be passed via the message registers, but the label is the proper place for this, as it gets a special treatment by the kernel for IPC gates (covered by the example below). The client then calls `l4_ipc()` with the tag, label and flags indicating it wants a send phase and a receive phase (as it wants a result back). The target is the capability index referring to a capability for the `L4::Thread` kernel object of the server.

To be able to receive an IPC message, the server has set up a UTCB of its own and called `l4_ipc()` with flags indicating it only wants to enter a receive phase and it accepts IPCs from any partner. This is called an **open wait**. (The alternative would be to specify a capability index referring to a `L4::Thread` capability to exclusively receive from.)

Both system calls (the send IPC initiated by the client and the receive IPC initiated by the server) may be seen by the kernel in any order but the IPC will not start before sender and receiver are ready. In that case the kernel will copy the relevant message registers the client specified in the message tag from the client's UTCB to the server's UTCB, in the current example just `MR[0]`. It will then switch the client to the receive phase of its call (which cannot yet be executed) and return the server's call with the message tag and label.

The server inspects the tag for the correct protocol value and number of untyped items passed, inspect the label to decide whether it maybe wants special treatment of this particular client, performs the computation on `MR[0]` and writes the result back to `MR[0]` (or to more words, depending on what exactly it computes). It sets up the tag in the usual way, but probably needs to pass no label, as the client knows who it is talking to.

For the reply, the server makes use the reply capability (see above). Since the client sent the last IPC to the server, the reply capability will point to it. So when the server calls `l4_ipc()` with the computed result in the message registers and using the reply capability as target, the kernel knows to forward this to the client's thread. The kernel copies the message registers from the server's UTCB to the client's UTCB, and the client's `l4_ipc()` system call, which is still stuck in the receive phase, is returned with the tag.

The client looks at the tag and then the message registers for its wanted result and the example is concluded.

3.1.2.2.2 Send Items

IPC between userland threads is also used to transfer typed items: Memory, I/O ports, and objects, all represented as flexpages. Typed items and untyped items can be part of the same IPC. As general rule, the sender specifies what he wants to send, the receiver where and how much it wants to receive, and the kernel checks the required permissions before doing the actual transfer. As written before, this mechanism is synchronous and the receiver cannot be transferred items against its will.

See also

[Flexpages](#)

Suppose a client wants a server to have read only access to a page of its memory. The client sets up a flexpage covering the page and with only the `L4_FPAGE_RO` right set. The server sets up a flexpage of a memory region where it will receive the mapping. This may be larger than one page, suppose for our case four pages, in which case the exact position of the mapping will be resolved by the send base provided by the sender. The client combines the hot spot and some flags into a machine word and writes it to `MR[0]` (see also `l4_map_control()`). At `MR[1]` follows the flexpage it wants to send (see also `l4_fpage()`). The server does almost the same but writes the words to `BR[0]` and `BR[1]`. (The server could also specify a hot spot, but it is currently ignored by the kernel.) The

client specifies 1 typed and 0 untyped items in the message tag. The server writes 0 to `BDR` to specify that the first receive item starts at the first buffer register.

Both client and server initiate their IPC, the client with only a send phase to the server, and the server with an open wait receive phase. The kernel checks the compatibility of the send items and the receive buffers (e.g. that no object capability flexpage is sent to a receive buffer describing a memory mapping flexpage) and updates its internal structures to reflect the change. In our case, the sender's hot spot indicates to which of the four pages that make up the receive buffer the sent page should be mapped. The kernel also translates the typed send item to the server's address space and stores it in the server's UTCB at `MR[0]` and `MR[1]`.

Once the server returns from its syscall, it will have read access to the client's shared page.

3.1.2.3 User Thread to User Object

A common use case for thread to thread communication is when a server implements a number of object interfaces and a client wants to invoke methods on them. For security reasons, the server does not want to hand out its thread capability to everyone it nonetheless wants to serve. It also may not want to allow every client access to everyone of its interfaces. For this purpose, IPC gates implemented by the kernel object `L4::ipc_gate` can be used. An IPC gate can be bound to a thread and forwards IPC to it. In doing so it applies two transformations

1. It sets the label to a predefined value.
2. It changes the rights of transferred items (see `L4_CAP_FPAGE_S`).

For each object of every interface the server implements, it creates an IPC gate and binds it to itself (see `L4::ipc_gate::bind_thread()`). It sets the gate's label to a unique value identifying the object. Then it hands the gate out to clients. Several clients can be handed the same gate and will all end up invoking methods on the same object.

Instead of setting the target as the server's thread kernel object, the client uses the IPC gate's instead. The label the client sends is irrelevant, as the gate will overwrite it with the value the server has set during the bind operation. The server executes an open wait, and the kernel performs the same operation as in the above [example](#) with the transformed IPC finally ending up in the server's thread.

The server checks that the received label refers to one of its objects. It then checks if the protocol value in the message tag matches the interface the object implements. Then it invokes the method specified in `MR[0]` with the rest of the arguments. Finally it returns the results via UTCB and message tag to the reply capability and waits for the next IPC.

3.2 Kernel ABI

This section details the binary representation of the IPC interface of the kernel.

It accompanies the [L4 Inter-Process Communication \(IPC\)](#) section. The details presented here are usually not relevant when developing L4Re applications and can therefore be skipped by many readers.

Note

The kernel ABI is subject to change. Please use the API instead of relying on particular binary representations.

The following notation is used to indicate how particular data fields are used:

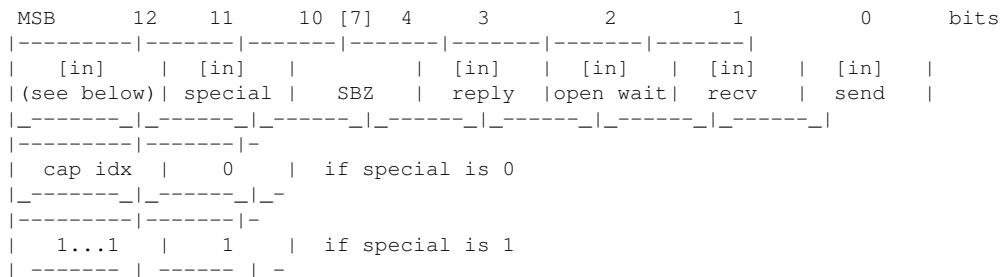
- `[in]`: The kernel reads and interprets this field.
- `[out]`: The kernel writes this field with information provided by the kernel.
- `[cpy]`: The kernel copies this field from sender to receiver (without interpretation if `[in]` is not listed as well).

The above indications may be combined.

3.2.1 Capability selector and flags

See [partner capability selector](#) and [IPC flags](#).

The kernel reads and interprets all the fields ([in]).



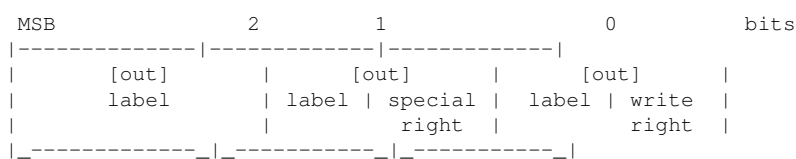
- Bits 0...3 [in]: These bits correspond to the capability selector flags. The individual bits correspond to [L4_SYSF_REPLY](#), [L4_SYSF_OPEN_WAIT](#), [L4_SYSF_RECV](#), [L4_SYSF_SEND](#). Note that not all combinations of those bits are valid; see [L4_SYSF_NONE](#) for an overview.
- Bits 4...10 [in] SBZ: should be zero
- Bit 11 [in] special: Set when using [L4_INVALID_CAP](#), otherwise unset.
- Bits 12...MSB [in]: Capability index if special is unset, otherwise all those bits should be one (see [L4_INVALID_CAP](#), [partner capability selector](#) and [l4_cap_idx_t](#)).

3.2.2 Label

See [IPC label](#).

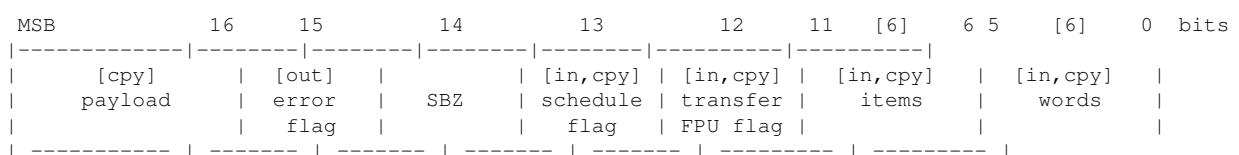
When IPC is sent via a thread capability, the label is copied to the receiver unchanged ([cpy]).

When IPC is sent via an IPC gate, the sent label is ignored and the kernel provides the bitwise OR (|) of the IPC gate label and the sender's write and special permissions (see [L4_CAP_FPAGE_W](#) and [L4_CAP_FPAGE_S](#)) of the used capability ([out]):



3.2.3 Message tag

See [IPC message tag](#). Note that, for a message tag returned by the kernel, if the error flag is set, all other contents of the message tag is undefined.



- Bits 0...5 [in,cpy] `words`: Number of (untyped) message words in the UTCB's message registers. See [l4_msgtag_words\(\)](#) and [l4_msgtag_t::words\(\)](#).
- Bits 6...11 [in,cpy] `items`: Number of typed message items in the UTCB's message registers. See [l4_msgtag_items\(\)](#) and [l4_msgtag_t::items\(\)](#).
- Bit 12 [in,cpy] `transfer` FPU flag: See [L4_MSGTAG_TRANSFER_FPU](#).
- Bit 13 [in,cpy] `schedule` flag: See [L4_MSGTAG_SCHEDULE](#).
- Bit 14 SBZ: should be zero
- Bit 15 [out] `error`: See [L4_MSGTAG_ERROR](#), [l4_msgtag_has_error\(\)](#) and [l4_msgtag_t::has_error\(\)](#).
- Bits 16...MSB [cpy] `payload`: Transferred to receiver unchanged; not interpreted by kernel (unless it is the communication partner). For IPC calls or send-only IPC, this is usually the protocol. For replies, this is usually used for return values and server error signaling. See [l4_msgtag_label\(\)](#) and [l4_msgtag_t::label\(\)](#).

3.2.4 Timeouts

See [IPC timeouts](#) and [l4_timeout_t](#).

The kernel reads and interprets all the fields ([in]).

```

31      [16]      16 15      [16]      0  bits
|-----|-----|
|      [in]      |      [in]      |
| send timeout   | receive timeout |
|_-----|_-----|

```

A timeout has the following format. There are two special timeout values:

- *Zero timeout*: Only bit 10 is set. See [L4_IPC_TIMEOUT_0](#).

```

15  [5]  11  10   9      [10]      0  bits
|-----|-----|-----|
|      0      |  1  |      0      |
|_-----|_-----|_-----|

```

- *Infinite timeout*: All bits are unset. See [L4_IPC_TIMEOUT_NEVER](#).

```

15      [16]      0  bits
|-----|
|      0      |
|_-----|

```

Otherwise, the timeout is either relative or absolute, which is specified by bit 15.

- *Relative timeout*: If bit 15 is unset, the timeout is `mantissa * 2 ^ exponent` micro seconds relative to the current time. The `mantissa` must not be zero:

```

15  14  [5]  10  9      [10]      0  bits
|----|-----|-----|
|  0  | exponent | mantissa ≠ 0 |
|_---|_-----|_-----|

```

- *Absolute timeout*: If bit 15 is set, an absolute timeout is specified in the UTCB's buffer registers starting at `buf reg idx` (the particular number of registers depends on the architecture; see [l4_timeout_s](#)):

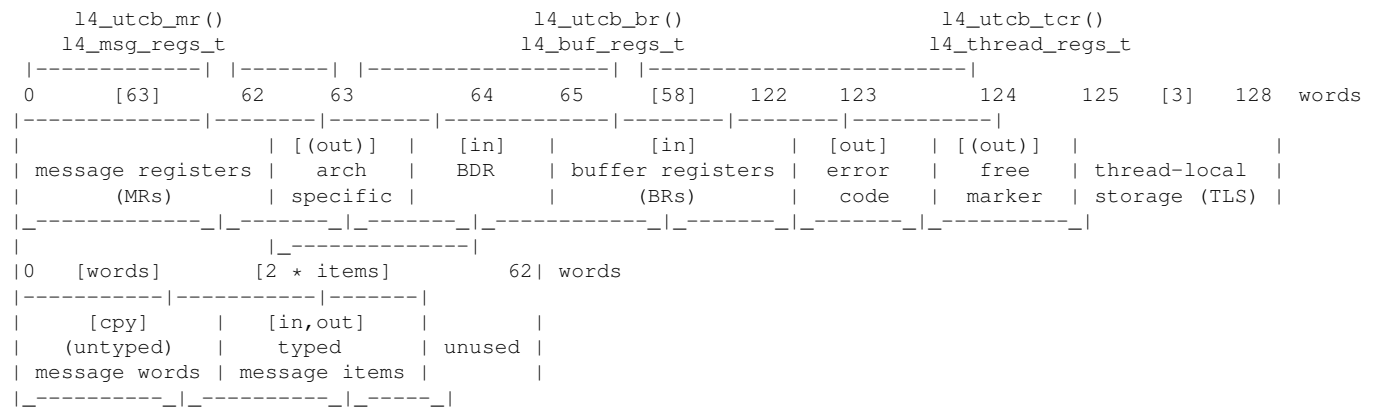
```

15  14      [9]      6 5      [6]      0  bits
|----|-----|-----|
|  1  |      SBZ      | buf reg idx |
|_---|_-----|_-----|

```

3.2.5 User-level thread control block (UTCB)

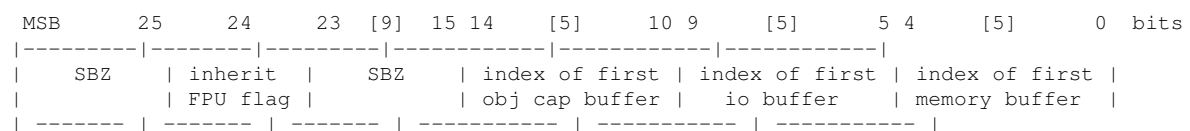
See [User-level thread control block \(UTCB\)](#).



- Words 0...62 MRs: See [IPC Message registers](#) and [l4_utcb_mr\(\)](#). The number of message registers is defined by [L4_UTCB_GENERIC_DATA_SIZE](#). The actually used message registers are defined by `words` and `items` in the [message tag](#). The layout of a typed message item varies depending on being an input or output value, see [typed message items](#).
- Word 63 [(out)]: Depending on the architecture, this word may be used by the kernel to signify the position of a thread's UTCB in memory. See architecture-specific implementation of [l4_utcb\(\)](#). If at all, the kernel writes this word when kernel-user memory is set up as UTCB while binding a thread to a task; see [l4_thread_control_bind\(\)](#), [L4::Thread::Attr::bind\(\)](#).
- Word 64 [in] BDR: See [buffer descriptor register](#).
- Words 65...122 [in] BRs: See [IPC Buffer Registers](#), [receive items](#) and [l4_utcb_br\(\)->br](#). The number of buffer registers is defined by [L4_UTCB_GENERIC_BUFFERS_SIZE](#).
- Word 123 [out] error code: See [IPC Thread Control Registers](#) and [l4_utcb_tcr\(\)->error](#).
- Word 124 [(out)] free marker: Written by the kernel, but not necessarily during IPC. See [IPC Thread Control Registers](#) and [l4_utcb_tcr\(\)->free_marker](#).
- Word 125...128 TLS: Ignored and left untouched by the kernel. See [IPC Thread Control Registers](#) and [l4_utcb_tcr\(\)->user](#).

3.2.5.1 Buffer descriptor register

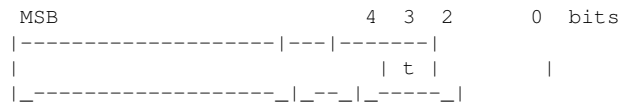
See [IPC Buffer Descriptor Register](#) and [l4_utcb_br\(\)->bdr](#).



3.2.6 Typed message items

The number of words in a typed message item varies depending on the particular kind of item. However, for the first word, the following properties are shared:

- *Void item*: If all bits of the first word of a typed message item are zero, then it is a void item.
- *Non-void item*: The first word of a non-void typed message item has the following binary layout:



Bit 3 (t) is the type bit. If t is set, the item is a map item. Currently, map item is the only supported type. Hence, this bit must be set for all items except for void items.

There are three sub-types of typed message items: *send items*, *receive items*, and *return items*; see [Message Items](#).

Many typed items make use of flexpages, therefore, these are described before the various kinds of typed items. Note that flexpages are also used outside of typed message items, e.g., for [L4::Task::unmap\(\)](#).

3.2.6.1 Flexpages

A flexpage consists of a single word and, except for some special values, describes a range in an address space, see [flex pages](#).

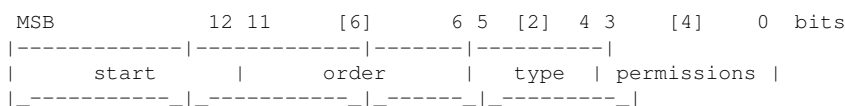
The general layout is defined as follows:



- Bits 4...5 type: See [l4_fpage_type\(\)](#) and [L4_fpage_type](#).

The type [L4_FPAGE_SPECIAL](#) only supports some selected values, which are only supported for selected interfaces; see [L4_FPAGE_SPECIAL](#).

The other types share the same layout:



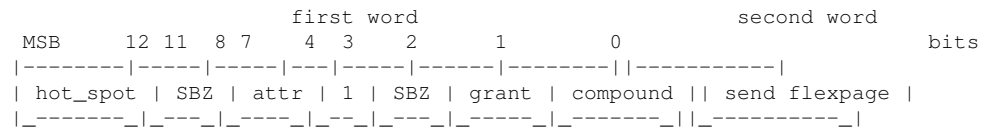
- Bits 0...3 permissions: See [l4_fpage_rights\(\)](#), [L4_fpage_rights](#) (memory space) and [L4_cap_fpage_rights](#) (object space). Should be zero for I/O port space.
- Bits 6...11 order: The \log_2 size of the flexpage. See [l4_fpage_size](#).
- Bits 12...MSB start: The starting page number / I/O port number / capability index of the flexpage. Must be aligned to the flexpage size. See [l4_fpage_page\(\)](#), [l4_fpage_memaddr\(\)](#), [l4_fpage_ioport\(\)](#) and [l4_fpage_obj\(\)](#).

Also see [l4_fpage\(\)](#) (memory space), [l4_iofpage\(\)](#) (I/O port space) and [l4_fpage_obj\(\)](#) (object space).

3.2.6.2 Send items

A send item consists of two words. The second word of a non-void send item is a [flexpage](#). The type of the flexpage determines the interpretation of the `attr` bits in the first word (see below).

If not void, the layout of the first word is defined as follows:



SBZ means “should be zero”.

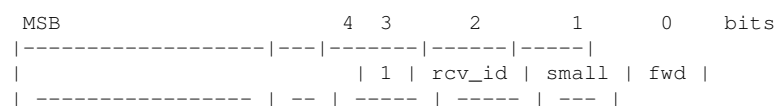
- Bit 0 (compound): Compound bit. See [L4_ITEM_CONT](#) and [L4::lpc::Snd_fpage::is_compound\(\)](#).
- Bit 1 (grant): Grant flag. See [L4_ITEM_MAP](#), [L4_MAP_ITEM_GRANT](#) and [L4::lpc::Snd_fpage::Map_type](#).
- Bits 7..4 (attr): Attributes. See [Attributes and additional permissions for object send items](#) and [l4_fpage_cacheability_opt_t](#), [L4::lpc::Snd_fpage::Cacheopt](#).
- Bits MSB..12 (hot_spot): Send base (also called hot spot). See [L4::lpc::Snd_fpage::snd_base\(\)](#).

For details, see [IPC Message registers](#).

3.2.6.3 Receive items

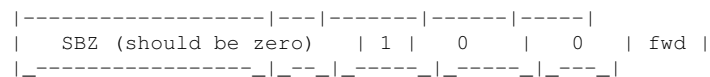
A non-void receive item consists of up to three words.

If not void, the general layout of the first word is defined as follows:

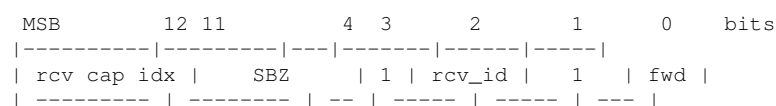


The `small` and `fwd` bits determine the details of the layout of the whole message item.

If `small` is unset, then also `rcv_id` must be unset, and the most significant bits should be zero:



If `small` is set, the most significant bits are layouted as follows:




```

|-----|-----|-----|---|-----|---||-----|
| hot_spot | order | type | 1 |    01    | c ||    undefined    |
|_-----|_-----|_-----|_--|_-----|_--||_-----|

```

10: Used if the receive item's `rcv_id` bit was set and the conditions for transferring an IPC gate label were fulfilled. In that case, no mapping is done for this item and the payload consists of the bitwise OR (|) of the IPC gate label and the write and special permissions (see [L4_CAP_FPAGE_W](#) and [L4_CAP_FPAGE_S](#)) that would have been mapped (also see [L4::lpc::Snd_fpage::id_received\(\)](#)):

```

                                     2 1      0 bits
|-----|-----|-----|---|-----|---||-----|
| hot_spot | order | type | 1 |    10    | c ||    label    | rights |
|_-----|_-----|_-----|_--|_-----|_--||_-----|_-----|

```

11: Used if the receive item's `rcv_id` bit was set and the conditions for transferring the sender's flexpage word were fulfilled. In that case, no mapping is done for this item and the payload is a copy of the sender's flexpage word (also see [L4::lpc::Snd_fpage::local_id_received\(\)](#)):

```

|-----|-----|-----|---|-----|---||-----|
| hot_spot | order | type | 1 |    11    | c ||    send flexpage    |
|_-----|_-----|_-----|_--|_-----|_--||_-----|

```

3.3 Capabilities and Naming

The [L4Re](#) system is a capability based system which uses and offers capabilities to implement fine-grained access control.

Generally, owning a capability means to be allowed to communicate with the object the capability points to. All user-visible kernel objects, such as tasks, threads, and IRQs, can only be accessed through a capability. Please refer to the [Kernel Objects](#) documentation for details. Capabilities are stored in per-task capability tables (the object space) and are referenced by capability selectors or object flexpages. In a simplified view, a capability selector is a natural number indexing into the capability table of the current task.

As a matter of fact, a system designed solely based on capabilities uses so-called 'local names' because each task can only access those objects made available to this task. Other objects are not visible to and accessible by the task.

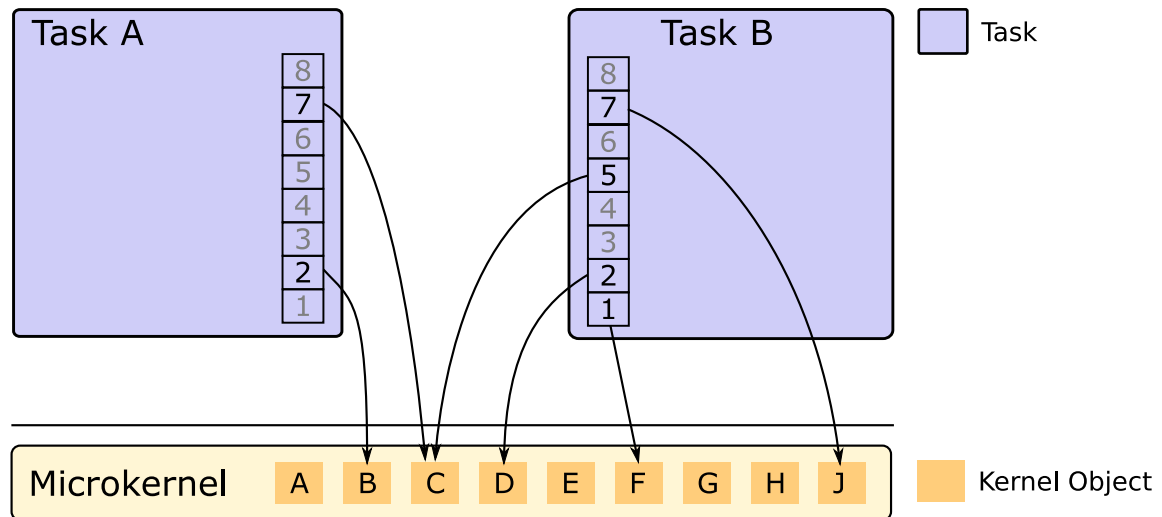


Figure 3.1 Capabilities and Local Naming in L4

So how does an application get access to a service? In general all applications are started with an initial set of available objects. This set of objects is predetermined by the creator of a new application process and granted directly to the new task before starting the first application thread. The application can then use these initial objects to request access to further objects or to transfer capabilities to its own objects to other applications. A central L4Re object for exchanging capabilities at runtime is the name-space object, implementing a store of named capabilities.

From a security perspective, the set of initial capabilities (access rights to objects) completely define the execution environment of an application. Mandatory security policies can be defined by well known properties of the initial objects and carefully handled access rights to them.

3.4 Spaces and Mappings

Each task in the L4Re system has access to two resource spaces (three on IA32) which are maintained by the kernel.

These are the

1. object space,
2. memory space, and
3. IO-port space (only on IA32).

The entities addressed in each space are capabilities to objects, virtual memory pages, and IO ports. The addresses are unsigned integers and the largest valid address depends on which space is referenced, the hardware, and the configuration of the kernel. Although a program can access memory at byte granularity, from the kernel's point of view the address granularity in the memory space is not bytes but pages, as determined by the hardware. The address of a capability is also called its "capability slot".

Flexpages describe a range in any of the spaces that has a power-of-two length and is also aligned to this length. They additionally hold access rights information and further space specific information.

When a resource is present at some address in a task's corresponding resource space, then we say that resource is mapped to that task. For example, a capability to the task's main thread may be mapped to capability slot 5, or the first page of the code segment a thread executes is mapped to virtual memory page 12345. However, there need not be any resource mapped to an address.

Tasks can exchange resources through a process called "mapping" during IPC and using the [L4::Task::map\(\)](#) method. The sending task specifies a send flexpage and the receiving task a receive flexpage. The resources mapped to the send flexpage will then be mapped to the receive flexpage by the kernel.

Memory mappings and IO port mappings are hierarchical: If a resource of such a type is subject of a map operation, the received mapping is a child mapping of the corresponding mapping in the sending task (parent mapping). The kernel usually respects the relationship between these two mappings (granting is an exception; see below): If rights of a parent mapping are revoked using [L4::Task::unmap\(\)](#), these rights are also removed from its child mappings. Also, if a mapping is completely removed (via [L4::Task::unmap\(\)](#) or by mapping something else at its place), then also all child mappings are removed. In contrast, revoking rights of a child mapping leaves the rights of its parent mapping untouched.

The mapping of a resource can be performed as *grant* operation (see [L4_MAP_ITEM_GRANT](#)): Such an operation includes the removal of all involved mappings from the send flexpage (basically a move operation). While with a map operation without grant the mapping in the send flexpage remains the parent of all child mappings (including the new child mapping in the receive flexpage), a grant operation moves the mappings covered by the send flexpage to the corresponding addresses from the receive flexpage while leaving the parent/child relationship of the moved mappings with other mappings untouched.

During a map operation at most the access rights of the source mapping(s) can be transferred but no additional rights can be added. So only rights that are present in the source mapping and that are specified in the send item/flexpage are transferred. This also holds for grant mappings, however, rights revocation is *not* guaranteed to be applied to descendant mappings in case of grant.

There are cases where a grant operation is not or cannot be performed as requested; see [L4_MAP_ITEM_GRANT](#) for details.

Object capabilities are not hierarchical – they have no children. The result of the map operation on an object capability is a copy of that capability in the object space of the destination task.

3.5 Initial Environment and Application Bootstrapping

New applications that are started by a loader conforming to [L4Re](#) get provided an [Initial Environment](#).

This environment comprises a set of capabilities to initial [L4Re](#) objects that are required to bootstrap and run this application. These capabilities include:

- A capability to an initial memory allocator for obtaining memory in the form of data spaces
- A capability to a factory which can be used to create additional kernel objects
- A capability to a Vcon object for debugging output and maybe input

- A set of named capabilities to application specific objects

During the bootstrapping of the application, the loader establishes data spaces for each individual region in the ELF binary. These include data spaces for the code and data sections, and a data space backed with RAM for the stack of the program's first thread.

One loader implementation is the `moe` root task. Moe usually starts an *init* process that is responsible for coordinating the further boot process. The default *init* process is `ned`, which implements a script-based configuration and startup of other processes. Ned uses Lua (<https://www.lua.org>) as its scripting language, see [Ned Script example](#) for more details.

3.5.1 Configuring an application before startup

The default L4Re init process (Ned) provides a Lua script based configuration of initial capabilities and application startup. Ned itself also has a set of initial objects available that can be used to create the environment for an application. The most important object is a kernel object factory that allows creation of kernel objects such as IPC gates (communication channels), tasks, threads, etc. Ned uses Lua tables (associative arrays) to represent sets of capabilities that shall be granted to application processes.

```
local caps = {
    name = some_capability
}
```

The L4 Lua package in Ned also has support functions to create application tasks, region-map objects, etc. to start an ELF binary in a new task. The package also contains Lua bindings for basic L4Re objects, for example, to generic factory objects, which are used to create kernel objects and also user-level objects provided by user-level servers.

```
L4.default_loader:start({ caps = { some_service = service } }, "rom/program --arg");
```

3.5.2 Connecting clients and servers

In general, a connection between a client and a server is represented by a communication channel (IPC gate) that is available to both of them. You can see the simplest connection between a client and a server in the following example.

```
local loader = L4.default_loader; -- which is Moe
local svc = loader:new_channel(); -- create an IPC gate
loader:start({ caps = { service = svc:svr() } }, "rom/my_server");
loader:start({ caps = { service = svc:m("rw") } }, "rom/my_client");
```

As you can see in the snippet, the first action is to create a new channel (IPC gate) using `loader:new_channel()`. The capability to the gate is stored in the variable `svc`. Then the binary `my_server` is started in a new task, and full (`:svr()`) access to the IPC gate is granted to the server as initial object. The gate is accessible to the server application as "service" in the set of its initial capabilities. Virtually in parallel a second task, running the client application, is started and also given access to the IPC gate with less rights (`:m("rw")`, note, this is essential). The server can now receive messages via the IPC gate and provide some service and the client can call operations on the IPC gate to communicate with the server.

Services that keep client specific state need to implement per-client server objects. Usually it is the responsibility of some authority (e.g., Ned) to request such an object from the service via a generic factory object that the service provides initially.

```
local loader = L4.default_loader; -- which is Moe
local svc = loader:new_channel():m("rws"); -- create an IPC gate with rws rights
loader:start({ caps = { service = svc:svr() } }, "rom/my-service");
loader:start({ caps = { foo_service = svc:create(object_to_create, "param") } }, "rom/client");
```

This example is quite similar to the first one, however, the difference is that Ned itself calls the `create` method on the factory object provided by the server and passes the returned capability of that request as "foo_service" to the client process.

Note

The `svc:create(...)` call blocks on the server. This means the script execution blocks until the my-service application handles the create request.

3.6 Memory management - Data Spaces and the Region Map

3.6.1 User-level paging

Memory management in L4-based systems is done by user-level applications, the role is usually called *pager*. Tasks can give other tasks full or restricted access rights to parts of their own memory. The kernel offers means to give access to memory in a secure way, often referred to as *memory mapping*.

The mapping mechanism allows one task to resolve page faults of another: A thread usually has a pager assigned to it. When the thread causes a page fault, the kernel sends an IPC message to the pager with information about the page fault. The pager answers this IPC by either providing a backing page, or with an error. The kernel will map the backing page into the address space of the faulting thread's task.

These mechanisms can be used to construct a memory and paging hierarchy among tasks. The root of the hierarchy is `sigma0`, which initially gets all system resources and hands them out once on a first-come-first-served basis. Memory resources can be mapped between tasks at a page-size granularity. This size is predetermined by the CPU's memory management unit and is commonly set to 4 kB.

3.6.1.1 Data spaces

A data space is the [L4Re](#) abstraction for objects which may be accessed in a memory mapped fashion (i.e., using normal memory read and write instructions). Examples include the sections of a binary which the loader attaches to the application's address space, files in the ROM or on disk provided by a file server, the registers of memory-mapped devices and anonymous memory such as the heap or the stack.

Anonymous memory data spaces in particular (but in general all data spaces except memory mapped IO) can either be constructed entirely from a portion of the RAM or the current working set may be multiplexed on some portion of the RAM. In the first case it is possible to eagerly insert all pages (more precisely page-frame capabilities) into the application's address space such that no further page faults occur when this data space is accessed. In general, however, only the pages for some portion are provided and further pages are inserted by the pager as a result of page faults.

3.6.1.2 Virtual Memory Handling

The virtual memory of each task is constructed from data spaces backing virtual memory regions (VMRs). The management of the VMRs is provided by an object called *region map*. A dedicated region-map object is associated with each task; it allows attaching and detaching data spaces to an address space as well as reserving areas of virtual memory. Since the region-map object possesses all knowledge about the virtual memory layout of a task, it also serves as an application's default pager.

3.6.1.3 Memory Allocation

Operating systems commonly use anonymous memory for implementing dynamic memory allocation (e.g., using `malloc` or `new`). In an L4Re-based system, each task gets assigned a memory allocator providing anonymous memory using data spaces.

See also

[L4Re::Dataspace](#) and [L4Re::Rm](#).

3.7 Program Input and Output

The initial environment provides a Vcon capability used as the standard input/output stream.

Output is usually connected to the parent of the program and displayed as debugging output. The standard output is also used as a back end to the C-style printf functions and the C++ streams.

Vcon services are implemented in Moe and the loader as well as by the [L4Re](#) Microkernel and connected either to the serial line or to the screen if available.

See also

[Virtual Console](#)

3.8 Initial Memory Allocator and Factory

The purpose of the memory allocator and of the factory is to provide the application with the means to allocate memory (in the form of data spaces) and kernel objects respectively.

An initial memory allocator and an initial factory are accessible via the initial [L4Re](#) environment.

See also

[L4Re::Mem_alloc](#)

The factory is a kernel object that provides the ability to create new kernel objects dynamically. A factory imposes a resource limit for kernel memory, and is thus a means to prevent denial of service attacks on kernel resources. A factory can also be used to create new factory objects.

See also

[Factory](#)

3.9 Application and Server Building Blocks

So far we have discussed the environment of applications in which a single thread runs and which may invoke services provided through their initial objects.

In the following we describe some building blocks to extend the application in various dimensions and to eventually implement a server which implements user-level objects that may in turn be accessed by other applications and servers.

3.9.1 Creating Additional Application Threads

To create application threads, one must allocate a stack on which this thread may execute, create a thread kernel object and setup the information required at startup time (instruction pointer, stack pointer, etc.). In [L4Re](#) this functionality is encapsulated in the pthread library.

3.9.2 Providing a Service

In capability systems, services are typically provided by transferring a capability to those applications that are authorised to access the object to which the capability refers to.

Let us discuss an example to illustrate how two parties can communicate with each other: Assume a simple file server, which implements an interface for accessing individual files: `read(pos, buf, length)` and `write(pos, data, length)`.

L4Re provides support for building servers based on the class `L4::Server_object`. `L4::Server_object` provides an abstract interface to be used with the `L4::Server` class. Specific server objects such as, in our case, files inherit from `L4::Server_object`. Let us call this class `File_object`. When invoked upon receiving a message, the `L4::Server` will automatically identify the corresponding server object based on the capability that has been provided to its clients and invoke this object's `dispatch` function with the incoming message as a parameter. Based on this message, the server must then decide which of the protocols it implements was invoked (if any). Usually, it will evaluate a protocol specific opcode that clients are required to transmit as one of the first words in the message. For example, assume our server assigns the following opcodes: `Read = 0` and `Write = 1`. The `dispatch` function calls the corresponding server function (i.e., `File_object::read()` or `File_object::write()`), which will in turn parse additional parameters given to the function. In our case, this would be the position and the amount of data to be read or written. In case the write function was called the server will now update the contents of the file with the data supplied. In case of a read it will store the requested part of the file in the message buffer. A reply to the client finishes the client request.

3.10 Pthread Support

L4Re supports the standard pthread library functionality.

Therefore L4Re itself does not contain any documentation for pthreads itself. Please refer to the standard pthread documentation instead.

The L4Re specific parts will be described herein.

- Include pthread-l4.h header file:

```
#include <pthread-l4.h>
```

- Return the local thread capability of a pthread thread:

Use `pthread_l4_cap(pthread_t t)` to get the capability index of the pthread `t`.

For example:

```
pthread_l4_cap(pthread_self());
```

- Setting the L4 priority of an L4 thread works with a special scheduling policy (other policies do not affect the L4 thread priority):

```
pthread_t t;
pthread_attr_t a;
struct sched_param sp;

pthread_attr_init(&a);
sp.sched_priority = l4_priority;
pthread_attr_setschedpolicy(&a, SCHED_L4);
pthread_attr_setschedparam(&a, &sp);
pthread_attr_setinheritsched(&a, PTHREAD_EXPLICIT_SCHED);

if (pthread_create(&t, &a, pthread_func, NULL))
    // failure...

pthread_attr_destroy(&a);
```

- You can prevent your pthread from running immediately after the call to `pthread_create(..)` by adding `PTHREAD_L4_ATTR_NO_START` to the `create_flags` of the pthread attributes. To finally start the thread you need to call `scheduler()->run_thread()` passing the capability of the pthread and scheduling parameters.

```
pthread_t t;
pthread_attr_t attr;

pthread_attr_init(&attr);
attr.create_flags |= PTHREAD_L4_ATTR_NO_START;

if (pthread_create(&t, &attr, pthread_func, nullptr))
    // failure...

pthread_attr_destroy(&attr);

// do stuff

auto ret = L4Re::Env::env()->scheduler()->run_thread(pthread_l4_cap(t),
                                                    l4_sched_param(2));
if (l4_error(ret))
    // failure...
```

Constraints on pthread_t, user-land capability slot, and kernel thread-object

- `pthread_l4_cap()` is guaranteed to return the valid capability slot of the pthread (A) until `pthread_join()` or `pthread_detach()` is invoked on (A)'s `pthread_t`.
- `pthread_l4_cap()` exposes internal state of the pthread management, take the necessary precautions as you would for any shared data in concurrent environments. If you use `pthread_l4_cap()` guarding against concurrency issues is your duty.
- There is no guarantee that a valid capability slot points to a present capability.

- **Example**

It is possible to obtain a valid thread capability slot and for `l4_task_cap_valid()` to return the capability as not present. The following example showcases a possible sequence of events.

```
// Assume: void some_func(void *)
pthread_t pthread = nullptr;
pthread_create(&pthread, nullptr, some_func, nullptr);

// pthread running some_func()
l4_cap_idx_t cap_idx = pthread_l4_cap(pthread);
l4_is_valid_cap(cap_idx); // --> true

long valid = l4_task_cap_valid(L4RE_THIS_TASK_CAP, cap_idx).label();
// valid == 1 --> capability object is present (refers to a kernel object).

// some_func() exits

cap_idx = pthread_l4_cap(pthread);
l4_is_valid_cap(cap_idx); // --> true

valid = l4_task_cap_valid(L4RE_THIS_TASK_CAP, cap_idx).label();
// valid == 0 --> capability object is not present (refers to no kernel object).

pthread_join(pthread, nullptr); // invalidates the cap slot and frees
                                // the pthread's data structures

// using cap_idx here is undefined behavior.
```

3.11 Interface Definition Language

An interface definition in L4Re is normally declared as a class derived from `L4::Kobject_t`.

For example, the [simple calculator example](#) declares its interface like that:

```
struct Calc : L4::Kobject_t<Calc, L4::Kobject>
{
```



```

L4_INLINE_RPC(long, sub, (l4_uint32_t a, l4_uint32_t b, l4_uint32_t *res));
L4_INLINE_RPC(long, neg, (l4_uint32_t a, l4_uint32_t *res));

using Rpc = L4::Typeid::Rpc<sub_t, neg_t>;
};

```

The signature of each function is first declared using one of the RPC macros (see below) and then all the functions need to be listed in the `Rpc` type.

Clients invoke these functions with the name given to the RPC macros, `sub` and `neg` above. Servers implement them by defining functions with an `op_` prepended, `op_sub` and `op_neg`. The types of the parameters in the macro definition, on the server side, and on the client side are not the same. The following section describes how they are related to each other.

3.11.1 Parameter types for RPC

Generally all value parameters, const reference parameters, and const pointer parameters to an RPC interface are considered as input parameters for the RPC and are transmitted from the client to the server.

Note

This means that `char const *` is treated as an input `char` and not as a zero terminated string value, for strings see `L4::lpc::String<>`.

Parameters that are non-const references or non-const pointers are treated as output parameters going from the server to the client.

There are special data types that appear on only one side (client or server) when used, see the following table for details.

```
L4_RPC(l4_ret_t, test, (int arg1, char const *arg2, unsigned *ret1));
```

The example shows the declaration of a method called `test` with `l4_ret_t` as return type, `arg1` is an `int` input, `arg2` a `char` input, and `ret1` an unsigned output parameter.

| Type | Direction | Client-Type | Server-Type |
|---|-----------|------------------------------------|--|
| <code>T</code> | Input | <code>T</code> | <code>T</code> |
| <code>T const &</code> | Input | <code>T const &</code> | <code>T const &</code> |
| <code>T const *</code> | Input | <code>T const *</code> | <code>T const &</code> |
| <code>T &</code> | Output | <code>T &</code> | <code>T &</code> |
| <code>T *</code> | Output | <code>T *</code> | <code>T &</code> |
| <code>L4::Ipc::In_out<T &></code> | In/Out | <code>T &</code> | <code>T &</code> |
| <code>L4::Ipc::In_out<T *></code> | In/Out | <code>T *</code> | <code>T &</code> |
| <code>L4::Ipc::Cap<T></code> | Input | <code>L4::Ipc::Cap<T></code> | <code>L4::Ipc::Snd_fpage</code> |
| <code>L4::Ipc::Out<L4::Cap<T>></code> | Output | <code>L4::Cap<T></code> | <code>L4::Ipc::Cap<T> &</code> |
| <code>L4::Ipc::Rcv_fpage</code> | Input | <code>L4::Ipc::Rcv_fpage</code> | <code>void</code> |
| <code>L4::Ipc::Small_buf</code> | Input | <code>L4::Ipc::Small_buf</code> | <code>void</code> |

Array types can be used to transmit arrays of variable length. They can either be stored in a client-provided buffer (`L4::lpc::Array`), copied into a server-provided buffer (`L4::lpc::Array_in_buf`) or directly read and written into the UTCB (`L4::lpc::Array_ref`).

Constraints on `L4::lpc::Array_ref`:

- the start position of this array type needs to be known in advance.
- it must be the last parameter of a message.
- the size of the array type is not transmitted to the server, only the client side knows the intended size of the input array. The server assumes the rest of the UTCB as the actual array. Different server-side behavior must be steered otherwise, e.g. through another parameter.

| Type | Direction | Client-Type | Server-Type |
|--|-----------|--|---|
| <code>L4::Ipc::Array<const T></code> | Input | <code>L4::Ipc::Array<const T></code> | <code>L4::Ipc::Array_ref<const T></code> |
| <code>L4::Ipc::Array<const T></code> | Input | <code>L4::Ipc::Array<const T></code> | <code>L4::Ipc::Array_in_buf<const T></code> |
| <code>L4::Ipc::Array<T> &</code> | Output | <code>L4::Ipc::Array<T> &</code> | <code>L4::Ipc::Array_ref<T> &</code> |
| <code>L4::Ipc::Array_ref<T> &</code> | Output | <code>L4::Ipc::Array_ref<T> &</code> | <code>L4::Ipc::Array_ref<T> &</code> |

Finally, there are some optional types where the sender can choose if the parameter should be included in the message. These types are for the implementation of some legacy message formats and should generally not be needed for the definition of ordinary interfaces.

| Type | Direction | Client-Type | Server-Type |
|--|-----------|---|--|
| <code>L4::Ipc::Opt<T></code> | Input | <code>L4::Ipc::Opt<T></code> | <code>T</code> |
| <code>L4::Ipc::Opt<const T*></code> | Input | <code>L4::Ipc::Opt<const T*></code> | <code>T</code> |
| <code>L4::Ipc::Opt<T &></code> | Output | <code>T &</code> | <code>L4::Ipc::Opt<T> &</code> |
| <code>L4::Ipc::Opt<T *></code> | Output | <code>T *</code> | <code>L4::Ipc::Opt<T> &</code> |
| <code>L4::Ipc::Opt<Array↔_ref<T> &></code> | Output | <code>Array_ref<T> &</code> | <code>L4::Ipc::Opt<Array↔_ref<T>> &</code> |

3.11.2 Server Side Interface

The server side function signature for the Calc example above is

```
long op_sub(Calc::Rights, 14_uint32_t a, 14_uint32_t b, 14_uint32_t &res);
```

The first rights parameter is a bitfield encoding the rights the client has on the used capability in the lower two bits. Currently, the W-right and S-right are supported. The rest of the Rights-bitfield is reserved.

The second and third arguments are input parameters as can be deduced from the tables above. The fourth parameter is an output parameter, delivered to the client with the return value of type long.

3.11.3 RPC Return Types

On the server side, the return type of an RPC handling function is always `14_ret_t`. The return value is transmitted via the label field in `14_msgtag_t` and is therefore restricted to its length. Per convention, a negative return value is interpreted as an error condition. If the return value is negative, output parameters are not transmitted back to the client.

Attention

The client must never rely on the content of output parameters when the return value is negative.

On the client-side, the return value of the RPC is set as defined in the RPC macro. If `l4_msgtag_t` is given, then the client has access to the full message tag, otherwise the return type should be `l4_ret_t`. Note that the client might not only receive the server return value in response but also an IPC error code.

3.11.4 RPC Method Declaration

RPC member functions can be declared using one of the following C++ macros.

For inline RPC stubs, where the RPC stub code itself is `inline`:

- `L4_INLINE_RPC(res, name, (args...), flags)`
Define an inline RPC call (type and callable).
- `L4_INLINE_RPC_OP(op, res, name, (args...), flags)`
Define an inline RPC call with specific opcode (type and callable).
- `L4_INLINE_RPC_NF(res, name, (args...), flags)`
Define an inline RPC call type (the type only, no callable).
- `L4_INLINE_RPC_NF_OP(opcode, Ret_type, func_name, (args...), flags)`
Define an inline RPC call type with specific opcode (the type only, no callable).

For external RPC stubs, where the RPC stub code must be defined in a separate compile unit (usually a `.cc` file):

- `L4_RPC(Ret_type, func_name, (args...), flags)`
Define an RPC call (type and callable).
- `L4_RPC_OP(opcode, Ret_type, func_name, (args...), flags)`
Define an RPC call with specific opcode (type and callable).
- `L4_RPC_NF(Ret_type, func_name, (args...), flags)`
Define an RPC call type (the type only, no callable).
- `L4_RPC_NF_OP(opcode, Ret_type, func_name, (args...), flags)`
Define an RPC call type with specific opcode (the type only, no callable).

To generate the implementation of an external RPC stub:

- `L4_RPC_DEF(class_name::func_name)`
Generate the definition of an RPC stub.

The NF versions of the macro generally do not generate a callable member function named `<name>` but do only generate the type `<name>_t`. This data type can be used to call the RPC stub explicitly using `<name>_t::call(L4::Cap<Iface_class> cap, args...)`.

3.12 L4Re Build System

L4Re uses a custom make-based build system, often simply referred to as *BID*.

This section explains how to use BID when writing applications and libraries for L4Re.

3.12.1 Building L4Re

Setting up the Build Directory

L4Re must be built out-of-source. Therefore the first mandatory step is creating and populating a build directory. From the root of the L4Re source tree run

```
make B=<builddir>
```

Other targets that can be executed in the source directory are

update

Update the source directory from svn. Only makes sense when you have downloaded L4Re from the official subversion repository.

help

Show a short help with the most important targets.

Invoking Make

Once the build directory is set up, BID make can be invoked in one of two ways:

1. Go to the build directory and invoke make without special options.
2. Go to a source directory with a BID make file and invoke `make O=<builddir> . . .`

The default target builds the source (as you would expect), other targets that are available in build mode are

cleanfast

Quickly cleans the build directory by removing all subdirectories that contain generated files. The configuration will remain untouched.

clean

Remove generated files. Slower than `make cleanfast` but can be used on selected packages. Use `S= . . .` to select the target package.

In addition to these targets, there are a number of targets to generate images which are explained elsewhere.

3.12.2 Writing BID Make Files

The BID build system exports different roles that define what should be done in the subdirectory. So a BID make file essentially consists of defining the role and a number of role-dependent make variables. The basic layout should look like this:

```
PKGDIR  ?= <path to package's root directory> # e.g., '.' or '..'
L4DIR   ?= <path to L4Re source directory>    # e.g. '$(PKGDIR)/../..'

<various definitions>

include $(L4DIR)/mk/<role>.mk
```

PKGDIR in the first line defines the root directory of the current package. L4DIR in the next line must be pointed to the root of the [L4Re](#) source tree the package should be built against. After this custom variable definitions for the role follow. In the final line of the file, the make file with the role-specific rules must be sourced.

The following roles are currently defined:

- project.mk - Sub-project Role
- subdir.mk - Directory Role
- [prog.mk - Application Role](#)
- lib.mk - Library Role
- [include.mk - Header File Role](#)
- doc.mk - Documentation Role
- [test.mk - Test Application Role](#)
- idl.mk - IDL File Role (currently unused)
- runux.mk - Tests in FiascoUX Role

BID-global Variables

This section lists variables that configure how the BID build system behaves. They are applicable for all roles.

| Variable | Description |
|----------|-------------------------|
| CC | C compiler for target |
| CXX | C++ compiler for target |
| HOST_CC | C compiler for host |
| HOST_CXX | C++ compiler for host |

3.12.3 prog.mk - Application Role

The prog role is used to build executable programs.

General Configuration Variables

The following variables can only be set globally for the Makefile:

MODE

Kind of target to build for. The following values are possible:

- `static` - build a statically linked binary (default)
- `shared` - build a dynamically linked binary
- `l4linux` - build a binary for running on L4Linux on the target platform
- `host` - build for host system
- `targetsys` - build a binary for the target platform with the compiler's default settings

SYSTEMS

List of architectures the target can be built for. The entries must be space-separated entries either naming an architecture (e.g. `amd64`) or an architecture and ABI (e.g. `arm-l4f`). When not defined, the target will be built for all possible platforms.

TARGET

Name or names of the binaries to compile. This variable may also be postfixed with a specific architecture.

Target-specific Configuration Variables

The following variables may either be used with or without a description suffix. Without suffix they will be used for all operations. With a specific description their use is restricted to a subset. These specifications include a target file and the architecture, both optional but in this order, separated by underscores. The specific variables will be used in addition to the more general ones.

`SRC_C / SRC_CC / SRC_F / SRC_S`

`.c, .cc, .f90, .S` source files.

`REQUIRES_LIBS`

List of libraries the binary depends on. This works only with libraries that export a `pkg_config` configuration file. Automatically adds any required include and link options.

DEPENDS_PKGS

List of packages this binary depends on. If one these packages is missing then building of the binary will be skipped.

CPPFLAGS / CFLAGS / CXXFLAGS / FFLAGS / ASFLAGS

Options for the C preprocessor, C compiler, C++ compiler, Fortran compiler and assembler. When used with suffix, the referred element is the source file, not the target file.

LDFLAGS

Options for the linker ld.

LIBS

Additional libraries to link against (with -l).

PRIVATE_LIBDIR

Additional directories to search for libraries.

CRT0 / CRTN

(expert use only) Files containing custom startup and finish code.

LDSCRIPT

(expert use only) Custom link script to use.

3.12.4 include.mk - Header File Role

The header file role is responsible for installing header file at the appropriate location.

The following variables can be used for customizing the process:

INCSRC_DIR

Source directory where the headers can be found. Default is the directory where the Makefile resides.

TARGET

List of header files to install. If left undefined, then INCSRC_DIR will be scanned for files with suffix .h or .i.

Supports the specification of special filenames to allow for different source and target filenames to be installed. The syntax is TARGET<SRC, where a filename including the path of SRC is installed as TARGET. An example is

```
libfoo.h<contrib/libfoo_linux.h
```

which installs the header from the contrib directory under the name without that contrib directory and without the platform specific suffix.

EXTRA_TARGET

When TARGET is undefined, then add these files to the headers found by scanning the source directory. Has no effect if TARGET has been defined.

The filenames specified allow for the same rule specifications as supported by TARGET.

CONTRIB_HEADERS

When set, the headers will be installed in \${BUILDDIR}/include/contrib/\${PKGNAME} rather than \${BUILDDIR}/include/l4/\${PKGNAME}.

INSTALL_INC_PREFIX

Base directory where to install the headers. Overwrites CONTRIB_HEADERS. The headers will then be found under \${BUILDDIR}/include/\${INSTALL_INC_PREFIX}.

PC_FILENAME

When set, a pkg_config configuration file is created with the given name.

3.12.5 test.mk - Test Application Role

The test role is very similar to the application role, it also builds an executable binary.

The difference is that it also builds for each target a test script that executes the test target either on the host (MODE=host) or a target platform (currently only qemu).

The role accepts all make variables that are accepted by the application role. The only difference is that the `TARGET` variable is not required. If it is missing, the source directory will be scanned for source files that fit the pattern `test_*.c[c]` and create one target for each of them.

Note

It is possible to still use `SRC_C[C]` when targets are determined automatically. In that case the specified sources will be used in addition* to the main `test_*.c[c]` source.

In addition to the variables above, there are a number of variables that control how the test is executed. All these variables may be used as a global variable that applies to all test or, if the target name is added as a suffix, set for a specific target only.

TEST_TARGET

Name of binary containing the test (default: same as `TARGET`).

TARGET_\$ (ARCH)

When `TARGET` is undefined, these targets are added to the list of targets for the specified architecture. For all targets `SRC_C[C]` files must be defined separately.

TEST_KERNEL_ARGS

Arguments to append to the kernel command line. These are also appended when specifying custom ones via a `.t`-file's `-f` parameter or when using `-d`.

TEST_EXPECTED

File containing expected output. By default the variable is empty, which means the test binary is expected to produce TAP test output, that can be directly processed. When the `TEST_TAP_PLUGINS` variable is given, `TEST_EXPECTED` is ignored.

TEST_EXPECTED_REPEAT

Number of times the expected output should be repeated, by default 1. When set to 0 then output is expected to repeat forever. This is particularly useful to make sure that stress tests that are meant to run in an endless loop are still alive. Note that such endless tests can only be run by directly executing the test script. They will be skipped when run in a test harness like `prove`.

`TEST_TAP_PLUGINS`

Specify the plugins that are used to process the output of the test run. The syntax is of the values is:

`plugin1:arg1=a,arg2=b plugin2:arg=foo`

Multiple plugins separated by a space are loaded in order. Spaces are not allowed inside a plugin specification. One or more arguments are optionally passed to the plugin separated by commas and delimited by a colon.

If the variable is not specified the plugins for `TAPOutput` and `OutputMatching` (depending on the `TEST_EXPECTED` variable) are automatically loaded.

For the supported plugins and their options please refer to their in-line documentation in `tool/lib/L4/TapWrapper/Plugin/`. The plugin name corresponds to the file stem name in that directory.

`TEST_TIMEOUT`

Non-standard timeout after which the test run is aborted (useful for tests involving sleep).

`NED_CFG`

LUA configuration file for startup to give to Ned

`REQUIRED_MODULES`

Additional modules needed to run the test. By adding `[opts]` to the name of a module you can add module options that are reflected in the generated `modules.list`.

`BOOTSTRAP_ARGS`

Additional parameters to supply to bootstrap.

`QEMU_ARGS`

Additional parameters to supply to QEMU.

`MOE_ARGS`

Additional parameters to supply to moe.

`TEST_ARGS`

Additional arguments for the `TEST_STARTER` (tapper-wrapper per default).

`TEST_ROOT_TASK`

Alternative root task to be used during a test instead of moe.

`TEST_ROOT_TASK_ARGS`

Arguments passed to `TEST_ROOT_TASK` if `TEST_ROOT_TASK` is different from moe.

`KERNEL_CONF`

Features the [L4Re](#) Microkernel must have been compiled with. A space-separated list of config options as used by Kconfig. `run_test` looks for a `globalconfig.out` file in the same directory as the kernel and checks that all options are enabled. If not, the test is skipped. Has only an effect if the `globalconfig.out` file is present.

`L4RE_CONF`

Features the [L4Re](#) userland must have been compiled with. A space-separated list of config options as used by Kconfig. `run_test` will look for these in the `.kconfig` file in the [L4Re](#) build directory.

`L4LINUX_CONF`

Features the L4Linux kernel must have been compiled with. Similar to `KERNEL_CONF` but checks for a `.config` file in the directory of the L4Linux kernel.

`TEST_SETUP`

Command to execute before the test is run. The test will only be executed if the command returns 0. If the exit code is 69, the test is marked as skipped with the reason provided in the final line of stdout.

`TEST_LOGFILE`

Append output of test execution to the given file unless `TEST_WORKDIR` is given.

`TEST_WORKDIR`

Create logs, temp and other files below the given directory. That directory is taken as base dir for more automatically created subdir levels using the current test path, in order to guarantee conflict-free usage when running many different tests with a common workdir. When `TEST_WORKDIR` is provided then `TEST_LOGFILE` is ignored as it is organized below workdir.

`TEST_TAGS`

List of conditions for tags provided during execution of a test. A tag can be set to 1, set to 0 or be unspecified via `TEST_RUN_TAGS` during execution. Therefore there are 4 possible conditions for a tag that can be specified in `TEST_TAGS`: `tag`, `!tag`, `+tag` and `-tag`. The following table shows the conditions they represent.

| <code>TEST_RUN_TAGS \ TEST_TAGS</code> | <code>tag</code> | <code>!tag</code> | <code>+tag</code> | <code>-tag</code> |
|--|------------------|-------------------|-------------------|-------------------|
| <code>tag</code> or <code>tag=1</code> | y | | y | |
| unspecified | | y | y | |
| <code>tag=0</code> | | y | | y |

Example usage:

The tag `long-running` is used by tests which take a long time and should be skipped by default. These tests are marked with the tag `long-running` unprefixd.

The tag `hardware` is set to 1 at runtime when the tests will run on real hardware. Tests that must not run on real hardware are marked with `!hardware`.

The tag `+impl-def` is used by tests that test implementation details. Due to the nature of this flag we require the "+" prefix to be used, so they are run by default but can be excluded from execution by setting `TEST_RUN_TAGS` to `impl-def=0` at runtime.

If you want to specify multiple tag conditions they need to be separated with a comma.

TEST_PLATFORM_ALLOW and TEST_PLATFORM_DENY

Deny and allow lists of platforms a test is banned from or limited to. If you list platforms in the `TEST_PLATFORM_ALLOW` variable the test will only be run on these listed platforms and will be skipped on any other platform. If you list platforms in the `TEST_PLATFORM_DENY` variable the test will be skipped on the listed platforms and will be run on any other platform. You can only use one of these variables per test, not both. See `mk/platforms/` for the various identifiers.

Example usage:

```
# Do not run this test on the Raspberry Pi platform
TEST_PLATFORM_DENY_test_xyz := rpi

# Only run this test on this test on the RCar3 platform.
TEST_PLATFORM_ALLOW_test_abc := rcar3
```

TAPARCHIVE

Filename for an archive file to store the resulting TAP output.

In addition to compiled tests, it is also possible to create tests where the test binary or script comes from a different source. These tests must be listed in `EXTRA_TARGET` and for each target a custom `TEST_TARGET` must be provided.

Running Tests

The make role creates a test script which can be found in `<builddir>/test/t/<arch>/<api>`. It is possible to organise the tests further in subdirectories below by specifying a `TEST_GROUP`.

To be able to execute the test, a minimal test environment needs to be set up by exporting the following environment variables:

`KERNEL_<arch>, KERNEL`

L4Re Microkernel binary to use. The test runner is able to check if the kernel has all features necessary for the test and skip tests accordingly. In order for this to work, the `globalconfig.out` config file from the build directory needs to be available in the same directory as the kernel.

`L4LX_KERNEL_<arch>, L4LX_KERNEL`

L4Linux binary to use. This is only required to run tests in `mode=l4linux`. If no L4Linux kernel is set then these tests will simply be skipped. The test runner is also able to check if the kernel has all features compiled in that are required to run the test successfully (see make variable `L4LINUX_CONF` above). For this to work, the `.config` configuration file from the build directory needs to be available in the same directory as the kernel.

`LINUX_RAMDISK_<arch>, LINUX_RAMDISK`

Ramdisk to mount as root in L4Linux. This is only required to run tests in `mode=l4linux`. If not supplied, L4Linux tests will be skipped. The ramdisk must be set up to start the test directly after the initial startup is finished. The name of the test binary is supplied via the kernel command line option `l4re_testprog`. The `tool/test` directory contains an example script `launch-l4linux-test`, which can be copied onto the ramdisk and started by the init script.

TEST_HWCONFIG and TEST_FIASCOCONFIG

Some userland tests rely on external information about the underlying platform and the configuration of the [L4Re](#) Microkernel to decide whether or not to test specific features or to determine which and how much resources are available. Some examples for this are whether or not virtualization is supported by the platform, how many cores the platform has, how many cores the kernel supports or how much memory the platform provides. To convey this information to these tests you can set the two environment variables `TEST_HWCONFIG` and `TEST_FIASCOCONFIG`.

Using `TEST_HWCONFIG` requires a plain text document containing key-value pairs separated by a `=` symbol. On top of that comment lines starting with `#` are supported. Simply create a plain text file such as the following and set `TEST_HWCONFIG` to its absolute path.

```
VIRTUALIZATION=y
MP=y
NUM_CORES=4
MEMORY=2048
```

Using `TEST_FIASCOCONFIG` is easier since it only needs to contain the absolute path of the `globalconfig.out` file in the [L4Re](#) Microkernel's build directory. The build system will then extract the information when a test is started.

When starting a test the build system will read both files and provide their content as a lua table to the test. A lua script can then make decisions based on them. To simplify some decisions the build system merges some information by itself, e.g. virtualization is only available if both the platform and the [L4Re](#) Microkernel support this feature. More details can be obtained from the perl module in `tool/lib/L4/TestEnvLua.pm`.

In addition to these variables, the following BID variables can be overwritten at runtime: `PT` (for the platform type) and `TEST_TIMEOUT`. You may also supply `QEMU_ARGS` and `MOE_ARGS` which will be appended to the parameters specified in the BID test make file.

Once the environment is set up, the tests can be run either by simply executing all of them from the build directory with

```
make test
```

or executing them directly, like

```
test/t/amd64_amdfam10/14f/14re-core/moe/test_namespace.t
```

or running one or more tests through the test harness `prove`, like

```
prove test/t/amd64_amdfam10/14f/14re-core/moe/test_namespace.t
prove -r test/t/amd64_amdfam10/14f/14re-core/
prove -rv test/t/amd64_amdfam10/14f/14re-core/
```

`TEST_TAGS` allow for a way to include or exclude whole groups of tests during execution, primarily with `prove`. You can specify which tests to run at runtime using one of the following ways:

```
$ test/t/amd64_amdfam10/14f/14re-core/test_one.t --run-tags slow,gtest-shuffle=0
$ test/t/amd64_amdfam10/14f/14re-core/test_one.t -T slow,gtest-shuffle=0
$ prove -r test/t/amd64_amdfam10/14f/14re-core/ : -T slow,gtest-shuffle=0
$ TEST_RUN_TAGS=slow,gtest-shuffle=0 prove -r test/t/amd64_amdfam10/14f/14re-core/
```

For each test tag requirements defined in the corresponding `TEST_TAGS` Makefile variable are tested. If the requirements for tags do not match the test is skipped. The `SKIP` message will provide insight why the test was skipped:

```
$ make test
...
test/t/amd64_amdfam10/test_one.t .... ok
test/t/amd64_amdfam10/test_two.t .... skipped: Running this test requires tag slow to be set to 1.
test/t/amd64_amdfam10/test_three.t .. ok
```

When tags are provided, the tests requiring those tags are now also executed while the tests that forbid them are skipped:

```
$ TEST_RUN_TAGS=slow,gtest-shuffle
$ make test
...
test/t/amd64_amdfam10/test_one.t .... ok
test/t/amd64_amdfam10/test_two.t .... ok
test/t/amd64_amdfam10/test_three.t .. skipped: Running this test requires tag gtest-shuffle to be set to 0 or
```

For further details on how values in `TEST_TAGS` and `TEST_RUN_TAGS` interact, see the help text for `TEST_TAGS`.

Running Tests in External Programs

You can hand-over test execution to an external program by setting the environment variable `EXTERNAL_TEST_STARTER` to the full path of that program:

```
export EXTERNAL_TEST_STARTER=/path/to/external/test-starter
make test
```

`EXTERNAL_TEST_STARTER`

This variable is evaluated by `tool/bin/run_test` (the backend behind `make test`) and contains the full path to the tool which actually starts the test instead of the test itself.

The `EXTERNAL_TEST_STARTER` can be any program instead of the default execution via `make qemu E=make test`. Its output is taken by `run_test` as the test output.

Usually it is just a bridge to prepare the test execution, e.g., it could create the test as image and start that image via a simulator.

Running Tests in a Simulator

Based on above mechanism there is a dedicated external test starter `tool/bin/teststarter-image-telnet.pl` shipped in BID which assumes an image to be started with another program which provides test execution output on a network port.

This can be used to execute tests in a simulator, like this:

```
export EXTERNAL_TEST_STARTER=$L4RE_SRC/tool/bin/teststarter-image-telnet.pl
export SIMULATOR_START=/path/to/configured/simulator-exe
make test
```

After building the image and starting the simulator it contacts the simulator via a network port (sometimes called "telnet" port) to pass-through its execution output as its own output so it gets captured by `run_test` as usual.

The following variables control `teststarter-image-telnet.pl`:

SIMULATOR_START

This points to the full path of the program that actually starts the prepared test image. Most often this is the frontend script of your simulator environment which is pre-configured so that it actually works in the way that `teststarter-image-telnet.pl` expects from the following settings.

SIMULATOR_IMAGETYPE

The image type to be generated via `make $SIMULATOR_IMAGETYPE E=maketest`. Default is `elfimage`.

SIMULATOR_HOST

The simulator will be contacted via socket on that host to read its output. Default is `localhost`.

SIMULATOR_PORT

The simulator will be contacted via socket on that port to read its output. Default is `11111`.

SIMULATOR_START_SLEEP_TIME

After starting the simulator it waits that many seconds before reading from the port. Default is `1` (second).

Running tests without tapper-wrapper

In case you want to replace the `tapper-wrapper` test starter, you can replace the default one by setting the environment variable `TEST_STARTER` to the path of your test starter. Then your test starter can use the same environment which is normally set up for the default starter, which includes environment variables provided by the build system as well as the test itself. Among these are `SEARCHPATH`, `MODE`, `ARCH`, `MOE_CFG`, `MOE_ARGS`, `TEST_TIMEOUT`, `TEST_TARGET`, `TEST_EXPECTED`, `QEMU_ARGS` and many more.

Debugging Tests

The test script is only a thin wrapper that sets up the test environment as it was defined in the make file and then executes two scripts: `tapper-wrapper` and `run_test`.

The main work horse of the two is `tool/bin/run_test`. It collects the necessary files and starts `qemu` to execute the test. This script is always required.

There is then a second script wrapped around the test runner: `tool/bin/tapper-wrapper`. This tool inspects the output of the test runner and reformats it, so that it can be read by tools like `prove`. If the test produces tap output, then the script scans for this output and filters away all the debug output. If `TEST_EXPECTED` was defined, then the script scans the output for the expected lines and prints a suitable TAP message with success or failure. It also makes sure that `qemu` is killed as soon as the test is finished.

There are a number of command-line parameters that allow to quickly change test parameters for debugging purposes. Run the test with `'-help'` for more information about available parameters.

3.13 Kernel Factory

The kernel factory is a kernel object that provides the ability to create new kernel objects dynamically.

The kernel factory enforces a memory quota. This quota defines the maximum amount of kernel memory the factory service can use to construct the requested objects. When the quota is depleted, the factory refuses the creation of new objects.

The quota may be higher than the amount of available kernel memory; ultimately, the amount of available kernel memory is the strict limit for the factory to remain operational.

The kernel factory creates the following kinds of objects:

- [DMA space](#)
- [L4::Factory](#)
- [L4::lpc_gate](#) ([L4_PROTO_NONE](#), [L4_PROTO_KOBJECT](#))
- [L4::Irq](#) ([L4_PROTO_IRQ_SENDER](#))
- [L4::Semaphore](#)
- [L4::Task](#)
- [L4::Thread](#)
- [L4::Vm](#)
- [L4::Vcpu_context](#)

The protocol IDs for objects in this list are given in [L4_msgtag_protocol](#). Kernel objects whose protocol ID is not immediately clear from the documentation of [L4_msgtag_protocol](#) have their protocol IDs stated within parenthesis. As an exception, [L4::lpc_gate](#) can be identified by more than one protocol IDs. The protocol ID shall be used as the second argument for [L4::Factory.create](#)([Cap<void>](#), long, [l4_utcb_t *](#)).

For the C++ interface see [L4::Factory](#), for the C interface see [Factory](#).

3.13.1 Passing parameters for the create stream

[L4::Factory.create\(\)](#) returns a [create stream](#) that allows arguments to be forwarded to the constructor of the object to be created.

Objects that support additional parameters on their creation are presented with a non-empty list of parameters. The parameters are listed in the order they should be provided to a create stream returned by [L4::Factory.create\(\)](#).

- [Dmar_space\(\)](#)
- [L4::Factory\(l4_umword_t\)](#)
 - Argument: factory quota (in bytes).
 - See [L4::Factory.create_factory\(\)](#) for details.
- [L4::lpc_gate\(\)](#)
 - Creates an unbound IPC gate.
 - Alternatively, an IPC gate can be immediately bound to a thread upon creation using [L4::Factory.create_gate\(\)](#).

- [L4::Irq\(\)](#)
- [L4::Semaphore\(\)](#)
- [L4::Task\(l4_fpage_t\)](#)
 - Argument: utcb_area
 - See [L4::Factory.create_task\(\)](#) for details.
- [L4::Thread\(\)](#)
- [L4::Vm\(\)](#)
- [L4::Vcpu_context\(\)](#)

Chapter 4

L4Re Servers

Here you shall find a quick overview over the standard services running on the [L4Re](#) Microkernel.

Sigma0, the Root Pager

Sigma0 is a special server running on [L4](#) because it is responsible of resolving page faults for the root task, the first useful task on [L4Re](#). Sigma0 can be seen as part of the kernel, however it runs in unprivileged mode. To run something useful on the [L4Re](#) Microkernel you usually need to run Sigma0, nevertheless it is possible to replace Sigma0 by a different implementation.

For more details see [Sigma0, the Root-Pager](#)

Moe, the Root Task

Moe is our implementation of the [L4](#) root task that is responsible for bootstrapping the system, and to provide basic resource management services to the applications on top. Therefore Moe provides [L4Re](#) resource management and multiplexing services:

- **Memory** in the form of memory allocators ([L4Re::Mem_alloc](#), [L4::Factory](#)) and data spaces ([L4Re::Dataspace](#))
- **Cpu** in the form of basic scheduler objects ([L4::Scheduler](#))
- **Vcon** multiplexing for debug output (output only)
- **Virtual memory management** for applications, [L4Re::Rm](#)

Moe further provides an implementation of [L4Re](#) name spaces ([L4Re::Namespace](#)), which are for example used to provide a read only directory of all multi-boot modules. In the case of a boot loader, like grub that enables a VESA frame buffer, there is also a single instance of an [L4Re](#) graphics session ([L4Re::Goos](#)).

To start the system Moe starts a single ELF program, the init process. The init process (usually Ned, see the next section) gets access to all resources managed by Moe and to the Sigma0 root pager interface.

For more details see [Moe, the Root-Task](#).

Ned, the Default Init Process

To keep the root task free from complicated scripting engines and to avoid circular dependencies in application startup (that could lead to dead locks) the configuration and startup of the real system is managed by an extra task, the init process.

Ned is such an init process that allows system configuration via Lua scripts.

For more information see [Ned](#).

Io, the Platform and Device Resource Manager

Because all peripheral management in [L4Re](#) is done in user-level applications, there is the need to have a centralized management of the resources belonging to the platform and to peripheral devices.

This is the job of Io. Io provides portable abstractions for iterating and accessing devices and their resources (IRQ's, IO Memory...), as well as delegating access to those resources to other applications (e.g., device drivers).

For more details see [Io, the Io Server](#).

Other Servers

The following additional server package are available on top of the core [L4Re](#) environment.

- [Rtc, the Real-Time Clock Server](#)
is a simple multiplexer for real-time clock hardware on your platform.
- [fb-drv, the Low-Level Graphics Driver](#)
provides low-level access and initialization of various graphics hardware. It has support for running VESA BIOS calls on Intel x86 platforms, as well as support for various ARM display controllers. `fb-drv` provides a single instance of the `L4Re::Goos` interface and can serve as a back end for the Mag server, in particular, if there is no graphics support in the boot loader.
- [Virtio Net P2P, a virtual network point-to-point link](#)
- [Virtio Net Switch, a virtual network switch](#)
- [Uvmm, the virtual machine monitor](#)
- [RTC driver](#)
- [Mag, the GUI Multiplexer](#)
- [Sigma0, the Root-Pager](#)
- [Cons, the Console Multiplexer](#)

4.1 Sigma0, the Root-Pager

Sigma0 is a special [L4](#) server that serves as the origin for mapping memory.

It is started by Fiasco.OC on the system boot and gets full access to all userland RAM and device memory. It functions as the pager (main memory provider) for Moe and as the provider for device memory for Io. Moe and Io are trusted and usually the only applications besides Ned that get a capability for Sigma0. Memory can be requested from Sigma0 directly via an IPC, or indirectly by causing page faults and having them resolved by Sigma0.

4.1.1 Factory

There is only one instance of Sigma0 in an [L4Re](#) system, which is made accessible to Moe via an IPC gate capability. Using this capability, Moe can request Sigma0 to create new communication channels to itself by creating additional IPC gate capabilities. This request is done using the [L4::Factory](#) interface. This is the only kind of object that can be created by the factory in Sigma0.

List of objects that the Sigma0 Factory can create:

- Sigma0 ()
 - Use protocol id [L4_PROTO_SIGMA0](#) for creation
 - No arguments supported

See also

[Sigma0 API](#)

4.2 Moe

Moe is the default root-task implementation for L4Re-based systems.

Moe is the first task which is usually started in L4Re-based systems. The microkernel starts *Moe* as the Root-Task.

4.2.1 Moe objects

Moe provides a default implementation for the basic [L4Re](#) abstractions, such as data spaces ([L4Re::Dataspace](#)), region maps ([L4Re::Rm](#)), memory allocators ([L4::Factory](#), [L4Re::Mem_alloc](#)), name spaces ([L4Re::Namespace](#)) and so on (see [L4Re Interface](#)). These are described in the following subsections.

4.2.1.1 Factory

The factory in Moe is responsible for all kinds of dynamic object allocation.

Moe's factory allows allocation of the following objects:

- [L4Re::Namespace](#)
- [L4Re::Dataspace](#), RAM allocation
- [L4Re::Dma_space](#), memory management for DMA-capable devices
- [L4Re::Rm](#), virtual memory management for application tasks
- [L4::Vcon](#) (output only)
- [L4::Scheduler](#), to provide a restricted priority / CPU range for clients
- [L4::Factory](#), to provide a quota limited allocation for clients

Note

[L4::Scheduler](#) objects can be only created through the user factory provided by Moe to the initial application. Other factory instances cannot create this object.

4.2.1.1.1 Passing parameters to the create stream

`L4::Factory.create()` returns a [create stream](#) that allows arguments to be forwarded to the object creation in Moe.

Namespace

Moe provides a name space conforming to the [L4Re::Namespace](#) interface (see [Namespace](#)). Per default Moe creates a single name space for the [Boot FS](#). That is available as `rom` in the initial objects of the init process.

4.2.1.1.2 Boot FS

The Boot FS subsystem provides access to the files loaded during the platform boot (or available in ROM). These files are either linked into the boot image or loaded via a flexible boot loader, such as GRUB.

The subsystem provides an [L4Re::Namespace](#) object as directory and an [L4Re::Dataspace](#) object for each file.

By default all files are read only and visible in the namespace `rom`. As an option, files can be supplied with the argument `:rw` to mark them as writable modules. Moe will allow read and write access to these dataspaces and make them visible in a different namespace called `rwfs`.

An example entry in 'modules.list' would look like this:

```
module somemodule :rw
```

Note

In order for a client to receive write permissions to the dataspace, the corresponding cap also needs write permissions.

Capabilities

- `svr`
Server Capability of application. Endpoint for IPC calls
Mandatory capability.

Command Line Options

- `--debug=<flags>`
This option enables debug messages from Moe itself.
Multiple Flags in one string separated through '|', '+' or ','.
Possible values for `<flags>` are
 - info
 - warn
 - boot
 - server
 - exceptions
 - loader
 - ns

- all
- `--init=<init_process>`

This options allows to set the init process binary.

String value.

Default: `rom/ned`
- `--l4re-dbg=<flags>`

This option allows to set the debug options for the [L4Re](#) runtime environment of the init process.

Multiple Flags in one string separated through '|', '+' or ','.

Possible values for `<flags>` are

 - info
 - warn
 - boot
 - server
 - exceptions
 - loader
 - ns
 - all
- `--ldr-flags=<flags>`

This option allows setting some loader options for the [L4Re](#) runtime environment.

Multiple Flags in one string separated through '|', '+' or ','.

Possible values for `<flags>` are

 - pre_alloc
 - all_segs_cow
 - pinned_segs
- `--brk=<address>`

This option is only present on systems without MMU. It restricts dynamic allocations to addresses equal or higher than `<address>`. Use it to prevent moe from allocating memory in regions that shall later be used by other applications or virtual machines.

Hexadecimal number without '0x' prefix.
- `-- <init_options>`

All command-line parameters after the special `--` option are passed directly to the init process.

Namespace

Call: `create (L4.Proto.Namespace)`

Dataspace

Dataspaces can be allocated with an arbitrary size. The granularity for memory allocation however is the machine page size ([L4_PAGESIZE](#)). A dataspace user must be aware that, as a result of this page-size granularity, there may be padding memory at the end of a dataspace which is accessible to each client. Moe currently allows most dataspace operations on this padding area. Nonetheless, the client must not make any assumptions about the size or content of the padding area, as this behaviour might change in the future.

The provided data spaces can have different characteristics:

- Physically contiguous and pre-allocated
- Non contiguous and on-demand allocated with possible copy on write (COW)

Dataspaces allocated via the Moe's factory allow mappings with any combination of the read-write-execute (RWX) rights, subject to a possible restriction of the writable right for client capabilities lacking the 'W' right.

Call: `create(L4.Proto.Dataspace, size [, flags, align])`

- `size`
Size in bytes.
Numerical value.
- `flags`
Special dataspace properties
Bitmap of length 4 with the single bits holding the following meaning:
 - 0x01: Continuous Allocate physically contiguous memory.
 - 0x02: Pinned Deprecated, use [L4Re::Dma_space](#) instead.
 - 0x04: Super pages Allocate super pages.
 - 0x08: Fixed physical Address Allocate at fixed physical address. Only honored on no-MMU systems. Will fail on MMU systems.
- `align`
Log2 alignment of dataspace if supported by allocator
Numerical value.

DMA space

Call: `create(L4.Proto.Dma_space)`

Region Map

Call: `create(L4.Proto.Rm)`

Virtual console

The logging facility of Moe provides per application tagged and synchronized log output.

Call: `create(L4.Proto.Log [, label, "color=(string|int)"])`

- `label`

Label used as prefix for the console output.

String value.

- `"color=(string|int) "`

Color of client's output

The Value for this parameter can be one of the following:

- `string`

Define the color by the first character of the given string

Possible values are

- * `n`: gray
- * `r`: red
- * `g`: green
- * `y`: yellow
- * `b`: blue
- * `m`: magenta
- * `c`: cyan
- * `w`: white
- * `N`: black
- * `R`: light red
- * `G`: light green
- * `Y`: light yellow
- * `B`: light blue
- * `M`: light magenta
- * `C`: light cyan
- * `W`: bright white

- `int`

Define the color by integer.

Possible values are

- * `0`: gray
- * `1`: red
- * `2`: green
- * `3`: yellow
- * `4`: blue
- * `5`: magenta
- * `6`: cyan
- * `7`: white
- * `8`: black
- * `9`: light red
- * `10`: light green
- * `11`: light yellow
- * `12`: light blue
- * `13`: light magenta
- * `14`: light cyan
- * `15`: bright white

Default: white

Scheduler

The scheduler subsystem provides a simple scheduler proxy for scheduling policy enforcement.

The priority offset provided on the creation of a scheduler proxy defines the minimum priority assigned to threads which are scheduled by that instance of the scheduler proxy. The offset is implicitly added to priorities provided to `L4::Scheduler.run_thread()`.

Call: `create(L4.Proto.Scheduler, limit, offset [, bitmap])`

- `limit`
Maximum priority.
Numerical value.
- `offset`
Priority offset.
Numerical value.
- `bitmap`
Bitmap of CPUs - can be repeated to address higher order CPUs
Numerical value.

Factory

Call: `create(L4.Proto.Factory [, quota])`

- `quota`
Limit in bytes. The limit is deducted from the limit of the factory that creates the new factory.
Numerical value.
 - Must not be zero.

4.3 Ned, the Init Process

Ned's job is to bootstrap the system running on [L4Re](#).

The main thing to do here is to coordinate the startup of services and applications as well as to provide the communication channels for them. The central facility in Ned is the Lua (<http://www.lua.org>) script interpreter with the [L4Re](#) and ELF-loader bindings.

The boot process is based on the execution of one or more Lua scripts that create communication channels (IPC gates), instantiate other [L4Re](#) objects, organize capabilities to these objects in sets, and start application processes with access to those objects (or based on those objects).

For starting applications, Ned depends on the services of [Moe](#) or another *loader*, which must provide data spaces and region maps. Ned also uses the 'rom' capability as source for Lua scripts and at least the 'l4re' binary (the runtime environment core) running in each application.

Each application Ned starts is equipped with an [L4Re::Env](#) environment that provides information about all the initial objects made accessible to this application.

4.3.1 Lua Bindings for L4Re

Ned provides various bindings for [L4Re](#) abstractions. These bindings are located in the 'L4' package (`require "L4"`).

4.3.1.1 Tutorial

For a verbose example using the Ned Lua bindings, see [tutorial.lua](#).

4.3.1.2 Capabilities in Lua

Capabilities are handled as normal values in Lua. They can be stored in normal variables or Lua compound structures (tables). A capability in Lua possesses additional information about the access rights that shall be transferred to other tasks when the capability is transferred. To support implementation of the Principle of Least Privilege, minimal rights are assigned by default. Extended rights can be added using the method `mode("...")` (short `m("...")`) that returns a new reference to the capability with the given rights.

Note

It is generally impossible to elevate the real access rights to an object. This means that if Ned has only restricted rights to an object it is not possible to upgrade the access rights with the `mode` method.

The capabilities in Lua also carry dynamic type information about the referenced objects. They thereby provide type-specific operations on the objects, such as the `create` operation on a generic factory or the `query` and `register` operations on a name space.

4.3.1.3 Access to L4Re::Env Capabilities

The initial objects provided to Ned itself are accessible via the table `L4.Env`. The default (usually unnamed) capabilities are accessible as `factory`, `log`, `mem_alloc`, `parent`, `rm`, and `scheduler` in the `L4.Env` table.

4.3.1.4 Constants

Protocols

The protocol constants are defined by default in the [L4](#) package's table `L4.Proto`. The definition is not complete and only covers what is usually needed to configure and start applications. The protocols are for example used as first argument to the `Factory:create` method.

```
Proto = {
  Dataspace = 0x4000,
  Namespace = 0x4001,
  Goos      = 0x4003,
  Mem_alloc = 0x4004,
  Rm        = 0x4005,
  Event     = 0x4006,
  Inhibitor = 0x4007,
  Sigma0    = -6,
  Log       = -13,
  Scheduler = -14,
  Factory   = -15,
  Vm        = -16,
  Dma_space = -17,
  Irq_sender = -18,
  Semaphore = -20,
```

```

Iommu      = -22,
Ipc_gate   = 0,
}

```

Rights

The rights of a Lua capability can be defined in two ways via the `:mode()` interface. Either via a string representation of the rights or via an integer value. An example for the former is `:mode("rsnc")` while the latter equivalent is `:mode(L4.Rights.r | L4.Rights.s | L4.Rights.n | L4.Rights.c)`. The following listing shows the integer constants. The constant names can be used in the string parameter to `:mode()`.

```

Rights = {
  s = 2,
  w = 1,
  r = 4,
  d = 8,
  n = 16,
  c = 32,
  ro = 4,
  rw = 5,
  rws = 7,
}

```

Debugging Flags

Debugging flags used for the applications [L4Re](#) core:

```

Dbg = {
  Info      = 1,
  Warn      = 2,
  Boot      = 4,
  Server    = 0x10,
  Exceptions = 0x20,
  Cmd_line  = 0x40,
  Loader    = 0x80,
  Name_space = 0x400,
  All       = 0xffffffff,
}

```

Loader Flags

Flags for configuring the loading process of an application.

```

Ldr_flags = {
  eager_map    = 0x1, -- L4RE_AUX_LDR_FLAG_EAGER_MAP
  all_segs_cow = 0x2, -- L4RE_AUX_LDR_FLAG_ALL_SEGS_COW
  pinned_segs  = 0x4, -- L4RE_AUX_LDR_FLAG_PINNED_SEGS
}

```

4.3.1.5 Application Startup Details

The central facility for starting a new task with Ned is the class `L4.Loader`. This class provides interfaces for conveniently configuring and starting programs. It provides three operations:

- `new_channel()` Returns a new IPC gate that can be used to connect two applications
- `start()` and `startv()` Start a new application process and return a process object

The `new_channel()` call is used to provide a service application with a communication channel to bind its initial service to. The concrete behavior of the object and the number of IPC gates required by a server depends on the server implementation. The channel can be passed to client applications as well or can be used for operations within the script itself.

`start()` and `startv()` always require at least two arguments. The first one is a table that contains information about the initial objects an application shall get. The second argument is a string, which for `start()` is the program name plus a white-space-separated list of program arguments (`argv`). For `startv()` the second argument is just the program binary name – which may contain spaces –, and the program arguments are provided as separate string arguments following the binary name (allowing spaces in arguments, too). The last optional argument is a table containing the POSIX environment variables for the program.

The Loader class uses reasonable defaults for most of the initial objects. However, you can override any initial object with some user-defined values. The main elements of the initial object table are:

- `factory` The factory used by the new process to create new kernel objects, such as threads etc. This must be a capability to an object implementing the [L4Re::Factory](#) protocol and defaults to the factory object provided to Ned.
- `mem` The memory allocator provided to the application and used by Ned allocates data spaces for the process. This defaults to Ned's memory allocator object (see [L4Re::Mem_alloc](#)).
- `rm_fab` The generic factory object used to allocate the region-map object for the process. (defaults to Ned's memory allocator).
- `log_fab` The generic factory to create the [L4Re::Log](#) object for the application's output (defaults to Ned's memory allocator). The `create` method of the `log_fab` object is called with `log_tag` and `log_color`, from this table, as arguments.
- `log` A table with parameters passed to the `log_fab`:
 - The first item is a short string defining the tag used for tagging log output of this process (defaults to the program name).
 - The second item is a string defining the color used for the log tag and the log string (defaults to "white").
 - Further parameters might be evaluated by certain implementations of the [L4Re::Log](#) interface.
- `scheduler` The scheduler object used for the process' threads (defaults to Ned's own scheduler).
- `caps` The table with application-specific named capabilities (default is an empty table). If the table does not contain a capability with the name 'rom', the 'rom' capability from Ned's initial caps is inserted into the table.

Less frequently used parameters:

- `l4re_dbg` An integer value overriding the debug level of the ITAS used for this application. Default is 2 (Warn). See *Debugging Flags* above.
- `ldr_flags` An integer value for setting additional flags for attaching regions, see *Loader Flags* above.

The `start()` and `startv()` calls return a task object that supports a number of operations.

- `state()` returns a string with the current task state: "running" or "zombie" if the task has terminated. If the task was already reaped (`wait()` returned) or if `kill()` was called, then the function will return `nil`.
- `exit_code()` returns the exit code if the task has terminated or `nil` if it was either killed or has been reaped.
- `kill()` forcefully terminates the task. Returns "killed" if the task was terminated or the exit code if the task was already gone. Returns `nil` if the task was already reaped.
- `exit_handler(function)` registers a Lua function that is invoked when the task terminates. If the task has already terminated, the function is called immediately. Returns `true` if the callback is pending, otherwise `false`. The callback function gets the exit code (`nil` if killed) of the task as its only parameter. The return value of the function is ignored. Only one callback can be registered.
- `wait()` suspends execution until the task has terminated. It's better to use `exit_handler()` instead. While the Lua code executes `wait()`, no `exit_handler()` will be dispatched.

4.3.1.6 Reacting on task termination

Ned can react on the termination of child tasks. The preferred mechanism is to register an `exit_handler()` for a task:

```
task = L4.default_loader:start(...)
task:exit_handler(function(exit_code)
  if exit_code == nil then
    print("Task was killed")
  else
    print("Task has terminated w/ code [" .. exit_code .. "]")
  end
end)
```

If the task did already terminate then the callback is invoked immediately. It is also possible to suspend execution of the Ned script until a task has terminated:

```
task = L4.default_loader:start(...)
task:wait()
```

This method should be used with caution, though. The Ned script will wait until the child task has terminated. Neither will any of the registered `exit_handler()` will be called during this time, nor will the [remote command interface](#) be able to execute commands either.

4.3.1.7 Control scheduling

Scheduling of L4Re applications is controlled by creating scheduler proxies. The proxy restricts the threads of an application to run on a subset of the available CPUs and to set their minimum and maximum priority. Use the loader factory function to create the proxy and pass it to the application:

```
sched_proxy = L4.default_loader:create_sched_proxy{ cpus=L4.Cpu_set:new("0-3") }
L4.default_loader:start({ scheduler = sched_proxy }, ...)
```

The `create_sched_proxy` function takes the following table arguments. All arguments are optional:

`cpus`: Set of allowed CPUs. Default: all CPUs. `prio_offset`: Base priority of all threads. Default: 0. `prio_limit`: Maximum priority of all threads. Default: `prio_offset + 10`. `fab`: Scheduler proxy factory capability. Default: `loader sched_fab`.

The `Cpu_set` constructor takes any number of CPU numbers or ranges:

```
Cpu_set:new()           -- all CPUs (because no argument was passed)
Cpu_set:new{}           -- empty CPU set (because an empty table was passed)
Cpu_set:new(42)         -- single CPU: 42
Cpu_set:new("0-3")     -- 4 CPUs: 0..3
Cpu_set:new("0-3", 42) -- 5 CPUs: 0..3, 42
```

A limited number of operations are defined on CPU sets. To compute the union of two sets (all CPUs of both sets), use the `|` operator. To compute the intersection of two sets (all CPUs common to both sets), use the `&` operator.

4.3.1.8 Access to the kernel debugger

Applications can enrich the kernel debugger with information using the API defined in [l4/sys/debugger](#). In order to do so, the developer has to assign access to the kernel debugger kernel object to the application. This can be done like this:

```
L4.default_loader:start({ caps = { jdb = L4.Env.jdb; }}, "rom/example")
```

4.3.1.9 Using the interactive ned prompt

Ned can be used in interactive mode by connecting the small ned-prompt helper tool to the command capability. Add the following code snippet at the end of your ned script:

```
local L4 = require("L4");
local l = L4.default_loader;

cmd = l:new_channel()

l:start({ log = L4.Env.log, caps = { svr = cmd } }, "rom/ned-prompt")

L4.server_loop(cmd)
```

The script hands in ned's own log capability to ned-prompt. This ensures that input and output of ned and the prompt appear on the same console.

ned-prompt needs to be added to your modules list.

4.3.2 Command Line Options

Ned's command line syntax is:

```
[--exit|--wait-and-exit] [--disable-backtracer-autoload] [--execute|-e STATEMENT] <lua script> [options passed]
```

Ned interprets the first non-option argument `<lua script>` as the Lua script which it should load and run. All arguments following the first non-option argument are passed as arguments to the Lua script via Lua's global `arg` table.

- Exit Options: **exit**, **wait-and-exit** (must be first if used)
 - **exit**: terminates the application after the script has run through even if there are still tasks running. `wait` for tasks at the end of the script to ensure they do not die forcefully.
 - **exit-and-wait**: terminates the application after the script has run through and all tasks started by ned have signaled their exit.
- Execute Statement Option: **execute**, **e**
Execute the Lua statement `STATEMENT`.
- **disable-backtracer-autoload**: Do not load the backtracer automatically.

4.4 Io, the Io Server

The Io server handles all platform devices and resources such as I/O memory, ports (on x86) and interrupts, and grants access to those to clients.

Upon startup Io discovers all platform devices using available means on the system, e.g. on x86 the PCI bus is scanned and the ACPI subsystem initialised. Available I/O resource can also be configured via configuration scripts.

Io's configuration consists of two parts:

- the description of the real hardware
- the description of virtual buses

Both descriptions represent a hierarchical (tree) structure of device nodes. Where each device has a set of resources attached to it. And a device that has child devices can be considered a bus.

Hardware Description

The hardware description represents the devices that are available on the particular platform including their resource descriptions, such as MMIO regions, IO-Port regions, IRQs, bus numbers etc.

The root of the hardware devices is formed by a system bus device (accessible in the configuration via `lo.system↔_bus()`). As mentioned before, platforms that support methods for device discovery may populate the hardware description automatically, for example from ACPI. On platforms that do not have support for such methods you have to specify the hardware description by hand. A simple example for this is `x86-legacy.devs`.

Virtual Bus Description

Each lo server client is provided with its own virtual bus which it can iterate to find devices. A virtual PCI bus may be a part of this virtual bus.

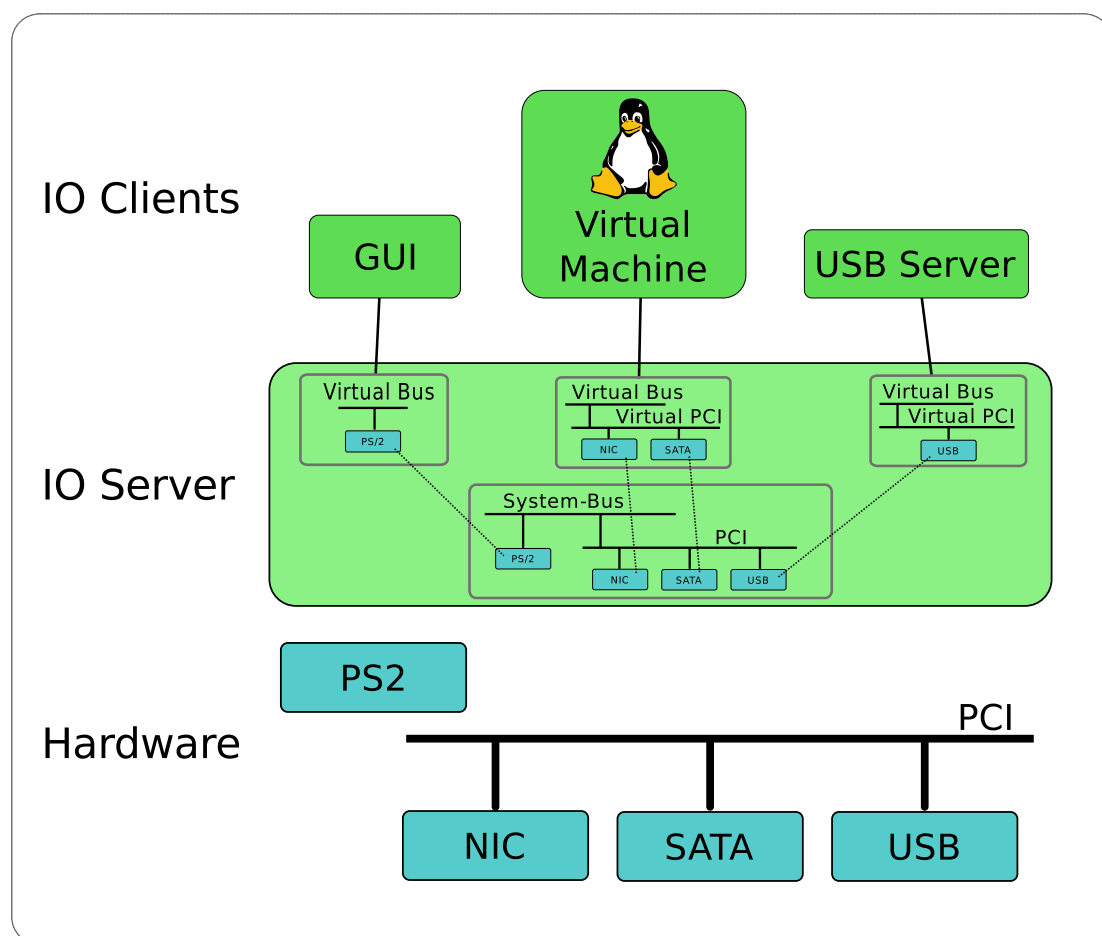


Figure 4.1 IO Service Architecture Overview

The lo server must be configured to create virtual buses for its clients.

This is done with at least one configuration file specifying static resources as well as virtual buses for clients. The configuration may be split across several configuration files passed to lo through the command line.

To allow clients access to available devices, a virtual system bus needs to be created that lists the devices and their resources that should be available to that client. The names of the buses correspond to the capabilities given to lo in its launch configuration.

A very simple configuration for Io could look like this:

```
-- vim:ft=lua
-- Example configuration for io

-- Configure two platform devices to be known to io
Io.Dt.add_children(Io.system_bus(), function()

    -- create a new hardware device called "FOODEVICE"
    FOODEVICE = Io.Hw.Device(function()
        -- set the compatibility IDs for this device
        -- a client tries to match against these IDs and configures
        -- itself accordingly
        -- the list should be sorted from specific to less specific IDs
        compatible = {"dev-foo,mmio", "dev-foo"};

        -- set the 'hid' property of the device, the hid can also be used
        -- as a compatible ID when matching clients
        Property.hid = "dev-foo,Example";

        -- note: names for resources are truncated to 4 letters and a client
        -- can determine the name from the ID field of a l4vbus_resource_t
        -- add two resources 'irq0' and 'reg0' to the device
        Resource.irq0 = Io.Res.irq(17);
        Resource.reg0 = Io.Res.mmio(0x6f000000, 0x6f007fff);
    end);

    -- create a new hardware device called "BARDEVICE"
    BARDEVICE = Io.Hw.Device(function()
        -- set the compatibility IDs for this device
        -- a client tries to match against these IDs and configures
        -- itself accordingly
        -- the list should be sorted from specific to less specific IDs
        compatible = {"dev-bar,mmio", "dev-bar"};

        -- set the 'hid' property of the device, the hid can also be used
        -- as a compatible ID when matching clients
        Property.hid = "dev-bar,Example";

        -- Specify that this device is able to use direct memory access (DMA).
        -- This is needed to allow clients to gain access to DMA addresses
        -- used by this device to directly access memory.
        Property.flags = Io.Hw_device_DF_dma_supported;

        -- note: names for resources are truncated to 4 letters and a client
        -- can determine the name from the ID field of a l4vbus_resource_t
        -- add three resources 'irq0', 'irq1', and 'reg0' to the device
        Resource.irq0 = Io.Res.irq(19);
        Resource.irq1 = Io.Res.irq(20);
        Resource.reg0 = Io.Res.mmio(0x6f100000, 0x6f100fff);
    end);
end);

Io.add_vbusses
{
    -- Create a virtual bus for a client and give access to FOODEVICE
    client1 = Io.Vi.System_bus(function ()
        dev = wrap(Io.system_bus():match("dev-foo,mmio"));
    end);

    -- Create a virtual bus for another client and give it access to BARDEVICE
    client2 = Io.Vi.System_bus(function ()
        dev = wrap(Io.system_bus():match("dev-bar,Example"));
    end);
}
```

Each device supports a 'compatible' property. It is a list of compatibility strings. A client matches itself against one (or multiple) compatibility IDs and configures itself accordingly. All other device members are handled according to their type. If the type is a resource (Io.Res) it is added as a named resource. Note that resource names are truncated to 4 letters and are stored in the ID field of a [l4vbus_resource_t](#). If the type is a device it is added as a child device to the current one. All other types are treated as a device property which can be used to configure a device driver. Right now, device properties are internal to Io only.

Matching and Assigning PCI Devices

Assigning clients PCI devices could look like this:

```
-- This is a configuration snippet for PCI device selection

local hw = Io.system_bus();

Io.add_vbusses
{
  pciclient = Io.Vi.System_bus(function ()
    PCI = Io.Vi.PCI_bus(function ()
      pci_mm      = wrap(hw:match("PCI/CC_04"));
      pci_net     = wrap(hw:match("PCI/CC_02"));
      pci_storage = wrap(hw:match("PCI/CC_01"));
    end)
  end)
}
```

The "PCI/" is followed by a bus-specific ID string. The format of the PCI ID string may be one of the following:

- PCI/CC_cc
- PCI/CC_ccss
- PCI/CC_ccssp
- PCI/VEN_vvvv
- PCI/DEV_ddd
- PCI/SUBSYS_sssssss
- PCI/REV_rr
- PCI/ADR_xxxx:xx:xx.x

Where:

- `cc` is the hexadecimal representation of the class code byte
- `ss` is the hexadecimal representation of the subclass code byte
- `pp` is the hexadecimal representation of the programming interface byte
- `vvvv` is the hexadecimal representation of the vendor ID
- `ddd` is the hexadecimal representation of the device ID
- `ssssssss` is the hexadecimal representation of the subsystem ID
- `rr` is the hexadecimal representation of the revision byte
- `xxxx:xx:xx.x` is the bus address in PCI nomenclature

As a special extension Io supports replacing the ID string with a human-readable common PCI class name. The following table gives an overview of the names known to Io and their respective PCI class and subclass.

| Common Name | Description | PCI ID string |
|-------------|-------------------------|---------------|
| storage | Mass storage controller | CC_01 |
| scsi | SCSI storage controller | CC_0100 |
| ide | IDE interface | CC_0101 |
| floppy | Floppy disk controller | CC_0102 |
| raid | RAID bus controller | CC_0104 |
| ata | ATA controller | CC_0105 |

| Common Name | Description | PCI ID string |
|----------------------|------------------------------------|---------------|
| sata | SATA controller | CC_0106 |
| sas | Serial attached SCSI controller | CC_0107 |
| nvm | Non-volatile memory controller | CC_0108 |
| - | - | - |
| network | Network controller | CC_02 |
| ethernet | Ethernet controller | CC_0200 |
| token_ring | Token ring network controller | CC_0201 |
| fddi | FDDI network controller | CC_0202 |
| atm | ATM network controller | CC_0203 |
| isdn | ISDN controller | CC_0204 |
| picmg | PICMG controller | CC_0206 |
| net_infiniband | Infiniband controller | CC_0207 |
| fabric | Fabric controller | CC_0208 |
| network_nw | Network controller e.g. Wifi | CC_0280 |
| - | - | - |
| display | Display controller | CC_03 |
| vga | VGA compatible controller | CC_0300 |
| xga | XGA compatible controller | CC_0301 |
| - | - | - |
| media | Multimedia controller | CC_04 |
| mm_video | Multimedia video controller | CC_0400 |
| mm_audio | Multimedia audio controller | CC_0401 |
| telephony | Computer telephony device | CC_0402 |
| audio | Audio device | CC_0403 |
| - | - | - |
| bridge | Bridge | CC_06 |
| br_host | Host bridge | CC_0600 |
| br_isa | ISA bridge | CC_0601 |
| br_eisa | EISA bridge | CC_0602 |
| br_microchannel | MicroChannel bridge | CC_0603 |
| br_pci | PCI bridge | CC_0604 |
| br_pcmcia | PCMCIA bridge | CC_0605 |
| br_nubus | NuBus bridge | CC_0606 |
| br_cardbus | CardBus bridge | CC_0607 |
| br_raceway | RACEway bridge | CC_0608 |
| br_semi_pci | Semi-transparent PCI-to-PCI bridge | CC_0609 |
| br_infiniband_to_pci | InfiniBand to PCI host bridge | CC_060a |
| - | - | - |
| com | Communication controller | CC_07 |
| com_serial | Serial controller | CC_0700 |
| com_parallel | Parallel controller | CC_0701 |

| Common Name | Description | PCI ID string |
|-------------------|-----------------------------|---------------|
| com_multiport_ser | Multiport serial controller | CC_0702 |
| com_modem | Modem | CC_0703 |
| com_gpib | GPiB controller | CC_0704 |
| com_smart_card | Smart card controller | CC_0705 |
| - | - | - |
| serial_bus | Serial bus controller | CC_0c |
| firewire | FireWire (IEEE 1394) | CC_0c00 |
| access_bus | ACCESS bus | CC_0c01 |
| ssa | SSA | CC_0c02 |
| usb | USB controller | CC_0c03 |
| fibre_channel | Fibre channel | CC_0c04 |
| smbus | SMBus | CC_0c05 |
| bus_infiniband | InfiniBand | CC_0c06 |
| ipmi_smic | IPMI SMIC interface | CC_0c07 |
| sercos | SERCOS interface | CC_0c08 |
| canbus | CAN bus | CC_0c09 |
| - | - | - |
| wireless | Wireless controller | CC_0d |
| bluetooth | Bluetooth | CC_0d11 |
| w_8021a | 802.1a controller | CC_0d20 |
| w_8021b | 802.1b controller | CC_0d21 |

Strong Matching of PCI Devices

If more specific matching of PCI devices is required it is possible to concatenate multiple ID strings using &. An example where a specific device from a specific vendor at a fixed bus address is matched would use the string `PCI/VEN_vvvv&DEV_dddd&ADR_xxxx:xx:xx.x`.

Isolation of PCIe devices

PCIe encodes device communication with a network-like protocol with destination headers and packet fragmentation allowing a devices to talk directly to other devices. This potentially works against security boundaries for a system. E.g. two network cards could exchange packets and thereby leak information from one security domain to the other without involvement of the OS.

PCIe introduced an optional capability named PCI Access Control Services (PCI/ACS) to control communication between PCIe devices.

With PCI/ACS it is possible to restrict inter-device communication between PCIe devices.

PCI/ACS is optional and for Intel chipsets, it is usually only implemented on high-end PCI platform controller hubs (PCHs), and is missing on low-end and mobile PCHs. On some Intel-PCHs there exist facilities that allow for similar isolation.

If IO encounters a supported PCH, it will enable those facilities in order to enforce device isolation.

Command Line Options

The lo Server supports the following optional parameters:

```
[--verbose|v] [--transparent-msi] [--trace <trace_mask>] [--acpi-debug-level <debug_level>] [config_files]
```

- **verbose|v**

By default, error debug messages are enabled. This option increments the verbosity level, and can be applied multiple times to reach the desired debug level. The available debug levels are ordered as: `DBG_ERR` (default, level 1), `DBG_WARN`, `DBG_INFO`, `DBG_DEBUG`, `DBG_DEBUG2` and `DBG_ALL` (level 6).

- **transparent-msi**

Enable MSI on PCI devices which support this feature. This is transparent to clients, as there are no changes in the API used to interact with PCI device via interrupts.

- **acpi-debug-level <level_mask>**

Set the ACPI debug level. The `<level_mask>` is a mask that selects components of interest for debugging. It can be constructed from the ACPI debug constants defined in the linux kernel, see [ACPI Debug Output](#) for details. By default, the ACPI debug level is set to `ACPI_LV_INIT | ACPI_LV_TABLES | ACPI_LV_VERBOSE_INFO`.

- **trace <trace_mask>**

Enable tracing of events matching `trace_mask`. The only supported trace mask is 1 and this matches ACPI events.

- **config_files**

Space separated list of Lua configuration files specifying real hardware and virtual buses. See example on [Virtual Bus Description](#).

4.5 Virtio Net P2P, a virtual network point-to-point link

The virtual network point-to-point server (p2p) connects two clients with a virtual network connection. It uses virtio as the transport mechanism. Each virtual network p2p endpoint implements the device-side of a virtio network device. Each client can access its endpoint using the driver-side semantics of a virtio network device.

Capabilities

- **dataspace**

Trusted dataspaces

Multiple capability names can be provided by the `--register-ds` command line parameter.

- **svr**

Server Capability of application. Endpoint for IPC calls

Mandatory capability.

Command Line Options

The following command line options are supported:

- `-p <num_usec>, --poll <num_usec>`
 Enable polling mode and set the poll interval. IRQ notification is disabled for queues while in polling mode.
 Numerical value.
 - Must be a positive integer specified in microseconds.
- `-s <num>, --size <num>`
 Set the maximum queue size for the device-side virtio queues.
 Numerical value.
 - Must be a power of 2 in the range of 1 to 32768 inclusive.
 Default: 256
- `-d <cap_name>, --register-ds <cap_name>`
 Register a trusted dataspace capability. If this option is used, it is not possible to communicate with the server via dataspaces other than the registered ones.
 Can be used multiple times.
 Name of a provided capability that adheres to the dataspace protocol.

Building and Configuration

The virtual network p2p server can be built using the [L4Re](#) build system by placing this project into the `pkg` directory.

Starting the service

The virtual network p2p server can be started with Lua like this:

```
local p2p = L4.default_loader:new_channel();
L4.default_loader:start(
{
  caps = {
    svr = p2p:svr(),
  },
},
"rom/l4vio_net_p2p [<options>]");
```

First an IPC gate (`p2p`) is created which is used between the virtual network p2p server and a client to create new virtual ports. The server-side is assigned to the mandatory `svr` capability of the virtual network p2p server. See the section below on how to create a new virtual port and connect a client to it.

Virtual p2p port

Prior to connecting a client to a virtual network p2p server port it has to be created using the following Lua function. It has to be called on the client side of the IPC gate capability whose server side is bound to the virtual network p2p server.

The "key=value" pairs passed to create() can be omitted and their order is not important.

Call: `create(0 [, "ds-max=<max>", "mac=<addr>"])`

- "ds-max=<max>"

Specifies the upper limit of the number of dataspace the client is allowed to register with the server for virtio DMA.

Numerical value.

- Must be in the range of 1 to 80 inclusive.

Default: 2

- "mac=<addr>"

Specify the MAC address of the endpoint where <mac_address> is of the form xx:xx:xx:xx:xx:xx.

String value.

- Must be in the form xx:xx:xx:xx:xx:xx.

If the `create()` call is successful a new capability which references a virtual network p2p server port is returned. A client uses this capability to talk to the virtual network p2p server using the virtio network protocol.

Examples

A couple of examples on how to create ports with different properties are listed below.

```
-- two normal ports with at most 4 data spaces
net0 = p2p:create(0, "ds-max=4")
net1 = p2p:create(0, "ds-max=4")
-- normal port with 4 data spaces and MAC address
net0 = p2p:create(0, "ds-max=4", "mac=11:22:33:44:55:66")
```

4.6 Virtio Net Switch, a virtual network switch

The virtual network switch connects multiple clients with a virtual network connection. It uses Virtio as the transport mechanism. Each virtual switch port implements the host-side of a Virtio network device (virtio-net).

Capabilities

- `dataspace`

Trusted dataspace

Multiple capability names can be provided by the `--register-ds` command line parameter.

- `svr`

Server Capability of application. Endpoint for IPC calls

Mandatory capability.

Command Line Options

In the example above the virtual network switch is started in its default configuration with a maximum of 5 virtual ports. To customize the configuration the virtual network switch accepts the following command line options:

- `-D <component=level>, --debug <component=level>`
 Configure individual debug levels per component. Verbosity increases from `quiet` up to `trace`.
 Can be used multiple times.
 Possible values for component are | `core, virtio, port, request, queue, packet`.
 Possible values for level are | `quiet, warn, info, debug, trace`.
- `-m, --mac`
 Ignored. Provided for compatibility with older switch versions.
 Flag. True if provided.
- `-M`
 Do not assign a random MAC address to ports by default. It is always possible to set an explicit MAC address by passing the `mac=` to the factory call, regardless of this option. If `-M` is passed and no `mac=` address was given, it is the responsibility of the Virtio driver to choose an appropriate address.
 Flag. True if provided.
- `-p <num>, --ports <num>`
 Set the maximum number of virtual ports.
 Numerical value.
 Default: 256
- `-q, --quiet`
 Silence all output except for error messages.
 Flag. True if provided.
- `-s <num>, --size <num>`
 Set the maximum queue size for the device-side Virtio queues.
 Numerical value.
 - Must be a power of 2 in the range of 1 to 32768 inclusive.
 Default: 256
- `-v, --verbose`
 Increase the global verbosity level. Individual levels per component can be set using the `-D` option.
 Can be used up to 3 times.
 Flag. True if provided.
- `-d <cap_name>, --register-ds <cap_name>`
 Register a trusted dataspace capability. If this option gets used, it is not possible to communicate with the server via dataspace other than the registered ones. Can be used multiple times for multiple dataspace.
 The option's parameter is the name of a dataspace capability.
 Can be used multiple times.
 Name of a provided capability that adheres to the dataspace protocol.

The virtual network switch can be setup to feature exactly one monitor port. All traffic passing through the switch is mirrored to the monitor port. The monitor port is read-only, and has no TX capability. An optional packet filter can be configured and implemented to filter data sent to the monitor port.

Configuration

Certain features of the virtual network switch are configurable at compile-time. Configuration is done through the build-time configuration of the [L4Re](#) build tree.

Starting the service

The virtual network switch can be started in Ned like this:

```
local switch = L4.default_loader:new_channel();
L4.default_loader:start(
{
  caps = {
    svr = switch:svr(),
  },
},
"rom/l4vio_switch");
```

First a communication channel (`switch`) is created which is used to create virtual network ports. It is connected to the switch component via its mandatory `svr` capability. See the section below on how to create a new virtual port and connect a client to it.

Hardware devices

To plug hardware devices into the switch, provide a `Vbus` capability with the name `vbus` when starting the switch. To use this feature, you have to enable the `VNS_IXL` config option.

Virtual switch port

First, a virtual network port has to be created using the following Ned-Lua function. It has to be called on the communication channel called `switch`, which has been created earlier.

```
Call: create(0 [, "ds-max=<max>", "name=<name>", "type=<port type>", "vlan=(access=<vlan id>|trunk=<vlan id>[,<vlan id>]*)", "mac=<addr>"])
```

- "ds-max=<max>"

Specifies the upper limit of the number of dataspace the client is allowed to register with the virtual network switch for Virtio DMA.

Numerical value.

- In the range of 1 to 80 inclusive

- "name=<name>"

Sets the name of port in debug messages to `<name>`. A name may consist of at most 19 characters, all other characters are dropped. If there is enough space left, the name will get a postfix of "`[<port number>]`", e.g. "name=foo" -> foo[1].

String value.

- "type=<port type>"

Optionally specify the port type.

Possible values for `<port type>` are

- monitor: Monitor Port
- normal: Normal Port

Default: normal

- "vlan=(access=<vlan id>|trunk=<vlan id>[,<vlan id>]*)"

Configure the port to participate in an IEEE 802.1Q compatible VLAN. Fundamentally there are two types of ports: access ports and trunk ports.

The Value for this parameter can be one of the following:

- "access=<vlan id>"

Configures the port as access port for VLAN <vlan id> where the id must be a decimal number greater than 0 and less than 4095 in accordance to the standard. Packets on an access port belong to the configured VLAN and are only forwarded to ports that belong to the same VLAN or trunk ports that participate in the particular VLAN. The packets on this port will not have a VLAN tag attached to them so that a guest connected to this port does not see that the port is part of a VLAN.

An optional monitor port will see packets from an access port as VLAN tagged packets with the <vlan id> given for the port.

Numerical value.

- * In the range of 0 to 4095 exclusive

- "trunk=<vlan id>[, <vlan id>]*"

Configures the port as trunk port. It participates either in all VLANs, if specified by the keyword 'all', or in the list of VLANs given as comma separated list. There must be no whitespace in the list. Each id must be a decimal number greater than 0 and less than 4095 in accordance to the standard. Outgoing packets on this port will be tagged with an IEEE 802.1Q compatible tag. Incoming packets must be tagged with a VLAN tag from the given list. Packets that have no tag or a tag not in the vlan id list are dropped silently. They are not forwarded to the monitor port either. Currently there is no support for IEEE 802.1p. The PCP and DEI sub-fields in the TCI field will be set to zero on outgoing packets and are ignored for incoming packets.

Numerical value.

- "mac=<addr>"

Explicitly sets the MAC address of the port. It will be checked that no other port on the switch has the same address. It is the responsibility of the user to ensure the validity of the address and its global uniqueness, though.

String value.

- In form xx:xx:xx:xx:xx:xx

If the `create()` call is successful a new capability which references a virtual switch port is returned. A client uses this capability to talk to the virtual network switch using the Virtio network protocol.

Examples

Here are couple of examples on how to create ports with different properties:

```
-- normal port with at most 4 data spaces
net0 = switch:create(0, "ds-max=4")
-- like the previous but with name foo
net0 = switch:create(0, "ds-max=4", "name=foo")
-- like the previous but the port is a monitor port
net0 = switch:create(0, "ds-max=4", "name=foo", "type=monitor")
-- normal port with 4 data spaces as access port to VLAN 1
net0 = switch:create(0, "ds-max=4", "name=v11", "vlan=access=1")
-- normal port with 4 data spaces as trunk port participating in VLAN 1 & 2
net0 = switch:create(0, "ds-max=4", "name=v11", "vlan=trunk=1,2")
```

4.7 Uvmm, the virtual machine monitor

Uvmm provides a virtual machine for running an unmodified guest in non- privileged mode.

Capabilities

- `ram`

The memory for the guest provided via a simple dataspace.

- `pfc`

Uvmm can be instructed to inform a PM manager of PM events through the [L4::Platform_control](#) interface. To that end, uvmm may be equipped with a `pfc` cap. On suspend, uvmm will call [l4_platform_ctl_system_suspend\(\)](#).

The `pfc` cap can also be implemented by IO. In that case the guest can start a machine suspend/shutdown/reboot.

Command Line Options

- `--mon <option>`

Control the handling of guest reads/writes to non-existing memory.

Possible values for `<option>` are

- `true`: The monitor interface is enabled but no server implementing the client side of the monitor interface is started. The monitor interface can still be utilized via cons but no readline functionality will be available.
- `some_binary`: If a string is passed as the value of `mon`, the monitor interface is enabled and the string is interpreted as the name of a server binary which implements the client side of the monitor interface. This server is automatically started and has access to a `vcon` capability named `mon` at startup through which it can make use of the monitor interface. Unless you have written your own server you should specify `'uvmm_cli'` which is a server implementing a simple readline interface.
- `false`: The monitor interface is disabled at runtime.

Default: `true`

- `-c <guest command line>,--cmdline <guest command line>`

Command line that is passed to the guest on boot.

String value.

- `-k <kernel image name>,--kernel <kernel image name>`

The name of the guest-kernel image file present in the ROM namespace.

File path in ROM namespace.

- `-d <DTB overlay>,--dtb <DTB overlay>`

The name of the device tree file present in the ROM namespace. The device tree will be placed in the upmost region of guest memory. Optionally, a user may use an additional parameter in the form of "`<DTB overlay>:limit=0xffffffff`" to set an upper limit for the device tree location.

This option can be issued multiple times. uvmm will merge all provided device tree overlays.

File path in ROM namespace.

- `-r <RAM disk name>,--ramdisk <RAM disk name>`

The name of the RAM disk file present in the ROM namespace

File path in ROM namespace.

- `-b <Base address of the guest RAM>,--rambase <Base address of the guest RAM>`

Physical start address for the guest RAM. This value is platform specific.

Numerical value.

- `-D <component=level>, --debug <component=level>`

Control the verbosity level of the uvmm.

Using the `component` prefix, the verbosity level of each uvmm component is configurable.

For example, the following command line sets the verbosity of all uvmm components to `info` except for IRQ handling, which is set to `trace`:

```
uvmm -D info -D irq=trace
```

Can be used multiple times.

Possible values for `component` are | `core, cpu, mmio, irq, dev, pm, vbus_event`.

Possible values for `level` are | `quiet, warn, info, trace`.

Options `-q, --quiet, -v, --verbose` and `-D, --debug` cancel each other out.

- `-f, --fault-mode`

Control the handling of guest reads/writes to non-existing memory.

Possible values are

- `ignore`: Invalid writes are ignored. Invalid reads either return 0 or are skipped. The guest may experience undefined behaviour.
- `halt`: Halt the VM on the first invalid memory access.
- `inject`: Try to forward the invalid access to the guest. This is not supported on all architectures. Falls back to `halt` if the error could not be forwarded to the guest.

Default: `ignore`

- `-q, --quiet`

Silence all uvmm output.

Flag. True if provided.

Options `-q, --quiet, -v, --verbose` and `-D, --debug` cancel each other out.

- `-v, --verbose`

Increase the verbosity of the uvmm. Repeating the option increases the verbosity by another level.

Can be used multiple times.

Flag. True if provided.

Options `-q, --quiet, -v, --verbose` and `-D, --debug` cancel each other out.

- `-W, --wakeup-on-system-resume`

When set, the uvmm resumes when the host system resumes after a suspend call.

Flag. True if provided.

- `-i, --ram-identity-mapping`

When set, the option forces the guest RAM to be mapped to its corresponding host-physical addresses.

Flag. True if provided.

Setting up guest memory

In the most simple setup, memory for the guest can be provided via a simple dataspace. In your ned script, create a new dataspace of the required size and hand it into uvmm as the `ram` capability:

```
local ramds = L4.Env.user_factory:create(L4.Proto.Dataspace, 0x4000000)

L4.default_loader::startv({caps = {ram = ramds:m("rw")}}, "rom/uvmm")
```

The memory will be mapped to the most appropriate place and a memory node added to the device tree, so that the guest can find the memory.

For a more complex setup, the memory can be configured via the device tree. uvmm scans for memory nodes and tries to set up the memory from them. A memory device node should look like this:

```
memory@0 {
    device_type = "memory";
    reg = <0x00000000 0x00100000
          0x00200000 0xffffffff>;
    l4vmm,dscap = "memcap";
    dma-ranges = <>;
};
```

The `device_type` property is mandatory and needs to be set to `memory`.

`l4vmm,dscap` contains the name of the capability containing the dataspace to be used for the RAM. `reg` describe the memory regions to use for the memory. The regions will be filled up to the size of the supplied dataspace. If they are larger, then the remaining area will be cut.

If the optional `dma-ranges` property is given, the host-physical address ranges for the memory regions will be added here. Note that the property is not cleared first, so it should be left empty.

For more details see [RAM configuration](#).

Memory layout

uvmm populates the RAM with the following data:

- kernel binary
- (optional) ramdisk
- (optional) device tree

The kernel binary is put at the predefined address. For ELF binaries, this is an absolute physical address. If the binary supports relative addressing, the binary is put to the requested offset relative to beginning of the first 'memory' region defined in the device tree.

The ramdisk and device tree are placed as far as possible to the end of the regions defined in the first 'memory' node.

If there is a part of RAM that must remain empty, then define an extra memory node for it in the device tree. uvmm only writes to memory in the first memory node it finds.

Warning: uvmm does not touch any unpopulated memory. In particular, it does not ensure that the memory is cleared. It is the responsibility of the provider of the RAM dataspace to make sure that no data leakage can happen. Normally this is not an issue because dataspaces are guaranteed to be cleaned when they are newly created but users should be careful when reusing memory or dataspaces, for example, when restarting the uvmm.

Forwarding hardware resources to the guest

Hardware resources must be specified in two places: the device tree contains the description of all hardware devices the guest could see and the Vbus describes which resources are actually available to the uvmm.

The vbus allows the uvmm access to hardware resources in the same way as any other [L4](#) application. uvmm expects a capability named 'vbus' where it can access its hardware resources. It is possible to leave out the capability for purely virtual guests (Note that this is not actually practical on some architectures. On ARM, for example, the guest needs hardware access to the interrupt controller. Without a 'vbus' capability, interrupts will not work.) For information on how to configure a vbus, see the [IO documentation](#).

The device tree needs to contain the hardware description the guest should see. For hardware devices this usually means to use a device tree that would also be used when running the guest directly on hardware.

On startup, uvmm scans the device tree for any devices that require memory or interrupt resources and compares the required resources with the ones available from its vbus. When all resources are available, it sets up the appropriate forwarding, so that the guest now has direct access to the hardware. If the resources are not available, the device will be marked as 'disabled'. This mechanism allows to work with a standard device tree for all guests in the system while handling the actual resource allocation in a flexible manner via the vbus configuration.

The default mechanism assigns all resources 1:1, i.e. with the same memory address and interrupt number as on hardware. It is also possible to map a hardware device to a different location. In this case, the assignment between vbus device and device tree device must be known in advance and marked in the device tree using the `l4vmm, vbus-dev` property.

The following device will for example be bound with the vbus device with the HID 'l4-test,dev':

```
test@e0000000 {
    compatible = "memdev,bar";
    reg = <0 0xe0000000 0 0x50000>,
        <0 0xe1000000 0 0x50000>;
    l4vmm,vbus-dev = "l4-test,dev";
    interrupts-extended = <&gic 0 139 4>;
};
```

Resources are then matched by name. Memory resources in the vbus must be named `reg0` to `reg9` to match against the address ranges in the device tree `reg` property. Interrupts must be called `irq0` to `irq9` and will be matched against `interrupts` or `interrupts-extended` entries in the device tree. The vbus must expose resources for all resources defined in the device tree entry or the initialisation will fail.

An appropriate IO entry for the above device would thus be:

```
MEM = Io.Hw.Device(function()
    Property.hid = "l4-test,dev"
    Resource.reg0 = Io.Res.mmio(0x41000000, 0x4104ffff)
    Resource.reg1 = Io.Res.mmio(0x42000000, 0x4204ffff)
    Resource.irq0 = Io.Res.irq(134);
end)
```

Please note that HIDs on the vbus are not necessarily unique. If multiple devices with the HID given in `l4vmm, vbus-dev` are available on the vbus, then one device is chosen at random.

If no vbus device with the given HID is available, the device is disabled.

How to enable guest suspend/resume

Note

Currently only supported on ARM. It should work fine with Linux version 4.4 or newer.

Uvmm (partially) implements the power state coordination interface (PSCI), which is the standard ARM power management interface. To make use of this interface, you have to announce its availability to the guest operating system via the device tree like so:

```
psci {
    compatible = "arm,psci-0.2";
    method = "hvc";
};
```

The Linux guest must be configured with at least these options:

```
CONFIG_SUSPEND=y
CONFIG_ARM_PSCI=y
```

How to communicate power management (PM) events

Uvmm can be instructed to inform a PM manager of PM events through the [L4::Platform_control](#) interface. To that end, uvmm may be equipped with a `pf_c` cap. On suspend, uvmm will call `l4_platform_ctl_system_suspend()`.

The `pf_c` cap can also be implemented by IO. In that case the guest can start a machine suspend/shutdown/reboot.

Ram block device support

The example ramdisk works by loading a file system into RAM, which needs RAM block device support to work. In the Linux kernel configuration add: `CONFIG_BLK_DEV_RAM=y`

Framebuffer support for uvmm/amd64 guests

Uvmm can be instructed to pass along a framebuffer to the Linux guest. To enable this three things need to be done:

1. Configure Linux to support a simple framebuffer by enabling

```
CONFIG_FB_SIMPLE=y
CONFIG_X86_SYSFB=y
```

2. Configure a simple framebuffer device in the device tree (currently only read by uvmm, linearer framebuffer at [0xf0000000 - 0xf1000000])

```
simplefb {
    compatible = "simple-framebuffer";
    reg = <0x0 0xf0000000 0x0 0x1000000>;
    l4vmm,fbcap = "fb";
};
```

3. Start a framebuffer instance and connect it to uvmm e.g.

```
-- Start fb-drv (but only if we need to)
local fbdrv_fb = L4.Env.vesa;
if (not fbdrv_fb) then
    fbdrv_fb = l:new_channel();
    l:start({
        caps = {
            vbus = io_busses.fbdrv,
            fb    = fbdrv_fb:svr(),
        },
        log = { "fbdrv", "r" },
    },
    "rom/fb-drv");
end
vmm.start_vm{
    ext_caps = { fb = fbdrv_fb },
    -- ...
```

Requirements on the Fiasco.OC configuration on amd64

The kernel configuration must feature `CONFIG_SYNC_TSC=y` in order for the emulated timers to reach a sufficiently high resolution.

Recommended Linux configuration options for uvmm/amd64 guests

The following options are recommended in addition to the amd64 defaults provided by a `make defconfig`:

Virtio support is required to access virtual devices provided by uvmm:

```
CONFIG_VIRTIO=y
CONFIG_VIRTIO_PCI=y
CONFIG_VIRTIO_BLK=y
CONFIG_BLK_MQ_VIRTIO=y
CONFIG_VIRTIO_CONSOLE=y
CONFIG_VIRTIO_INPUT=y
CONFIG_VIRTIO_NET=y
```

It is highly recommended to use the X2APIC, which needs virtualization awareness to work under uvmm:

```
CONFIG_X86_X2APIC=y
CONFIG_PARAVIRT=y
CONFIG_PARAVIRT_SPINLOCKS=y
```

KVM clock for uvmm/amd64 guests

When executing [L4Re](#) + uvmm on QEMU, the PIT as clock source is not reliable. The paravirtualized KVM clock provides the guest with a stable clock source.

A KVM clock device is available to the guest, if the device tree contains the corresponding entry:

```
kvm_clock {
    compatible = "kvm-clock";
    reg = <0x0 0x0 0x0 0x0>;
};
```

To make use of this clock, the Linux guest must be built with the following configuration options:

```
CONFIG_HYPERVISOR_GUEST=y
CONFIG_KVM_GUEST=y
CONFIG_PTP_1588_CLOCK_KVM is not set
```

Note

KVM calls besides the KVM clock are unhandled and lead to failure in the uvmm, e.g. `vmcall 0x9` for the `PTP_1588_CLOCK_KVM`.

This is considered a development feature. The KVM clock is not required when running on physical hardware as TSC calibration via the PIT works as expected.

Development notes for amd64

When you are developing on Linux using QEMU please note that nested virtualization support is necessary on your host system to run uvmm guests. Your host Linux version should be 4.12 or greater, **excluding 4.20**.

Check if your KVM module has nested virtualization enabled via:

```
cat /sys/module/kvm_intel/parameters/nested
Y
```

In case it shows N instead of Y enable nested virtualization support via:

```
modprobe kvm_intel nested=1
```

On AMD platforms the module name is `kvm_amd`.

QEMU network setup for a uvmm guest on amd64

```
qemu-system-x86_64 -M q35 -cpu host -enable-kvm -device intel-iommu
                  -device e1000e,netdev=net0 -netdev bridge,id=net0,br=virbr0
```

where 'virbr0' is the name of the host's bridge device. The line 'allow virbr0' needs to be present in `/etc/qemu/bridge.conf`. The bridge can either be created via the network manager or via the command line:

```
brctl addbr virbr0
ip addr add 192.168.124.1/24 dev virbr0
ip link set up dev virbr0
```

In the guest linux with `eth0` as network device:

```
ip a a 192.168.124.5/24 dev eth0
ip li se up dev eth0
```

Now the host and guest can ping each other using their respective IPs.

Of course, uvmm needs to be connected to `lo` and `lo` needs a vbus configuration for the uvmm client like this:

```
Io.add_vbusses
{
  vm_pci = Io.Vi.System_bus(function ()
    Property.num_msis = 6
    PCI = Io.Vi.PCI_bus(function ()
      pci_net = wrap(Io.system_bus():match("PCI/CC_0200"))
    end)
  end)
}
```

QEMU emulated VirtIO devices and IO-MMU on amd64

QEMU does not route VirtIO devices through the IO-MMU per default. To use QEMU emulated VirtIO devices add the `disable-legacy=on,disable-modern=off,iommu_platform=on` flags to the option list of the device. The `e1000e` card in the network example above can be replaced with an `virtio-net-pci` card like this:

```
-device virtio-net-pci,disable-legacy=on,disable-modern=off,
      iommu_platform=on,netdev=net0
```

For more information on VirtIO devices and their options see <https://wiki.qemu.org/Features/VT-d>.

Using the uvmm monitor interface

Uvmm implements an interface with which parts of the guest's state can be queried and manipulated at runtime. This monitor interface needs to be enabled during compilation as well as during startup of uvmm. This is described in detail below.

Compiling uvmm with monitor interface support

To compile uvmm with monitor interface support pass the `CONFIG_MONITOR=y` option during the `make` step (or set in in the `Makefile.config`). This option is available on all architectures but note that the set of available monitor interface features may vary significantly between them. Also note that the monitor interface will always be disabled in release mode, i.e. if `CONFIG_RELEASE_MODE=y`.

Enabling the monitor interface at runtime

When starting a uvmm instance from inside a `ned` script using the `uvmm.start_vm` function, the `mon` argument controls whether the monitor interface is enabled at runtime. There are three cases to distinguish:

- `mon=true` (default): The monitor interface is enabled but no server implementing the client side of the monitor interface is started. The monitor interface can still be utilized via `cons` but no readline functionality will be available.
- `mon='some_binary'`: If a string is passed as the value of `mon`, the monitor interface is enabled and the string is interpreted as the name of a server binary which implements the client side of the monitor interface. This server is automatically started and has access to a `vcon` capability named `mon` at startup through which it can make use of the monitor interface. Unless you have written your own server you should specify `'uvmm_cli'` which is a server implementing a simple readline interface.
- `mon=false`: The monitor interface is disabled at runtime.

Using the monitor interface

If the monitor interface was enabled you can connect to it via `cons` under the name `mon<n>` where `<n>` is a unique integer for every uvmm instance that is started with the monitor interface enabled (numbered starting from one in order of corresponding `uvmm.start_vm` calls). If `mon='uvmm_cli'` was specified, readline functionality such as command completion and history will be available. Enter a command followed by enter to run that command. To obtain a list of all available commands issue the `help` command, to obtain usage information for a specific command `foo` issue `help foo`.

Note

Some commands will modify the guests state. Since it should be obvious to which ones this applies this is usually not specifically highlighted. Exercise reasonable caution.

Using the guest debugger

The guest debugger provides monitoring functionality akin to a very bare-bone GDB interface, e.g. guest RAM and page table dumping, breakpointing and single stepping. Additional functionality might be added in the future.

Note

The guest debugger is currently still under development. The guest debugger may also not be available on all architectures. To check whether the guest debugger is available check if `help dbg` returns usage information.

If the guest debugger is available, you have to manually load it at runtime using the monitor interface. This saves resources if the guest debugger is not used. To enable the guest debugger, issue the `dbg on` monitor command. Once enabled, the guest debugger can not be disabled again.

To list available guest debugger subcommands, issue `dbg help` after `dbg on`.

Note

When using SMP, most guest debugger subcommands require you to explicitly specify a guest vcpu using an index starting from zero.

4.7.1 RAM configuration

RAM configuration for uvmm

Without a memory node in the device tree

- setup default RAM for guest VM.
- RAM starts either
 - at base-address which defaults to 0x0 or the base address value set via the -b cmdline option or
 - in case of identity mapping at the host-physical address of the dataspace allocated for the RAM

With a memory node in the device tree

The memory node needs at least the properties `device_type` and `l4vmm,dscap`:

```
memory@0 {  
    device_type = "memory";  
    l4vmm,dscap = "ram";  
}
```

Where the given `l4vmm,dscap` name is accessible in the capability namespace of the uvmm. If the capability is invalid, the memory node is disabled.

If memory nodes are given, but none provides valid RAM the configuration is invalid and uvmm refuses to boot.

Additional properties of the memory node are `reg` and `dma-ranges`.

The `reg` property describes the location in the guest's address space that should be backed by RAM.

The `dma-ranges` property describes the offset between guest-physical and host-physical addresses. The guest can evaluate this non-standard property to derive the correct DMA addresses to program into passed-through devices. Usage of this property **requires** modification of guest code.

Without `reg` and `dma-ranges` properties

The `reg` property is optional only in case the uvmm maps the guest's RAM into the VM under the host-physical addresses of the backing memory (`l4vmm,dscap`).

This case can be forced via the cmdline parameter `-i` and is the default for platforms without IOMMU, but with DMA capable devices on the configured vBus.

Without a `reg` property, but with a `dma-ranges` property

If the `-i` cmdline parameter is given, identity mapping is forced and the behavior is the same as in the case above. Additionally, the `dma-ranges` property is written

In case no `-i` cmdline parameter is given, the configuration is invalid and uvmm refuses to boot.

With a reg property

uvmm parses the reg property of the memory node and maps the memory into the VM to the given range(s).

If the -i cmdline parameter is set, the reg property is ignored and the memory is mapped into the VM under the corresponding host-physical addresses of the backing memory (l4vmm,dscap)

With a reg and dma-ranges property

uvmm parses the reg property of the memory node and maps the memory into the VM to the given range(s).

The dma-ranges property is filled with the corresponding host-physical addresses of the backing memory (l4vmm,dscap).

4.8 RTC driver

The RTC driver can drive various real-time clocks and provides an interface for other components, e.g., uvmm, to read and write the time.

It needs access to the hardware, depending on the clock, either via a vbus or via an I2C device.

Command Line Options

There are no command line options.

Environment

Several capabilities can be used to interact with the environment:

- 'rtc' (server)
The capability with which clients talk to the service. Note that writing to the RTC is only possible when having the rtc capability with write rights, read-only clients must have the capability with read rights only.
- 'vbus'
The vbus used to access hardware for port-based clock on X86 and pl031 on arm.
The vbus is also needed for receiving the inhibitor signal from IO.
- 'ds3231' (optional)
The capability that provides access to an I2C DS3231 device. The capability needs to be created with type 1 (i2c-device).

4.9 Mag, the GUI Multiplexer

Mag is the default multiplexer for graphics hardware. Mag is a Nitpicker derivate that allows secure multiplexing of the graphics and input hardware among multiple applications and multiple complete windowing environments. It is not, and does not attempt to be, a fully-fledged window manager.

Capabilities

None.

Command Line Options

The following command line options are supported:

```
<$FILE.lua | libmag-$LIBNAME.so>
```

As Mag's only command line option it supports loading additional plugins via the application's command line.

Can be used multiple times.

File path in ROM namespace.

- Plugins must be either a Lua file or a shared library. Shared libraries must be named libmag-\$LIBNAME.so.

Session Factory

As a simple nitpicker clone Mag supports the so-called Xray mode. This mode displays all session labels and draws a colored frame around them. The session that currently has the input focus is highlighted. The Xray mode is activated via the special keys Scroll or NEXTSONG.

Call: `create(0 [, "label|l=<LABEL>", "col=(string|hex)"])`

- "label|l=<LABEL>"

Set the session's text label to LABEL.

String value.

- The label is restricted to a length of 256 characters.

- "col=(string|hex) "

Set the session's color which is used in Xray mode to tint the session's screen area and the border drawn around it. The argument can be either one of the following letters or a hexadecimal representation of the RGB values.

The Value for this parameter can be one of the following:

- string

Define the color by the given character

Possible values are

- * r, R: Red color
- * g, G: Green color
- * b, B: Blue color
- * w, W: White color
- * y, Y: Yellow color
- * v, V: Magenta color

- hex

Define the color by its RGB values in hexadecimal representation.

Hexadecimal number without '0x' prefix.

Mag Client Session

A client with a mag client session gets access to the whole screen. The client has to allocate and manage its own buffers and has to position them on the screen on its own.

Call: `create(L4.Proto.Goos [, "default-background|dfl-bg"])`

- "default-background|dfl-bg"
Marks this session as the default background.
Flag. True if provided.

Mag Client Framebuffer Session

For a client framebuffer session mag allocates a view of the requested size and displays it at the requested coordinates on the screen.

Call: `create(L4.Proto.Goos [, "geometry|g=<GEOMETRY>", "focus", "shaded", "fixed", "barheight=<X>"])`

- "geometry|g=<GEOMETRY>"
Set the session's geometry and position on the screen. GEOMETRY is provided in an X11-style format: WIDTHxHEIGHT+X_OFFSET+Y_OFFSET.
String value.
- "focus"
Set the focus to this session.
Flag. True if provided.
- "shaded"
The window is collapsed and only the title bar is visible. The window can be expanded by clicking into the title bar with the middle mouse button. Collapsing and expanding works also independently of this option.
Flag. True if provided.
- "fixed"
The window cannot be moved on the screen.
Flag. True if provided.
- "barheight=<X>"
Set the height of the title bar in pixels.
Numerical value.

Examples

Creating a session

```
-- set label to "Linux" and use a light blue color
fb = mag_client:create(L4.Proto.Goos, "l=Linux", "col=98d9ff");
```

Requesting a framebuffer via a client framebuffer session

```
-- create a window of 640x480 pixels at position (100,100) on the screen.
fb = mag_fb:create(L4.Proto.Goos, "g=640x480+100+100");
```

4.10 Cons, the Console Multiplexer

`cons` is an interactive multiplexer for console input and output. It buffers the output from different L4 clients and allows to switch between them to redirect input.

Multiplexers and Frontends

`cons` is able to connect multiple clients to multiple console I/O servers. All clients are connected to all configured multiplexers

Multiplexers and frontends come in pairs where the actual I/O is handled by the frontend. From the point-of-view of `cons`, a frontend consists of an IPC channel to a server that speaks an appropriate server protocol. By default the `L4.Env.log` capability is used. Only frontends that speak the `L4::Vcon` protocol are supported.

A multiplexer handles and routes the I/O of each client to and from its respective frontend.

Each client's console settings (e.g. color, visibility) apply to all multiplexers and cannot be changed individually. A user can connect to all clients through all multiplexers. It is not possible to assign individual clients to distinct multiplexers.

Client sessions

For clients `cons` implements the `L4::Vcon` and the Virtio console interface. A client session can be requested through a `create` call to `cons`'s factory. `cons` binds its factory capability to the `cons` capability. See the example below on how this can be set up.

Starting the service

The `cons` server can be started with Lua like this:

```
local log_server = L4.default_loader:new_channel()
L4.default_loader:start({
  caps = {
    cons = log_server:svr(),
  },
  log = L4.Env.log,
},
"rom/cons")
```

First an IPC gate (`log_server`) is created which is used between the `cons` server and a client to request a new session. The server side is assigned to the mandatory `cons` capability of `cons`. This example explicitly assigns the kernel's log capability (`L4.Env.log`) to `cons`'s `log` capability in order to allow `cons` to provide input and output for its clients. The default log factory (usually provided by `moe`) doesn't provide input capabilities.

Command Line Options

cons accepts the following command line switches:

- `-a, --show-all`
Initially show output from all clients.
- `-B <size>, --defaultbufsize <size>`
Default buffer size per client in bytes. Default: 40960
- `-c <client>, --autoconnect <client>`
Automatically connect to the client with the given name. That means that output of this client will be visible and input will be routed to it.
- `-f <cap>, --frontend <cap>`
Set the frontend for the current multiplexer. Output for the multiplexer is then sent to the capability with the given name `<cap>`. The server connected to the capability needs to understand the [L4::Vcon](#) protocol.
- `-V <cap>, --virtio-device-frontend <cap>`
Set a virtio-console frontend for the current multiplexer. This behaves identical to the `--frontend` option except that this frontend implements a virtio-console device that –for example– can be connected to a virtio proxy in uvmm.

Example lua:

```
-- use 'cons_svr' to create client sessions
local cons_svr = loader:new_channel()
-- connect 'virtio_device' with a virtio driver (e.g. Uvmm)
local virtio_device = loader:new_channel()
loader:start(
{
  caps = {
    virtio = virtio_device,
    cons = cons_svr:svr(),
  },
  log = {"cons", "blue"},
}, "cons -m guests -V virtio")
```

- `-k, --keep`
Keep the console buffer when a client disconnects.
- `-l, --no-line-buffering`
By default, merge the client output to entire lines. If the client writes characters without a final newline, the following client output is merged with the current line content. Specifying this switch disables the line buffered mode by default.
- `--line-buffering-ms <timeout>`
Timeout in milliseconds before buffered client output is written even without a newline. Default value is 50.
- `-m <prompt name>, --mux <prompt name>`
Add a new multiplexer named `<prompt name>`. This is necessary if output should be sent to different frontends. This option must be used in conjunction with the `-f` frontend option
- `-n, --defaultname`
Default name for the multiplexer prompt. Default: `cons`.
- `-t, --timestamp`
Prefix the output with timestamps.

Connecting a client

```
create(backend_type, ["client_name"], ["color"], ["option"] [, "option"] ...)
```

- `backend_type`

The type of backend that should be created for the client. The type is a positive integer and currently the following types are supported:

- `L4.Proto.Log: L4::Vcon` client
- `1`: Virtio console client

`cons` accepts the following per-client options:

- `bufsz=n`

Use a buffer of `n` bytes for this client, deviating from the default buffer size.

- `keep / no-keep`

The console buffer is kept / thrown away when the client disconnects.

- `key=<key>`

Assign `<key>` as keyboard shortcut to this client.

- `line-buffering / no-linux-buffering`

Line buffering is enabled / disabled for this client.

- `show / hide`

Output from this client is initially shown / hidden.

- `timestamp / no-timestamp`

Do / do not prefix the output of this client with timestamps.

Chapter 5

uvmm_dtg The device tree generator for Uvmm

A virtual machine in Uvmm is configured with a device tree that contains information about the VMs resources, memory layout, virtual CPUs and peripheral devices.

Uvmm_dtg is a tool to generate such a device tree at runtime according to its command line.

Capabilities

- dt

The dataspace that the device tree is put into.

Command Line Options

`<file | -->`

-- prints to stdout. On [L4Re](#), the string given as `<file>` is interpreted as a named capability which needs to be backed by a sufficiently large Dataspace. On Linux, a file with the given name is created. In both cases, uvmm_dtg will output into the named file.

String value.

- -h, --help

Show help.

Flag. True if provided.

- --arch `<target architecture>`

Select the target architecture.

Possible values for `<target architecture>` are x86, x86_64, arm32, arm64, mips32, mips64

- --format `<format>`

Select the output format.

Possible values for `<format>` are

- txt: The device tree will be printed as plain text (dts).
- bin: The device tree will be output as binary (dtb).
- --mem-base <membase>
Configure the start of the memory distribution. membase can be defined in both decimal and hex notations. uvmm_dtg rounds the given base up to the platforms page size.
This value can be overridden by memory devices with fixed addresses.
Numerical value.
- --device <devicename:[Option1,Option2=value,Option3=value,...]>
This configures a device.
To get a list of supported devices, use --device help.
To get help for a specific device, use --device devicename:help.
String value.

Examples

Usage in L4Re

Example lua script for Ned:

```
-- Create DS holding device tree
local dt = L4.Env.user_factory:create(L4.Proto.Dataspace, 4 * 1024):m("rw");

-- Start the generator
L4.default_loader:start(
{
  caps = { dt = dt },
}, "rom/uvmm_dtg dt"):wait();

-- Start uvmm
vmm.start_vm
{
  ...
  ext_caps = { dt = dt },
  fdt = "dt",
  ...
}
```

Please notice the :wait() when starting uvmm_dtg. This makes Ned pause until uvmm_dtg has exited and put the device tree into the dataspace such that Uvmm can commence.

Chapter 6

Bootstrap, the L4 kernel bootstrapper

Bootstrap Command Line Options

`bootstrap` and the kernel can be configured through command line switches. `bootstrap` is responsible for parsing both command lines: `bootstrap` options are the ones directly given to `bootstrap`, whereas kernel options are those directly given to the kernel, respectively.

When using Multiboot boot, the first module directly after `bootstrap` is considered the kernel: An example using GRUB2 might look like this:

```
multiboot /path/to/bootstrap bootstrap -bs-boolean-opt -bs-opt-with-argument=foo
module /path/to/fiasco fiasco -kernel-opt-with-argument=bar -kernel-boolean-opt
```

Note

The exact way to provide the command line to `bootstrap` is platform-dependent. On platforms supporting booting via Multiboot Specification the command line can be specified in the boot configuration (currently x86/amd64 only, e.g. using GRUB) whereas other platforms need the command line to be compiled into the `bootstrap` binary.

Platforms utilising flattened device trees usually also allow specifying a command line through the DT. This mechanism is **not** currently supported in `bootstrap`!

Note

`bootstrap` will ignore options it does not understand. For most cases, this also holds true if an option's arguments cannot be understood.

bootstrap options

Command line options directly understood by `bootstrap` itself are as follows (passed via `bootstrap` command line):

- `-comirq=<irqno>` (x86/amd64 only)

If serial logging is enabled (default on), `<irqno>` defines which IRQ to use for serial port communication. This option is ignored if `-noserial` is also specified (see below).

- `-comport=<portspec>` (x86/amd64 only)

If serial logging is enabled (default on), `<portspec>` defines which serial port to use, being one of:

- `<number>`
Use legacy port `<number>`, e.g. use `-comport=1` for the port commonly known as *COM1*, or
- `pci:<card>:<port>`
Use serial port number `<port>` at PCI card number `<card>`, e.g. use `-comport=pci:0:1` for the second port on the first PCI card.
- `pci:probe`
Use this to have `bootstrap` autodiscover all PCI serial lines. On each discovered line a message will be displayed, telling the correct `<portspec>` to be given to use the respective line.

Note

`bootstrap` does not support specifying the serial port's baudrate and always uses 115200 bps.

- `-noserial`

Disable serial logging.

- `-wait`

Wait for key press at early startup.

Note

This is not to be confused with the kernel's `-wait` option, see below.

- `-maxmem=<mbytes>`

Limit the available memory to at most `<mbytes>` MiB.

- `-mem=<size>@<offset>`

Add a region of memory of `<size>` at `<offset>` to the system's memory map. Both `<size>` and `<offset>` may be suffixed by either G, M, or K/k to denote GiB, MiB, or KiB, respectively.

This option may be specified multiple times. If the option is not given, a platform-specific method for determining the memory layout will be used, if available.

- `-presetmem=<intval>`

Initialise memory regions with `<intval>` before starting the kernel.

- `-modaddr=<paddr>`

Relocate modules to the physical address `<paddr>`. Use this when utilising a version of GRUB that lacks support for the `modaddr` command.

Kernel Options

Command line options for the kernel (passed on kernel command line, i.e. to first module) `bootstrap` understands are as follows.

Note

Availability of individual options might depend on used platform and/or actual kernel configuration.

- `-wait`
Enter debugger directly after startup, prior to executing any task.
- `-serial_esc`
Enable entering the debugger over serial line by pressing `Esc`. This option also activates `-serial_input`.
- `-serial_input`
Allow the kernel to read from the serial line either triggered by the UART receive interrupt or (if that is not possible) by polling the UART receive register from the kernel timer interrupt handler. This is required for Vlog clients to receive input.
- `-noserial`
Disable serial logging.

Note

If this is given as kernel command line argument, it does not affect `bootstrap` but only the kernel.

- `-noscreen`
Disable VGA console.
- `-esc`
Enable entering the debugger by pressing `Esc` on attached keyboard.
- `-nojdb`
Disable the kernel debugger.
- `-nohlt`
Enable quirk for broken HLT instruction.
- `-apic`
Use Advanced Programmable Interrupt Controller (APIC) if available and known to be well-behaving.
- `-loadcnt`
Use load counter for performance counting.
- `-watchdog`
Enable watchdog timer.
- `-irq0`
Allow IRQ 0 to be used by userland. This enables some sanity checks to ensure IRQ 0 is not used by the kernel, e.g. for profiling or scheduling purposes. This only has an effect on x86.
- `-nosfn`
Disable SFN (special fully nested) mode of interrupt controller. This only has an effect on x86 with PIC8259 interrupt controller.
- `-jdb_never_stop`
Prevent system from stopping to enter JDB. This only has an effect on x86.

- `-kmemsize=<KB>`
Reserve `<KB>` KiB of memory for the kernel.
- `-tbuf_entries=<number>`
Specify the `<number>` of trace buffer entries.
- `-out_buf=<length>`
Specify length of console buffer to be `<length>` bytes.
- `-jdb_cmd=<ctrlseq>`
Execute JDB command sequence `<ctrlseq>` right after start-up. If `-wait` is also given, `<ctrlseq>` is executed right before entering JDB.

Module options

Bootstrap supports module attributes for `sigma0` and the `roottask`. They need to be specified in `modules.list`, e.g.:

```
sigma0[attr:nodes=4-7] ...
```

Attributes are not supported when using multi-boot on platforms that support it. The following attributes are supported:

- `nodes`

This is a colon separated list of AMP node ranges. A range can also be a single number. Examples:

- `nodes=1` – Only node 1
- `nodes=1-3` – Nodes 1 to 3 (inclusive)
- `nodes=0:2-3` – Nodes 0, 2 and 3

If not present, the `sigma0/roottask` module is applicable to all AMP nodes.

- `reloc`

Normally the `sigma0` or `roottask` images are loaded at the preferred load address if the RAM is available at the desired location. If this is not possible, they will be relocated to some free RAM region. Setting the "reloc" module attribute to a non-empty string will always request the dynamic relocation.

This attribute can be used on no-MMU systems to maximize the size of contiguous free RAM regions.

Chapter 7

Deprecated List

Global `L4::Rcv_endpoint::bind_thread (Ipc::Cap< Thread > t, I4_umword_t label)`

Use `bind_snd_destination()` instead.

Global `L4_CAP_SIZE`

Superseded by `L4_CAP_OFFSET`.

Global `I4_kip_clock_lw (I4_kernel_info_t const *kip) L4_NOTHROW`

Use `I4_kip_clock()` instead.

Global `L4Re::Util::Registry_server< LOOP_HOOKS >::Registry_server (I4_utcb_t *, L4::Cap< L4::Thread > server, L4::Cap< L4::Factory > factory)`

Note that this variant of the constructor is deprecated, please do not supply the UTCB pointer, it's not used.

Global `I4util_kip_for_each_feature (s)`

Use `I4_kip_for_each_feature()`.

Global `I4util_kip_kernel_has_feature (I4_kernel_info_t const *k, char const *str)`

Use `I4_kip_kernel_has_feature()`.

Global `I4util_micros2I4to (I4_uint64_t us) L4_NOTHROW`

Use `I4_timeout_from_us()`.

Global `start_virtio_switch (ports, prio, cpus, switch_type, ext_caps)`

This function exists for backwards compatibility reasons and calls `start_virtio_switch_tbl` with an appropriate `options` table

Chapter 8

Topic Index

8.1 Topics

Here is a list of all topics with brief descriptions:

| | |
|---|-----|
| Base API | 149 |
| Basic Macros | 152 |
| Fiasco extensions | 157 |
| Kernel Debugger | 161 |
| Kernel Information Dump | 169 |
| Kernel Tracing | 170 |
| Flexpages | 175 |
| C++ IPC Interface Definition | 193 |
| Internal Helpers | 194 |
| Cache Consistency | 194 |
| Memory related | 198 |
| Error codes | 208 |
| Object Invocation | 210 |
| Message Items | 229 |
| Attributes and additional permissions for object send items | 235 |
| Timeouts | 236 |
| Error Handling | 245 |
| Realtime API | 251 |
| Message Tag | 251 |
| Virtual Registers (UTCBs) | 264 |
| Message Registers (MRs) | 269 |
| Exception registers | 270 |
| Buffer Registers (BRs) | 273 |
| Thread Control Registers (TCRs) | 275 |
| ARM Virtual Registers (UTCB) | 276 |
| ARM64 Virtual Registers (UTCB) | 276 |
| AMD64 Virtual Registers (UTCB) | 277 |
| x86 Virtual Registers (UTCB) | 277 |
| RISC-V Virtual Registers (UTCB) | 278 |
| Kernel Objects | 278 |
| IPC-Gate API | 281 |
| DMA space | 286 |
| L4 kernel object type information | 286 |
| Factory | 288 |

| | |
|--|-----|
| Virtual Machines | 300 |
| VM API for SVM | 300 |
| VM API for VMX | 301 |
| VM API for TZ | 321 |
| Interrupt controller | 321 |
| IRQs | 338 |
| Platform Control C API | 353 |
| Scheduler | 359 |
| Kernel-provided semaphore | 367 |
| Task | 369 |
| Thread | 382 |
| Thread control | 405 |
| vCPU API | 411 |
| Thread groups | 416 |
| Virtual Console | 418 |
| Kernel Interface Page | 433 |
| Memory descriptors (C version) | 440 |
| Capabilities | 445 |
| Memory Operations | 449 |
| Integer Types | 452 |
| EDID parsing functionality | 454 |
| IO interface | 458 |
| IPC Helpers | 466 |
| IRQ handling library | 467 |
| Interface using direct functionality. | 468 |
| Interface using direct functionality. | 473 |
| Interface for asynchronous ISR handlers. | 475 |
| Interface for asynchronous ISR handlers with a given IRQ capability. | 477 |
| L4 IPC Opcodes | 478 |
| L4 VIRTIO Interface | 483 |
| L4 VIRTIO Transport Layer | 483 |
| L4 VIRTIO Block Device | 492 |
| L4 VIRTIO Input Device | 494 |
| L4 VIRTIO Network Device | 494 |
| L4 Vbus functions | 495 |
| L4vbus GPIO functions | 505 |
| L4vbus PCI functions | 516 |
| L4vbus power management functions | 522 |
| L4Re C Interface | 524 |
| L4Re Util C Interface | 526 |
| Dataspace interface | 526 |
| Debug interface | 532 |
| DMA Space Interface | 533 |
| Event interface | 537 |
| Log interface | 540 |
| Memory allocator | 543 |
| Namespace interface | 548 |
| Parent interface | 552 |
| Region map interface | 552 |
| Capability allocator | 570 |
| Kumem allocator utility | 571 |
| Video API | 572 |
| Initial Environment | 580 |
| L4Re C++ Interface | 585 |
| L4Re Util C++ Interface | 587 |
| L4Re Capability API | 588 |

| | |
|--|-----|
| Kumem utilities | 591 |
| Console API | 592 |
| Debugging API | 593 |
| L4Re ELF Auxiliary Information | 593 |
| Event API | 596 |
| Auxiliary data | 596 |
| Logging interface | 597 |
| Name-space API | 598 |
| Parent API | 598 |
| L4Re Protocol identifiers | 599 |
| Region map API | 600 |
| Video API | 601 |
| C++ Exceptions | 602 |
| Vbus API | 603 |
| L4SHM-based ring buffer implementation | 604 |
| Sender | 605 |
| Receiver | 605 |
| Internal | 605 |
| Shared Memory Library | 607 |
| Chunks | 613 |
| Producer | 618 |
| Consumer | 621 |
| Signals | 626 |
| Producer | 629 |
| Consumer | 630 |
| Sigma0 API | 634 |
| Internal constants | 639 |
| Small C++ Template Library | 640 |
| The L4Re IPC Framework | 644 |
| Server-Side IPC framework | 644 |
| Utility Functions | 646 |
| CPU related functions | 653 |
| IA32 Port I/O API | 656 |
| Timestamp Counter | 662 |
| Atomic Instructions | 669 |
| Internal functions | 687 |
| Bit Manipulation | 687 |
| ELF binary format | 694 |
| Kernel Interface Page API | 719 |
| Comfortable Command Line Parsing | 721 |
| Random number support | 723 |
| Low-Level Thread Functions | 724 |
| Virtio Net Switch | 724 |
| vCPU Support Library | 725 |
| Extended vCPU support | 732 |

Chapter 9

Namespace Index

9.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

| | | |
|----------------------------------|--|-----|
| cxx | Our C++ library | 735 |
| cxx::Bits | Internal helpers for the cxx package | 741 |
| Enum_bitops | Mechanism to opt-in for enum bitwise operators | 742 |
| Enum_bitops_impl | Bitwise operators on enumeration types | 742 |
| L4 | | |
| | L4 low-level kernel interface | 743 |
| L4::lpc | IPC related functionality | 756 |
| L4::lpc::Msg | IPC Message related functionality | 764 |
| L4::lpc_svr | Helper classes for L4::Server instantiation | 771 |
| L4::Typeid | Definition of interface data-type helpers | 772 |
| L4::Types | | |
| | L4 basic type helpers for C++ | 772 |
| L4Re | | |
| | L4Re C++ Interfaces | 774 |
| L4Re::Util | Documentation of the L4 Runtime Environment utility functionality in C++ | 799 |
| L4Re::Vfs | Virtual file system for interfaces in POSIX libc | 816 |
| L4vbus | C++ interface of the Vbus API. | 817 |
| L4virtio | L4-VIRTIO Transport C++ API | 818 |

Chapter 10

Hierarchical Index

10.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

| | |
|--|-----|
| cxx::arith::Ld< V > | 821 |
| cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc > | 842 |
| cxx::Base_slab< sizeof(Type), L4_PAGESIZE, 2, New_allocator > | 842 |
| cxx::Slab< Type, Slab_size, Max_free, Alloc > | 973 |
| cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc > | 847 |
| cxx::Base_slab_static< sizeof(Type), L4_PAGESIZE, 2, New_allocator > | 847 |
| cxx::Slab_static< Type, Slab_size, Max_free, Alloc > | 978 |
| cxx::Bitfield< T, LSB, MSB > | 852 |
| cxx::Bitfield< T, LSB, MSB >Value_base< TT > | 861 |
| cxx::Bitfield< T, LSB, MSB >Value< Base_type & > | 860 |
| cxx::Bitfield< T, LSB, MSB >Value< Base_type volatile & > | 860 |
| cxx::Bitfield< T, LSB, MSB >Value< Base_type const > | 860 |
| cxx::Bitfield< T, LSB, MSB >Value_unshifted< Base_type & > | 862 |
| cxx::Bitfield< T, LSB, MSB >Value_unshifted< Base_type volatile & > | 862 |
| cxx::Bitfield< T, LSB, MSB >Value_unshifted< Base_type const > | 862 |
| cxx::Bitfield< T, LSB, MSB >Value< TT > | 860 |
| cxx::Bitfield< T, LSB, MSB >Value_unshifted< TT > | 862 |
| cxx::Bitmap_base | 868 |
| cxx::Bitmap< BITS > | 864 |
| cxx::Bitmap_base::Bit | 881 |
| cxx::Bitmap_base::Char< BITS > | 881 |
| cxx::Bitmap_base::Word< BITS > | 882 |
| cxx::Bits::Avl_map_get_key< KEY_TYPE > | 883 |
| cxx::Bits::Avl_set_get_key< KEY_TYPE > | 884 |
| cxx::Bits::Avl_set_iter< Node, Key, Node_op > | 884 |
| cxx::Bits::Avl_set_iter< _Node, Const_item_type, Fwd > | 884 |
| cxx::Bits::Avl_set_iter< _Node, Const_item_type, Rev > | 884 |
| cxx::Bits::Avl_set_iter< _Node, Item_type, Fwd > | 884 |
| cxx::Bits::Avl_set_iter< _Node, Item_type, Rev > | 884 |
| cxx::Bits::Avl_set_iter< Node, Non_const_key, Node_op > | 884 |
| cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY > | 888 |
| cxx::Avl_map< Region, Hdlr, cxx::Lt_functor, Alloc > | 822 |
| cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >Node | 898 |

| | |
|--|------|
| cxx::Bits::Base_avl_set< ITEM_TYPE, Lt_func< ITEM_TYPE >, New_allocator, Bits::Avl_set_get_↵ key< ITEM_TYPE > > | 888 |
| cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC > | 828 |
| cxx::Bits::Base_avl_set< Pair< KEY_TYPE, DATA_TYPE >, Lt_func< KEY_TYPE >, New_allocator, Bits::Avl_map_get_key< KEY_TYPE > > | 888 |
| cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC > | 822 |
| cxx::Bits::Basic_list< POLICY > | 900 |
| cxx::H_list< Timeout > | 926 |
| cxx::H_list< Observer > | 926 |
| cxx::H_list< cxx::Base_slab::Slab_i > | 926 |
| cxx::S_list< T, POLICY > | 970 |
| cxx::H_list< T, POLICY > | 926 |
| cxx::Bits::Basic_list< Bits::Basic_list_policy< T, H_list_item_t< T > > > | 900 |
| cxx::H_list< T, Bits::Basic_list_policy< T, H_list_item_t< T > > > | 926 |
| cxx::H_list_t< T > | 937 |
| cxx::Bits::Basic_list< Bits::Basic_list_policy< T, S_list_item > > | 900 |
| cxx::S_list< T, Bits::Basic_list_policy< T, S_list_item > > | 970 |
| cxx::S_list< T, POLICY > | 970 |
| cxx::Bits::Basic_list< Bits::Basic_list_policy< Weak_ref_base, H_list_item_t< Weak_ref_base > > > | 900 |
| cxx::H_list< Weak_ref_base, Bits::Basic_list_policy< Weak_ref_base, H_list_item_t< Weak_ref_↵ base > > > | 926 |
| cxx::H_list_t< Weak_ref_base > | 937 |
| cxx::Weak_ref_base::List | 996 |
| cxx::Bits::Bst< Node, Get_key, Compare > | 903 |
| cxx::Avl_tree< Entry, Names_get_key > | 832 |
| cxx::Avl_tree< _Node, GET_KEY, COMPARE > | 832 |
| cxx::Avl_tree< Node, Get_key, Compare > | 832 |
| cxx::Bits::Bst< _Node, Bits::Avl_map_get_key< KEY_TYPE >, Lt_func< KEY_TYPE > > > | 903 |
| cxx::Avl_tree< _Node, Bits::Avl_map_get_key< KEY_TYPE >, Lt_func< KEY_TYPE > > > | 832 |
| cxx::Bits::Bst< _Node, Bits::Avl_set_get_key< ITEM_TYPE >, Lt_func< ITEM_TYPE > > > | 903 |
| cxx::Avl_tree< _Node, Bits::Avl_set_get_key< ITEM_TYPE >, Lt_func< ITEM_TYPE > > > | 832 |
| cxx::Bits::Bst_node | 916 |
| cxx::Avl_tree_node | 840 |
| cxx::Bits::Direction | 918 |
| cxx::Bits::Smart_ptr_list< ITEM > | 920 |
| cxx::Bits::Smart_ptr_list_item< T, STORE_T > | 923 |
| cxx::Bits::Smart_ptr_list_item< T, cxx::Ref_ptr< T > > | 923 |
| cxx::Ref_obj_list_item< T > | 962 |
| cxx::Elide_dtor< T > | 925 |
| cxx::Error | 926 |
| cxx::H_list_item_t< ELEM_TYPE > | 934 |
| cxx::H_list_item_t< void > | 934 |
| L4::lpc_svr::Timeout | 1278 |
| cxx::H_list_item_t< Weak_ref_base > | 934 |
| cxx::Weak_ref_base | 993 |
| cxx::Weak_ref< T > | 991 |
| cxx::List< D, Alloc > | 941 |
| cxx::List< D, Alloc >Iter | 943 |
| cxx::List_alloc | 944 |
| cxx::List_item | 948 |
| cxx::List_item::Iter | 951 |
| cxx::List_item::T_iter< E > | 953 |
| cxx::List_item::T_iter< T, Poly > | 953 |
| cxx::Lt_func< Obj > | 955 |

| | |
|--|------|
| cxx::New_allocator< _Type > | 955 |
| cxx::Nothrow | 957 |
| cxx::Pair< First, Second > | 957 |
| cxx::Pair_first_compare< Cmp, Typ > | 960 |
| cxx::Ref_ptr< T, CNT > | 964 |
| cxx::Ref_ptr< L4Re::Vfs::File > | 964 |
| cxx::Ref_ptr< Mount_tree > | 964 |
| cxx::Ref_ptr< T, CNT > | 964 |
| cxx::Result< T > | 969 |
| cxx::static_vector< T, IDX > | 982 |
| cxx::String | 983 |
| L4Re::Util::Names::Name | 1860 |
| Elf32_Auxv | 1000 |
| Elf32_Dyn | 1001 |
| Elf32_Ehdr | 1002 |
| Elf32_Phdr | 1005 |
| Elf32_Rel | 1006 |
| Elf32_Rela | 1007 |
| Elf32_Shdr | 1008 |
| Elf32_Sym | 1010 |
| Elf64_Auxv | 1011 |
| Elf64_Dyn | 1012 |
| Elf64_Ehdr | 1013 |
| Elf64_Phdr | 1015 |
| Elf64_Rel | 1017 |
| Elf64_Rela | 1017 |
| Elf64_Shdr | 1018 |
| Elf64_Sym | 1020 |
| L4::Alloc_list | 1027 |
| L4::Basic_registry | 1032 |
| L4Re::Util::Object_registry | 1868 |
| L4::Cap_base | 1044 |
| L4::Cap< A > | 1038 |
| L4::Cap< L4Re::Namespace > | 1038 |
| L4::Cap< L4Re::Dataspace > | 1038 |
| L4::Cap< L4::Vcon > | 1038 |
| L4::Cap< L4::Semaphore > | 1038 |
| L4::Cap< L4Re::Rm > | 1038 |
| L4::Cap< L4::Irq > | 1038 |
| L4::Cap< L4::Thread > | 1038 |
| L4::Cap< L4::Factory > | 1038 |
| L4::Cap< L4Re::Video::Goos > | 1038 |
| L4::Cap< L4vbus::Vbus > | 1038 |
| L4::Cap< L4virtio::Device > | 1038 |
| L4::Cap< L4::Debugger > | 1038 |
| L4::Smart_cap< T, Smart_count_cap< L4_FP_ALL_SPACES > > | 1431 |
| L4::Smart_cap< T, Smart_count_cap< L4_FP_DELETE_OBJ > > | 1431 |
| L4::Cap< T > | 1038 |
| L4::Smart_cap< T, SMART > | 1431 |
| L4::Epiface | 1077 |
| L4::Epiface_t0< IFACE, L4::Epiface > | 1085 |
| L4::Epiface_t< Derived, IFACE, BASE, bool > | 1082 |
| L4::Epiface_t0< void, Epiface > | 1085 |
| L4::Irqep_t< Irq_object > | 1307 |
| L4::Irqep_t< Host_irq > | 1307 |
| L4virtio::Svr::Virtio_gpio< Request_handler, Epiface >Host_irq | 2265 |
| L4virtio::Svr::Virtio_i2c< Request_handler, Epiface >Host_irq | 2277 |

| | |
|--|------|
| L4virtio::Svr::Virtio_rng< Rnd_state, Epiface >Host_irq | 2288 |
| L4virtio::Svr::Virtio_spi< Spi_request_handler, Epiface >Host_irq | 2299 |
| L4::Irqep_t< Del_cap_irq > | 1307 |
| L4::Irqep_t< Kick_irq > | 1307 |
| L4::Irqep_t< Derived, BASE, bool > | 1307 |
| L4::Epiface_t0< L4::Kobject, L4::Epiface > | 1085 |
| L4::Epiface_t< Null_handler, L4::Kobject > | 1082 |
| L4::Epiface_t0< L4virtio::Device, L4::Epiface > | 1085 |
| L4::Epiface_t< Block_dev< Ds_data >, L4virtio::Device > | 1082 |
| L4::Epiface_t< Virtio_gpio< Request_handler, L4virtio::Device >, L4virtio::Device > | 1082 |
| L4virtio::Svr::Virtio_gpio< Request_handler, Epiface > | 2261 |
| L4::Epiface_t< Virtio_i2c< Request_handler, L4virtio::Device >, L4virtio::Device > | 1082 |
| L4virtio::Svr::Virtio_i2c< Request_handler, Epiface > | 2273 |
| L4::Epiface_t< Virtio_rng< Rnd_state >, L4virtio::Device > | 1082 |
| L4virtio::Svr::Virtio_rng< Rnd_state, Epiface > | 2284 |
| L4::Epiface_t< Virtio_spi< Spi_request_handler, L4virtio::Device >, L4virtio::Device > | 1082 |
| L4virtio::Svr::Virtio_spi< Spi_request_handler, Epiface > | 2294 |
| L4::Epiface_t< Virtio_net, L4virtio::Device > | 1082 |
| Virtio_net | 2423 |
| L4virtio_port | 2357 |
| L4::Epiface_t0< L4::Factory, L4::Epiface > | 1085 |
| L4::Epiface_t< Switch_factory, L4::Factory > | 1082 |
| Switch_factory | 2392 |
| L4::Epiface_t0< Virtio_net_switch::Statistics_if, L4::Epiface > | 1085 |
| L4::Epiface_t< Stats_reader, Virtio_net_switch::Statistics_if > | 1082 |
| L4::Epiface_t0< RPC_IFACE, BASE > | 1085 |
| L4::Server_object | 1417 |
| L4::Server_object_t< Kobject > | 1422 |
| L4::Irq_handler_object | 1303 |
| L4::Server_object_t< IFACE, L4::Server_object > | 1422 |
| L4::Server_object_x< Derived, IFACE, BASE > | 1427 |
| L4::Server_object_t< IFACE, BASE > | 1422 |
| L4::Server_object_x< Derived, IFACE, BASE > | 1427 |
| L4::Epiface_t0< IFACE, BASE > | 1085 |
| L4::Epiface_t0< void, BASE > | 1085 |
| L4::Exception_tracer | 1090 |
| L4::Base_exception | 1030 |
| L4::Invalid_capability | 1123 |
| L4::Runtime_error | 1393 |
| L4::Bounds_error | 1035 |
| L4::Com_error | 1057 |
| L4::Element_already_exists | 1070 |
| L4::Element_not_found | 1074 |
| L4::Out_of_memory | 1353 |
| L4::Unknown_error | 1558 |
| L4::Factory::Lstr | 1102 |
| L4::Factory::Nil | 1104 |
| L4::Factory::S | 1104 |
| L4::IOModifier | 1134 |
| L4::lpc::Array_in_buf< ELEM_TYPE, LEN_TYPE, MAX > | 1137 |
| L4::lpc::Array_ref< ELEM_TYPE, LEN_TYPE > | 1140 |
| L4::lpc::Array_ref< A, LEN > | 1140 |
| L4::lpc::Array< A, LEN > | 1134 |
| L4::lpc::Array< A, LEN > & | 1134 |
| L4::lpc::Array_ref< char const, unsigned long > | 1140 |
| L4::lpc::Array< char const, unsigned long > | 1134 |

| | |
|---|------|
| L4::lpc::Array_ref< ELEM_TYPE, Array_len_default > | 1140 |
| L4::lpc::Array< ELEM_TYPE, LEN_TYPE > | 1134 |
| L4::lpc::Array_ref< X, LEN_TYPE > | 1140 |
| L4::lpc::Array< X, LEN_TYPE > | 1134 |
| L4::lpc::As_value< T > | 1141 |
| L4::lpc::Call | 1142 |
| L4::lpc::Call_t< RIGHTS > | 1143 |
| L4::lpc::Call_zero_send_timeout | 1144 |
| L4::lpc::Cap< T > | 1145 |
| L4::lpc::Gen_fpage | 1150 |
| L4::lpc::Rcv_fpage | 1209 |
| L4::lpc::Snd_fpage | 1220 |
| L4::lpc::In_out< T > | 1153 |
| L4::lpc::Istream | 1164 |
| L4::lpc::Iostream | 1154 |
| L4::lpc::Msg::Cnt_val_ops< MTYPE, DIR, CLASS > | 1175 |
| L4::lpc::Msg::Cnt_val_ops< Detail::_Plain< T >type, DIR, CLASS > | 1175 |
| L4::lpc::Msg::Cnt_val_ops< Detail::_Plain< T >type, typename Direction< T >type, typename Class< typename Detail::_Plain< T >type >type > | 1175 |
| L4::lpc::Msg::Cnt_val_ops< Detail::_Plain< typename _Elem< A * >arg_type >type, typename Direction< A * >type, typename Class< typename Detail::_Plain< A * >type >type > | 1175 |
| L4::lpc::Msg::Cnt_val_ops< Detail::_Plain< typename _Elem< A const * >arg_type >type, typename Direction< A const * >type, typename Class< typename Detail::_Plain< A const * >type >type > | 1175 |
| L4::lpc::Msg::Cnt_val_ops< Detail::_Plain< typename _Elem< Array< A, LEN > & >arg_type >type, typename Direction< Array< A, LEN > & >type, typename Class< typename Detail::_Plain< Array< A, LEN > & >type >type > | 1175 |
| L4::lpc::Msg::Cls_buffer | 1176 |
| L4::lpc::Msg::Do_rcv_buffers | 1187 |
| L4::lpc::Msg::Cls_data | 1178 |
| L4::lpc::Msg::Do_in_data | 1182 |
| L4::lpc::Msg::Do_out_data | 1184 |
| L4::lpc::Msg::Cls_item | 1179 |
| L4::lpc::Msg::Do_in_items | 1183 |
| L4::lpc::Msg::Do_out_items | 1185 |
| L4::lpc::Msg::Dir_in | 1180 |
| L4::lpc::Msg::Do_in_data | 1182 |
| L4::lpc::Msg::Do_in_items | 1183 |
| L4::lpc::Msg::Do_rcv_buffers | 1187 |
| L4::lpc::Msg::Dir_out | 1181 |
| L4::lpc::Msg::Do_out_data | 1184 |
| L4::lpc::Msg::Do_out_items | 1185 |
| L4::lpc::Msg::Elem< Array< A, LEN > & > | 1188 |
| L4::lpc::Msg::Elem< Array< A, LEN > > | 1189 |
| L4::lpc::Msg::Elem< Array_ref< A, LEN > & > | 1190 |
| L4::lpc::Msg::Svr_arg_pack< IPC_TYPE > | 1195 |
| L4::lpc::Msg::Svr_val_ops< MTYPE, DIR, CLASS > | 1195 |
| L4::lpc::Msg::Svr_val_ops< L4::lpc::Snd_fpage, Dir_in, CLASS > | 1195 |
| L4::lpc::Msg_ptr< T > | 1199 |
| L4::lpc::Opt< T > | 1200 |
| L4::lpc::Ostream | 1202 |
| L4::lpc::Iostream | 1154 |
| L4::lpc::Out< T > | 1208 |
| L4::lpc::Ret_array< T > | 1216 |
| L4::lpc::Send_only | 1217 |

| | |
|--|------|
| L4::lpc::Small_buf | 1218 |
| L4::lpc::Str_cp_in< T > | 1230 |
| L4::lpc::Varg | 1231 |
| L4::lpc::Varg_list_ref | 1240 |
| L4::lpc::Varg_list< MAX > | 1237 |
| L4::lpc::Varg_list_ref::iterator | 1242 |
| L4::lpc_svr::Compound_reply | 1254 |
| L4::lpc_svr::Default_loop_hooks | 1257 |
| L4::Server< L4::lpc_svr::Default_loop_hooks > | 1411 |
| L4Re::Util::Registry_server< LOOP_HOOKS > | 1878 |
| L4::Server< LOOP_HOOKS > | 1411 |
| L4Re::Util::Br_manager_hooks | 1823 |
| L4::lpc_svr::Default_setup_wait | 1260 |
| L4::lpc_svr::Default_timeout | 1261 |
| L4::lpc_svr::Default_loop_hooks | 1257 |
| L4Re::Util::Br_manager_hooks | 1823 |
| L4::lpc_svr::Direct_dispatch< R > | 1263 |
| L4::lpc_svr::Dbg_dispatch< R, Exc, Printer > | 1255 |
| L4::lpc_svr::Exc_dispatch< R, Exc > | 1267 |
| L4::lpc_svr::Direct_dispatch< R * > | 1265 |
| L4::lpc_svr::Ignore_errors | 1269 |
| L4::lpc_svr::Default_loop_hooks | 1257 |
| L4Re::Util::Br_manager_hooks | 1823 |
| L4Re::Util::Br_manager_timeout_hooks | 1826 |
| L4::lpc_svr::Server_iface | 1270 |
| L4::lpc_svr::Timeout_queue_hooks< Br_manager_timeout_hooks, Br_manager > | 1285 |
| L4Re::Util::Br_manager_timeout_hooks | 1826 |
| L4::lpc_svr::Br_manager_no_buffers | 1249 |
| L4::lpc_svr::Default_loop_hooks | 1257 |
| L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN > | 1285 |
| L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN > | 1285 |
| L4Re::Util::Br_manager | 1815 |
| L4Re::Util::Br_manager_hooks | 1823 |
| L4::lpc_svr::Server_iface::Mem_window | 1278 |
| L4::lpc_svr::Timeout_queue | 1282 |
| L4::Kip::Mem_desc | 1311 |
| L4::Kobject | 1322 |
| L4::Kobject_t< Arm_smccc, L4::Kobject, PROTO, Type_info::Demand_t<> > | 1334 |
| L4::Kobject_t< Debugger, Kobject, L4_PROTO_DEBUGGER > | 1334 |
| L4::Debugger | 1060 |
| L4::Kobject_t< Exception, L4::Kobject, PROTO, Type_info::Demand_t<> > | 1334 |
| L4::Kobject_t< Factory, Kobject, L4_PROTO_FACTORY > | 1334 |
| L4::Factory | 1091 |
| L4::Kobject_t< Mem_alloc, L4::Factory, L4RE_PROTO_MEM_ALLOC > | 1334 |
| L4Re::Mem_alloc | 1730 |
| L4::Kobject_t< Io_pager, L4::Kobject, PROTO, Type_info::Demand_t<> > | 1334 |
| L4::Kobject_t< Irq_eoi, L4::Kobject, PROTO, Type_info::Demand_t<> > | 1334 |
| L4::Kobject_t< Derived, L4::Kobject, L4::PROTO_ANY, Type_info::Demand_t<> > | 1334 |
| L4::Kobject_t< Meta, Kobject, L4_PROTO_META > | 1334 |
| L4::Meta | 1347 |
| L4::Kobject_t< Platform_control, Kobject, L4_PROTO_PLATFORM_CTL > | 1334 |
| L4::Platform_control | 1360 |
| L4::Kobject_t< Rcv_endpoint, Kobject, L4_PROTO_KOBJECT, Type_info::Demand_t< 1 > > | 1334 |
| L4::Rcv_endpoint | 1375 |
| L4::Kobject_2t< Irq, Triggerable, Rcv_endpoint, L4_PROTO_IRQ_SENDER > | 1326 |

| | |
|---|------|
| L4::Irq | 1290 |
| L4::Kobject_t< Ipc_gate, Rcv_endpoint, L4_PROTO_KOBJECT, Type_info::Demand_t< 1 > > | 1334 |
| L4::Ipc_gate | 1244 |
| L4::Kobject_t< Snd_destination, Kobject, L4_PROTO_KOBJECT > | 1334 |
| L4::Kobject_t< Task, Kobject, L4_PROTO_TASK, Type_info::Demand_t< 2 > > | 1334 |
| L4::Task | 1436 |
| L4::Kobject_t< Vm, Task, L4_PROTO_VM > | 1334 |
| L4::Vm | 1570 |
| L4::Kobject_t< Vcpu_context, Kobject, L4_PROTO_VCPU_CONTEXT > | 1334 |
| L4::Kobject_t< Dataspace, L4::Kobject, L4RE_PROTO_DATASPACE, L4::Type_info::Demand_t< 1 > > | 1334 |
| L4Re::Dataspace | 1650 |
| L4::Kobject_3t< Vbus, L4Re::Dataspace, L4Re::Inhibitor, L4Re::Event > | 1330 |
| L4vbus::Vbus | 2052 |
| L4::Kobject_t< Dbg_events, L4::Kobject, 0, L4::Type_info::Demand_t< 2 > > | 1334 |
| L4::Kobject_t< Debug_obj, L4::Kobject, L4RE_PROTO_DEBUG > | 1334 |
| L4Re::Debug_obj | 1665 |
| L4::Kobject_t< Dma_space, L4::Kobject, PROTO, L4::Type_info::Demand_t< 1 > > | 1334 |
| L4::Kobject_t< Inhibitor, L4::Kobject, L4RE_PROTO_INHIBITOR > | 1334 |
| L4Re::Inhibitor | 1708 |
| L4::Kobject_3t< Vbus, L4Re::Dataspace, L4Re::Inhibitor, L4Re::Event > | 1330 |
| L4::Kobject_t< Itas, L4::Kobject, L4RE_PROTO_ITAS, L4::Type_info::Demand_t< 2 > > | 1334 |
| L4Re::Itas | 1714 |
| L4::Kobject_t< Mmio_space, L4::Kobject, L4RE_PROTO_MMIO_SPACE > | 1334 |
| L4Re::Mmio_space | 1739 |
| L4::Kobject_t< Namespace, L4::Kobject, L4RE_PROTO_NAMESPACE, L4::Type_info::Demand_t< 1 > > | 1334 |
| L4Re::Namespace | 1744 |
| L4::Kobject_t< Cmd_control, L4::Kobject, L4::PROTO_ANY, Type_info::Demand_t<> > | 1334 |
| L4::Kobject_t< Parent, L4::Kobject, L4RE_PROTO_PARENT > | 1334 |
| L4Re::Parent | 1756 |
| L4::Kobject_t< Goos, L4::Kobject, L4RE_PROTO_GOOS > | 1334 |
| L4Re::Video::Goos | 1946 |
| L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND > | 1326 |
| L4::Kobject_2t< Console, Video::Goos, Event, L4::PROTO_EMPTY > | 1326 |
| L4Re::Console | 1642 |
| L4::Kobject_2t< Debug_obj_t< BASE >, BASE, Debug_obj, L4::PROTO_EMPTY > | 1326 |
| L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND > | 1330 |
| L4::Kobject_demand< T > | 1333 |
| L4::Kobject_t< Derived, Base, PROTO, S_DEMAND > | 1334 |
| L4::Kobject_t< Remote_access, Dataspace, L4RE_PROTO_REMOTE_ACCESS > | 1334 |
| L4::Kobject_typeid< T > | 1336 |
| L4::Kobject_typeid< void > | 1340 |
| L4::Kobject_x< Derived, ARGS > | 1343 |
| L4::Kobject_x< Iommu, Proto_t< L4_PROTO_IOMMU >, Type_info::Demand_t< 1 > > | 1343 |
| L4::Iommu | 1129 |
| L4::Lock_guard | 1344 |
| L4::Poll_timeout_counter | 1368 |
| L4::Poll_timeout_kipclock | 1371 |
| L4::Proto_t< P > | 1375 |
| L4::Registry_iface | 1381 |
| L4Re::Util::Object_registry | 1868 |
| L4::Reply_cap | 1386 |
| L4::Reply_cap_alloc | 1390 |
| L4::Reply_cap_idx | 1392 |
| L4::Server< LOOP_HOOKS > | 1411 |

| | |
|---|------|
| L4::String | 1435 |
| L4::Thread::Attr | 1464 |
| L4::Thread::Modify_senders | 1470 |
| L4::Type_info | 1480 |
| L4::Type_info::Demand | 1481 |
| L4::Type_info::Demand_t< __l::Max< Kobject_typeid< T1 >Demand::Caps, Kobject_demand< T2...> >Caps >Res, Kobject_typeid< T1 >Demand::Flags Kobject_demand< T2... >Flags, <__l::Max< Kobject_typeid< T1 >Demand::Mem, Kobject_demand< T2... >Mem >Res, __l::Max< Kobject_typeid< T1 >Demand::Ports, Kobject_demand< T2... >Ports >Res > | 1485 |
| L4::Type_info::Demand_union_t< Kobject_typeid< T1 >Demand, Kobject_demand< T2... > > | 1488 |
| L4::Type_info::Demand_t< __l::Max< D1::Caps, D2::Caps >Res, D1::Flags D2Flags, __l::Max< D1::Mem, D2::Mem >Res, __l::Max< D1::Ports, D2::Ports >Res > | 1485 |
| L4::Type_info::Demand_union_t< D1, D2 > | 1488 |
| L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS > | 1485 |
| L4::Typeid::Detail::_Rpc< OPCODE, O, Default_op< R > >Rpc< Y > | 1493 |
| L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... >Rpc< Y > | 1496 |
| L4::Typeid::Detail::Rpc_end | 1497 |
| L4::Typeid::Detail::type< OPCODE, 1,... > | 1492 |
| L4::Typeid::Detail::_Rpc< void, 0, OPERATION > | 1492 |
| L4::Typeid::Rpc_nocode< OPERATION > | 1500 |
| L4::Typeid::Detail::_Rpc< L4::Opcode, 0, RPCS... > | 1492 |
| L4::Typeid::Rpc< set_status_t, config_queue_t, register_ds_t, device_config_t, device_<notification_irq_t > | 1502 |
| L4::Typeid::Rpc< execute_t > | 1502 |
| L4::Typeid::Rpc< num_interfaces_t, interface_t, supports_t > | 1502 |
| L4::Typeid::Rpc< map_t, clear_t, info_t, copy_in_t, allocate_t, map_info_t > | 1502 |
| L4::Typeid::Rpc< request_backtrace_t > | 1502 |
| L4::Typeid::Rpc< map_t, unmap_t, associate_t, disassociate_t > | 1502 |
| L4::Typeid::Rpc< get_buffer_t, get_num_streams_t, get_stream_info_t, get_stream_info_for_id<_t, get_axis_info_t, get_stream_state_for_id_t > | 1502 |
| L4::Typeid::Rpc< acquire_t, release_t, next_lock_info_t > | 1502 |
| L4::Typeid::Rpc< register_thread_t, unregister_thread_t, sigaction_t, sigaltstack_t, sigprocmask<_t, sigpending_t, setitimer_t, getitimer_t, raise_t > | 1502 |
| L4::Typeid::Rpc< info_t > | 1502 |
| L4::Typeid::Rpc< mmio_read_t, mmio_write_t > | 1502 |
| L4::Typeid::Rpc< query_t, register_obj_t, unlink_t > | 1502 |
| L4::Typeid::Rpc< signal_t > | 1502 |
| L4::Typeid::Rpc< get_random_t > | 1502 |
| L4::Typeid::Rpc< read_mem_t, write_mem_t, terminate_t > | 1502 |
| L4::Typeid::Rpc< attach_t, detach_t, find_t, reserve_area_t, free_area_t, get_regions_t, get<areas_t, get_info_t > | 1502 |
| L4::Typeid::Rpc< info_t, get_static_buffer_t, create_buffer_t, create_view_t, delete_buffer<_t, delete_view_t, view_info_t, set_view_info_t, view_stack_t, view_refresh_t, refresh_t > | 1502 |
| L4::Typeid::Rpc< RPCS > | 1502 |
| L4::Typeid::Detail::_Rpc< OPCODE_TYPE, 0, RPCS... > | 1492 |
| L4::Typeid::Rpc_code< OPCODE_TYPE >F< RPCS > | 1505 |
| L4::Typeid::Detail::_Rpc< l4_umword_t, 0, ARG... > | 1492 |
| L4::Typeid::Rpc_sys< bind_thread_t, get_infos_t > | 1508 |
| L4::Typeid::Rpc_sys< bind_t, unbind_t, info_t, msi_info_t, unmask_t, mask_t, set_mode_t > | 1508 |
| L4::Typeid::Rpc_sys< system_suspend_t, system_shutdown_t, cpu_allow_shutdown_t, cpu<enable_t, cpu_disable_t > | 1508 |
| L4::Typeid::Rpc_sys< bind_thread_t > | 1508 |
| L4::Typeid::Rpc_sys< info_t, run_thread_t, idle_time_t > | 1508 |
| L4::Typeid::Rpc_sys< ARG > | 1508 |
| L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... > | 1493 |
| L4::Typeid::Detail::_Rpc< OPCODE, O, X > | 1492 |
| L4::Typeid::Rpc_nocode< call_t > | 1500 |

| | |
|---|------|
| L4::Typeid::Rpc_nocode< exception_t > | 1500 |
| L4::Typeid::Rpc_nocode< create_t > | 1500 |
| L4::Typeid::Rpc_nocode< io_page_fault_t > | 1500 |
| L4::Typeid::Rpc_nocode< page_fault_t > | 1500 |
| L4::Typeid::Rpc_nocode< debug_t > | 1500 |
| L4::Typeid::P_dispatch< LIST > | 1497 |
| L4::Typeid::Raw_ipc< CLASS > | 1498 |
| L4::Typeid::Rpcs_code< OPCODE_TYPE > | 1504 |
| L4::Types::__Add_rvalue_reference_helper< T, typename > | 1510 |
| L4::Types::__Add_rvalue_reference_helper< T, Void< T && > > | 1511 |
| L4::Types::__Underlying_type_helper< T, bool > | 1513 |
| L4::Types::Underlying_type< T > | 1543 |
| L4::Types::__Underlying_type_helper< T, false > | 1514 |
| L4::Types::Add_rvalue_reference< T > | 1516 |
| L4::Types::Bool< V > | 1517 |
| L4::Types::Bool< false > | 1517 |
| L4::Types::False | 1518 |
| Enum_bitops::Has_marker< T, Void< decltype(enum_bitops_enable(declval< T >()))> > | 1025 |
| Enum_bitops::Has_marker< typename, typename > | 1022 |
| L4::Types::Same< A, B > | 1536 |
| L4::Types::Same_template< T, Template > | 1538 |
| L4::Types::Bool< I1::Proto !=PROTO_EMPTY &&I1::Proto==I2::Proto > | 1517 |
| L4::Types::Bool< true > | 1517 |
| L4::Types::True | 1541 |
| L4::Ipc::Msg::Is_valid_rpc_type< A * > | 1193 |
| Enum_bitops::Has_marker< T, Void< decltype(enum_bitops_enable(declval< T >()))> > | 1025 |
| L4::Ipc::Msg::Is_valid_rpc_type< T > | 1193 |
| L4::Types::Flags< BITS_ENUM, UNDERLYING > | 1520 |
| L4::Types::Flags_ops_t< DT > | 1524 |
| L4::Types::Flags_ops_t< Flags > | 1524 |
| L4::Types::Flags_ops_t< Flags_t< DT, T > > | 1524 |
| L4::Types::Flags_t< DT, T > | 1527 |
| L4::Types::Int_for_size< SIZE, bool > | 1531 |
| L4::Types::Int_for_type< T > | 1531 |
| L4::Types::Integral_constant< T, Value > | 1532 |
| Enum_bitops::Enable< T > | 1021 |
| L4::Types::Integral_constant< bool, __is_enum(T)> | 1532 |
| L4::Types::Is_enum< T > | 1534 |
| L4::Uart | 1544 |
| L4::Uart_apb | 1551 |
| I4_buf_regs_t | 1574 |
| I4_exc_regs_t | 1575 |
| I4_fpage_t | 1579 |
| I4_icu_info_t | 1579 |
| L4::Icu::Info | 1121 |
| I4_icu_msi_info_t | 1581 |
| I4_kernel_info_mem_desc_t | 1582 |
| I4_kernel_info_t | 1583 |
| I4_msg_regs_t | 1584 |
| I4_msgtag_t | 1585 |
| I4_sched_cpu_set_t | 1588 |
| I4_sched_param_t | 1592 |
| I4_snd_fpage_t | 1593 |
| I4_thread_regs_t | 1594 |
| I4_timeout_s | 1595 |
| I4_timeout_t | 1596 |

| | |
|---|----------------------|
| l4_vcon_attr_t | 1597 |
| l4_vcpu_arch_state_t | 1599 |
| l4_vcpu_ipc_regs_t | 1599 |
| l4_vcpu_regs_t | 1601 |
| l4_vcpu_state_t | 1604 |
| L4vcpu::Vcpu | 2067 |
| l4_vm_state_t | 1608 |
| l4_vm_svm_vmcb_control_area | 1608 |
| l4_vm_svm_vmcb_state_save_area | 1609 |
| l4_vm_svm_vmcb_state_save_area_seg | 1610 |
| l4_vm_svm_vmcb_t | 1611 |
| l4_vm_tz_state | 1612 |
| l4_vm_vmx_vcpu_infos_t | 1612 |
| l4_vm_vmx_vcpu_state_t | 1613 |
| l4_vm_vmx_vcpu_vmcs_t | 1615 |
| l4_vmx_offset_table_t | 1616 |
| L4drivers::Register_block< MAX_BITS, BLOCK > | 1620 |
| L4drivers::Register_block_base< MAX_BITS > | 1624 |
| L4drivers::Register_block_impl< BASE, MAX_BITS > | 1625 |
| L4drivers::Register_block_impl< Mmio_register_block< 32 >, 32 > | 1625 |
| L4drivers::Mmio_register_block< MAX_BITS > | 1618 |
| L4drivers::Register_block_tmpl< BLOCK > | 1627 |
| L4drivers::Register_block_tmpl< Register_block_base< MAX_BITS > > | 1627 |
| L4drivers::Register_block_tmpl< Register_block_base< MAX_BITS > const > | 1627 |
| L4drivers::Ro_register_block< MAX_BITS, BLOCK > | 1634 |
| L4drivers::Ro_register_tmpl< BITS, BLOCK > | 1636 |
| L4drivers::Register_tmpl< MAX_BITS, Block > | 1628 |
| L4drivers::Register_tmpl< BITS, BLOCK > | 1628 |
| L4Re::Cap_alloc | 1639 |
| L4Re::Util::Cap_alloc | 1797 |
| L4Re::Core::Ref_ptr< T, CNT > | 1645 |
| L4Re::Dataspace::F | 1663 |
| L4Re::Dataspace::Stats | 1664 |
| L4Re::Default_event_payload | 1669 |
| L4Re::Env | 1677 |
| L4Re::Event_buffer_t< PAYLOAD > | 1703 |
| L4Re::Event_buffer_t< PAYLOAD >Event | 1707 |
| L4Re::Event_buffer_t< Default_event_payload > | 1703 |
| L4Re::Util::Event_buffer_t< Default_event_payload > | 1849 |
| L4Re::Util::Event_buffer_consumer_t< Default_event_payload > | 1846 |
| L4Re::Event_buffer_t< PAYLOAD > | 1703 |
| L4Re::Util::Event_buffer_t< PAYLOAD > | 1849 |
| L4Re::Util::Event_buffer_consumer_t< PAYLOAD > | 1846 |
| L4Re::Mem_alloc::Stats | 1736 |
| L4Re::Rm::Area | 1786 |
| L4Re::Rm::F | 1786 |
| L4Re::Rm::Region | 1788 |
| L4Re::Rm::Unique_region< T > | 1789 |
| L4Re::Smart_cap_auto< Unmap_flags > | 1795 |
| L4Re::Smart_count_cap< Unmap_flags > | 1795 |
| L4Re::Util::Bitmap_base | 1805 |
| cxx::Bitmap< BITS > | 864 |
| L4Re::Util::Bitmap_base::Bit | 1813 |
| L4Re::Util::Bitmap_base::Char< BITS > | 1813 |
| L4Re::Util::Bitmap_base::Word< BITS > | 1814 |
| L4Re::Util::Cap_alloc_base | 1829 |

| | |
|--|------|
| L4Re::Util::Counter< COUNTER > | 1830 |
| L4Re::Util::Counter_atomic< COUNTER > | 1831 |
| L4Re::Util::Counting_cap_alloc< COUNTERTYPE, Dbg > | 1833 |
| L4Re::Util::Dataspace_svr | 1839 |
| L4Re::Util::Event_svr< SVR > | 1853 |
| L4Re::Util::Event_t< PAYLOAD > | 1855 |
| L4Re::Util::Item_alloc_base | 1859 |
| L4Re::Util::Names::Name_space | 1863 |
| L4Re::Util::Ref_cap< T > | 1875 |
| L4Re::Util::Ref_del_cap< T > | 1877 |
| L4Re::Util::Smart_cap_auto< Unmap_flags > | 1884 |
| L4Re::Util::Smart_count_cap< Unmap_flags > | 1885 |
| L4Re::Util::Vcon_svr< SVR > | 1886 |
| L4Re::Util::Video::Goos_svr | 1887 |
| L4Re::Vfs::Directory | 1899 |
| L4Re::Vfs::File | 1904 |
| L4Re::Vfs::Be_file | 1891 |
| L4Re::Vfs::File_system | 1908 |
| L4Re::Vfs::Be_file_system | 1896 |
| L4Re::Vfs::Fs | 1910 |
| L4Re::Vfs::Ops | 1925 |
| L4Re::Vfs::Generic_file | 1915 |
| L4Re::Vfs::File | 1904 |
| L4Re::Vfs::Mman | 1924 |
| L4Re::Vfs::Ops | 1925 |
| L4Re::Vfs::Regular_file | 1928 |
| L4Re::Vfs::File | 1904 |
| L4Re::Vfs::Special_file | 1938 |
| L4Re::Vfs::File | 1904 |
| L4Re::Video::Color_component | 1941 |
| L4Re::Video::Goos::Info | 1956 |
| L4Re::Video::Pixel_info | 1958 |
| L4Re::Video::View | 1969 |
| L4Re::Video::View::Info | 1976 |
| l4re_aux_t | 1978 |
| l4re_ds_stats_t | 1979 |
| l4re_elf_aux_mword_t | 1979 |
| l4re_elf_aux_t | 1980 |
| l4re_elf_aux_vma_t | 1981 |
| l4re_env_cap_entry_t | 1981 |
| l4re_env_t | 1983 |
| l4re_event_t | 1986 |
| l4re_video_color_component_t | 1987 |
| l4re_video_goos_info_t | 1987 |
| l4re_video_pixel_info_t | 1989 |
| l4re_video_view_info_t | 1990 |
| l4re_video_view_t | 1992 |
| l4shmc_ringbuf_head_t | 1993 |
| l4shmc_ringbuf_t | 1994 |
| l4util_l4mod_info | 1995 |
| l4util_l4mod_mod | 1997 |
| l4util_mb_addr_range_t | 1998 |
| l4util_mb_apm_t | 1999 |
| l4util_mb_drive_t | 1999 |
| l4util_mb_info_t | 2000 |
| l4util_mb_mod_t | 2002 |

| | |
|--|------|
| l4util_mb_vbe_ctrl_t | 2003 |
| l4util_mb_vbe_mode_t | 2003 |
| L4vbus::Gpio_module::Pin_slice | 2024 |
| L4vbus::Pm< DEC > | 2049 |
| L4vbus::Pm< Device > | 2049 |
| L4vbus::Device | 2007 |
| L4vbus::Gpio_module | 2017 |
| L4vbus::Gpio_pin | 2024 |
| L4vbus::Icu | 2033 |
| L4vbus::Pci_dev | 2037 |
| L4vbus::Pci_host_bridge | 2043 |
| l4vbus_device_t | 2063 |
| l4vbus_resource_t | 2064 |
| L4vcpu::State | 2065 |
| L4virtio::Driver::Block_device::Handle | 2098 |
| L4virtio::Driver::Device | 2098 |
| L4virtio::Driver::Block_device | 2090 |
| L4virtio::Driver::Virtio_net_device | 2111 |
| L4virtio::Driver::Virtio_net_device::Packet | 2120 |
| L4virtio::Ptr< T > | 2130 |
| L4virtio::Svr::Bad_descriptor | 2133 |
| L4virtio::Svr::Block_request< Ds_data > | 2144 |
| L4virtio::Svr::Console::Control_message | 2146 |
| L4virtio::Svr::Console::Control_request | 2148 |
| L4virtio::Svr::Console::Port | 2172 |
| L4virtio::Svr::Console::Device_port | 2163 |
| L4virtio::Svr::Console::Port::Transition | 2176 |
| L4virtio::Svr::Data_buffer | 2194 |
| Buffer | 819 |
| L4virtio::Svr::Dev_config | 2198 |
| L4virtio::Svr::Dev_features | 2210 |
| L4virtio::Svr::Console::Features | 2167 |
| L4virtio::Svr::Dev_status | 2213 |
| L4virtio::Svr::Device_t< DATA > | 2216 |
| L4virtio::Svr::Device_t< Ds_data > | 2216 |
| L4virtio::Svr::Block_dev_base< Ds_data > | 2135 |
| L4virtio::Svr::Device_t< No_custom_data > | 2216 |
| L4virtio::Svr::Console::Virtio_con | 2177 |
| L4virtio::Svr::Console::Device | 2149 |
| L4virtio::Svr::Scmi::Scmi_dev | 2255 |
| L4virtio::Svr::Virtio_gpio< Request_handler, Epiface > | 2261 |
| L4virtio::Svr::Virtio_i2c< Request_handler, Epiface > | 2273 |
| L4virtio::Svr::Virtio_rng< Rnd_state, Epiface > | 2284 |
| L4virtio::Svr::Virtio_spi< Spi_request_handler, Epiface > | 2294 |
| Virtio_net | 2423 |
| L4virtio::Svr::Driver_mem_list_t< DATA > | 2223 |
| L4virtio::Svr::Driver_mem_region_t< DATA > | 2229 |
| L4virtio::Svr::Driver_mem_region_t< Ds_data > | 2229 |
| L4virtio::Svr::Driver_mem_region_t< No_custom_data > | 2229 |
| L4virtio::Svr::Request_processor | 2236 |
| L4virtio::Svr::Virtio_gpio< Request_handler, Epiface >Request_processor | 2270 |
| L4virtio::Svr::Virtio_i2c< Request_handler, Epiface >Request_processor | 2281 |
| L4virtio::Svr::Virtio_rng< Rnd_state, Epiface >Request_processor | 2292 |
| L4virtio::Svr::Virtio_spi< Spi_request_handler, Epiface >Request_processor | 2303 |
| L4virtio::Svr::Scmi::Base_attr_t | 2243 |
| L4virtio::Svr::Scmi::Performance_attr_t | 2249 |

| | |
|--|------|
| L4virtio::Svr::Scmi::Performance_describe_level_t | 2250 |
| L4virtio::Svr::Scmi::Performance_describe_levels_n_t | 2250 |
| L4virtio::Svr::Scmi::Performance_domain_attr_t | 2252 |
| L4virtio::Svr::Scmi::Proto< OBSERV > | 2254 |
| L4virtio::Svr::Scmi::Proto< Scmi_dev > | 2254 |
| L4virtio::Svr::Scmi::Base_proto | 2244 |
| L4virtio::Svr::Scmi::Perf_proto | 2246 |
| L4virtio::Svr::Scmi::Scmi_hdr_t | 2259 |
| L4virtio::Svr::Virtio_gpio< Request_handler, Epiface >Irq_handler | 2269 |
| L4virtio::Svr::Virtqueue::Head_desc | 2318 |
| L4virtio::Virtqueue | 2321 |
| L4virtio::Driver::Virtqueue | 2121 |
| L4virtio::Svr::Virtqueue | 2306 |
| L4virtio::Virtqueue::Avail | 2338 |
| L4virtio::Virtqueue::Avail::Flags | 2339 |
| L4virtio::Virtqueue::Desc | 2340 |
| L4virtio::Virtqueue::Desc::Flags | 2342 |
| L4virtio::Virtqueue::Used | 2344 |
| L4virtio::Virtqueue::Used::Flags | 2345 |
| L4virtio::Virtqueue::Used_elem | 2347 |
| l4virtio_block_config_t | 2348 |
| l4virtio_block_discard_t | 2349 |
| l4virtio_block_header_t | 2350 |
| l4virtio_config_hdr_t | 2351 |
| l4virtio_config_queue_t | 2352 |
| l4virtio_input_absinfo_t | 2353 |
| l4virtio_input_config_t | 2354 |
| l4virtio_input_devids_t | 2354 |
| l4virtio_input_event_t | 2355 |
| l4virtio_net_config_t | 2356 |
| l4virtio_net_header_t | 2356 |
| Mac_addr | 2363 |
| Mac_table< Size > | 2363 |
| Net_transfer | 2366 |
| Rm::Area | 2383 |
| Rm::F | 2384 |
| Rm::Region | 2386 |
| Rm::Unique_region< T > | 2387 |
| utrace::Ring_buffer< SEQUENCE_TYPE > | 2396 |
| utrace::Ring_buffer_consumer_raw< SEQUENCE_TYPE, ITEM_TYPE, GENERATION_PTR > | 2406 |
| utrace::Ring_buffer_consumer< SEQUENCE_TYPE, ITEM_TYPE, GENERATION_PTR > | 2400 |
| utrace::Ring_buffer_producer< SEQUENCE_TYPE, ITEM_TYPE, GENERATION_PTR > | 2411 |
| utrace::Ring_buffer< Sequence > | 2396 |
| utrace::Ring_buffer_consumer_raw< Sequence, Item, GENERATION_PTR > | 2406 |
| utrace::Ring_buffer_consumer< Sequence, Item, &Item::number > | 2400 |
| utrace::Ring_slot< ITEM_TYPE, GENERATION_PTR > | 2415 |
| utrace::Ring_status< SEQUENCE_TYPE > | 2417 |
| utrace::Tracebuffer | 2420 |
| utrace::Tracebuffer::Index_desc | 2423 |
| Virtio_net_request | 2428 |
| Virtio_switch | 2431 |
| Virtio_vlan_mangle | 2436 |

Chapter 11

Data Structure Index

11.1 Data Structures

Here are the data structures with brief descriptions:

| | |
|--|-----|
| Buffer | |
| Data buffer used to transfer packets | 819 |
| cxx::arith::Ld< V > | |
| Computes the binary logarithm of the given number at compile time | 821 |
| cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC > | |
| AVL tree based associative container | 822 |
| cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC > | |
| AVL set for simple comparable items | 828 |
| cxx::Avl_tree< Node, Get_key, Compare > | |
| A generic AVL tree | 832 |
| cxx::Avl_tree_node | |
| Node of an AVL tree | 840 |
| cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc > | |
| Basic slab allocator | 842 |
| cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::Slab_i | |
| Type of a slab | 846 |
| cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc > | |
| Merged slab allocator (allocators for objects of the same size are merged together) | 847 |
| cxx::Bitfield< T, LSB, MSB > | |
| Definition for a member (part) of a bit field | 852 |
| cxx::Bitfield< T, LSB, MSB >::Value< TT > | |
| Internal helper type | 860 |
| cxx::Bitfield< T, LSB, MSB >::Value_base< TT > | |
| Internal helper type | 861 |
| cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT > | |
| Internal helper type | 862 |
| cxx::Bitmap< BITS > | |
| A static bitmap | 864 |
| cxx::Bitmap_base | |
| Basic bitmap abstraction | 868 |
| cxx::Bitmap_base::Bit | |
| A writable bit in a bitmap | 881 |
| cxx::Bitmap_base::Char< BITS > | |
| Helper abstraction for a byte contained in the bitmap | 881 |
| cxx::Bitmap_base::Word< BITS > | |
| Helper abstraction for a word contained in the bitmap | 882 |

| | |
|---|-----|
| cxx::Bits::Avl_map_get_key< KEY_TYPE > | |
| Key-getter for Avl_map | 883 |
| cxx::Bits::Avl_set_get_key< KEY_TYPE > | |
| Internal, key-getter for Avl_set nodes | 884 |
| cxx::Bits::Avl_set_iter< Node, Key, Node_op > | |
| Generic iterator for the AVL-tree based set | 884 |
| cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY > | |
| AVL set with internally managed nodes | 888 |
| cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node | |
| A smart pointer to a tree item | 898 |
| cxx::Bits::Basic_list< POLICY > | |
| Internal: Common functions for all head-based list implementations | 900 |
| cxx::Bits::Bst< Node, Get_key, Compare > | |
| Basic binary search tree (BST) | 903 |
| cxx::Bits::Bst_node | |
| Basic type of a node in a binary search tree (BST) | 916 |
| cxx::Bits::Direction | |
| The direction to go in a binary search tree | 918 |
| cxx::Bits::Smart_ptr_list< ITEM > | |
| List of smart-pointer-managed objects | 920 |
| cxx::Bits::Smart_ptr_list_item< T, STORE_T > | |
| List item for an arbitrary item in a Smart_ptr_list | 923 |
| cxx::Elide_dtor< T > | |
| Wrapper class to remove destructor calls | 925 |
| cxx::Error | |
| Error value | 926 |
| cxx::H_list< T, POLICY > | |
| General double-linked list of unspecified cxx::H_list_item elements | 926 |
| cxx::H_list_item_t< ELEM_TYPE > | |
| Basic element type for a double-linked H_list | 934 |
| cxx::H_list_t< T > | |
| Double-linked list of typed H_list_item_t elements | 937 |
| cxx::List< D, Alloc > | |
| Doubly linked list, with internal allocation | 941 |
| cxx::List< D, Alloc >::Iter | |
| Iterator | 943 |
| cxx::List_alloc | |
| Standard list-based allocator | 944 |
| cxx::List_item | |
| Basic list item | 948 |
| cxx::List_item::Iter | |
| Iterator for a list of ListItem -s | 951 |
| cxx::List_item::T_iter< T, Poly > | |
| Iterator for derived classes from ListItem | 953 |
| cxx::Lt_functor< Obj > | |
| Generic comparator class that defaults to the less-than operator | 955 |
| cxx::New_allocator< _Type > | |
| Standard allocator based on <code>operator new ()</code> | 955 |
| cxx::Nothrow | |
| Helper type to distinguish the <code>operator new</code> version that does not throw exceptions | 957 |
| cxx::Pair< First, Second > | |
| Pair of two values | 957 |
| cxx::Pair_first_compare< Cmp, Typ > | |
| Comparison functor for Pair | 960 |
| cxx::Ref_obj_list_item< T > | |
| Item for list linked via cxx::Ref_ptr with default reference counting | 962 |
| cxx::Ref_ptr< T, CNT > | |
| A reference-counting pointer with automatic cleanup | 964 |

| | |
|---|------|
| cxx::Result< T > | |
| A result of a function call | 969 |
| cxx::S_list< T, POLICY > | |
| Simple single-linked list | 970 |
| cxx::Slab< Type, Slab_size, Max_free, Alloc > | |
| Slab allocator for object of type <code>Type</code> | 973 |
| cxx::Slab_static< Type, Slab_size, Max_free, Alloc > | |
| Merged slab allocator (allocators for objects of the same size are merged together) | 978 |
| cxx::static_vector< T, IDX > | |
| Simple encapsulation for a dynamically allocated array | 982 |
| cxx::String | |
| Allocation free string class with explicit length field | 983 |
| cxx::Weak_ref< T > | |
| Typed weak reference to an object of type <code>T</code> | 991 |
| cxx::Weak_ref_base | |
| Generic (base) weak reference to some object | 993 |
| cxx::Weak_ref_base::List | |
| The list type for keeping all weak references to an object | 996 |
| Elf32_Auxv | |
| Auxiliary vector (32-bit) | 1000 |
| Elf32_Dyn | |
| ELF32 dynamic entry | 1001 |
| Elf32_Ehdr | |
| ELF32 header | 1002 |
| Elf32_Phdr | |
| ELF32 program header | 1005 |
| Elf32_Rel | |
| ELF32 relocation entry w/o addend | 1006 |
| Elf32_Rela | |
| ELF32 relocation entry w/ addend | 1007 |
| Elf32_Shdr | |
| ELF32 section header | 1008 |
| Elf32_Sym | |
| ELF32 symbol table entry | 1010 |
| Elf64_Auxv | |
| Auxiliary vector (64-bit) | 1011 |
| Elf64_Dyn | |
| ELF64 dynamic entry | 1012 |
| Elf64_Ehdr | |
| ELF64 header | 1013 |
| Elf64_Phdr | |
| ELF64 program header | 1015 |
| Elf64_Rel | |
| ELF64 relocation entry w/o addend | 1017 |
| Elf64_Rela | |
| ELF64 relocation entry w/ addend | 1017 |
| Elf64_Shdr | |
| ELF64 section header | 1018 |
| Elf64_Sym | |
| ELF64 symbol table entry | 1020 |
| Enum_bitops::Enable< T > | |
| Check whether the given enum type opts in for the bitwise operators | 1021 |
| Enum_bitops::Has_marker< typename, typename > | |
| Marker for the opt-in ADL function | 1022 |
| Enum_bitops::Has_marker< T, Void< decltype(enum_bitops_enable(declval< T >()))> > | |
| Marker for the opt-in ADL function | 1025 |
| L4::Alloc_list | |
| A simple list-based allocator | 1027 |

| | | |
|---|---|------|
| L4::Arm_smccc | Wrapper for function calls that follow the ARM SMC/HVC calling convention | 1028 |
| L4::Base_exception | Base class for all exceptions, thrown by the L4Re framework | 1030 |
| L4::Basic_registry | This registry returns the corresponding server object based on the label of an lpc_gate | 1032 |
| L4::Bounds_error | Access out of bounds | 1035 |
| L4::Cap< T > | C++ interface for capabilities | 1038 |
| L4::Cap_base | Base class for all kinds of capabilities | 1044 |
| L4::Com_error | Error conditions during IPC | 1057 |
| L4::Debugger | C++ kernel debugger API | 1060 |
| L4::Element_already_exists | Exception for duplicate element insertions | 1070 |
| L4::Element_not_found | Exception for a failed lookup (element not found) | 1074 |
| L4::Epiface | Base class for interface implementations | 1077 |
| L4::Epiface_t< Derived, IFACE, BASE, bool > | Epiface implementation for Kobject-based interface implementations | 1082 |
| L4::Epiface_t0< RPC_IFACE, BASE > | Epiface mixin for generic Kobject-based interfaces | 1085 |
| L4::Exception | Exception interface | 1088 |
| L4::Exception_tracer | Back-trace support for exceptions | 1090 |
| L4::Factory | C++ Factory interface, see Factory for the C interface | 1091 |
| L4::Factory::Lstr | Special type to add a pascal string into the factory create stream | 1102 |
| L4::Factory::Nil | Special type to add a void argument into the factory create stream | 1104 |
| L4::Factory::S | Stream class for the create() argument stream | 1104 |
| L4::lcu | C++ lcu interface, see Interrupt controller for the C interface | 1111 |
| L4::lcu::Info | This class encapsulates information about an ICU | 1121 |
| L4::Invalid_capability | Indicates that an invalid object was invoked | 1123 |
| L4::lo_pager | lo_pager interface | 1126 |
| L4::lommu | Interface for IO-MMUs used for DMA remapping | 1129 |
| L4::IOModifier | Modifier class for the IO stream | 1134 |
| L4::lpc::Array< ELEM_TYPE, LEN_TYPE > | Array data type for dynamically sized arrays in RPCs | 1134 |
| L4::lpc::Array_in_buf< ELEM_TYPE, LEN_TYPE, MAX > | Server-side copy in buffer for Array | 1137 |
| L4::lpc::Array_ref< ELEM_TYPE, LEN_TYPE > | Array reference data type for arrays located in the message | 1140 |
| L4::lpc::As_value< T > | Pass the argument as plain data value | 1141 |

| | | |
|---|---|------|
| L4::lpc::Call | RPC attribute for a standard RPC call | 1142 |
| L4::lpc::Call_t< RIGHTS > | RPC attribute for an RPC call with required rights | 1143 |
| L4::lpc::Call_zero_send_timeout | RPC attribute for an RPC call, with zero send timeout | 1144 |
| L4::lpc::Cap< T > | Capability type for RPC interfaces (see L4::Cap<T>) | 1145 |
| L4::lpc::Gen_fpage | Generic RPC base for typed message items | 1150 |
| L4::lpc::In_out< T > | Mark an argument as in-out argument | 1153 |
| L4::lpc::Iostream | Input/Output stream for IPC [un]marshalling | 1154 |
| L4::lpc::Istream | Input stream for IPC unmarshalling | 1164 |
| L4::lpc::Msg::CInt_val_ops< MTYPE, DIR, CLASS > | Defines client-side handling of 'MTYPE' as RPC argument | 1175 |
| L4::lpc::Msg::Cls_buffer | Marker type for receive buffer values | 1176 |
| L4::lpc::Msg::Cls_data | Marker type for data values | 1178 |
| L4::lpc::Msg::Cls_item | Marker type for item values | 1179 |
| L4::lpc::Msg::Dir_in | Marker type for input values | 1180 |
| L4::lpc::Msg::Dir_out | Marker type for output values | 1181 |
| L4::lpc::Msg::Do_in_data | Marker for Input data | 1182 |
| L4::lpc::Msg::Do_in_items | Marker for Input items | 1183 |
| L4::lpc::Msg::Do_out_data | Marker for Output data | 1184 |
| L4::lpc::Msg::Do_out_items | Marker for Output items | 1185 |
| L4::lpc::Msg::Do_rcv_buffers | Marker for receive buffers | 1187 |
| L4::lpc::Msg::Elem< Array< A, LEN > & > | Array as output argument | 1188 |
| L4::lpc::Msg::Elem< Array< A, LEN > > | Array as input arguments | 1189 |
| L4::lpc::Msg::Elem< Array_ref< A, LEN > & > | Array_ref as output argument | 1190 |
| L4::lpc::Msg::False | False meta value | 1191 |
| L4::lpc::Msg::Is_valid_rpc_type< T > | Type trait defining a valid RPC parameter type | 1193 |
| L4::lpc::Msg::Svr_arg_pack< IPC_TYPE > | Server-side RPC arguments data structure used to provide arguments to the server-side implementation of an RPC function | 1195 |
| L4::lpc::Msg::Svr_val_ops< MTYPE, DIR, CLASS > | Defines server-side handling for MTYPE server arguments | 1195 |
| L4::lpc::Msg::True | True meta value | 1196 |
| L4::lpc::Msg_ptr< T > | Pointer to an element of type T in an lpc::Istream | 1199 |

| | | |
|---|--|------|
| L4::lpc::Opt< T > | Attribute for defining an optional RPC argument | 1200 |
| L4::lpc::Ostream | Output stream for IPC marshalling | 1202 |
| L4::lpc::Out< T > | Mark an argument as a output value in an RPC signature | 1208 |
| L4::lpc::Rcv_fpage | Non-small receive item | 1209 |
| L4::lpc::Ret_array< T > | Dynamically sized output array of type T | 1216 |
| L4::lpc::Send_only | RPC attribute for a send-only RPC | 1217 |
| L4::lpc::Small_buf | A receive item for receiving a single object capability | 1218 |
| L4::lpc::Snd_fpage | Send item or return item | 1220 |
| L4::lpc::Str_cp_in< T > | Abstraction for extracting a zero-terminated string from an lpc::Istream | 1230 |
| L4::lpc::Varg | Variably sized RPC argument | 1231 |
| L4::lpc::Varg_list< MAX > | Self-contained list of variable-sized RPC parameters | 1237 |
| L4::lpc::Varg_list_ref | List of variable-sized RPC parameters as received by the server | 1240 |
| L4::lpc::Varg_list_ref::Iterator | Iterator for Valists | 1242 |
| L4::lpc_gate | The C++ IPC gate interface, see IPC-Gate API for the C interface | 1244 |
| L4::lpc_svr::Br_manager_no_buffers | Empty implementation of Server_iface | 1249 |
| L4::lpc_svr::Compound_reply | Mix in for LOOP_HOOKS to always use compound reply and wait | 1254 |
| L4::lpc_svr::Dbg_dispatch< R, Exc, Printer > | Dispatch helper that, in addition to what Exc_dispatch does, prints exception messages | 1255 |
| L4::lpc_svr::Default_loop_hooks | Default LOOP_HOOKS | 1257 |
| L4::lpc_svr::Default_setup_wait | Mix in for LOOP_HOOKS for setup_wait no op | 1260 |
| L4::lpc_svr::Default_timeout | Mix in for LOOP_HOOKS to use a 0 send and an infinite receive timeout | 1261 |
| L4::lpc_svr::Direct_dispatch< R > | Direct dispatch helper, for forwarding dispatch calls to a registry <i>R</i> | 1263 |
| L4::lpc_svr::Direct_dispatch< R * > | Direct dispatch helper, for forwarding dispatch calls via a pointer to a registry <i>R</i> | 1265 |
| L4::lpc_svr::Exc_dispatch< R, Exc > | Dispatch helper wrapping try {} catch {} around the dispatch call | 1267 |
| L4::lpc_svr::Ignore_errors | Mix in for LOOP_HOOKS to ignore IPC errors | 1269 |
| L4::lpc_svr::Server_iface | Interface for server-loop related functions | 1270 |
| L4::lpc_svr::Server_iface::Mem_window | Memory receive window | 1278 |
| L4::lpc_svr::Timeout | Callback interface for Timeout_queue | 1278 |
| L4::lpc_svr::Timeout_queue | Timeout queue to be used in l4re server loop | 1282 |
| L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN > | Loop hooks mixin for integrating a timeout queue into the server loop | 1285 |

| | | |
|---|--|------|
| L4::Irq | C++ Irq interface, see IRQs for the C interface | 1290 |
| L4::Irq_eoi | Interface for sending an unmask message to an object | 1300 |
| L4::Irq_handler_object | Server object base class for handling IRQ messages | 1303 |
| L4::Irqp_t< Derived, BASE, bool > | Epiface implementation for interrupt handlers | 1307 |
| L4::Kip::Mem_desc | Memory descriptors stored in the kernel interface page | 1311 |
| L4::Kobject | Base class for all kinds of kernel objects and remote objects, referenced by capabilities | 1322 |
| L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND > | Helper class to create an L4Re interface class that is derived from two base classes (see L4::Kobject_t) | 1326 |
| L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND > | Helper class to create an L4Re interface class that is derived from three base classes (see L4::Kobject_t) | 1330 |
| L4::Kobject_demand< T > | Get the combined server-side resource requirements for all type T. | 1333 |
| L4::Kobject_t< Derived, Base, PROTO, S_DEMAND > | Helper class to create an L4Re interface class that is derived from a single base class | 1334 |
| L4::Kobject_typeid< T > | Meta object for handling access to type information of Kobjects | 1336 |
| L4::Kobject_typeid< void > | Minimalistic ID for <code>void</code> interface | 1340 |
| L4::Kobject_x< Derived, ARGS > | Generic Kobject inheritance template | 1343 |
| L4::Lock_guard | Basic lock guard implementation that prevents forgotten unlocks on exit paths from a method or a block of code | 1344 |
| L4::Meta | Meta interface that shall be implemented by each L4Re object and gives access to the dynamic type information for L4Re objects | 1347 |
| L4::Out_of_memory | Exception signalling insufficient memory | 1353 |
| L4::Pager | Pager interface including the Io_pager interface | 1357 |
| L4::Platform_control | L4 C++ interface for controlling platform-wide properties, see Platform Control C API for the C interface | 1360 |
| L4::Poll_timeout_counter | Evaluate an expression for a maximum number of times | 1368 |
| L4::Poll_timeout_kipclock | A polling timeout based on the L4Re clock | 1371 |
| L4::Proto_t< P > | Data type for defining protocol numbers | 1375 |
| L4::Rcv_endpoint | Interface for kernel objects that allow to receive IPC from them | 1375 |
| L4::Registry_iface | Abstract interface for object registries | 1381 |
| L4::Reply_cap | An explicit reply capability | 1386 |
| L4::Reply_cap_alloc | Interface to a reply capability allocator | 1390 |
| L4::Reply_cap_idx | Value class for a reply capability index | 1392 |

| | |
|---|------|
| L4::Runtime_error | |
| Exception for an abstract runtime error | 1393 |
| L4::Scheduler | |
| C++ interface of the Scheduler kernel object, see Scheduler for the C interface | 1397 |
| L4::Semaphore | |
| C++ Kernel-provided semaphore interface, see Kernel-provided semaphore for the C interface | 1406 |
| L4::Server< LOOP_HOOKS > | |
| Basic server loop for handling client requests | 1411 |
| L4::Server_object | |
| Abstract server object to be used with L4::Server and L4::Basic_registry | 1417 |
| L4::Server_object_t< IFACE, BASE > | |
| Base class (template) for server implementing server objects | 1422 |
| L4::Server_object_x< Derived, IFACE, BASE > | |
| Helper class to implement p_dispatch based server objects | 1427 |
| L4::Smart_cap< T, SMART > | |
| Smart capability class | 1431 |
| L4::String | |
| A null-terminated string container class | 1435 |
| L4::Task | |
| C++ interface of the Task kernel object, see Task for the C interface | 1436 |
| L4::Thread | |
| C++ L4 kernel thread interface, see Thread for the C interface | 1449 |
| L4::Thread::Attr | |
| Thread attributes used for control() | 1464 |
| L4::Thread::Modify_senders | |
| Class wrapping a list of rules which modify the sender label of IPC messages inbound to this thread | 1470 |
| L4::Thread_group | |
| C++ L4 kernel thread group interface, see Thread groups for the C interface | 1472 |
| L4::Triggerable | |
| Interface that allows an object to be triggered by some source | 1476 |
| L4::Type_info | |
| Dynamic Type Information for L4Re Interfaces | 1480 |
| L4::Type_info::Demand | |
| Data type for expressing the needed receive buffers at the server-side of an interface | 1481 |
| L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS > | |
| Template type statically describing demand of receive buffers | 1485 |
| L4::Type_info::Demand_union_t< D1, D2 > | |
| Template type statically describing the combination of two Demand object | 1488 |
| L4::Typeid::Detail::_Rpc< OPCODE, O, X > | |
| Empty list of RPCs | 1492 |
| L4::Typeid::Detail::_Rpc< OPCODE, O, Default_op< R > >::Rpc< Y > | |
| Find the given RPC in the list | 1493 |
| L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... > | |
| Non-empty list of RPCs | 1493 |
| L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... >::Rpc< Y > | |
| Find the given RPC in the list | 1496 |
| L4::Typeid::Detail::Rpc_end | |
| Internal end-of-list marker | 1497 |
| L4::Typeid::P_dispatch< LIST > | |
| Use for protocol based dispatch stage | 1497 |
| L4::Typeid::Raw_ipc< CLASS > | |
| RPCs list for passing raw incoming IPC to the server object | 1498 |
| L4::Typeid::Rpc_nocode< OPERATION > | |
| List of RPCs of an interface using a single operation without an opcode | 1500 |
| L4::Typeid::Rpc< RPCS > | |
| Standard list of RPCs of an interface | 1502 |

| | |
|---|------|
| L4::TypeId::Rpccs_code< OPCODE_TYPE > | |
| List of RPCs of an interface using a special opcode type | 1504 |
| L4::TypeId::Rpccs_code< OPCODE_TYPE >::F< RPCS > | 1505 |
| L4::TypeId::Rpccs_sys< ARG > | |
| List of RPCs typically used for kernel interfaces | 1508 |
| L4::Types::__Add_rvalue_reference_helper< T, typename > | |
| Helper template for Add_rvalue_reference | 1510 |
| L4::Types::__Add_rvalue_reference_helper< T, Void< T && > > | |
| Helper template for Add_rvalue_reference | 1511 |
| L4::Types::__Underlying_type_helper< T, bool > | |
| Helper template for Underlying_type | 1513 |
| L4::Types::__Underlying_type_helper< T, false > | |
| Helper template for Underlying_type | 1514 |
| L4::Types::Add_rvalue_reference< T > | |
| Create an rvalue reference of the given type | 1516 |
| L4::Types::Bool< V > | |
| Boolean meta type | 1517 |
| L4::Types::False | |
| False meta value | 1518 |
| L4::Types::Flags< BITS_ENUM, UNDERLYING > | |
| Template for defining typical Flags bitmaps | 1520 |
| L4::Types::Flags_ops_t< DT > | |
| Mixin class to define a set of friend bitwise operators on DT | 1524 |
| L4::Types::Flags_t< DT, T > | |
| Template type to define a flags type with bitwise operations | 1527 |
| L4::Types::Int_for_size< SIZE, bool > | |
| Metafunction to get an unsigned integral type for the given size | 1531 |
| L4::Types::Int_for_type< T > | |
| Metafunction to get an integral type of the same size as T | 1531 |
| L4::Types::Integral_constant< T, Value > | |
| Wrapper for a static constant of the given type | 1532 |
| L4::Types::Is_enum< T > | |
| Check whether the given type is an enumeration type | 1534 |
| L4::Types::Same< A, B > | |
| Compare two data types for equality | 1536 |
| L4::Types::Same_template< T, Template > | |
| Check if a type T is an instantiation of a given template | 1538 |
| L4::Types::True | |
| True meta value | 1541 |
| L4::Types::Underlying_type< T > | |
| Get an underlying type of an enumeration type | 1543 |
| L4::Uart | |
| Uart driver abstraction | 1544 |
| L4::Uart_apb | |
| Driver for the Advanced Peripheral Bus (APB) UART from the Cortex-M System Design Kit (CMSDK) | 1551 |
| L4::Unknown_error | |
| Exception for an unknown condition | 1558 |
| L4::Vcon | |
| C++ L4 Vcon interface, see Virtual Console for the C interface | 1560 |
| L4::Vm | |
| Virtual machine host address space | 1570 |
| l4_buf_regs_t | |
| Encapsulation of the buffer-registers block in the UTCB | 1574 |
| l4_exc_regs_t | |
| UTCB structure for exceptions | 1575 |
| l4_fpage_t | |
| L4 flexpage type | 1579 |

| | | |
|--|---|------|
| l4_icu_info_t | Info structure for an ICU | 1579 |
| l4_icu_msi_info_t | Info to use for a specific MSI | 1581 |
| l4_kernel_info_mem_desc_t | Memory descriptor data structure | 1582 |
| l4_kernel_info_t | L4 Kernel Interface Page | 1583 |
| l4_msg_regs_t | Encapsulation of the message-register block in the UTCB | 1584 |
| l4_msgtag_t | Message tag data structure | 1585 |
| l4_sched_cpu_set_t | CPU sets | 1588 |
| l4_sched_param_t | Scheduler parameter set | 1592 |
| l4_snd_fpage_t | Send-flexpage types | 1593 |
| l4_thread_regs_t | Encapsulation of the thread-control-register block of the UTCB | 1594 |
| l4_timeout_s | Basic timeout specification | 1595 |
| l4_timeout_t | Timeout pair | 1596 |
| l4_vcon_attr_t | Vcon attribute structure | 1597 |
| l4_vcpu_arch_state_t | Architecture-specific vCPU state | 1599 |
| l4_vcpu_ipc_regs_t | vCPU message registers | 1599 |
| l4_vcpu_regs_t | vCPU registers | 1601 |
| l4_vcpu_state_t | State of a vCPU | 1604 |
| l4_vm_state_t | L4 extended vCPU state for RISC-V | 1608 |
| l4_vm_svm_vmcb_control_area | VMCB structure for SVM VMs | 1608 |
| l4_vm_svm_vmcb_state_save_area | State save area structure for SVM VMs | 1609 |
| l4_vm_svm_vmcb_state_save_area_seg | State save area segment selector struct | 1610 |
| l4_vm_svm_vmcb_t | Control structure for SVM VMs | 1611 |
| l4_vm_tz_state | State structure for TrustZone VMs | 1612 |
| l4_vm_vmx_vcpu_infos_t | VMX information members | 1612 |
| l4_vm_vmx_vcpu_state_t | VMX vCPU state | 1613 |
| l4_vm_vmx_vcpu_vmcs_t | VMX software VMCS | 1615 |
| l4_vmx_offset_table_t | Software VMCS field offset table | 1616 |
| L4drivers::Mmio_register_block< MAX_BITS > | An MMIO block with up to 64-bit register access (32-bit default) and little endian byte order | 1618 |
| L4drivers::Register_block< MAX_BITS, BLOCK > | Handles a reference to a register block of the given maximum access width | 1620 |

| | |
|---|------|
| L4drivers::Register_block_base< MAX_BITS > | |
| Abstract register block interface | 1624 |
| L4drivers::Register_block_impl< BASE, MAX_BITS > | |
| Implementation helper for register blocks | 1625 |
| L4drivers::Register_block_tmpl< BLOCK > | |
| Helper template that translates to the Register_block_base interface | 1627 |
| L4drivers::Register_tmpl< BITS, BLOCK > | |
| Single hardware register inside a Register_block_base interface | 1628 |
| L4drivers::Ro_register_block< MAX_BITS, BLOCK > | |
| Handles a reference to a read only register block of the given maximum access width | 1634 |
| L4drivers::Ro_register_tmpl< BITS, BLOCK > | |
| Single read only register inside a Register_block_base interface | 1636 |
| L4Re::Cap_alloc | |
| Capability allocator interface | 1639 |
| L4Re::Console | |
| Console class | 1642 |
| L4Re::Core::Ref_ptr< T, CNT > | |
| A reference-counting pointer with automatic cleanup | 1645 |
| L4Re::Dataspace | |
| Interface for memory-like objects | 1650 |
| L4Re::Dataspace::F | |
| Dataspace flags definitions | 1663 |
| L4Re::Dataspace::Stats | |
| Information about the dataspace | 1664 |
| L4Re::Debug_obj | |
| Debug interface | 1665 |
| L4Re::Default_event_payload | |
| Default event stream payload | 1669 |
| L4Re::Dma_space | |
| Managed DMA Address Space | 1670 |
| L4Re::Env | |
| C++ interface of the initial environment that is provided to an L4 task | 1677 |
| L4Re::Event | |
| Event class | 1693 |
| L4Re::Event_buffer_t< PAYLOAD > | |
| Event buffer class | 1703 |
| L4Re::Event_buffer_t< PAYLOAD >::Event | |
| Event structure used in buffer | 1707 |
| L4Re::Inhibitor | |
| Set of inhibitor locks, which inhibit specific actions when held | 1708 |
| L4Re::Itas | |
| Interface to the ITAS | 1714 |
| L4Re::Log | |
| Log interface class | 1724 |
| L4Re::Mem_alloc | |
| Memory allocation interface | 1730 |
| L4Re::Mem_alloc::Stats | |
| Statistics about memory-allocator | 1736 |
| L4Re::Mmio_space | |
| Interface for memory-like address space accessible via IPC | 1739 |
| L4Re::Namespace | |
| Name-space interface | 1744 |
| L4Re::Ned::Cmd_control | |
| Direct control interface for Ned | 1754 |
| L4Re::Parent | |
| Parent interface | 1756 |
| L4Re::Random | |
| Low-bandwidth interface for random number generators | 1761 |

| | |
|---|------|
| L4Re::Rm | |
| Region map | 1765 |
| L4Re::Rm::Area | |
| An area is a range of virtual addresses which is reserved, see L4Re::Rm::reserve_area() . . . | 1786 |
| L4Re::Rm::F | |
| Rm flags definitions | 1786 |
| L4Re::Rm::Region | |
| A region is a range of virtual addresses which is backed by content | 1788 |
| L4Re::Rm::Unique_region< T > | |
| Unique region | 1789 |
| L4Re::Smart_cap_auto< Unmap_flags > | |
| Helper for Unique_cap and Unique_del_cap | 1795 |
| L4Re::Smart_count_cap< Unmap_flags > | |
| Helper for Ref_cap and Ref_del_cap | 1795 |
| L4Re::Util::_Cap_alloc | |
| Adapter to expose the cap allocator implementation as L4Re::Cap_alloc compatible class . . . | 1797 |
| L4Re::Util::Bitmap< BITS > | |
| A static bitmap | 1800 |
| L4Re::Util::Bitmap_base | |
| Basic bitmap abstraction | 1805 |
| L4Re::Util::Bitmap_base::Bit | |
| A writable bit in a bitmap | 1813 |
| L4Re::Util::Bitmap_base::Char< BITS > | |
| Helper abstraction for a byte contained in the bitmap | 1813 |
| L4Re::Util::Bitmap_base::Word< BITS > | |
| Helper abstraction for a word contained in the bitmap | 1814 |
| L4Re::Util::Br_manager | |
| Buffer-register (BR) manager for L4::Server | 1815 |
| L4Re::Util::Br_manager_hooks | |
| Predefined server-loop hooks for a server loop using the Br_manager | 1823 |
| L4Re::Util::Br_manager_timeout_hooks | |
| Predefined server-loop hooks for a server with using the Br_manager and a timeout queue . . . | 1826 |
| L4Re::Util::Cap_alloc_base | |
| Capability allocator | 1829 |
| L4Re::Util::Counter< COUNTER > | |
| Counter for Counting_cap_alloc with variable data width | 1830 |
| L4Re::Util::Counter_atomic< COUNTER > | |
| Thread safe version of counter for Counting_cap_alloc | 1831 |
| L4Re::Util::Counting_cap_alloc< COUNTERTYPE, Dbg > | |
| Internal reference-counting cap allocator | 1833 |
| L4Re::Util::Dataspace_svr | |
| Dataspace server class | 1839 |
| L4Re::Util::Event_buffer_consumer_t< PAYLOAD > | |
| An event buffer consumer | 1846 |
| L4Re::Util::Event_buffer_t< PAYLOAD > | |
| Event_buffer utility class | 1849 |
| L4Re::Util::Event_svr< SVR > | |
| Convenience wrapper for implementing an event server | 1853 |
| L4Re::Util::Event_t< PAYLOAD > | |
| Convenience wrapper for getting access to an event object | 1855 |
| L4Re::Util::Item_alloc_base | |
| Item allocator | 1859 |
| L4Re::Util::Names::Name | |
| Name class | 1860 |
| L4Re::Util::Names::Name_space | |
| Abstract server-side implementation of the L4::Namespace interface | 1863 |

| | |
|--|------|
| L4Re::Util::Object_registry | |
| A registry that manages server objects and their attached IPC gates for a single server loop for a specific thread | 1868 |
| L4Re::Util::Ref_cap< T > | |
| Automatic capability that implements automatic free and unmap of the capability selector . . . | 1875 |
| L4Re::Util::Ref_del_cap< T > | |
| Automatic capability that implements automatic free and unmap+delete of the capability selector | 1877 |
| L4Re::Util::Registry_server< LOOP_HOOKS > | |
| A server loop object which has a Object_registry included | 1878 |
| L4Re::Util::Smart_cap_auto< Unmap_flags > | |
| Helper for Unique_cap and Unique_del_cap | 1884 |
| L4Re::Util::Smart_count_cap< Unmap_flags > | |
| Helper for Ref_cap and Ref_del_cap | 1885 |
| L4Re::Util::Vcon_svr< SVR > | |
| Console server template class | 1886 |
| L4Re::Util::Video::Goos_svr | |
| Goos server class | 1887 |
| L4Re::Vfs::Be_file | |
| Boiler plate class for implementing an open file for L4Re::Vfs | 1891 |
| L4Re::Vfs::Be_file_system | |
| Boilerplate class for implementing a L4Re::Vfs::File_system | 1896 |
| L4Re::Vfs::Directory | |
| Interface for a POSIX file that is a directory | 1899 |
| L4Re::Vfs::File | |
| The basic interface for an open POSIX file | 1904 |
| L4Re::Vfs::File_system | |
| Basic interface for an L4Re::Vfs file system | 1908 |
| L4Re::Vfs::Fs | |
| POSIX File-system related functionality | 1910 |
| L4Re::Vfs::Generic_file | |
| The common interface for an open POSIX file | 1915 |
| L4Re::Vfs::Mman | |
| Interface for POSIX memory management | 1924 |
| L4Re::Vfs::Ops | |
| Interface for the POSIX backends of an application | 1925 |
| L4Re::Vfs::Regular_file | |
| Interface for a POSIX file that provides regular file semantics | 1928 |
| L4Re::Vfs::Special_file | |
| Interface for a POSIX file that provides special file semantics | 1938 |
| L4Re::Video::Color_component | |
| A color component | 1941 |
| L4Re::Video::Goos | |
| Class that abstracts framebuffers | 1946 |
| L4Re::Video::Goos::Info | |
| Information structure of a Goos | 1956 |
| L4Re::Video::Pixel_info | |
| Pixel information | 1958 |
| L4Re::Video::View | |
| View of a framebuffer | 1969 |
| L4Re::Video::View::Info | |
| Information structure of a view | 1976 |
| l4re_aux_t | |
| Auxiliary descriptor | 1978 |
| l4re_ds_stats_t | |
| Information about the data space | 1979 |
| l4re_elf_aux_mword_t | |
| Auxiliary vector element for a single unsigned data word | 1979 |

| | | |
|--|--|------|
| l4re_elf_aux_t | Generic header for each auxiliary vector element | 1980 |
| l4re_elf_aux_vma_t | Auxiliary vector element for a reserved virtual memory area | 1981 |
| l4re_env_cap_entry_t | Entry in the L4Re environment array for the named initial objects | 1981 |
| l4re_env_t | Initial environment data structure | 1983 |
| l4re_event_t | Event structure used in buffer | 1986 |
| l4re_video_color_component_t | Color component structure | 1987 |
| l4re_video_goos_info_t | Goos information structure | 1987 |
| l4re_video_pixel_info_t | Pixel_info structure | 1989 |
| l4re_video_view_info_t | View information structure | 1990 |
| l4re_video_view_t | C representation of a goos view | 1992 |
| l4shmc_ringbuf_head_t | Head field of a ring buffer | 1993 |
| l4shmc_ringbuf_t | Ring buffer | 1994 |
| l4util_l4mod_info | Base module structure | 1995 |
| l4util_l4mod_mod | A single module | 1997 |
| l4util_mb_addr_range_t | INT-15, AX=E820 style "AddressRangeDescriptor" ...with a "size" parameter on the front which is the structure size - 4, pointing to the next one, up until the full buffer length of the memory map has been reached | 1998 |
| l4util_mb_apm_t | APM BIOS info | 1999 |
| l4util_mb_drive_t | Drive Info structure | 1999 |
| l4util_mb_info_t | MultiBoot Info description | 2000 |
| l4util_mb_mod_t | The structure type "mod_list" is used by the multiboot_info structure | 2002 |
| l4util_mb_vbe_ctrl_t | VBE controller information | 2003 |
| l4util_mb_vbe_mode_t | VBE mode information | 2003 |
| L4vbus::Device | Device on a L4vbus::Vbus | 2007 |
| L4vbus::Gpio_module | A Gpio_module groups multiple GPIO pins together | 2017 |
| L4vbus::Gpio_module::Pin_slice | A slice of the pins provided by this module | 2024 |
| L4vbus::Gpio_pin | A GPIO pin | 2024 |
| L4vbus::Icu | Vbus Interrupt controller API | 2033 |
| L4vbus::Pci_dev | A PCI device | 2037 |
| L4vbus::Pci_host_bridge | A Pci host bridge | 2043 |

| | |
|--|------|
| L4vbus::Pm< DEC > | |
| Power-management API mixin | 2049 |
| L4vbus::Vbus | |
| The virtual bus (Vbus) interface | 2052 |
| l4vbus_device_t | |
| Detailed information about a vbus device | 2063 |
| l4vbus_resource_t | |
| Description of a single vbus resource | 2064 |
| L4vcpu::State | |
| C++ implementation of state word in the vCPU area | 2065 |
| L4vcpu::Vcpu | |
| C++ implementation of the vCPU save state area | 2067 |
| L4virtio::Device | |
| IPC interface for virtio over L4 IPC | 2080 |
| L4virtio::Driver::Block_device | |
| Simple class for accessing a virtio block device synchronously | 2090 |
| L4virtio::Driver::Block_device::Handle | |
| Handle to an ongoing request | 2098 |
| L4virtio::Driver::Device | |
| Client-side implementation for a general virtio device | 2098 |
| L4virtio::Driver::Virtio_net_device | |
| Simple class for accessing a virtio net device | 2111 |
| L4virtio::Driver::Virtio_net_device::Packet | |
| Structure for a network packet (header including data) with maximum size, assuming that no extra features have been negotiated | 2120 |
| L4virtio::Driver::Virtqueue | |
| Driver-side implementation of a Virtqueue | 2121 |
| L4virtio::Ptr< T > | |
| Pointer used in virtio descriptors | 2130 |
| L4virtio::Svr::Bad_descriptor | |
| Exception used by Queue to indicate descriptor errors | 2133 |
| L4virtio::Svr::Block_dev_base< Ds_data > | |
| Base class for virtio block devices | 2135 |
| L4virtio::Svr::Block_request< Ds_data > | |
| A request to read or write data | 2144 |
| L4virtio::Svr::Console::Control_message | |
| Virtio console control message | 2146 |
| L4virtio::Svr::Console::Control_request | |
| Specialised Virtqueue::Request providing access to control message payload | 2148 |
| L4virtio::Svr::Console::Device | |
| Base class implementing a virtio console device with L4Re -based notification handling | 2149 |
| L4virtio::Svr::Console::Device_port | |
| A console port with associated read/write state | 2163 |
| L4virtio::Svr::Console::Features | |
| Virtio console specific feature bits | 2167 |
| L4virtio::Svr::Console::Port | |
| Representation of a Virtio console port | 2172 |
| L4virtio::Svr::Console::Port::Transition | |
| State transition from last report state to current state | 2176 |
| L4virtio::Svr::Console::Virtio_con | |
| Base class implementing a virtio console functionality | 2177 |
| L4virtio::Svr::Data_buffer | |
| Abstract data buffer | 2194 |
| L4virtio::Svr::Dev_config | |
| Abstraction for L4-Virtio device config memory | 2198 |
| L4virtio::Svr::Dev_features | |
| Type for device feature bitmap | 2210 |

| | |
|--|------|
| L4virtio::Svr::Dev_status | |
| Type of the device status register | 2213 |
| L4virtio::Svr::Device_t< DATA > | |
| Server-side L4-VIRTIO device stub | 2216 |
| L4virtio::Svr::Driver_mem_list_t< DATA > | |
| List of driver memory regions assigned to a single L4-VIRTIO transport instance | 2223 |
| L4virtio::Svr::Driver_mem_region_t< DATA > | |
| Region of driver memory, that shall be managed locally | 2229 |
| L4virtio::Svr::Request_processor | |
| Encapsulate the state for processing a VIRTIO request | 2236 |
| L4virtio::Svr::Scmi::Base_attr_t | |
| SCMI base protocol attributes | 2243 |
| L4virtio::Svr::Scmi::Base_proto | |
| Base class for the SCMI base protocol | 2244 |
| L4virtio::Svr::Scmi::Perf_proto | |
| Base class for the SCMI performance protocol | 2246 |
| L4virtio::Svr::Scmi::Performance_attr_t | |
| SCMI performance protocol attributes | 2249 |
| L4virtio::Svr::Scmi::Performance_describe_level_t | |
| SCMI performance describe level | 2250 |
| L4virtio::Svr::Scmi::Performance_describe_levels_n_t | |
| SCMI performance describe levels numbers | 2250 |
| L4virtio::Svr::Scmi::Performance_domain_attr_t | |
| SCMI performance domain protocol attributes | 2252 |
| L4virtio::Svr::Scmi::Proto< OBSERV > | |
| Base class for all protocols | 2254 |
| L4virtio::Svr::Scmi::Scmi_dev | |
| A server implementation of the virtio-scmi protocol | 2255 |
| L4virtio::Svr::Scmi::Scmi_hdr_t | |
| SCMI header | 2259 |
| L4virtio::Svr::Virtio_gpio< Request_handler, Epiface > | |
| A server implementation of the virtio-gpio protocol | 2261 |
| L4virtio::Svr::Virtio_gpio< Request_handler, Epiface >::Host_irq | |
| Handler for the host irq | 2265 |
| L4virtio::Svr::Virtio_gpio< Request_handler, Epiface >::Irq_handler | |
| Handler for an gpio pin irq | 2269 |
| L4virtio::Svr::Virtio_gpio< Request_handler, Epiface >::Request_processor | |
| Generic handler for the Virtio requests | 2270 |
| L4virtio::Svr::Virtio_i2c< Request_handler, Epiface > | |
| A server implementation of the virtio-i2c protocol | 2273 |
| L4virtio::Svr::Virtio_i2c< Request_handler, Epiface >::Host_irq | |
| Handler for the host irq | 2277 |
| L4virtio::Svr::Virtio_i2c< Request_handler, Epiface >::Request_processor | |
| Handler for the Virtio requests | 2281 |
| L4virtio::Svr::Virtio_rng< Rnd_state, Epiface > | |
| A server implementation of the virtio-rng protocol | 2284 |
| L4virtio::Svr::Virtio_rng< Rnd_state, Epiface >::Host_irq | |
| Handler for the host irq | 2288 |
| L4virtio::Svr::Virtio_rng< Rnd_state, Epiface >::Request_processor | |
| Handler for the Virtio requests | 2292 |
| L4virtio::Svr::Virtio_spi< Spi_request_handler, Epiface > | |
| A server implementation of the virtio-spi protocol | 2294 |
| L4virtio::Svr::Virtio_spi< Spi_request_handler, Epiface >::Host_irq | |
| Handler for the host IRQ | 2299 |
| L4virtio::Svr::Virtio_spi< Spi_request_handler, Epiface >::Request_processor | |
| Handler for the Virtio requests | 2303 |
| L4virtio::Svr::Virtqueue | |
| Virtqueue implementation for the device | 2306 |

| | |
|---|------|
| L4virtio::Svr::Virtqueue::Head_desc | |
| VIRTIO request, essentially a descriptor from the available ring | 2318 |
| L4virtio::Virtqueue | |
| Low-level Virtqueue | 2321 |
| L4virtio::Virtqueue::Avail | |
| Type of available ring, this is read-only for the host | 2338 |
| L4virtio::Virtqueue::Avail::Flags | |
| Flags of the available ring | 2339 |
| L4virtio::Virtqueue::Desc | |
| Descriptor in the descriptor table | 2340 |
| L4virtio::Virtqueue::Desc::Flags | |
| Type for descriptor flags | 2342 |
| L4virtio::Virtqueue::Used | |
| Used ring | 2344 |
| L4virtio::Virtqueue::Used::Flags | |
| Flags for the used ring | 2345 |
| L4virtio::Virtqueue::Used_elem | |
| Type of an element of the used ring | 2347 |
| l4virtio_block_config_t | |
| Device configuration for block devices | 2348 |
| l4virtio_block_discard_t | |
| Structure used for the write zeroes and discard commands | 2349 |
| l4virtio_block_header_t | |
| Header structure of a request for a block device | 2350 |
| l4virtio_config_hdr_t | |
| L4-VIRTIO config header, provided in shared data space | 2351 |
| l4virtio_config_queue_t | |
| Queue configuration entry | 2352 |
| l4virtio_input_absinfo_t | |
| Information about the absolute axis in the underlying evdev implementation | 2353 |
| l4virtio_input_config_t | |
| Device configuration for input devices | 2354 |
| l4virtio_input_devids_t | |
| Device ID information for the device | 2354 |
| l4virtio_input_event_t | |
| Single event in event or status queue | 2355 |
| l4virtio_net_config_t | |
| Device configuration for network devices | 2356 |
| l4virtio_net_header_t | |
| Header structure of a request for a network device | 2356 |
| L4virtio_port | |
| A Port on the Virtio Net Switch | 2357 |
| Mac_addr | |
| A wrapper class around the value of a MAC address | 2363 |
| Mac_table< Size > | |
| Mac_table manages a 1:n association between ports and MAC addresses | 2363 |
| Net_transfer | |
| A network request to only a single destination | 2366 |
| Rm | |
| Region map | 2369 |
| Rm::Area | |
| An area is a range of virtual addresses which is reserved, see L4Re::Rm::reserve_area() | 2383 |
| Rm::F | |
| Rm flags definitions | 2384 |
| Rm::Region | |
| A region is a range of virtual addresses which is backed by content | 2386 |
| Rm::Unique_region< T > | |
| Unique region | 2387 |

| | |
|--|------|
| Switch_factory | |
| The IPC interface for creating ports | 2392 |
| utrace::Ring_buffer< SEQUENCE_TYPE > | |
| Ring buffer | 2396 |
| utrace::Ring_buffer_consumer< SEQUENCE_TYPE, ITEM_TYPE, GENERATION_PTR > | |
| Blocking ring buffer consumer | 2400 |
| utrace::Ring_buffer_consumer_raw< SEQUENCE_TYPE, ITEM_TYPE, GENERATION_PTR > | |
| Polling ring buffer consumer | 2406 |
| utrace::Ring_buffer_producer< SEQUENCE_TYPE, ITEM_TYPE, GENERATION_PTR > | |
| Ring buffer producer | 2411 |
| utrace::Ring_slot< ITEM_TYPE, GENERATION_PTR > | |
| Ring buffer slot | 2415 |
| utrace::Ring_status< SEQUENCE_TYPE > | |
| Ring buffer status area | 2417 |
| utrace::Tracebuffer | |
| Tracebuffer abstraction | 2420 |
| utrace::Tracebuffer::Index_desc | |
| Mapping of the dynamic tracebuffer log indexes to names | 2423 |
| Virtio_net | |
| The Base class of a Port | 2423 |
| Virtio_net_request | |
| Abstraction for a network request | 2428 |
| Virtio_switch | |
| The Virtio switch contains all ports and processes network requests | 2431 |
| Virtio_vlan_mangle | |
| Class for VLAN packet rewriting | 2436 |

Chapter 12

File Index

12.1 File List

Here is a list of all documented files with brief descriptions:

| | |
|--|------|
| pkg/drivers-frst/include/asm_access_gen.h | 2446 |
| pkg/drivers-frst/include/hw_mmio_register_block | 2447 |
| pkg/drivers-frst/include/hw_register_block | 2448 |
| pkg/drivers-frst/include/io_regblock.h | 2451 |
| pkg/drivers-frst/include/io_regblock_port.h | 2453 |
| pkg/drivers-frst/include/Makefile | 2497 |
| pkg/drivers-frst/include/poll_timeout_counter.h | 2454 |
| pkg/drivers-frst/include/ARCH-amd64/asm_access.h | 2441 |
| pkg/drivers-frst/include/ARCH-arm/asm_access.h | 2442 |
| pkg/drivers-frst/include/ARCH-arm64/asm_access.h | 2443 |
| pkg/drivers-frst/include/ARCH-mips/asm_access.h | 2444 |
| pkg/drivers-frst/include/ARCH-ppc32/asm_access.h | 2444 |
| pkg/drivers-frst/include/ARCH-riscv/asm_access.h | 2444 |
| pkg/drivers-frst/include/ARCH-sparc/asm_access.h | 2445 |
| pkg/drivers-frst/include/ARCH-x86/asm_access.h | 2446 |
| pkg/drivers-frst/uart/include/device.h | 2455 |
| pkg/drivers-frst/uart/include/Makefile | 2497 |
| pkg/drivers-frst/uart/include/uart_16550.h | 2456 |
| pkg/drivers-frst/uart/include/uart_16550_dw.h | 2457 |
| pkg/drivers-frst/uart/include/uart_apb.h | 2458 |
| pkg/drivers-frst/uart/include/uart_base.h | 2458 |
| pkg/drivers-frst/uart/include/uart_bcm2835.h | 2460 |
| pkg/drivers-frst/uart/include/uart_cadence.h | 2460 |
| pkg/drivers-frst/uart/include/uart_dcc-v6.h | 2461 |
| pkg/drivers-frst/uart/include/uart_dm.h | 2461 |
| pkg/drivers-frst/uart/include/uart_dummy.h | 2462 |
| pkg/drivers-frst/uart/include/uart_geni.h | 2462 |
| pkg/drivers-frst/uart/include/uart_imx.h | 2463 |
| pkg/drivers-frst/uart/include/uart_leon3.h | 2464 |
| pkg/drivers-frst/uart/include/uart_linflex.h | 2465 |
| pkg/drivers-frst/uart/include/uart_lpuart.h | 2465 |
| pkg/drivers-frst/uart/include/uart_mvebu.h | 2466 |
| pkg/drivers-frst/uart/include/uart_of.h | 2466 |
| pkg/drivers-frst/uart/include/uart_omap35x.h | 2467 |
| pkg/drivers-frst/uart/include/uart_pl011.h | 2467 |

| | |
|---|------|
| pkg/drivers-frst/uart/include/uart_s3c2410.h | 2468 |
| pkg/drivers-frst/uart/include/uart_sa1000.h | 2469 |
| pkg/drivers-frst/uart/include/uart_sbi.h | 2469 |
| pkg/drivers-frst/uart/include/uart_sh.h | 2470 |
| pkg/drivers-frst/uart/include/uart_sifive.h | 2470 |
| pkg/drivers-frst/uart/include/uart_tegra-tcu.h | 2471 |
| pkg/l4re-core/ned/doc/tutorial.lua | 2471 |
| pkg/l4re-core/ned/lib/include/cmd_control | 2474 |
| pkg/l4re-core/ned/lib/include/Makefile | 2497 |
| pkg/uvmm/configs/vmm.lua | 2475 |
| pkg/virtio-net-switch/server/switch/debug.h | 2483 |
| pkg/virtio-net-switch/server/switch/filter.cc | 2485 |
| pkg/virtio-net-switch/server/switch/filter.h | 2486 |
| pkg/virtio-net-switch/server/switch/mac_addr.h | 2486 |
| pkg/virtio-net-switch/server/switch/mac_table.h | 2487 |
| pkg/virtio-net-switch/server/switch/main.cc | 2489 |
| pkg/virtio-net-switch/server/switch/Makefile | 2498 |
| pkg/virtio-net-switch/server/switch/options.cc | 2498 |
| pkg/virtio-net-switch/server/switch/options.h | 2500 |
| pkg/virtio-net-switch/server/switch/port.h | 2501 |
| pkg/virtio-net-switch/server/switch/port_ixl.h | 2503 |
| pkg/virtio-net-switch/server/switch/port_l4virtio.h | 2505 |
| pkg/virtio-net-switch/server/switch/request.h | 2509 |
| pkg/virtio-net-switch/server/switch/request_ixl.h | 2510 |
| pkg/virtio-net-switch/server/switch/request_l4virtio.h | 2512 |
| pkg/virtio-net-switch/server/switch/stats.h | 2515 |
| pkg/virtio-net-switch/server/switch/switch.cc | 2516 |
| pkg/virtio-net-switch/server/switch/switch.h | 2519 |
| pkg/virtio-net-switch/server/switch/virtio_net.h | 2520 |
| pkg/virtio-net-switch/server/switch/virtio_net_buffer.h | 2524 |
| pkg/virtio-net-switch/server/switch/vlan.h | 2524 |
| amd64/l4/sys/__kip-arch.h | 2565 |
| amd64/l4/sys/__vcpu-arch.h | |
| AMD64-specific vCPU interface | 2567 |
| amd64/l4/sys/ktrace_events.h | 2580 |
| amd64/l4/sys/linkage.h | |
| Linkage | 2598 |
| amd64/l4/sys/segment.h | |
| Segment handling (AMD64) | 2525 |
| amd64/l4/sys/vm.h | 2605 |
| amd64/l4/sys/arch/cache.h | |
| Cache functions | 3135 |
| amd64/l4/sys/arch/consts.h | |
| Common L4 constants, AMD64 version | 3153 |
| amd64/l4/sys/arch/ipc.h | 3268 |
| amd64/l4/sys/arch/kip.h | 3457 |
| amd64/l4/sys/arch/l4int.h | |
| Fixed sized integer types, AMD64 version | 3314 |
| amd64/l4/sys/arch/platform_control.h | 3333 |
| amd64/l4/sys/arch/task.h | 3363 |
| amd64/l4/sys/arch/thread.h | 2611 |
| amd64/l4/sys/arch/utcb.h | |
| UTCB definitions for AMD64 | 3384 |
| amd64/l4/util/bitops_arch.h | |
| Amd64 bit manipulation functions | 2626 |
| amd64/l4/util/cpu.h | |
| CPU related functions | 2633 |

| | |
|---|------|
| amd64/l4/util/ irq.h | |
| Some PIC and hardware interrupt related functions | 2774 |
| amd64/l4/util/ mbi_argv.h | |
| Command line handling | 2638 |
| amd64/l4/util/ perform.h | |
| Performance Monitoring using P5/P6 Measurement Counters | 2534 |
| amd64/l4/util/ port_io.h | |
| X86 port I/O | 2546 |
| amd64/l4/util/ rdtsc.h | |
| Timestamp counter related functions | 2554 |
| amd64/l4/util/ spin.h | |
| Spinning for amd64 | 2563 |
| amd64/l4/util/arch/ l4_macros.h | |
| Main function | 2608 |
| amd64/l4/util/arch/ thread.h | 2611 |
| arm/l4/sys/ __kip-arch.h | 2565 |
| arm/l4/sys/ __vcpu-arch.h | |
| ARM-specific vCPU interface | 2569 |
| arm/l4/sys/ atomic.h | 3419 |
| arm/l4/sys/ ktrace_events.h | 2583 |
| arm/l4/sys/ linkage.h | |
| Linkage | 2599 |
| arm/l4/sys/ mem_op.h | |
| Memory access functions (ARM specific) | 2602 |
| arm/l4/sys/ syscall_defs.h | |
| Syscall entry definitions | 2604 |
| arm/l4/sys/ vm.h | |
| ARM virtualization interface | 2605 |
| arm/l4/sys/arch/ cache.h | |
| Cache functions | 3137 |
| arm/l4/sys/arch/ consts.h | |
| Common L4 constants, arm version | 3154 |
| arm/l4/sys/arch/ ipc.h | 3269 |
| arm/l4/sys/arch/ kip.h | 3458 |
| arm/l4/sys/arch/ l4int.h | |
| Fixed sized integer types, arm version | 3315 |
| arm/l4/sys/arch/ platform_control.h | 3333 |
| arm/l4/sys/arch/ task.h | 3363 |
| arm/l4/sys/arch/ thread.h | |
| ARM-specific thread related definitions | 2611 |
| arm/l4/sys/arch/ utcb.h | |
| UTCB definitions for ARM | 3386 |
| arm/l4/util/ bitops_arch.h | |
| ARM specific implementation of bitops functions | 2629 |
| arm/l4/util/ cpu.h | |
| CPU related functions | 2636 |
| arm/l4/util/ irq.h | |
| ARM specific implementation of irq functions | 2775 |
| arm/l4/util/ mbi_argv.h | |
| Multiboot | 2640 |
| arm/l4/util/arch/ l4_macros.h | |
| Main function | 2609 |
| arm/l4/util/arch/ thread.h | 2612 |
| amd64/l4/sys/ __kip-arch.h | 2566 |
| amd64/l4/sys/ __vcpu-arch.h | |
| ARM64-specific vCPU interface | 2573 |
| amd64/l4/sys/ ktrace_events.h | 2586 |
| amd64/l4/sys/ linkage.h | 2600 |

| | |
|---|------|
| arm64/l4/sys/vm.h | 2607 |
| arm64/l4/sys/arch/cache.h | |
| Cache functions | 3139 |
| arm64/l4/sys/arch/consts.h | |
| Common L4 constants, arm version | 3155 |
| arm64/l4/sys/arch/ipc.h | 3269 |
| arm64/l4/sys/arch/kip.h | 3458 |
| arm64/l4/sys/arch/l4int.h | |
| Fixed sized integer types, arm version | 3316 |
| arm64/l4/sys/arch/platform_control.h | 3334 |
| arm64/l4/sys/arch/task.h | 3363 |
| arm64/l4/sys/arch/thread.h | |
| ARM64-specific thread related definitions | 2612 |
| arm64/l4/sys/arch/utcb.h | |
| UTCB definitions for ARM64 | 3389 |
| contrib/libio-io/l4/io/io.h | 2643 |
| contrib/libio-io/l4/io/types.h | 3378 |
| l4/cxx/alloc.h | |
| Alloc list | 2646 |
| l4/cxx/arith | 2647 |
| l4/cxx/atomic.h | |
| Atomic template | 3419 |
| l4/cxx/avl_map | |
| AVL map | 2648 |
| l4/cxx/avl_set | |
| AVL set | 2651 |
| l4/cxx/avl_tree | |
| AVL tree | 2656 |
| l4/cxx/basic_ostream | |
| Basic IO stream | 2662 |
| l4/cxx/basic_vector.h | |
| Basic vector | 2666 |
| l4/cxx/bitfield | 2667 |
| l4/cxx/bitmap | 2669 |
| l4/cxx/dlist | 2690 |
| l4/cxx/elide_dtor | 2695 |
| l4/cxx/exceptions | |
| Base exceptions | 2696 |
| l4/cxx/hlist | 2700 |
| l4/cxx/iostream | |
| IO Stream | 2702 |
| l4/cxx/ipc_helper | |
| IPC helper | 2703 |
| l4/cxx/ipc_server | |
| IPC server loop | 3202 |
| l4/cxx/ipc_stream | |
| IPC stream | 2705 |
| l4/cxx/ipc_timeout_queue | 2723 |
| l4/cxx/l4iostream | |
| L4 IO stream | 2725 |
| l4/cxx/l4types.h | |
| L4 Types | 2726 |
| l4/cxx/list | 2727 |
| l4/cxx/list_alloc | 2731 |
| l4/cxx/lock_guard.h | |
| Lock guard implementation | 2737 |
| l4/cxx/main_thread | |
| Main thread | 2738 |

| | |
|---|------|
| l4/cxx/minmax | 2740 |
| l4/cxx/numeric | 2741 |
| l4/cxx/observer | 2741 |
| l4/cxx/pair | |
| Pair implementation | 2742 |
| l4/cxx/ref_ptr | 2744 |
| l4/cxx/ref_ptr_list | |
| Implementation of a list of ref-ptr-managed objects | 2747 |
| l4/cxx/result | 2749 |
| l4/cxx/slab_alloc | 2751 |
| l4/cxx/slist | 2754 |
| l4/cxx/static_container | 2757 |
| l4/cxx/static_vector | 2758 |
| l4/cxx/std_alloc | 2758 |
| l4/cxx/std_ops | 2759 |
| l4/cxx/string | 2759 |
| l4/cxx/string.h | |
| String | 2762 |
| l4/cxx/thread | |
| Thread implementation | 3368 |
| l4/cxx/type_list | 2764 |
| l4/cxx/type_traits | 2764 |
| l4/cxx/unique_ptr | 2769 |
| l4/cxx/unique_ptr_list | |
| Implementation of a list of unique-ptr-managed objects | 2770 |
| l4/cxx/utils | 2772 |
| l4/cxx/weak_ref | 2772 |
| l4/cxx/bits/bst.h | |
| AVL tree | 2672 |
| l4/cxx/bits/bst_base.h | |
| AVL tree | 2676 |
| l4/cxx/bits/bst_iter.h | |
| AVL tree | 2679 |
| l4/cxx/bits/list_basics.h | 2682 |
| l4/cxx/bits/smart_ptr_list.h | |
| Implementation of a list of smart-pointer-managed objects | 2684 |
| l4/cxx/bits/type_traits.h | 2687 |
| l4/irq/irq.h | |
| IRQ handling routines | 2777 |
| l4/l4re_vfs/backend | 2785 |
| l4/l4re_vfs/vfs.h | 2813 |
| l4/l4re_vfs/impl/default_ops_impl.h | 2788 |
| l4/l4re_vfs/impl/fd_store.h | 2789 |
| l4/l4re_vfs/impl/fd_store_impl.h | 2790 |
| l4/l4re_vfs/impl/ns_fs.h | 2790 |
| l4/l4re_vfs/impl/ns_fs_impl.h | 2791 |
| l4/l4re_vfs/impl/ro_file.h | 2796 |
| l4/l4re_vfs/impl/ro_file_impl.h | 2796 |
| l4/l4re_vfs/impl/vcon_stream.h | 2798 |
| l4/l4re_vfs/impl/vcon_stream_impl.h | 2798 |
| l4/l4re_vfs/impl/vfs_impl.h | 2801 |
| l4/l4virtio/l4virtio | 2827 |
| l4/l4virtio/virtio.h | 2890 |
| l4/l4virtio/virtio_block.h | 2893 |
| l4/l4virtio/virtio_input.h | 2894 |
| l4/l4virtio/virtio_net.h | 2523 |
| l4/l4virtio/virtqueue | 2894 |
| l4/l4virtio/client/l4virtio | 2825 |

| | |
|---|------|
| l4/l4virtio/client/ virtio-block | 2844 |
| l4/l4virtio/client/ virtio-net | 2822 |
| l4/l4virtio/server/ l4virtio | 2829 |
| l4/l4virtio/server/ virtio | 2840 |
| l4/l4virtio/server/ virtio-block | 2848 |
| l4/l4virtio/server/ virtio-console | 2854 |
| l4/l4virtio/server/ virtio-console-device | 2860 |
| l4/l4virtio/server/ virtio-gpio-device | 2864 |
| l4/l4virtio/server/ virtio-i2c-device | 2869 |
| l4/l4virtio/server/ virtio-rng-device | 2874 |
| l4/l4virtio/server/ virtio-scmi-device | 2876 |
| l4/l4virtio/server/ virtio-spi-device | 2885 |
| l4/libedid/ edid.h | 2899 |
| l4/re/ cap_alloc | |
| Abstract capability-allocator interface | 3004 |
| l4/re/ console | 2938 |
| l4/re/ consts | |
| Constants | 3171 |
| l4/re/ consts.h | |
| Constants | 3156 |
| l4/re/ dataspace | |
| Dataspace interface | 2938 |
| l4/re/ dataspace-sys.h | |
| Dataspace protocol definition | 2941 |
| l4/re/ dbg_events | 2942 |
| l4/re/ debug | |
| Debug interface | 3021 |
| l4/re/ dma_space | 2942 |
| l4/re/ elf_aux.h | |
| Auxiliary information for binaries | 2945 |
| l4/re/ env | |
| Environment interface | 2947 |
| l4/re/ env.h | |
| Environment interface | 2949 |
| l4/re/ error_helper | |
| Error helper | 2953 |
| l4/re/ event | 3025 |
| l4/re/ event-sys.h | 2956 |
| l4/re/ event.h | |
| Events | 2909 |
| l4/re/ event_enums.h | 2956 |
| l4/re/ inhibitor | 2971 |
| l4/re/ inhibitor-sys.h | 2971 |
| l4/re/ itas | 2972 |
| l4/re/ l4aux.h | |
| Auxiliary definitions | 2973 |
| l4/re/ log | |
| Log interface | 2974 |
| l4/re/ log-sys.h | |
| Log protocol definition | 2976 |
| l4/re/ mem_alloc | |
| Memory allocator interface | 2976 |
| l4/re/ mem_alloc-sys.h | |
| Memory allocator protocol definitions | 2979 |
| l4/re/ mmio_space | |
| Interface definition to emit MMIO-like accesses via IPC | 2980 |
| l4/re/ namespace | |
| Namespace interface | 2981 |

| | |
|--|------|
| l4/re/namespace-sys.h | |
| Namespace protocol definitions | 2984 |
| l4/re/parent | |
| Parent interface | 2985 |
| l4/re/parent-sys.h | |
| Parent protocol definition | 2987 |
| l4/re/protocols.h | |
| L4Re Protocol Constants (C version) | 2987 |
| l4/re/random | |
| Random number generator interface definition | 2988 |
| l4/re/remote_access | 2990 |
| l4/re/rm | |
| Region mapper interface | 2991 |
| l4/re/rm-sys.h | |
| Region mapper protocol definitions | 2996 |
| l4/re/shared_cap | |
| Shared_cap / Shared_del_cap | 3057 |
| l4/re/unique_cap | |
| Unique_cap / Unique_del_cap | 3065 |
| l4/re/c/dataspace.h | |
| Data space C interface | 2900 |
| l4/re/c/debug.h | |
| Debug C interface | 2484 |
| l4/re/c/dma_space.h | |
| DMA space C interface | 2903 |
| l4/re/c/event.h | |
| Event C interface | 2906 |
| l4/re/c/event_buffer.h | 2910 |
| l4/re/c/inhibitor.h | |
| Inhibitor C interface | 2911 |
| l4/re/c/log.h | |
| Log C interface | 2914 |
| l4/re/c/mem_alloc.h | |
| Memory allocator C interface | 2915 |
| l4/re/c/namespace.h | |
| Namespace functions, C interface | 2917 |
| l4/re/c/parent.h | |
| Parent C interface | 2919 |
| l4/re/c/rm.h | |
| Region map interface, C interface | 2921 |
| l4/re/c/util/cap_alloc.h | |
| Capability allocator C interface | 2925 |
| l4/re/c/util/kumem_alloc.h | |
| Kumem allocator utility C interface | 2927 |
| l4/re/c/util/video/goos_fb.h | |
| Framebuffer utility functionality | 2929 |
| l4/re/c/video/colors.h | 2930 |
| l4/re/c/video/goos.h | 2932 |
| l4/re/c/video/view.h | 2935 |
| l4/re/impl/dataspace_impl.h | |
| Dataspace client stub implementation | 2963 |
| l4/re/impl/mem_alloc_impl.h | |
| Memory allocator client stub implementation | 2965 |
| l4/re/impl/namespace_impl.h | |
| Namespace client stub implementation | 2966 |
| l4/re/impl/rm_impl.h | |
| Region map client stub implementation | 2969 |

| | |
|---|------|
| l4/re/util/bitmap_cap_alloc | |
| Bitmap capability allocator | 2997 |
| l4/re/util/br_manager | 2999 |
| l4/re/util/cap | |
| Capability utility functions | 3002 |
| l4/re/util/cap_alloc | |
| Capability allocator | 3007 |
| l4/re/util/cap_alloc_impl.h | |
| Capability allocator implementation | 3010 |
| l4/re/util/counting_cap_alloc | |
| Reference-counting capability allocator | 3013 |
| l4/re/util/dataspace_svr | 3018 |
| l4/re/util/debug | 3022 |
| l4/re/util/env_ns | 3024 |
| l4/re/util/event | 3028 |
| l4/re/util/event_buffer | 3030 |
| l4/re/util/event_svr | 3031 |
| l4/re/util/icu_svr | 3032 |
| l4/re/util/item_alloc | |
| Item allocator | 3035 |
| l4/re/util/kumem_alloc | |
| Kumem allocator helper | 3037 |
| l4/re/util/meta | 3323 |
| l4/re/util/name_space_svr | 3038 |
| l4/re/util/object_registry | 3044 |
| l4/re/util/poll_timeout_kipclock | 3046 |
| l4/re/util/region_mapping | |
| Region handling | 3047 |
| l4/re/util/region_mapping_svr | 3053 |
| l4/re/util/reply_cap_hooks | 3056 |
| l4/re/util/shared_cap | |
| Shared_cap / Shared_del_cap | 3061 |
| l4/re/util/unique_cap | |
| Unique_cap / Unique_del_cap | 3067 |
| l4/re/util/vcon_svr | 3069 |
| l4/re/util/video/goos_fb | 3070 |
| l4/re/util/video/goos_svr | 3072 |
| l4/re/video/colors | 3074 |
| l4/re/video/goos | 3075 |
| l4/re/video/goos-sys.h | |
| Goos protocol definition | 3078 |
| l4/re/video/view | 3080 |
| l4/shmc/ringbuf.h | 3080 |
| l4/shmc/shmc.h | |
| Shared memory library header file | 3089 |
| l4/sigma0/sigma0.h | |
| Sigma0 interface | 3094 |
| l4/sys/_kernel_object_impl.h | 3098 |
| l4/sys/_ktrace-impl.h | |
| L4 kernel event tracing | 3098 |
| l4/sys/_l4_fpage.h | 3101 |
| l4/sys/_platform_control-arm.h | 3105 |
| l4/sys/_task-arm.h | 3106 |
| l4/sys/_timeout.h | 3106 |
| l4/sys/_typeinfo.h | |
| Type information handling | 3109 |
| l4/sys/_vcpu-arm.h | 3121 |
| l4/sys/_vm-svm.h | 3122 |

| | |
|--|------|
| l4/sys/__vm-vmx.h | 3124 |
| l4/sys/arm_smccc | |
| ARM secure monitor call functions | 3130 |
| l4/sys/arm_smccc.h | |
| ARM secure monitor call functions | 3132 |
| l4/sys/assert.h | |
| Low-level assert implementation | 3413 |
| l4/sys/cache.h | |
| Cache-consistency functions | 3141 |
| l4/sys/capability | |
| L4::Cap related definitions | 3146 |
| l4/sys/compiler.h | |
| L4 compiler related defines | 3149 |
| l4/sys/consts.h | |
| Common constants | 3158 |
| l4/sys/debugger | |
| The debugger interface specifies common debugging related definitions | 3238 |
| l4/sys/debugger.h | |
| Debugger related definitions | 3241 |
| l4/sys/err.h | |
| Error codes | 3247 |
| l4/sys/exception | |
| Exception C++ interface | 3249 |
| l4/sys/factory | |
| Common factory related definitions | 3250 |
| l4/sys/factory.h | |
| Common factory related definitions | 3254 |
| l4/sys/icu | |
| Interrupt controller | 3260 |
| l4/sys/icu.h | |
| Interrupt controller | 3262 |
| l4/sys/iommu | 3268 |
| l4/sys/ipc.h | |
| Common IPC interface | 3270 |
| l4/sys/ipc_gate | |
| The C++ IPC gate interface | 3279 |
| l4/sys/ipc_gate.h | |
| The C IPC gate interface, see L4::lpc_gate for the C++ interface | 3281 |
| l4/sys/irq | |
| C++ Irq interface | 3284 |
| l4/sys/irq.h | |
| C Irq interface | 2779 |
| l4/sys/kdebug.h | |
| Functionality for invoking the kernel debugger | 3288 |
| l4/sys/kdump.h | |
| Functionality for dumping kernel information | 3305 |
| l4/sys/kernel_object.h | |
| Kernel object system calls | 3307 |
| l4/sys/kip | 3309 |
| l4/sys/kip.h | |
| Kernel Info Page access functions | 3458 |
| l4/sys/kobject | 3312 |
| l4/sys/ktrace.h | |
| L4 kernel event tracing | 3312 |
| l4/sys/l4int.h | |
| Fixed sized integer types, generic version | 3317 |
| l4/sys/memdesc.h | |
| Memory description functions | 3320 |

| | | |
|---|---|------|
| l4/sys/meta | Meta interface for getting dynamic type information about objects behind capabilities | 3324 |
| l4/sys/obj_info.h | Debugger related functions | 3325 |
| l4/sys/pager | Pager and lo_pager C++ interface | 3329 |
| l4/sys/platform_control | Platform control object | 3331 |
| l4/sys/platform_control.h | Platform control object | 3334 |
| l4/sys/rcv_endpoint | The C++ Receive endpoint interface | 3338 |
| l4/sys/rcv_endpoint.h | Receive endpoint C interface | 3340 |
| l4/sys/scheduler | Scheduler object functions | 3342 |
| l4/sys/scheduler.h | Scheduler object functions | 3344 |
| l4/sys/semaphore | Semaphore class definition | 3349 |
| l4/sys/semaphore.h | C semaphore interface | 3351 |
| l4/sys/smart_capability | L4::Capability class | 3354 |
| l4/sys/snd_destination | The C++ Sender destination interface | 3357 |
| l4/sys/snd_destination.h | Sender destination endpoint C interface | 3359 |
| l4/sys/task | Common task related definitions | 3360 |
| l4/sys/task.h | Common task related definitions | 3363 |
| l4/sys/thread | Common thread related definitions | 3370 |
| l4/sys/thread.h | Common thread related definitions | 2614 |
| l4/sys/thread_group | | 3373 |
| l4/sys/thread_group.h | | 3374 |
| l4/sys/typeinfo_svr | Type information server template | 3376 |
| l4/sys/types.h | Common L4 ABI Data Types | 3378 |
| l4/sys/utcb.h | UTCB definitions | 3391 |
| l4/sys/vcon | C++ Virtual console interface | 3400 |
| l4/sys/vcon.h | Virtual console interface | 3402 |
| l4/sys/vcpu.h | VCPU API | 3515 |
| l4/sys/vcpu_context | Hardware vCPU context interface | 3408 |
| l4/sys/vcpu_context.h | | 3409 |
| l4/sys/vm | Virtualization interface | 3409 |
| l4/sys/cxx/capability.h | | 3167 |
| l4/sys/cxx/consts | | 3172 |
| l4/sys/cxx/ipc_array | | 3173 |

| | |
|--|------|
| l4/sys/cxx/ipc_basics | 3176 |
| l4/sys/cxx/ipc_client | 3180 |
| l4/sys/cxx/ipc_epiface | 3183 |
| l4/sys/cxx/ipc_iface | |
| Interface Definition Language | 3188 |
| l4/sys/cxx/ipc_legacy | 3200 |
| l4/sys/cxx/ipc_ret_array | 3200 |
| l4/sys/cxx/ipc_server | 3203 |
| l4/sys/cxx/ipc_server_loop | 3208 |
| l4/sys/cxx/ipc_string | 3212 |
| l4/sys/cxx/ipc_types | 3213 |
| l4/sys/cxx/ipc_varg | 3222 |
| l4/sys/cxx/limits | 3227 |
| l4/sys/cxx/smart_capability_1x | 3229 |
| l4/sys/cxx/types | 3232 |
| l4/umalloc/umalloc.h | |
| Public API for the basic next-fit memory allocator | 3411 |
| l4/util/assert.h | |
| Some useful assert-style macros | 3416 |
| l4/util/atomic.h | |
| Atomic operations header and generic implementations | 3421 |
| l4/util/backtrace.h | |
| Backtrace | 3428 |
| l4/util/base64.h | |
| Base 64 encoding and decoding functions adapted from Bob Trower 08/04/01 | 3430 |
| l4/util/bitops.h | |
| Bit manipulation functions | 3432 |
| l4/util/elf.h | |
| ELF definition | 3437 |
| l4/util/getopt.h | |
| Getopt | 3454 |
| l4/util/keymap.h | |
| Event to ASCII key mapping | 3456 |
| l4/util/kip.h | 3463 |
| l4/util/kprintf.h | |
| Printf using the kernel debugger | 3464 |
| l4/util/l4_macros.h | |
| Some useful generic macros, L4f version | 2609 |
| l4/util/l4mod.h | |
| L4mod structures and constants | 3465 |
| l4/util/list_alloc.h | |
| Simple list-based allocator | 3467 |
| l4/util/lock.h | |
| Simple lock implementation | 3470 |
| l4/util/mb_info.h | |
| Multiboot info structure as defined by GRUB | 3472 |
| l4/util/parse_cmd.h | |
| Comfortable command-line parsing | 3480 |
| l4/util/printf_helpers.h | 3482 |
| l4/util/rand.h | |
| Simple Pseudo-Random Number Generator | 3482 |
| l4/util/splitlog2.h | |
| Split a range in log2 aligned and size-aligned chunks | 3483 |
| l4/util/thread.h | |
| Low-level Thread Functions | 2623 |
| l4/util/util.h | 3485 |
| l4/utrace/ring_buffer | |
| Shared-memory multiple-producer multiple-consumer head-drop ring buffer implementation | 3486 |

| | |
|---|------|
| l4/utrace/utrace | |
| L4Re tracebuffer access | 3491 |
| l4/vbus/vbus | 3494 |
| l4/vbus/vbus.h | |
| Description of the vbus C API | 3495 |
| l4/vbus/vbus_generic | 3499 |
| l4/vbus/vbus_gpio | 3499 |
| l4/vbus/vbus_gpio-ops.h | 3501 |
| l4/vbus/vbus_gpio.h | 3501 |
| l4/vbus/vbus_i2c.h | 3502 |
| l4/vbus/vbus_inhibitor.h | 3502 |
| l4/vbus/vbus_interfaces.h | |
| This header contains the definition of VBUS sub-interfaces and convenience functions to work with the interface IDs | 3502 |
| l4/vbus/vbus_mcspi.h | 3506 |
| l4/vbus/vbus_pci | 3506 |
| l4/vbus/vbus_pci-ops.h | 3507 |
| l4/vbus/vbus_pci.h | 3508 |
| l4/vbus/vbus_pm-ops.h | 3508 |
| l4/vbus/vbus_pm.h | 3509 |
| l4/vbus/vbus_types.h | |
| This header file contains descriptions of vbus related data types and constants | 3509 |
| l4/vbus/vdevice-ops.h | 3513 |
| l4/vcpu/vcpu | |
| VCPU support library (C++ interface) | 3513 |
| l4/vcpu/vcpu.h | |
| VCPU support library (C interface) | 3518 |
| riscv/l4/sys/__kip-arch.h | 2566 |
| riscv/l4/sys/__vcpu-arch.h | |
| RISC-V-specific vCPU interface | 2576 |
| riscv/l4/sys/ktrace_events.h | 2589 |
| riscv/l4/sys/linkage.h | |
| Linkage | 2600 |
| riscv/l4/sys/vm.h | 2607 |
| riscv/l4/sys/arch/cache.h | |
| Cache functions | 3143 |
| riscv/l4/sys/arch/consts.h | |
| Common L4 constants, RISC-V version | 3165 |
| riscv/l4/sys/arch/ipc.h | 3277 |
| riscv/l4/sys/arch/kip.h | 3464 |
| riscv/l4/sys/arch/l4int.h | |
| Fixed sized integer types, RISC-V version | 3318 |
| riscv/l4/sys/arch/platform_control.h | 3337 |
| riscv/l4/sys/arch/task.h | 3367 |
| riscv/l4/sys/arch/thread.h | 2625 |
| riscv/l4/sys/arch/utcb.h | |
| UTCB definitions for RISC-V | 3395 |
| x86/l4/sys/__kip-arch.h | 2567 |
| x86/l4/sys/__vcpu-arch.h | |
| X86-specific vCPU interface | 2578 |
| x86/l4/sys/ktrace_events.h | 2595 |
| x86/l4/sys/linkage.h | |
| Linkage | 2601 |
| x86/l4/sys/segment.h | |
| Segment handling (x86) | 2532 |
| x86/l4/sys/vm.h | 2608 |
| x86/l4/sys/arch/cache.h | |
| Cache functions | 3145 |

| | |
|---|------|
| x86/l4/sys/arch/ consts.h | |
| Common L4 constants, x86 version | 3166 |
| x86/l4/sys/arch/ ipc.h | |
| L4 IPC System Calls, x86 | 3277 |
| x86/l4/sys/arch/ kip.h | 3464 |
| x86/l4/sys/arch/ l4int.h | |
| Fixed sized integer types, x86 version | 3319 |
| x86/l4/sys/arch/ platform_control.h | 3338 |
| x86/l4/sys/arch/ task.h | 3368 |
| x86/l4/sys/arch/ thread.h | 2625 |
| x86/l4/sys/arch/ utcb.h | |
| UTCB definitions for x86 | 3397 |
| x86/l4/util/ bitops_arch.h | |
| X86 bit manipulation functions | 2630 |
| x86/l4/util/ cpu.h | |
| CPU related functions | 2636 |
| x86/l4/util/ irq.h | |
| Some PIC and hardware interrupt related functions | 2783 |
| x86/l4/util/ mbi_argv.h | |
| Command line handling | 2641 |
| x86/l4/util/ perform.h | |
| Performance Monitoring using P5/P6 Measurement Counters | 2540 |
| x86/l4/util/ port_io.h | |
| X86 port I/O | 2550 |
| x86/l4/util/ rdtsc.h | |
| Timestamp counter related functions | 2558 |
| x86/l4/util/ spin.h | |
| Spinning for x86 | 2564 |
| x86/l4/util/arch/ l4_macros.h | |
| Main function | 2610 |
| x86/l4/util/arch/ thread.h | 2625 |

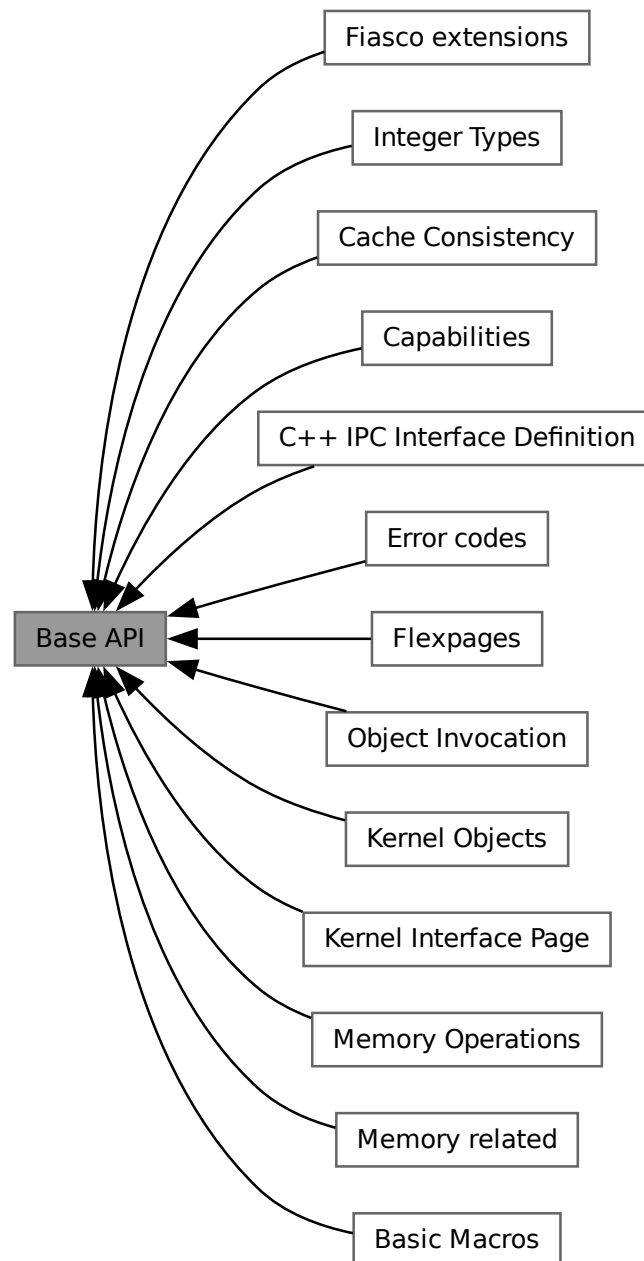
Chapter 13

Topic Documentation

13.1 Base API

Interfaces for all kinds of base functionality.

Collaboration diagram for Base API:



Topics

- Basic Macros [152](#)
[L4](#) standard macros for header files, function definitions, and public APIs etc.
- Fiasco extensions [157](#)

| | |
|--|-----|
| <i>Extensions of the Fiasco L4 implementation.</i> | |
| • Flexpages | 175 |
| <i>Flexpage-related API.</i> | |
| • C++ IPC Interface Definition | 193 |
| <i>APIs for defining IPC interfaces using C++ as language.</i> | |
| • Cache Consistency | 194 |
| <i>Various functions for cache consistency.</i> | |
| • Memory related | 198 |
| <i>Memory related constants, data types and functions.</i> | |
| • Error codes | 208 |
| <i>Common error codes.</i> | |
| • Object Invocation | 210 |
| <i>API for L4 object invocation.</i> | |
| • Kernel Objects | 278 |
| <i>API of kernel objects.</i> | |
| • Kernel Interface Page | 433 |
| <i>Kernel Interface Page.</i> | |
| • Capabilities | 445 |
| <i>C interface for capabilities.</i> | |
| • Memory Operations | 449 |
| <i>Operations for memory access.</i> | |
| • Integer Types | 452 |

Files

- file [cache.h](#)
Cache-consistency functions.
- file [compiler.h](#)
L4 compiler related defines.
- file [consts.h](#)
Common constants.
- file [debugger.h](#)
Debugger related definitions.
- file [factory.h](#)
Common factory related definitions.
- file [icu](#)
Interrupt controller.
- file [icu.h](#)
Interrupt controller.

- file [ipc.h](#)
Common IPC interface.
- file [irq.h](#)
C Irq interface.
- file [kip](#)
- file [kip.h](#)
Kernel Info Page access functions.
- file [memdesc.h](#)
Memory description functions.
- file [semaphore.h](#)
C semaphore interface.
- file [types.h](#)
Common L4 ABI Data Types.
- file [consts.h](#)
Common L4 constants, arm version.
- file [consts.h](#)
Common L4 constants, arm version.
- file [consts.h](#)
Common L4 constants, AMD64 version.
- file [consts.h](#)
Common L4 constants, x86 version.
- file [ipc.h](#)
L4 IPC System Calls, x86.
- file [consts.h](#)
Common L4 constants, RISC-V version.

13.1.1 Detailed Description

Interfaces for all kinds of base functionality.

Some notes on Inter Process Communication (IPC)

IPC in L4 is always synchronous and unbuffered: a message is transferred from the sender to the recipient if and only if the recipient has invoked a corresponding IPC operation. The sender blocks until this happens or a timeout specified by the sender elapsed without the destination becoming ready to receive.

13.1.2 Basic Macros

L4 standard macros for header files, function definitions, and public APIs etc.

Collaboration diagram for Basic Macros:



Files

- file [linkage.h](#)
Linkage.
- file [linkage.h](#)
Linkage.
- file [linkage.h](#)
Linkage.
- file [linkage.h](#)
Linkage.

Macros

- **#define L4_INLINE**
L4 Inline function attribute.
- **#define L4_ALWAYS_INLINE**
Always inline a function.
- **#define L4_BEGIN_DECLS**
Start section with C types and functions.
- **#define L4_END_DECLS**
End section with C types and functions.
- **#define L4_NOTHROW**
Mark a function declaration and definition as never throwing an exception.
- **#define L4_EXPORT**
Attribute to mark functions, variables, and data types as being exported from a library.
- **#define L4_HIDDEN**
Attribute to mark functions, variables, and data types as being explicitly hidden from users of a library.
- **#define L4_CONSTEXPR**
Constexpr function attribute.
- **#define L4_NORETURN**
Noreturn function attribute.
- **#define L4_NOINSTRUMENT**
No instrumentation function attribute.
- **#define L4_LIKELY(x)**
Expression is likely to execute.
- **#define L4_UNLIKELY(x)**
Expression is unlikely to execute.
- **#define L4_STICKY(x)**
Mark symbol sticky (even not there).
- **#define L4_DEPRECATED(s)**
Mark symbol deprecated.
- **#define L4_stringify_helper(x)**
stringify helper.
- **#define L4_stringify(x)**
stringify.
- **#define L4_CV**
Define calling convention.
- **#define L4_CV**
Define calling convention.
- **#define L4_CV**
Define calling convention.
- **#define L4_CV**
Define calling convention.
- **#define L4_CV**
Define calling convention.
- **#define L4_CV**
Define calling convention.

Functions

- unsigned long [l4_align_stack_for_direct_fncall](#) (unsigned long stack)
Specify the desired alignment of the stack pointer.
- void [l4_barrier](#) (void)
Memory barrier.
- void [l4_mb](#) (void)
Memory barrier.
- void [l4_wmb](#) (void)
Write memory barrier.
- [L4_NORETURN](#) void [l4_infinite_loop](#) (void)
Infinite loop.

13.1.2.1 Detailed Description

[L4](#) standard macros for header files, function definitions, and public APIs etc.

Include File

```
#include <l4/sys/compiler.h>
```

13.1.2.2 Macro Definition Documentation

13.1.2.2.1 L4_EXPORT

```
#define L4_EXPORT
```

Attribute to mark functions, variables, and data types as being exported from a library.

All data types, functions, and global variables that shall be exported from a library shall be marked with this attribute. The default may become to hide everything that is not marked as `L4_EXPORT` from the users of a library and provide the possibility for aggressive optimization of all those internal functionality of a library.

Usage:

```
class L4_EXPORT My_class
{
    ...
};

int L4_EXPORT function(void);

int L4_EXPORT global_data; // global data is not recommended
```

Definition at line 220 of file [compiler.h](#).

Referenced by [l4io_iterate_devices\(\)](#), [l4io_lookup_device\(\)](#), [l4io_lookup_resource\(\)](#), [l4io_release_iomem\(\)](#), [l4io_release_ioport\(\)](#), [l4io_request_all_ioports\(\)](#), [l4io_request_icu\(\)](#), [l4io_request_iomem\(\)](#), [l4io_request_iomem_region\(\)](#), [l4io_request_ioport\(\)](#), [l4io_request_resource_iomem\(\)](#), [l4re_inhibitor_acquire\(\)](#), and [l4re_inhibitor_release\(\)](#).

13.1.2.2.2 L4_HIDDEN

```
#define L4_HIDDEN
```

Attribute to mark functions, variables, and data types as being explicitly hidden from users of a library.

This attribute is intended for functions, data, and data types that shall never be visible outside of a library. In particular, for shared libraries this may result in much faster code within the library and short linking times.

```
class L4_HIDDEN My_class
{
    ...
};

int L4_HIDDEN function(void);

int L4_HIDDEN global_data; // global data is not recommended
```

Definition at line 217 of file [compiler.h](#).

13.1.2.2.3 L4_NOTHROW

```
#define L4_NOTHROW
```

Mark a function declaration and definition as never throwing an exception.

(Also for C code).

This macro shall be used to mark C and C++ functions that never throw any exception. Note that also C functions may throw exceptions according to the compilers ABI and shall be marked with L4_NOTHROW if they never do. In C++ this is equivalent to `throw()`.

```
int foo() L4_NOTHROW;
...
int foo() L4_NOTHROW
{
    ...
    return result;
}
```

Definition at line 167 of file [compiler.h](#).

Referenced by [__l4_kdebug_3_text\(\)](#), [__l4_kdebug_op\(\)](#), [__l4_kdebug_op_1\(\)](#), [__l4_kdebug_text\(\)](#), [l4_msgtag_t::flags\(\)](#), [l4_msgtag_t::has_error\(\)](#), [l4_msgtag_t::is_exception\(\)](#), [l4_msgtag_t::is_io_page_fault\(\)](#), [l4_msgtag_t::is_page_fault\(\)](#), [l4_msgtag_t::is_sigma0\(\)](#), [l4_msgtag_t::items\(\)](#), [l4_arm_smccc_call\(\)](#), [l4_bdr\(\)](#), [l4_bytes_to_mwords\(\)](#), [l4_cache_clean_data\(\)](#), [l4_cache_coherent\(\)](#), [l4_cache_dma_coherent\(\)](#), [l4_cache_dma_coherent_full\(\)](#), [l4_cache_flush_data\(\)](#), [l4_cache_inv_data\(\)](#), [l4_capability_equal\(\)](#), [l4_capability_next\(\)](#), [l4_debugger_add_image_info\(\)](#), [l4_debugger_get_object_name\(\)](#), [l4_debugger_global_id\(\)](#), [l4_debugger_kobj_to_id\(\)](#), [l4_debugger_query_log_name\(\)](#), [l4_debugger_query_log_typeid\(\)](#), [l4_debugger_query_obj_infos\(\)](#), [l4_debugger_query_object_name\(\)](#), [l4_debugger_set_object_name\(\)](#), [l4_debugger_switch_log\(\)](#), [l4_error\(\)](#), [l4_factory_create\(\)](#), [l4_factory_create_factory\(\)](#), [l4_factory_create_gate\(\)](#), [l4_factory_create_irq\(\)](#), [l4_factory_create_task\(\)](#), [l4_factory_create_thread\(\)](#), [l4_factory_create_thread_group\(\)](#), [l4_factory_create_vcpu_context\(\)](#), [l4_factory_create_vm\(\)](#), [l4_fpage\(\)](#), [l4_fpage_all\(\)](#), [l4_fpage_contains\(\)](#), [l4_fpage_invalid\(\)](#), [l4_fpage_ioport\(\)](#), [l4_fpage_memaddr\(\)](#), [l4_fpage_obj\(\)](#), [l4_fpage_page\(\)](#), [l4_fpage_rights\(\)](#), [l4_fpage_set_rights\(\)](#), [l4_fpage_size\(\)](#), [l4_fpage_type\(\)](#), [l4_icu_bind\(\)](#), [l4_icu_bind_u\(\)](#), [l4_icu_info\(\)](#), [l4_icu_info_u\(\)](#), [l4_icu_mask\(\)](#), [l4_icu_mask_u\(\)](#), [l4_icu_msi_info\(\)](#), [l4_icu_msi_info_u\(\)](#), [l4_icu_set_mode\(\)](#), [l4_icu_set_mode_u\(\)](#), [l4_icu_unbind\(\)](#), [l4_icu_unbind_u\(\)](#), [l4_icu_unmask\(\)](#), [l4_icu_unmask_u\(\)](#), [l4_iofpage\(\)](#), [l4_ipc\(\)](#), [l4_ipc_call\(\)](#), [l4_ipc_error\(\)](#), [l4_ipc_error_code\(\)](#), [l4_ipc_is_rcv_error\(\)](#), [l4_ipc_is_snd_error\(\)](#), [l4_ipc_receive\(\)](#), [l4_ipc_reply\(\)](#), [l4_ipc_reply_and_wait\(\)](#), [l4_ipc_send\(\)](#), [l4_ipc_send_and_wait\(\)](#), [l4_ipc_sleep\(\)](#), [l4_ipc_sleep_ms\(\)](#), [l4_ipc_sleep_us\(\)](#), [l4_ipc_timeout\(\)](#), [l4_ipc_to_errno\(\)](#), [l4_ipc_wait\(\)](#), [l4_irq_bind_vcpu\(\)](#), [l4_irq_bind_vcpu_u\(\)](#), [l4_irq_detach\(\)](#), [l4_irq_detach_u\(\)](#), [l4_irq_receive\(\)](#), [l4_irq_receive_u\(\)](#), [l4_irq_trigger\(\)](#), [l4_irq_trigger_u\(\)](#), [l4_irq_unmask\(\)](#), [l4_irq_unmask_u\(\)](#), [l4_irq_wait\(\)](#), [l4_irq_wait_u\(\)](#), [l4_is_fpage_valid\(\)](#), [l4_is_fpage_writable\(\)](#), [l4_is_invalid_cap\(\)](#), [l4_is_valid_cap\(\)](#), [l4_kd_enter\(\)](#), [l4_kernel_info_get_mem_desc_end\(\)](#), [l4_kernel_info_get_mem_desc_is_virtual\(\)](#), [l4_kernel_info_get_mem_desc_start\(\)](#), [l4_kernel_info_get_mem_desc_subtype\(\)](#), [l4_kernel_info_get_mem_desc_type\(\)](#), [l4_kernel_info_get_mem_descs\(\)](#),

l4_kernel_info_get_num_mem_descs(), l4_kernel_info_set_mem_desc(), l4_kernel_info_version_offset(), l4_kip(), l4_kip_clock(), l4_kip_clock_lw(), l4_kip_clock_ns(), l4_kip_version(), l4_kip_version_string(), l4_map_control(), l4_map_obj_control(), l4_msgtag(), l4_msgtag_flags(), l4_msgtag_has_error(), l4_msgtag_is_exception(), l4_msgtag_is_io_page_fault(), l4_msgtag_is_page_fault(), l4_msgtag_is_sigma0(), l4_msgtag_items(), l4_msgtag_label(), l4_msgtag_words(), l4_obj_fpage(), l4_platform_ctl_cpu_allow_shutdown(), l4_platform_ctl_cpu_disable(), l4_platform_ctl_cpu_enable(), l4_platform_ctl_set_task_asid(), l4_platform_ctl_system_shutdown(), l4_platform_ctl_system_suspend(), l4_rcv_timeout(), l4_round_page(), l4_round_size(), l4_sched_cpu_set(), l4_sched_param(), l4_scheduler_idle_time(), l4_scheduler_info(), l4_scheduler_info_with_classes(), l4_scheduler_is_online(), l4_scheduler_run_thread(), l4_semaphore_down(), l4_semaphore_up(), l4_sleep(), l4_sleep_forever(), l4_snd_timeout(), l4_sndpage_add(), l4_task_add_ku_mem(), l4_task_cap_equal(), l4_task_cap_valid(), l4_task_delete_obj(), l4_task_map(), l4_task_release_cap(), l4_task_unmap(), l4_task_unmap_batch(), l4_task_vgicc_map(), l4_thread_arm_set_tpidruro(), l4_thread_control_alien(), l4_thread_control_bind(), l4_thread_control_commit(), l4_thread_control_exc_handler(), l4_thread_control_pager(), l4_thread_control_start(), l4_thread_ex_regs(), l4_thread_ex_regs_ret(), l4_thread_ex_regs_ret_u(), l4_thread_ex_regs_u(), l4_thread_group_add(), l4_thread_group_remove(), l4_thread_modify_sender_add(), l4_thread_modify_sender_commit(), l4_thread_modify_sender_start(), l4_thread_register_del_irq(), l4_thread_register_doorbell_irq(), l4_thread_stats_time(), l4_thread_switch(), l4_thread_vcpu_control(), l4_thread_vcpu_control_ext(), l4_thread_vcpu_control_ext_u(), l4_thread_vcpu_control_u(), l4_thread_vcpu_resume_commit(), l4_thread_vcpu_resume_start(), l4_thread_yield(), l4_timeout(), l4_timeout_abs(), l4_timeout_get(), l4_timeout_is_absolute(), l4_timeout_rel(), l4_timeout_rel_get(), l4_touch_ro(), l4_touch_rw(), l4_trunc_page(), l4_trunc_size(), l4_usleep(), l4_utcb(), l4_utcb_br(), l4_utcb_exc(), l4_utcb_exc_is_ex_regs_exception(), l4_utcb_exc_is_ex_regs_exception(), l4_utcb_exc_is_pf(), l4_utcb_exc_is_pf(), l4_utcb_exc_pc(), l4_utcb_exc_pc(), l4_utcb_exc_pc_set(), l4_utcb_exc_pfa(), l4_utcb_exc_pfa(), l4_utcb_exc_typeval(), l4_utcb_exc_typeval(), l4_utcb_inherit_fpu(), l4_utcb_mr(), l4_utcb_mr64_idx(), l4_utcb_tcr(), l4_vcon_get_attr(), l4_vcon_get_attr_u(), l4_vcon_read(), l4_vcon_read_u(), l4_vcon_read_with_flags(), l4_vcon_send(), l4_vcon_send_u(), l4_vcon_set_attr(), l4_vcon_set_attr_raw(), l4_vcon_set_attr_u(), l4_vcon_write(), l4_vcon_write_u(), l4_vcpu_check_version(), l4_vm_vmx_clear(), l4_vm_vmx_field_len(), l4_vm_vmx_field_order(), l4_vm_vmx_get_caps(), l4_vm_vmx_get_caps_default1(), l4_vm_vmx_get_cr2_index(), l4_vm_vmx_get_hw_vmcs(), l4_vm_vmx_ptr_load(), l4_vm_vmx_read(), l4_vm_vmx_read_16(), l4_vm_vmx_read_32(), l4_vm_vmx_read_64(), l4_vm_vmx_read_nat(), l4_vm_vmx_set_hw_vmcs(), l4_vm_vmx_write(), l4_vm_vmx_write_16(), l4_vm_vmx_write_32(), l4_vm_vmx_write_64(), l4_vm_vmx_write_nat(), l4re_debug_obj_debug(), l4re_dma_space_associate(), l4re_dma_space_map(), l4re_dma_space_unmap(), l4re_ds_allocate(), l4re_ds_clear(), l4re_ds_copy_in(), l4re_ds_flags(), l4re_ds_info(), l4re_ds_map_info(), l4re_ds_size(), l4re_env(), l4re_env_cap_entry_t::l4re_env_cap_entry_t(), l4re_env_cap_entry_t::l4re_env_cap_entry_t(), l4re_env_get_cap(), l4re_env_get_cap_e(), l4re_env_get_cap_l(), l4re_event_get_axis_info(), l4re_event_get_buffer(), l4re_event_get_num_streams(), l4re_event_get_stream_info(), l4re_event_get_stream_info_for_id(), l4re_kip(), l4re_log_print(), l4re_log_print_srv(), l4re_log_printn(), l4re_log_printn_srv(), l4re_ma_alloc(), l4re_ma_alloc_align(), l4re_ma_alloc_align_srv(), l4re_ns_query_srv(), l4re_ns_query_to_srv(), l4re_ns_register_obj_srv(), l4re_rm_attach(), l4re_rm_attach_srv(), l4re_rm_attach_w_info(), l4re_rm_attach_w_info_srv(), l4re_rm_detach(), l4re_rm_detach_ds(), l4re_rm_detach_ds_unmap(), l4re_rm_detach_srv(), l4re_rm_detach_unmap(), l4re_rm_find(), l4re_rm_find_srv(), l4re_rm_free_area(), l4re_rm_free_area_srv(), l4re_rm_get_info(), l4re_rm_get_info_srv(), l4re_rm_reserve_area(), l4re_rm_reserve_area_srv(), l4re_rm_show_lists(), l4re_rm_show_lists_srv(), l4re_util_cap_alloc(), l4re_util_cap_free(), l4re_util_cap_free_um(), l4re_util_cap_last(), l4re_util_kumem_alloc(), l4re_video_goos_create_buffer(), l4re_video_goos_create_view(), l4re_video_goos_delete_buffer(), l4re_video_goos_delete_view(), l4re_video_goos_get_static_buffer(), l4re_video_goos_get_view(), l4re_video_goos_info(), l4re_video_goos_refresh(), l4re_video_view_get_info(), l4re_video_view_refresh(), l4re_video_view_set_info(), l4re_video_view_set_viewport(), l4re_video_view_stack(), l4util_micros2l4to(), l4vcpu_ext_alloc(), l4vcpu_irq_disable(), l4vcpu_irq_disable_save(), l4vcpu_irq_enable(), l4vcpu_irq_restore(), l4vcpu_is_irq_entry(), l4vcpu_is_page_fault_entry(), l4vcpu_print_state(), l4vcpu_wait_for_event(), l4virtio_config_queue(), l4virtio_device_config_ds(), l4virtio_device_notification_irq(), l4virtio_register_ds(), l4virtio_set_status(), l4_msgtag_t::label(), l4_msgtag_t::label(), umalloc_area_create(), and l4_msgtag_t::words().

13.1.2.3 Function Documentation

13.1.2.3.1 l4_align_stack_for_direct_fncall()

```
unsigned long l4_align_stack_for_direct_fncall (
    unsigned long stack) [inline]
```

Specify the desired alignment of the stack pointer.

BIGGEST_ALIGNMENT provides the largest alignment ever used for any data type on the target machine. This is normally identical to desired stack alignment. Align stack pointer for directly invoked functions.

The stack needs to be aligned to `L4_STACK_ALIGN` for being able to access certain data on the stack. On x86/AMD64, a function call is performed using the 'call' instruction decrementing the stack pointer and writing the return address onto the stack. The called function considers this when adapting the stack pointer after function entry. If the called function was not invoked by a 'call' instruction, the stack pointer is actually off by a machine word leading to stack alignment issues when executing SSE instructions.

This function fixes the stack pointer for directly invoked functions. For architectures not automatically pushing the stack pointer during a function call, just enforce the `L4_STACK_ALIGN` alignment.

Definition at line 284 of file [compiler.h](#).

References [L4_INLINE](#).

13.1.2.3.2 l4_infinite_loop()

```
L4_NORETURN void l4_infinite_loop (
    void ) [inline]
```

Infinite loop.

Will never return. Use [l4_sleep_forever\(\)](#) if at all possible.

Definition at line 358 of file [compiler.h](#).

References [l4_barrier\(\)](#), [L4_INLINE](#), and [L4_NORETURN](#).

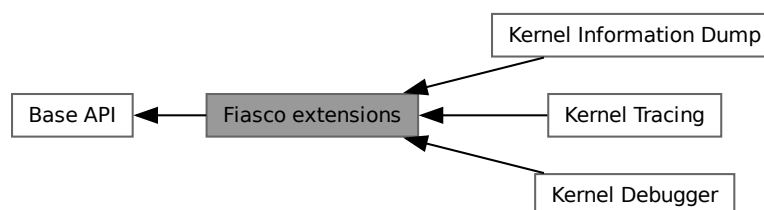
Here is the call graph for this function:



13.1.3 Fiasco extensions

Extensions of the Fiasco [L4](#) implementation.

Collaboration diagram for Fiasco extensions:



Topics

- [Kernel Debugger](#) 161
Kernel debugger related functionality.
- [Kernel Information Dump](#) 169
Kernel information dumping related functionality.
- [Kernel Tracing](#) 170
Kernel tracing related functionality.

Files

- file [__ktrace-impl.h](#)
L4 kernel event tracing.
- file [ktrace.h](#)
L4 kernel event tracing.
- file [obj_info.h](#)
Debugger related functions.
- file [segment.h](#)
Segment handling (AMD64).
- file [segment.h](#)
Segment handling (x86).

Functions

- long [fiasco_ldt_set](#) ([l4_cap_idx_t](#) task, void *ldt, unsigned int num_desc, unsigned int entry_number_start, [l4_utcb_t](#) *utcb)
Set LDT segments descriptors.
- long [fiasco_gdt_set](#) ([l4_cap_idx_t](#) thread, void *desc, unsigned int size, unsigned int entry_number_start, [l4_utcb_t](#) *utcb)
Set GDT segment descriptors.
- unsigned [fiasco_gdt_get_entry_offset](#) ([l4_cap_idx_t](#) thread, [l4_utcb_t](#) *utcb)
Return the offset of the entry in the GDT.

13.1.3.1 Detailed Description

Extensions of the Fiasco [L4](#) implementation.

13.1.3.2 Function Documentation

13.1.3.2.1 fiasco_gdt_get_entry_offset()

```
unsigned fiasco_gdt_get_entry_offset (
    l4_cap_idx_t thread,
    l4_utcb_t * utcb) [inline]
```

Return the offset of the entry in the GDT.

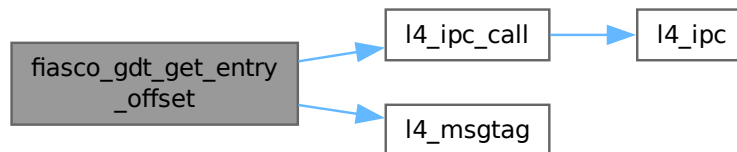
Parameters

| | |
|---------------|--|
| <i>thread</i> | Thread to get info from. |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |

Definition at line 166 of file [segment.h](#).

References [l4_ipc_call\(\)](#), [L4_IPC_NEVER](#), [l4_msgtag\(\)](#), [L4_PROTO_THREAD](#), and [L4_THREAD_X86_GDT_OP](#).

Here is the call graph for this function:



13.1.3.2.2 fiasco_gdt_set()

```
long fiasco_gdt_set (
    l4_cap_idx_t thread,
    void * desc,
    unsigned int size,
    unsigned int entry_number_start,
    l4_utcb_t * utcb) [inline]
```

Set GDT segment descriptors.

Fiasco supports 4 consecutive entries, starting at the value returned by [fiasco_gdt_get_entry_offset\(\)](#).

Parameters

| | |
|---------------|----------------------------------|
| <i>thread</i> | Thread to set the GDT entry for. |
| <i>desc</i> | Pointer to GDT descriptors. |

| | |
|---------------------------|--|
| <i>size</i> | Size of the descriptors in bytes (multiple of 8). |
| <i>entry_number_start</i> | Entry number to start (valid values: 0-3). |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |

Return values

| | |
|---------------|--|
| <i><0</i> | At least one provided GDT descriptor is considered unsafe by the kernel, and not all selected GDT descriptors have been updated. |
| <i>L4_EOK</i> | Success. |

Definition at line 220 of file [segment.h](#).

References [l4_ipc_call\(\)](#), [L4_IPC_NEVER](#), [l4_msgtag\(\)](#), [L4_PROTO_THREAD](#), and [L4_THREAD_X86_GDT_OP](#).

Here is the call graph for this function:



13.1.3.2.3 fiasco_ldt_set()

```

long fiasco_ldt_set (
    l4_cap_idx_t task,
    void * ldt,
    unsigned int num_desc,
    unsigned int entry_number_start,
    l4_utcb_t * utcb) [inline]

```

Set LDT segments descriptors.

Parameters

| | |
|---------------------------|--|
| <i>task</i> | Task to set the segment for. |
| <i>ldt</i> | Pointer to LDT hardware descriptors. |
| <i>num_desc</i> | Number of descriptors. |
| <i>entry_number_start</i> | Entry number to start. |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |

Return values

| | |
|-------------------------|--|
| <code>-L4_ENOSYS</code> | The kernel configuration doesn't support this feature. |
| <code>-L4_EINVAL</code> | Invalid descriptor or invalid entry number. |
| <code>L4_EOK</code> | Success. |

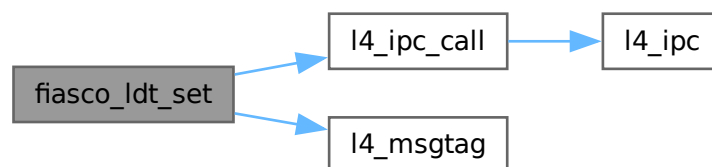
Note

This feature is not available if the kernel is configured with page table isolation.

Definition at line 153 of file [segment.h](#).

References [L4_EINVAL](#), [l4_ipc_call\(\)](#), [L4_IPC_NEVER](#), [l4_msgtag\(\)](#), [L4_PROTO_TASK](#), [L4_TASK_LDT_SET_X86_OP](#), [L4_TASK_LDT_X86_ENTRY_SIZE](#), and [L4_TASK_LDT_X86_MAX_ENTRIES](#).

Here is the call graph for this function:

**13.1.3.3 Kernel Debugger**

Kernel debugger related functionality.

Collaboration diagram for Kernel Debugger:

**Files**

- file [kdebug.h](#)

Functionality for invoking the kernel debugger.

Functions

- `l4_msgtag_t l4_debugger_set_object_name (l4_cap_idx_t cap, const char *name) L4_NOTHROW`
Set the name of a kernel object.
- `l4_msgtag_t l4_debugger_query_object_name (l4_cap_idx_t cap, unsigned id, char *name, unsigned size) L4_NOTHROW`
Get name of the kernel object with id `id`.
- `l4_msgtag_t l4_debugger_get_object_name (l4_cap_idx_t cap, char *name, unsigned size) L4_NOTHROW`
Get name of a kernel object.
- `unsigned long l4_debugger_global_id (l4_cap_idx_t cap) L4_NOTHROW`
Get the globally unique ID of the object behind a capability.
- `unsigned long l4_debugger_kobj_to_id (l4_cap_idx_t cap, l4_addr_t kobjp) L4_NOTHROW`
Get the globally unique ID of the object behind the kobject pointer.
- `l4_ret_t l4_debugger_query_log_typeid (l4_cap_idx_t cap, const char *name, unsigned idx) L4_NOTHROW`
Query the log-id for a log type.
- `l4_ret_t l4_debugger_query_log_name (l4_cap_idx_t cap, unsigned idx, char *name, unsigned namelen, char *shortname, unsigned shortnamelen) L4_NOTHROW`
Query the name of a log type given the ID.
- `l4_msgtag_t l4_debugger_switch_log (l4_cap_idx_t cap, const char *name, int on_off) L4_NOTHROW`
Set or unset log.
- `l4_msgtag_t l4_debugger_add_image_info (l4_cap_idx_t cap, l4_addr_t base, const char *name) L4_NOTHROW`
Add loaded image information for a task.

13.1.3.3.1 Detailed Description

Kernel debugger related functionality.

Attention

This API is subject to change!

This is a debugging facility, any call to any function might be invalid. Do not rely on it in any real code.

Include File

```
#include <l4/sys/debugger.h>
```

13.1.3.3.2 Function Documentation

13.1.3.3.2.1 l4_debugger_add_image_info()

```
l4_msgtag_t l4_debugger_add_image_info (
    l4_cap_idx_t cap,
    l4_addr_t base,
    const char * name) [inline]
```

Add loaded image information for a task.

Parameters

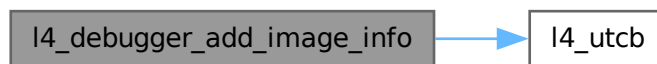
| | |
|-------------|---|
| <i>cap</i> | Capability which refers to the task object. |
| <i>base</i> | Load base address of image. |
| <i>name</i> | Image base name. |

This is a debugging facility, the call might be invalid.

Definition at line 472 of file [debugger.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:

**13.1.3.3.2.2 l4_debugger_get_object_name()**

```

l4_msgtag_t l4_debugger_get_object_name (
    l4_cap_idx_t cap,
    char * name,
    unsigned size) [inline]
  
```

Get name of a kernel object.

Parameters

| | | |
|-----|-------------|---|
| | <i>cap</i> | Capability which refers to the kernel object. |
| out | <i>name</i> | Buffer to copy the name into. The buffer must be allocated by the caller. |
| | <i>size</i> | Length of the <code>name</code> buffer. |

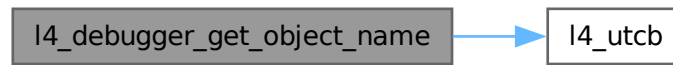
Returns

Syscall return tag

Definition at line 465 of file [debugger.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



13.1.3.3.2.3 l4_debugger_global_id()

```

unsigned long l4_debugger_global_id (
    l4_cap_idx_t cap) [inline]
  
```

Get the globally unique ID of the object behind a capability.

Parameters

| | |
|------------|------------|
| <i>cap</i> | Capability |
|------------|------------|

Return values

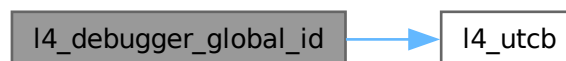
| | |
|------------------|--------------------------|
| <i>~0UL</i> | Capability is not valid. |
| <i>otherwise</i> | Global debugger id. |

This is a debugging facility, the call might be invalid.

Definition at line 423 of file [debugger.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



13.1.3.3.2.4 l4_debugger_kobj_to_id()

```
unsigned long l4_debugger_kobj_to_id (  
    l4_cap_idx_t cap,  
    l4_addr_t kobjp) [inline]
```

Get the globally unique ID of the object behind the kobject pointer.

Parameters

| | |
|--------------|-----------------|
| <i>cap</i> | Capability |
| <i>kobjp</i> | Kobject pointer |

Return values

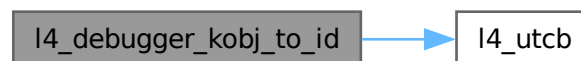
| | |
|------------------|--|
| <i>~0UL</i> | The capability or the kobject pointer are invalid. |
| <i>otherwise</i> | The globally unique id. |

This is a debugging facility, the call might be invalid.

Definition at line 429 of file [debugger.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



13.1.3.3.2.5 l4_debugger_query_log_name()

```
l4_ret_t l4_debugger_query_log_name (  
    l4_cap_idx_t cap,  
    unsigned idx,  
    char * name,  
    unsigned namelen,  
    char * shortname,  
    unsigned shortnamelen) [inline]
```

Query the name of a log type given the ID.

Parameters

| | |
|---------------------|---|
| <i>cap</i> | Debugger capability. |
| <i>idx</i> | ID to query. |
| <i>name</i> | Buffer to copy name to. |
| <i>namelen</i> | Buffer length of name. |
| <i>shortname</i> | Buffer to copy <code>shortname</code> to. |
| <i>shortnamelen</i> | Buffer length of <code>shortname</code> . |

Return values

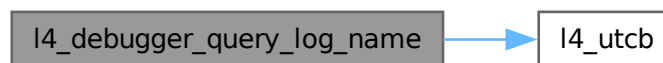
| | |
|----|---------|
| 0 | Success |
| <0 | Error |

This is a debugging facility, the call might be invalid.

Definition at line 442 of file [debugger.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:

**13.1.3.3.2.6 l4_debugger_query_log_typeid()**

```

l4_ret_t l4_debugger_query_log_typeid (
    l4_cap_idx_t cap,
    const char * name,
    unsigned idx) [inline]
  
```

Query the log-id for a log type.

Parameters

| | |
|-------------|--------------------------------------|
| <i>cap</i> | Debugger capability |
| <i>name</i> | Name to query for. |
| <i>idx</i> | Idx to start searching, start with 0 |

Returns

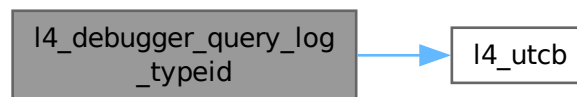
positive ID, or negative error code

This is a debugging facility, the call might be invalid.

Definition at line 435 of file [debugger.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:

**13.1.3.3.2.7 l4_debugger_query_object_name()**

```

l4_msgtag_t l4_debugger_query_object_name (
    l4_cap_idx_t cap,
    unsigned id,
    char * name,
    unsigned size) [inline]
  
```

Get name of the kernel object with Id *id*.

Parameters

| | | |
|-----|-------------|---|
| | <i>cap</i> | Capability of the debugger object. |
| | <i>id</i> | Global id of the object whose name is asked. |
| out | <i>name</i> | Buffer to copy the name into. The buffer must be allocated by the caller. |
| | <i>size</i> | Length of the <i>name</i> buffer. |

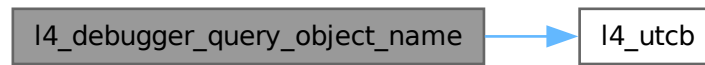
Returns

Syscall return tag

Definition at line 458 of file [debugger.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



13.1.3.3.2.8 l4_debugger_set_object_name()

```
l4_msgtag_t l4_debugger_set_object_name (  
    l4_cap_idx_t cap,  
    const char * name) [inline]
```

Set the name of a kernel object.

Parameters

| | |
|-------------|--|
| <i>cap</i> | Capability which refers to the kernel object. |
| <i>name</i> | Name of the kernel object that is e.g. displayed in the kernel debugger. |

This is a debugging facility, the call might be invalid.

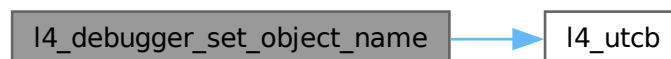
Examples

[examples/libs/shmc/prodcons.c](#), and [examples/sys/aliens/main.c](#).

Definition at line 416 of file [debugger.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



13.1.3.3.2.9 l4_debugger_switch_log()

```
l4_msgtag_t l4_debugger_switch_log (  
    l4_cap_idx_t cap,  
    const char * name,  
    int on_off) [inline]
```

Set or unset log.

Parameters

| | |
|---------------|---------------------------------|
| <i>cap</i> | Debugger object. |
| <i>name</i> | Name of the log type. |
| <i>on_off</i> | 1: turn log on, 0: turn log off |

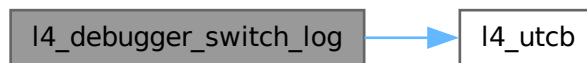
Returns

Syscall return tag

Definition at line 451 of file [debugger.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



13.1.3.4 Kernel Information Dump

Kernel information dumping related functionality.

Collaboration diagram for Kernel Information Dump:



Kernel information dumping related functionality.

Functions that dump various kernel internal information to the console. Probably only present in kernel debug builds.

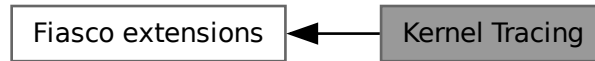
Include File

```
#include <l4/sys/kdump.h>
```

13.1.3.5 Kernel Tracing

Kernel tracing related functionality.

Collaboration diagram for Kernel Tracing:



Functions

- [l4_msgtag_t fiasco_tbuf_validate](#) (void)
Validate the kernel base debugger capability.
- [l4_umword_t fiasco_tbuf_log](#) (const char *text)
Create new trace-buffer entry with describing <text>.
- [l4_umword_t fiasco_tbuf_log_3val](#) (const char *text, [l4_umword_t](#) v1, [l4_umword_t](#) v2, [l4_umword_t](#) v3)
Create new trace-buffer entry with describing <text> and three additional values.
- [l4_umword_t fiasco_tbuf_log_binary](#) (const unsigned char *data)
Create new trace-buffer entry with binary data.
- void **fiasco_tbuf_clear** (void)
Clear trace-buffer.
- void **fiasco_tbuf_dump** (void)
Dump trace-buffer to kernel console.
- [l4_msgtag_t fiasco_tbuf_map_status](#) ([l4_fpage_t](#) *ku_mem)
Map kernel trace-buffer status page.
- [l4_msgtag_t fiasco_tbuf_map_slots](#) ([l4_fpage_t](#) *ku_mem)
Map kernel trace-buffer slots.

13.1.3.5.1 Detailed Description

Kernel tracing related functionality.

Attention

This API is subject to change!

This is a tracing facility for the Fiasco kernel trace buffer. Any call to any function might be invalid. Do not rely on it in any real code.

Include File

```
#include <l4/sys/ktrace.h>
```

13.1.3.5.2 Function Documentation

13.1.3.5.2.1 fiasco_tbuf_log()

```
l4_umword_t fiasco_tbuf_log (
    const char * text) [inline]
```

Create new trace-buffer entry with describing <text>.

Parameters

| | |
|-------------|--------------|
| <i>text</i> | Logging text |
|-------------|--------------|

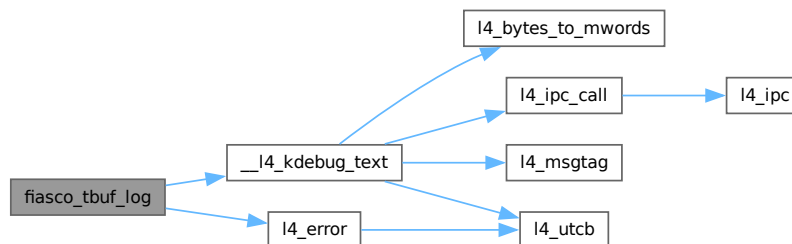
Returns

Pointer to trace-buffer entry

Definition at line 29 of file [__ktrace-impl.h](#).

References [__l4_kdebug_text\(\)](#), and [l4_error\(\)](#).

Here is the call graph for this function:



13.1.3.5.2.2 fiasco_tbuf_log_3val()

```
l4_umword_t fiasco_tbuf_log_3val (
    const char * text,
    l4_umword_t v1,
    l4_umword_t v2,
    l4_umword_t v3) [inline]
```

Create new trace-buffer entry with describing <text> and three additional values.

Parameters

| | |
|-------------|--------------|
| <i>text</i> | Logging text |
|-------------|--------------|

| | |
|-----------|--------------|
| <i>v1</i> | first value |
| <i>v2</i> | second value |
| <i>v3</i> | third value |

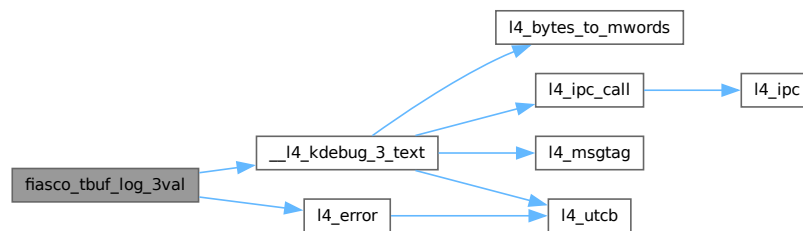
Returns

Pointer to trace-buffer entry

Definition at line 36 of file [__ktrace-impl.h](#).

References [__l4_kdebug_3_text\(\)](#), and [l4_error\(\)](#).

Here is the call graph for this function:

**13.1.3.5.2.3 fiasco_tbuf_log_binary()**

```
l4_umword_t fiasco_tbuf_log_binary (
    const unsigned char * data) [inline]
```

Create new trace-buffer entry with binary data.

Parameters

| | |
|-------------|-------------|
| <i>data</i> | binary data |
|-------------|-------------|

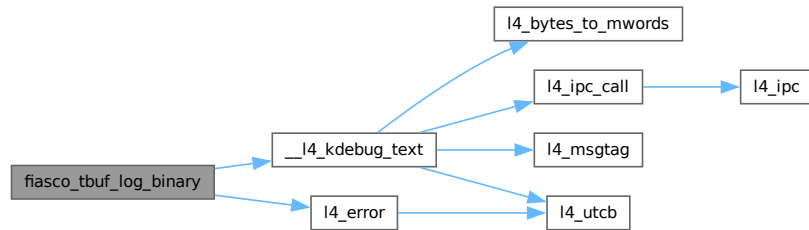
Returns

Pointer to trace-buffer entry

Definition at line 59 of file [__ktrace-impl.h](#).

References [__l4_kdebug_text\(\)](#), and [l4_error\(\)](#).

Here is the call graph for this function:



13.1.3.5.2.4 fiasco_tbuf_map_slots()

```
l4_msgtag_t fiasco_tbuf_map_slots (
    l4_fpage_t * ku_mem) [inline]
```

Map kernel trace-buffer slots.

Parameters

| | |
|---------------|--|
| <i>ku_mem</i> | Flexpage where to map the trace-buffer slots. Expected to be the exact size as defined by the status page. |
|---------------|--|

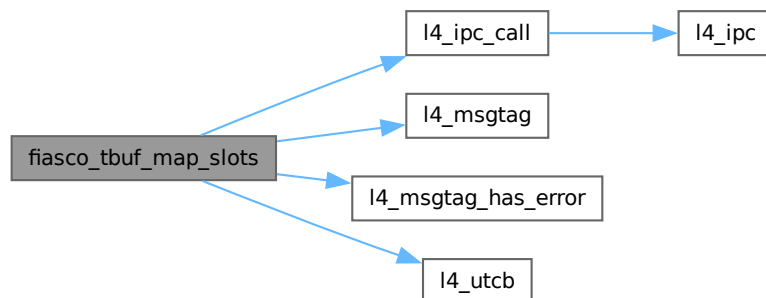
Returns

IPC message tag.

Definition at line 85 of file [__ktrace-impl.h](#).

References [L4_BASE_DEBUGGER_CAP](#), [l4_ipc_call\(\)](#), [L4_IPC_NEVER](#), [l4_msgtag\(\)](#), [l4_msgtag_has_error\(\)](#), [L4_PROTO_DEBUGGER](#), [l4_utcb\(\)](#), [l4_msg_regs_t::mr](#), and [l4_fpage_t::raw](#).

Here is the call graph for this function:



13.1.3.5.2.5 fiasco_tbuf_map_status()

```
l4_msgtag_t fiasco_tbuf_map_status (
    l4_fpage_t * ku_mem) [inline]
```

Map kernel trace-buffer status page.

Parameters

| | |
|---------------|--|
| <i>ku_mem</i> | Flexpage where to map the trace-buffer status page. Expected to be exactly one page in size. |
|---------------|--|

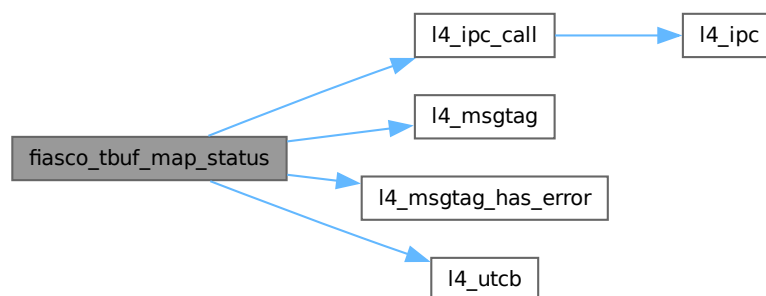
Returns

IPC message tag.

Definition at line 66 of file `__ktrace-impl.h`.

References [L4_BASE_DEBUGGER_CAP](#), [l4_ipc_call\(\)](#), [L4_IPC_NEVER](#), [l4_msgtag\(\)](#), [l4_msgtag_has_error\(\)](#), [L4_PROTO_DEBUGGER](#), [l4_utcb\(\)](#), [l4_msg_regs_t::mr](#), and [l4_fpage_t::raw](#).

Here is the call graph for this function:



13.1.3.5.2.6 fiasco_tbuf_validate()

```
l4_msgtag_t fiasco_tbuf_validate (
    void ) [inline]
```

Validate the kernel base debugger capability.

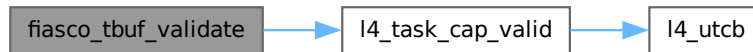
Return values

| | |
|--------------|--|
| <i>true</i> | The base debugger capability is valid and accessible. |
| <i>false</i> | The base debugger capability is not valid or not accessible. |

Definition at line 25 of file [__ktrace-impl.h](#).

References [L4_BASE_DEBUGGER_CAP](#), [L4_BASE_TASK_CAP](#), and [l4_task_cap_valid\(\)](#).

Here is the call graph for this function:



13.1.4 Flexpages

Flexpage-related API.

Collaboration diagram for Flexpages:



Data Structures

- union [l4_fpage_t](#)
L4 flexpage type.

Enumerations

- enum [L4_fpage_consts](#) {
[L4_FPAGE_RIGHTS_SHIFT](#) = 0 , [L4_FPAGE_TYPE_SHIFT](#) = 4 , [L4_FPAGE_SIZE_SHIFT](#) = 6 ,
[L4_FPAGE_ADDR_SHIFT](#) = 12 ,
[L4_FPAGE_RIGHTS_BITS](#) = 4 , [L4_FPAGE_TYPE_BITS](#) = 2 , [L4_FPAGE_SIZE_BITS](#) = 6 , [L4_FPAGE_ADDR_BITS](#)
= [L4_MWORD_BITS](#) - [L4_FPAGE_ADDR_SHIFT](#) ,
[L4_FPAGE_RIGHTS_MASK](#) , [L4_FPAGE_TYPE_MASK](#) , [L4_FPAGE_SIZE_MASK](#) , [L4_FPAGE_ADDR_](#)
[_MASK](#) = ~0UL << [L4_FPAGE_ADDR_SHIFT](#) ,
[L4_FPAGE_RIGHTS_ALL](#) = [L4_FPAGE_RIGHTS_MASK](#) }
L4 flexpage structure.
- enum { [L4_WHOLE_ADDRESS_SPACE](#) = 63 }
Constants for flexpages.
- enum [L4_fpage_rights](#) {
[L4_FPAGE_X](#) = 1 , [L4_FPAGE_W](#) = 2 , [L4_FPAGE_RO](#) = 4 , [L4_FPAGE_RW](#) = [L4_FPAGE_RO](#) | [L4_](#)
[FPAGE_W](#) ,
[L4_FPAGE_RX](#) = [L4_FPAGE_RO](#) | [L4_FPAGE_X](#) , [L4_FPAGE_RWX](#) = [L4_FPAGE_RW](#) | [L4_FPAGE_X](#) }

Memory and IO port flexpage rights.

- enum `L4_cap_fpage_rights` {
`L4_CAP_FPAGE_W` = 0x1 , `L4_CAP_FPAGE_S` = 0x2 , `L4_CAP_FPAGE_R` = 0x4 , `L4_CAP_FPAGE_RO` = 0x4 ,
`L4_CAP_FPAGE_D` = 0x8 , `L4_CAP_FPAGE_RW` = `L4_CAP_FPAGE_R` | `L4_CAP_FPAGE_W` ,
`L4_CAP_FPAGE_RS` = `L4_CAP_FPAGE_R` | `L4_CAP_FPAGE_S` , `L4_CAP_FPAGE_RWS` = `L4_CAP_FPAGE_R` | `L4_CAP_FPAGE_W` | `L4_CAP_FPAGE_S` ,
`L4_CAP_FPAGE_RWSD` = `L4_CAP_FPAGE_RWS` | `L4_CAP_FPAGE_D` , `L4_CAP_FPAGE_RWD` = `L4_CAP_FPAGE_R` | `L4_CAP_FPAGE_W` | `L4_CAP_FPAGE_D` , `L4_CAP_FPAGE_RSD` = `L4_CAP_FPAGE_RS` | `L4_CAP_FPAGE_D` }

Object flexpage rights.

- enum `L4_fpage_type` { `L4_FPAGE_SPECIAL` = 0 , `L4_FPAGE_MEMORY` = 1 , `L4_FPAGE_IO` = 2 , `L4_FPAGE_OBJ` = 3 }

Flexpage type.

- enum `L4_fpage_control` { `L4_FPAGE_CONTROL_OFFSET_SHIFT` = 12 , `L4_FPAGE_CONTROL_MASK` = $\sim 0UL \ll L4_FPAGE_CONTROL_OFFSET_SHIFT$ }

Flexpage map control flags.

- enum { `L4_WHOLE_IOADDRESS_SPACE` = 16 , `L4_IOPORT_MAX` = $(1L \ll L4_WHOLE_IOADDRESS_SPACE)$ }

Special constants for IO flexpages.

Functions

- `L4_CONSTEXPR l4_fpage_t l4_fpage (l4_addr_t address, unsigned int order, unsigned char rights) L4_NOTHROW`

Create a memory flexpage.

- `L4_CONSTEXPR l4_fpage_t l4_fpage_all (void) L4_NOTHROW`

Get a flexpage, describing all address spaces at once.

- `L4_CONSTEXPR l4_fpage_t l4_fpage_invalid (void) L4_NOTHROW`

Get an invalid flexpage.

- `L4_CONSTEXPR l4_fpage_t l4_iofpage (unsigned long port, unsigned int order) L4_NOTHROW`

Create an IO-port flexpage.

- `L4_CONSTEXPR l4_fpage_t l4_obj_fpage (l4_cap_idx_t obj, unsigned int order, unsigned char rights) L4_NOTHROW`

Create a kernel-object flexpage.

- `L4_CONSTEXPR int l4_is_fpage_writable (l4_fpage_t fp) L4_NOTHROW`

Test if the flexpage is writable.

- `L4_CONSTEXPR unsigned l4_fpage_rights (l4_fpage_t f) L4_NOTHROW`

Return rights from a flexpage.

- `L4_CONSTEXPR unsigned l4_fpage_type (l4_fpage_t f) L4_NOTHROW`

Return type from a flexpage.

- `L4_CONSTEXPR unsigned l4_fpage_size (l4_fpage_t f) L4_NOTHROW`

Return size (log2) from a flexpage.

- `L4_CONSTEXPR unsigned long l4_fpage_page (l4_fpage_t f) L4_NOTHROW`

Return the page part from a flexpage.

- `L4_CONSTEXPR l4_addr_t l4_fpage_memaddr (l4_fpage_t f) L4_NOTHROW`

Return the memory address from the memory flexpage.

- `L4_CONSTEXPR l4_cap_idx_t l4_fpage_obj (l4_fpage_t f) L4_NOTHROW`

Return the capability index from the object flexpage.

- `L4_CONSTEXPR unsigned long l4_fpage_ioport (l4_fpage_t f) L4_NOTHROW`

Return the IO port number from the IO flexpage.

- `L4_CONSTEXPR l4_fpage_t l4_fpage_set_rights (l4_fpage_t src, unsigned char new_rights) L4_NOTHROW`

Set new right in a flexpage.

- `L4_CONSTEXPR int l4_fpage_contains (l4_fpage_t fpage, l4_addr_t addr, unsigned order) L4_NOTHROW`

Test whether a given range is completely within an fpage.

- `L4_CONSTEXPR unsigned char l4_fpage_max_order (unsigned char order, l4_addr_t addr, l4_addr_t min↔_addr, l4_addr_t max_addr, l4_addr_t hotspot=0)`

Determine maximum flexpage size of a region.

- `L4_CONSTEXPR int l4_is_fpage_valid (l4_fpage_t fp) L4_NOTHROW`

Test if the flexpage is valid.

13.1.4.1 Detailed Description

Flexpage-related API.

A flexpage is a page with a variable size, that can describe memory, IO-Ports (IA32 only), and sets of kernel objects.

A flexpage describes an always size aligned region of an address space. The size is given in a log2 scale. This means the size in elements (bytes for memory, ports for IO-Ports, and capabilities for kernel objects) is always a power of two.

A flexpage also carries type and access right information for the described region. The type information selects the address space in which the flexpage is valid. Access rights have a meaning depending on the specific address space (type).

There exists a special type for defining *receive windows* or for the `l4_task_unmap()` method, that can be used to describe all address spaces (all types) with a single flexpage.

Include File

```
#include <l4/sys/types.h>
```

13.1.4.2 Enumeration Type Documentation

13.1.4.2.1 anonymous enum

anonymous enum

Constants for flexpages.

Enumerator

| | |
|------------------------|---|
| L4_WHOLE_ADDRESS_SPACE | Whole address space size. This value does not only specify the log2 size of the biggest possible memory flexpage. It can be also used as size for a special flexpage to define a flexpage which completely covers all spaces. |
|------------------------|---|

Definition at line 84 of file `__l4_fpage.h`.

13.1.4.2.2 anonymous enum

`anonymous enum`

Special constants for IO flexpages.

Enumerator

| | |
|--------------------------|--|
| L4_WHOLE_IOADDRESS_SPACE | Whole I/O address space size. In contrast to L4_WHOLE_ADDRESS_SPACE , this value forms the log2 size of the biggest possible I/O flexpage. |
| L4_IOPORT_MAX | Maximum I/O port address plus 1. |

Definition at line 326 of file [__l4_fpage.h](#).

13.1.4.2.3 L4_cap_fpage_rights

`enum L4_cap_fpage_rights`

Object flexpage rights.

Capabilities are modified or transferred with map and unmap operations. For that, capabilities are wrapped into flexpage objects. The flexpage carries a set of rights the sender wants to hand over to the receiver along with the capability.

For the user only the 'S' and the 'W' right are visible. Other rights such as the 'D' right are internal to the corresponding kernel object and cannot be evaluated by the receiver.

Note that additional object attributes and permissions can be specified in a send item, see [Attributes and additional permissions for objects](#).

Note

A thread can also map a capability from its task's capability table with a reduced set of rights into another slot of its own capability table.

Enumerator

| | |
|-----------------|--|
| L4_CAP_FPAGE_W | Interface specific 'W' right for capability flexpages. The semantics of the 'W' right is defined by the protocol. For example in case of a dataspace cap, the 'W' right is needed to get a writable dataspace. |
| L4_CAP_FPAGE_S | Interface specific 'S' right for capability flexpages. The semantics of the 'S' right is defined by the interface. When transferring object capabilities via IPC, the kernel masks this right with the 'S' right of the capability used to address the IPC partner. Thus, the 'S' right of sent capabilities is only transferred if both the flexpage and the IPC gate or thread capability specifying the IPC partner have the 'S' right. For L4::Task::map() , the 'S' right is only transferred if the flexpage, the source and destination task capabilities have the 'S' right. |
| L4_CAP_FPAGE_R | Read right for capability flexpages. This is always required, otherwise no capability is mapped. |
| L4_CAP_FPAGE_RO | Read right for capability flexpages. This is always required, otherwise no capability is mapped. |

| | |
|-------------------|--|
| L4_CAP_FPAGE_D | Delete right for capability flexpages. This allows the receiver to delete the corresponding kernel object using <code>unmap()</code> regardless of other tasks still holding a capability to the kernel object. Such capabilities are set to an empty capability if the object is deleted. |
| L4_CAP_FPAGE_RW | Read and interface specific 'W' right for capability flexpages. The semantics of the 'W' right is defined by the interface. See also L4_CAP_FPAGE_W |
| L4_CAP_FPAGE_RS | Read and interface specific 'S' right for capability flexpages. The semantics of the 'S' right is defined by the interface. See also L4_CAP_FPAGE_S |
| L4_CAP_FPAGE_RWS | Read, interface specific 'W', and 'S' rights for capability flexpages. The semantics of the 'W' and 'S' right are defined by the interface. See also L4_CAP_FPAGE_R , L4_CAP_FPAGE_W , and L4_CAP_FPAGE_S |
| L4_CAP_FPAGE_RWSD | Full rights for capability flexpages. See also L4_CAP_FPAGE_R , L4_CAP_FPAGE_W , L4_CAP_FPAGE_S , and L4_CAP_FPAGE_D |
| L4_CAP_FPAGE_RWD | Read, write, and delete right for capability flexpages. See also L4_CAP_FPAGE_R , L4_CAP_FPAGE_W , and L4_CAP_FPAGE_D |
| L4_CAP_FPAGE_RSD | Read, 'S', and delete right for capability flexpages. See also L4_CAP_FPAGE_R , L4_CAP_FPAGE_S , and L4_CAP_FPAGE_D |

Definition at line 148 of file [__l4_fpage.h](#).

13.1.4.2.4 L4_fpage_consts

enum [L4_fpage_consts](#)

[L4](#) flexpage structure.

Enumerator

| | |
|-----------------------|--|
| L4_FPAGE_RIGHTS_SHIFT | Access permissions shift. |
| L4_FPAGE_TYPE_SHIFT | Flexpage type shift (memory, IO port, obj...). |

| | |
|----------------------|---|
| L4_FPAGE_SIZE_SHIFT | Flexpage size shift (log2-based). |
| L4_FPAGE_ADDR_SHIFT | Page address shift. |
| L4_FPAGE_RIGHTS_BITS | Access permissions size. |
| L4_FPAGE_TYPE_BITS | Flexpage type size (memory, IO port, obj...). |
| L4_FPAGE_SIZE_BITS | Flexpage size size (log2-based). |
| L4_FPAGE_ADDR_BITS | Page address size. |
| L4_FPAGE_RIGHTS_MASK | Mask to get the flexpage rights. |
| L4_FPAGE_RIGHTS_ALL | Specify as flexpage rights during grant. |

Definition at line 48 of file [__l4_fpage.h](#).

13.1.4.2.5 L4_fpage_control

```
enum L4_fpage_control
```

Flexpage map control flags.

Enumerator

| | |
|-------------------------------|---|
| L4_FPAGE_CONTROL_OFFSET_SHIFT | Number of bits an index must be shifted or an address must be aligned to in the control word. |
| L4_FPAGE_CONTROL_MASK | Mask for truncating the lower bits of the send base or the index of the control word. |

Definition at line 243 of file [__l4_fpage.h](#).

13.1.4.2.6 L4_fpage_rights

```
enum L4_fpage_rights
```

Memory and IO port flexpage rights.

For IO flexpages, bit 1 and bit 2 are a combined read/write right. In a map operation, the receiver receives the IO port capability when the sender possesses it and at least one of these bits is present. For an unmap operation, the absence of one of those bits is sufficient to unmap the IO port capability.

Note that more memory attributes can be specified in a send item, see [l4_fpage_cacheability_opt_t](#).

Enumerator

| | |
|-------------|----------------------|
| L4_FPAGE_X | Executable flexpage. |
| L4_FPAGE_W | Writable flexpage. |
| L4_FPAGE_RO | Read-only flexpage. |
| L4_FPAGE_RW | Read-write flexpage. |

| | |
|--------------|------------------------------|
| L4_FPAGE_RX | Read-execute flexpage. |
| L4_FPAGE_RWX | Read-write-execute flexpage. |

Definition at line 118 of file [__l4_fpage.h](#).

13.1.4.2.7 L4_fpage_type

enum [L4_fpage_type](#)

Flexpage type.

Enumerator

| | |
|------------------|--|
| L4_FPAGE_SPECIAL | Special flexpage, either l4_fpage_invalid() or l4_fpage_all() ; only supported by selected interfaces. |
| L4_FPAGE_MEMORY | Flexpage for memory spaces. |
| L4_FPAGE_IO | Flexpage for I/O port spaces. |
| L4_FPAGE_OBJ | Flexpage for object spaces. |

Definition at line 230 of file [__l4_fpage.h](#).

13.1.4.3 Function Documentation

13.1.4.3.1 l4_fpage()

```
L4_CONSTEXPR l4_fpage_t l4_fpage (
    l4_addr_t address,
    unsigned int order,
    unsigned char rights) [inline]
```

Create a memory flexpage.

Parameters

| | |
|----------------|--|
| <i>address</i> | Flexpage start address |
| <i>order</i> | Flexpage size (log2), L4_WHOLE_ADDRESS_SPACE to specify the whole address space (with address 0). The minimum log2 size of a memory flexpage is defined by L4_LOG2_PAGESIZE according to the size of the smallest virtual page supported by the MMU. |
| <i>rights</i> | Access rights, see L4_fpage_rights |

Returns

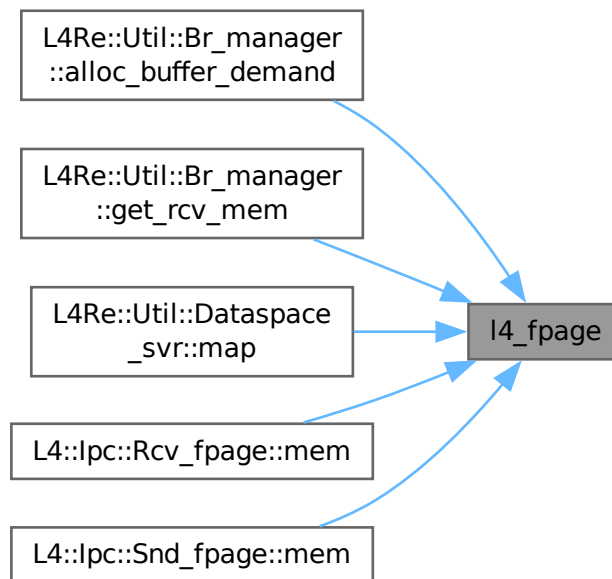
Memory flexpage

Definition at line 719 of file [__l4_fpage.h](#).

References [L4_FPAGE_MEMORY](#), and [L4_NOTHROW](#).

Referenced by [L4Re::Util::Br_manager::alloc_buffer_demand\(\)](#), [L4Re::Util::Br_manager::get_rcv_mem\(\)](#), [L4Re::Util::Dataspace_svr::map\(\)](#), [L4::lpc::Rcv_fpage::mem\(\)](#), and [L4::lpc::Snd_fpage::mem\(\)](#).

Here is the caller graph for this function:

**13.1.4.3.2 l4_fpage_all()**

```
L4_CONSTEXPR l4_fpage_t l4_fpage_all (
    void ) [inline]
```

Get a flexpage, describing all address spaces at once.

Returns

Special *all-spaces* flexpage.

Note

This flexpage can be used to define a receive window where the sender can send objects of any type, or for an unmap item completely covering all spaces of the target task. It does not make sense to use this flexpage as send item.

Definition at line 739 of file [__l4_fpage.h](#).

References [L4_FPAGE_SPECIAL](#), [L4_NOTHROW](#), and [L4_WHOLE_ADDRESS_SPACE](#).

13.1.4.3.3 l4_fpage_contains()

```
L4_CONSTEXPR int l4_fpage_contains (
    l4_fpage_t fpage,
    l4_addr_t addr,
    unsigned order) [inline]
```

Test whether a given range is completely within an fpage.

Parameters

| | |
|--------------|------------------------|
| <i>fpage</i> | Flexpage |
| <i>addr</i> | Address |
| <i>order</i> | Size of range in log2. |

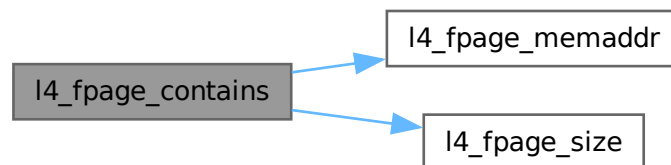
Return values

| | |
|------------------|---|
| <code>==0</code> | The range is not completely in the fpage. |
| <code>!=0</code> | The range is within the fpage. |

Definition at line 771 of file [__l4_fpage.h](#).

References [l4_fpage_memaddr\(\)](#), [l4_fpage_size\(\)](#), and [L4_NOTHROW](#).

Here is the call graph for this function:



13.1.4.3.4 l4_fpage_invalid()

```
L4_CONSTEXPR l4_fpage_t l4_fpage_invalid (
    void ) [inline]
```

Get an invalid flexpage.

Returns

Special *invalid* flexpage.

Definition at line 745 of file [__l4_fpage.h](#).

References [L4_FPAGE_SPECIAL](#), and [L4_NOTHROW](#).

13.1.4.3.5 l4_fpage_ioport()

```
L4_CONSTEXPR unsigned long l4_fpage_ioport (
    l4_fpage_t f) [inline]
```

Return the IO port number from the IO flexpage.

Parameters

| | |
|----------|----------|
| <i>f</i> | Flexpage |
|----------|----------|

Returns

IO port number from the given IO flexpage.

Precondition

f must be an IO flexpage (`l4_fpage_type(f) == L4_FPAGE_IO`) and

The function does not enforce size alignment of the read memory address. The caller must ensure the input fpage is correct.

Definition at line 671 of file `__l4_fpage.h`.

References `L4_FPAGE_ADDR_SHIFT`, and `L4_NOTHROW`.

13.1.4.3.6 l4_fpage_max_order()

```
L4_CONSTEXPR unsigned char l4_fpage_max_order (
    unsigned char order,
    l4_addr_t addr,
    l4_addr_t min_addr,
    l4_addr_t max_addr,
    l4_addr_t hotspot = 0) [inline]
```

Determine maximum flexpage size of a region.

Parameters

| | |
|-----------------|---|
| <i>order</i> | Order value to start with (e.g. for memory <code>L4_LOG2_PAGESIZE</code> would be used) |
| <i>addr</i> | Address to be covered by the flexpage. |
| <i>min_addr</i> | Start of region / minimal address (including). |
| <i>max_addr</i> | End of region / maximal address (excluding). |
| <i>hotspot</i> | (Optional) hot spot. |

Returns

Maximum order (log2-size) possible.

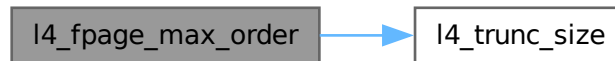
Note

The start address of the flexpage can be determined with `l4_trunc_size(addr, returnvalue)`

Definition at line 779 of file `__l4_fpage.h`.

References `l4_trunc_size()`.

Here is the call graph for this function:

**13.1.4.3.7 l4_fpage_memaddr()**

```

L4_CONSTEXPR l4_addr_t l4_fpage_memaddr (
    l4_fpage_t f) [inline]
  
```

Return the memory address from the memory flexpage.

Parameters

| | |
|----------|----------|
| <i>f</i> | Flexpage |
|----------|----------|

Returns

Page address from the given memory flexpage.

Precondition

f must be a memory flexpage (`l4_fpage_type(f) == L4_FPAGE_MEMORY`).

The function does not enforce size alignment of the read memory address. The caller must ensure the input *fpage* is correct.

Definition at line 677 of file `__l4_fpage.h`.

References `L4_NOTHROW`.

Referenced by `l4_fpage_contains()`.

Here is the caller graph for this function:



13.1.4.3.8 l4_fpage_obj()

```
L4_CONSTEXPR l4_cap_idx_t l4_fpage_obj (
    l4_fpage_t f) [inline]
```

Return the capability index from the object flexpage.

Parameters

| | |
|----------|----------|
| <i>f</i> | Flexpage |
|----------|----------|

Returns

Capability index from the given object flexpage.

Precondition

f must be an object flexpage (`l4_fpage_type(f) == L4_FPAGE_OBJ`)

The function does not enforce size alignment of the read memory address. The caller must ensure the input fpage is correct.

Definition at line 683 of file [__l4_fpage.h](#).

References [L4_NOTHROW](#).

13.1.4.3.9 l4_fpage_page()

```
L4_CONSTEXPR unsigned long l4_fpage_page (
    l4_fpage_t f) [inline]
```

Return the page part from a flexpage.

Parameters

| | |
|----------|----------|
| <i>f</i> | Flexpage |
|----------|----------|

Returns

Page part of the given flexpage.

Note

The meaning of the page part depends on the flexpage type.

Definition at line 665 of file [__l4_fpage.h](#).

References [L4_FPAGE_ADDR_SHIFT](#), and [L4_NOTHROW](#).

13.1.4.3.10 l4_fpage_rights()

```
L4_CONSTEXPR unsigned l4_fpage_rights (  
    l4_fpage_t f) [inline]
```

Return rights from a flexpage.

Parameters

| | |
|----------|----------|
| <i>f</i> | Flexpage |
|----------|----------|

Returns

Size part of the given flexpage.

Definition at line 647 of file [__l4_fpage.h](#).

References [L4_FPAGE_RIGHTS_MASK](#), [L4_FPAGE_RIGHTS_SHIFT](#), and [L4_NOTHROW](#).

Referenced by [l4_is_fpage_writable\(\)](#).

Here is the caller graph for this function:



13.1.4.3.11 l4_fpage_set_rights()

```
L4_CONSTEXPR l4_fpage_t l4_fpage_set_rights (  
    l4_fpage_t src,  
    unsigned char new_rights) [inline]
```

Set new right in a flexpage.

Parameters

| | |
|-------------------|------------|
| <i>src</i> | Flexpage |
| <i>new_rights</i> | New rights |

Returns

Modified flexpage with new rights.

Definition at line 708 of file [__l4_fpage.h](#).

References [L4_FPAGE_RIGHTS_MASK](#), [L4_FPAGE_RIGHTS_SHIFT](#), and [L4_NOTHROW](#).

Referenced by [L4::lpc::Snd_fpage::io\(\)](#).

Here is the caller graph for this function:

**13.1.4.3.12 l4_fpage_size()**

```
L4_CONSTEXPR unsigned l4_fpage_size (
    l4_fpage_t f) [inline]
```

Return size (log2) from a flexpage.

Parameters

| | |
|----------|----------|
| <i>f</i> | Flexpage |
|----------|----------|

Returns

Size part of the given flexpage.

See also

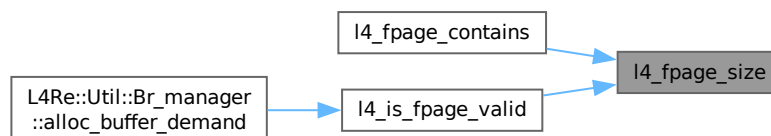
[l4_fpage_memaddr\(\)](#), [l4_fpage_obj\(\)](#), [l4_fpage_ioport\(\)](#)

Definition at line 659 of file [__l4_fpage.h](#).

References [L4_FPAGE_SIZE_SHIFT](#), and [L4_NOTHROW](#).

Referenced by [l4_fpage_contains\(\)](#), and [l4_is_fpage_valid\(\)](#).

Here is the caller graph for this function:



13.1.4.3.13 l4_fpage_type()

```
L4_CONSTEXPR unsigned l4_fpage_type (
    l4_fpage_t f) [inline]
```

Return type from a flexpage.

Parameters

| | |
|----------|----------|
| <i>f</i> | Flexpage |
|----------|----------|

Returns

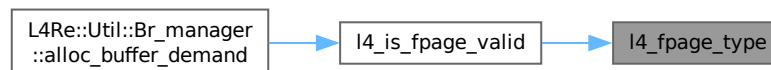
Type part of the given flexpage.

Definition at line 653 of file `__l4_fpage.h`.

References [L4_FPAGE_TYPE_SHIFT](#), and [L4_NOTHROW](#).

Referenced by [l4_is_fpage_valid\(\)](#).

Here is the caller graph for this function:



13.1.4.3.14 l4_iofpage()

```
L4_CONSTEXPR l4_fpage_t l4_iofpage (
    unsigned long port,
    unsigned int order) [inline]
```

Create an IO-port flexpage.

Parameters

| | |
|--------------|---|
| <i>port</i> | I/O-flexpage port base |
| <i>order</i> | I/O-flexpage size (log2), L4_WHOLE_IOADDRESS_SPACE to specify the whole I/O address space (with port 0) |

Returns

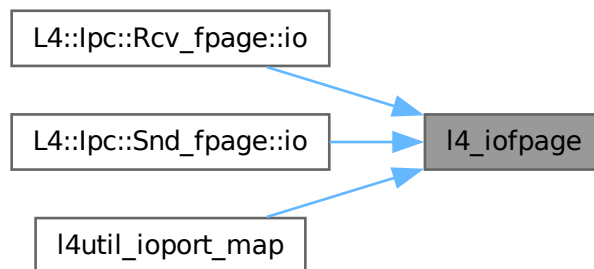
I/O flexpage

Definition at line 725 of file [__l4_fpage.h](#).

References [L4_FPAGE_ADDR_SHIFT](#), [L4_FPAGE_IO](#), [L4_FPAGE_RW](#), and [L4_NOTHROW](#).

Referenced by [L4::lpc::Rcv_fpage::io\(\)](#), [L4::lpc::Snd_fpage::io\(\)](#), and [l4util_ioport_map\(\)](#).

Here is the caller graph for this function:

**13.1.4.3.15 l4_is_fpage_valid()**

```
L4_CONSTEXPR int l4_is_fpage_valid (
    l4_fpage_t fp) [inline]
```

Test if the flexpage is valid.

Parameters

| | |
|-----------|-----------|
| <i>fp</i> | Flexpage. |
|-----------|-----------|

Return values

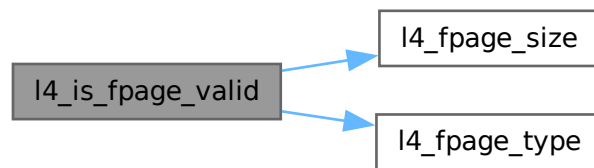
| | |
|------------------|---------------------------|
| <code>!=0</code> | if flexpage is valid. |
| <code>==0</code> | if flexpage is not valid. |

Definition at line 803 of file [__l4_fpage.h](#).

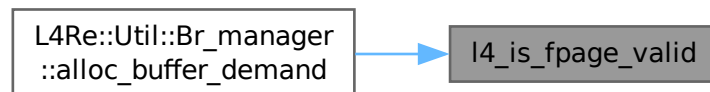
References [l4_fpage_size\(\)](#), [L4_FPAGE_SPECIAL](#), [l4_fpage_type\(\)](#), and [L4_NOTHROW](#).

Referenced by [L4Re::Util::Br_manager::alloc_buffer_demand\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.1.4.3.16 l4_is_fpage_writable()

```
L4_CONSTEXPR int l4_is_fpage_writable (
    l4_fpage_t fp) [inline]
```

Test if the flexpage is writable.

Parameters

| | |
|-----------|-----------|
| <i>fp</i> | Flexpage. |
|-----------|-----------|

Return values

| | |
|------------|------------------------------|
| <i>!=0</i> | if flexpage is writable. |
| <i>==0</i> | if flexpage is not writable. |

Definition at line 752 of file [__l4_fpage.h](#).

References [l4_fpage_rights\(\)](#), [L4_FPAGE_W](#), and [L4_NOTHROW](#).

Here is the call graph for this function:



13.1.4.3.17 l4_obj_fpage()

```

L4_CONSTEXPR l4_fpage_t l4_obj_fpage (
    l4_cap_idx_t obj,
    unsigned int order,
    unsigned char rights) [inline]
  
```

Create a kernel-object flexpage.

Parameters

| | |
|---------------|--|
| <i>obj</i> | Base capability selector. |
| <i>order</i> | Log2 size (number of capabilities). |
| <i>rights</i> | Access rights, see L4_cap_fpage_rights |

Returns

Flexpage for a set of kernel objects.

Note

[L4_CAP_FPAGE_R](#) is always required, otherwise no capability is mapped.

Examples

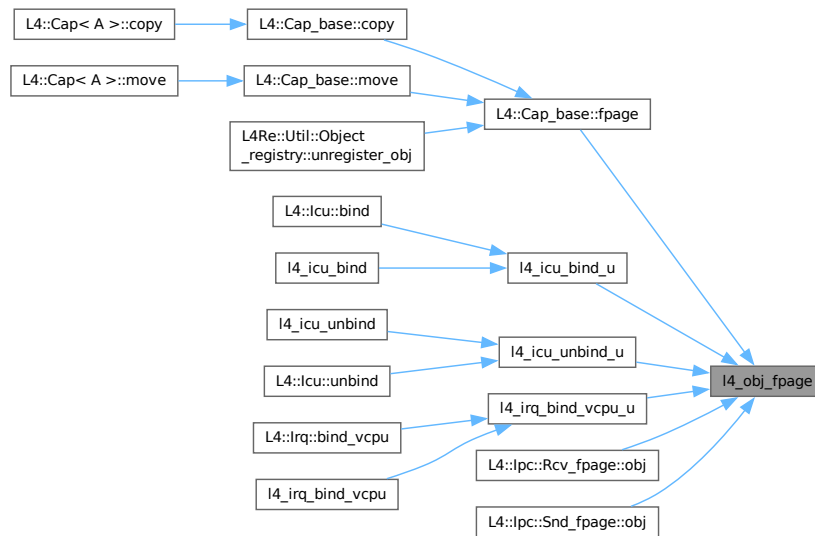
[examples/sys/utcb-ipc/main.c](#).

Definition at line 731 of file [__l4_fpage.h](#).

References [L4_CAP_SHIFT](#), [L4_FPAGE_ADDR_SHIFT](#), [L4_FPAGE_OBJ](#), and [L4_NOTHROW](#).

Referenced by [L4::Cap_base::fpage\(\)](#), [l4_icu_bind_u\(\)](#), [l4_icu_unbind_u\(\)](#), [l4_irq_bind_vcpu_u\(\)](#), [L4::lpc::Rcv_fpage::obj\(\)](#), and [L4::lpc::Snd_fpage::obj\(\)](#).

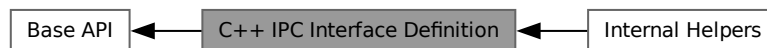
Here is the caller graph for this function:



13.1.5 C++ IPC Interface Definition

APIs for defining IPC interfaces using C++ as language.

Collaboration diagram for C++ IPC Interface Definition:



Topics

- Internal•Helpers 194

Namespaces

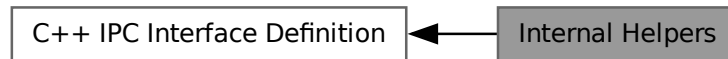
- namespace [L4::Typeid](#)
Definition of interface data-type helpers.

13.1.5.1 Detailed Description

APIs for defining IPC interfaces using C++ as language.

13.1.5.2 Internal Helpers

Collaboration diagram for Internal Helpers:



Data Structures

- struct `L4::Types::Bool< V >`
Boolean meta type.
- struct `L4::Types::False`
False meta value.
- struct `L4::Types::True`
True meta value.
- struct `L4::Types::Same< A, B >`
Compare two data types for equality.

13.1.5.2.1 Detailed Description

13.1.6 Cache Consistency

Various functions for cache consistency.

Collaboration diagram for Cache Consistency:



Functions

- `L4_BEGIN_DECLS` `int l4_cache_clean_data` (unsigned long start, unsigned long end) `L4_NOTHROW`
Cache clean a range in D-cache; writes back to PoC.
- `int l4_cache_flush_data` (unsigned long start, unsigned long end) `L4_NOTHROW`
Cache flush a range; writes back to PoC.
- `int l4_cache_inv_data` (unsigned long start, unsigned long end) `L4_NOTHROW`
Cache invalidate a range; might write back to PoC.
- `int l4_cache_coherent` (unsigned long start, unsigned long end) `L4_NOTHROW`
Make memory coherent between I-cache and D-cache; writes back to PoU.
- `int l4_cache_dma_coherent` (unsigned long start, unsigned long end) `L4_NOTHROW`
Make memory coherent for use with external memory; writes back to PoC.
- `int l4_cache_dma_coherent_full` (void) `L4_NOTHROW`
Make memory coherent for use with external memory; writes back to PoC.

13.1.6.1 Detailed Description

Various functions for cache consistency.

These functions shall be used to ensure that

- all blocks (e.g. CPU cores, devices, DMA engines) are guaranteed to see the same copy of a memory location (Point of Coherency – PoC),
- instruction and data caches of a core are guaranteed to see the same copy of a memory location (Point of Unification – PoU).

Certain functions are NOPs on certain architectures, for example on Intel it's not necessary to explicitly make caches coherent to PoU.

13.1.6.2 Function Documentation

13.1.6.2.1 `l4_cache_clean_data()`

```
L4_BEGIN_DECLS
int l4_cache_clean_data (
    unsigned long start,
    unsigned long end) [inline]
```

Cache clean a range in D-cache; writes back to PoC.

Parameters

| | |
|--------------|----------------------------|
| <i>start</i> | Start of range (inclusive) |
| <i>end</i> | End of range (exclusive) |

Return values

| | |
|----------------|---|
| <i>0</i> | on success |
| <i>-EFAULT</i> | in the case of an unresolved page fault in the given area |

Writes back any dirty cache lines in the range but leaves them in the cache and marks the cached copies clean.

Examples

[examples/libs/l4re/c++/shared_ds/ds_clnt.cc](#).

Definition at line 68 of file [cache.h](#).

References [L4_NOTHROW](#).

13.1.6.2.2 l4_cache_coherent()

```
int l4_cache_coherent (
    unsigned long start,
    unsigned long end) [inline]
```

Make memory coherent between I-cache and D-cache; writes back to PoU.

Parameters

| | |
|--------------|----------------------------|
| <i>start</i> | Start of range (inclusive) |
| <i>end</i> | End of range (exclusive) |

Return values

| | |
|----------------|---|
| <i>0</i> | on success |
| <i>-EFAULT</i> | in the case of an unresolved page fault in the given area |

Definition at line 92 of file [cache.h](#).

References [L4_NOTHROW](#).

13.1.6.2.3 l4_cache_dma_coherent()

```
int l4_cache_dma_coherent (
    unsigned long start,
    unsigned long end) [inline]
```

Make memory coherent for use with external memory; writes back to PoC.

Parameters

| | |
|--------------|----------------------------|
| <i>start</i> | Start of range (inclusive) |
| <i>end</i> | End of range (exclusive) |

Return values

| | |
|---------|---|
| 0 | on success |
| -EFAULT | in the case of an unresolved page fault in the given area |

Definition at line 100 of file [cache.h](#).

References [L4_NOTHROW](#).

13.1.6.2.4 l4_cache_flush_data()

```
int l4_cache_flush_data (
    unsigned long start,
    unsigned long end) [inline]
```

Cache flush a range; writes back to PoC.

Parameters

| | |
|--------------|----------------------------|
| <i>start</i> | Start of range (inclusive) |
| <i>end</i> | End of range (exclusive) |

Return values

| | |
|---------|---|
| 0 | on success |
| -EFAULT | in the case of an unresolved page fault in the given area |

Writes back any dirty cache lines and invalidates all cache entries in the range.

Definition at line 76 of file [cache.h](#).

References [L4_NOTHROW](#).

13.1.6.2.5 l4_cache_inv_data()

```
int l4_cache_inv_data (
    unsigned long start,
    unsigned long end) [inline]
```

Cache invalidate a range; might write back to PoC.

Parameters

| | |
|--------------|----------------------------|
| <i>start</i> | Start of range (inclusive) |
| <i>end</i> | End of range (exclusive) |

Return values

| | |
|---------|---|
| 0 | on success |
| -EFAULT | in the case of an unresolved page fault in the given area |

Invalidates all cache entries in the range but does not necessarily write back dirty cache lines.

Note

Implementations may choose to write back dirty lines nonetheless if this is more efficient.

Definition at line 84 of file [cache.h](#).

References [L4_NOTHROW](#).

13.1.7 Memory related

Memory related constants, data types and functions.

Collaboration diagram for Memory related:



Macros

- **#define L4_PAGESIZE**
Minimal page size (in bytes).
- **#define L4_PAGEMASK**
Mask for the page number.
- **#define L4_LOG2_PAGESIZE**
Number of bits used for page offset.
- **#define L4_SUPERPAGESIZE**
Size of a large page.
- **#define L4_SUPERPAGEMASK**
Mask for the number of a large page.

- `#define L4_LOG2_SUPERPAGESIZE`
Number of bits used as offset for a large page.
- `#define L4_INVALID_PTR ((void *)L4_INVALID_ADDR)`
Invalid address as pointer type.
- `#define L4_PAGESHIFT 12`
Size of a page, log2-based.
- `#define L4_SUPERPAGESHIFT 21`
Size of a large page, log2-based.
- `#define L4_PAGESHIFT 12`
Size of a page, log2-based.
- `#define L4_SUPERPAGESHIFT 21`
Size of a large page, log2-based.
- `#define L4_PAGESHIFT 12`
Size of a page, log2-based.
- `#define L4_SUPERPAGESHIFT 21`
Size of a large page, log2-based.
- `#define L4_PAGESHIFT 12`
Size of a page log2-based.
- `#define L4_SUPERPAGESHIFT 22`
Size of a large page log2-based.
- `#define L4_PAGESHIFT 12`
Size of a page in log2.
- `#define L4_SUPERPAGESHIFT 21`
Size of a large page in log2.

Enumerations

- `enum l4_addr_consts_t { L4_INVALID_ADDR = ~0UL }`
Address related constants.

Functions

- `l4_addr_t l4_trunc_page (l4_addr_t address) L4_NOTHROW`
Round an address down to the next lower page boundary.
- `l4_addr_t l4_trunc_size (l4_addr_t address, unsigned char bits) L4_NOTHROW`
Round an address down to the next lower flexpage with size bits.
- `l4_addr_t l4_round_page (l4_addr_t address) L4_NOTHROW`
Round address up to the next page.
- `l4_addr_t l4_round_size (l4_addr_t value, unsigned char bits) L4_NOTHROW`
Round value up to the next alignment with bits size.
- `unsigned l4_bytes_to_mwords (unsigned size) L4_NOTHROW`
Determine how many machine words (`l4_umword_t`) are required to store a buffer of 'size' bytes.

13.1.7.1 Detailed Description

Memory related constants, data types and functions.

13.1.7.2 Macro Definition Documentation

13.1.7.2.1 L4_LOG2_PAGESIZE

```
#define L4_LOG2_PAGESIZE
```

Number of bits used for page offset.

Size of page in log2.

Definition at line 428 of file [consts.h](#).

Referenced by [L4Re::Dataspace::map\(\)](#), [L4Re::Dataspace::map_region\(\)](#), and [L4Re::Util::Dataspace_svr::page_shift\(\)](#).

13.1.7.2.2 L4_LOG2_SUPERPAGESIZE

```
#define L4_LOG2_SUPERPAGESIZE
```

Number of bits used as offset for a large page.

Size of large page in log2

Definition at line 454 of file [consts.h](#).

13.1.7.2.3 L4_PAGEMASK

```
#define L4_PAGEMASK
```

Mask for the page number.

Note

The most significant bits are set.

Definition at line 419 of file [consts.h](#).

Referenced by [L4virtio::Driver::Device::driver_connect\(\)](#), [l4_round_page\(\)](#), and [l4_trunc_page\(\)](#).

13.1.7.2.4 L4_PAGESHIFT [1/4]

```
#define L4_PAGESHIFT 12
```

Size of a page, log2-based.

Sizeof a page in log2.

Size of a page log2-based.

Definition at line 24 of file [consts.h](#).

13.1.7.2.5 L4_PAGESHIFT [2/4]

```
#define L4_PAGESHIFT 12
```

Size of a page, log2-based.

Sizeof a page in log2.

Size of a page log2-based.

Examples

[examples/libs/l4re/c++/mem_alloc/ma+rm.cc](#), [examples/libs/l4re/c/ma+rm.c](#), [examples/libs/l4re/streammap/client.cc](#),
and [examples/libs/l4re/streammap/server.cc](#).

Definition at line 26 of file [consts.h](#).

Referenced by [L4Re::Rm::attach\(\)](#), [L4virtio::Driver::Device::driver_connect\(\)](#), [L4Re::Rm::free_area\(\)](#), [L4Re::Util::Dataspace_svr::map](#),
[L4Re::Rm::reserve_area\(\)](#), [L4Re::Rm::reserve_area\(\)](#), [L4virtio::Driver::Block_device::setup_device\(\)](#), and
[L4virtio::Driver::Virtio_net_device::setup_device\(\)](#).

13.1.7.2.6 L4_PAGESHIFT [3/4]

```
#define L4_PAGESHIFT 12
```

Size of a page, log2-based.

Sizeof a page in log2.

Size of a page log2-based.

Definition at line 26 of file [consts.h](#).

13.1.7.2.7 L4_PAGESHIFT [4/4]

```
#define L4_PAGESHIFT 12
```

Size of a page log2-based.

Sizeof a page in log2.

Definition at line 24 of file [consts.h](#).

13.1.7.2.8 L4_SUPERPAGEMASK

```
#define L4_SUPERPAGEMASK
```

Mask for the number of a large page.

Note

The most significant bits are set.

Definition at line 446 of file [consts.h](#).

13.1.7.2.9 L4_SUPERPAGESHIFT [1/4]

```
#define L4_SUPERPAGESHIFT 21
```

Size of a large page, log2-based.

Sizeof a large page in log2.

Size of a large page log2-based.

Definition at line 30 of file [consts.h](#).

13.1.7.2.10 L4_SUPERPAGESHIFT [2/4]

```
#define L4_SUPERPAGESHIFT 21
```

Size of a large page, log2-based.

Sizeof a large page in log2.

Size of a large page log2-based.

Examples

[examples/libs/l4re/c++/mem_alloc/ma+rm.cc](#), and [examples/libs/l4re/c/ma+rm.c](#).

Definition at line 31 of file [consts.h](#).

Referenced by [L4virtio::Svr::Driver_mem_region_t< No_custom_data >::Driver_mem_region_t\(\)](#).

13.1.7.2.11 L4_SUPERPAGESHIFT [3/4]

```
#define L4_SUPERPAGESHIFT 21
```

Size of a large page, log2-based.

Sizeof a large page in log2.

Size of a large page log2-based.

Definition at line 31 of file [consts.h](#).

13.1.7.2.12 L4_SUPERPAGESHIFT [4/4]

```
#define L4_SUPERPAGESHIFT 22
```

Size of a large page log2-based.

Sizeof a large page in log2.

Definition at line 30 of file [consts.h](#).

13.1.7.2.13 L4_SUPERPAGESIZE

```
#define L4_SUPERPAGESIZE
```

Size of a large page.

A large page is a *super page* on IA32 or a *section* on ARM.

Definition at line 437 of file [consts.h](#).

Referenced by [L4virtio::Driver::Device::driver_connect\(\)](#).

13.1.7.3 Enumeration Type Documentation

13.1.7.3.1 l4_addr_consts_t

```
enum l4_addr_consts_t
```

Address related constants.

Enumerator

| | |
|-----------------|------------------|
| L4_INVALID_ADDR | Invalid address. |
|-----------------|------------------|

Definition at line 522 of file [consts.h](#).

13.1.7.4 Function Documentation

13.1.7.4.1 l4_bytes_to_mwords()

```
unsigned l4_bytes_to_mwords (  
    unsigned size) [inline]
```

Determine how many machine words ([l4_umword_t](#)) are required to store a buffer of 'size' bytes.

Parameters

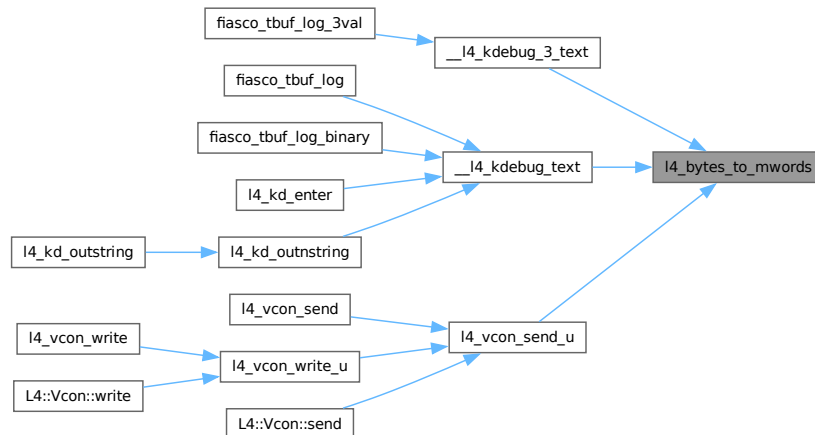
| | |
|------|--|
| size | The number of bytes to be translated into machine words. |
|------|--|

Definition at line 515 of file [consts.h](#).

References [L4_INLINE](#), and [L4_NOTHROW](#).

Referenced by [__l4_kdebug_3_text\(\)](#), [__l4_kdebug_text\(\)](#), and [l4_vcon_send_u\(\)](#).

Here is the caller graph for this function:



13.1.7.4.2 l4_round_page()

```
l4_addr_t l4_round_page (
    l4_addr_t address) [inline]
```

Round address up to the next page.

The address is rounded up to the next minimal page boundary. On most architectures this is a 4k page. Check [L4_PAGESIZE](#) for the minimal page size.

Parameters

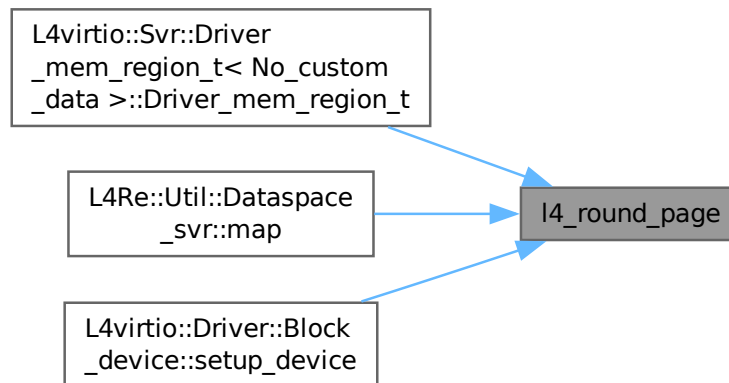
| | |
|----------------|--------------------------|
| <i>address</i> | The address to round up. |
|----------------|--------------------------|

Definition at line [492](#) of file [consts.h](#).

References [L4_INLINE](#), [L4_NOTHROW](#), [L4_PAGEMASK](#), and [L4_PAGESIZE](#).

Referenced by [L4virtio::Svr::Driver_mem_region_t< No_custom_data >::Driver_mem_region_t\(\)](#), [L4Re::Util::Dataspace_svr::map\(\)](#), and [L4virtio::Driver::Block_device::setup_device\(\)](#).

Here is the caller graph for this function:



13.1.7.4.3 l4_round_size()

```
l4_addr_t l4_round_size (
    l4_addr_t value,
    unsigned char bits) [inline]
```

Round value up to the next alignment with *bits* size.

Parameters

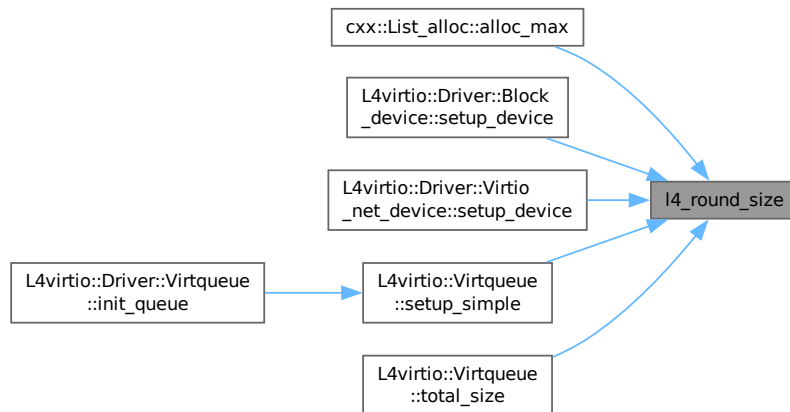
| | |
|--------------|---|
| <i>value</i> | The value to round up to the next size-alignment. |
| <i>bits</i> | The size of the alignment (log2). |

Definition at line 503 of file [consts.h](#).

References [L4_INLINE](#), and [L4_NOTHROW](#).

Referenced by [cxx::List_alloc::alloc_max\(\)](#), [L4virtio::Driver::Block_device::setup_device\(\)](#), [L4virtio::Driver::Virtio_net_device::setup_d](#), [L4virtio::Virtqueue::setup_simple\(\)](#), and [L4virtio::Virtqueue::total_size\(\)](#).

Here is the caller graph for this function:



13.1.7.4.4 l4_trunc_page()

```
l4_addr_t l4_trunc_page (
    l4_addr_t address) [inline]
```

Round an address down to the next lower page boundary.

The address is rounded down to the next lower minimal page boundary. On most architectures this is a 4k page. Check [L4_PAGESIZE](#) for the minimal page size.

Parameters

| | |
|----------------|-----------------------|
| <i>address</i> | The address to round. |
|----------------|-----------------------|

Examples

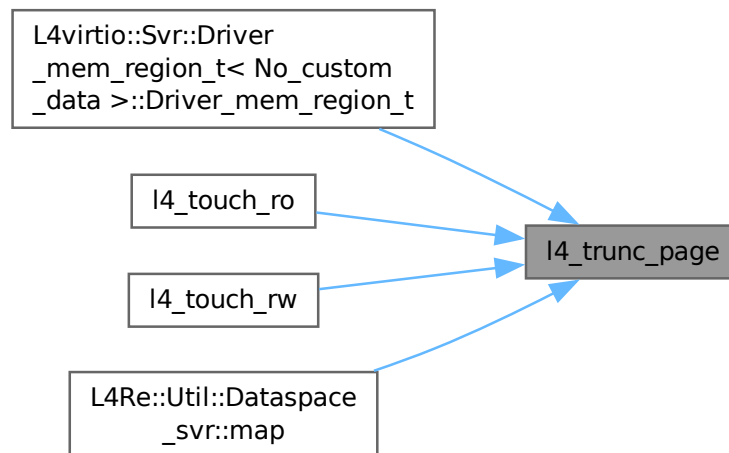
[examples/libs/l4re/c++/mem_alloc/ma+rm.cc](#), and [examples/libs/l4re/c/ma+rm.c](#).

Definition at line 467 of file [consts.h](#).

References [L4_INLINE](#), [L4_NOTHROW](#), and [L4_PAGEMASK](#).

Referenced by [L4virtio::Svr::Driver_mem_region_t< No_custom_data >::Driver_mem_region_t\(\)](#), [l4_touch_ro\(\)](#), [l4_touch_rw\(\)](#), and [L4Re::Util::Dataspace_svr::map\(\)](#).

Here is the caller graph for this function:



13.1.7.4.5 l4_trunc_size()

```
l4_addr_t l4_trunc_size (
    l4_addr_t address,
    unsigned char bits) [inline]
```

Round an address down to the next lower flexpage with size *bits*.

Parameters

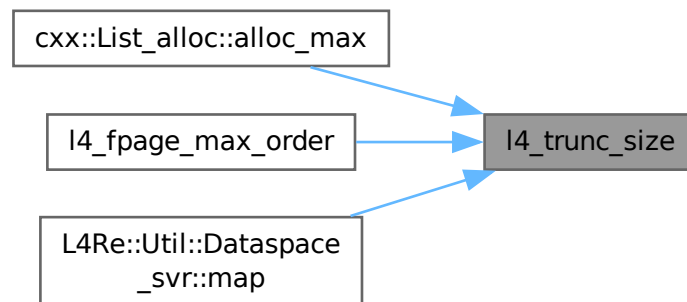
| | |
|----------------|----------------------------------|
| <i>address</i> | The address to round. |
| <i>bits</i> | The size of the flexpage (log2). |

Definition at line 478 of file [consts.h](#).

References [L4_INLINE](#), and [L4_NOTHROW](#).

Referenced by [cxx::List_alloc::alloc_max\(\)](#), [l4_fpage_max_order\(\)](#), and [L4Re::Util::Dataspace_svr::map\(\)](#).

Here is the caller graph for this function:



13.1.8 Error codes

Common error codes.

Collaboration diagram for Error codes:



Enumerations

- enum `l4_error_code_t` {
`L4_EOK` = 0 , `L4_EPERM` = 1 , `L4_ENOENT` = 2 , `L4_EIO` = 5 ,
`L4_ENXIO` = 6 , `L4_E2BIG` = 7 , `L4_EAGAIN` = 11 , `L4_ENOMEM` = 12 ,
`L4_EACCESS` = 13 , `L4_EFAULT` = 14 , `L4_EBUSY` = 16 , `L4_EEXIST` = 17 ,
`L4_ENODEV` = 19 , `L4_ENOTDIR` = 20 , `L4_EINVAL` = 22 , `L4_ENOSPC` = 28 ,
`L4_ERANGE` = 34 , `L4_ENAMETOOLONG` = 36 , `L4_ENOSYS` = 38 , `L4_EBADPROTO` = 39 ,
`L4_EADDRNOTAVAIL` = 99 , `L4_ERRNOMAX` = 100 , `L4_ENOREPLY` = 1000 , `L4_MSGTOOSHORT` = 1001 ,
`L4_MSGTOOLONG` = 1002 , `L4_MSGMISSARG` = 1003 , `L4_MSGERRRANGE` = 1004 ,
`L4_EDROPREPLY` = 1005 ,
`L4_EIPC_LO` = 2000 , `L4_EIPC_HI` = 2000 + 0x1f }
L4 error codes.

13.1.8.1 Detailed Description

Common error codes.

Include File

```
#include <l4/sys/err.h>
```

13.1.8.2 Enumeration Type Documentation

13.1.8.2.1 l4_error_code_t

```
enum l4_error_code_t
```

L4 error codes.

Those error codes are used by both the kernel and the user programs.

Enumerator

| | |
|------------------|-----------------------------------|
| L4_EOK | Ok. |
| L4_EPERM | No permission. |
| L4_ENOENT | No such entity. |
| L4_EIO | I/O error. |
| L4_ENXIO | No such device or address. |
| L4_E2BIG | Argument value too big. |
| L4_EAGAIN | Try again. |
| L4_ENOMEM | No memory. |
| L4_EACCESS | Permission denied. |
| L4_EFAULT | Invalid memory address. |
| L4_EBUSY | Object currently busy, try later. |
| L4_EEXIST | Already exists. |
| L4_ENODEV | No such thing. |
| L4_ENOTDIR | Not a directory. |
| L4_EINVAL | Invalid argument. |
| L4_ENOSPC | No space left on device. |
| L4_ERANGE | Range error. |
| L4_ENAMETOOLONG | Name too long. |
| L4_ENOSYS | No sys. |
| L4_EBADPROTO | Unsupported protocol. |
| L4_EADDRNOTAVAIL | Address not available. |
| L4_ERRNOMAX | Maximum error value. |
| L4_ENOREPLY | No reply. |
| L4_EMMSGTOOSHORT | Message too short. |

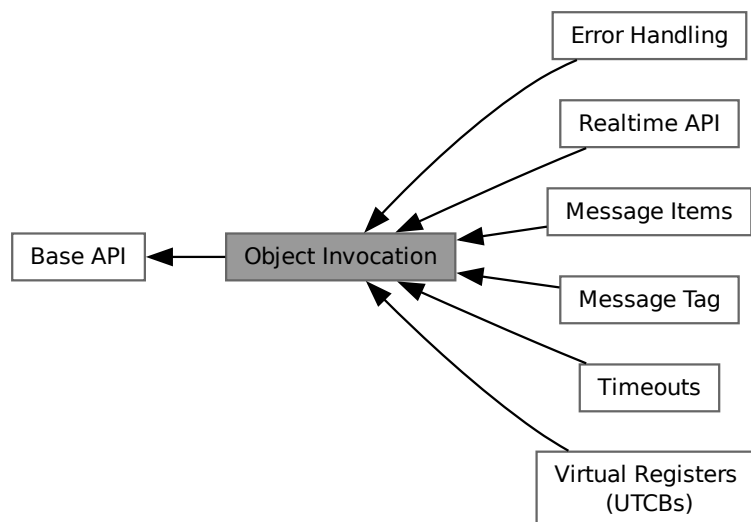
| | |
|----------------|----------------------------------|
| L4_MSGTOOLONG | Message too long. |
| L4_MSGMISSARG | Message has invalid capability. |
| L4_MSGERRRANGE | Error code range error. |
| L4_EDROPREPLY | Server dropped reply capability. |
| L4_EIPC_LO | Communication error-range low. |
| L4_EIPC_HI | Communication error-range high. |

Definition at line 31 of file [err.h](#).

13.1.9 Object Invocation

API for [L4](#) object invocation.

Collaboration diagram for Object Invocation:



Topics

- [Message Items](#) 229
Message-item-related functionality.
- [Timeouts](#) 236
All kinds of timeouts and time related functions.
- [Error Handling](#) 245
Error handling for [L4](#) object invocation.
-

- Realtime API 251
- Message Tag 251
API related to the message tag data type.
- Virtual Registers (UTCBs) 264
L4 Virtual Registers (UTCB).

Files

- file [utcb.h](#)
UTCB definitions.

Macros

- #define [L4_SYSF_NONE](#)
Capability selector flags.

Functions

- [l4_msgtag_t l4_ipc_send](#) ([l4_cap_idx_t](#) dest, [l4_utcb_t](#) *utcb, [l4_msgtag_t](#) tag, [l4_timeout_t](#) timeout) [L4_NOTHROW](#)
Send a message to an object (do **not** wait for a reply).
- [l4_msgtag_t l4_ipc_wait](#) ([l4_utcb_t](#) *utcb, [l4_umword_t](#) *label, [l4_timeout_t](#) timeout) [L4_NOTHROW](#)
Wait for an incoming message from any possible sender.
- [l4_msgtag_t l4_ipc_receive](#) ([l4_cap_idx_t](#) object, [l4_utcb_t](#) *utcb, [l4_timeout_t](#) timeout) [L4_NOTHROW](#)
Wait for a message from a specific source.
- [l4_msgtag_t l4_ipc_call](#) ([l4_cap_idx_t](#) object, [l4_utcb_t](#) *utcb, [l4_msgtag_t](#) tag, [l4_timeout_t](#) timeout) [L4_NOTHROW](#)
Object call (usual invocation).
- [l4_msgtag_t l4_ipc_reply](#) ([l4_cap_idx_t](#) reply_cap, [l4_utcb_t](#) *utcb, [l4_msgtag_t](#) tag, [l4_timeout_t](#) timeout) [L4_NOTHROW](#)
Reply operation (uses a reply capability).
- [l4_msgtag_t l4_ipc_reply_and_wait](#) ([l4_utcb_t](#) *utcb, [l4_msgtag_t](#) tag, [l4_umword_t](#) *label, [l4_timeout_t](#) timeout) [L4_NOTHROW](#)
Reply and wait operation (uses the reply capability).
- [l4_msgtag_t l4_ipc_send_and_wait](#) ([l4_cap_idx_t](#) dest, [l4_utcb_t](#) *utcb, [l4_msgtag_t](#) tag, [l4_umword_t](#) *label, [l4_timeout_t](#) timeout) [L4_NOTHROW](#)
Send a message and do an open wait.
- [l4_msgtag_t l4_ipc](#) ([l4_cap_idx_t](#) dest, [l4_utcb_t](#) *utcb, [l4_umword_t](#) flags, [l4_umword_t](#) slabel, [l4_msgtag_t](#) tag, [l4_umword_t](#) *rlabel, [l4_timeout_t](#) timeout) [L4_NOTHROW](#)
Generic L4 object invocation.
- [l4_msgtag_t l4_ipc_sleep](#) ([l4_timeout_t](#) timeout) [L4_NOTHROW](#)
Sleep for an amount of time.
- [l4_msgtag_t l4_ipc_sleep_ms](#) ([l4_uint32_t](#) ms) [L4_NOTHROW](#)
Sleep for a certain amount of milliseconds.
- [l4_msgtag_t l4_ipc_sleep_us](#) ([l4_uint64_t](#) us) [L4_NOTHROW](#)
Sleep for a certain amount of microseconds.
- int [l4_sndfpage_add](#) ([l4_fpage_t](#) const snd_fpage, unsigned long snd_base, [l4_msgtag_t](#) *tag) [L4_NOTHROW](#)
Add a flexpage to be sent to the UTCB.

13.1.9.1 Detailed Description

API for [L4](#) object invocation.

Include File

```
#include <l4/sys/ipc.h>
```

General abstractions for [L4](#) object invocation. The basic principle is that all objects are denoted by a capability that is accessed via a capability selector (see [Capabilities](#)).

This set of functions is common to all kinds of objects provided by the [L4](#) micro kernel. The concrete semantics of an invocation depends on the object that shall be invoked.

Objects may be invoked in various ways, the most common way is to use a *call* operation ([l4_ipc_call\(\)](#)). However, there are a lot more flavours available that have a semantics depending on the object.

See also

[IPC-Gate API](#)

[L4 Inter-Process Communication \(IPC\)](#)

13.1.9.2 Timeouts during IPC

IPC operation between two communication partners may consist of up to two phases (send phase and receive phase). For both phases, a timeout may be specified (send timeout and receive timeout).

Note

When IPC communication happens across CPU cores and a timeout is specified, then the counting of the timeout only begins after the target thread has been scheduled at least once. In particular, this means that an IPC timeout, including a timeout of zero, may be delayed depending on the scheduling on the target CPU core. If a higher priority thread on the target core is executing a busy loop, that delay may even be indefinitely.

See also

[Timeouts](#)

13.1.9.3 Macro Definition Documentation

13.1.9.3.1 L4_SYSF_NONE

```
#define L4_SYSF_NONE
```

Capability selector flags.

These flags determine the concrete operation when a kernel object is invoked.

The following combinations of flags are supported when invoking IPC (see [l4_ipc\(\)](#)); with other combinations, behavior is undefined:

- [L4_SYSF_SEND](#): send to specified partner
- [L4_SYSF_RECV](#): receive from specified partner
- [L4_SYSF_RECV](#) | [L4_SYSF_OPEN_WAIT](#): receive from any sending partner; see [L4_SYSF_WAIT](#)
- [L4_SYSF_SEND](#) | [L4_SYSF_RECV](#): call specified partner; see [L4_SYSF_CALL](#)
- [L4_SYSF_SEND](#) | [L4_SYSF_RECV](#) | [L4_SYSF_OPEN_WAIT](#): send to specified partner and receive from any sending partner; see [L4_SYSF_SEND_AND_WAIT](#)
- [L4_SYSF_REPLY](#) | [L4_SYSF_SEND](#): reply to caller
- [L4_SYSF_REPLY](#) | [L4_SYSF_SEND](#) | [L4_SYSF_RECV](#): call the caller
- [L4_SYSF_REPLY](#) | [L4_SYSF_SEND](#) | [L4_SYSF_RECV](#) | [L4_SYSF_OPEN_WAIT](#): reply to caller and receive from any sending partner; see [L4_SYSF_REPLY_AND_WAIT](#) Empty set of flags.

Definition at line 55 of file [consts.h](#).

13.1.9.4 Function Documentation

13.1.9.4.1 [l4_ipc\(\)](#)

```
l4_msgtag_t l4_ipc (
    l4_cap_idx_t dest,
    l4_utcb_t * utcb,
    l4_umword_t flags,
    l4_umword_t slabel,
    l4_msgtag_t tag,
    l4_umword_t * rlabel,
    l4_timeout_t timeout) [inline]
```

Generic [L4](#) object invocation.

Parameters

| | | |
|-----|----------------|--|
| | <i>dest</i> | Destination object. L4_INVALID_CAP denotes the current thread. An IPC to the current thread will always abort after the specified timeout and can be used for sleeping without busy waiting. |
| | <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |
| | <i>flags</i> | Invocation flags (see L4_SYSF_NONE). |
| | <i>slabel</i> | Send label if applicable (may be seen by the receiver). |
| | <i>tag</i> | Sending message tag. |
| out | <i>rlabel</i> | Receiving label. |
| | <i>timeout</i> | Timeout pair (see l4_timeout_t). |

Returns

return tag

13.1.9.4.2 l4_ipc_call()

```
l4_msgtag_t l4_ipc_call (
    l4_cap_idx_t object,
    l4_utcb_t * utcb,
    l4_msgtag_t tag,
    l4_timeout_t timeout) [inline]
```

Object call (usual invocation).

Parameters

| | |
|----------------|--|
| <i>object</i> | Capability selector for the object to call. A value of L4_INVALID_CAP denotes the current thread and will abort the IPC after the time specified in the <code>snd</code> part of the <code>timeout</code> parameter has expired. |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |
| <i>tag</i> | Message tag to describe the message to be sent. |
| <i>timeout</i> | Timeout pair for send and receive phase (see l4_timeout_t). |

Returns

result tag

A message is sent to the object and the invoker waits for a reply from the object. Messages from other sources are not accepted.

Note

The send-to-receive transition needs no time, the object can reply with a send timeout of zero.

If a finite receive timeout is specified, the IPC receive operation could abort before the partner was able to send the reply message. Under certain circumstances the partner may still have the temporary reply capability to the calling thread and may use this capability to reply to the caller at a later, unexpected time specifying an arbitrary IPC label. This case is relevant for servers which call another, possibly untrusted, server while serving a client request.

See also

[L4 Inter-Process Communication \(IPC\)](#)

Examples

[examples/sys/aliens/main.c](#), [examples/sys/ipc/ipc_example.c](#), and [examples/sys/singlestep/main.c](#).

Definition at line 599 of file [ipc.h](#).

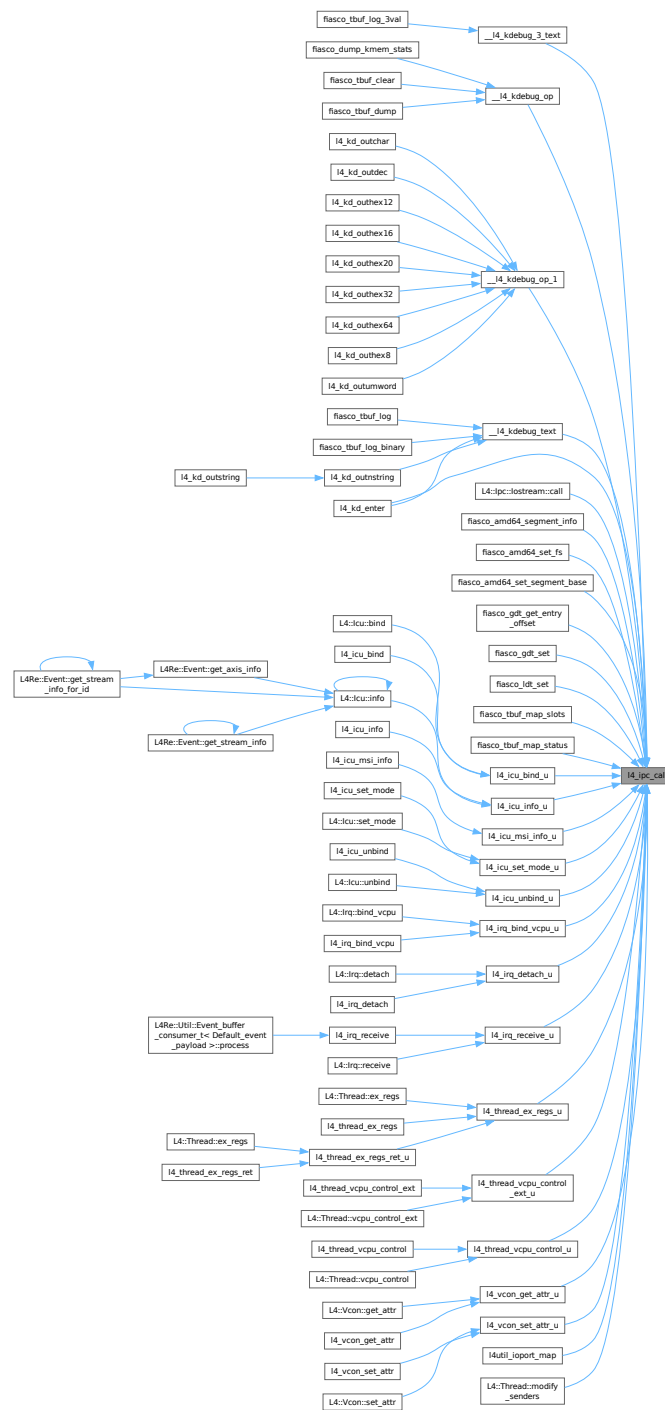
References [l4_ipc\(\)](#), [L4_NOTHROW](#), and [L4_SYSF_CALL](#).

Referenced by [__l4_kdebug_3_text\(\)](#), [__l4_kdebug_op\(\)](#), [__l4_kdebug_op_1\(\)](#), [__l4_kdebug_text\(\)](#), [L4::lpc::lostream::call\(\)](#), [fiasco_amd64_segment_info\(\)](#), [fiasco_amd64_set_fs\(\)](#), [fiasco_amd64_set_segment_base\(\)](#), [fiasco_gdt_get_entry_offset\(\)](#), [fiasco_gdt_set\(\)](#), [fiasco_ldt_set\(\)](#), [fiasco_tbuf_map_slots\(\)](#), [fiasco_tbuf_map_status\(\)](#), [l4_icu_bind_u\(\)](#), [l4_icu_info_u\(\)](#), [l4_icu_msi_info_u\(\)](#), [l4_icu_set_mode_u\(\)](#), [l4_icu_unbind_u\(\)](#), [l4_irq_bind_vcpu_u\(\)](#), [l4_irq_detach_u\(\)](#), [l4_irq_receive_u\(\)](#), [l4_kd_enter\(\)](#), [l4_thread_ex_regs_u\(\)](#), [l4_thread_vcpu_control_ext_u\(\)](#), [l4_thread_vcpu_control_u\(\)](#), [l4_vcon_get_attr_u\(\)](#), [l4_vcon_set_attr_u\(\)](#), [l4util_ioport_map\(\)](#), and [L4::Thread::modify_senders\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.1.9.4.3 l4_ipc_receive()

```
14_msgtag_t 14_ipc_receive (
    14_cap_idx_t object,
    14_utcb_t * utcb,
    14_timeout_t timeout) [inline]
```

Wait for a message from a specific source.

Parameters

| | |
|----------------|---|
| <i>object</i> | Thread or IRQ object to receive a message from. A value of L4_INVALID_CAP denotes the current thread. This can be used for sleeping without busy waiting for the time specified in the <code>rcv</code> part of the <code>timeout</code> parameter. |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |
| <i>timeout</i> | Timeout pair (see l4_timeout_t , only the receive part matters). |

Returns

result tag.

This operation waits for a message from the specified object. Messages from other sources are not accepted by this operation. The operation does not include a send phase, this means no message is sent to the object. See [l4_ipc_wait\(\)](#) for general message receiving.

Note

This operation is used to receive messages from a specific IRQ or thread. Receiving from IPC gates or any other objects does not work. It is not common to use the receive-only IPC operation for normal client/server communication.

See also

[L4 Inter-Process Communication \(IPC\)](#)

Examples

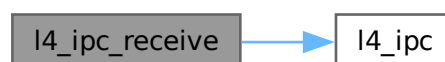
[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 644 of file [ipc.h](#).

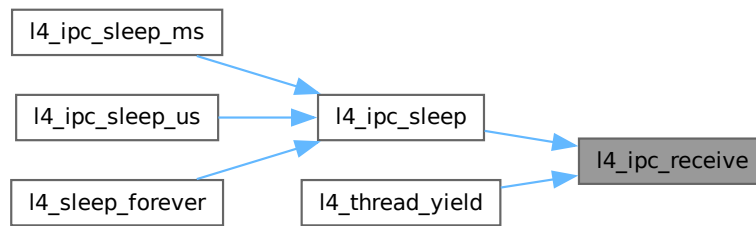
References [l4_ipc\(\)](#), [L4_NOTHROW](#), [L4_SYSF_RECV](#), and [l4_msgtag_t::raw](#).

Referenced by [l4_ipc_sleep\(\)](#), and [l4_thread_yield\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.1.9.4.4 l4_ipc_reply()

```

l4_msgtag_t l4_ipc_reply (
    l4_cap_idx_t reply_cap,
    l4_utcb_t * utcb,
    l4_msgtag_t tag,
    l4_timeout_t timeout) [inline]
  
```

Reply operation (uses a *reply* capability).

Parameters

| | |
|------------------|--|
| <i>reply_cap</i> | Reply capability selector. A value of L4_INVALID_CAP denotes the implicit reply capability of the current thread. Otherwise, to use an explicit reply capability, the <code>L4_REPLY_CAP_BIT</code> must be set. |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |
| <i>tag</i> | Describes the message to be sent as reply. |
| <i>timeout</i> | Timeout pair (see l4_timeout_t). |

Returns

result tag

A message is sent to a previous caller using the given reply capability. The reply capability is always invalidated by the call, even if the IPC operation fails.

See also

[L4 Inter-Process Communication \(IPC\)](#)

Definition at line 606 of file [ipc.h](#).

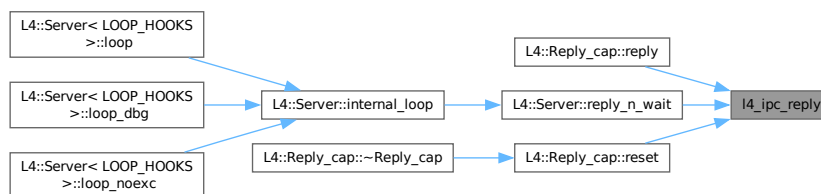
References [l4_ipc\(\)](#), [L4_NOTHROW](#), [L4_SYSF_REPLY](#), and [L4_SYSF_SEND](#).

Referenced by [L4::Reply_cap::reply\(\)](#), [L4::Server< LOOP_HOOKS >::reply_n_wait\(\)](#), and [L4::Reply_cap::reset\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.1.9.4.5 l4_ipc_reply_and_wait()

```

l4_msgtag_t l4_ipc_reply_and_wait (
    l4_utcb_t * utcb,
    l4_msgtag_t tag,
    l4_umword_t * label,
    l4_timeout_t timeout) [inline]
  
```

Reply and wait operation (uses the *reply* capability).

Parameters

| | | |
|-----|----------------|--|
| | <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |
| | <i>tag</i> | Describes the message to be sent as reply. |
| out | <i>label</i> | Label assigned to the source object of the received message. |
| | <i>timeout</i> | Timeout pair (see l4_timeout_t). |

Returns

result tag

A message is sent to the previous caller using the implicit reply capability. Afterwards the invoking thread waits for a message from any source.

Note

This is the standard server operation: it sends a reply to the actual client and waits for the next incoming request, which may come from any other client.

In case of multiple senders trying to send to the thread performing this system call, the thread receives from a sender with the highest priority. In this respect, IRQ sources have the highest priority 255.

See also

[L4 Inter-Process Communication \(IPC\)](#)

Examples

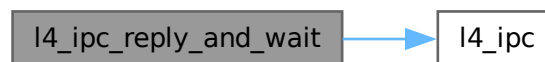
[examples/sys/ipc/ipc_example.c](#).

Definition at line 614 of file [ipc.h](#).

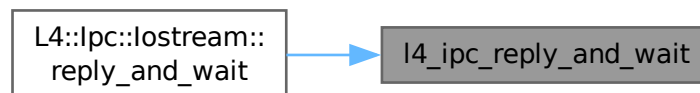
References [L4_INVALID_CAP](#), [l4_ipc\(\)](#), [L4_NOTHROW](#), and [L4_SYSF_REPLY_AND_WAIT](#).

Referenced by [L4::lpc::loststream::reply_and_wait\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.1.9.4.6 l4_ipc_send()

```
l4_msgtag_t l4_ipc_send (
    l4_cap_idx_t dest,
    l4_utcb_t * utcb,
    l4_msgtag_t tag,
    l4_timeout_t timeout) [inline]
```

Send a message to an object (do **not** wait for a reply).

Parameters

| | |
|----------------|---|
| <i>dest</i> | Capability selector for the destination object. A value of L4_INVALID_CAP denotes the current thread and could be used for sleeping without busy waiting for the time specified in the <code>snd</code> part of the <code>timeout</code> parameter. |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |
| <i>tag</i> | Descriptor for the message to be sent. |
| <i>timeout</i> | Timeout pair (see l4_timeout_t) only send part is relevant. |

Returns

Syscall return tag for the send-only operation, this means there is no return value except [L4_MSGTAG_ERROR](#) indicating success or failure of the send operation. Use [l4_ipc_error\(\)](#) to check for errors and **do not** use [l4_error\(\)](#).

A message is sent to the destination object. There is no receive phase included. The invoker continues working after sending the message.

Note

This is a special-purpose message transfer. Objects usually support only invocation via [l4_ipc_call\(\)](#) consisting of a send phase and a receive phase for returning the result of the object invocation. For example, [l4_icu_unmask\(\)](#), [l4_icu_mask\(\)](#) and [l4_irq_trigger\(\)](#) use send-only IPC operations for object invocation.

See also

[L4 Inter-Process Communication \(IPC\)](#)

Examples

[examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 628 of file [ipc.h](#).

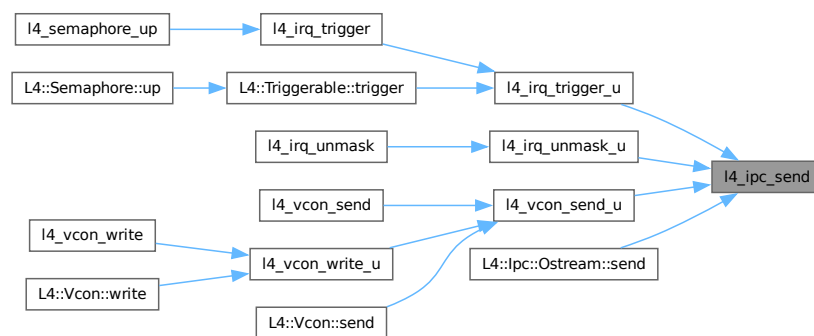
References [l4_ipc\(\)](#), [L4_NOTHROW](#), and [L4_SYSF_SEND](#).

Referenced by [l4_irq_trigger_u\(\)](#), [l4_irq_unmask_u\(\)](#), [l4_vcon_send_u\(\)](#), and [L4::lpc::Ostream::send\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.1.9.4.7 l4_ipc_send_and_wait()

```
14_msgtag_t 14_ipc_send_and_wait (
    14_cap_idx_t dest,
    14_utcb_t * utcb,
    14_msgtag_t tag,
    14_umword_t * label,
    14_timeout_t timeout) [inline]
```

Send a message and do an open wait.

Parameters

| | | |
|-----|----------------|---|
| | <i>dest</i> | Object to send a message to. A value of L4_INVALID_CAP denotes the current thread and will abort the IPC after the time specified in the <code>snd</code> part of the <code>timeout</code> parameter has expired. |
| | <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |
| | <i>tag</i> | Describes the message that shall be sent. |
| out | <i>label</i> | Label assigned to the source object of the receive phase. |
| | <i>timeout</i> | Timeout pair (see l4_timeout_t). |

Returns

result tag

A message is sent to the destination object and the invoking thread waits for a reply from any source.

Note

This is a special-purpose operation and shall not be used in general applications.

In case of multiple senders trying to send to the thread performing this system call, the thread receives from a sender with the highest priority. In this respect, IRQ sources have the highest priority 255.

See also

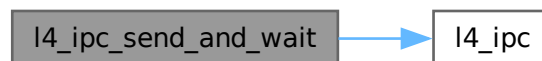
[L4 Inter-Process Communication \(IPC\)](#)

Definition at line 621 of file [ipc.h](#).

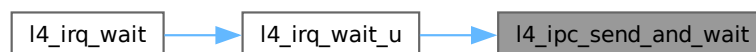
References [l4_ipc\(\)](#), [L4_NOTHROW](#), and [L4_SYSF_SEND_AND_WAIT](#).

Referenced by [l4_irq_wait_u\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.1.9.4.8 l4_ipc_sleep()

```
l4_msgtag_t l4_ipc_sleep (
    l4_timeout_t timeout) [inline]
```

Sleep for an amount of time.

Parameters

| | |
|----------------|---|
| <i>timeout</i> | Timeout pair (see l4_timeout_t , the receive part matters). |
|----------------|---|

Returns

error code:

- [L4_IPC_RETIMEOUT](#): success
- [L4_IPC_RECANCELED](#) woken up by a different thread ([l4_thread_ex_regs\(\)](#)).

The invoking thread waits until the timeout is expired or the wait was aborted by another thread by [l4_thread_ex_regs\(\)](#).

See also

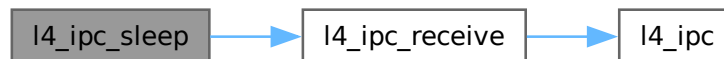
[L4 Inter-Process Communication \(IPC\)](#)

Definition at line 653 of file [ipc.h](#).

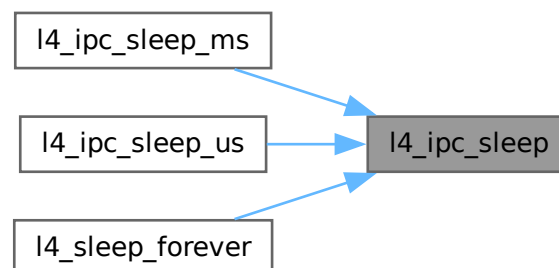
References [L4_INVALID_CAP](#), [l4_ipc_receive\(\)](#), and [L4_NOTHROW](#).

Referenced by [l4_ipc_sleep_ms\(\)](#), [l4_ipc_sleep_us\(\)](#), and [l4_sleep_forever\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.1.9.4.9 l4_ipc_sleep_ms()

```
l4_msgtag_t l4_ipc_sleep_ms (
    l4_uint32_t ms) [inline]
```

Sleep for a certain amount of milliseconds.

Parameters

| | |
|-----------|---------------------------------|
| <i>ms</i> | Number of milliseconds to wait. |
|-----------|---------------------------------|

Returns

error code:

- [L4_IPC_RETIMEOUT](#): success
- [L4_IPC_RECANCELED](#) woken up by a different thread ([l4_thread_ex_regs\(\)](#)).

The invoking thread waits until the timeout is expired or the wait was aborted by another thread by [l4_thread_ex_regs\(\)](#).

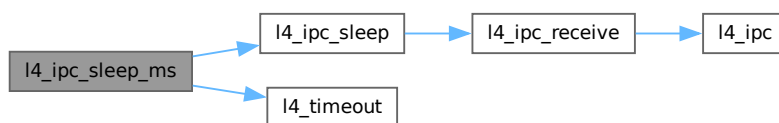
See also

[L4 Inter-Process Communication \(IPC\)](#)

Definition at line 657 of file [ipc.h](#).

References [l4_ipc_sleep\(\)](#), [L4_IPC_TIMEOUT_NEVER](#), [L4_NOTHROW](#), and [l4_timeout\(\)](#).

Here is the call graph for this function:



13.1.9.4.10 l4_ipc_sleep_us()

```
l4_msgtag_t l4_ipc_sleep_us (
    l4_uint64_t us) [inline]
```

Sleep for a certain amount of microseconds.

Parameters

| | |
|-----------|---------------------------------|
| <i>us</i> | Number of microseconds to wait. |
|-----------|---------------------------------|

Returns

error code:

- [L4_IPC_RETIMEOUT](#): success
- [L4_IPC_RECANCELED](#) woken up by a different thread ([l4_thread_ex_regs\(\)](#)).

The invoking thread waits until the timeout is expired or the wait was aborted by another thread by [l4_thread_ex_regs\(\)](#).

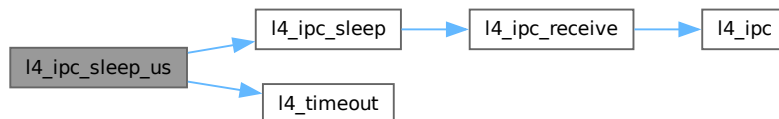
See also

[L4 Inter-Process Communication \(IPC\)](#)

Definition at line 664 of file [ipc.h](#).

References [l4_ipc_sleep\(\)](#), [L4_IPC_TIMEOUT_NEVER](#), [L4_NOTHROW](#), and [l4_timeout\(\)](#).

Here is the call graph for this function:

**13.1.9.4.11 l4_ipc_wait()**

```

l4_msgtag_t l4_ipc_wait (
    l4_utcb_t * utcb,
    l4_umword_t * label,
    l4_timeout_t timeout) [inline]
  
```

Wait for an incoming message from any possible sender.

Parameters

| | | |
|-----|----------------|--|
| | <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |
| out | <i>label</i> | Label assigned to the source object (IPC gate or IRQ). |
| | <i>timeout</i> | Timeout pair (see l4_timeout_t , only the receive part is used). |

Returns

return tag

This operation does an open wait, and therefore needs no capability to denote the possible source of a message. This means the calling thread waits for an incoming message from any possible source. There is no send phase included in this operation.

The usual usage of this function is to call that function when entering a server loop in a user-level server that implements user-level objects, see also [l4_ipc_reply_and_wait\(\)](#).

Note

In case of multiple senders trying to send to the thread performing this system call, the thread receives from a sender with the highest priority. In this respect, IRQ sources have the highest priority 255.

See also

[L4 Inter-Process Communication \(IPC\)](#)

Examples

[examples/sys/ipc/ipc_example.c](#).

Definition at line 635 of file [ipc.h](#).

References [L4_INVALID_CAP](#), [l4_ipc\(\)](#), [L4_NOTHROW](#), [L4_SYSF_WAIT](#), and [l4_msgtag_t::raw](#).

Referenced by [L4::lpc::lstream::wait\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.1.9.4.12 l4_sndfpage_add()

```
int l4_sndfpage_add (
    l4_fpage_t const snd_fpage,
    unsigned long snd_base,
    l4_msgtag_t * tag) [inline]
```

Add a flexpage to be sent to the UTCB.

Parameters

| | | |
|---------|------------------|---|
| | <i>snd_fpage</i> | Flexpage. |
| | <i>snd_base</i> | Send base. |
| in, out | <i>tag</i> | Tag to be updated. Only the number of items is incremented in the updated tag, all other members remain unmodified. |

Returns

0 on success, negative error code otherwise

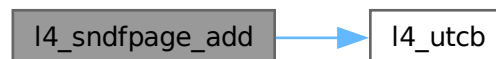
See also

[L4 Inter-Process Communication \(IPC\)](#)

Definition at line 727 of file [ipc.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

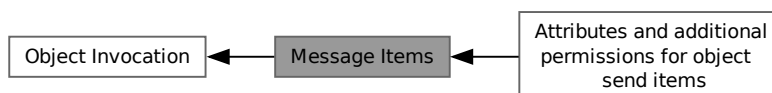
Here is the call graph for this function:



13.1.9.5 Message Items

Message-item-related functionality.

Collaboration diagram for Message Items:



Topics

- [Attributes and additional permissions for object send items](#) 235
These rights need to be added to the `snd_base` when mapping and control internal behavior.

Data Structures

- [struct `l4_snd_fpage_t`](#)
Send-flexpage types.

Enumerations

- [enum `l4_fpage_cacheability_opt_t`](#) { [L4_FPAGE_CACHE_OPT](#) = 0x1 , [L4_FPAGE_CACHEABLE](#) = 0x3 , [L4_FPAGE_BUFFERABLE](#) = 0x5 , [L4_FPAGE_UNCACHEABLE](#) = 0x1 }
Cacheability options for memory send items.
- [enum `l4_msg_item_consts_t`](#) {
[L4_ITEM_MAP](#) = 8 , [L4_ITEM_CONT](#) = 1 , [L4_MAP_ITEM_GRANT](#) = 2 , [L4_MAP_ITEM_MAP](#) = 0 ,
[L4_RCV_ITEM_FORWARD_MAPPINGS](#) = 1 , [L4_RCV_ITEM_SINGLE_CAP](#) = [L4_ITEM_MAP](#) | 2 ,
[L4_RCV_ITEM_LOCAL_ID](#) = 4 }
Constants for message items.

Functions

- [L4_CONSTEXPR `l4_umword_t` `l4_map_control` \(`l4_umword_t` spot, unsigned char cache, unsigned grant\) \[L4_NOTHROW\]\(#\)](#)
Create the first word for a map item that is a send item for the memory space.
- [L4_CONSTEXPR `l4_umword_t` `l4_map_obj_control` \(`l4_umword_t` spot, unsigned grant\) \[L4_NOTHROW\]\(#\)](#)
Create the first word for a map item that is a send item for the object space.

13.1.9.5.1 Detailed Description

Message-item-related functionality.

Message items are typed items that are used for transferring capabilities during IPC. There are three sub-types of typed message items with variations in the layout:

1. Typed message items set by the sender in its message registers (MRs) of the UTCB for specifying what shall be sent.
2. Typed message items set by the receiver in its buffer registers (BRs) of the UTCB for specifying which types of capabilities may be received at which addresses.
3. Typed message items set by the kernel in the receiver's message registers (MRs) of the UTCB for providing information about the transfer to the receiver.

They are abbreviated by *send item*, *receive item*, and *return item*, respectively.

A typed message item in the message registers (case 1 and case 3) always consists of two words (even if it is a void item). The size of a typed message item in the buffer registers (case 2) is determined by its first word. The size is up to three words (see [L4_RCV_ITEM_SINGLE_CAP](#) and [L4_RCV_ITEM_FORWARD_MAPPINGS](#)). A void item in the buffer registers consists of a single word.

Include File

```
#include <l4/sys/types.h>
```


13.1.9.5.2 Enumeration Type Documentation

13.1.9.5.2.1 l4_fpage_cacheability_opt_t

```
enum l4_fpage_cacheability_opt_t
```

Cacheability options for memory send items.

Only the IPC sender and the thread performing the map operation can specify the caching mode of the target mapping. By default, the caching mode of the sender is used as caching mode for the target mapping. If `L4_FPAGE_CACHE_OPT` is set in the send item, the caching mode is overridden by the respective mode from below.

Enumerator

| | |
|----------------------|--|
| L4_FPAGE_CACHE_OPT | Enable the cacheability option in a memory send item. Without this flag, the options are copied from the sender. |
| L4_FPAGE_CACHEABLE | Cacheability option to enable caches for the mapping. Implies L4_FPAGE_CACHE_OPT . |
| L4_FPAGE_BUFFERABLE | Cacheability option to enable buffered writes for the mapping. Implies L4_FPAGE_CACHE_OPT . |
| L4_FPAGE_UNCACHEABLE | Cacheability option to disable caching for the mapping. Implies L4_FPAGE_CACHE_OPT . |

Definition at line 303 of file [__l4_fpage.h](#).

13.1.9.5.2.2 l4_msg_item_consts_t

```
enum l4_msg_item_consts_t
```

Constants for message items.

Enumerator

| | |
|--------------|---|
| L4_ITEM_MAP | Identify a message item as <i>map item</i> . |
| L4_ITEM_CONT | Denote that the following item shall be put into the same receive item as this one. |

| | |
|------------------------------|---|
| L4_MAP_ITEM_GRANT | <p>Flag as <i>grant</i> instead of <i>map</i> operation. This means, the sender delegates access to the receiver and the kernel removes the rights from the sender (basically a move operation). The mapping in the receiver gets the new parent of any child mappings of the mapping of the sender. Rights revocation via send item/flexpage is <i>not</i> guaranteed to be applied to descendant mappings in case of grant. See Spaces and Mappings for more details on map/grant.</p> <p>Note</p> <p>The grant operation is not performed if the resulting rights of the receiver mapping would not contain the L4_CAP_FPAGE_R bit (for object capabilities) or none of the L4_FPAGE_RWX bits (memory and IO ports). In that case, the mapping is not created in the receiver space and not removed from the sender space.</p> <p>If the removal of the whole mapping from the sender is not possible because the size of the mapped frame at the sender exceeds the size defined by the send or receive flexpage, the grant operation is turned into a regular map operation and the mapping is <i>not</i> removed from the sender. This would happen if, for example, a smaller part of an L4 superpage mapping shall be granted.</p> |
| L4_MAP_ITEM_MAP | Flag as usual <i>map</i> operation. |
| L4_RCV_ITEM_FORWARD_MAPPINGS | <p>This flag specifies if received capabilities shall be mapped to a particular task instead of the invoking task. This flag may be used only if L4_RCV_ITEM_LOCAL_ID is unset.</p> <p>Setting this flag increases the size of the buffer item by one word. This word is used to specify a capability index for the task that shall receive the mappings.</p> |
| L4_RCV_ITEM_SINGLE_CAP | <p>Mark the receive buffer to be a small receive item that describes a buffer for a single object capability. A receive item needs to specify a <i>receive window</i>. The receive window determines which kind of capabilities (object, memory, I/O ports) may be received where in the respective space. If this flag is unset, the receive window is specified in the second word of the receive item via a flexpage. If this flag is set, the receive window consists of a single capability index in the object space and the capability index is specified in the most significant bits of the first word of the receive item (see L4_CAP_SHIFT).</p> |

| | |
|----------------------|---|
| L4_RCV_ITEM_LOCAL_ID | <p>The receiver requests to receive a local ID instead of a mapping whenever possible. This flag may be used only if L4_RCV_ITEM_SINGLE_CAP is set and L4_RCV_ITEM_FORWARD_MAPPINGS is unset.</p> <p>When this flag is set, then,</p> <ul style="list-style-type: none"> • when sender and receiver are bound to the same task, then no mapping is done for this item and just the raw flexpage (l4_fpage_t) is transferred, • otherwise, when the sender specified an IPC gate for transfer that is bound to a thread that is bound to the same task as the receiving thread, then no mapping is done for this item and just the bitwise OR () of the label and the L4_CAP_FPAGE_W and L4_CAP_FPAGE_S permissions that would have been mapped is transferred, • otherwise a regular mapping is done for this item. |
|----------------------|---|

Definition at line 231 of file [consts.h](#).

13.1.9.5.3 Function Documentation

13.1.9.5.3.1 l4_map_control()

```
L4_CONSTEXPR l4_umword_t l4_map_control (
    l4_umword_t spot,
    unsigned char cache,
    unsigned grant) [inline]
```

Create the first word for a map item that is a send item for the memory space.

Parameters

| | |
|--------------|---|
| <i>spot</i> | Hot spot address, used to determine what is actually mapped when send and receive flexpage have differing sizes. |
| <i>cache</i> | Cacheability hints for memory flexpages. See Cacheability options . |
| <i>grant</i> | Indicates if it is a map or a grant item. Allowed values: L4_MAP_ITEM_MAP , L4_MAP_ITEM_GRANT . |

Returns

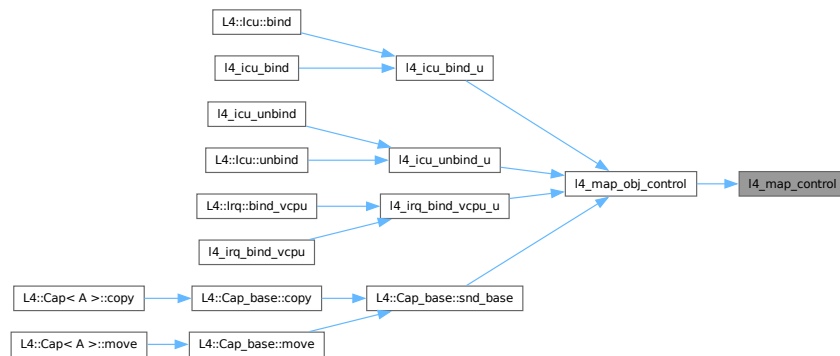
The value to be used as first word in a send item for memory.

Definition at line 758 of file [__l4_fpage.h](#).

References [L4_FPAGE_CONTROL_MASK](#), [L4_ITEM_MAP](#), and [L4_NOTHROW](#).

Referenced by [l4_map_obj_control\(\)](#).

Here is the caller graph for this function:



13.1.9.5.3.2 l4_map_obj_control()

```

L4_CONSTEXPR l4_umword_t l4_map_obj_control (
    l4_umword_t spot,
    unsigned grant) [inline]
  
```

Create the first word for a map item that is a send item for the object space.

Parameters

| | |
|--------------|--|
| <i>spot</i> | Hot spot address, used to determine what is actually mapped when send and receive flexpages have different size. |
| <i>grant</i> | Indicates if it is a map item or a grant item. Allowed values: L4_MAP_ITEM_MAP , L4_MAP_ITEM_GRANT . |

Returns

The value to be used as first word in a send item for kernel objects or IO-ports.

Definition at line 765 of file [__l4_fpage.h](#).

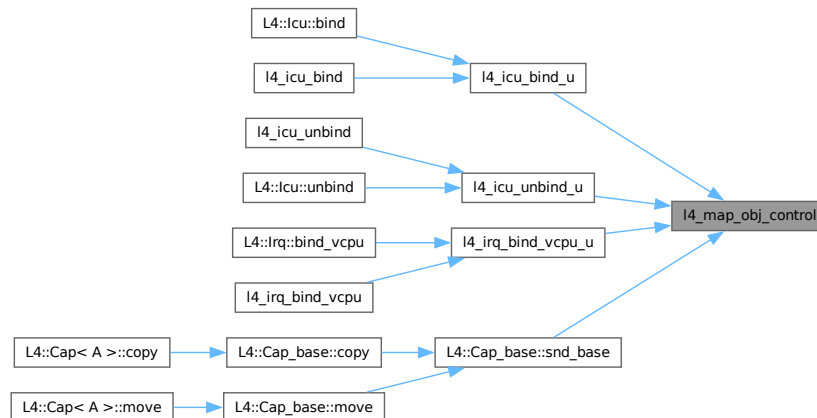
References [l4_map_control\(\)](#), and [L4_NOTHROW](#).

Referenced by [l4_icu_bind_u\(\)](#), [l4_icu_unbind_u\(\)](#), [l4_irq_bind_vcpu_u\(\)](#), and [L4::Cap_base::snd_base\(\)](#).

Here is the call graph for this function:



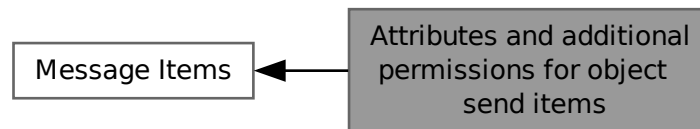
Here is the caller graph for this function:



13.1.9.5.4 Attributes and additional permissions for object send items

These rights need to be added to the snd_base when mapping and control internal behavior.

Collaboration diagram for Attributes and additional permissions for object send items:



Macros

- **#define L4_FPAGE_C_REF_CNT** 0x00
Mapping is reference-counted (default).
- **#define L4_FPAGE_C_NO_REF_CNT** 0x10
Don't increase the reference counter.
- **#define L4_FPAGE_C_OBJ_RIGHT1** 0x20
Object-type specific right.
- **#define L4_FPAGE_C_OBJ_RIGHT2** 0x40
Object-type specific right.
- **#define L4_FPAGE_C_OBJ_RIGHT3** 0x80
Object-type specific right.
- **#define L4_FPAGE_C_OBJ_RIGHTS** 0xe0
All Object-type specific right bits.
- **#define L4_FPAGE_C_IPCGATE_SVR** L4_FPAGE_C_OBJ_RIGHT1
The receiver may invoke IPC-gate-specific functions on the capability, for example, bind a thread to the gate and modify the label.

13.1.9.5.4.1 Detailed Description

These rights need to be added to the `snd_base` when mapping and control internal behavior.

The exact meaning depends on the type of capability (currently used only with IPC gates).

13.1.9.5.4.2 Macro Definition Documentation

L4_FPAGE_C_IPCGATE_SVR

```
#define L4_FPAGE_C_IPCGATE_SVR L4_FPAGE_C_OBJ_RIGHT1
```

The receiver may invoke IPC-gate-specific functions on the capability, for example, bind a thread to the gate and modify the label.

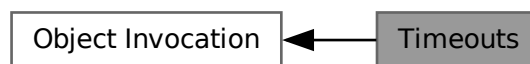
Needed if the receiver implements the server side of an IPC gate.

Definition at line 288 of file `__l4_fpage.h`.

13.1.9.6 Timeouts

All kinds of timeouts and time related functions.

Collaboration diagram for Timeouts:



Data Structures

- struct `l4_timeout_s`
Basic timeout specification.
- union `l4_timeout_t`
Timeout pair.

Macros

- `#define L4_IPC_TIMEOUT_0 ((l4_timeout_s){0x0400})`
Timeout constants.
- `#define L4_IPC_TIMEOUT_NEVER ((l4_timeout_s){0})`
never timeout
- `#define L4_IPC_NEVER_INITIALIZER {0}`
never timeout, initializer
- `#define L4_IPC_NEVER ((l4_timeout_t){0})`
never timeout
- `#define L4_IPC_RECV_TIMEOUT_0 ((l4_timeout_t){0x00000400})`
0 receive timeout
- `#define L4_IPC_SEND_TIMEOUT_0 ((l4_timeout_t){0x04000000})`
0 send timeout
- `#define L4_IPC_BOTH_TIMEOUT_0 ((l4_timeout_t){0x04000400})`
0 receive and send timeout
- `#define L4_TIMEOUT_US_NEVER (~0ULL)`
The waiting period in microseconds which is interpreted as "never" by l4_timeout_from_us().
- `#define L4_TIMEOUT_US_MAX ((1ULL << 41) - 1)`
The longest waiting period in microseconds accepted by l4_timeout_from_us().

Typedefs

- `typedef struct l4_timeout_s l4_timeout_s`
Basic timeout specification.
- `typedef union l4_timeout_t l4_timeout_t`
Timeout pair.

Functions

- `L4_CONSTEXPR l4_timeout_s l4_timeout_rel (unsigned man, unsigned exp) L4_NOTHROW`
Get relative timeout consisting of mantissa and exponent.
- `L4_CONSTEXPR l4_timeout_t l4_ipc_timeout (unsigned snd_man, unsigned snd_exp, unsigned rcv_man, unsigned rcv_exp) L4_NOTHROW`
Convert explicit timeout values to l4_timeout_t type.
- `L4_CONSTEXPR l4_timeout_t l4_timeout (l4_timeout_s snd, l4_timeout_s rcv) L4_NOTHROW`
Combine send and receive timeout in a timeout.
- `L4_CONSTEXPR void l4_snd_timeout (l4_timeout_s snd, l4_timeout_t *to) L4_NOTHROW`
Set send timeout in given to timeout.
- `L4_CONSTEXPR void l4_rcv_timeout (l4_timeout_s rcv, l4_timeout_t *to) L4_NOTHROW`
Set receive timeout in given to timeout.
- `L4_CONSTEXPR l4_kernel_clock_t l4_timeout_rel_get (l4_timeout_s to) L4_NOTHROW`
Get clock value of out timeout.
- `L4_CONSTEXPR unsigned l4_timeout_is_absolute (l4_timeout_s to) L4_NOTHROW`
Return whether the given timeout is absolute or not.
- `L4_CONSTEXPR l4_kernel_clock_t l4_timeout_get (l4_kernel_clock_t cur, l4_timeout_s to) L4_NOTHROW`
Get clock value for a clock + a timeout.
- `l4_timeout_s l4_timeout_abs (l4_kernel_clock_t pint, int br) L4_NOTHROW`
Set an absolute timeout.
- `unsigned l4_utcb_mr64_idx (unsigned idx) L4_NOTHROW`
Get index into 64bit message registers alias from native-sized index.

13.1.9.6.1 Detailed Description

All kinds of timeouts and time related functions.

13.1.9.6.2 Macro Definition Documentation

13.1.9.6.2.1 L4_IPC_TIMEOUT_0

```
#define L4_IPC_TIMEOUT_0 ((l4_timeout_s){0x0400})
```

Timeout constants.

0 timeout

Definition at line 73 of file [__timeout.h](#).

Referenced by [L4::ipc_svr::Timeout_queue_hooks<HOOKS, BR_MAN>::timeout\(\)](#).

13.1.9.6.2.2 L4_TIMEOUT_US_MAX

```
#define L4_TIMEOUT_US_MAX ((1ULL << 41) - 1)
```

The longest waiting period in microseconds accepted by [l4_timeout_from_us\(\)](#).

See [l4_timeout_from_us\(\)](#) for an explanation.

Definition at line 91 of file [__timeout.h](#).

13.1.9.6.3 Typedef Documentation

13.1.9.6.3.1 l4_timeout_s

```
typedef struct l4_timeout_s l4_timeout_s
```

Basic timeout specification.

If bit 15 == 0, basically a floating point number with 10 bits mantissa and 5 bits exponent ($t = m \cdot 2^e$).

If the mantissa is zero, the exponent encodes special values, see [L4_IPC_TIMEOUT_0](#) and [L4_IPC_TIMEOUT_NEVER](#).

If bit 15 == 1 the timeout is absolute and the lower 6 bits encode the index of the UTCB buffer register(s) holding the absolute 64-bit timeout value. On 32-bit systems, two consecutive UTCB buffer registers are used.

13.1.9.6.3.2 l4_timeout_t

```
typedef union l4_timeout_t l4_timeout_t
```

Timeout pair.

For IPC there are usually a send and a receive timeout. So this structure contains a pair of timeouts.

13.1.9.6.4 Function Documentation

13.1.9.6.4.1 l4_ipc_timeout()

```
L4_CONSTEXPR l4_timeout_t l4_ipc_timeout (
    unsigned snd_man,
    unsigned snd_exp,
    unsigned rcv_man,
    unsigned rcv_exp) [inline]
```

Convert explicit timeout values to [l4_timeout_t](#) type.

Parameters

| | |
|----------------|------------------------------|
| <i>snd_man</i> | Mantissa of send timeout. |
| <i>snd_exp</i> | Exponent of send timeout. |
| <i>rcv_man</i> | Mantissa of receive timeout. |
| <i>rcv_exp</i> | Exponent of receive timeout. |

Definition at line 207 of file [__timeout.h](#).

References [L4_NOTHROW](#), and [l4_timeout\(\)](#).

Here is the call graph for this function:



13.1.9.6.4.2 l4_rcv_timeout()

```
L4_CONSTEXPR void l4_rcv_timeout (
    l4_timeout_s rcv,
    l4_timeout_t * to) [inline]
```

Set receive timeout in given to timeout.

Parameters

| | | |
|------------|------------|----------------------------|
| | <i>rcv</i> | Receive timeout |
| <i>out</i> | <i>to</i> | L4 timeout |

Definition at line 231 of file [__timeout.h](#).

References [L4_NOTHROW](#).

13.1.9.6.4.3 l4_snd_timeout()

```
L4_CONSTEXPR void l4_snd_timeout (
    l4_timeout_s snd,
    l4_timeout_t * to) [inline]
```

Set send timeout in given to timeout.

Parameters

| | | |
|-----|------------|--------------|
| | <i>snd</i> | Send timeout |
| out | <i>to</i> | L4 timeout |

Definition at line 224 of file [__timeout.h](#).

References [L4_NOTHROW](#).

13.1.9.6.4.4 l4_timeout()

```
L4_CONSTEXPR l4_timeout_t l4_timeout (
    l4_timeout_s snd,
    l4_timeout_s rcv) [inline]
```

Combine send and receive timeout in a timeout.

Parameters

| | |
|------------|-----------------|
| <i>snd</i> | Send timeout |
| <i>rcv</i> | Receive timeout |

Returns

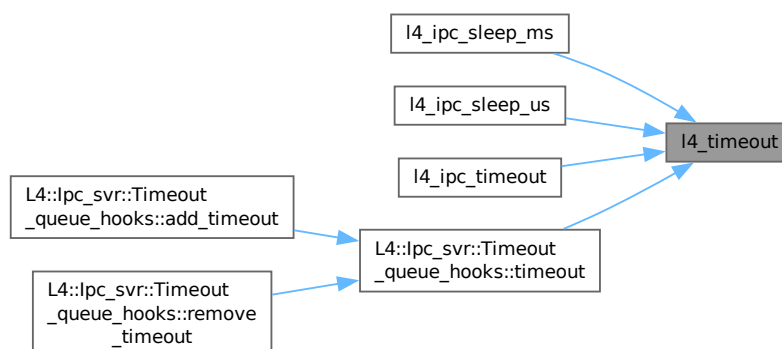
L4 timeout

Definition at line 217 of file [__timeout.h](#).

References [L4_NOTHROW](#).

Referenced by [l4_ipc_sleep_ms\(\)](#), [l4_ipc_sleep_us\(\)](#), [l4_ipc_timeout\(\)](#), and [L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN](#)

Here is the caller graph for this function:



13.1.9.6.4.5 l4_timeout_abs()

```
l4_timeout_s l4_timeout_abs (
    l4_kernel_clock_t pint,
    int br) [inline]
```

Set an absolute timeout.

Parameters

| | |
|-------------|---|
| <i>pint</i> | Point in time in clocks |
| <i>br</i> | The buffer register the timeout shall be placed in. (|

Note

On 32bit architectures the timeout needs two consecutive buffers.)

The absolute timeout value will be placed into the buffer register *br* of the current thread.

Returns

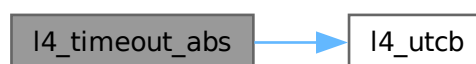
timeout value

Definition at line 412 of file [utcb.h](#).

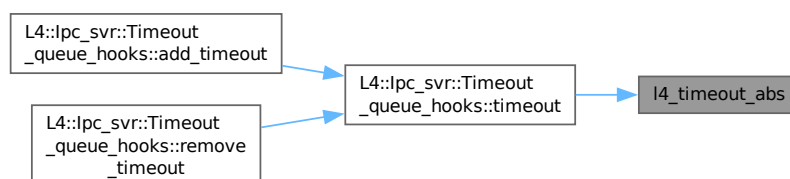
References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Referenced by [L4::lpc_svr::Timeout_queue_hooks<HOOKS, BR_MAN>::timeout\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.1.9.6.4.6 l4_timeout_get()

```
L4_CONSTEXPR l4_kernel_clock_t l4_timeout_get (  
    l4_kernel_clock_t cur,  
    l4_timeout_s to) [inline]
```

Get clock value for a clock + a timeout.

Parameters

| | |
|------------|-------------|
| <i>cur</i> | Clock value |
| <i>to</i> | L4 timeout |

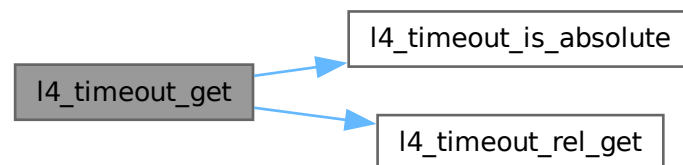
Returns

Clock sum

Definition at line 261 of file [__timeout.h](#).

References [L4_NOTHROW](#), [l4_timeout_is_absolute\(\)](#), and [l4_timeout_rel_get\(\)](#).

Here is the call graph for this function:



13.1.9.6.4.7 l4_timeout_is_absolute()

```
L4_CONSTEXPR unsigned l4_timeout_is_absolute (  
    l4_timeout_s to) [inline]
```

Return whether the given timeout is absolute or not.

Parameters

| | |
|-----------|------------|
| <i>to</i> | L4 timeout |
|-----------|------------|

Returns

!= 0 if absolute, 0 if relative

Definition at line 254 of file [__timeout.h](#).

References [L4_NOTHROW](#).

Referenced by [l4_timeout_get\(\)](#).

Here is the caller graph for this function:

**13.1.9.6.4.8 l4_timeout_rel()**

```
L4_CONSTEXPR l4_timeout_s l4_timeout_rel (  
    unsigned man,  
    unsigned exp) [inline]
```

Get relative timeout consisting of mantissa and exponent.

Parameters

| | |
|------------|---------------------|
| <i>man</i> | Mantissa of timeout |
| <i>exp</i> | Exponent of timeout |

Returns

timeout value

Definition at line 238 of file [__timeout.h](#).

References [L4_NOTHROW](#).

13.1.9.6.4.9 l4_timeout_rel_get()

```
L4_CONSTEXPR l4_kernel_clock_t l4_timeout_rel_get (
    l4_timeout_s to) [inline]
```

Get clock value of out timeout.

Parameters

| | |
|-----------|------------|
| <i>to</i> | L4 timeout |
|-----------|------------|

Returns

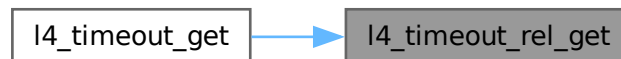
Clock value

Definition at line 245 of file [__timeout.h](#).

References [L4_NOTHROW](#).

Referenced by [l4_timeout_get\(\)](#).

Here is the caller graph for this function:



13.1.9.6.4.10 l4_utcb_mr64_idx()

```
unsigned l4_utcb_mr64_idx (
    unsigned idx) [inline]
```

Get index into 64bit message registers alias from native-sized index.

Parameters

| | |
|------------|--|
| <i>idx</i> | Index to native-sized message register |
|------------|--|

Returns

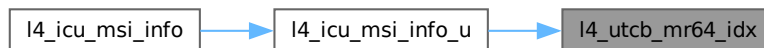
Index to 64bit message register alias

Definition at line 415 of file [utcb.h](#).

References [L4_INLINE](#), and [L4_NOTHROW](#).

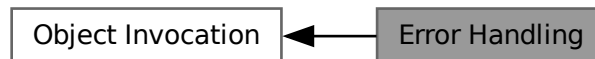
Referenced by [l4_icu_msi_info_u\(\)](#).

Here is the caller graph for this function:

**13.1.9.7 Error Handling**

Error handling for [L4](#) object invocation.

Collaboration diagram for Error Handling:

**Enumerations**

- enum [l4_ipc_tcr_error_t](#) {
[L4_IPC_ERROR_MASK](#) = 0x1F , [L4_IPC_SND_ERR_MASK](#) = 0x01 , [L4_IPC_ENOT_EXISTENT](#) = 0x04 ,
[L4_IPC_RETIMEOUT](#) = 0x03 ,
[L4_IPC_SETIMEOUT](#) = 0x02 , [L4_IPC_RECANCELED](#) = 0x07 , [L4_IPC_SECANCELED](#) = 0x06 ,
[L4_IPC_REMAPFAILED](#) = 0x11 ,
[L4_IPC_SEMAPFAILED](#) = 0x10 , [L4_IPC_RESNDPFTO](#) = 0x0b , [L4_IPC_SESNDPFTO](#) = 0x0a ,
[L4_IPC_RERCVPFTO](#) = 0x0d ,
[L4_IPC_SERCVPFTO](#) = 0x0c , [L4_IPC_REABORTED](#) = 0x0f , [L4_IPC_SEABORTED](#) = 0x0e ,
[L4_IPC_REMSGCUT](#) = 0x09 ,
[L4_IPC_SEMSGCUT](#) = 0x08 , [L4_IPC_NO_REPLY_CAP](#) = 0x13 }

Error codes in the error TCR.

Functions

- [l4_umword_t l4_ipc_error](#) ([l4_msgtag_t](#) tag, [l4_utcb_t](#) *utcb) [L4_NOTHROW](#)
Get the IPC error code for an IPC operation.
- [l4_ret_t l4_error](#) ([l4_msgtag_t](#) tag) [L4_NOTHROW](#)
Get IPC error code if any or message tag label otherwise for an IPC call.
- [int l4_ipc_is_snd_error](#) ([l4_utcb_t](#) *utcb) [L4_NOTHROW](#)
Returns whether an error occurred in send phase of an invocation.
- [int l4_ipc_is_rcv_error](#) ([l4_utcb_t](#) *utcb) [L4_NOTHROW](#)
Returns whether an error occurred in receive phase of an invocation.
- [int l4_ipc_error_code](#) ([l4_utcb_t](#) *utcb) [L4_NOTHROW](#)
Get the error condition of the last invocation from the TCR.

13.1.9.7.1 Detailed Description

Error handling for [L4](#) object invocation.

Include File

```
#include <l4/sys/ipc.h>
```

13.1.9.7.2 Enumeration Type Documentation

13.1.9.7.2.1 l4_ipc_tcr_error_t

```
enum l4_ipc_tcr_error_t
```

Error codes in the *error* TCR.

The error codes are accessible via the *error* TCR, see [l4_thread_regs_t.error](#).

Enumerator

| | |
|--------------------------------------|---|
| L4_IPC_ERROR_MASK | Mask for error bits. |
| L4_IPC_SND_ERR_MASK | Send error mask. |
| L4_IPC_ENOT_EXISTENT | Non-existing destination or source. |
| L4_IPC_RETIMEOUT | Timeout during receive operation. |
| L4_IPC_SETIMEOUT | Timeout during send operation. |
| L4_IPC_RECANCELED | Receive operation canceled. |
| L4_IPC_SECANCELED | Send operation canceled. |
| L4_IPC_REMAPFAILED | Map flexpage failed in receive operation. |
| L4_IPC_SEMAPFAILED | Map flexpage failed in send operation. |
| L4_IPC_RESNDPFTO | Send-pagefault timeout in receive operation. |
| L4_IPC_SESNDPFTO | Send-pagefault timeout in send operation. |
| L4_IPC_RERCVPFTO | Receive-pagefault timeout in receive operation. |

| | |
|---------------------|--|
| L4_IPC_SERCVPFTO | Receive-pagefault timeout in send operation. |
| L4_IPC_REABORTED | Receive operation aborted. |
| L4_IPC_SEABORTED | Send operation aborted. |
| L4_IPC_REMSGCUT | Received message truncated. Usually returned when the typed items to be sent by the IPC partner exceed the buffer registers of the respective types. |
| L4_IPC_SEMSGCUT | Sent message truncated. Usually returned when the typed items to be sent exceed the IPC partner's buffer registers of the respective types. |
| L4_IPC_NO_REPLY_CAP | Receive operation using explicit reply capability failed. Either the reply capability index was too large or the callers memory quota was exhausted. |

Definition at line 81 of file [ipc.h](#).

13.1.9.7.3 Function Documentation

13.1.9.7.3.1 l4_error()

```
l4_ret_t l4_error (
    l4_msgtag_t tag) [inline]
```

Get IPC error code if any or message tag label otherwise for an IPC call.

This function shall only be used if the IPC operation includes a receive phase (usually a call operation), otherwise no tag label is received and the return value of this function is undefined.

Parameters

| | |
|------------|---------------------------------------|
| <i>tag</i> | Message tag returned by the IPC call. |
|------------|---------------------------------------|

Returns

In case of an IPC error, a negative error code in the range of [L4_EIPC_LO](#) to [L4_EIPC_HI](#) (see [l4_ipc_to_errno\(\)](#) and [l4_ipc_tcr_error_t](#)), otherwise the tag label. By convention, the callee can signal errors via a negative tag label (negated value from [l4_error_code_t](#)) and success via a non-negative value.

Examples

[examples/libs/l4re/streammap/client.cc](#), [examples/sys/aliens/main.c](#), [examples/sys/isr/main.c](#), [examples/sys/migrate/thread_migration.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 688 of file [ipc.h](#).

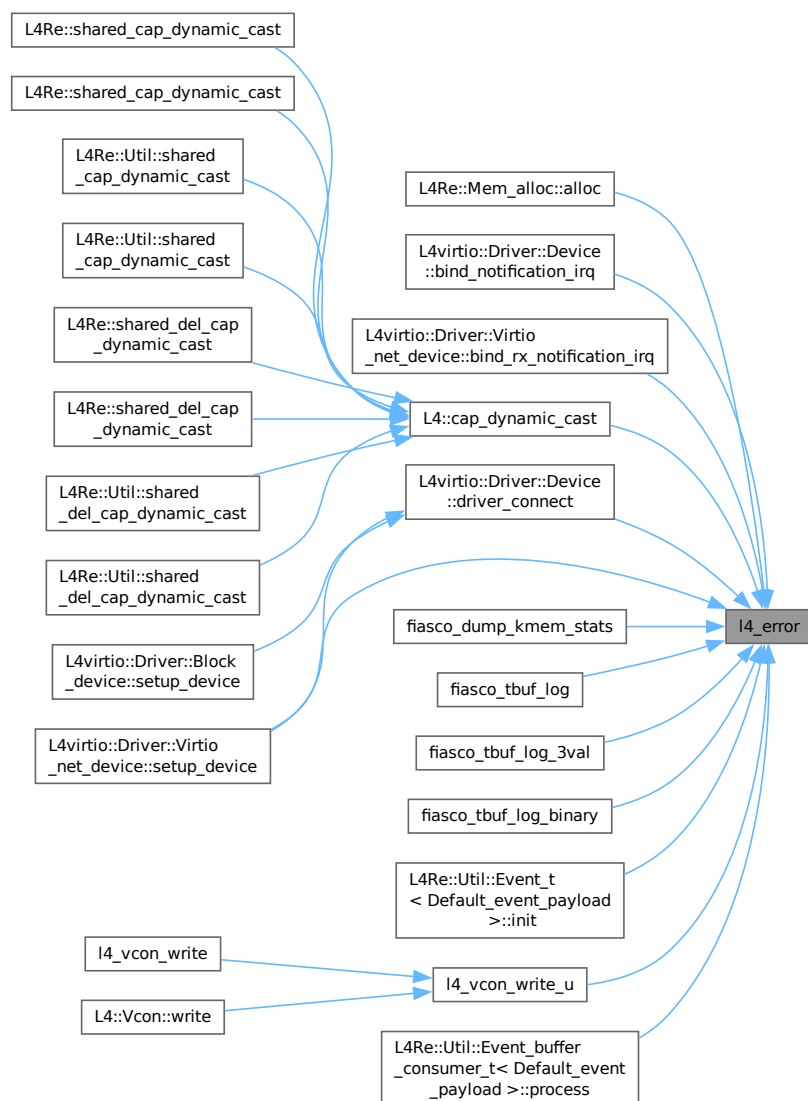
References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Referenced by [L4Re::Mem_alloc::alloc\(\)](#), [L4virtio::Driver::Device::bind_notification_irq\(\)](#), [L4virtio::Driver::Virtio_net_device::bind_rx_notification_irq\(\)](#), [L4::cap_dynamic_cast\(\)](#), [L4virtio::Driver::Device::driver_connect\(\)](#), [fiasco_dump_kmem_stats\(\)](#), [fiasco_tbuf_log\(\)](#), [fiasco_tbuf_log_3val\(\)](#), [fiasco_tbuf_log_binary\(\)](#), [L4Re::Util::Event_t< Default_event_payload >::init\(\)](#), [l4_vcon_write_u\(\)](#), [L4Re::Util::Event_buffer_consumer_t< Default_event_payload >::process\(\)](#), and [L4virtio::Driver::Virtio_net_device::setup_device\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.1.9.7.3.2 l4_ipc_error()

```
l4_umword_t l4_ipc_error (
    l4_msgtag_t tag,
    l4_utcb_t * utcb) [inline]
```

Get the IPC error code for an IPC operation.

Parameters

| | |
|-------------|--|
| <i>tag</i> | Message tag returned by the IPC operation. |
| <i>utcb</i> | UTCB that was used for the IPC operation. |

Returns

0 if no error condition is set, error code otherwise (see [l4_ipc_tcr_error_t](#)).

Examples

[examples/sys/ipc/ipc_example.c](#), and [examples/sys/start-with-exc/main.c](#).

Definition at line 671 of file [ipc.h](#).

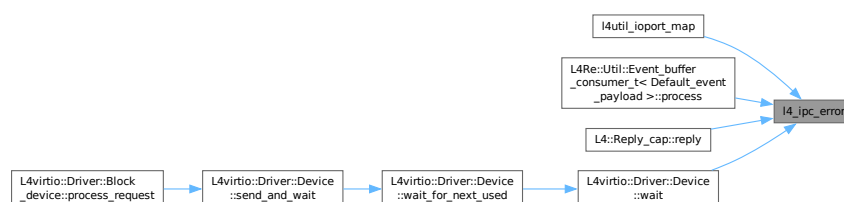
References [L4_IPC_ERROR_MASK](#), [L4_LIKELY](#), [l4_msgtag_has_error\(\)](#), and [L4_NOTHROW](#).

Referenced by [l4util_ioport_map\(\)](#), [L4Re::Util::Event_buffer_consumer_t< Default_event_payload >::process\(\)](#), [L4::Reply_cap::reply\(\)](#), and [L4virtio::Driver::Device::wait\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.1.9.7.3.3 `l4_ipc_error_code()`

```
int l4_ipc_error_code (  
    l4\_utcb\_t * utcb)    [inline]
```

Get the error condition of the last invocation from the TCR.

Precondition

`l4_msgtag_has_error(tag) == true`

Parameters

| | |
|-------------|----------------|
| <i>utcb</i> | UTCB to check. |
|-------------|----------------|

Returns

Error condition of type [l4_ipc_tcr_error_t](#).

Definition at line 700 of file [ipc.h](#).

References [L4_INLINE](#), [L4_IPC_ERROR_MASK](#), and [L4_NOTHROW](#).

13.1.9.7.3.4 `l4_ipc_is_rcv_error()`

```
int l4_ipc_is_rcv_error (  
    l4\_utcb\_t * utcb)    [inline]
```

Returns whether an error occurred in receive phase of an invocation.

Precondition

`l4_msgtag_has_error(tag) == true`

Parameters

| | |
|-------------|----------------|
| <i>utcb</i> | UTCB to check. |
|-------------|----------------|

Returns

Boolean value.

Definition at line 697 of file [ipc.h](#).

References [L4_INLINE](#), and [L4_NOTHROW](#).

13.1.9.7.3.5 l4_ipc_is_snd_error()

```
int l4_ipc_is_snd_error (  
    l4_utcb_t * utcb) [inline]
```

Returns whether an error occurred in send phase of an invocation.

Precondition

`l4_msgtag_has_error(tag) == true`

Parameters

| | |
|-------------|----------------|
| <i>utcb</i> | UTCB to check. |
|-------------|----------------|

Returns

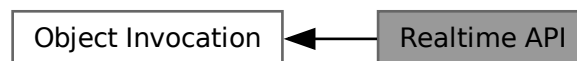
Boolean value.

Definition at line 694 of file [ipc.h](#).

References [L4_INLINE](#), and [L4_NOTHROW](#).

13.1.9.8 Realtime API

Collaboration diagram for Realtime API:



13.1.9.9 Message Tag

API related to the message tag data type.

Collaboration diagram for Message Tag:



Data Structures

- struct [l4_msgtag_t](#)
Message tag data structure.

Enumerations

- enum [L4_platform_ctl_proto](#) { [L4_PROTO_PLATFORM_CTL](#) = 0 }
Predefined protocol type for messages to platform-control objects.
- enum [L4_msgtag_protocol](#) {
[L4_PROTO_NONE](#) = 0 , [L4_PROTO_ALLOW_SYSCALL](#) = 1 , [L4_PROTO_PF_EXCEPTION](#) = 1 ,
[L4_PROTO_IRQ](#) = -1L ,
[L4_PROTO_PAGE_FAULT](#) = -2L , [L4_PROTO_EXCEPTION](#) = -5L , [L4_PROTO_SIGMA0](#) = -6L ,
[L4_PROTO_IO_PAGE_FAULT](#) = -8L ,
[L4_PROTO_THREAD_GROUP](#) = -9L , [L4_PROTO_KOBJECT](#) = -10L , [L4_PROTO_TASK](#) = -11L ,
[L4_PROTO_THREAD](#) = -12L ,
[L4_PROTO_LOG](#) = -13L , [L4_PROTO_SCHEDULER](#) = -14L , [L4_PROTO_FACTORY](#) = -15L ,
[L4_PROTO_VM](#) = -16L ,
[L4_PROTO_DMA_SPACE](#) = -17L , [L4_PROTO_IRQ_SENDER](#) = -18L , [L4_PROTO_SEMAPHORE](#) = -20L ,
[L4_PROTO_META](#) = -21L ,
[L4_PROTO_IOMMU](#) = -22L , [L4_PROTO_DEBUGGER](#) = -23L , [L4_PROTO_SMCCC](#) = -24L ,
[L4_PROTO_VCPU_CONTEXT](#) = -25L }
Message tag for IPC operations.
- enum [L4_msgtag_flags](#) { [L4_MSGTAG_ERROR](#) , [L4_MSGTAG_TRANSFER_FPU](#) , [L4_MSGTAG_SCHEDULE](#) ,
[L4_MSGTAG_PROPAGATE](#) = 0x4000 , [L4_MSGTAG_FLAGS](#) }
Flags for message tags.

Functions

- long [l4_msgtag_label](#) ([l4_msgtag_t](#)) [L4_NOTHROW](#)
Get the protocol of tag.
- unsigned [l4_msgtag_words](#) ([l4_msgtag_t](#)) [L4_NOTHROW](#)
Get the number of untyped words.
- unsigned [l4_msgtag_items](#) ([l4_msgtag_t](#)) [L4_NOTHROW](#)
Get the number of typed items.
- unsigned [l4_msgtag_flags](#) ([l4_msgtag_t](#)) [L4_NOTHROW](#)
Get the flags.
- unsigned [l4_msgtag_has_error](#) ([l4_msgtag_t](#)) [L4_NOTHROW](#)
Test for error indicator flag.
- unsigned [l4_msgtag_is_page_fault](#) ([l4_msgtag_t](#)) [L4_NOTHROW](#)
Test for page-fault protocol.
- unsigned [l4_msgtag_is_exception](#) ([l4_msgtag_t](#)) [L4_NOTHROW](#)
Test for exception protocol.
- unsigned [l4_msgtag_is_sigma0](#) ([l4_msgtag_t](#)) [L4_NOTHROW](#)
Test for sigma0 protocol.
- unsigned [l4_msgtag_is_io_page_fault](#) ([l4_msgtag_t](#)) [L4_NOTHROW](#)
Test for IO-page-fault protocol.
- [l4_msgtag_t](#) [l4_msgtag](#) (long label, unsigned words, unsigned items, unsigned flags) [L4_NOTHROW](#)
Create a message tag from the specified values.

13.1.9.9.1 Detailed Description

API related to the message tag data type.

Include File

```
#include <l4/sys/types.h>
```

13.1.9.9.2 Enumeration Type Documentation

13.1.9.9.2.1 L4_msgtag_flags

```
enum L4_msgtag_flags
```

Flags for message tags.

Enumerator

| | |
|------------------------|---|
| L4_MSGTAG_ERROR | Error indicator flag. |
| L4_MSGTAG_TRANSFER_FPU | Enable FPU transfer flag for IPC. By enabling this flag when sending IPC, the sender indicates that the contents of the FPU shall be transferred to the receiving thread. However, the receiver has to indicate its willingness to receive FPU context in its buffer descriptor register (BDR). |
| L4_MSGTAG_SCHEDULE | Enable schedule in IPC flag. Usually IPC operations donate the remaining time slice of a thread to the called thread. Enabling this flag when sending IPC does a real scheduling decision. However, this flag decreases IPC performance. |
| L4_MSGTAG_FLAGS | Mask for all flags. |

Definition at line 100 of file [types.h](#).

13.1.9.9.2.2 L4_msgtag_protocol

```
enum L4_msgtag_protocol
```

Message tag for IPC operations.

All predefined protocols used by the kernel.

Enumerator

| | |
|------------------------|--|
| L4_PROTO_NONE | Default protocol tag to reply to kernel. |
| L4_PROTO_ALLOW_SYSCALL | Allow an alien the system call. |
| L4_PROTO_PF_EXCEPTION | Make an exception out of a page fault. |
| L4_PROTO_IRQ | IRQ message. |
| L4_PROTO_PAGE_FAULT | Page fault message. |

| | |
|------------------------|--|
| L4_PROTO_EXCEPTION | Exception. |
| L4_PROTO_SIGMA0 | Sigma0 protocol. |
| L4_PROTO_IO_PAGE_FAULT | I/O page fault message. |
| L4_PROTO_THREAD_GROUP | Protocol for messages to a thread group obj. |
| L4_PROTO_KOBJECT | Protocol for messages to a generic kobject. |
| L4_PROTO_TASK | Protocol for messages to a task object. |
| L4_PROTO_THREAD | Protocol for messages to a thread object. |
| L4_PROTO_LOG | Protocol for messages to a log object. |
| L4_PROTO_SCHEDULER | Protocol for messages to a scheduler object. |
| L4_PROTO_FACTORY | Protocol for messages to a factory object. |
| L4_PROTO_VM | Protocol for messages to a virtual machine object. |
| L4_PROTO_DMA_SPACE | Protocol for (creating) kernel DMA space objects. |
| L4_PROTO_IRQ_SENDER | Protocol for IRQ senders (IRQ -> IPC). |
| L4_PROTO_SEMAPHORE | Protocol for semaphore objects. |
| L4_PROTO_META | Meta information protocol. |
| L4_PROTO_IOMMU | Protocol ID for IO-MMUs. |
| L4_PROTO_DEBUGGER | Protocol ID for the debugger. |
| L4_PROTO_SMCCC | Protocol ID for ARM SMCCC calls. |
| L4_PROTO_VCPU_CONTEXT | Protocol for hardware vCPU contexts. |

Definition at line 52 of file [types.h](#).

13.1.9.9.2.3 L4_platform_ctl_proto

```
enum L4_platform_ctl_proto
```

Predefined protocol type for messages to platform-control objects.

Enumerator

| | |
|-----------------------|---|
| L4_PROTO_PLATFORM_CTL | Protocol messages to a platform control object. See L4_platform_ctl_ops for allowed operations. |
|-----------------------|---|

Definition at line 174 of file [platform_control.h](#).

13.1.9.9.3 Function Documentation

13.1.9.9.3.1 l4_msgtag()

```
l4_msgtag_t l4_msgtag (
    long label,
    unsigned words,
    unsigned items,
    unsigned flags) [inline]
```


Create a message tag from the specified values.

Message tag functions.

Parameters

| | |
|--------------|---|
| <i>label</i> | The user-defined label |
| <i>words</i> | The number of untyped words within the UTCB |
| <i>items</i> | The number of typed items (e.g., flexpages) within the UTCB |
| <i>flags</i> | The IPC flags for realtime and FPU extensions |

Returns

Message tag

Examples

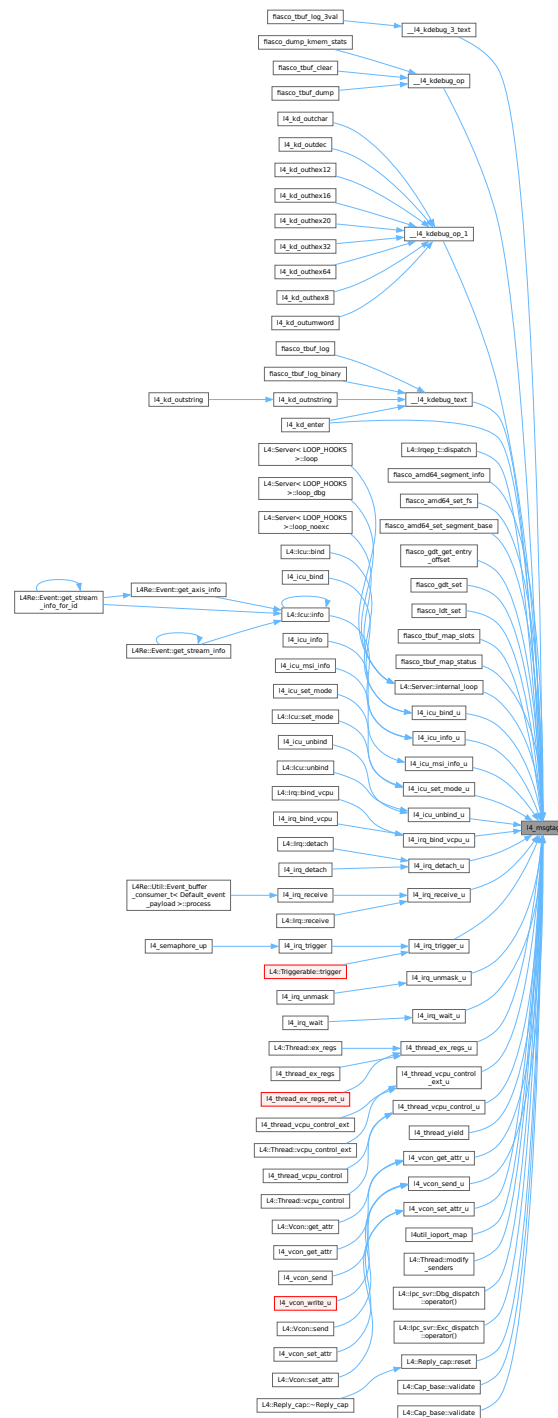
[examples/sys/aliens/main.c](#), [examples/sys/ipc/ipc_example.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#) and [examples/sys/utcb-ipc/main.c](#).

Definition at line 426 of file [types.h](#).

References [L4_NOTHROW](#).

Referenced by [__l4_kdebug_3_text\(\)](#), [__l4_kdebug_op\(\)](#), [__l4_kdebug_op_1\(\)](#), [__l4_kdebug_text\(\)](#), [L4::Irqep_t< Derived, BASE, bool>::internal_loop\(\)](#), [fiasco_amd64_segment_info\(\)](#), [fiasco_amd64_set_fs\(\)](#), [fiasco_amd64_set_segment_base\(\)](#), [fiasco_gdt_get_entry_offset\(\)](#), [fiasco_gdt_set\(\)](#), [fiasco_ldt_set\(\)](#), [fiasco_tbuf_map_slots\(\)](#), [fiasco_tbuf_map_status\(\)](#), [L4::Server< LOOP_HOOKS >::internal_loop\(\)](#), [l4_icu_bind_u\(\)](#), [l4_icu_info_u\(\)](#), [l4_icu_msi_info_u\(\)](#), [l4_icu_set_mode_u\(\)](#), [l4_icu_unbind_u\(\)](#), [l4_irq_bind_vcpu_u\(\)](#), [l4_irq_detach_u\(\)](#), [l4_irq_receive_u\(\)](#), [l4_irq_trigger_u\(\)](#), [l4_irq_unmask_u\(\)](#), [l4_irq_wait_u\(\)](#), [l4_kd_enter\(\)](#), [l4_thread_ex_regs_u\(\)](#), [l4_thread_vcpu_control_ext_u\(\)](#), [l4_thread_vcpu_control_u\(\)](#), [l4_thread_yield\(\)](#), [l4_vcon_get_attr_u\(\)](#), [l4_vcon_send_u\(\)](#), [l4_vcon_set_attr_u\(\)](#), [l4util_ioport_map\(\)](#), [L4::Thread::modify_senders\(\)](#), [L4::lpc_svr::Dbg_dispatch< R, Exc, Print>::operator\(\)](#), [L4::lpc_svr::Exc_dispatch< R, Exc >::operator\(\)](#), [L4::Reply_cap::reset\(\)](#), [L4::Cap_base::validate\(\)](#), and [L4::Cap_base::validate\(\)](#).

Here is the caller graph for this function:



13.1.9.9.3.2 l4_msgtag_flags()

```
unsigned l4_msgtag_flags (
    l4_msgtag_t t) [inline]
```

Get the flags.

The flag are defined by [L4_msgtag_flags](#).

Parameters

| | |
|----------|---------|
| <i>t</i> | The tag |
|----------|---------|

Returns

Flags

Definition at line 456 of file [types.h](#).

References [L4_NOTHROW](#).

Referenced by [l4_msgtag_t::flags\(\)](#).

Here is the caller graph for this function:



13.1.9.9.3.3 l4_msgtag_has_error()

```
unsigned l4_msgtag_has_error (  
    l4\_msgtag\_t t) [inline]
```

Test for error indicator flag.

Parameters

| | |
|----------|---------|
| <i>t</i> | The tag |
|----------|---------|

Returns

>0 for yes, 0 for no

Return whether the kernel operation caused a communication error, e.g. with IPC. if true: `utcb->error` is valid, otherwise `utcb->error` is not valid

Examples

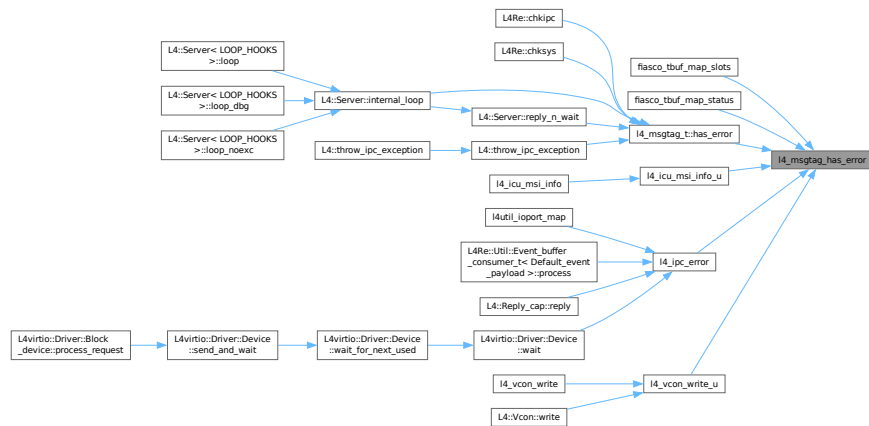
[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 461 of file [types.h](#).

References [L4_MSGTAG_ERROR](#), and [L4_NOTHROW](#).

Referenced by [fiasco_tbuf_map_slots\(\)](#), [fiasco_tbuf_map_status\(\)](#), [l4_msgtag_t::has_error\(\)](#), [l4_icu_msi_info_u\(\)](#), [l4_ipc_error\(\)](#), and [l4_vcon_write_u\(\)](#).

Here is the caller graph for this function:



13.1.9.9.3.4 l4_msgtag_is_exception()

```
unsigned l4_msgtag_is_exception (
    l4_msgtag_t t) [inline]
```

Test for exception protocol.

Parameters

| | |
|----------|---------|
| <i>t</i> | The tag |
|----------|---------|

Returns

Boolean value

Examples

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), and [examples/sys/start-with-exc/main.c](#).

Definition at line 469 of file [types.h](#).

References [L4_INLINE](#), [l4_msgtag_label\(\)](#), [L4_NOTHROW](#), and [L4_PROTO_EXCEPTION](#).

Referenced by [l4_msgtag_t::is_exception\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.1.9.9.3.5 l4_msgtag_is_io_page_fault()

```

unsigned l4_msgtag_is_io_page_fault (
    l4_msgtag_t t) [inline]
  
```

Test for IO-page-fault protocol.

Parameters

| | |
|----------|---------|
| <i>t</i> | The tag |
|----------|---------|

Returns

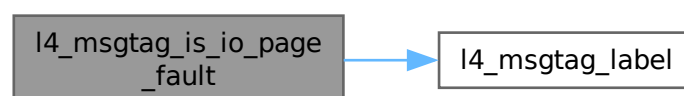
Boolean value

Definition at line 475 of file [types.h](#).

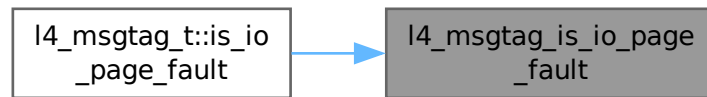
References [L4_INLINE](#), [l4_msgtag_label\(\)](#), [L4_NOTHROW](#), and [L4_PROTO_IO_PAGE_FAULT](#).

Referenced by [l4_msgtag_t::is_io_page_fault\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.1.9.9.3.6 l4_msgtag_is_page_fault()

```

unsigned l4_msgtag_is_page_fault (
    l4_msgtag_t t) [inline]
  
```

Test for page-fault protocol.

Parameters

| | |
|----------|---------|
| <i>t</i> | The tag |
|----------|---------|

Returns

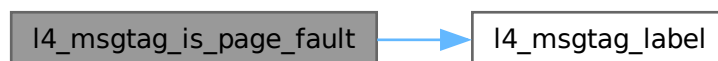
Boolean value

Definition at line 466 of file [types.h](#).

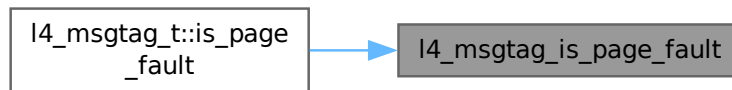
References [L4_INLINE](#), [l4_msgtag_label\(\)](#), [L4_NOTHROW](#), and [L4_PROTO_PAGE_FAULT](#).

Referenced by [l4_msgtag_t::is_page_fault\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.1.9.3.7 l4_msgtag_is_sigma0()

```
unsigned l4_msgtag_is_sigma0 (  
    l4_msgtag_t t) [inline]
```

Test for sigma0 protocol.

Parameters

| | |
|----------|---------|
| <i>t</i> | The tag |
|----------|---------|

Returns

Boolean value

Definition at line 472 of file [types.h](#).

References [L4_INLINE](#), [l4_msgtag_label\(\)](#), [L4_NOTHROW](#), and [L4_PROTO_SIGMA0](#).

Referenced by [l4_msgtag_t::is_sigma0\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.1.9.9.3.8 l4_msgtag_items()

```
unsigned l4_msgtag_items (  
    l4_msgtag_t t) [inline]
```

Get the number of typed items.

Parameters

| | |
|----------|---------|
| <i>t</i> | The tag |
|----------|---------|

Returns

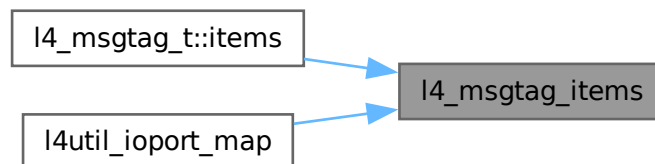
Number of items.

Definition at line 452 of file [types.h](#).

References [L4_NOTHROW](#).

Referenced by [l4_msgtag_t::items\(\)](#), and [l4util_ioport_map\(\)](#).

Here is the caller graph for this function:



13.1.9.9.3.9 l4_msgtag_label()

```
long l4_msgtag_label (  
    l4_msgtag_t t) [inline]
```

Get the protocol of tag.

Parameters

| | |
|----------|---------|
| <i>t</i> | The tag |
|----------|---------|

Returns

Label

Examples

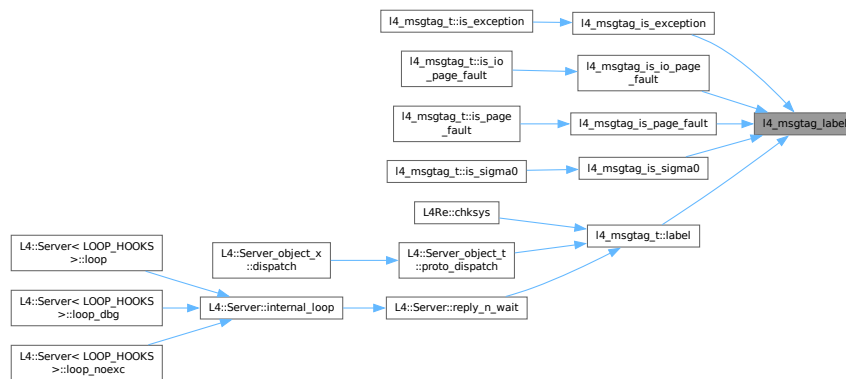
[examples/sys/singlestep/main.c](#), and [examples/sys/start-with-exc/main.c](#).

Definition at line 438 of file [types.h](#).

References [L4_NOTHROW](#).

Referenced by [l4_msgtag_is_exception\(\)](#), [l4_msgtag_is_io_page_fault\(\)](#), [l4_msgtag_is_page_fault\(\)](#), [l4_msgtag_is_sigma0\(\)](#), and [l4_msgtag_t::label\(\)](#).

Here is the caller graph for this function:

**13.1.9.9.3.10 l4_msgtag_words()**

```
unsigned l4_msgtag_words (
    l4_msgtag_t t) [inline]
```

Get the number of untyped words.

Parameters

| | |
|----------|---------|
| <i>t</i> | The tag |
|----------|---------|

Returns

Number of words

Examples

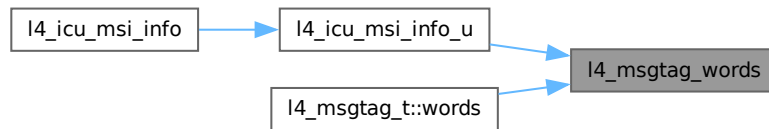
[examples/sys/utcb-ipc/main.c](#).

Definition at line 448 of file [types.h](#).

References [L4_NOTHROW](#).

Referenced by [l4_icu_msi_info_u\(\)](#), and [l4_msgtag_t::words\(\)](#).

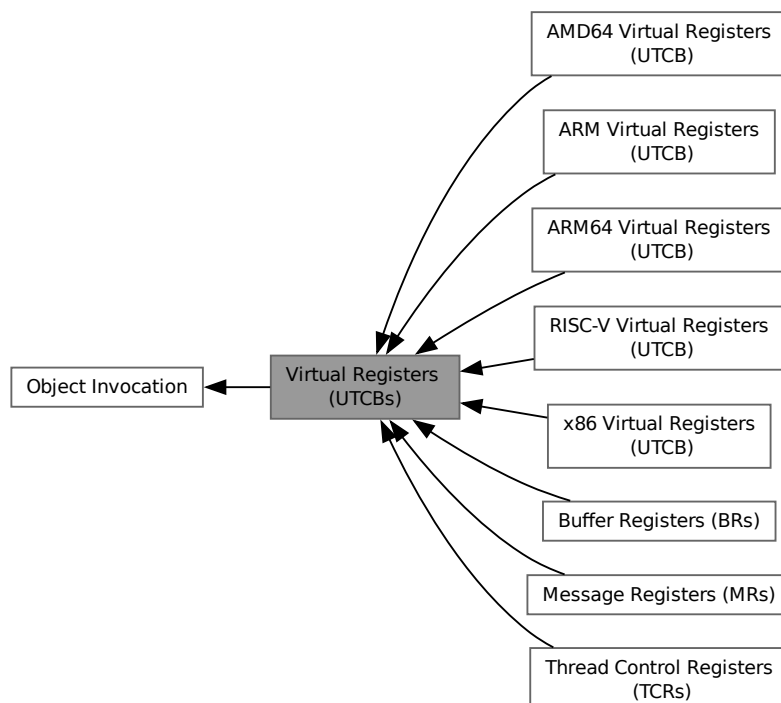
Here is the caller graph for this function:



13.1.9.10 Virtual Registers (UTCBS)

[L4 Virtual Registers \(UTCBS\)](#).

Collaboration diagram for Virtual Registers (UTCBS):



Topics

- [Message Registers \(MRs\)](#) 269
- [Buffer Registers \(BRs\)](#) 273
- [Thread Control Registers \(TCRs\)](#) 275
- [ARM Virtual Registers \(UTCB\)](#) 276
- [ARM64 Virtual Registers \(UTCB\)](#) 276
- [AMD64 Virtual Registers \(UTCB\)](#) 277
- [x86 Virtual Registers \(UTCB\)](#) 277
- [RISC-V Virtual Registers \(UTCB\)](#) 278

Files

- file [utcb.h](#)
UTCB definitions for ARM.
- file [utcb.h](#)
UTCB definitions for ARM64.
- file [utcb.h](#)
UTCB definitions for AMD64.
- file [utcb.h](#)
UTCB definitions for x86.
- file [utcb.h](#)
UTCB definitions for RISC-V.

Typedefs

- typedef struct [l4_utcb_t](#) [l4_utcb_t](#)
Opaque type for the UTCB.

Functions

- [l4_utcb_t](#) * [l4_utcb](#) (void) [L4_NOTHROW](#) [L4_PURE](#)
Get the UTCB address.
- [l4_msg_regs_t](#) * [l4_utcb_mr](#) (void) [L4_NOTHROW](#) [L4_PURE](#)
Get the message-register block of a UTCB.
- [l4_buf_regs_t](#) * [l4_utcb_br](#) (void) [L4_NOTHROW](#) [L4_PURE](#)
Get the buffer-register block of a UTCB.
- [l4_thread_regs_t](#) * [l4_utcb_tcr](#) (void) [L4_NOTHROW](#) [L4_PURE](#)
Get the thread-control-register block of a UTCB.

13.1.9.10.1 Detailed Description

[L4](#) Virtual Registers (UTCB).

Include File

```
#include <l4/sys/utcb.h>
```

The virtual registers are part of the micro-kernel API and are located in the user-level thread control block (UTCB). The UTCB is a data structure defined by the micro kernel and located on kernel-provided memory. Each [L4](#) thread gets a unique UTCB assigned when it is bound to a task (see [Thread Control](#) , [l4_thread_control_bind\(\)](#) for more information).

The UTCB is arranged in three blocks of virtual registers.

- [Thread Control Registers \(TCRs\)](#)
- [Message Registers \(MRs\)](#)
- [Buffer Registers \(BRs\)](#)

To access the contents of the virtual registers the [l4_utcb_mr\(\)](#), [l4_utcb_tcr\(\)](#), and [l4_utcb_br\(\)](#) functions must be used.

13.1.9.10.2 Typedef Documentation

13.1.9.10.2.1 [l4_utcb_t](#)

```
typedef struct l4_utcb_t l4_utcb_t
```

Opaque type for the UTCB.

To access the contents of the virtual registers the [l4_utcb_mr\(\)](#), [l4_utcb_tcr\(\)](#), and [l4_utcb_br\(\)](#) functions must be used.

Examples

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 56 of file [utcb.h](#).

13.1.9.10.3 Function Documentation

13.1.9.10.3.1 l4_utcb_br()

```
l4_buf_regs_t * l4_utcb_br (  
    void )    [inline]
```

Get the buffer-register block of a UTCB.

Returns

A pointer to the buffer-register block of `u`.

Definition at line 384 of file `utcb.h`.

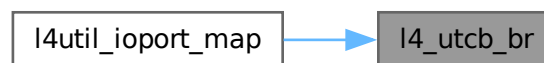
References [L4_INLINE](#), [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Referenced by [l4util_ioport_map\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.1.9.10.3.2 l4_utcb_mr()

```
l4_msg_regs_t * l4_utcb_mr (  
    void )    [inline]
```

Get the message-register block of a UTCB.

Returns

A pointer to the message-register block of `u`.

Examples

[examples/sys/aliens/main.c](#), [examples/sys/ipc/ipc_example.c](#), [examples/sys/singlestep/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 381 of file [utcb.h](#).

References [L4_INLINE](#), [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Referenced by [l4util_ioport_map\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.1.9.10.3.3 l4_utcb_tcr()

```
l4_thread_regs_t * l4_utcb_tcr (  
    void ) [inline]
```

Get the thread-control-register block of a UTCB.

Returns

A pointer to the thread-control-register block of `u`.

Definition at line 387 of file `utcb.h`.

References [L4_INLINE](#), [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:

**13.1.9.10.4 Message Registers (MRs)**

Collaboration diagram for Message Registers (MRs):

**Topics**

- [Exception registers](#) 270
Overlay definition of the MRs for exception messages.

Data Structures

- union [l4_msg_regs_t](#)
Encapsulation of the message-register block in the UTCB.

Typedefs

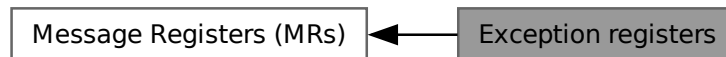
- typedef union `l4_msg_regs_t` **`l4_msg_regs_t`**
Encapsulation of the message-register block in the UTCB.

13.1.9.10.4.1 Detailed Description

13.1.9.10.4.2 Exception registers

Overlay definition of the MRs for exception messages.

Collaboration diagram for Exception registers:



Functions

- `l4_umword_t l4_utcb_exc_pc (l4_exc_regs_t const *u) L4_NOTHROW L4_PURE`
Access function to get the program counter of the exception state.
- `void l4_utcb_exc_pc_set (l4_exc_regs_t *u, l4_addr_t pc) L4_NOTHROW`
Set the program counter register in the exception state.
- `unsigned long l4_utcb_exc_typeval (l4_exc_regs_t const *u) L4_NOTHROW L4_PURE`
Get the value out of an exception UTCTB that describes the type of exception.
- `int l4_utcb_exc_is_pf (l4_exc_regs_t const *u) L4_NOTHROW L4_PURE`
Check whether an exception IPC is a page fault.
- `l4_addr_t l4_utcb_exc_pfa (l4_exc_regs_t const *u) L4_NOTHROW L4_PURE`
Function to get the L4 style page fault address out of an exception.
- `int l4_utcb_exc_is_ex_regs_exception (l4_exc_regs_t const *u) L4_NOTHROW L4_PURE`
Check whether an exception IPC was triggered via `l4_thread_ex_regs()`.
- `l4_exc_regs_t * l4_utcb_exc (void) L4_NOTHROW L4_PURE`
Get the message-register block of a UTCTB (for an exception IPC).

Detailed Description

Overlay definition of the MRs for exception messages.

Function Documentation

`l4_utcb_exc()`

```
l4_exc_regs_t * l4_utcb_exc (
    void ) [inline]
```

Get the message-register block of a UTCTB (for an exception IPC).

Returns

A pointer to the exception message in `u`.

Examples

[examples/sys/aliens/main.c](#), and [examples/sys/singlestep/main.c](#).

Definition at line 390 of file `utcb.h`.

References [L4_INLINE](#), [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:

**`l4_utcb_exc_is_ex_regs_exception()`**

```
int l4_utcb_exc_is_ex_regs_exception (
    l4_exc_regs_t const * u) [inline]
```

Check whether an exception IPC was triggered via [l4_thread_ex_regs\(\)](#).

Return values

| | |
|------------------|--|
| <code>0</code> | Exception was not triggered through <code>ex_regs</code> . |
| <code>!=0</code> | Exception was triggered through <code>ex_regs</code> . |

This function checks if the exception was emitted by using the `L4_THREAD_EX_REGS_TRIGGER_EXCEPTION` flag in an [l4_thread_ex_regs\(\)](#) call.

Definition at line 100 of file `utcb.h`.

References [L4_INLINE](#), [L4_NOTHROW](#), and [l4_utcb_exc_typeval\(\)](#).

Here is the call graph for this function:



l4_utcb_exc_is_pf()

```
int l4_utcb_exc_is_pf (
    l4_exc_regs_t const * u) [inline]
```

Check whether an exception IPC is a page fault.

Returns

0 if not, != 0 if yes

Function to check whether an exception IPC is a page fault, also applies to I/O pagefaults.

Definition at line 90 of file [utcb.h](#).

References [L4_INLINE](#), and [L4_NOTHROW](#).

l4_utcb_exc_pc()

```
l4_umword_t l4_utcb_exc_pc (
    l4_exc_regs_t const * u) [inline]
```

Access function to get the program counter of the exception state.

Parameters

| | |
|----------|------|
| <i>u</i> | UTCB |
|----------|------|

Returns

The program counter register out of the exception state.

Examples

[examples/sys/aliens/main.c](#), and [examples/sys/singlestep/main.c](#).

Definition at line 75 of file [utcb.h](#).

References [L4_INLINE](#), and [L4_NOTHROW](#).

l4_utcb_exc_pc_set()

```
void l4_utcb_exc_pc_set (
    l4_exc_regs_t * u,
    l4_addr_t pc) [inline]
```

Set the program counter register in the exception state.

Parameters

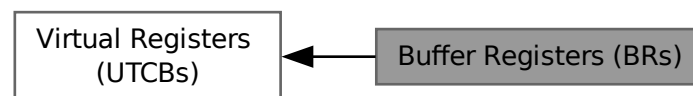
| | |
|-----------|-----------------------------|
| <i>u</i> | UTCB |
| <i>pc</i> | The program counter to set. |

Definition at line 80 of file [utcb.h](#).

References [L4_INLINE](#), and [L4_NOTHROW](#).

13.1.9.10.5 Buffer Registers (BRs)

Collaboration diagram for Buffer Registers (BRs):

**Data Structures**

- struct [l4_buf_regs_t](#)
Encapsulation of the buffer-registers block in the UTCB.

Typedefs

- typedef struct [l4_buf_regs_t](#) **l4_buf_regs_t**
Encapsulation of the buffer-registers block in the UTCB.

Enumerations

- enum [l4_buffer_desc_consts_t](#) { [L4_BDR_MEM_SHIFT](#) = 0 , [L4_BDR_IO_SHIFT](#) = 5 , [L4_BDR_OBJ_SHIFT](#) = 10 , [L4_BDR_OFFSET_MASK](#) = (1UL << 20) - 1 }
- Constants for buffer descriptors.*

Functions

- void **l4_utcb_inherit_fpu** (int switch_on) [L4_NOTHROW](#)
Enable or disable inheritance of FPU state to receiver.
- [l4_umword_t](#) **l4_bdr** ([l4_umword_t](#) mem, [l4_umword_t](#) io, [l4_umword_t](#) obj, [l4_umword_t](#) flags) [L4_NOTHROW](#)
Create a buffer descriptor.

13.1.9.10.5.1 Detailed Description

13.1.9.10.5.2 Enumeration Type Documentation

l4_buffer_desc_consts_t

enum [l4_buffer_desc_consts_t](#)

Constants for buffer descriptors.

Enumerator

| | |
|------------------|---|
| L4_BDR_MEM_SHIFT | Bit offset for the memory-buffer index. |
| L4_BDR_IO_SHIFT | Bit offset for the IO-buffer index. |
| L4_BDR_OBJ_SHIFT | Bit offset for the capability-buffer index. |

Definition at line [322](#) of file [consts.h](#).

13.1.9.10.5.3 Function Documentation

l4_bdr()

```
l4\_umword\_t l4_bdr (
    l4\_umword\_t mem,
    l4\_umword\_t io,
    l4\_umword\_t obj,
    l4\_umword\_t flags) [inline]
```

Create a buffer descriptor.

Parameters

| | |
|--------------|--|
| <i>mem</i> | Index of the first memory receive buffer. |
| <i>io</i> | Index of the first IO-port receive buffer. |
| <i>obj</i> | Index of the first object receive buffer. |
| <i>flags</i> | Flags (only L4_UTCB_INHERIT_FPU so far). |

Definition at line [362](#) of file [utcb.h](#).

References [L4_INLINE](#), and [L4_NOTHROW](#).

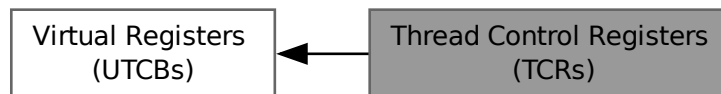
Referenced by [L4Re::Util::Br_manager::setup_wait\(\)](#).

Here is the caller graph for this function:



13.1.9.10.6 Thread Control Registers (TCRs)

Collaboration diagram for Thread Control Registers (TCRs):



Data Structures

- struct [l4_thread_regs_t](#)
Encapsulation of the thread-control-register block of the UTCB.

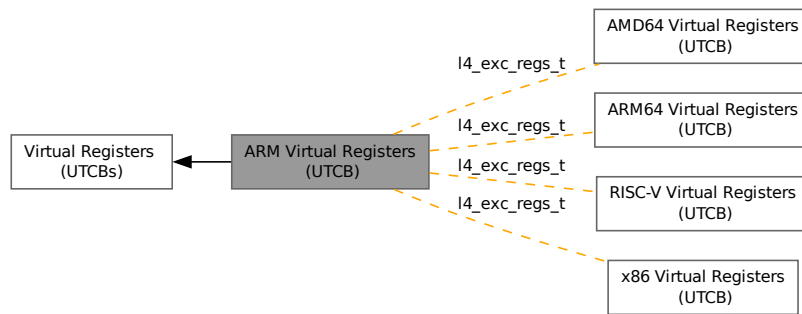
Typedefs

- typedef struct l4_thread_regs_t **l4_thread_regs_t**
Encapsulation of the thread-control-register block of the UTCB.

13.1.9.10.6.1 Detailed Description

13.1.9.10.7 ARM Virtual Registers (UTCB)

Collaboration diagram for ARM Virtual Registers (UTCB):



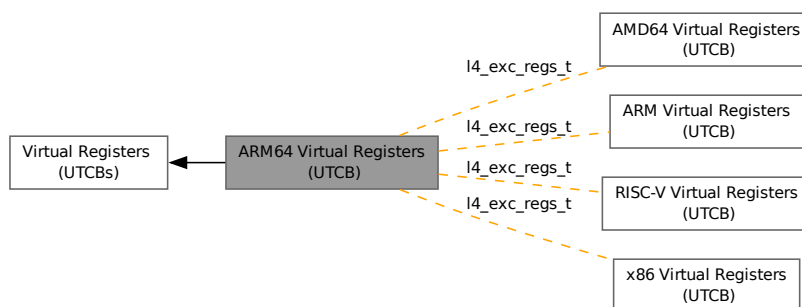
Data Structures

- struct [l4_exc_regs_t](#)
UTCB structure for exceptions.

13.1.9.10.7.1 Detailed Description

13.1.9.10.8 ARM64 Virtual Registers (UTCB)

Collaboration diagram for ARM64 Virtual Registers (UTCB):



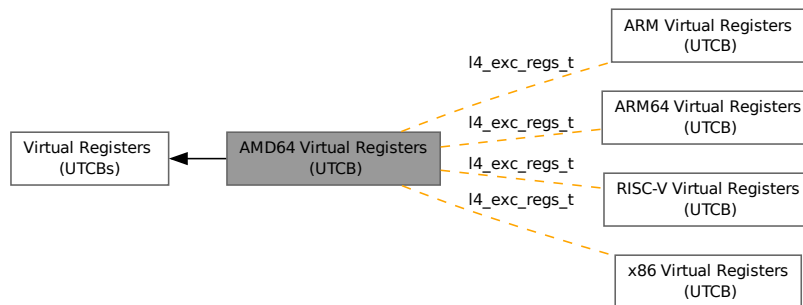
Data Structures

- struct [l4_exc_regs_t](#)
UTCB structure for exceptions.

13.1.9.10.8.1 Detailed Description

13.1.9.10.9 AMD64 Virtual Registers (UTCB)

Collaboration diagram for AMD64 Virtual Registers (UTCB):



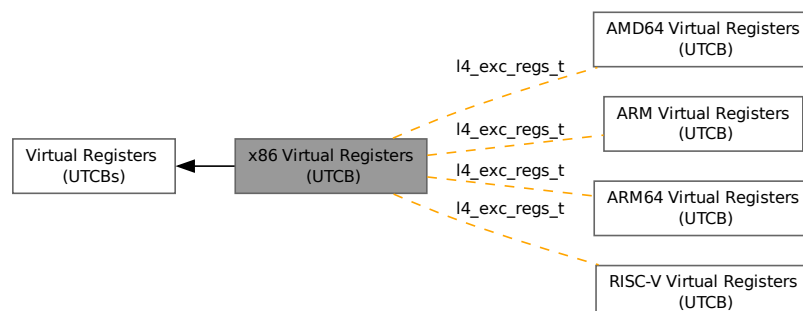
Data Structures

- struct [l4_exc_regs_t](#)
UTCB structure for exceptions.

13.1.9.10.9.1 Detailed Description

13.1.9.10.10 x86 Virtual Registers (UTCB)

Collaboration diagram for x86 Virtual Registers (UTCB):



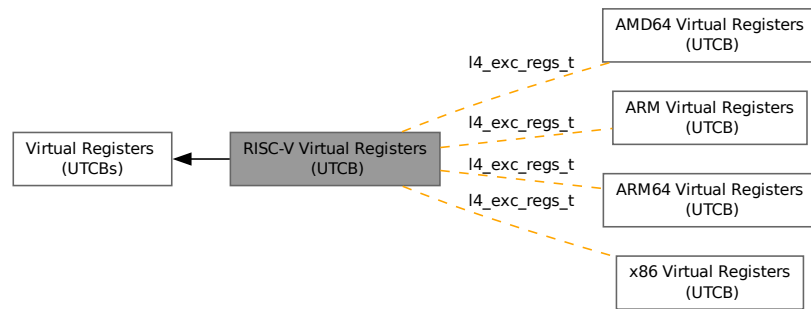
Data Structures

- struct [l4_exc_regs_t](#)
UTCB structure for exceptions.

13.1.9.10.10.1 Detailed Description

13.1.9.10.11 RISC-V Virtual Registers (UTCB)

Collaboration diagram for RISC-V Virtual Registers (UTCB):



Data Structures

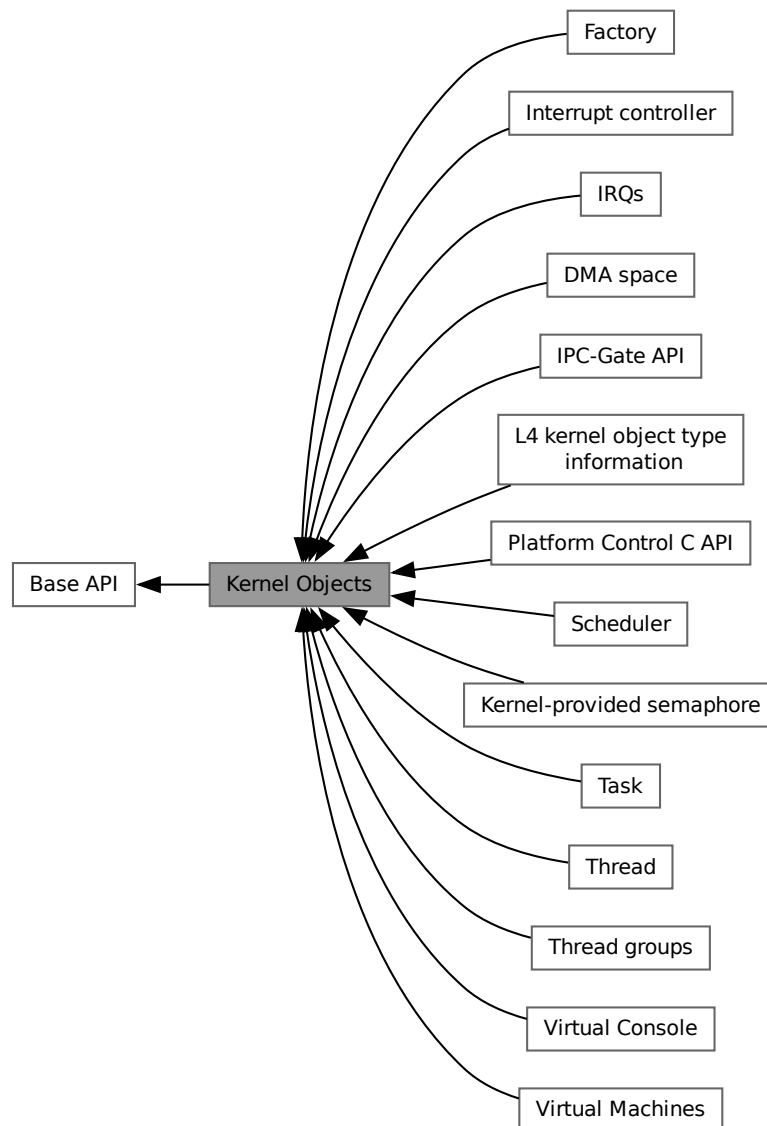
- struct [l4_exc_regs_t](#)
UTCB structure for exceptions.

13.1.9.10.11.1 Detailed Description

13.1.10 Kernel Objects

API of kernel objects.

Collaboration diagram for Kernel Objects:



Topics

- IPC-Gate API 281
The C IPC gate interface, see [L4::ipc_gate](#) for the C++ interface.
- DMA space 286
A DMA space represents a device memory address space managed by an IOMMU.
- L4 kernel object type information 286
Type information for [L4](#) server objects that can be called via IPC.
-

| | |
|---|-----|
| Factory• | 288 |
| <i>C factory interface to create objects, see L4::Factory for the C++ interface.</i> | |
| • | |
| Virtual Machines | 300 |
| <i>Virtual Machine API.</i> | |
| • | |
| Interrupt controller | 321 |
| <i>The C lcu interface, see L4::lcu for the C++ interface.</i> | |
| • | |
| IRQs • | 338 |
| <i>C IRQ interface, see L4::Irq for the C++ interface.</i> | |
| • | |
| Platform Control C API | 353 |
| <i>C interface for controlling platform-wide properties, see L4::Platform_control for the C++ interface.</i> | |
| • | |
| Scheduler | 359 |
| <i>C interface of the Scheduler kernel object, see L4::Scheduler for the C++ interface.</i> | |
| • | |
| Kernel-provided semaphore | 367 |
| <i>C semaphore interface, see L4::Semaphore for the C++ interface.</i> | |
| • | |
| Task • | 369 |
| <i>C interface of the Task kernel object, see L4::Task for the C++ interface.</i> | |
| • | |
| Thread• | 382 |
| <i>C Thread object interface, see L4::Thread for the C++ interface.</i> | |
| • | |
| Thread groups | 416 |
| <i>C thread group interface, see L4::Thread_group for the C++ interface.</i> | |
| • | |
| Virtual Console | 418 |
| <i>C Virtual console interface for simple character based input and output, see L4::Vcon for the C++ interface.</i> | |

Data Structures

- class [L4::Kobject](#)
Base class for all kinds of kernel objects and remote objects, referenced by capabilities.
- class [L4::Vm](#)
Virtual machine host address space.

13.1.10.1 Detailed Description

API of kernel objects.

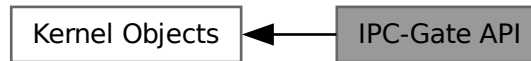
Include File

```
#include <l4/sys/kernel_object.h>
```

13.1.10.2 IPC-Gate API

The C IPC gate interface, see [L4::ipc_gate](#) for the C++ interface.

Collaboration diagram for IPC-Gate API:



Functions

- [l4_msgtag_t l4_ipc_gate_get_infos](#) ([l4_cap_idx_t](#) gate, [l4_umword_t](#) *label)
Get information about the IPC-gate.
- [l4_msgtag_t l4_rcv_ep_bind_thread](#) ([l4_cap_idx_t](#) ep, [l4_cap_idx_t](#) thread, [l4_umword_t](#) label)
Bind the IPC receive endpoint to a thread.
- [l4_msgtag_t l4_rcv_ep_bind_snd_destination](#) ([l4_cap_idx_t](#) ep, [l4_cap_idx_t](#) snd_dst, [l4_umword_t](#) label)
Bind the IPC receive endpoint to a send destination (a thread).

13.1.10.2.1 Detailed Description

The C IPC gate interface, see [L4::ipc_gate](#) for the C++ interface.

IPC gates are used to create secure communication channels between protection domains. An IPC gate can be created using the [Factory](#) interface.

Depending on the permissions of the capability used, an IPC gate forwards IPC to the [Thread](#) the IPC gate is *bound* to (cf. [l4_rcv_ep_bind_thread\(\)](#) and [l4_rcv_ep_bind_snd_destination\(\)](#)). If the capability has the [L4_FPAGE_C_IPCGATE_SVR](#) permission, only IPC using a protocol different from the [L4_PROTO_KOBJECT](#) protocol is forwarded. Without the [L4_FPAGE_C_IPCGATE_SVR](#) permission, all IPC is forwarded. The latter is the usual case for a client in a client/server scenario. When not bound to a thread or thread group yet, the forwarded IPC blocks until the IPC gate is bound to a thread or thread group, or the IPC times out.

Forwarded IPC is always forwarded to the userland of the thread the IPC gate is bound to, either directly or indirectly using a thread group. That means, the [Thread](#) interface of that thread is not accessible via an IPC gate. The [IPC-Gate API](#) of an IPC gate is only accessible if the capability used has the [L4_FPAGE_C_IPCGATE_SVR](#) permission (cf. previous paragraph). Conversely that means, if the capability used lacks the [L4_FPAGE_C_IPCGATE_SVR](#) permission, [IPC-Gate API](#) calls are forwarded to the thread or thread group the IPC gate is bound to instead of being processed by the IPC gate itself. In a client/server scenario, a client should only get IPC gate capabilities without [L4_FPAGE_C_IPCGATE_SVR](#) permission so the client cannot tamper with the IPC gate.

When binding an IPC gate to a thread or thread group, a user-defined, kernel protected, machine-word sized payload called the IPC gate's *label* is assigned to the IPC gate (note that the two least significant bits of the label must be zero; cf. [l4_rcv_ep_bind_thread\(\)](#) and [l4_rcv_ep_bind_snd_destination\(\)](#)). When a send-only IPC or call IPC is forwarded via an IPC gate, the label provided by the sender is ignored and replaced by the IPC gate's label where the two least significant bits are set to the [L4_CAP_FPAGE_S](#) and [L4_CAP_FPAGE_W](#) permissions of the capability used. The replaced label is only visible to the thread the IPC gate is bound to upon receive (or to the

selected thread from the thread group the IPC gate is bound to). However, the configured label of an IPC gate can also be queried via [l4_ipc_gate_get_infos\(\)](#) if the capability used has the [L4_FPAGE_C_IPCGATE_SVR](#) permission.

When deleting an IPC gate or when unbinding it from a thread or thread group, the label of IPC already in flight won't be changed. To ensure that no IPC from this IPC gate is received by a thread with an unexpected label, [l4_thread_modify_sender_start\(\)](#) shall be used to change the labels of every pending IPC to that gate. This is also required if the label of an already bound IPC gate is changed. It is not necessary after binding the IPC gate to a thread or thread group for the first time.

When binding a currently bound IPC gate to a new thread or thread group, the same label should be used that was used with the old thread. Otherwise the old and the new thread need to synchronize to avoid IPC messages with unexpected labels.

Include File

```
#include <l4/sys/ipc_gate.h>
```

For the C++ interface refer to the [L4::lpc_gate](#) documentation.

See also

[Object Invocation](#)

13.1.10.2.2 Function Documentation

13.1.10.2.2.1 l4_ipc_gate_get_infos()

```
l4_msgtag_t l4_ipc_gate_get_infos (
    l4_cap_idx_t gate,
    l4_umword_t * label) [inline]
```

Get information about the IPC-gate.

Parameters

| | | |
|-----|--------------|---|
| | <i>gate</i> | The IPC gate object to get information about. |
| out | <i>label</i> | The label of the IPC gate is returned here. |

Returns

System call return tag.

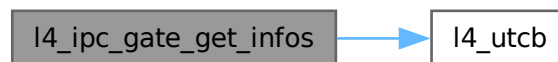
Precondition

If `gate` does not possess the [L4_FPAGE_C_IPCGATE_SVR](#) right, the kernel will not perform this operation. Instead, the underlying IPC message will be forwarded to the thread or thread group the IPC gate is bound to, blocking the caller if the IPC gate is not bound yet.

Definition at line 166 of file [ipc_gate.h](#).

References [l4_utcb\(\)](#).

Here is the call graph for this function:

**13.1.10.2.2.2 l4_rcv_ep_bind_snd_destination()**

```

l4_msgtag_t l4_rcv_ep_bind_snd_destination (
    l4_cap_idx_t ep,
    l4_cap_idx_t snd_dst,
    l4_umword_t label) [inline]
  
```

Bind the IPC receive endpoint to a send destination (a thread).

Parameters

| | |
|----------------|---|
| <i>ep</i> | The IPC receive endpoint object. |
| <i>snd_dst</i> | The send destination (thread) object <i>ep</i> shall be bound to. |
| <i>label</i> | Label to assign to <i>ep</i> . For IPC gates, the two least significant bits must be set to zero. |

Returns

Syscall return tag containing one of the following return codes.

Return values

| | |
|-------------------|--|
| <i>L4_EOK</i> | Operation successful. |
| <i>-L4_EINVAL</i> | <i>snd_dst</i> is not a thread object or other arguments were malformed. |
| <i>-L4_EPERM</i> | No L4_CAP_FPAGE_S right on <i>ep</i> or <i>snd_dst</i> . |

Precondition

If `ep` is an IPC gate capability without the `L4_FPAGE_C_IPCGATE_SVR` right, the kernel will not perform this operation. Instead, the underlying IPC message will be forwarded to the send destination the IPC gate is bound to, blocking the caller if the IPC gate was not bound yet.

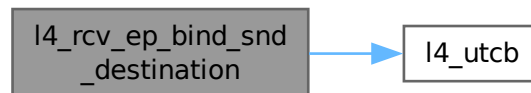
The specified `label` is passed to the receiver of the incoming IPC. It is possible to re-bind a receive endpoint to the same or a different send destination. In this case, IPC already in flight will be delivered with the old label to the previously bound thread unless `l4_thread_modify_sender_start()` is used to change these labels.

Definition at line 138 of file `rcv_endpoint.h`.

References `l4_utcb()`.

Referenced by `L4::Rcv_endpoint::bind_snd_destination()`.

Here is the call graph for this function:



Here is the caller graph for this function:

**13.1.10.2.2.3 l4_rcv_ep_bind_thread()**

```

l4_msgtag_t l4_rcv_ep_bind_thread (
    l4_cap_idx_t ep,
    l4_cap_idx_t thread,
    l4_umword_t label) [inline]
  
```

Bind the IPC receive endpoint to a thread.

Parameters

| | |
|---------------|---|
| <i>ep</i> | The IPC receive endpoint object. |
| <i>thread</i> | The thread object <i>ep</i> shall be bound to. |
| <i>label</i> | Label to assign to <i>ep</i> . For IPC gates, the two least significant bits must be set to zero. |

Returns

Syscall return tag containing one of the following return codes.

Return values

| | |
|-------------------|---|
| <i>L4_EOK</i> | Operation successful. |
| <i>-L4_EINVAL</i> | <i>thread</i> is not a thread object or other arguments were malformed. |
| <i>-L4_EPERM</i> | Insufficient permissions; see precondition. |

Precondition

The capabilities *ep* and *thread* both must have the permission [L4_CAP_FPAGE_S](#).

If *ep* is an IPC gate capability without the [L4_FPAGE_C_IPCGATE_SVR](#) right, the kernel will not perform this operation. Instead, the underlying IPC message will be forwarded to the thread the IPC gate is bound to, blocking the caller if the IPC gate was not bound yet.

The specified *label* is passed to the receiver of the incoming IPC. It is possible to re-bind a receive endpoint to the same or a different thread. In this case, IPC already in flight will be delivered with the old label to the previously bound thread unless [l4_thread_modify_sender_start\(\)](#) is used to change these labels.

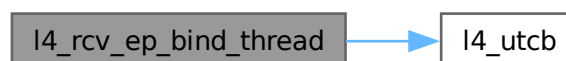
Examples

[examples/sys/isr/main.c](#).

Definition at line 131 of file [rcv_endpoint.h](#).

References [l4_utcb\(\)](#).

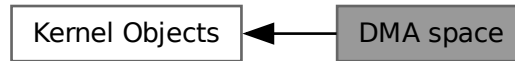
Here is the call graph for this function:



13.1.10.3 DMA space

A DMA space represents a device memory address space managed by an IOMMU.

Collaboration diagram for DMA space:



A DMA space represents a device memory address space managed by an IOMMU.

That is, it manages the translation of virtual addresses used by devices to physical addresses. It is accessed via the [L4::Task](#) interface, but with the following caveats:

- No threads can be bound to it.
- No objects (and IO ports on IA32) can be mapped to it.
- No kernel-user memory can be added to it.
- It must be constructed by passing the `L4_PROTO_DMA_SPACE` protocol constant to the kernel factory's [L4::Factory.create\(\)](#) call.

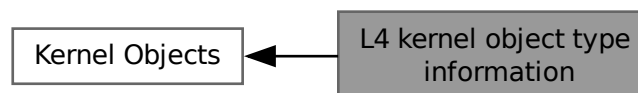
A DMA space must be bound to an [L4::iommu](#) to enable the address translation for specific devices.

The kernel factory allows to create DMA spaces only if the kernel has been configured with IOMMU support and if an IOMMU was detected.

13.1.10.4 L4 kernel object type information

Type information for [L4](#) server objects that can be called via IPC.

Collaboration diagram for L4 kernel object type information:



Data Structures

- struct [L4::Type_info](#)
Dynamic Type Information for [L4Re](#) Interfaces.
- struct [L4::Kobject_typeid](#)< T >
Meta object for handling access to type information of Kobjects.
- class [L4::Kobject_t](#)< Derived, Base, PROTO, S_DEMAND >
Helper class to create an [L4Re](#) interface class that is derived from a single base class.
- class [L4::Kobject_2t](#)< Derived, Base1, Base2, PROTO, S_DEMAND >
Helper class to create an [L4Re](#) interface class that is derived from two base classes (see [L4::Kobject_t](#)).
- struct [L4::Kobject_3t](#)< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >
Helper class to create an [L4Re](#) interface class that is derived from three base classes (see [L4::Kobject_t](#)).
- struct [L4::Kobject_x](#)< Derived, ARGS >
Generic [Kobject](#) inheritance template.

Functions

- template<typename T>
[Type_info](#) const * [L4::kobject_typeid](#) () noexcept
Get the [L4::Type_info](#) for the [L4Re](#) interface given in T.

13.1.10.4.1 Detailed Description

Type information for [L4](#) server objects that can be called via IPC.

This type information consists of inheritance information, the protocol number assigned to an interface as well as the demand on server-side resources.

13.1.10.4.2 Function Documentation

13.1.10.4.2.1 kobject_typeid()

```
template<typename T>
Type\_info const * L4::kobject\_typeid () [inline], [noexcept]
```

Get the [L4::Type_info](#) for the [L4Re](#) interface given in T.

Template Parameters

| | |
|-------------------|---|
| T | The type (L4Re interface) for which the information shall be returned. |
|-------------------|---|

Returns

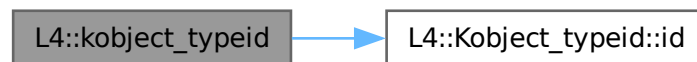
A pointer to the [L4::Type_info](#) structure for T.

Definition at line 682 of file [__typeinfo.h](#).

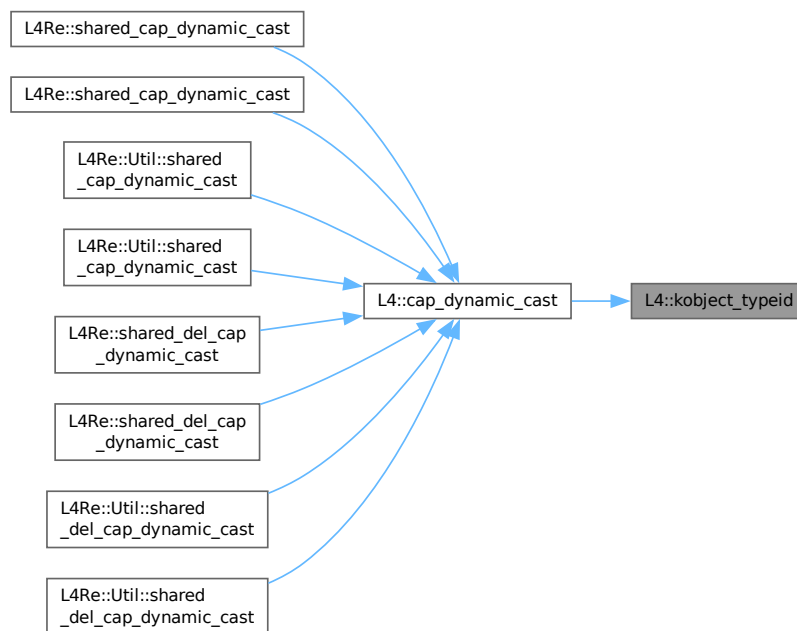
References [L4::Kobject_typeid< T >::id\(\)](#).

Referenced by [cap_dynamic_cast\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**13.1.10.5 Factory**

C factory interface to create objects, see [L4::Factory](#) for the C++ interface.

Collaboration diagram for Factory:



Functions

- [l4_msgtag_t l4_factory_create_task](#) ([l4_cap_idx_t](#) factory, [l4_cap_idx_t](#) target_cap, [l4_fpage_t](#) *utcb_area) [L4_NOTHROW](#)
Create a new task.
- [l4_msgtag_t l4_factory_create_thread](#) ([l4_cap_idx_t](#) factory, [l4_cap_idx_t](#) target_cap) [L4_NOTHROW](#)
Create a new thread.
- [l4_msgtag_t l4_factory_create_factory](#) ([l4_cap_idx_t](#) factory, [l4_cap_idx_t](#) target_cap, unsigned long limit) [L4_NOTHROW](#)
Create a new factory.
- [l4_msgtag_t l4_factory_create_gate](#) ([l4_cap_idx_t](#) factory, [l4_cap_idx_t](#) target_cap, [l4_cap_idx_t](#) snd_dst_↔ cap, [l4_umword_t](#) label) [L4_NOTHROW](#)
Create a new IPC gate, optionally bound to a send destination (a thread or thread group).
- [l4_msgtag_t l4_factory_create_irq](#) ([l4_cap_idx_t](#) factory, [l4_cap_idx_t](#) target_cap) [L4_NOTHROW](#)
Create a new IRQ sender.
- [l4_msgtag_t l4_factory_create_vm](#) ([l4_cap_idx_t](#) factory, [l4_cap_idx_t](#) target_cap) [L4_NOTHROW](#)
Create a new virtual machine.
- [l4_msgtag_t l4_factory_create_vcpu_context](#) ([l4_cap_idx_t](#) factory, [l4_cap_idx_t](#) target_cap) [L4_NOTHROW](#)
Create a new vCPU context.
- [l4_msgtag_t l4_factory_create_thread_group](#) ([l4_cap_idx_t](#) factory, [l4_cap_idx_t](#) target_cap, unsigned policy) [L4_NOTHROW](#)
Create a new thread group.
- [l4_msgtag_t l4_factory_create](#) ([l4_cap_idx_t](#) factory, long obj, [l4_cap_idx_t](#) target) [L4_NOTHROW](#)
Create a new object.

13.1.10.5.1 Detailed Description

C factory interface to create objects, see [L4::Factory](#) for the C++ interface.

A factory is used to create all kinds of kernel objects:

- [Task](#)
- [Thread](#)
- [Factory](#)
- [IPC-Gate API](#)
- [IRQs](#)

- [Virtual Machines](#)

To create a new kernel object the caller has to specify the factory to use for creation. The caller has to allocate a capability slot where the kernel stores the new object's capability.

The factory is equipped with a limit that limits the amount of kernel memory available for that factory.

Note

The limit does not give any guarantee for the amount of available kernel memory.

Include File

```
#include <l4/sys/factory.h>
```

For the C++ interface refer to [L4::Factory](#).

13.1.10.5.2 Function Documentation

13.1.10.5.2.1 l4_factory_create()

```
l4_msgtag_t l4_factory_create (
    l4_cap_idx_t factory,
    long obj,
    l4_cap_idx_t target) [inline]
```

Create a new object.

Parameters

| | | |
|-----|----------------|---|
| | <i>factory</i> | Factory to use for creation. |
| | <i>obj</i> | Protocol ID to describe the type of the object to create. |
| out | <i>target</i> | The kernel stores the new object's capability into this slot. |

Return values

| | |
|------------------|---|
| <i>L4_EOK</i> | No error occurred. |
| <i>-L4_EPERM</i> | Insufficient permissions; see precondition. |
| <i><0</i> | Error code. |

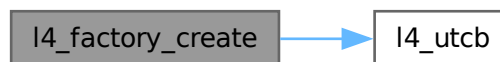
Precondition

The capability `factory` must have the permission [L4_CAP_FPAGE_S](#).

Definition at line 707 of file [factory.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:

**13.1.10.5.2.2 l4_factory_create_factory()**

```

l4_msgtag_t l4_factory_create_factory (
    l4_cap_idx_t factory,
    l4_cap_idx_t target_cap,
    unsigned long limit) [inline]
  
```

Create a new factory.

Parameters

| | | |
|-----|-------------------|--|
| | <i>factory</i> | Capability selector for factory to use for creation. |
| out | <i>target_cap</i> | The kernel stores the new factory's capability into this slot. |
| | <i>limit</i> | Limit for the new factory in bytes. |

Returns

Syscall return tag

Return values

| | |
|------------------|---|
| <i>L4_EOK</i> | No error occurred. |
| <i>-L4_EPERM</i> | Insufficient permissions; see precondition. |
| <i><0</i> | Error code. |

Precondition

The capability `factory` must have the permission [L4_CAP_FPAGE_S](#).

Note

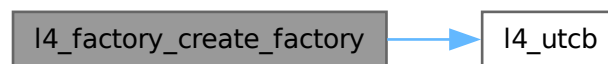
The limit of the new factory is subtracted from the available amount of the factory used for creation.

This method is only guaranteed to work with the [Kernel Factory](#). For other services, use the generic [L4::Factory::create\(\)](#) method and consult the service documentation for information on the arguments that need to be passed to the create stream.

Definition at line 545 of file [factory.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:

**13.1.10.5.2.3 l4_factory_create_gate()**

```

l4_msgtag_t l4_factory_create_gate (
    l4_cap_idx_t factory,
    l4_cap_idx_t target_cap,
    l4_cap_idx_t snd_dst_cap,
    l4_umword_t label) [inline]
  
```

Create a new IPC gate, optionally bound to a send destination (a thread or thread group).

Parameters

| | | |
|-----|--------------------|---|
| | <i>factory</i> | Capability selector for factory to use for creation. |
| out | <i>target_cap</i> | The kernel stores the new IPC gate's capability into this slot. |
| | <i>snd_dst_cap</i> | Optional capability selector of a thread or thread group to bind the gate to. Use L4_INVALID_CAP to create an unbound IPC gate. |
| | <i>label</i> | Optional label of the gate (precisely used if <i>snd_dst_cap</i> is valid). If <i>snd_dst_cap</i> is valid, <i>label</i> must be present. |

Returns

Syscall return tag containing one of the following return codes.

Return values

| | |
|---------------|--------------------|
| <i>L4_EOK</i> | No error occurred. |
|---------------|--------------------|

| | |
|------------|---|
| -L4_ENOMEM | Out-of-memory during allocation of the <code>lpc_gate</code> object. |
| -L4_EINVAL | <code>snd_dst_cap</code> is void or points to something that is not a thread or thread group. |
| -L4_EPERM | Insufficient permissions; see precondition. |

Precondition

The capability `factory` must have the permission `L4_CAP_FPAGE_S`. Also `snd_dst_cap` (if not `L4_INVALID_CAP`) must have the permission `L4_CAP_FPAGE_S`.

An unbound IPC gate can be bound to a thread using `l4_rcv_ep_bind_thread()`.

See also

[IPC-Gate API](#)

Definition at line 553 of file `factory.h`.

References [L4_NOTHROW](#), and `l4_utcb()`.

Here is the call graph for this function:

**13.1.10.5.2.4 l4_factory_create_irq()**

```

l4_msgtag_t l4_factory_create_irq (
    l4_cap_idx_t factory,
    l4_cap_idx_t target_cap) [inline]
  
```

Create a new IRQ sender.

Parameters

| | | |
|-----|-------------------|--|
| | <i>factory</i> | Factory to use for creation. |
| out | <i>target_cap</i> | The kernel stores the new IRQ's capability into this slot. |

Return values

| | |
|------------------|---|
| <i>L4_EOK</i> | No error occurred. |
| <i>-L4_EPERM</i> | Insufficient permissions; see precondition. |
| <i><0</i> | Error code. |

Precondition

The capability *factory* must have the permission [L4_CAP_FPAGE_S](#).

See also

[IRQs](#)

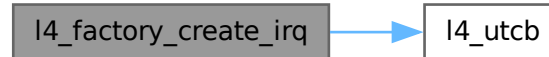
Examples

[examples/sys/isr/main.c](#).

Definition at line [561](#) of file [factory.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



13.1.10.5.2.5 l4_factory_create_task()

```

l4_msgtag_t l4_factory_create_task (
    l4_cap_idx_t factory,
    l4_cap_idx_t target_cap,
    l4_fpage_t * utcb_area) [inline]
  
```

Create a new task.

Parameters

| | | |
|---------|-------------------|--|
| | <i>factory</i> | Capability selector for factory to use for creation. |
| out | <i>target_cap</i> | The kernel stores the new task's capability into this slot. |
| in, out | <i>utcb_area</i> | Pointer to flexpage that describes an area of kernel-user memory that can be used for UTCBs and vCPU state-save-areas of the new task. |

On systems without MMU, the flexpage is adjusted to reflect the actually allocated physical address.

Returns

Syscall return tag.

Return values

| | |
|------------------------|---|
| <code>L4_EOK</code> | No error occurred. |
| <code>-L4_EPERM</code> | Insufficient permissions; see precondition. |
| <code><0</code> | Error code. |

Precondition

The capability `factory` must have the permission `L4_CAP_FPAGE_S`.

Note

The size of the UTCB area specifies indirectly the number of UTCBs available for this task. Refer to `l4_task_add_ku_mem()` for adding more of this type of memory.

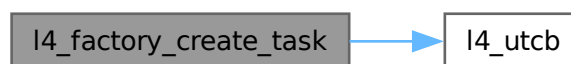
See also

[Task](#)

Definition at line 531 of file `factory.h`.

References `L4_NOTHROW`, and `l4_utcb()`.

Here is the call graph for this function:

**13.1.10.5.2.6 l4_factory_create_thread()**

```
l4_msgtag_t l4_factory_create_thread (
    l4_cap_idx_t factory,
    l4_cap_idx_t target_cap) [inline]
```

Create a new thread.

Parameters

| | | |
|-----|-------------------|---|
| | <i>factory</i> | Capability selector for factory to use for creation. |
| out | <i>target_cap</i> | The kernel stores the new thread's capability into this slot. |

Returns

Syscall return tag

Return values

| | |
|------------------|---|
| <i>L4_EOK</i> | No error occurred. |
| <i>-L4_EPERM</i> | Insufficient permissions; see precondition. |
| <i><0</i> | Error code. |

Precondition

The capability *factory* must have the permission [L4_CAP_FPAGE_S](#).

See also

[Thread](#)

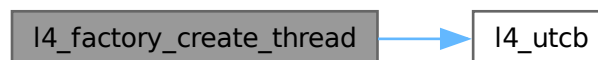
Examples

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line [538](#) of file [factory.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



13.1.10.5.2.7 l4_factory_create_thread_group()

```
l4_msgtag_t l4_factory_create_thread_group (
    l4_cap_idx_t factory,
    l4_cap_idx_t target_cap,
    unsigned policy) [inline]
```

Create a new thread group.

An IPC endpoint can be bound to a thread group. When a message arrives at the IPC endpoint, a specific thread of the thread group is selected to actually receive the message. A thread group is a send destination for an IPC endpoint.

Parameters

| | | |
|-----|-------------------|--|
| | <i>factory</i> | Capability selector for factory to use for creation. |
| out | <i>target_cap</i> | The kernel stores the new thread group's capability into this slot. |
| | <i>policy</i> | Policy parameter for the thread group. See #L4_thread_group_policy for a list of supported values. |

Returns

Syscall return tag

Return values

| | |
|-------------------|--|
| <i>L4_EOK</i> | No error occurred. |
| <i>-L4_ENOMEM</i> | Out-of-memory during allocation of the thread group object. |
| <i>-L4_EINVAL</i> | Invalid policy parameter. |
| <i>-L4_EPERM</i> | The factory instance requires L4_CAP_FPAGE_S rights on <code>factory</code> and L4_CAP_FPAGE_S is not present. |
| <i><0</i> | Error code. |

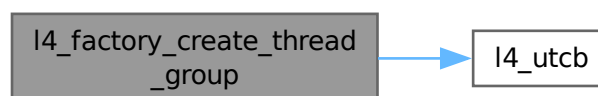
See also

[Thread groups](#)

Definition at line 582 of file [factory.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



13.1.10.5.2.8 l4_factory_create_vcpu_context()

```
l4_msgtag_t l4_factory_create_vcpu_context (
    l4_cap_idx_t factory,
    l4_cap_idx_t target_cap) [inline]
```

Create a new vCPU context.

A vCPU context typically represents a hardware structure that captures the state of a vCPU on a CPU (e.g. VMX VMCS).

Parameters

| | | |
|-----|-------------------|---|
| | <i>factory</i> | Capability selector for factory to use for creation. |
| out | <i>target_cap</i> | The kernel stores the new vCPU context's capability into this slot. |

Returns

Syscall return tag

Return values

| | |
|------------------|---|
| <i>L4_EOK</i> | No error occurred. |
| <i>-L4_EPERM</i> | Insufficient permissions; see precondition. |
| <i><0</i> | Error code. |

Precondition

The capability *factory* must have the permission [L4_CAP_FPAGE_S](#).

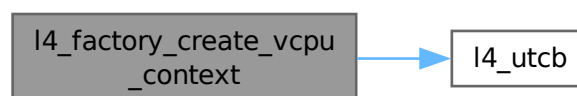
See also

[Virtual Machines](#)

Definition at line 575 of file [factory.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



13.1.10.5.2.9 l4_factory_create_vm()

```
l4_msgtag_t l4_factory_create_vm (
    l4_cap_idx_t factory,
    l4_cap_idx_t target_cap) [inline]
```

Create a new virtual machine.

Parameters

| | | |
|-----|-------------------|---|
| | <i>factory</i> | Capability selector for factory to use for creation. |
| out | <i>target_cap</i> | The kernel stores the new VM's capability into this slot. |

Returns

Syscall return tag

Return values

| | |
|------------------|---|
| <i>L4_EOK</i> | No error occurred. |
| <i>-L4_EPERM</i> | Insufficient permissions; see precondition. |
| <i><0</i> | Error code. |

Precondition

The capability *factory* must have the permission [L4_CAP_FPAGE_S](#).

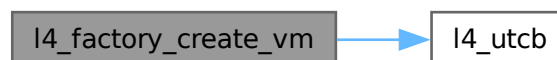
See also

[Virtual Machines](#)

Definition at line [568](#) of file [factory.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

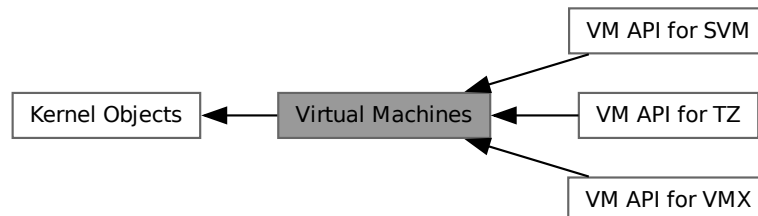
Here is the call graph for this function:



13.1.10.6 Virtual Machines

Virtual Machine API.

Collaboration diagram for Virtual Machines:



Topics

- VM API for SVM 300
Virtual machine API for SVM.
- VM API for VMX 301
Virtual machine API for VMX.
- VM API for TZ 321
Virtual Machine API for ARM TrustZone.

13.1.10.6.1 Detailed Description

Virtual Machine API.

13.1.10.6.2 VM API for SVM

Virtual machine API for SVM.

Collaboration diagram for VM API for SVM:



Data Structures

- struct [l4_vm_svm_vmcb_control_area](#)
VMCB structure for SVM VMs.
- struct [l4_vm_svm_vmcb_state_save_area_seg](#)
State save area segment selector struct.
- struct [l4_vm_svm_vmcb_state_save_area](#)
State save area structure for SVM VMs.
- struct [l4_vm_svm_vmcb_t](#)
Control structure for SVM VMs.

Typedefs

- typedef struct [l4_vm_svm_vmcb_control_area](#) [l4_vm_svm_vmcb_control_area_t](#)
VMCB structure for SVM VMs.
- typedef struct [l4_vm_svm_vmcb_state_save_area_seg](#) [l4_vm_svm_vmcb_state_save_area_seg_t](#)
State save area segment selector struct.
- typedef struct [l4_vm_svm_vmcb_state_save_area](#) [l4_vm_svm_vmcb_state_save_area_t](#)
State save area structure for SVM VMs.
- typedef struct [l4_vm_svm_vmcb_t](#) [l4_vm_svm_vmcb_t](#)
Control structure for SVM VMs.

13.1.10.6.2.1 Detailed Description

Virtual machine API for SVM.

13.1.10.6.3 VM API for VMX

Virtual machine API for VMX.

Collaboration diagram for VM API for VMX:



Data Structures

- struct [l4_vmx_offset_table_t](#)
Software VMCS field offset table.
- struct [l4_vm_vmx_vcpu_vmcs_t](#)
VMX software VMCS.
- struct [l4_vm_vmx_vcpu_infos_t](#)
VMX information members.
- struct [l4_vm_vmx_vcpu_state_t](#)
VMX vCPU state.

Typedefs

- typedef struct l4_vmx_offset_table_t [l4_vmx_offset_table_t](#)
Software VMCS field offset table.
- typedef struct l4_vm_vmx_vcpu_vmcs_t [l4_vm_vmx_vcpu_vmcs_t](#)
VMX software VMCS.
- typedef struct l4_vm_vmx_vcpu_infos_t [l4_vm_vmx_vcpu_infos_t](#)
VMX information members.
- typedef struct l4_vm_vmx_vcpu_state_t [l4_vm_vmx_vcpu_state_t](#)
VMX vCPU state.

Enumerations

- enum [l4_vm_vmx_caps_regs](#) {
[L4_VM_VMX_BASIC_REG](#) = 0 , [L4_VM_VMX_TRUE_PINBASED_CTLS_REG](#) = 1 , [L4_VM_VMX_TRUE_PROCBASED_CTLS_REG](#) = 2 , [L4_VM_VMX_TRUE_EXIT_CTLS_REG](#) = 3 ,
[L4_VM_VMX_TRUE_ENTRY_CTLS_REG](#) = 4 , [L4_VM_VMX_MISC_REG](#) = 5 , [L4_VM_VMX_CR0_FIXED0_REG](#) = 6 , [L4_VM_VMX_CR0_FIXED1_REG](#) = 7 ,
[L4_VM_VMX_CR4_FIXED0_REG](#) = 8 , [L4_VM_VMX_CR4_FIXED1_REG](#) = 9 , [L4_VM_VMX_VMCS_ENUM_REG](#) = 10 , [L4_VM_VMX_PROCBASED_CTLS2_REG](#) = 11 ,
[L4_VM_VMX_EPT_VPID_CAP_REG](#) = 12 , [L4_VM_VMX_NESTED_REVISION](#) = 13 , [L4_VM_VMX_NUM_CAPS_REGS](#)
}

Exported VMX capability registers.
- enum [l4_vm_vmx_dfl1_regs](#) {
[L4_VM_VMX_PINBASED_CTLS_DFL1_REG](#) = 0 , [L4_VM_VMX_PROCBASED_CTLS_DFL1_REG](#) = 1 ,
[L4_VM_VMX_EXIT_CTLS_DFL1_REG](#) = 2 , [L4_VM_VMX_ENTRY_CTLS_DFL1_REG](#) = 3 ,
[L4_VM_VMX_NUM_DFL1_REGS](#) }

Exported VMX capability registers (default to 1 bits).
- enum [l4_vm_vmx_sw_fields](#) {
[L4_VM_VMX_VMCS_CR2](#) = 0x6880 , [L4_VM_VMX_VMCS_NAT_ARG0](#) = 0x6882 , [L4_VM_VMX_VMCS_NAT_ARG1](#) = 0x6884 , [L4_VM_VMX_VMCS_NAT_ARG2](#) = 0x6886 ,
[L4_VM_VMX_VMCS_NAT_ARG3](#) = 0x6888 , [L4_VM_VMX_VMCS_XCR0](#) = 0x2880 , [L4_VM_VMX_VMCS_MSR_SYSCALL_MSR](#) = 0x2882 , [L4_VM_VMX_VMCS_MSR_LSTAR](#) = 0x2884 ,
[L4_VM_VMX_VMCS_MSR_CSTAR](#) = 0x2886 , [L4_VM_VMX_VMCS_MSR_TSC_AUX](#) = 0x2888 ,
[L4_VM_VMX_VMCS_MSR_STAR](#) = 0x288a , [L4_VM_VMX_VMCS_MSR_KERNEL_GS_BASE](#) = 0x288c }

Additional (software-defined) VMCS fields.
- enum [l4_vm_vmx_vmcs_sizes](#) { [L4_VM_VMX_VMCS_SIZE_VALUES](#) = 2560 , [L4_VM_VMX_VMCS_SIZE_DIRTY_BITMAP](#) = 320 }

Sizes of software VMCS members.

Functions

- [l4_uint64_t l4_vm_vmx_get_caps](#) ([l4_vm_vmx_vcpu_state_t](#) const *vcpu_state, enum [l4_vm_vmx_caps_regs](#) caps_reg) [L4_NOTHROW](#)
Get a capability register for VMX.
- [l4_uint32_t l4_vm_vmx_get_caps_default1](#) ([l4_vm_vmx_vcpu_state_t](#) const *vcpu_state, enum [l4_vm_vmx_dfl1_regs](#) dfl1_reg) [L4_NOTHROW](#)
Get a default to one capability register for VMX.
- unsigned [l4_vm_vmx_field_len](#) (unsigned field) [L4_NOTHROW](#)
Return length in bytes of a VMCS field.
- unsigned [l4_vm_vmx_field_order](#) (unsigned field) [L4_NOTHROW](#)
Return length in power of two (bytes) of a VMCS field.

- `void l4_vm_vmx_clear (l4_vm_vmx_vcpu_vmcs_t *vmcs, l4_vm_vmx_vcpu_vmcs_t *dest_vmcs) L4_NOTHROW`
Save the content from the software VMCS to a different software VMCS.
- `void l4_vm_vmx_ptr_load (l4_vm_vmx_vcpu_vmcs_t *vmcs, l4_vm_vmx_vcpu_vmcs_t *src_vmcs) L4_NOTHROW`
Load the content from a different software VMCS to the software VMCS.
- `l4_uint32_t l4_vm_vmx_get_cr2_index (l4_vm_vmx_vcpu_vmcs_t const *vmcs) L4_NOTHROW`
Get the software VMCS field index of the virtual CR2 register.
- `l4_umword_t l4_vm_vmx_read_nat (l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field) L4_NOTHROW`
Read a natural-width software VMCS field.
- `l4_uint16_t l4_vm_vmx_read_16 (l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field) L4_NOTHROW`
Read a 16-bit software VMCS field.
- `l4_uint32_t l4_vm_vmx_read_32 (l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field) L4_NOTHROW`
Read a 32-bit software VMCS field.
- `l4_uint64_t l4_vm_vmx_read_64 (l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field) L4_NOTHROW`
Read a 64-bit software VMCS field.
- `l4_uint64_t l4_vm_vmx_read (l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field) L4_NOTHROW`
Read any software VMCS field.
- `void l4_vm_vmx_write_nat (l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field, l4_umword_t val) L4_NOTHROW`
Write to a natural-width software VMCS field.
- `void l4_vm_vmx_write_16 (l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field, l4_uint16_t val) L4_NOTHROW`
Write to a 16-bit software VMCS field.
- `void l4_vm_vmx_write_32 (l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field, l4_uint32_t val) L4_NOTHROW`
Write to a 32-bit software VMCS field.
- `void l4_vm_vmx_write_64 (l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field, l4_uint64_t val) L4_NOTHROW`
Write to a 64-bit software VMCS field.
- `void l4_vm_vmx_write (l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field, l4_uint64_t val) L4_NOTHROW`
Write to an arbitrary software VMCS field.
- `void l4_vm_vmx_set_hw_vmcs (l4_vm_vmx_vcpu_vmcs_t *vmcs, l4_cap_idx_t vmcs_cap) L4_NOTHROW`
Associate the software VMCS with a vCPU context, i.e.
- `l4_cap_idx_t l4_vm_vmx_get_hw_vmcs (l4_vm_vmx_vcpu_vmcs_t *vmcs) L4_NOTHROW`
Get the vCPU context (i.e.

13.1.10.6.3.1 Detailed Description

Virtual machine API for VMX.

13.1.10.6.3.2 Typedef Documentation

`l4_vm_vmx_vcpu_state_t`

```
typedef struct l4_vm_vmx_vcpu_state_t l4_vm_vmx_vcpu_state_t
```

VMX vCPU state.

This is a specialization of the generic vCPU state for VMX. This data structure represents the following memory layout:

- 0x000 - 0x1ff: Standard vCPU state (with padding). See [l4_vcpu_state_t](#).
- 0x200 - 0x3ff: VMX information members (with padding). See [l4_vm_vmx_vcpu_infos_t](#).
- 0x400 - 0xffff: VMX software VMCS. See [l4_vm_vmx_vcpu_vmcs_t](#).

Note

The memory layout is documented here for reference purposes. However, the users are strongly discouraged from accessing the data structure directly. The API functions defined in this file are the preferred way of achieving the functionality.

l4_vm_vmx_vcpu_vmcs_t

```
typedef struct l4_vm_vmx_vcpu_vmcs_t l4_vm_vmx_vcpu_vmcs_t
```

VMX software VMCS.

This data structure represents the following memory layout:

- 0x000 - 0x007: Reserved (ignored by the kernel). In the hardware VMCS, the revision identifier and the abort indicator are stored in this area. Hereby we simply ignore these two entries.
- 0x008 - 0x00f: User space data (ignored by the kernel). This currently stores the pointer to a different software VMCS whose content has been loaded to this software VMCS.
- 0x010 - 0x013: VMCS field index of the software-defined CR2 field in the software VMCS.
- 0x014 - 0x017: Reserved.
- 0x018 - 0x01f: Capability of the vCPU context, i.e. the hardware VMCS object (with padding).
- 0x020 - 0x047: Software VMCS field offset table. See [l4_vmx_offset_table_t](#).
- 0x048 - 0x0bf: Reserved.
- 0x0c0 - 0xabf: Software VMCS fields (with padding).
- 0xac0 - 0xbff: Software VMCS fields dirty bitmap (with padding).

Note

The memory layout is documented here for reference purposes. However, the users are strongly discouraged from accessing the data structure directly. The API functions defined in this file are the preferred way of achieving the functionality.

l4_vmx_offset_table_t

```
typedef struct l4_vmx_offset_table_t l4_vmx_offset_table_t
```

Software VMCS field offset table.

This data structure represents the following memory layout:

- 0x00 - 0x02: 3 offsets for 16-bit fields.
- 0x03: Reserved.
- 0x04 - 0x06: 3 offsets for 64-bit fields.
- 0x07: Reserved.
- 0x08 - 0x0a: 3 offsets for 32-bit fields.

- 0x0b: Reserved.
- 0x0c - 0x0e: 3 offsets for natural-width fields.
- 0x0f: Reserved.
- 0x10 - 0x12: 3 limits for 16-bit fields.
- 0x13: Reserved.
- 0x14 - 0x16: 3 limits for 64-bit fields.
- 0x17: Reserved.
- 0x18 - 0x1a: 3 limits for 32-bit fields.
- 0x1b: Reserved.
- 0x1c - 0x1e: 3 limits for natural-width fields.
- 0x1f: Reserved.
- 0x20 - 0x23: 4 index shifts.
- 0x24: Offset of the first software VMCS field.
- 0x25: Size of the software VMCS fields.
- 0x26 - 0x27: Reserved.

The offsets/limits in each size category are in the following order:

- Control fields.
- Read-only fields.
- Guest fields.

The index shifts are in the following order:

- 16-bit.
- 64-bit.
- 32-bit.
- Natural-width.

All offsets/limits/sizes are represented in a 64-byte granule.

The offsets (after being multiplied by 64) are indexes in the values array in [l4_vm_vmx_vcpu_vmcs_t](#) and bit indexes in the dirty_bitmap array in [l4_vm_vmx_vcpu_vmcs_t](#).

The limits (after being multiplied by 64) represent the range of the available indexes.

Note

The memory layout is documented here for reference purposes. However, the users are strongly discouraged from accessing the data structure directly. The API functions defined in this file are the preferred way of achieving the functionality.

13.1.10.6.3.3 Enumeration Type Documentation

L4_vm_vmx_caps_regs

enum [L4_vm_vmx_caps_regs](#)

Exported VMX capability registers.

Enumerator

| | |
|------------------------------------|---|
| L4_VM_VMX_BASIC_REG | Basic VMX capabilities. |
| L4_VM_VMX_TRUE_PINBASED_CTLIS_REG | True pin-based control caps. |
| L4_VM_VMX_TRUE_PROCBASED_CTLIS_REG | True processor based control caps. |
| L4_VM_VMX_TRUE_EXIT_CTLIS_REG | True exit control caps. |
| L4_VM_VMX_TRUE_ENTRY_CTLIS_REG | True entry control caps. |
| L4_VM_VMX_MISC_REG | Misc caps. |
| L4_VM_VMX_CR0_FIXED0_REG | Fixed to 0 bits of CR0. |
| L4_VM_VMX_CR0_FIXED1_REG | Fixed to 1 bits of CR0. |
| L4_VM_VMX_CR4_FIXED0_REG | Fixed to 0 bits of CR4. |
| L4_VM_VMX_CR4_FIXED1_REG | Fixed to 1 bits of CR4. |
| L4_VM_VMX_VMCS_ENUM_REG | VMCS enumeration info. |
| L4_VM_VMX_PROCBASED_CTLIS2_REG | Processor based control 2 caps. |
| L4_VM_VMX_EPT_VPID_CAP_REG | EPT and VPID caps. |
| L4_VM_VMX_NESTED_REVISION | Nested VMCS revision. |
| L4_VM_VMX_NUM_CAPS_REGS | Total number of VMX capability registers. |

Definition at line 28 of file [__vm-vmx.h](#).

L4_vm_vmx_dfl1_regs

enum [L4_vm_vmx_dfl1_regs](#)

Exported VMX capability registers (default to 1 bits).

Enumerator

| | |
|------------------------------------|---|
| L4_VM_VMX_PINBASED_CTLIS_DFL1_REG | Default 1 bits in pin-based controls. |
| L4_VM_VMX_PROCBASED_CTLIS_DFL1_REG | Default 1 bits in processor-based controls. |
| L4_VM_VMX_EXIT_CTLIS_DFL1_REG | Default 1 bits in exit controls. |
| L4_VM_VMX_ENTRY_CTLIS_DFL1_REG | Default 1 bits in entry controls. |
| L4_VM_VMX_NUM_DFL1_REGS | Total number of default on registers. |

Definition at line 51 of file [__vm-vmx.h](#).

L4_vm_vmx_sw_fields

```
enum L4_vm_vmx_sw_fields
```

Additional (software-defined) VMCS fields.

The VMCS offsets defined here are actually not in the hardware VMCS. However our VMMs run in user mode and need to have access to certain registers available in kernel mode only. So we put them into our software VMCS.

Enumerator

| | |
|-----------------------------------|--|
| L4_VM_VMX_VMCS_CR2 | Software VMCS offset for CR2. Note You usually need to check this value against the value you get from l4_vm_vmx_get_cr2_index() to make sure you are running on a compatible kernel. |
| L4_VM_VMX_VMCS_NAT_ARG0 | Custom argument passed from kernel to user space. |
| L4_VM_VMX_VMCS_NAT_ARG1 | Custom argument passed from kernel to user space. |
| L4_VM_VMX_VMCS_NAT_ARG2 | Custom argument passed from kernel to user space. |
| L4_VM_VMX_VMCS_NAT_ARG3 | Custom argument passed from kernel to user space. |
| L4_VM_VMX_VMCS_XCR0 | VMCS offset of extended control register XCR0. |
| L4_VM_VMX_VMCS_MSR_SYSCALL_MASK | VMCS offset of system call flag mask MSR. |
| L4_VM_VMX_VMCS_MSR_LSTAR | VMCS offset of IA32e mode system call target address MSR. |
| L4_VM_VMX_VMCS_MSR_CSTAR | VMCS offset of IA32 mode system call target address MSR. |
| L4_VM_VMX_VMCS_MSR_TSC_AUX | VMCS offset of auxiliary TSC signature MSR. |
| L4_VM_VMX_VMCS_MSR_STAR | VMCS offset of system call target address MSR. |
| L4_VM_VMX_VMCS_MSR_KERNEL_GS_BASE | VMCS offset of GS base address swap target MSR. |

Definition at line 69 of file [__vm-vmx.h](#).

L4_vm_vmx_vmcs_sizes

```
enum L4_vm_vmx_vmcs_sizes
```

Sizes of software VMCS members.

Enumerator

| | |
|----------------------------------|--|
| L4_VM_VMX_VMCS_SIZE_VALUES | Size of the software VMCS values member. |
| L4_VM_VMX_VMCS_SIZE_DIRTY_BITMAP | Size of the software VMCS dirty bitmap member. |

Definition at line 170 of file [__vm-vmx.h](#).

13.1.10.6.3.4 Function Documentation

l4_vm_vmx_clear()

```
void l4_vm_vmx_clear (
    l4_vm_vmx_vcpu_vmcs_t * vmcs,
    l4_vm_vmx_vcpu_vmcs_t * dest_vmcs) [inline]
```

Save the content from the software VMCS to a different software VMCS.

Parameters

| | |
|------------------|---|
| <i>vmcs</i> | Pointer to the source software VMCS. |
| <i>dest_vmcs</i> | Pointer to the destination software VMCS. |

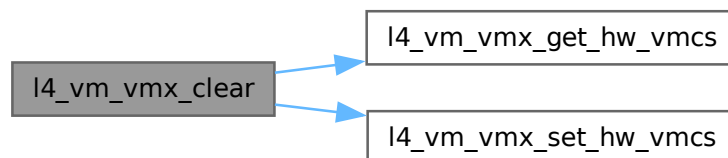
This function is comparable to the VMX VMCLEAR instruction.

Definition at line 698 of file [__vm-vmx.h](#).

References [L4_NOTHROW](#), [l4_vm_vmx_get_hw_vmcs\(\)](#), [l4_vm_vmx_set_hw_vmcs\(\)](#), and [L4_VM_VMX_VMCS_SIZE_DIRTY_BITM](#)

Referenced by [l4_vm_vmx_ptr_load\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



l4_vm_vmx_field_len()

```
unsigned l4_vm_vmx_field_len (  
    unsigned field) [inline]
```

Return length in bytes of a VMCS field.

Parameters

| | |
|--------------|---------------|
| <i>field</i> | Field number. |
|--------------|---------------|

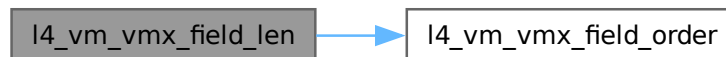
Returns

Width of field in bytes.

Definition at line 593 of file [__vm-vmx.h](#).

References [L4_NOTHROW](#), and [l4_vm_vmx_field_order\(\)](#).

Here is the call graph for this function:

**l4_vm_vmx_field_order()**

```
unsigned l4_vm_vmx_field_order (  
    unsigned field) [inline]
```

Return length in power of two (bytes) of a VMCS field.

Parameters

| | |
|--------------|---------------|
| <i>field</i> | Field number. |
|--------------|---------------|

Returns

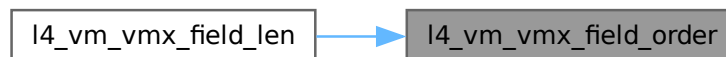
Width of field in power of two (bytes).

Definition at line 600 of file [__vm-vmx.h](#).

References [L4_NOTHROW](#).

Referenced by [l4_vm_vmx_field_len\(\)](#).

Here is the caller graph for this function:

**l4_vm_vmx_get_caps()**

```
l4_uint64_t l4_vm_vmx_get_caps (  
    l4_vm_vmx_vcpu_state_t const * vcpu_state,  
    enum L4_vm_vmx_caps_regs caps_reg) [inline]
```

Get a capability register for VMX.

Parameters

| | |
|-------------------|---|
| <i>vcpu_state</i> | Pointer to the vCPU state. |
| <i>caps_reg</i> | Capability register index (see L4_vm_vmx_caps_regs). |

Returns

The value of the capability register.

Definition at line 884 of file [__vm-vmx.h](#).

References [L4_NOTHROW](#).

l4_vm_vmx_get_caps_default1()

```
l4_uint32_t l4_vm_vmx_get_caps_default1 (
    l4_vm_vmx_vcpu_state_t const * vcpu_state,
    enum L4_vm_vmx_dfll_regs dfll_reg) [inline]
```

Get a default to one capability register for VMX.

Parameters

| | |
|-------------------|--|
| <i>vcpu_state</i> | Pointer to the vCPU state. |
| <i>dfll_reg</i> | Default to 1 capability register index (see L4_vm_vmx_dfll_regs). |

Returns

The value of the capability register.

Definition at line 892 of file [__vm-vmx.h](#).

References [L4_NOTHROW](#).

l4_vm_vmx_get_cr2_index()

```
l4_uint32_t l4_vm_vmx_get_cr2_index (
    l4_vm_vmx_vcpu_vmcs_t const * vmcs) [inline]
```

Get the software VMCS field index of the virtual CR2 register.

Parameters

| | |
|-------------|-------------------------------|
| <i>vmcs</i> | Pointer to the software VMCS. |
|-------------|-------------------------------|

Returns

The field index used for the virtual CR2 register as used by the current Fiasco.OC interface.

The CR2 register is actually not in the hardware VMCS, however our VMMs run in user mode and need to have access to this register so we put it into our software VMCS.

See also

[L4_VM_VMX_VMCS_CR2](#)

Definition at line 900 of file [__vm-vmx.h](#).

References [L4_NOTHROW](#).

l4_vm_vmx_get_hw_vmcs()

```
l4_cap_idx_t l4_vm_vmx_get_hw_vmcs (
    l4_vm_vmx_vcpu_vmcs_t * vmcs) [inline]
```

Get the vCPU context (i.e.

the hardware VMCS object) associated with the software VMCS.

Parameters

| | |
|-------------|-------------------------------|
| <i>vmcs</i> | Pointer to the software VMCS. |
|-------------|-------------------------------|

Returns

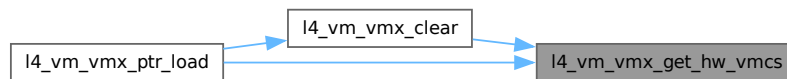
vCPU context (hardware VMCS object) capability.

Definition at line 915 of file [__vm-vmx.h](#).

References [L4_CAP_MASK](#), and [L4_NOTHROW](#).

Referenced by [l4_vm_vmx_clear\(\)](#), and [l4_vm_vmx_ptr_load\(\)](#).

Here is the caller graph for this function:



l4_vm_vmx_ptr_load()

```
void l4_vm_vmx_ptr_load (
    l4_vm_vmx_vcpu_vmcs_t * vmcs,
    l4_vm_vmx_vcpu_vmcs_t * src_vmcs) [inline]
```

Load the content from a different software VMCS to the software VMCS.

Parameters

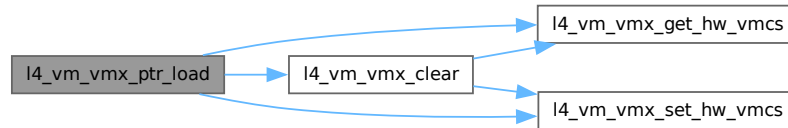
| | |
|-----------------|---|
| <i>vmcs</i> | Pointer to the destination software VMCS. |
| <i>src_vmcs</i> | Pointer to the source software VMCS. |

This function is comparable to the VMX VMPTRLD instruction.

Definition at line 719 of file [__vm-vmx.h](#).

References [L4_NOTHROW](#), [l4_vm_vmx_clear\(\)](#), [l4_vm_vmx_get_hw_vmcs\(\)](#), [l4_vm_vmx_set_hw_vmcs\(\)](#), and [L4_VM_VMX_VMCS_SIZE_DIRTY_BITMAP](#).

Here is the call graph for this function:



`l4_vm_vmx_read()`

```

l4_uint64_t l4_vm_vmx_read (
    l4_vm_vmx_vcpu_vmcs_t * vmcs,
    unsigned field) [inline]
  
```

Read any software VMCS field.

Parameters

| | |
|--------------|---|
| <i>vmcs</i> | Pointer to the software VMCS. |
| <i>field</i> | The VMCS field index as used on VMX hardware. |

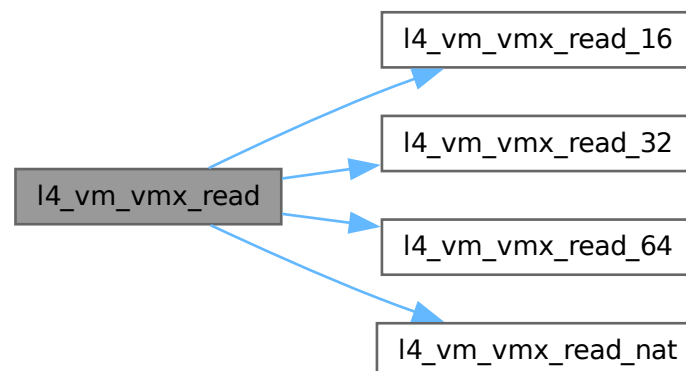
Returns

The value of the software VMCS field with the given index.

Definition at line 787 of file `__vm-vmx.h`.

References [L4_NOTHROW](#), [l4_vm_vmx_read_16\(\)](#), [l4_vm_vmx_read_32\(\)](#), [l4_vm_vmx_read_64\(\)](#), and [l4_vm_vmx_read_nat\(\)](#).

Here is the call graph for this function:



l4_vm_vmx_read_16()

```
l4_uint16_t l4_vm_vmx_read_16 (  
    l4_vm_vmx_vcpu_vmcs_t * vmcs,  
    unsigned field) [inline]
```

Read a 16-bit software VMCS field.

Parameters

| | |
|--------------|---|
| <i>vmcs</i> | Pointer to the software VMCS. |
| <i>field</i> | The VMCS field index as used on VMX hardware. |

Returns

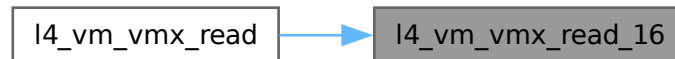
The value of the software VMCS field with the given index.

Definition at line 754 of file [__vm-vmx.h](#).

References [L4_NOTHROW](#).

Referenced by [l4_vm_vmx_read\(\)](#).

Here is the caller graph for this function:



l4_vm_vmx_read_32()

```
l4_uint32_t l4_vm_vmx_read_32 (  
    l4_vm_vmx_vcpu_vmcs_t * vmcs,  
    unsigned field) [inline]
```

Read a 32-bit software VMCS field.

Parameters

| | |
|--------------|---|
| <i>vmcs</i> | Pointer to the software VMCS. |
| <i>field</i> | The VMCS field index as used on VMX hardware. |

Returns

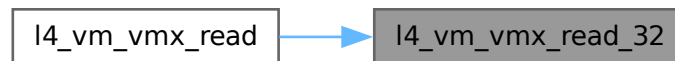
The value of the software VMCS field with the given index.

Definition at line 765 of file [__vm-vmx.h](#).

References [L4_NOTHROW](#).

Referenced by [l4_vm_vmx_read\(\)](#).

Here is the caller graph for this function:

**l4_vm_vmx_read_64()**

```
l4_uint64_t l4_vm_vmx_read_64 (  
    l4_vm_vmx_vcpu_vmcs_t * vmcs,  
    unsigned field) [inline]
```

Read a 64-bit software VMCS field.

Parameters

| | |
|--------------|---|
| <i>vmcs</i> | Pointer to the software VMCS. |
| <i>field</i> | The VMCS field index as used on VMX hardware. |

Returns

The value of the software VMCS field with the given index.

Definition at line 776 of file [__vm-vmx.h](#).

References [L4_NOTHROW](#).

Referenced by [l4_vm_vmx_read\(\)](#).

Here is the caller graph for this function:



l4_vm_vmx_read_nat()

```
l4_umword_t l4_vm_vmx_read_nat (
    l4_vm_vmx_vcpu_vmcs_t * vmcs,
    unsigned field) [inline]
```

Read a natural-width software VMCS field.

Parameters

| | |
|--------------|---|
| <i>vmcs</i> | Pointer to the software VMCS. |
| <i>field</i> | The VMCS field index as used on VMX hardware. |

Returns

The value of the software VMCS field with the given index.

Definition at line 743 of file [__vm-vmx.h](#).

References [L4_NOTHROW](#).

Referenced by [l4_vm_vmx_read\(\)](#).

Here is the caller graph for this function:

**l4_vm_vmx_set_hw_vmcs()**

```
void l4_vm_vmx_set_hw_vmcs (
    l4_vm_vmx_vcpu_vmcs_t * vmcs,
    l4_cap_idx_t vmcs_cap) [inline]
```

Associate the software VMCS with a vCPU context, i.e.

a hardware VMCS object.

The VMX extended vCPU state is unable to be resumed unless it is associated with a vCPU context, i.e. a hardware VMCS object: An `L4::Vcpu_context` from the user space point of view with its kernel counterpart `Vmx_vmcs`.

Note

When replacing the vCPU context, the dirty bitmap of the software VMCS is not touched, neither by the kernel nor by the API functions. This is on purpose, to enable efficient switching between separate VMs in the common case. If there is a logical discrepancy between the content of the software VMCS and the replaced vCPU context, the user is responsible for explicitly setting the relevant software VMCS fields and/or the relevant software VMCS dirty bitmap bits to ensure that the discrepancy is rectified on the next vCPU resume. This needs to be done regardless of using the API functions (the preferred way) or accessing the data structures directly (the discouraged way).

Replacing the vCPU context while the vCPU is currently running has no immediate effect until the next vCPU resume. In addition to that, the kernel might cache the vCPU context internally (in other words, the capability is not looked up on every vCPU resume). To remove the association of the current vCPU context, simply replace it by an another vCPU context. The reference count of the previous vCPU context will be decremented accordingly on the next vCPU resume.

To remove the association of the current vCPU context without replacing it by an another vCPU context, pass an invalid capability with the bit 3 set and trigger a vCPU resume. The vCPU resume will fail in this case (due to the missing vCPU context), but the reference count of the previous vCPU context will be decremented accordingly.

There is no need to explicitly remove the association of the current vCPU context before deleting the software VMCS. Deleting the software VMCS automatically disassociates it from the vCPU context and a vCPU context with a reference count of 0 will be eventually deleted as well.

If the hardware limitations on the usage of the vCPU context are not observed (i.e. no hardware VMCS being active on more than one physical CPU), the vCPU will fail to resume.

Parameters

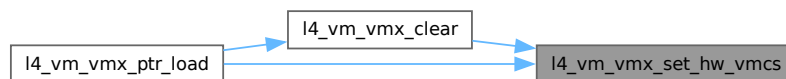
| | |
|-----------------|---|
| <i>vmcs</i> | Pointer to the software VMCS. |
| <i>vmcs_cap</i> | vCPU context (hardware VMCS object) capability. |

Definition at line 907 of file [__vm-vmx.h](#).

References [L4_NOTHROW](#).

Referenced by [l4_vm_vmx_clear\(\)](#), and [l4_vm_vmx_ptr_load\(\)](#).

Here is the caller graph for this function:

**`l4_vm_vmx_write()`**

```

void l4_vm_vmx_write (
    l4_vm_vmx_vcpu_vmcs_t * vmcs,
    unsigned field,
    l4_uint64_t val) [inline]
  
```

Write to an arbitrary software VMCS field.

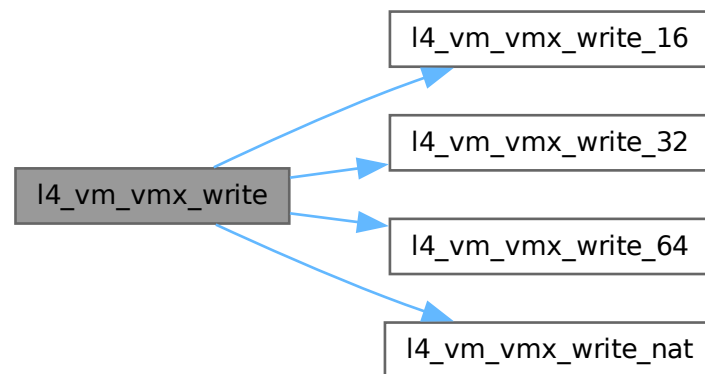
Parameters

| | |
|--------------|---|
| <i>vmcs</i> | Pointer to the software VMCS. |
| <i>field</i> | The VMCS field index as used on VMX hardware. |
| <i>val</i> | The value that shall be written to the given field. |

Definition at line 868 of file [__vm-vmx.h](#).

References [L4_NOTHROW](#), [l4_vm_vmx_write_16\(\)](#), [l4_vm_vmx_write_32\(\)](#), [l4_vm_vmx_write_64\(\)](#), and [l4_vm_vmx_write_nat\(\)](#).

Here is the call graph for this function:



`l4_vm_vmx_write_16()`

```
void l4_vm_vmx_write_16 (
    l4_vm_vmx_vcpu_vmcs_t * vmcs,
    unsigned field,
    l4_uint16_t val) [inline]
```

Write to a 16-bit software VMCS field.

Parameters

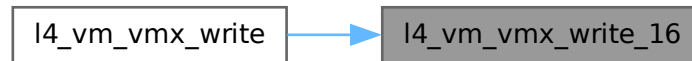
| | |
|--------------|---|
| <i>vmcs</i> | Pointer to the software VMCS. |
| <i>field</i> | The VMCS field index as used on VMX hardware. |
| <i>val</i> | The value that shall be written to the given field. |

Definition at line 820 of file [__vm-vmx.h](#).

References [L4_NOTHROW](#).

Referenced by [l4_vm_vmx_write\(\)](#).

Here is the caller graph for this function:



`l4_vm_vmx_write_32()`

```
void l4_vm_vmx_write_32 (  
    l4_vm_vmx_vcpu_vmcs_t * vmcs,  
    unsigned field,  
    l4_uint32_t val) [inline]
```

Write to a 32-bit software VMCS field.

Parameters

| | |
|--------------|---|
| <i>vmcs</i> | Pointer to the software VMCS. |
| <i>field</i> | The VMCS field index as used on VMX hardware. |
| <i>val</i> | The value that shall be written to the given field. |

Definition at line [836](#) of file [__vm-vmx.h](#).

References [L4_NOTHROW](#).

Referenced by [l4_vm_vmx_write\(\)](#).

Here is the caller graph for this function:



l4_vm_vmx_write_64()

```
void l4_vm_vmx_write_64 (
    l4_vm_vmx_vcpu_vmcs_t * vmcs,
    unsigned field,
    l4_uint64_t val) [inline]
```

Write to a 64-bit software VMCS field.

Parameters

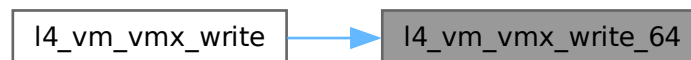
| | |
|--------------|---|
| <i>vmcs</i> | Pointer to the software VMCS. |
| <i>field</i> | The VMCS field index as used on VMX hardware. |
| <i>val</i> | The value that shall be written to the given field. |

Definition at line 852 of file [__vm-vmx.h](#).

References [L4_NOTHROW](#).

Referenced by [l4_vm_vmx_write\(\)](#).

Here is the caller graph for this function:

**l4_vm_vmx_write_nat()**

```
void l4_vm_vmx_write_nat (
    l4_vm_vmx_vcpu_vmcs_t * vmcs,
    unsigned field,
    l4_umword_t val) [inline]
```

Write to a natural-width software VMCS field.

Parameters

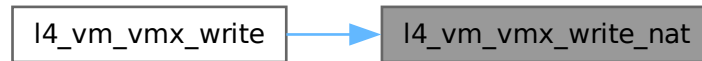
| | |
|--------------|---|
| <i>vmcs</i> | Pointer to the software VMCS. |
| <i>field</i> | The VMCS field index as used on VMX hardware. |
| <i>val</i> | The value that shall be written to the given field. |

Definition at line 804 of file [__vm-vmx.h](#).

References [L4_NOTHROW](#).

Referenced by [l4_vm_vmx_write\(\)](#).

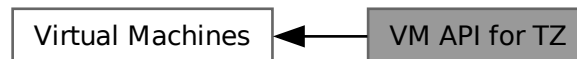
Here is the caller graph for this function:



13.1.10.6.4 VM API for TZ

Virtual Machine API for ARM TrustZone.

Collaboration diagram for VM API for TZ:



Data Structures

- struct [l4_vm_tz_state](#)
state structure for TrustZone VMs

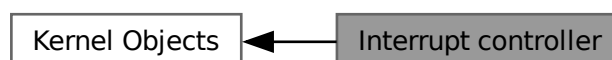
13.1.10.6.4.1 Detailed Description

Virtual Machine API for ARM TrustZone.

13.1.10.7 Interrupt controller

The C lcu interface, see [L4::lcu](#) for the C++ interface.

Collaboration diagram for Interrupt controller:



Data Structures

- struct [l4_icu_info_t](#)
Info structure for an ICU.

Typedefs

- typedef struct [l4_icu_info_t](#) [l4_icu_info_t](#)
Info structure for an ICU.

Enumerations

- enum [l4_icu_flags](#) { [L4_ICU_FLAG_MSI](#) }
Flags for IRQ numbers used for the ICU.

Functions

- [l4_msgtag_t l4_icu_bind](#) ([l4_cap_idx_t](#) icu, unsigned irqnum, [l4_cap_idx_t](#) irq) [L4_NOTHROW](#)
Bind an interrupt line of an interrupt controller to an interrupt object.
- [l4_msgtag_t l4_icu_bind_u](#) ([l4_cap_idx_t](#) icu, unsigned irqnum, [l4_cap_idx_t](#) irq, [l4_utcb_t](#) *utcb) [L4_NOTHROW](#)
Bind an interrupt line of an interrupt controller to an interrupt object.
- [l4_msgtag_t l4_icu_unbind](#) ([l4_cap_idx_t](#) icu, unsigned irqnum, [l4_cap_idx_t](#) irq) [L4_NOTHROW](#)
Remove binding of an interrupt line from the interrupt controller object.
- [l4_msgtag_t l4_icu_unbind_u](#) ([l4_cap_idx_t](#) icu, unsigned irqnum, [l4_cap_idx_t](#) irq, [l4_utcb_t](#) *utcb) [L4_NOTHROW](#)
Remove binding of an interrupt line from the interrupt controller object.
- [l4_msgtag_t l4_icu_set_mode](#) ([l4_cap_idx_t](#) icu, unsigned irqnum, [l4_umword_t](#) mode) [L4_NOTHROW](#)
Set interrupt mode.
- [l4_msgtag_t l4_icu_set_mode_u](#) ([l4_cap_idx_t](#) icu, unsigned irqnum, [l4_umword_t](#) mode, [l4_utcb_t](#) *utcb) [L4_NOTHROW](#)
Set interrupt mode.
- [l4_msgtag_t l4_icu_info](#) ([l4_cap_idx_t](#) icu, [l4_icu_info_t](#) *info) [L4_NOTHROW](#)
Get information about the ICU features.
- [l4_msgtag_t l4_icu_info_u](#) ([l4_cap_idx_t](#) icu, [l4_icu_info_t](#) *info, [l4_utcb_t](#) *utcb) [L4_NOTHROW](#)
Get information about the ICU features.
- [l4_msgtag_t l4_icu_msi_info](#) ([l4_cap_idx_t](#) icu, unsigned irqnum, [l4_uint64_t](#) source, [l4_icu_msi_info_t](#) *msi_info) [L4_NOTHROW](#)
Get MSI info about IRQ.
- [l4_msgtag_t l4_icu_msi_info_u](#) ([l4_cap_idx_t](#) icu, unsigned irqnum, [l4_uint64_t](#) source, [l4_icu_msi_info_t](#) *msi_info, [l4_utcb_t](#) *utcb) [L4_NOTHROW](#)
Get MSI info about IRQ.
- [l4_msgtag_t l4_icu_unmask](#) ([l4_cap_idx_t](#) icu, unsigned irqnum, [l4_umword_t](#) *label, [l4_timeout_t](#) to) [L4_NOTHROW](#)
Unmask an IRQ line.
- [l4_msgtag_t l4_icu_unmask_u](#) ([l4_cap_idx_t](#) icu, unsigned irqnum, [l4_umword_t](#) *label, [l4_timeout_t](#) to, [l4_utcb_t](#) *utcb) [L4_NOTHROW](#)
Unmask the given interrupt line.
- [l4_msgtag_t l4_icu_mask](#) ([l4_cap_idx_t](#) icu, unsigned irqnum, [l4_umword_t](#) *label, [l4_timeout_t](#) to) [L4_NOTHROW](#)
Mask an IRQ line.
- [l4_msgtag_t l4_icu_mask_u](#) ([l4_cap_idx_t](#) icu, unsigned irqnum, [l4_umword_t](#) *label, [l4_timeout_t](#) to, [l4_utcb_t](#) *utcb) [L4_NOTHROW](#)
Mask an IRQ line.

13.1.10.7.1 Detailed Description

The C Icu interface, see [L4::Icu](#) for the C++ interface.

Note

"ICU" is short for "interrupt control unit".

These functions define the interface for interrupt controllers, for binding IRQ objects to interrupt lines and other interrupt sources, as well as functions for masking and unmasking of interrupts.

To setup an IRQ line the following steps are required:

1. [l4_icu_set_mode\(\)](#) (optional if IRQ has a default mode)
2. [l4_rcv_ep_bind_thread\(\)](#) or [l4_rcv_ep_bind_snd_destination\(\)](#) to attach the IRQ object to a thread object.
3. [l4_icu_bind\(\)](#)
4. [l4_icu_unmask\(\)](#) to receive the first IRQ

For certain interrupt sources only some of these steps are necessary and supported, see [Scheduler](#) and [Virtual Console](#).

At most one [IRQs](#) object can be bound to a certain interrupt source and a certain [IRQs](#) object can be bound to at most one interrupt source.

Include File

```
#include <l4/sys/icu.h>
```

13.1.10.7.2 Typedef Documentation

13.1.10.7.2.1 l4_icu_info_t

```
typedef struct l4_icu_info_t l4_icu_info_t
```

Info structure for an ICU.

This structure contains information about the features of an ICU.

See also

[l4_icu_info\(\)](#).

13.1.10.7.3 Enumeration Type Documentation

13.1.10.7.3.1 L4_icu_flags

enum [L4_icu_flags](#)

Flags for IRQ numbers used for the ICU.

Enumerator

| | |
|-----------------|--|
| L4_ICU_FLAG_MSI | Flag to denote that the IRQ is actually an MSI. This flag may be used for l4_icu_bind() and l4_icu_unbind() functions to denote that the IRQ number is meant to be an MSI. |
|-----------------|--|

Definition at line 53 of file [icu.h](#).

13.1.10.7.4 Function Documentation

13.1.10.7.4.1 l4_icu_bind()

```
l4_msgtag_t l4_icu_bind (
    l4_cap_idx_t icu,
    unsigned irqnum,
    l4_cap_idx_t irq) [inline]
```

Bind an interrupt line of an interrupt controller to an interrupt object.

Parameters

| | |
|---------------|-----------------------------------|
| <i>icu</i> | ICU object to bind <i>irq</i> to. |
| <i>irqnum</i> | IRQ line at the ICU. |
| <i>irq</i> | IRQ object to bind to this ICU. |

Returns

Syscall return tag. The caller should check the return value using [l4_error\(\)](#) to check for errors and to identify the correct method for unmasking the interrupt. Return values < 0 indicate an error. A return value of 0 means a direct unmask via the IRQ object using [l4_irq_unmask\(\)](#). A return value of 1 means that the interrupt has to be unmasked via the ICU using [l4_icu_unmask\(\)](#).

Return values

| | |
|------------|---|
| -L4_EINVAL | <i>irq</i> is bound to an interrupt source. |
| -L4_EPERM | Insufficient permissions; see precondition. |

Precondition

The capability `irq` must have the permission [L4_CAP_FPAGE_W](#).

In case the `irq` is already bound to an interrupt source, it is unbound first. In case the `irq` is bound and the interrupt source is bound to a different IRQ object, only the unbinding happens. An IRQ object that is bound to an interrupt source will get unbound if the IRQ object is deleted.

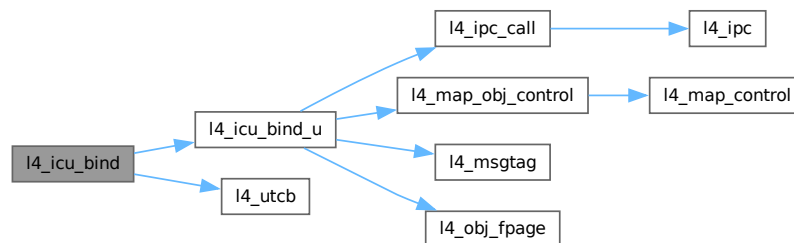
Examples

[examples/sys/isr/main.c](#).

Definition at line 496 of file [icu.h](#).

References [l4_icu_bind_u\(\)](#), [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:

**13.1.10.7.4.2 l4_icu_bind_u()**

```

l4_msgtag_t l4_icu_bind_u (
    l4_cap_idx_t icu,
    unsigned irqnum,
    l4_cap_idx_t irq,
    l4_utcb_t * utcb) [inline]

```

Bind an interrupt line of an interrupt controller to an interrupt object.

Parameters

| | |
|---------------|--|
| <i>icu</i> | The ICU object to bind <code>irq</code> to. |
| <i>irqnum</i> | IRQ line at the ICU. |
| <i>irq</i> | IRQ object for the given IRQ line to bind to this ICU. |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |

Returns

Syscall return tag. The caller should check the return value using [l4_error\(\)](#) to check for errors and to identify the correct method for unmasking the interrupt. Return values < 0 indicate an error. A return value of 0 means a direct unmask via the IRQ object using [L4::irq::unmask](#). A return value of 1 means that the interrupt has to be unmasked via the ICU using [L4::icu::unmask](#).

Return values

| | |
|----------------------------|---|
| -L4_EINVAL | <code>irq</code> is bound to an interrupt source. |
| -L4_EPERM | Insufficient permissions; see precondition. |

Precondition

The capability `irq` must have the permission [L4_CAP_FPAGE_W](#).

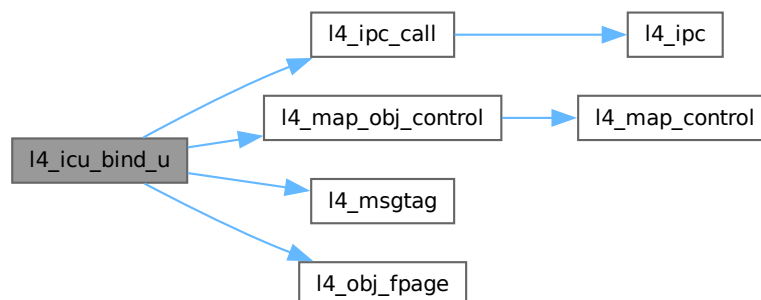
In case the `irq` is already bound to an interrupt source, it is unbound first. In case the `irq` is bound and the interrupt source is bound to a different [L4::irq](#) object, only the unbinding happens. An [L4::irq](#) object that is bound to an interrupt source will get unbound if the [L4::irq](#) object is deleted.

Definition at line 396 of file [icu.h](#).

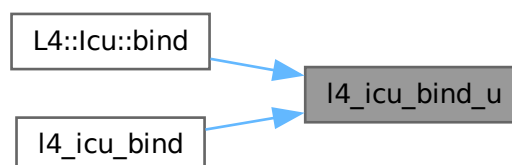
References [L4_CAP_FPAGE_RWS](#), [L4_ICU_OP_BIND](#), [l4_ipc_call\(\)](#), [L4_IPC_NEVER](#), [l4_map_obj_control\(\)](#), [l4_msgtag\(\)](#), [L4_NOTHROW](#), [l4_obj_fpage\(\)](#), [L4_PROTO_IRQ](#), [l4_msg_regs_t::mr](#), and [l4_fpage_t::raw](#).

Referenced by [L4::icu::bind\(\)](#), and [l4_icu_bind\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.1.10.7.4.3 l4_icu_info()

```
l4_msgtag_t l4_icu_info (
    l4_cap_idx_t icu,
    l4_icu_info_t * info) [inline]
```

Get information about the ICU features.

Parameters

| | | |
|-----|-------------|---|
| | <i>icu</i> | The ICU object from which information shall be retrieved. |
| out | <i>info</i> | Info structure to be filled with information. |

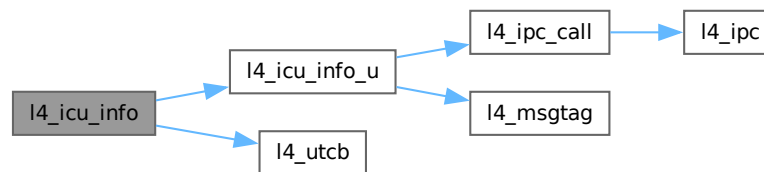
Returns

Syscall return tag

Definition at line 504 of file [icu.h](#).

References [l4_icu_info_u\(\)](#), [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



13.1.10.7.4.4 l4_icu_info_u()

```
l4_msgtag_t l4_icu_info_u (
    l4_cap_idx_t icu,
    l4_icu_info_t * info,
    l4_utcb_t * utcb) [inline]
```

Get information about the ICU features.

Parameters

| | | |
|-----|-------------|--|
| | <i>icu</i> | The ICU object from which MSI information shall be retrieved. |
| out | <i>info</i> | Info structure to be filled with information. |
| | <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |

Returns

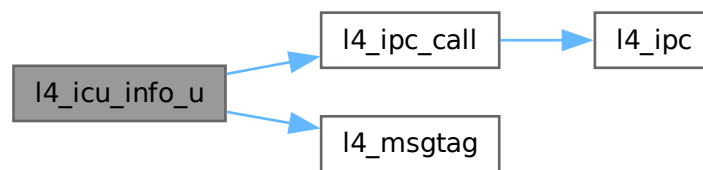
Syscall return tag

Definition at line 420 of file [icu.h](#).

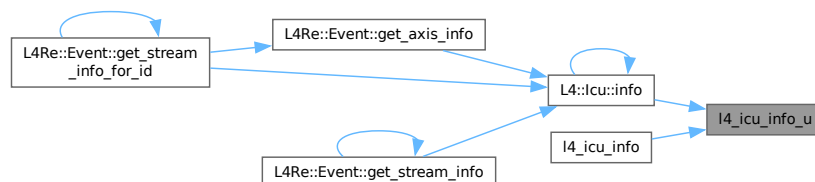
References [L4_ICU_OP_INFO](#), [l4_ipc_call\(\)](#), [L4_IPC_NEVER](#), [l4_msgtag\(\)](#), [L4_NOTHROW](#), [L4_PROTO_IRQ](#), and [l4_msg_regs_t::mr](#).

Referenced by [L4::Icu::info\(\)](#), and [l4_icu_info\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**13.1.10.7.4.5 l4_icu_mask()**

```

l4_msgtag_t l4_icu_mask (
    l4_cap_idx_t icu,
    unsigned irqnum,
    l4_umword_t * label,
    l4_timeout_t to) [inline]
  
```

Mask an IRQ line.

Parameters

| | |
|------------|--|
| <i>icu</i> | The ICU object where the IRQ line shall be masked. |
|------------|--|

| | |
|---------------|---|
| <i>irqnum</i> | IRQ line at the ICU. |
| <i>label</i> | If non-NULL, the function also performs an open wait IPC operation waiting for the next message, and the received label is returned here. |
| <i>to</i> | IPC timeout, if unsure use L4_IPC_NEVER. |

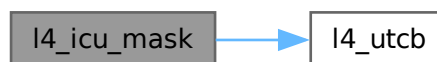
Returns

Syscall return tag. If *label* is NULL, this function performs an IPC send-only operation and there is no return value except [L4_MSGTAG_ERROR](#) indicating success or failure of the send operation. In this case use [l4_ipc_error\(\)](#) to check for errors and **do not** use [l4_error\(\)](#).

Definition at line 518 of file [icu.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



13.1.10.7.4.6 l4_icu_mask_u()

```

l4_msgtag_t l4_icu_mask_u (
    l4_cap_idx_t icu,
    unsigned irqnum,
    l4_umword_t * label,
    l4_timeout_t to,
    l4_utcb_t * utcb) [inline]
  
```

Mask an IRQ line.

Parameters

| | |
|---------------|--|
| <i>icu</i> | The ICU object where the IRQ line shall be masked. |
| <i>irqnum</i> | IRQ line at the ICU. |
| <i>label</i> | If NULL, this function is a send-only message to the ICU. If not NULL, this function will enter an open wait after sending the mask message and the received label is returned here. |
| <i>to</i> | The timeout-pair (send and receive) that shall be used for this operation. The receive timeout is used with a non-NULL <i>label</i> only. |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |

Returns

Syscall return tag. If `label` is `NULL`, this function performs an IPC send-only operation and there is no return value except `L4_MSGTAG_ERROR` indicating success or failure of the send operation. In this case use `l4_ipc_error()` to check for errors and **do not** use `l4_error()`.

Definition at line 483 of file `icu.h`.

References `L4_NOTHROW`.

Referenced by `L4::Icu::mask()`.

Here is the caller graph for this function:

**13.1.10.7.4.7 l4_icu_msi_info()**

```

l4_msgtag_t l4_icu_msi_info (
    l4_cap_idx_t icu,
    unsigned irqnum,
    l4_uint64_t source,
    l4_icu_msi_info_t * msi_info) [inline]
  
```

Get MSI info about IRQ.

Parameters

| | | |
|-----|-----------------|---|
| | <i>icu</i> | The ICU object from which MSI information shall be retrieved. |
| | <i>irqnum</i> | IRQ line at the ICU. |
| | <i>source</i> | Platform dependent requester ID for MSIs. On IA32 we use a 20bit source filter value as described in the Intel IRQ remapping specification. |
| out | <i>msi_info</i> | A <code>l4_icu_msi_info_t</code> structure receiving the address and the data value to trigger this MSI. |

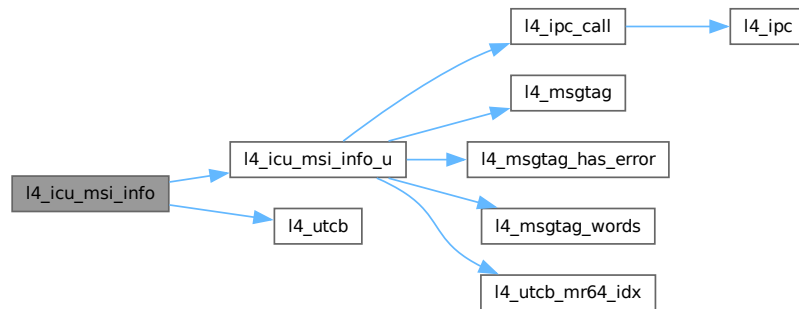
Returns

Syscall return tag

Definition at line 508 of file `icu.h`.

References `l4_icu_msi_info_u()`, `L4_NOTHROW`, and `l4_utcb()`.

Here is the call graph for this function:



13.1.10.7.4.8 l4_icu_msi_info_u()

```

l4_msgtag_t l4_icu_msi_info_u (
    l4_cap_idx_t icu,
    unsigned irqnum,
    l4_uint64_t source,
    l4_icu_msi_info_t * msi_info,
    l4_utcb_t * utcb) [inline]
  
```

Get MSI info about IRQ.

Parameters

| | | |
|-----|-----------------|---|
| | <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |
| | <i>icu</i> | The ICU object from which MSI information shall be retrieved. |
| | <i>irqnum</i> | IRQ line at the ICU. |
| | <i>source</i> | Platform dependent requester ID for MSIs. On IA32 we use a 20bit source filter value as described in the Intel IRQ remapping specification. |
| out | <i>msi_info</i> | A l4_icu_msi_info_t structure receiving the address and the data value to trigger this MSI. |

Returns

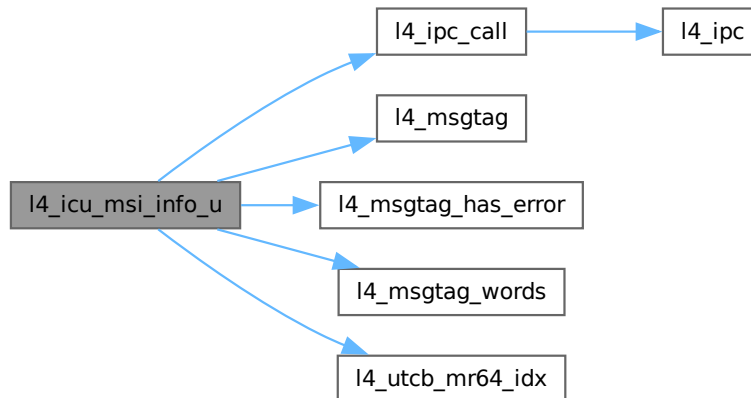
Syscall return tag

Definition at line 434 of file [icu.h](#).

References [L4_ICU_OP_MSI_INFO](#), [l4_ipc_call\(\)](#), [L4_IPC_NEVER](#), [l4_msgtag\(\)](#), [l4_msgtag_has_error\(\)](#), [l4_msgtag_words\(\)](#), [L4_NOTHROW](#), [L4_PROTO_IRQ](#), [L4_UNLIKELY](#), [l4_utcb_mr64_idx\(\)](#), [l4_msg_regs_t::mr](#), and [l4_msg_regs_t::mr64](#).

Referenced by [l4_icu_msi_info\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.1.10.7.4.9 l4_icu_set_mode()

```

l4_msgtag_t l4_icu_set_mode (
    l4_cap_idx_t icu,
    unsigned irqnum,
    l4_umword_t mode) [inline]
  
```

Set interrupt mode.

Parameters

| | |
|---------------|---|
| <i>icu</i> | The ICU object. |
| <i>irqnum</i> | IRQ line at the ICU. |
| <i>mode</i> | Mode, see L4_irq_mode . |

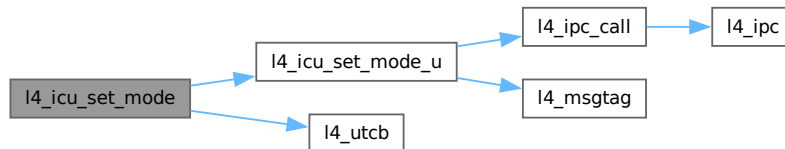
Returns

Syscall return tag

Definition at line 523 of file [icu.h](#).

References [l4_icu_set_mode_u\(\)](#), [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:

**13.1.10.7.4.10 l4_icu_set_mode_u()**

```

l4_msgtag_t l4_icu_set_mode_u (
    l4_cap_idx_t icu,
    unsigned irqnum,
    l4_umword_t mode,
    l4_utcb_t * utcb) [inline]
  
```

Set interrupt mode.

Parameters

| | |
|---------------|--|
| <i>icu</i> | The ICU object. |
| <i>irqnum</i> | IRQ line at the ICU. |
| <i>mode</i> | Mode, see L4_irq_mode . |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |

Returns

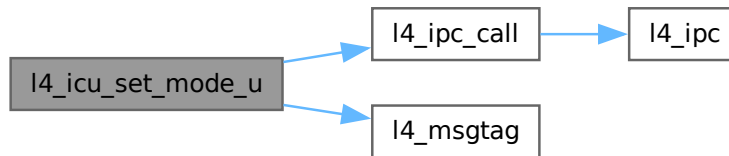
Syscall return tag

Definition at line 457 of file [icu.h](#).

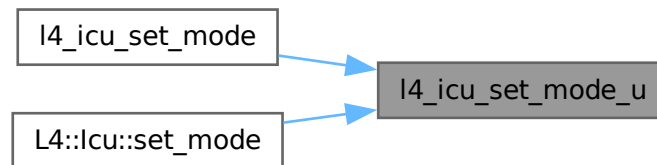
References [L4_ICU_OP_SET_MODE](#), [l4_ipc_call\(\)](#), [L4_IPC_NEVER](#), [l4_msgtag\(\)](#), [L4_NOTHROW](#), [L4_PROTO_IRQ](#), and [l4_msg_regs_t::mr](#).

Referenced by [l4_icu_set_mode\(\)](#), and [L4::l4u::set_mode\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.1.10.7.4.11 l4_icu_unbind()

```

l4_msgtag_t l4_icu_unbind (
    l4_cap_idx_t icu,
    unsigned irqnum,
    l4_cap_idx_t irq) [inline]
  
```

Remove binding of an interrupt line from the interrupt controller object.

Parameters

| | |
|---------------|---|
| <i>icu</i> | The ICU object from where the binding shall be removed. |
| <i>irqnum</i> | IRQ line at the ICU. |
| <i>irq</i> | IRQ object to remove from the ICU. |

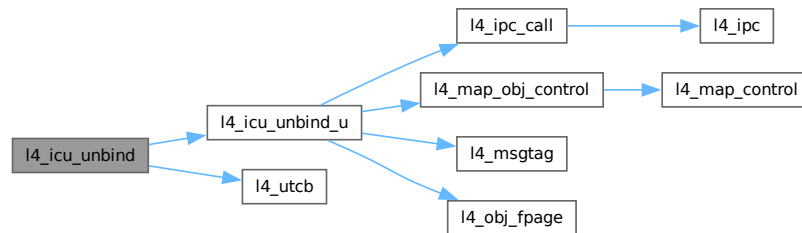
Returns

Syscall return tag

Definition at line 500 of file [icu.h](#).

References [l4_icu_unbind_u\(\)](#), [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:

**13.1.10.7.4.12 l4_icu_unbind_u()**

```

l4_msgtag_t l4_icu_unbind_u (
    l4_cap_idx_t icu,
    unsigned irqnum,
    l4_cap_idx_t irq,
    l4_utcb_t * utcb) [inline]
  
```

Remove binding of an interrupt line from the interrupt controller object.

Parameters

| | |
|---------------|--|
| <i>icu</i> | The ICU object from where the binding shall be removed. |
| <i>irqnum</i> | IRQ line at the ICU. |
| <i>irq</i> | IRQ object to remove from the ICU. |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |

Returns

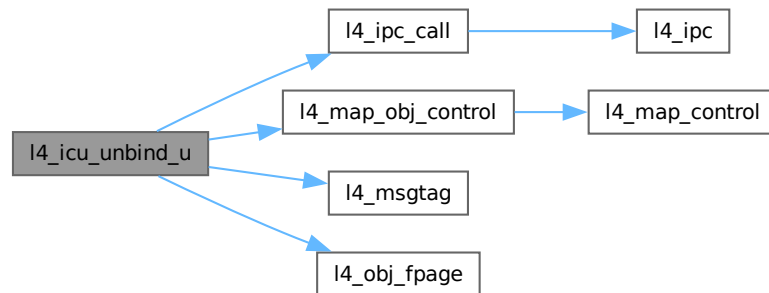
Syscall return tag

Definition at line 408 of file [icu.h](#).

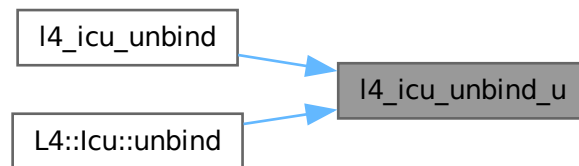
References [L4_CAP_FPAGE_RWS](#), [L4_ICU_OP_UNBIND](#), [l4_ipc_call\(\)](#), [L4_IPC_NEVER](#), [l4_map_obj_control\(\)](#), [l4_msgtag\(\)](#), [L4_NOTHROW](#), [l4_obj_fpage\(\)](#), [L4_PROTO_IRQ](#), [l4_msg_regs_t::mr](#), and [l4_fpage_t::raw](#).

Referenced by [l4_icu_unbind\(\)](#), and [L4::l4u::unbind\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.1.10.7.4.13 l4_icu_unmask()

```

l4_msgtag_t l4_icu_unmask (
    l4_cap_idx_t icu,
    unsigned irqnum,
    l4_umword_t * label,
    l4_timeout_t to) [inline]
  
```

Unmask an IRQ line.

Parameters

| | |
|---------------|---|
| <i>icu</i> | The ICU object where the IRQ line shall be unmasked. |
| <i>irqnum</i> | IRQ line at the ICU. |
| <i>label</i> | If non-NULL, the function also performs an open wait IPC operation waiting for the next message, and the received label is returned here. |
| <i>to</i> | IPC timeout, if unsure use <code>L4_IPC_NEVER</code> . |

Returns

Syscall return tag. If `label` is `NULL`, this function performs an IPC send-only operation and there is no return value except `L4_MSGTAG_ERROR` indicating success or failure of the send operation. In this case use `l4_ipc_error()` to check for errors and **do not** use `l4_error()`.

Definition at line 513 of file `icu.h`.

References `L4_NOTHROW`, and `l4_utcb()`.

Here is the call graph for this function:

**13.1.10.7.4.14 l4_icu_unmask_u()**

```

l4_msgtag_t l4_icu_unmask_u (
    l4_cap_idx_t icu,
    unsigned irqnum,
    l4_umword_t * label,
    l4_timeout_t to,
    l4_utcb_t * utcb) [inline]
  
```

Unmask the given interrupt line.

Parameters

| | |
|------------|--|
| <i>icu</i> | The ICU object where the IRQ line shall be unmasked. When the object is an IRQ, the given interrupt line is ignored and instead the line which the IRQ is bound to (if any) is unmasked. |
|------------|--|

Its counterpart for explicitly masking an interrupt line is `L4::Icu::mask()`.

Parameters

| | | |
|-----|---------------|---|
| | <i>irqnum</i> | The interrupt line that shall be unmasked. Ignored if the object is an IRQ. |
| out | <i>label</i> | If <code>NULL</code> , this is a send-only unmask. If not <code>NULL</code> , this operation enters an open wait and the <i>protected label</i> shall be received here. |
| | <i>to</i> | The timeout-pair (send and receive) that shall be used for this operation. The receive timeout is used with a non- <code>NULL label</code> only. |
| | <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See <code>l4_utcb</code> . |

Returns

Syscall return tag. If `label` is `NULL`, this function performs an IPC send-only operation and there is no return value except `L4_MSGTAG_ERROR` indicating success or failure of the send operation. In this case use `l4_ipc_error()` to check for errors and **do not** use `l4_error()`.

Definition at line 488 of file `icu.h`.

References `L4_NOTHROW`.

13.1.10.8 IRQs

C IRQ interface, see `L4::Irq` for the C++ interface.

Collaboration diagram for IRQs:



Enumerations

- enum `L4_irq_mode` {
`L4_IRQ_F_NONE` = 0 , `L4_IRQ_F_SET_MODE` = 0x1 , `L4_IRQ_F_LEVEL` = 0x2 , `L4_IRQ_F_EDGE` = 0x0 ,
`L4_IRQ_F_POS` = 0x0 , `L4_IRQ_F_NEG` = 0x4 , `L4_IRQ_F_BOTH` = 0x8 , `L4_IRQ_F_LEVEL_HIGH` =
`L4_IRQ_F_SET_MODE` | `L4_IRQ_F_LEVEL` | `L4_IRQ_F_POS` ,
`L4_IRQ_F_LEVEL_LOW` = `L4_IRQ_F_SET_MODE` | `L4_IRQ_F_LEVEL` | `L4_IRQ_F_NEG` , `L4_IRQ_F_POS_EDGE`
= `L4_IRQ_F_SET_MODE` | `L4_IRQ_F_EDGE` | `L4_IRQ_F_POS` , `L4_IRQ_F_NEG_EDGE` = `L4_IRQ_F_`
`SET_MODE` | `L4_IRQ_F_EDGE` | `L4_IRQ_F_NEG` , `L4_IRQ_F_BOTH_EDGE` = `L4_IRQ_F_SET_MODE` |
`L4_IRQ_F_EDGE` | `L4_IRQ_F_BOTH` ,
`L4_IRQ_F_MASK` = 0xf , `L4_IRQ_F_SET_WAKEUP` = 0x10 , `L4_IRQ_F_CLEAR_WAKEUP` = 0x20 }

Interrupt attributes.

Functions

- `l4_msgtag_t l4_irq_detach (l4_cap_idx_t irq) L4_NOTHROW`
Detach from an interrupt source.
- `l4_msgtag_t l4_irq_detach_u (l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW`
Detach from this interrupt.
- `l4_msgtag_t l4_irq_bind_vcpu (l4_cap_idx_t irq, l4_cap_idx_t thread, l4_umword_t cfg) L4_NOTHROW`
Bind a thread to this Irq for vCPU interrupt forwarding.
- `l4_msgtag_t l4_irq_bind_vcpu_u (l4_cap_idx_t irq, l4_cap_idx_t thread, l4_umword_t cfg, l4_utcb_t *utcb) L4_NOTHROW`
Bind a thread to this Irq for vCPU interrupt forwarding.
- `l4_msgtag_t l4_irq_trigger (l4_cap_idx_t irq) L4_NOTHROW`
Trigger an IRQ.

- [l4_msgtag_t l4_irq_trigger_u](#) ([l4_cap_idx_t](#) irq, [l4_utcb_t](#) *utcb) [L4_NOTHROW](#)
Trigger the object.
- [l4_msgtag_t l4_irq_receive](#) ([l4_cap_idx_t](#) irq, [l4_timeout_t](#) to) [L4_NOTHROW](#)
Unmask and wait for specified IRQ.
- [l4_msgtag_t l4_irq_receive_u](#) ([l4_cap_idx_t](#) irq, [l4_timeout_t](#) timeout, [l4_utcb_t](#) *utcb) [L4_NOTHROW](#)
Unmask and wait for this IRQ.
- [l4_msgtag_t l4_irq_wait](#) ([l4_cap_idx_t](#) irq, [l4_umword_t](#) *label, [l4_timeout_t](#) to) [L4_NOTHROW](#)
Unmask IRQ and wait for any message.
- [l4_msgtag_t l4_irq_wait_u](#) ([l4_cap_idx_t](#) irq, [l4_umword_t](#) *label, [l4_timeout_t](#) timeout, [l4_utcb_t](#) *utcb) [L4_NOTHROW](#)
Unmask IRQ and (open) wait for any message.
- [l4_msgtag_t l4_irq_unmask](#) ([l4_cap_idx_t](#) irq) [L4_NOTHROW](#)
Unmask IRQ.
- [l4_msgtag_t l4_irq_unmask_u](#) ([l4_cap_idx_t](#) irq, [l4_utcb_t](#) *utcb) [L4_NOTHROW](#)
Unmask this IRQ.

13.1.10.8.1 Detailed Description

C IRQ interface, see [L4::Irq](#) for the C++ interface.

The IRQ interface provides access to abstract interrupts provided by the microkernel. Interrupts may be

- hardware interrupts provided by the platform interrupt controller,
- virtual device interrupts provided by the microkernel's virtual devices (virtual serial or trace buffer) or
- virtual interrupts that can be triggered by user programs (IRQs) via [l4_irq_trigger\(\)](#).

For hardware and virtual device interrupts the `Irq` object must be bound to an interrupt source, see [Interrupt controller](#). To receive interrupts, the `Irq` object must be bound to a thread, see [l4_rcv_ep_bind_thread\(\)](#) and [l4_rcv_ep_bind_snd_destination\(\)](#).

IRQ objects can be created using a factory, see the [Factory](#) API (use [l4_factory_create_irq\(\)](#)).

Include File

```
#include <l4/sys/irq.h>
```

For the C++ interface refer to the [L4::Irq](#) API for an overview.

13.1.10.8.2 Enumeration Type Documentation

13.1.10.8.2.1 L4_irq_mode

enum `L4_irq_mode`

Interrupt attributes.

Enumerator

| | |
|------------------------------------|---|
| <code>L4_IRQ_F_NONE</code> | Flow types. None |
| <code>L4_IRQ_F_SET_MODE</code> | Valid flag, if not set, the <code>set_mode</code> operation does nothing. |
| <code>L4_IRQ_F_LEVEL</code> | Level triggered. |
| <code>L4_IRQ_F_EDGE</code> | Edge triggered. |
| <code>L4_IRQ_F_POS</code> | Positive trigger. |
| <code>L4_IRQ_F_NEG</code> | Negative trigger. |
| <code>L4_IRQ_F_BOTH</code> | Both edges trigger. |
| <code>L4_IRQ_F_LEVEL_HIGH</code> | Level high trigger. |
| <code>L4_IRQ_F_LEVEL_LOW</code> | Level low trigger. |
| <code>L4_IRQ_F_POS_EDGE</code> | Positive edge trigger. |
| <code>L4_IRQ_F_NEG_EDGE</code> | Negative edge trigger. |
| <code>L4_IRQ_F_BOTH_EDGE</code> | Both edges trigger. |
| <code>L4_IRQ_F_MASK</code> | Mask. |
| <code>L4_IRQ_F_SET_WAKEUP</code> | Wakeup source? Use irq as wakeup source |
| <code>L4_IRQ_F_CLEAR_WAKEUP</code> | Do not use irq as wakeup source. |

Definition at line 70 of file `icu.h`.

13.1.10.8.3 Function Documentation

13.1.10.8.3.1 l4_irq_bind_vcpu()

```
l4_msgtag_t l4_irq_bind_vcpu (
    l4_cap_idx_t irq,
    l4_cap_idx_t thread,
    l4_umword_t cfg) [inline]
```

Bind a thread to this Irq for vCPU interrupt forwarding.

If the interrupt is triggered, the kernel will directly inject the interrupt into the guest. This requires that the thread is currently in extended vCPU user mode. Otherwise the interrupt will stay pending and gets injected on the next vCPU user mode transition. Optionally a doorbell Irq can be registered on the thread (see `Thread::register_doorbell_irq()`) that is triggered in this case.

If a guest has acknowledged the interrupt but has not yet issued an EOI (i.e. the interrupt is in "active" state), it is not possible to bind the Irq to a new thread object. Either wait for the guest to issue the EOI or `detach()` from the current thread. In this case the interrupt will stay active in the guest and it is the responsibility of the VMM to handle the eventual EOI of the guest.

Parameters

| | |
|---------------|--|
| <i>irq</i> | The IRQ object that shall be bound. |
| <i>thread</i> | Thread object this Irq shall be bound to. |
| <i>cfg</i> | Architecture specific interrupt configuration. |

Returns

Syscall return tag

Return values

| | |
|-------------------------|---|
| <code>-L4_EPERM</code> | Insufficient permissions; see precondition. |
| <code>-L4_EBUSY</code> | Cannot bind to new thread because interrupt is active on previous thread and guest has to issue end-of-interrupt first. |
| <code>-L4_ENOSYS</code> | The kernel does not support direct interrupt forwarding. |

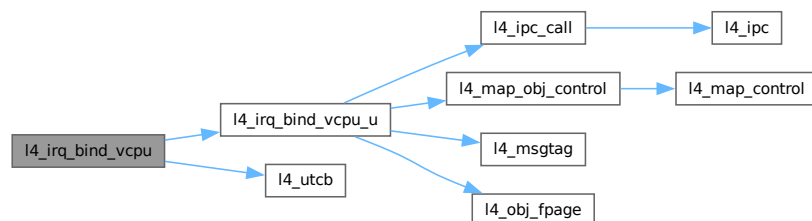
Precondition

The capabilities `irq` and `thread` both must have the permission `L4_CAP_FPAGE_S`.

Definition at line 300 of file `irq.h`.

References `l4_irq_bind_vcpu_u()`, `L4_NOTHROW`, and `l4_utcb()`.

Here is the call graph for this function:

**13.1.10.8.3.2 l4_irq_bind_vcpu_u()**

```

l4_msgtag_t l4_irq_bind_vcpu_u (
    l4_cap_idx_t irq,
    l4_cap_idx_t thread,
    l4_umword_t cfg,
    l4_utcb_t * utcb) [inline]

```

Bind a thread to this Irq for vCPU interrupt forwarding.

Parameters

| | |
|------------|--|
| <i>irq</i> | The IRQ object that shall be bound. If the interrupt is triggered, the kernel will directly inject the interrupt into the guest. This requires that the thread is currently in extended vCPU user mode. Otherwise the interrupt will stay pending and gets injected on the next vCPU user mode transition. Optionally a doorbell Irq can be registered on the thread (see <code>Thread::register_doorbell_irq()</code>) that is triggered in this case. |
|------------|--|

If a guest has acknowledged the interrupt but has not yet issued an EOI (i.e. the interrupt is in "active" state), it is not possible to bind the Irq to a new thread object. Either wait for the guest to issue the EOI or `detach()` from the current thread. In this case the interrupt will stay active in the guest and it is the responsibility of the VMM to handle the eventual EOI of the guest.

Parameters

| | |
|---------------|--|
| <i>thread</i> | Thread object this Irq shall be bound to. |
| <i>cfg</i> | Architecture specific interrupt configuration. |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |

Returns

Syscall return tag

Return values

| | |
|-------------------------|---|
| <code>-L4_EPERM</code> | Insufficient permissions; see precondition. |
| <code>-L4_EBUSY</code> | Cannot bind to the new thread because interrupt is active on previous thread and guest has to issue end-of-interrupt first. |
| <code>-L4_ENOSYS</code> | The kernel does not support direct interrupt forwarding. |

Precondition

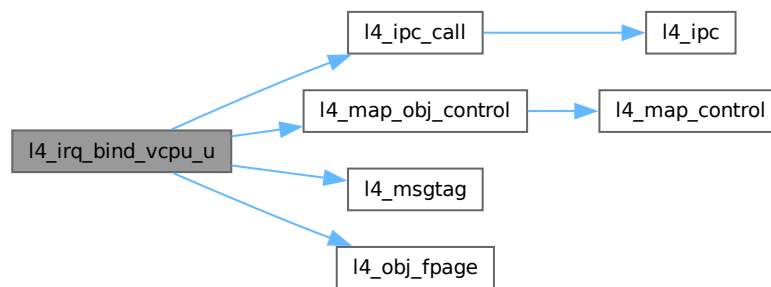
The invoked Irq capability and the capability `thread` both must have the permission [L4_CAP_FPAGE_S](#).

Definition at line [250](#) of file [irq.h](#).

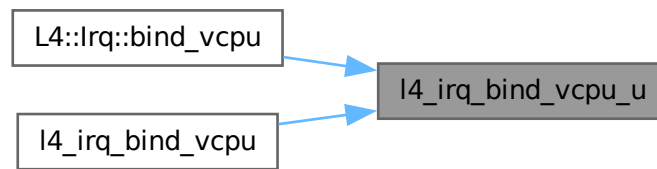
References [L4_CAP_FPAGE_RWS](#), [l4_ipc_call\(\)](#), [L4_IPC_NEVER](#), [l4_map_obj_control\(\)](#), [l4_msgtag\(\)](#), [L4_NOTHROW](#), [l4_obj_fpage\(\)](#), [L4_PROTO_IRQ_SENDER](#), [l4_msg_regs_t::mr](#), and [l4_fpage_t::raw](#).

Referenced by [L4::Irq::bind_vcpu\(\)](#), and [l4_irq_bind_vcpu\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.1.10.8.3.3 l4_irq_detach()

```
l4_msgtag_t l4_irq_detach (
    l4_cap_idx_t irq) [inline]
```

Detach from an interrupt source.

Parameters

| | |
|------------|--|
| <i>irq</i> | The IRQ object that shall be detached. |
|------------|--|

Returns

Syscall return tag

Return values

| | |
|-----------|--|
| 0 | Successfully detached, there was no interrupt pending. |
| 1 | Successfully detached, there was an interrupt pending. |
| 2 | Successfully detached, an active vIRQ was abandoned. |
| -L4_EPERM | Insufficient permissions; see precondition. |

Precondition

The capability `irq` must have the permission `L4_CAP_FPAGE_S`.

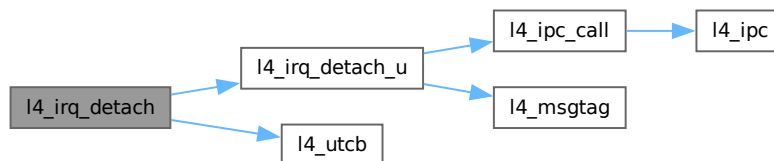
Examples

[examples/sys/isr/main.c](#).

Definition at line 294 of file `irq.h`.

References [l4_irq_detach_u\(\)](#), [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



13.1.10.8.3.4 l4_irq_detach_u()

```

l4_msgtag_t l4_irq_detach_u (
    l4_cap_idx_t irq,
    l4_utcb_t * utcb) [inline]

```

Detach from this interrupt.

Parameters

| | |
|-------------|--|
| <i>irq</i> | The IRQ object that shall be detached. |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |

Returns

Syscall return tag

Return values

| | |
|-----------|--|
| 0 | Successfully detached, there was no interrupt pending. |
| 1 | Successfully detached, there was an interrupt pending. |
| 2 | Successfully detached, an active vIRQ was abandoned. |
| -L4_EPERM | Insufficient permissions; see precondition. |

Precondition

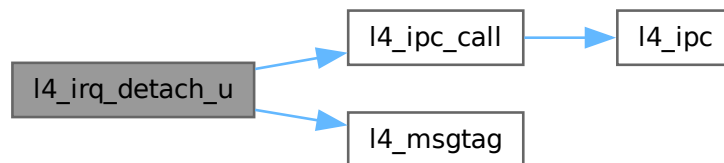
The invoked Irq capability must have the permission [L4_CAP_FPAGE_S](#).

Definition at line 242 of file [irq.h](#).

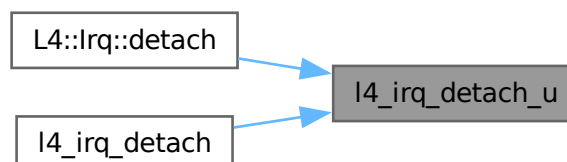
References [l4_ipc_call\(\)](#), [L4_IPC_NEVER](#), [l4_msgtag\(\)](#), [L4_NOTHROW](#), and [L4_PROTO_IRQ_SENDER](#).

Referenced by [L4::Irq::detach\(\)](#), and [l4_irq_detach\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.1.10.8.3.5 l4_irq_receive()

```
l4_msgtag_t l4_irq_receive (
    l4_cap_idx_t irq,
    l4_timeout_t to) [inline]
```

Unmask and wait for specified IRQ.

Parameters

| | |
|------------|--|
| <i>irq</i> | The IRQ object that shall be unmasked. |
| <i>to</i> | Timeout. |

Returns

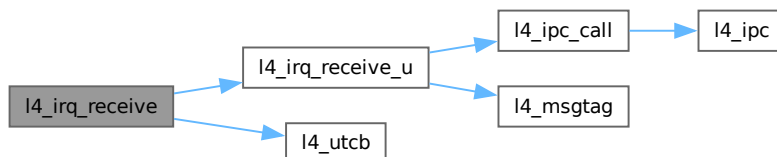
Syscall return tag

Definition at line 313 of file [irq.h](#).

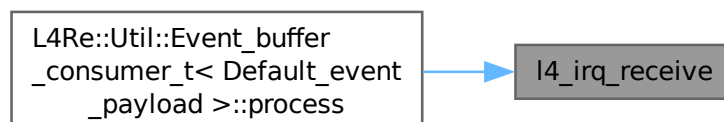
References [l4_irq_receive_u\(\)](#), [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Referenced by [L4Re::Util::Event_buffer_consumer_t< Default_event_payload >::process\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.1.10.8.3.6 l4_irq_receive_u()

```
l4_msgtag_t l4_irq_receive_u (
    l4_cap_idx_t irq,
    l4_timeout_t timeout,
    l4_utcb_t * utcb) [inline]
```

Unmask and wait for this IRQ.

Parameters

| | |
|----------------|--|
| <i>irq</i> | The IRQ object that shall be unmasked. |
| <i>timeout</i> | Timeout. |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |

Returns

Syscall return tag

Note

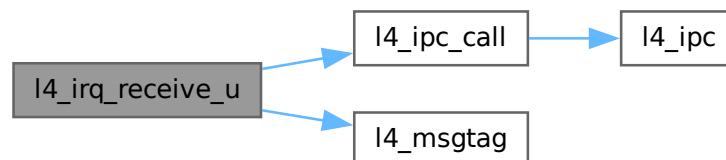
If this is the function normally used for your IRQs consider using [L4::Semaphore](#) instead of [L4::Irq](#).

Definition at line 270 of file [irq.h](#).

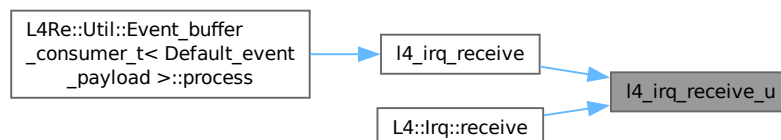
References [l4_ipc_call\(\)](#), [l4_msgtag\(\)](#), [L4_NOTHROW](#), and [L4_PROTO_IRQ](#).

Referenced by [l4_irq_receive\(\)](#), and [L4::Irq::receive\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.1.10.8.3.7 l4_irq_trigger()

```
l4_msgtag_t l4_irq_trigger (
    l4_cap_idx_t irq) [inline]
```

Trigger an IRQ.

Parameters

| | |
|------------|---|
| <i>irq</i> | The IRQ object that shall be triggered. |
|------------|---|

Returns

Syscall return tag.

Note that this function is a send only operation, i.e. there is no return value except for a failed send operation. Especially [l4_error\(\)](#) will return an error value from the message tag which still contains the IRQ protocol used for the send operation.

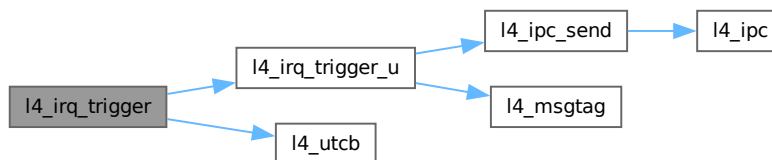
Use [l4_ipc_error\(\)](#) to check for (send) errors.

Definition at line 307 of file [irq.h](#).

References [l4_irq_trigger_u\(\)](#), [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Referenced by [l4_semaphore_up\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.1.10.8.3.8 l4_irq_trigger_u()

```
l4_msgtag_t l4_irq_trigger_u (
    l4_cap_idx_t irq,
    l4_utcb_t * utcb) [inline]
```

Trigger the object.

Parameters

| | |
|-------------|--|
| <i>irq</i> | The IRQ object that shall be triggered. |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |

Returns

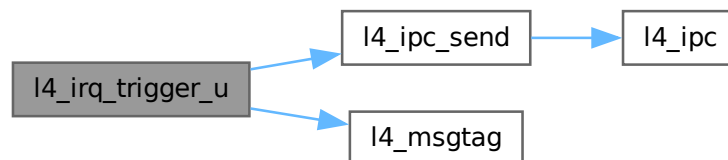
Syscall return tag for a send-only operation, this means there is no return value except [L4_MSGTAG_ERROR](#) indicating success or failure of the send operation. Use [l4_ipc_error\(\)](#) to check for errors and **do not** use [l4_error\(\)](#).

Definition at line 263 of file [irq.h](#).

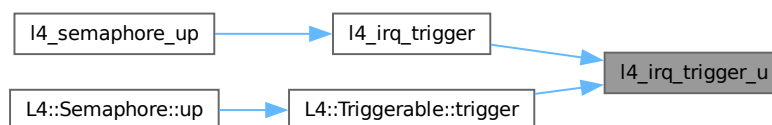
References [L4_IPC_BOTH_TIMEOUT_0](#), [l4_ipc_send\(\)](#), [l4_msgtag\(\)](#), [L4_NOTHROW](#), and [L4_PROTO_IRQ](#).

Referenced by [l4_irq_trigger\(\)](#), and [L4::Triggerable::trigger\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.1.10.8.3.9 l4_irq_unmask()

```
l4_msgtag_t l4_irq_unmask (
    l4_cap_idx_t irq) [inline]
```

Unmask IRQ.

Parameters

| | |
|------------|--|
| <i>irq</i> | The IRQ object that shall be unmasked. |
|------------|--|

Returns

Syscall return tag

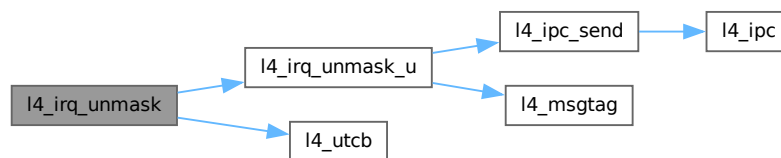
Note

[l4_irq_wait\(\)](#) and [l4_irq_receive\(\)](#) are doing the unmask themselves.

Definition at line 326 of file [irq.h](#).

References [l4_irq_unmask_u\(\)](#), [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



13.1.10.8.3.10 l4_irq_unmask_u()

```
l4_msgtag_t l4_irq_unmask_u (
    l4_cap_idx_t irq,
    l4_utcb_t * utcb) [inline]
```

Unmask this IRQ.

Parameters

| | |
|-------------|--|
| <i>irq</i> | The IRQ object that shall be unmasked. |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |

Returns

Syscall return tag for a send-only operation, this means there is no return value except [L4_MSGTAG_ERROR](#) indicating success or failure of the send operation. Use [l4_ipc_error\(\)](#) to check for errors and **do not** use [l4_error\(\)](#).

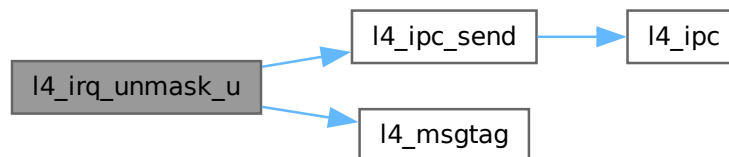
`Irq::wait()` and `Irq::receive()` operations already include an `unmask()`, do not use an extra `unmask()` in these cases.

Definition at line 286 of file [irq.h](#).

References [L4_IPC_NEVER](#), [l4_ipc_send\(\)](#), [l4_msgtag\(\)](#), [L4_NOTHROW](#), and [L4_PROTO_IRQ](#).

Referenced by [l4_irq_unmask\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**13.1.10.8.3.11 l4_irq_wait()**

```

l4_msgtag_t l4_irq_wait (
    l4_cap_idx_t irq,
    l4_umword_t * label,
    l4_timeout_t to) [inline]
  
```

Unmask IRQ and wait for any message.

Parameters

| | |
|--------------|--|
| <i>irq</i> | The IRQ object that shall be unmasked. |
| <i>label</i> | Receive label. |
| <i>to</i> | Timeout. |

Returns

Syscall return tag

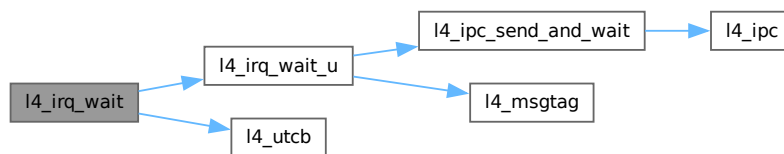
Examples

[examples/sys/isr/main.c](#).

Definition at line 319 of file [irq.h](#).

References [l4_irq_wait_u\(\)](#), [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



13.1.10.8.3.12 l4_irq_wait_u()

```

l4_msgtag_t l4_irq_wait_u (
    l4_cap_idx_t irq,
    l4_umword_t * label,
    l4_timeout_t timeout,
    l4_utcb_t * utcb) [inline]

```

Unmask IRQ and (open) wait for any message.

Parameters

| | |
|----------------|--|
| <i>irq</i> | The IRQ object that shall be unmasked. |
| <i>label</i> | The <i>protected label</i> shall be received here. |
| <i>timeout</i> | Timeout. |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |

Returns

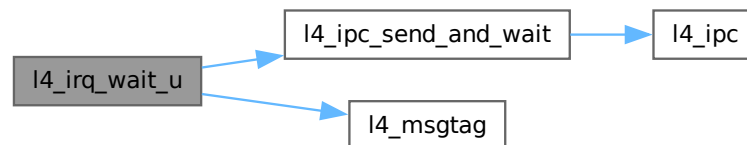
Syscall return tag

Definition at line 277 of file [irq.h](#).

References [l4_ipc_send_and_wait\(\)](#), [l4_msgtag\(\)](#), [L4_NOTHROW](#), and [L4_PROTO_IRQ](#).

Referenced by [l4_irq_wait\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**13.1.10.9 Platform Control C API**

C interface for controlling platform-wide properties, see [L4::Platform_control](#) for the C++ interface.

Collaboration diagram for Platform Control C API:



Functions

- [l4_msgtag_t l4_platform_ctl_set_task_asid](#) ([l4_cap_idx_t](#) pfc, [l4_cap_idx_t](#) task, [l4_umword_t](#) asid) [L4_NOTHROW](#)
Set ASID of task.
- [l4_msgtag_t l4_platform_ctl_system_suspend](#) ([l4_cap_idx_t](#) pfc, [l4_umword_t](#) extras) [L4_NOTHROW](#)
Enter suspend to RAM.
- [l4_msgtag_t l4_platform_ctl_system_shutdown](#) ([l4_cap_idx_t](#) pfc, [l4_umword_t](#) reboot) [L4_NOTHROW](#)
Shutdown or reboot the system.
- [l4_msgtag_t l4_platform_ctl_cpu_allow_shutdown](#) ([l4_cap_idx_t](#) pfc, [l4_umword_t](#) phys_id, [l4_umword_t](#) enable) [L4_NOTHROW](#)
Allow a CPU to be shut down.
- [l4_msgtag_t l4_platform_ctl_cpu_enable](#) ([l4_cap_idx_t](#) pfc, [l4_umword_t](#) phys_id) [L4_NOTHROW](#)
Enable an offline CPU.
- [l4_msgtag_t l4_platform_ctl_cpu_disable](#) ([l4_cap_idx_t](#) pfc, [l4_umword_t](#) phys_id) [L4_NOTHROW](#)
Disable an online CPU.

13.1.10.9.1 Detailed Description

C interface for controlling platform-wide properties, see [L4::Platform_control](#) for the C++ interface.

Include File

```
#include <l4/sys/platform_control.h>
```

The API allows a client to suspend, reboot or shutdown the system.

For the C++ interface refer to [L4::Platform_control](#)

13.1.10.9.2 Function Documentation

13.1.10.9.2.1 l4_platform_ctl_cpu_allow_shutdown()

```
l4\_msgtag\_t l4_platform_ctl_cpu_allow_shutdown (
    l4\_cap\_idx\_t pfc,
    l4\_umword\_t phys_id,
    l4\_umword\_t enable) [inline]
```

Allow a CPU to be shut down.

Parameters

| | |
|----------------|--|
| <i>pfc</i> | Capability selector for the platform-control object. |
| <i>phys_id</i> | Physical CPU id of CPU (e.g. local APIC id) to enable. |
| <i>enable</i> | Allow shutdown when 1, disallow when 0. |

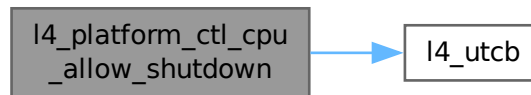
Returns

Syscall return tag

Definition at line 242 of file [platform_control.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:

**13.1.10.9.2.2 l4_platform_ctl_cpu_disable()**

```

l4_msgtag_t l4_platform_ctl_cpu_disable (
    l4_cap_idx_t pfc,
    l4_umword_t phys_id) [inline]
  
```

Disable an online CPU.

Parameters

| | |
|----------------|---|
| <i>pfc</i> | Capability to the platform control object. |
| <i>phys_id</i> | Physical CPU id of CPU (e.g. local APIC id) to disable. |

Returns

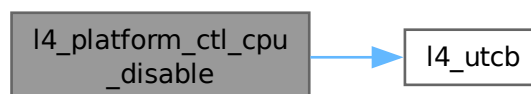
System call message tag

This function is currently only supported on the ARM EXYNOS platform.

Definition at line 281 of file [platform_control.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



13.1.10.9.2.3 l4_platform_ctl_cpu_enable()

```
l4_msgtag_t l4_platform_ctl_cpu_enable (
    l4_cap_idx_t pfc,
    l4_umword_t phys_id) [inline]
```

Enable an offline CPU.

Parameters

| | |
|----------------|--|
| <i>pfc</i> | Capability to the platform control object. |
| <i>phys_id</i> | Physical CPU id of CPU (e.g. local APIC id) to enable. |

Returns

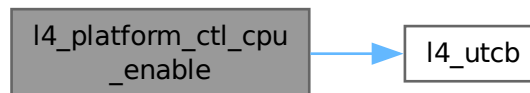
System call message tag

This function is currently only supported on the ARM EXYNOS platform.

Definition at line 274 of file [platform_control.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



13.1.10.9.2.4 l4_platform_ctl_set_task_asid()

```
l4_msgtag_t l4_platform_ctl_set_task_asid (
    l4_cap_idx_t pfc,
    l4_cap_idx_t task,
    l4_umword_t asid) [inline]
```

Set ASID of task.

On Cortex-R52 platforms, it might be necessary to control the VMID of a task or virtual machine explicitly. The IOMPU on such platforms will use it for further access control of device memory accesses. A privileged component can use this call to control the value.

The caller must have write permissions to the destination task.

Parameters

| | |
|--------------------|--|
| <i>pf</i> <i>c</i> | Capability selector for the platform-control object. |
| <i>task</i> | Capability selector of destination task |
| <i>asid</i> | New ASID value |

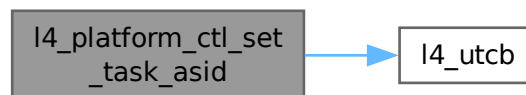
Returns

Syscall return tag

Definition at line 62 of file [__platform_control-arm.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:

**13.1.10.9.2.5 l4_platform_ctl_system_shutdown()**

```
l4_msgtag_t l4_platform_ctl_system_shutdown (  
    l4_cap_idx_t pfc,  
    l4_umword_t reboot) [inline]
```

Shutdown or reboot the system.

Parameters

| | |
|--------------------|--|
| <i>pf</i> <i>c</i> | Capability selector for the platform-control object. |
| <i>reboot</i> | Shutdown when 0, or reboot when 1. |

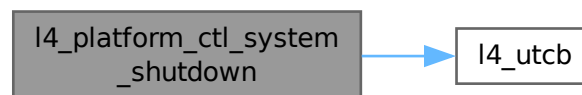
Returns

Syscall return tag

Definition at line 221 of file [platform_control.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:

**13.1.10.9.2.6 l4_platform_ctl_system_suspend()**

```

l4_msgtag_t l4_platform_ctl_system_suspend (
    l4_cap_idx_t pfc,
    l4_umword_t extras) [inline]
  
```

Enter suspend to RAM.

Precondition

Must only be invoked on the boot CPU. Furthermore it must be ensured that the invoking thread is not migrated to a different CPU during the suspend.

Parameters

| | |
|---------------|---|
| <i>pfc</i> | Capability selector for the platform-control object. |
| <i>extras</i> | Some extra platform-specific information needed to enter suspend to RAM. On x86 platforms and when using the Platform_control object provided by Fiasco, the value defines the sleep state. The sleep states are defined in the ACPI table. Other platforms as well as lo's Platform_control object don't make use of this value at the moment. |

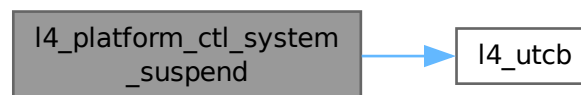
Returns

Syscall return tag

Definition at line 214 of file [platform_control.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:

**13.1.10.10 Scheduler**

C interface of the Scheduler kernel object, see [L4::Scheduler](#) for the C++ interface.

Collaboration diagram for Scheduler:

**Data Structures**

- struct [l4_sched_cpu_set_t](#)
CPU sets.
- struct [l4_sched_param_t](#)
Scheduler parameter set.

Typedefs

- typedef struct [l4_sched_cpu_set_t](#) **[l4_sched_cpu_set_t](#)**
CPU sets.
- typedef struct [l4_sched_param_t](#) **[l4_sched_param_t](#)**
Scheduler parameter set.

Enumerations

- enum [L4_scheduler_classes](#) { [L4_SCHEDULER_CLASS_FIXED_PRIO](#) = 1UL << 1, [L4_SCHEDULER_CLASS_WFQ](#) = 1UL << 2 }

Supported scheduler classes.

- enum [L4_scheduler_ops](#) { [L4_SCHEDULER_INFO_OP](#) = 0UL, [L4_SCHEDULER_RUN_THREAD_OP](#) = 1UL, [L4_SCHEDULER_IDLE_TIME_OP](#) = 2UL }

Operations on the Scheduler object.

Functions

- [l4_sched_cpu_set_t l4_sched_cpu_set](#) ([l4_umword_t](#) offset, unsigned char granularity, [l4_umword_t](#) map=1) [L4_NOTHROW](#)
- [l4_msgtag_t l4_scheduler_info](#) ([l4_cap_idx_t](#) scheduler, [l4_umword_t](#) *cpu_max, [l4_sched_cpu_set_t](#) *cpus) [L4_NOTHROW](#))
Get scheduler information.
- [l4_msgtag_t l4_scheduler_info_with_classes](#) ([l4_cap_idx_t](#) scheduler, [l4_umword_t](#) *cpu_max, [l4_sched_cpu_set_t](#) *cpus, [l4_umword_t](#) *sched_classes) [L4_NOTHROW](#))
Get scheduler information.
- [l4_sched_param_t l4_sched_param](#) (unsigned prio, [l4_umword_t](#) quantum=0) [L4_NOTHROW](#)
Construct scheduler parameter.
- [l4_msgtag_t l4_scheduler_run_thread](#) ([l4_cap_idx_t](#) scheduler, [l4_cap_idx_t](#) thread, [l4_sched_param_t](#) const *sp) [L4_NOTHROW](#))
Run a thread on a Scheduler.
- [l4_msgtag_t l4_scheduler_idle_time](#) ([l4_cap_idx_t](#) scheduler, [l4_sched_cpu_set_t](#) const *cpus, [l4_kernel_clock_t](#) *us) [L4_NOTHROW](#))
Query the idle time (in μ s) of a CPU.
- int [l4_scheduler_is_online](#) ([l4_cap_idx_t](#) scheduler, [l4_umword_t](#) cpu) [L4_NOTHROW](#)
Query if a CPU is online.

13.1.10.10.1 Detailed Description

C interface of the Scheduler kernel object, see [L4::Scheduler](#) for the C++ interface.

The Scheduler interface allows a client to manage CPU resources. The API provides functions to query scheduler information, check the online state of CPUs, query CPU idle time and to start threads on defined CPU sets.

The scheduler offers a virtual device IRQ which triggers when the number of online cores changes, e.g. due to hotplug events. In contrast to hardware IRQs, this IRQ implements a limited functionality:

- Only IRQ line 0 is supported, no MSI vectors.
- The IRQ is edge-triggered and the IRQ mode cannot be changed.
- As the IRQ is edge-triggered, it does not have to be explicitly unmasked.

It depends on the platform, which hotplug events actually trigger the IRQ. Many platforms only support triggering the IRQ when a CPU core different from the boot CPU goes online.

Include File

```
#include <l4/sys/scheduler.h>
```

13.1.10.10.2 Enumeration Type Documentation

13.1.10.10.2.1 L4_scheduler_classes

enum [L4_scheduler_classes](#)

Supported scheduler classes.

Enumerator

| | |
|-------------------------------|----------------------------------|
| L4_SCHEDULER_CLASS_FIXED_PRIO | Fixed-priority scheduler. |
| L4_SCHEDULER_CLASS_WFQ | Weighted fair queuing scheduler. |

Definition at line 46 of file [scheduler.h](#).

13.1.10.10.2.2 L4_scheduler_ops

enum [L4_scheduler_ops](#)

Operations on the Scheduler object.

Enumerator

| | |
|----------------------------|------------------------------------|
| L4_SCHEDULER_INFO_OP | Query infos about the scheduler. |
| L4_SCHEDULER_RUN_THREAD_OP | Run a thread on this scheduler. |
| L4_SCHEDULER_IDLE_TIME_OP | Query idle time for the scheduler. |

Definition at line 269 of file [scheduler.h](#).

13.1.10.10.3 Function Documentation

13.1.10.10.3.1 l4_sched_cpu_set()

```
l4_sched_cpu_set_t l4_sched_cpu_set (
    l4_umword_t offset,
    unsigned char granularity,
    l4_umword_t map = 1) [inline]
```

Parameters

| | |
|--------------------|--|
| <i>offset</i> | Offset. Must be a multiple of 2 ^{granularity} . |
| <i>granularity</i> | Granularity in log2 notation. |
| <i>map</i> | Bitmap of CPUs, defaults to 1 in C++. |

Returns

CPU set.

Examples

[examples/sys/migrate/thread_migrate.cc](#).

Definition at line 279 of file [scheduler.h](#).

References [l4_sched_cpu_set_t::gran_offset](#), [L4_NOTHROW](#), and [l4_sched_cpu_set_t::map](#).

Referenced by [l4_sched_param\(\)](#).

Here is the caller graph for this function:

**13.1.10.10.3.2 l4_sched_param()**

```

l4_sched_param_t l4_sched_param (
    unsigned prio,
    l4_umword_t quantum = 0) [inline]
  
```

Construct scheduler parameter.

Parameters

| | |
|----------------|-----------------------------|
| <i>prio</i> | Thread priority (1-255). |
| <i>quantum</i> | Timeslice in micro seconds. |

The [l4_sched_param_t::affinity](#) of the returned value contains all CPUs.

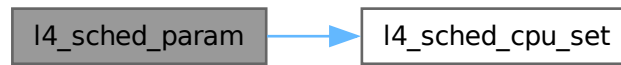
Examples

[examples/sys/aliens/main.c](#), [examples/sys/migrate/thread_migrate.cc](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 289 of file [scheduler.h](#).

References [l4_sched_param_t::affinity](#), [L4_NOTHROW](#), [l4_sched_cpu_set\(\)](#), [l4_sched_param_t::prio](#), and [l4_sched_param_t::quantum](#).

Here is the call graph for this function:



13.1.10.10.3.3 l4_scheduler_idle_time()

```

l4_msgtag_t l4_scheduler_idle_time (
    l4_cap_idx_t scheduler,
    l4_sched_cpu_set_t const * cpus,
    l4_kernel_clock_t * us) [inline]
  
```

Query the idle time (in μ s) of a CPU.

Parameters

| | | |
|-----|------------------|---|
| | <i>scheduler</i> | Scheduler object. |
| | <i>cpus</i> | Set of CPUs to query. Only the idle time of the first selected CPU in <code>cpus.map</code> is queried. |
| out | <i>us</i> | Idle time of queried CPU in μ s. |

Return values

| | |
|------------|-----------------------------------|
| 0 | Success. |
| -L4_EINVAL | Invalid CPU requested in cpu set. |

This function retrieves the idle time in μ s of the first selected CPU in `cpus.map`. The idle time is the accumulated time a CPU has spent in the idle thread since its last reset. To calculate a load estimate l one has to retrieve the idle time at the beginning ($i1$) and the end ($i2$) of a known time interval t . The load is then calculated as $l = 1 - (i2 - i1)/t$.

The idle time is only defined for online CPUs. Reading the idle time from offline CPUs is undefined and may result in either getting -L4_EINVAL or calculating an estimated (incorrect) load of 1.

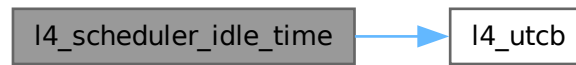
Note

The idle time statistics of remote CPUs is updated on context switch events only, hence may not be up-to-date when requested cross-CPU. To get up-to-date idle time you should use a thread running on the same CPU of which the idle time is requested.

Definition at line 403 of file [scheduler.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



13.1.10.10.3.4 l4_scheduler_info()

```

l4_msgtag_t l4_scheduler_info (
    l4_cap_idx_t scheduler,
    l4_umword_t * cpu_max,
    l4_sched_cpu_set_t * cpus) [inline]
  
```

Get scheduler information.

Parameters

| | | |
|---------|------------------|--|
| | <i>scheduler</i> | Scheduler object. |
| out | <i>cpu_max</i> | Maximum number of CPUs ever available. Optional, can be NULL. |
| in, out | <i>cpus</i> | <i>cpus.offset</i> is first CPU of interest. <i>cpus.granularity</i> (see l4_sched_cpu_set_t). <i>cpus.map</i> Bitmap of online CPUs. Must not be NULL. |

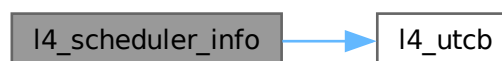
Return values

| | |
|------------|---|
| 0 | Success. |
| -L4_ERANGE | The given CPU offset is larger than the maximum number of CPUs. |

Definition at line 381 of file [scheduler.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



13.1.10.10.3.5 l4_scheduler_info_with_classes()

```
l4_msgtag_t l4_scheduler_info_with_classes (
    l4_cap_idx_t scheduler,
    l4_umword_t * cpu_max,
    l4_sched_cpu_set_t * cpus,
    l4_umword_t * sched_classes) [inline]
```

Get scheduler information.

Parameters

| | | |
|---------|----------------------|--|
| | <i>scheduler</i> | Scheduler object. |
| out | <i>cpu_max</i> | Maximum number of CPUs ever available. Optional, can be NULL. |
| in, out | <i>cpus</i> | <i>cpus.offset</i> is first CPU of interest. <i>cpus.granularity</i> (see l4_sched_cpu_set_t). <i>cpus.map</i> Bitmap of online CPUs. Must not be NULL. |
| out | <i>sched_classes</i> | A bitmap of available scheduling classes (see L4_scheduler_classes). Optional, can be NULL. |

Return values

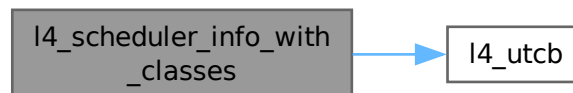
| | |
|------------|---|
| 0 | Success. |
| -L4_ERANGE | The given CPU offset is larger than the maximum number of CPUs. |

This function delivers the same information as [l4_scheduler_info](#) plus the available scheduler classes (see [L4_scheduler_classes](#)).

Definition at line 388 of file [scheduler.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



13.1.10.10.3.6 l4_scheduler_is_online()

```
int l4_scheduler_is_online (
    l4_cap_idx_t scheduler,
    l4_umword_t cpu) [inline]
```

Query if a CPU is online.

Parameters

| | |
|------------------|---|
| <i>scheduler</i> | Scheduler object. |
| <i>cpu</i> | CPU number whose online status should be queried. |

Return values

| | |
|--------------|--------------------|
| <i>true</i> | The CPU is online. |
| <i>false</i> | The CPU is offline |

Definition at line 410 of file [scheduler.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



13.1.10.10.3.7 l4_scheduler_run_thread()

```

l4_msgtag_t l4_scheduler_run_thread (
    l4_cap_idx_t scheduler,
    l4_cap_idx_t thread,
    l4_sched_param_t const * sp) [inline]
  
```

Run a thread on a Scheduler.

Parameters

| | |
|------------------|----------------------------------|
| <i>scheduler</i> | Scheduler object. |
| <i>thread</i> | Capability of the thread to run. |
| <i>sp</i> | Scheduling parameters. |

Return values

| | |
|-------------------|---|
| <i>0</i> | Success. |
| <i>-L4_EINVAL</i> | Invalid size of the scheduling parameter. |

This function launches a thread on a CPU determined by the scheduling parameter `sp.affinity`. A thread can be intentionally stopped by migrating it on an offline or an invalid CPU. The thread is only guaranteed to run if the CPU it is migrated to is currently online.

Note

If the target CPU is currently not online, there is no guarantee that the thread will ever run, even if the CPU comes online later on.

A scheduler may impose a policy with regard to selecting CPUs. However the scheduler is required to ensure the following two properties:

- Two threads with disjoint CPU sets must be scheduled to different CPUs.
- Two threads with identical CPU sets selecting only a single CPU must be scheduled to the same CPU.

Examples

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 396 of file [scheduler.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:

**13.1.10.11 Kernel-provided semaphore**

C semaphore interface, see [L4::Semaphore](#) for the C++ interface.

Collaboration diagram for Kernel-provided semaphore:

**Functions**

- [l4_msgtag_t l4_semaphore_up \(l4_cap_idx_t sem\) L4_NOTHROW](#)
Semaphore up operation (wrapper for trigger()).
- [l4_msgtag_t l4_semaphore_down \(l4_cap_idx_t sem, l4_timeout_t timeout\) L4_NOTHROW](#)
Semaphore down operation.

13.1.10.11.1 Detailed Description

C semaphore interface, see [L4::Semaphore](#) for the C++ interface.

Include File

```
#include <l4/sys/semaphore.h>
```

13.1.10.11.2 Function Documentation

13.1.10.11.2.1 l4_semaphore_down()

```
l4_msgtag_t l4_semaphore_down (
    l4_cap_idx_t sem,
    l4_timeout_t timeout) [inline]
```

Semaphore down operation.

Parameters

| | |
|----------------|---|
| <i>sem</i> | Semaphore object. |
| <i>timeout</i> | Timeout for blocking the semaphore down operation. Note: The receive timeout of this timeout-pair is significant for blocking, the send part is usually non-blocking. |

Returns

Syscall return tag. Use [l4_error\(\)](#) to check for errors.

Return values

| | |
|------------------------|---|
| <code>-L4_EPERM</code> | Insufficient permissions; see precondition. |
|------------------------|---|

Precondition

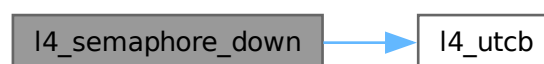
The capability `sem` must have the permission [L4_CAP_FPAGE_S](#).

This method decrements the semaphore counter by one, or blocks if the counter is already zero, until either a timeout or cancel condition hits or the counter is increased by an `up()` operation.

Definition at line 100 of file [semaphore.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



13.1.10.11.2.2 l4_semaphore_up()

```
l4_msgtag_t l4_semaphore_up (
    l4_cap_idx_t sem) [inline]
```

Semaphore up operation (wrapper for trigger()).

Parameters

| | |
|------------|-------------------|
| <i>sem</i> | Semaphore object. |
|------------|-------------------|

Returns

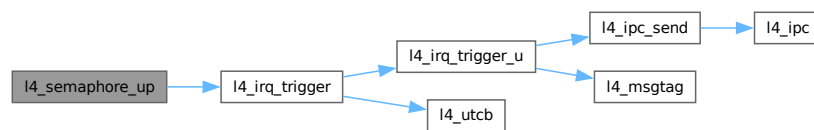
Send-only IPC message return tag. Use [l4_ipc_error\(\)](#) to check for errors, do **not** use [l4_error\(\)](#).

Increases the semaphore counter by one if it is smaller than an unspecified limit. The unspecified limit is guaranteed to be at least $2^{31}-1$.

Definition at line 45 of file [semaphore.h](#).

References [l4_irq_trigger\(\)](#), and [L4_NOTHROW](#).

Here is the call graph for this function:



13.1.10.12 Task

C interface of the Task kernel object, see [L4::Task](#) for the C++ interface.

Collaboration diagram for Task:



Enumerations

- enum [l4_unmap_flags_t](#) { [L4_FP_ALL_SPACES](#) , [L4_FP_DELETE_OBJ](#) , [L4_FP_OTHER_SPACES](#) }
Flags for the unmap operation.

Functions

- [l4_msgtag_t l4_task_vgicc_map](#) ([l4_cap_idx_t](#) task, [l4_fpage_t](#) vgicc_fpage) [L4_NOTHROW](#)
Map the GIC virtual CPU interface page to the task in case Fiasco detected a GIC version 2.
- [l4_msgtag_t l4_task_map](#) ([l4_cap_idx_t](#) dst_task, [l4_cap_idx_t](#) src_task, [l4_fpage_t](#) snd_fpage, [l4_umword_t](#) snd_base) [L4_NOTHROW](#)
Map resources available in the source task to a destination task.
- [l4_msgtag_t l4_task_unmap](#) ([l4_cap_idx_t](#) task, [l4_fpage_t](#) fpage, [l4_umword_t](#) map_mask) [L4_NOTHROW](#)
Revoke rights from the task.
- [l4_msgtag_t l4_task_unmap_batch](#) ([l4_cap_idx_t](#) task, [l4_fpage_t](#) const *fpages, unsigned num_fpages, [l4_umword_t](#) map_mask) [L4_NOTHROW](#)
Revoke rights from a task.
- [l4_msgtag_t l4_task_delete_obj](#) ([l4_cap_idx_t](#) task, [l4_cap_idx_t](#) obj) [L4_NOTHROW](#)
Release capability and delete object.
- [l4_msgtag_t l4_task_release_cap](#) ([l4_cap_idx_t](#) task, [l4_cap_idx_t](#) cap) [L4_NOTHROW](#)
Release object capability.
- [l4_msgtag_t l4_task_cap_valid](#) ([l4_cap_idx_t](#) task, [l4_cap_idx_t](#) cap) [L4_NOTHROW](#)
Check whether a capability is present (refers to an object).
- [l4_msgtag_t l4_task_cap_equal](#) ([l4_cap_idx_t](#) task, [l4_cap_idx_t](#) cap_a, [l4_cap_idx_t](#) cap_b) [L4_NOTHROW](#)
Test whether two capabilities point to the same object with the same permissions (only considering selected permissions).
- [l4_msgtag_t l4_task_add_ku_mem](#) ([l4_cap_idx_t](#) task, [l4_fpage_t](#) *ku_mem) [L4_NOTHROW](#)
Add kernel-user memory.

13.1.10.12.1 Detailed Description

C interface of the Task kernel object, see [L4::Task](#) for the C++ interface.

A task represents a combination of the address spaces provided by the [L4Re](#) micro kernel. A task consists of at least a memory address space and an object address space. On IA32 there is also an IO-port address space.

Task objects are created using the [Factory](#) interface.

Include File

```
#include <l4/sys/task.h>
```

13.1.10.12.2 Enumeration Type Documentation

13.1.10.12.2.1 l4_unmap_flags_t

```
enum l4_unmap_flags_t
```

Flags for the unmap operation.

See also

[L4::Task::unmap\(\)](#) and [l4_task_unmap\(\)](#)

Enumerator

| | |
|--------------------|--|
| L4_FP_ALL_SPACES | <p>Flag to tell the unmap operation to revoke permissions from all child mappings including the mapping in the invoked task.</p> <p>Note</p> <p>Object capabilities are not hierarchical – they have no children. The result of the map operation on an object capability is a copy of that capability in the object space of the destination task. An unmap operation on object capabilities is a no-op if this flag is not specified.</p> <p>See also</p> <p>L4::Task::unmap() l4_task_unmap()</p> |
| L4_FP_DELETE_OBJ | <p>Flag that indicates that an unmap operation on object capabilities shall try to delete the corresponding objects immediately. This flag implies the L4_FP_ALL_SPACES flag. The concept of deletion is only applicable to kernel objects. Therefore, for memory and I/O port capabilities, this flag has the same effect as L4_FP_ALL_SPACES alone.</p> <p>See also</p> <p>L4::Task::unmap() l4_task_unmap()</p> <p>Note</p> <p>Specifying L4_FP_DELETE_OBJ ^ L4_FP_ALL_SPACES is treated as L4_FP_OTHER_SPACES.</p> |
| L4_FP_OTHER_SPACES | <p>Counterpart to L4_FP_ALL_SPACES; revoke permissions from child mappings only.</p> <p>See also</p> <p>L4::Task::unmap() l4_task_unmap()</p> |

Definition at line 188 of file [consts.h](#).

13.1.10.12.3 Function Documentation

13.1.10.12.3.1 l4_task_add_ku_mem()

```
l4_msgtag_t l4_task_add_ku_mem (
    l4_cap_idx_t task,
    l4_fpage_t * ku_mem) [inline]
```

Add kernel-user memory.

Parameters

| | | |
|----------------|---------------|---|
| | <i>task</i> | Capability selector of the task to add the memory to. |
| <i>in, out</i> | <i>ku_mem</i> | Flexpage describing the virtual area the memory goes to. On systems without MMU, the flexpage is adjusted to reflect the actually allocated physical address. |

Returns

Syscall return tag

Kernel-user memory (*ku_mem*) is memory that is shared between the kernel and user-space. It is needed for the UTCB area of threads (see [l4_thread_control_bind\(\)](#)) and for (extended) vCPU state. Note that existing kernel-user memory cannot be unmapped or mapped somewhere else.

Note

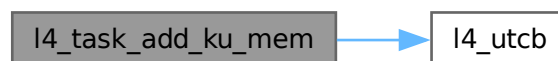
The amount of kernel-user memory that can be allocated at once is limited by the used kernel implementation. The minimum allocatable amount is one page ([L4_PAGE_SIZE](#)). A portable implementation should not depend on allocations greater than 16KiB to succeed.

This function is only guaranteed to work on [L4::Task](#) objects. It might or might not work on [L4::Vm](#) objects or on [L4Re::Dma_space](#) objects but there is no practical use for adding kernel-user memory to [L4::Vm](#) objects or to [L4Re::Dma_space](#) objects.

Definition at line 497 of file [task.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:

**13.1.10.12.3.2 l4_task_cap_equal()**

```

l4_msgtag_t l4_task_cap_equal (
    l4_cap_idx_t task,
    l4_cap_idx_t cap_a,
    l4_cap_idx_t cap_b) [inline]
  
```

Test whether two capabilities point to the same object with the same permissions (only considering selected permissions).

Parameters

| | |
|--------------------|---|
| <i>task</i> | Capability selector for the destination task to do the lookup in. |
| <i>cap↔ _a</i> | Capability selector for the first capability to compare. |
| <i>cap↔ _b</i> | Capability selector for the second capability to compare. |

Return values

| | |
|---------------------------------|--|
| <i>l4_msgtag_t::label()</i> = 1 | The compared capabilities point to the same object with same considered permission. |
| <i>l4_msgtag_t::label()</i> = 0 | The compared capabilities do not point to the same object or differ in the considered permission. |

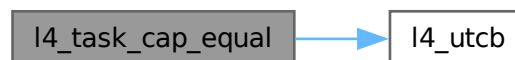
- For [L4::lpc_gate](#) objects, only the permissions [L4_CAP_FPAGE_W](#), [L4_CAP_FPAGE_S](#), and [L4_FPAGE_C_OBJ_RIGHT1](#) are considered for the comparison. Differences in other permissions are ignored.
- For other objects, only the permissions [L4_CAP_FPAGE_W](#) and [L4_CAP_FPAGE_S](#) are considered for the comparison. Differences in other permissions are ignored.

Note that having the [L4_CAP_FPAGE_R](#) permission is implicit in possessing the capability.

Definition at line 490 of file [task.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:

**13.1.10.12.3.3 l4_task_cap_valid()**

```

l4_msgtag_t l4_task_cap_valid (
    l4_cap_idx_t task,
    l4_cap_idx_t cap) [inline]
  
```

Check whether a capability is present (refers to an object).

Parameters

| | |
|-------------|----------------------------------|
| <i>task</i> | Task to check the capability in. |
|-------------|----------------------------------|

| | |
|------------|---|
| <i>cap</i> | Valid capability to check for presence. |
|------------|---|

Return values

| | |
|----------------------------------|--|
| <i>l4_msgtag_t::label()</i> > 0 | Capability is present (refers to an object). |
| <i>l4_msgtag_t::label()</i> == 0 | No capability present (void object). |

A capability is considered present when it refers to an existing kernel object.

Precondition

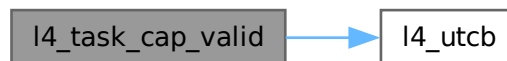
cap must be a valid capability index (i.e. not L4_INVALID_CAP or the like).

Definition at line 484 of file [task.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Referenced by [fiasco_tbuf_validate\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.1.10.12.3.4 l4_task_delete_obj()

```
l4_msgtag_t l4_task_delete_obj (
    l4_cap_idx_t task,
    l4_cap_idx_t obj) [inline]
```

Release capability and delete object.

Parameters

| | |
|-------------|---|
| <i>task</i> | Capability selector of destination task. |
| <i>obj</i> | Capability index of the object to delete. |

Returns

Syscall return tag

If *obj* has the delete permission, initiates the deletion of the object. This implies that all capabilities for that object are gone afterwards. However, kernel-internally, objects are not destroyed until all other kernel objects holding a reference to it drop the reference. Hence, quota used by that object might not be freed immediately.

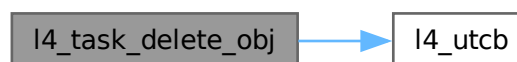
If *obj* does not have the delete permission, no error will be reported and only the capability *obj* is removed. (Note that, depending on the object's reference counter, this might still imply initiation of deletion.)

This operation is equivalent to [l4_task_unmap\(\)](#) with [L4_FP_DELETE_OBJ](#) flag.

Definition at line [463](#) of file [task.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



13.1.10.12.3.5 l4_task_map()

```
l4_msgtag_t l4_task_map (
    l4_cap_idx_t dst_task,
    l4_cap_idx_t src_task,
    l4_fpage_t snd_fpage,
    l4_umword_t snd_base) [inline]
```

Map resources available in the source task to a destination task.

Parameters

| | |
|------------------|--|
| <i>dst_task</i> | Capability selector of the destination task. |
| <i>src_task</i> | Capability selector of the source task. |
| <i>snd_fpage</i> | Send flexpage that describes an area in the address space or object space of the source task. |
| <i>snd_base</i> | Send base that describes an offset in the receive window of the destination task. The lower bits contain additional map control flags (see l4_fpage_cacheability_opt_t for memory mappings, Attributes and additional permissions for object send items for object mappings, and L4_MAP_ITEM_GRANT ; also see l4_map_control() and l4_map_obj_control()). |

Returns

Syscall return tag. The function [l4_error\(\)](#) shall be used to test if the map operation was successful.

Return values

| | |
|----------------------------|--|
| <i>L4_EOK</i> | Operation successful (but see notes below). |
| <i>-L4_EPERM</i> | Insufficient permissions; see precondition. |
| <i>-L4_EINVAL</i> | Invalid source task capability. |
| <i>-L4_IPC_SEMAPFAILED</i> | The map operation failed due to limited quota. |

Precondition

The capability *dst_task* must have the permission [L4_CAP_FPAGE_W](#).

This method allows for asynchronous transfer of capabilities, memory mappings, and IO-port mappings (on IA32) from one task to another. The receive window is the whole address space of *dst_task*. By specifying proper rights in *snd_fpage* and *snd_base*, it is possible to remove rights during transfer.

Note

If the send flexpage is of type [L4_FPAGE_OBJ](#), the [L4_CAP_FPAGE_S](#) right is removed from the transferred capability unless both the source and destination task capabilities possess the [L4_CAP_FPAGE_S](#) right themselves.

Even with [l4_error\(\)](#) returning [L4_EOK](#) there might be cases where not all pages of the send flexpage were mapped respectively granted to the destination task, for instance, if the corresponding mapping in the destination task does already exist.

For more information on spaces and mappings, see [Spaces and Mappings](#). The flexpage API is described in more detail at [Flexpages](#).

Note

For peculiarities when using `grant`, see [L4_MAP_ITEM_GRANT](#).

Definition at line 433 of file [task.h](#).

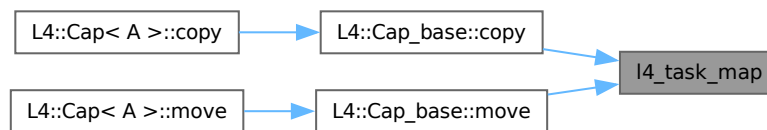
References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Referenced by [L4::Cap_base::copy\(\)](#), and [L4::Cap_base::move\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.1.10.12.3.6 l4_task_release_cap()

```

l4_msgtag_t l4_task_release_cap (
    l4_cap_idx_t task,
    l4_cap_idx_t cap) [inline]
  
```

Release object capability.

Parameters

| | |
|-------------|--|
| <i>task</i> | Capability selector of destination task |
| <i>cap</i> | Capability selector of object to release |

Returns

Syscall return tag

This operation unmaps the capability from the specified task. This operation is equivalent to unmapping a single object capability by specifying all object rights as unmap mask.

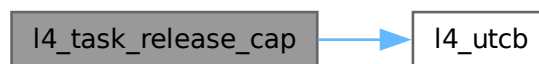
Note

If the reference counter of the kernel object referenced by `cap` goes down to zero, deletion of the object is initiated. Objects are not destroyed until all other kernel objects holding a reference to it drop the reference.

Definition at line 478 of file [task.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:

**13.1.10.12.3.7 l4_task_unmap()**

```

l4_msgtag_t l4_task_unmap (
    l4_cap_idx_t task,
    l4_fpage_t fpage,
    l4_umword_t map_mask) [inline]
  
```

Revoke rights from the task.

Parameters

| | |
|-----------------|---|
| <i>task</i> | Capability selector of destination task |
| <i>fpage</i> | Flexpage that describes an area in one capability space of the destination task and the rights to revoke. |
| <i>map_mask</i> | Unmap mask, see l4_unmap_flags_t |

Returns

Syscall return tag

This method allows to revoke rights from the destination task. The rights to revoke are specified in the flexpage, see [l4_fpage_rights\(\)](#). For a flexpage describing IO ports or memory, it also revokes rights from all the tasks that got the rights delegated from the destination task (i.e., this operation does a recursive rights revocation). The capability is unmapped if certain rights are specified, see below for details. It is guaranteed that the rights revocation is completed before this function returns.

Note that this function cannot be used to revoke the reference counting permission (see [L4_FPAGE_C_REF_CNT](#)) or the IPC-gate server permission (see [L4_FPAGE_C_IPCGATE_SVR](#)) from object capabilities.

It depends on the platform and the object type which rights need to be specified in the `rights` field of `fpage` to unmap a capability:

- An object capability is unmapped if and only if the [L4_CAP_FPAGE_R](#) right bit is set.
- An IO port is unmapped if and only if any right bit is set.
- Memory is unmapped if and only if the [L4_FPAGE_RO](#) right bit is set.

Note

Depending on the page-table features supported by the hardware, revocation of certain rights from a memory capability can be a no-op (i.e., the rights are not revoked). Further, revocation of certain rights may grant other rights which were not present before. For instance, on an architecture without support for NX, revoking X does nothing. For another example, revoking only X from an execute-only page grants read permission (because the mapping remains present in the page table).

If the reference counter of a kernel object referenced in `fpage` goes down to zero (as a result of deleting capabilities), the deletion of the object is initiated. Objects are not destroyed until all other kernel objects holding a reference to it drop the reference.

Examples

[examples/sys/utcb-ipc/main.c](#).

Definition at line [440](#) of file [task.h](#).

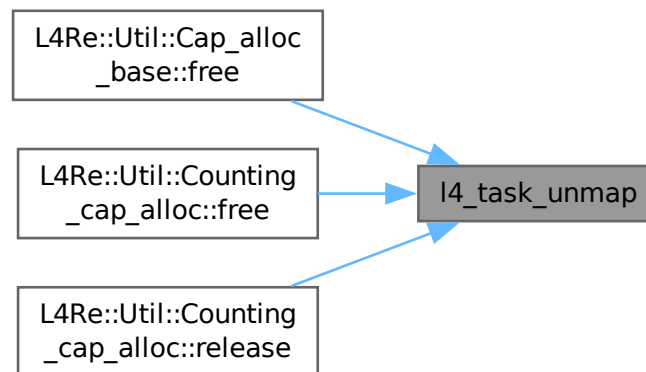
References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Referenced by [L4Re::Util::Cap_alloc_base::free\(\)](#), [L4Re::Util::Counting_cap_alloc< COUNTERTYPE, Dbg >::free\(\)](#), and [L4Re::Util::Counting_cap_alloc< COUNTERTYPE, Dbg >::release\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.1.10.12.3.8 l4_task_unmap_batch()

```

l4_msgtag_t l4_task_unmap_batch (
    l4_cap_idx_t task,
    l4_fpage_t const * fpages,
    unsigned num_fpages,
    l4_umword_t map_mask) [inline]
  
```

Revoke rights from a task.

Parameters

| | |
|-------------------|---|
| <i>task</i> | Capability selector of destination task |
| <i>fpages</i> | An array of flexpages. Each item describes an area in one capability space of the destination task. |
| <i>num_fpages</i> | The size of the fpages array in elements (number of fpages sent). |
| <i>map_mask</i> | Unmap mask, see l4_unmap_flags_t |

Returns

Syscall return tag

Revoke rights specified in an array of flexpages, see [l4_task_unmap](#) for details.

Precondition

The caller needs to take care that `num_fpages` is not bigger than `L4_UTCB_GENERIC_DATA_SIZE - 2`.

Definition at line 447 of file [task.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:

**13.1.10.12.3.9 l4_task_vgicc_map()**

```

l4_msgtag_t l4_task_vgicc_map (
    l4_cap_idx_t task,
    l4_fpage_t vgicc_fpage) [inline]
  
```

Map the GIC virtual CPU interface page to the task in case Fiasco detected a GIC version 2.

Parameters

| | |
|--------------------|---|
| <i>task</i> | Capability selector of destination task |
| <i>vgicc_fpage</i> | Flexpage that describes an area in the address space of the destination task to map the vGICC page to |

Returns

Syscall return tag

Definition at line 46 of file [__task-arm.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

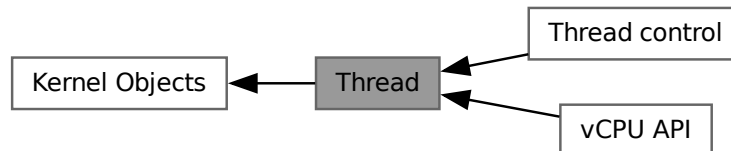
Here is the call graph for this function:



13.1.10.13 Thread

C Thread object interface, see [L4::Thread](#) for the C++ interface.

Collaboration diagram for Thread:



Topics

- [Thread control](#) 405
API for Thread Control method.
- [vCPU API](#) 411
vCPU API.

Enumerations

- enum [L4_thread_control_flags](#) { [L4_THREAD_CONTROL_SET_PAGER](#) = 0x0010000 , [L4_THREAD_CONTROL_BIND_TASK](#) = 0x0200000 , [L4_THREAD_CONTROL_ALIEN](#) = 0x0400000 , [L4_THREAD_CONTROL_SET_EXC_HANDLER](#) = 0x1000000 }
Flags for the thread control operation.
- enum [L4_thread_control_mr_indices](#) { [L4_THREAD_CONTROL_MR_IDX_FLAGS](#) = 0 , [L4_THREAD_CONTROL_MR_IDX_PAGER](#) = 1 , [L4_THREAD_CONTROL_MR_IDX_EXC_HANDLER](#) = 2 , [L4_THREAD_CONTROL_MR_IDX_FLAG_VALS](#) = 4 , [L4_THREAD_CONTROL_MR_IDX_BIND_UTCB](#) = 5 , [L4_THREAD_CONTROL_MR_IDX_BIND_TASK](#) = 6 }
Indices for the values in the message register for thread control.
- enum [L4_thread_ex_regs_flags](#) { [L4_THREAD_EX_REGS_CANCEL](#) = 0x10000UL , [L4_THREAD_EX_REGS_TRIGGER_EXC](#) = 0x20000UL , [L4_THREAD_EX_REGS_ARCH_MASK](#) = 0xff000000UL }
- enum [L4_thread_ex_regs_flags_arm](#) { [L4_THREAD_EX_REGS_ARM_SET_EL_MASK](#) = 0x3 << 24 , [L4_THREAD_EX_REGS_ARM_SET_EL_KEEP](#) = 0x0 << 24 , [L4_THREAD_EX_REGS_ARM_SET_EL_EL0](#) = 0x1 << 24 , [L4_THREAD_EX_REGS_ARM_SET_EL_EL1](#) = 0x2 << 24 }
- enum [L4_thread_ex_regs_flags_arm64](#) { [L4_THREAD_EX_REGS_ARM64_SET_EL_MASK](#) = 0x3 << 24 , [L4_THREAD_EX_REGS_ARM64_SET_EL_KEEP](#) = 0x0 << 24 , [L4_THREAD_EX_REGS_ARM64_SET_EL_EL0](#) = 0x1 << 24 , [L4_THREAD_EX_REGS_ARM64_SET_EL_EL1](#) = 0x2 << 24 }

Functions

- `l4_msgtag_t l4_thread_ex_regs (l4_cap_idx_t thread, l4_addr_t ip, l4_addr_t sp, l4_umword_t flags) L4_NOTHROW`
Exchange basic thread registers.
- `l4_msgtag_t l4_thread_ex_regs_u (l4_cap_idx_t thread, l4_addr_t ip, l4_addr_t sp, l4_umword_t flags, l4_utcb_t *utcb) L4_NOTHROW`
Exchange basic thread registers.
- `l4_msgtag_t l4_thread_ex_regs_ret (l4_cap_idx_t thread, l4_addr_t *ip, l4_addr_t *sp, l4_umword_t *flags) L4_NOTHROW`
Exchange basic thread registers and return previous values.
- `l4_msgtag_t l4_thread_ex_regs_ret_u (l4_cap_idx_t thread, l4_addr_t *ip, l4_addr_t *sp, l4_umword_t *flags, l4_utcb_t *utcb) L4_NOTHROW`
Exchange basic thread registers and return previous values.
- `l4_msgtag_t l4_thread_yield (void) L4_NOTHROW`
Yield current time slice.
- `l4_msgtag_t l4_thread_switch (l4_cap_idx_t to_thread) L4_NOTHROW`
Switch to another thread (and donate the remaining time slice).
- `l4_msgtag_t l4_thread_stats_time (l4_cap_idx_t thread, l4_kernel_clock_t *us) L4_NOTHROW`
Get consumed time of thread in μ s.
- `l4_msgtag_t l4_thread_vcpu_resume_start (void) L4_NOTHROW`
vCPU return from event handler.
- `l4_msgtag_t l4_thread_vcpu_resume_commit (l4_cap_idx_t thread, l4_msgtag_t tag) L4_NOTHROW`
Commit vCPU resume.
- `l4_msgtag_t l4_thread_vcpu_control (l4_cap_idx_t thread, l4_addr_t vcpu_state) L4_NOTHROW`
Enable the vCPU feature for the thread.
- `l4_msgtag_t l4_thread_vcpu_control_u (l4_cap_idx_t thread, l4_addr_t vcpu_state, l4_utcb_t *utcb) L4_NOTHROW`
Enable the vCPU feature for the thread.
- `l4_msgtag_t l4_thread_vcpu_control_ext (l4_cap_idx_t thread, l4_addr_t ext_vcpu_state) L4_NOTHROW`
Enable the extended vCPU feature for the thread.
- `l4_msgtag_t l4_thread_vcpu_control_ext_u (l4_cap_idx_t thread, l4_addr_t ext_vcpu_state, l4_utcb_t *utcb) L4_NOTHROW`
Enable the extended vCPU feature for the thread.
- `l4_msgtag_t l4_thread_register_del_irq (l4_cap_idx_t thread, l4_cap_idx_t irq) L4_NOTHROW`
Register an IRQ that will trigger upon deletion events.
- `l4_msgtag_t l4_thread_modify_sender_start (void) L4_NOTHROW`
Start a thread sender modification sequence.
- `int l4_thread_modify_sender_add (l4_umword_t match_mask, l4_umword_t match, l4_umword_t del_bits, l4_umword_t add_bits, l4_msgtag_t *tag) L4_NOTHROW`
Add a modification pattern to a sender modification sequence.
- `l4_msgtag_t l4_thread_modify_sender_commit (l4_cap_idx_t thread, l4_msgtag_t tag) L4_NOTHROW`
Apply (commit) a sender modification sequence.
- `l4_msgtag_t l4_thread_register_doorbell_irq (l4_cap_idx_t thread, l4_cap_idx_t irq) L4_NOTHROW`
Register an IRQ that will trigger when a forwarded virtual interrupt is pending.
- `l4_msgtag_t l4_thread_arm_set_tpidruro (l4_cap_idx_t thread, l4_addr_t tpidruro) L4_NOTHROW`
Set the TPIDRURO thread specific register.

13.1.10.13.1 Detailed Description

C Thread object interface, see [L4::Thread](#) for the C++ interface.

An [L4](#) thread is a thread of execution in the [L4](#) context. Usually user-level and kernel threads are mapped 1:1 to each other. Thread kernel objects are created using a factory, see [Factory](#) ([l4_factory_create_thread\(\)](#)).

Amongst other things an [L4](#) thread encapsulates:

- CPU state
 - General-purpose registers
 - Program counter
 - Stack pointer
- FPU state
- Scheduling parameters, see the [Scheduler](#) API
- Execution state
 - Blocked, Runnable, Running

Thread objects provide an API for

- Thread configuration and manipulation
- Thread switching.

The thread control functions are used to control various aspects of a thread. See [l4_thread_control_start\(\)](#) for more information.

On ARM newly created threads run in EL0 by default and the exception level can be changed there with [ex_regs\(\)](#).

Include File

```
#include <l4/sys/thread.h>
```

For the C++ interface refer to [L4::Thread](#).

13.1.10.13.2 Enumeration Type Documentation

13.1.10.13.2.1 L4_thread_control_flags

```
enum L4_thread_control_flags
```

Flags for the thread control operation.

Enumerator

| | |
|-----------------------------|--------------------------|
| L4_THREAD_CONTROL_SET_PAGER | The pager will be given. |
|-----------------------------|--------------------------|

| | |
|-----------------------------------|--|
| L4_THREAD_CONTROL_BIND_TASK | The task to bind the thread to will be given. |
| L4_THREAD_CONTROL_ALIEN | Alien state of the thread is set. |
| L4_THREAD_CONTROL_SET_EXC_HANDLER | The exception handler of the thread will be given. |

Definition at line 761 of file [thread.h](#).

13.1.10.13.2.2 L4_thread_control_mr_indices

```
enum L4_thread_control_mr_indices
```

Indices for the values in the message register for thread control.

Enumerator

| | |
|--------------------------------------|---|
| L4_THREAD_CONTROL_MR_IDX_FLAGS | See also L4_thread_control_flags . |
| L4_THREAD_CONTROL_MR_IDX_PAGER | Index for pager cap. |
| L4_THREAD_CONTROL_MR_IDX_EXC_HANDLER | Index for exception handler. |
| L4_THREAD_CONTROL_MR_IDX_FLAG_VALS | Index for feature values. |
| L4_THREAD_CONTROL_MR_IDX_BIND_UTCB | Index for UTCB address for bind. |
| L4_THREAD_CONTROL_MR_IDX_BIND_TASK | Index for task flexpage for bind. |

Definition at line 782 of file [thread.h](#).

13.1.10.13.2.3 L4_thread_ex_regs_flags

```
enum L4_thread_ex_regs_flags
```

Flags for the thread ex-regs operation.

Enumerator

| | |
|-------------------------------------|---|
| L4_THREAD_EX_REGS_CANCEL | Cancel ongoing IPC in the thread. |
| L4_THREAD_EX_REGS_TRIGGER_EXCEPTION | Trigger artificial exception in thread. |
| L4_THREAD_EX_REGS_ARCH_MASK | Arch specific flags. |

Definition at line 797 of file [thread.h](#).

13.1.10.13.2.4 L4_thread_ex_regs_flags_arm

enum `L4_thread_ex_regs_flags_arm`

Arm specific `L4::Thread::ex_regs()` flags.

Only one option must be used in calls to `L4::Thread::ex_regs()`. Using more than one option results in undefined behaviour.

Enumerator

| | |
|--|---|
| <code>L4_THREAD_EX_REGS_ARM_SET_EL_MASK</code> | Exception level set mask. |
| <code>L4_THREAD_EX_REGS_ARM_SET_EL_KEEP</code> | Keep current exception level of thread (default). |
| <code>L4_THREAD_EX_REGS_ARM_SET_EL_EL0</code> | Set exception level of thread to EL0 (usr mode). |
| <code>L4_THREAD_EX_REGS_ARM_SET_EL_EL1</code> | Set exception level of thread to EL1 (sys mode). |

Definition at line 37 of file `thread.h`.

13.1.10.13.2.5 L4_thread_ex_regs_flags_arm64

enum `L4_thread_ex_regs_flags_arm64`

Arm64 specific `L4::Thread::ex_regs()` flags.

Only one option must be used in calls to `L4::Thread::ex_regs()`. Using more than one option results in undefined behaviour.

Enumerator

| | |
|--|---|
| <code>L4_THREAD_EX_REGS_ARM64_SET_EL_MASK</code> | Exception level set mask. |
| <code>L4_THREAD_EX_REGS_ARM64_SET_EL_KEEP</code> | Keep current exception level of thread (default). |
| <code>L4_THREAD_EX_REGS_ARM64_SET_EL_EL0</code> | Set exception level of thread to EL0. |
| <code>L4_THREAD_EX_REGS_ARM64_SET_EL_EL1</code> | Set exception level of thread to EL1t. |

Definition at line 44 of file `thread.h`.

13.1.10.13.3 Function Documentation

13.1.10.13.3.1 l4_thread_arm_set_tpidruro()

```
l4_msgtag_t l4_thread_arm_set_tpidruro (
    l4_cap_idx_t thread,
    l4_addr_t tpidruro) [inline]
```

Set the TPIDRURO thread specific register.

Parameters

| | |
|-----------------|----------------------|
| <i>thread</i> | Thread to manipulate |
| <i>tpidruro</i> | The value to be set |

Returns

System call return tag

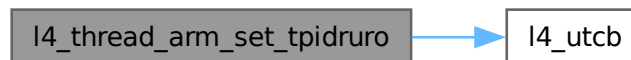
Note

When this function is invoked for a thread currently executing on a different core, then the changed register content will not be visible to that thread until a thread switch happens on that core.

Definition at line 70 of file [thread.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:

**13.1.10.13.3.2 l4_thread_ex_regs()**

```

l4_msgtag_t l4_thread_ex_regs (
    l4_cap_idx_t thread,
    l4_addr_t ip,
    l4_addr_t sp,
    l4_umword_t flags) [inline]
  
```

Exchange basic thread registers.

Parameters

| | |
|---------------|--|
| <i>thread</i> | Capability selector of the thread to manipulate. |
| <i>ip</i> | New instruction pointer, use <code>~0UL</code> to leave the instruction pointer unchanged. |
| <i>sp</i> | New stack pointer, use <code>~0UL</code> to leave the stack pointer unchanged. |
| <i>flags</i> | Ex-regs flags, see L4_thread_ex_regs_flags . |

Returns

System call return tag

This method allows to manipulate and start a thread. The basic functionality is to set the instruction pointer and the stack pointer of a thread. Additionally, this method allows also to cancel ongoing IPC operations and to force the thread to raise an artificial exception (see `flags`). If the thread is in an IPC operation or if `L4_THREAD_EX_REGS_TRIGGER_EXCEPTION` forces an IPC then changes in IP and SP take effect directly after returning from this IPC. On ARM this method allows to change the exception level, see `L4_thread_ex_regs_flags_arm` and `L4_thread_ex_regs_flags_arm64`.

The thread is started using `l4_scheduler_run_thread()`. However, if at the time `l4_scheduler_run_thread()` is called, the instruction pointer of the thread is invalid, a later call to `l4_thread_ex_regs()` with a valid instruction pointer might start the thread.

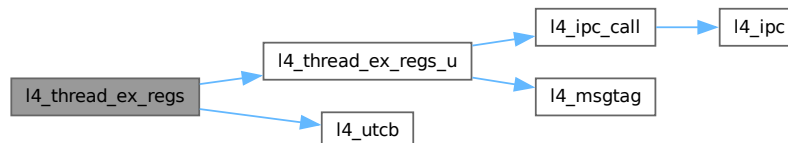
Examples

`examples/sys/aliens/main.c`, `examples/sys/singlestep/main.c`, `examples/sys/start-with-exc/main.c`, and `examples/sys/utcb-ipc/main.c`.

Definition at line 942 of file `thread.h`.

References `L4_NOTHROW`, `l4_thread_ex_regs_u()`, and `l4_utcb()`.

Here is the call graph for this function:



13.1.10.13.3.3 l4_thread_ex_regs_ret()

```

l4_msgtag_t l4_thread_ex_regs_ret (
    l4_cap_idx_t thread,
    l4_addr_t * ip,
    l4_addr_t * sp,
    l4_umword_t * flags) [inline]
  
```

Exchange basic thread registers and return previous values.

Parameters

| | | |
|---------|---------------|---|
| | <i>thread</i> | Capability selector of the thread to manipulate. |
| in, out | <i>ip</i> | New instruction pointer, use <code>~0UL</code> to leave the instruction pointer unchanged, return previous instruction pointer. |

| | | |
|----------------|--------------|--|
| <i>in, out</i> | <i>sp</i> | New stack pointer, use <code>~0UL</code> to leave the stack pointer unchanged, returns previous stack pointer. |
| <i>in, out</i> | <i>flags</i> | Ex-regs flags, see L4_thread_ex_regs_flags , return previous CPU flags of the thread. |

Returns

System call return tag

This method allows to manipulate and start a thread. The basic functionality is to set the instruction pointer and the stack pointer of a thread. Additionally, this method allows also to cancel ongoing IPC operations and to force the thread to raise an artificial exception (see `flags`). If the thread is in an IPC operation or if [L4_THREAD_EX_REGS_TRIGGER_EXCEPTION](#) forces an IPC then changes in IP and SP take effect directly after returning from this IPC. On ARM this method allows to change the exception level, see [L4_thread_ex_regs_flags_arm](#) and [L4_thread_ex_regs_flags_arm64](#).

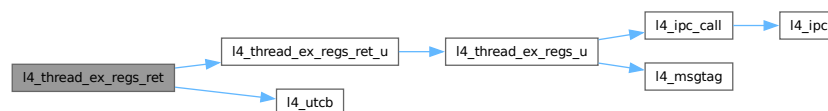
The thread is started using [l4_scheduler_run_thread\(\)](#). However, if at the time [l4_scheduler_run_thread\(\)](#) is called, the instruction pointer of the thread is invalid, a later call to [l4_thread_ex_regs\(\)](#) with a valid instruction pointer might start the thread.

Returned values are valid only if function returns successfully.

Definition at line 949 of file [thread.h](#).

References [L4_NOTHROW](#), [l4_thread_ex_regs_ret_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



13.1.10.13.3.4 l4_thread_ex_regs_ret_u()

```

l4_msgtag_t l4_thread_ex_regs_ret_u (
    l4_cap_idx_t thread,
    l4_addr_t * ip,
    l4_addr_t * sp,
    l4_umword_t * flags,
    l4_utcb_t * utcb) [inline]
  
```

Exchange basic thread registers and return previous values.

Parameters

| | | |
|--|---------------|--|
| | <i>thread</i> | Capability selector of the thread to manipulate. |
|--|---------------|--|

| | | |
|----------------|--------------|---|
| <i>in, out</i> | <i>ip</i> | New instruction pointer, use <code>~0UL</code> to leave the instruction pointer unchanged, return previous instruction pointer. |
| <i>in, out</i> | <i>sp</i> | New stack pointer, use <code>~0UL</code> to leave the stack pointer unchanged, returns previous stack pointer. |
| <i>in, out</i> | <i>flags</i> | Ex-regs flags, see L4_thread_ex_regs_flags , return previous CPU flags of the thread. |
| | <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb . |

Returns

System call return tag. [out] parameters are only valid if the function returns successfully. Use [l4_error\(\)](#) to check.

This method allows to manipulate and start a thread. The basic functionality is to set the instruction pointer and the stack pointer of a thread. Additionally, this method allows also to cancel ongoing IPC operations and to force the thread to raise an artificial exception (see `flags`). If the thread is in an IPC operation or if [L4_THREAD_EX_REGS_TRIGGER_EXCEPTION](#) forces an IPC then changes in IP and SP take effect directly after returning from this IPC. On ARM this method allows to change the exception level, see [L4_thread_ex_regs_flags_arm](#) and [L4_thread_ex_regs_flags_arm64](#).

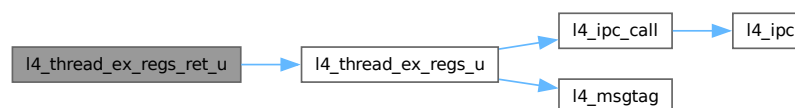
The thread is started using [L4::Scheduler::run_thread\(\)](#). However, if at the time [L4::Scheduler::run_thread\(\)](#) is called, the instruction pointer of the thread is invalid, a later call to `ex_regs()` with a valid instruction pointer might start the thread.

Definition at line 823 of file [thread.h](#).

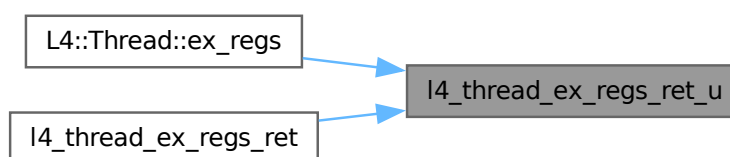
References [L4_NOTHROW](#), [l4_thread_ex_regs_u\(\)](#), and [l4_msg_regs_t::mr](#).

Referenced by [L4::Thread::ex_regs\(\)](#), and [l4_thread_ex_regs_ret\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.1.10.13.3.5 l4_thread_ex_regs_u()

```
l4_msgtag_t l4_thread_ex_regs_u (
    l4_cap_idx_t thread,
    l4_addr_t ip,
    l4_addr_t sp,
    l4_umword_t flags,
    l4_utcb_t * utcb) [inline]
```

Exchange basic thread registers.

Parameters

| | |
|---------------|--|
| <i>thread</i> | Capability selector of the thread to manipulate. |
| <i>ip</i> | New instruction pointer, use ~0UL to leave the instruction pointer unchanged. |
| <i>sp</i> | New stack pointer, use ~0UL to leave the stack pointer unchanged. |
| <i>flags</i> | Ex-regs flags, see L4_thread_ex_regs_flags . |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb . |

Returns

System call return tag.

This method allows to manipulate and start a thread. The basic functionality is to set the instruction pointer and the stack pointer of a thread. Additionally, this method allows also to cancel ongoing IPC operations and to force the thread to raise an artificial exception (see *flags*). If the thread is in an IPC operation or if [L4_THREAD_EX_REGS_TRIGGER_EXCEPTION](#) forces an IPC then changes in IP and SP take effect directly after returning from this IPC. On ARM this method allows to change the exception level, see [L4_thread_ex_regs_flags_arm](#) and [L4_thread_ex_regs_flags_arm64](#).

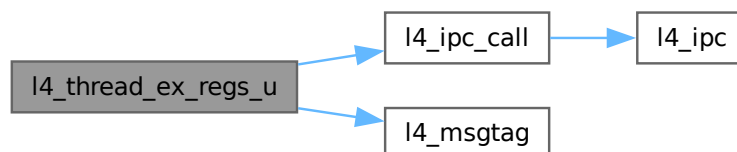
The thread is started using [L4::Scheduler::run_thread\(\)](#). However, if at the time [L4::Scheduler::run_thread\(\)](#) is called, the instruction pointer of the thread is invalid, a later call to *ex_regs()* with a valid instruction pointer might start the thread.

Definition at line 812 of file [thread.h](#).

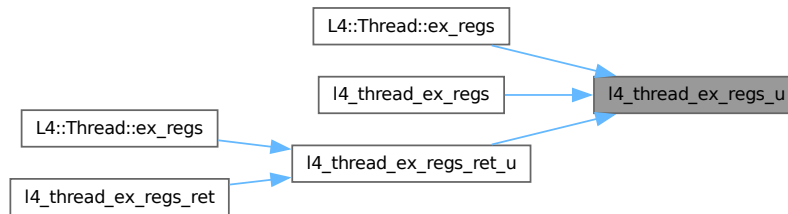
References [l4_ipc_call\(\)](#), [L4_IPC_NEVER](#), [l4_msgtag\(\)](#), [L4_NOTHROW](#), [L4_PROTO_THREAD](#), [L4_THREAD_EX_REGS_OP](#), and [l4_msg_regs_t::mr](#).

Referenced by [L4::Thread::ex_regs\(\)](#), [l4_thread_ex_regs\(\)](#), and [l4_thread_ex_regs_ret_u\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.1.10.13.3.6 l4_thread_modify_sender_add()

```

int l4_thread_modify_sender_add (
    l4_umword_t match_mask,
    l4_umword_t match,
    l4_umword_t del_bits,
    l4_umword_t add_bits,
    l4_msgtag_t * tag) [inline]

```

Add a modification pattern to a sender modification sequence.

Parameters

| | |
|-------------------|---|
| <i>tag</i> | Tag received from l4_thread_modify_sender_start() or previous l4_thread_modify_sender_add() calls from the same sequence. |
| <i>match_mask</i> | Bitmask of bits to match the label. |
| <i>match</i> | Bitmask that must be equal to the label after applying <i>match_mask</i> . |
| <i>del_bits</i> | Bits to be deleted from the label. |
| <i>add_bits</i> | Bits to be added to the label. |

Returns

0 on success, <0 on error

In pseudo code: if ((sender_label & match_mask) == match) { sender_label = (sender_label & ~del_bits) | add_bits; }

Only the first match is applied.

See also

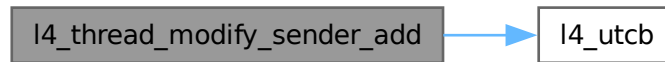
[l4_thread_modify_sender_start](#)

[l4_thread_modify_sender_commit](#)

Definition at line 1116 of file [thread.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



13.1.10.13.3.7 l4_thread_modify_sender_commit()

```

l4_msgtag_t l4_thread_modify_sender_commit (
    l4_cap_idx_t thread,
    l4_msgtag_t tag) [inline]
  
```

Apply (commit) a sender modification sequence.

The modification rules are applied to all IPCs to the thread (whether directly or by IPC gate) that are already in flight, that is that the sender is already blocking on.

Note

Modifying the senders of a thread running on a different CPU core is not supported.

To ensure that no in-flight senders are missed, either the thread itself must execute `modify_senders`, or the thread executing the `modify_senders` must synchronize with the target thread. This synchronization must ensure the following:

1. Before `modify_senders` is executed the target thread must execute at least shortly (so that pending DRQs are handled).
2. The target thread must pause its IPC dispatch, until `modify_senders` is completed. In other words, the target thread must not be receive ready, because otherwise an IPC message with an unmodified label can be transferred to its UTCB or vCPU state.

See also

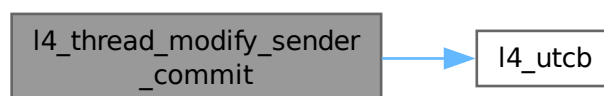
[l4_thread_modify_sender_start](#)

[l4_thread_modify_sender_add](#)

Definition at line 1127 of file [thread.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



13.1.10.13.3.8 l4_thread_modify_sender_start()

```
l4_msgtag_t l4_thread_modify_sender_start (
    void )    [inline]
```

Start a thread sender modification sequence.

Add modification rules with [l4_thread_modify_sender_add\(\)](#) and commit with [l4_thread_modify_sender_commit\(\)](#). Do not touch the UTCB between [l4_thread_modify_sender_start\(\)](#) and [l4_thread_modify_sender_commit\(\)](#).

This mechanism shall be used to change the source object labels of every pending IPC of an IPC gate or an IRQ if the labels in such pending IPC become invalid for the receiving thread, potentially because:

- an IPC gate / IRQ was unbound from a thread, or
- an IPC gate / IRQ was removed, or
- the label of an IPC gate /IRQ bound to a thread was changed.

It is not required to perform the modify_sender mechanism after an IPC gate or an IRQ was bound to a thread for the first time.

See also

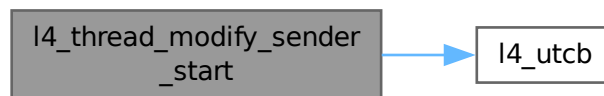
[l4_thread_modify_sender_add](#)

[l4_thread_modify_sender_commit](#)

Definition at line 1110 of file [thread.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



13.1.10.13.3.9 l4_thread_register_del_irq()

```
l4_msgtag_t l4_thread_register_del_irq (  
    l4_cap_idx_t thread,  
    l4_cap_idx_t irq) [inline]
```

Register an IRQ that will trigger upon deletion events.

Parameters

| | |
|---------------|---|
| <i>thread</i> | Thread to register IRQ for. |
| <i>irq</i> | Capability selector for the IRQ object to be triggered. |

Returns

System call return tag containing the return code.

Return values

| | |
|------------------|---|
| <i>-L4_EPERM</i> | Insufficient permissions; see precondition. |
|------------------|---|

Precondition

The capability *irq* must have the permission [L4_CAP_FPAGE_W](#).

In case the *irq* is already bound to an interrupt source, it is unbound first. When *irq* is deleted, it will be deregistered first. A registered deletion Irq can only be deregistered by deleting the Irq or the thread.

List of deletion events:

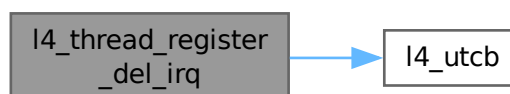
- deletion of one or several IPC gates bound to this thread.

When the deletion event is delivered, there is no indication about which IPC gate was deleted.

Definition at line [1037](#) of file [thread.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



13.1.10.13.3.10 `l4_thread_register_doorbell_irq()`

```
l4_msgtag_t l4_thread_register_doorbell_irq (
    l4_cap_idx_t thread,
    l4_cap_idx_t irq) [inline]
```

Register an IRQ that will trigger when a forwarded virtual interrupt is pending.

Parameters

| | |
|---------------|---|
| <i>thread</i> | Thread to register IRQ for. |
| <i>irq</i> | Capability selector for the IRQ object to be triggered. |

Returns

System call return tag containing the return code.

Return values

| | |
|------------------------|---|
| <code>-L4_EPERM</code> | Insufficient permissions; see precondition. |
|------------------------|---|

Precondition

The capability `irq` must have the permission [L4_CAP_FPAGE_W](#).

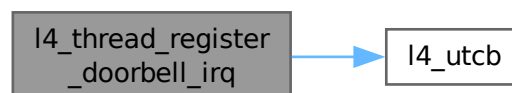
See [l4_irq_bind_vcpu\(\)](#) for more details about how interrupts can be forwarded directly by the kernel to extended vCPU user mode.

In case the `irq` is already bound to an interrupt source, it is unbound first. When `irq` is deleted, it will be deregistered first. A registered deletion `l4` can only be deregistered by deleting the `l4` or the thread.

Definition at line 1146 of file [thread.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



13.1.10.13.3.11 l4_thread_stats_time()

```
l4_msgtag_t l4_thread_stats_time (  
    l4_cap_idx_t thread,  
    l4_kernel_clock_t * us) [inline]
```

Get consumed time of thread in μ s.

Parameters

| | | |
|-----|---------------|---------------------------------------|
| | <i>thread</i> | Thread to get the consumed time from. |
| out | <i>us</i> | Consumed time in μ s. |

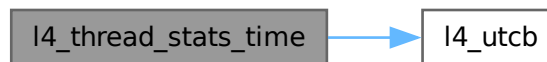
Returns

system call return tag

Definition at line 1005 of file [thread.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:

**13.1.10.13.3.12 l4_thread_switch()**

```
l4_msgtag_t l4_thread_switch (  
    l4_cap_idx_t to_thread) [inline]
```

Switch to another thread (and donate the remaining time slice).

Parameters

| | |
|------------------|--------------------------|
| <i>to_thread</i> | The thread to switch to. |
|------------------|--------------------------|

Returns

system call return tag

Definition at line 996 of file [thread.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:

**13.1.10.13.3.13 l4_thread_vcpu_control()**

```

l4_msgtag_t l4_thread_vcpu_control (
    l4_cap_idx_t thread,
    l4_addr_t vcpu_state) [inline]
  
```

Enable the vCPU feature for the thread.

Parameters

| | |
|-------------------|--|
| <i>thread</i> | Capability selector of the thread for which the vCPU feature shall be enabled. |
| <i>vcpu_state</i> | The virtual address where the kernel shall store the vCPU state in case of vCPU exits. The address must be a valid kernel-user-memory address (see l4_task_add_ku_mem()). |

Returns

Syscall return tag.

This function enables the vCPU feature of the `thread`.

The kernel-user memory area starting at `vcpu_state` must be at least 128-byte aligned and must cover the size of [l4_vcpu_state_t](#).

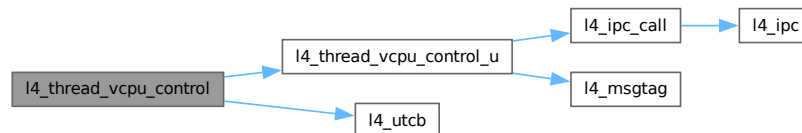
Note

Disabling of the vCPU feature is optional and currently not supported.

Definition at line 1054 of file [thread.h](#).

References [L4_NOTHROW](#), [l4_thread_vcpu_control_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:

**13.1.10.13.3.14 l4_thread_vcpu_control_ext()**

```

l4_msgtag_t l4_thread_vcpu_control_ext (
    l4_cap_idx_t thread,
    l4_addr_t ext_vcpu_state) [inline]
  
```

Enable the extended vCPU feature for the thread.

Parameters

| | |
|-----------------------|--|
| <i>thread</i> | Capability selector of the thread for which the extended vCPU feature shall be enabled. |
| <i>ext_vcpu_state</i> | The virtual address where the kernel shall store the vCPU state in case of vCPU exits. The address must be a valid kernel-user-memory address (see l4_task_add_ku_mem()). |

Returns

Syscall return tag.

The extended vCPU feature allows the use of hardware-virtualization features such as Intel's VT or AMD's SVM.

This function enables the extended vCPU feature of the `thread`. Enabling the extended vCPU feature also enables the vCPU feature.

The kernel-user memory area starting at `ext_vcpu_state` must be at least 4 KiB aligned and must cover a size of `L4_PAGESIZE`. It includes the data of [l4_vcpu_state_t](#) at offset 0, the extended vCPU state at offset `L4_VCPU_OFFSET_EXT_STATE`, and, on some platforms, the extended vCPU information at offset `L4_VCPU_OFFSET_EXT_INFOS`.

Note

Enabling the extended vCPU feature for a thread running on a different CPU core is currently not supported.

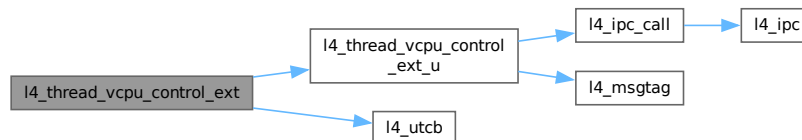
Disabling of the extended vCPU feature is currently not supported.

Upgrading from non-extended vCPU feature to extended vCPU feature is currently not supported.

Definition at line 1069 of file [thread.h](#).

References [L4_NOTHROW](#), [l4_thread_vcpu_control_ext_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



13.1.10.13.3.15 l4_thread_vcpu_control_ext_u()

```

l4_msgtag_t l4_thread_vcpu_control_ext_u (
    l4_cap_idx_t thread,
    l4_addr_t ext_vcpu_state,
    l4_utcb_t * utcb) [inline]
  
```

Enable the extended vCPU feature for the thread.

Parameters

| | |
|-----------------------|--|
| <i>thread</i> | Capability selector of the thread for which the extended vCPU feature shall be enabled. |
| <i>ext_vcpu_state</i> | The virtual address where the kernel shall store the vCPU state in case of vCPU exits. The address must be a valid kernel-user-memory address (see L4::Task::add_ku_mem()). |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb . |

Returns

Syscall return tag.

The extended vCPU feature allows the use of hardware-virtualization features such as Intel's VT-x (VMX) or AMD's AMD-V (SVM).

This function enables the extended vCPU feature of `this` thread. Enabling the extended vCPU feature also enables the vCPU feature.

The kernel-user memory area starting at `ext_vcpu_state` must be at least 4 KiB aligned and must cover a size of `L4_PAGESIZE`. It includes the data of [l4_vcpu_state_t](#) at offset 0, the extended vCPU state at offset `L4_VCPU_OFFSET_EXT_STATE`, and, on some platforms, the extended vCPU information at offset `L4_VCPU_OFFSET_EXT_INFOS`.

On Intel's VT-x (VMX), the extended vCPU state is [l4_vm_vmx_vcpu_vmcs_t](#) and the extended vCPU information is [l4_vm_vmx_vcpu_infos_t](#). Furthermore, the extended vCPU state needs to be associated with a vCPU context (see [l4_vm_vmx_set_hw_vmcs\(\)](#)).

On AMD's AMD-V (SVM), the extended vCPU state is [l4_vm_svm_vmcb_t](#).

Note

Enabling the extended vCPU feature for a thread running on a different CPU core is currently not supported.

Disabling of the extended vCPU feature is currently not supported.

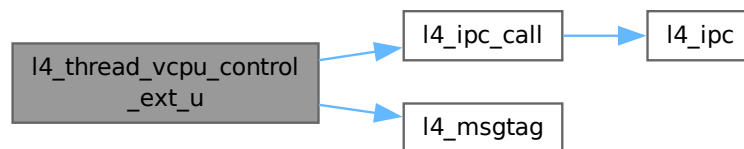
Upgrading from non-extended vCPU feature to extended vCPU feature is currently not supported.

Definition at line 1059 of file [thread.h](#).

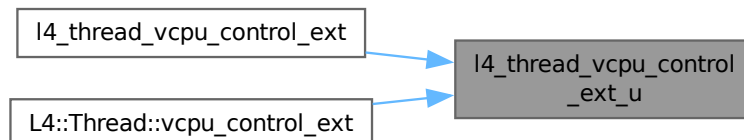
References [l4_ipc_call\(\)](#), [L4_IPC_NEVER](#), [l4_msgtag\(\)](#), [L4_NOTHROW](#), [L4_PROTO_THREAD](#), and [l4_msg_regs_t::mr](#).

Referenced by [l4_thread_vcpu_control_ext\(\)](#), and [L4::Thread::vcpu_control_ext\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.1.10.13.3.16 l4_thread_vcpu_control_u()

```

l4_msgtag_t l4_thread_vcpu_control_u (
    l4_cap_idx_t thread,
    l4_addr_t vcpu_state,
    l4_utcb_t * utcb) [inline]
  
```

Enable the vCPU feature for the thread.

Parameters

| | |
|---------------|--|
| <i>thread</i> | Capability selector of the thread for which the vCPU feature shall be enabled. |
|---------------|--|

| | |
|-------------------|--|
| <i>vcpu_state</i> | A virtual address pointing to a l4_vcpu_state_t . It must be a valid kernel-user-memory address (see L4::Task::add_ku_mem()). |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb . |

Returns

Syscall return tag.

This function enables the vCPU feature of `this` thread

The kernel-user memory starting at `vcpu_state` must be at least 128-byte aligned and must cover the size of [l4_vcpu_state_t](#).

The asynchronous IPC handling is described at [vCPU API](#).

Note

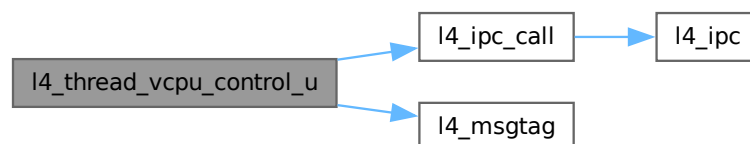
Disabling of the vCPU feature is optional and currently not supported.

Definition at line [1044](#) of file [thread.h](#).

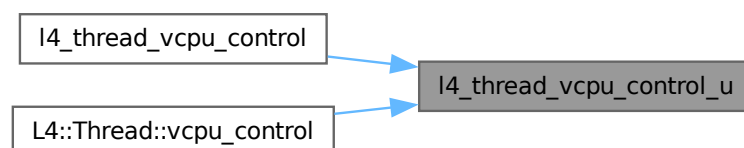
References [l4_ipc_call\(\)](#), [L4_IPC_NEVER](#), [l4_msgtag\(\)](#), [L4_NOTHROW](#), [L4_PROTO_THREAD](#), [L4_THREAD_VCPU_CONTROL_ON](#) and [l4_msg_regs_t::mr](#).

Referenced by [l4_thread_vcpu_control\(\)](#), and [L4::Thread::vcpu_control\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.1.10.13.3.17 `l4_thread_vcpu_resume_commit()`

```
l4_msgtag_t l4_thread_vcpu_resume_commit (
    l4_cap_idx_t thread,
    l4_msgtag_t tag) [inline]
```

Commit vCPU resume.

Parameters

| | |
|---------------|---|
| <i>thread</i> | Thread to be resumed, the invalid cap can be used for the current thread. |
| <i>tag</i> | Tag to use, returned by <code>l4_thread_vcpu_resume_start()</code> |

Returns

Syscall return tag containing one of the following return codes.

Return values

| | |
|-------------------|--|
| <i>0</i> | Indicates a VM exit, provided that <i>thread</i> is in extended vCPU mode with virtual interrupts cleared. |
| <i>1</i> | Indicates an incoming IPC message, provided that the <i>thread</i> is in extended vCPU mode with virtual interrupts cleared. |
| <i>-L4_EPERM</i> | The user task capability set in the vCPU state is missing the <code>L4_CAP_FPAGE_S</code> right. On Intel's VT-x (VMX): The vCPU context capability set in the extended vCPU state is missing the <code>L4_CAP_FPAGE_S</code> right. |
| <i>-L4_ENOENT</i> | The user task capability set in the vCPU state is invalid. |
| <i>-L4_EINVAL</i> | <i>thread</i> is not the current running thread, or does not have the vCPU feature enabled. On Intel's VT-x (VMX): No vCPU context associated with the extended vCPU state. |
| <i>-L4_EBUSY</i> | On Intel's VT-x (VMX): The vCPU context associated with the extended vCPU state is already active on a different CPU. |
| <i>-L4_ENODEV</i> | On Intel's VT-x (VMX): The vCPU context associated with the extended vCPU state cannot be initialized or activated. |
| <i><0</i> | A supplied mapping failed. |

All flexpages in the UTCB (added with `l4_sndfpage_add()` after `l4_thread_vcpu_resume_start()`) are unconditionally mapped into the user task configured in the vCPU state.

To resume into another address space, the capability to the target `Task` (or `L4::Vm`) must be set in `l4_vcpu_state_t::user_task` together with `L4_VCPU_F_USER_MODE`. The capability selector must have all lower bits clear (see `L4_CAP_MASK`). The kernel adds the `L4_SYSF_SEND` flag there to indicate that the capability has been referenced in the kernel. Consecutive resumes will not reference the task capability again until all lower bits are cleared again. To release a task use a different task capability or use an invalid capability with the `L4_SYSF_REPLY` flag set.

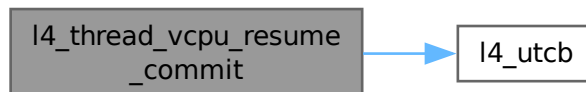
See also

[l4_vcpu_state_t](#)

Definition at line 1017 of file [thread.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



13.1.10.13.3.18 `l4_thread_vcpu_resume_start()`

```
l4_msgtag_t l4_thread_vcpu_resume_start (
    void ) [inline]
```

vCPU return from event handler.

Returns

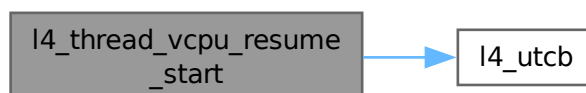
Message tag to be used for [l4_sndfpage_add\(\)](#) and [l4_thread_vcpu_resume_commit\(\)](#)

The vCPU resume functionality is split in multiple functions to allow the specification of additional send-flexpages using [l4_sndfpage_add\(\)](#).

Definition at line 1011 of file [thread.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



13.1.10.13.3.19 l4_thread_yield()

```
l4_msgtag_t l4_thread_yield (
    void ) [inline]
```

Yield current time slice.

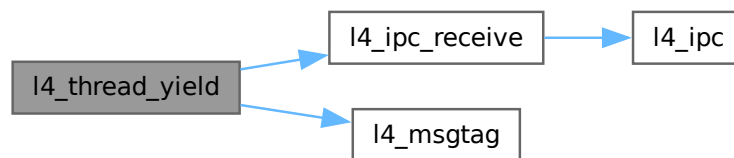
Returns

system call return tag

Definition at line 891 of file [thread.h](#).

References [L4_INVALID_CAP](#), [L4_IPC_BOTH_TIMEOUT_0](#), [l4_ipc_receive\(\)](#), [l4_msgtag\(\)](#), and [L4_NOTHROW](#).

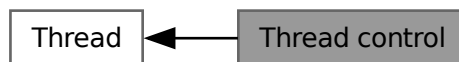
Here is the call graph for this function:



13.1.10.13.4 Thread control

API for Thread Control method.

Collaboration diagram for Thread control:



Functions

- void [l4_thread_control_start](#) (void) [L4_NOTHROW](#)
Start a thread control API sequence.
- void [l4_thread_control_pager](#) ([l4_cap_idx_t](#) pager) [L4_NOTHROW](#)
Set the pager.
- void [l4_thread_control_exc_handler](#) ([l4_cap_idx_t](#) exc_handler) [L4_NOTHROW](#)
Set the exception handler.
- void [l4_thread_control_bind](#) ([l4_utcb_t](#) *thread_utcb, [l4_cap_idx_t](#) task) [L4_NOTHROW](#)
Bind the thread to a task.
- void [l4_thread_control_alien](#) (int on) [L4_NOTHROW](#)
Enable alien mode.
- [l4_msgtag_t](#) [l4_thread_control_commit](#) ([l4_cap_idx_t](#) thread) [L4_NOTHROW](#)
Commit the thread control parameters.

13.1.10.13.4.1 Detailed Description

API for Thread Control method.

The thread control API provides access to almost any parameter of a thread object. The API is based on a single invocation of the thread object. However, because of the huge amount of parameters, the API provides a set of functions to set specific parameters of a thread and a commit function to commit the thread control call (see [l4_thread_control_commit\(\)](#)).

A thread control operation must always start with [l4_thread_control_start\(\)](#) and be committed with [l4_thread_control_commit\(\)](#). All other thread control parameter setter functions must be called between these two functions.

An example for a sequence of thread control API calls can be found below.

```
l4\_thread\_control\_start\(\);
l4\_thread\_control\_pager\(pager\_cap\);
l4\_thread\_control\_bind \(thread\_utcb, task\);
l4\_thread\_control\_commit\(thread\_cap\);
```

13.1.10.13.4.2 Function Documentation

[l4_thread_control_alien\(\)](#)

```
void l4_thread_control_alien (
    int on) [inline]
```

Enable alien mode.

Parameters

| | |
|-----------|--|
| <i>on</i> | Boolean value defining the state of the feature. |
|-----------|--|

For a thread in alien mode the kernel produces just an exception IPC for each IPC and exception caused by the alien thread instead of handling these events regularly. (Page faults of alien threads and interrupts occurring while the alien thread is running are always handled regularly.) While the alien thread is blocking, the exception handler can inspect and modify the state of the alien thread and potentially also the system call arguments. If the exception handler replies with [L4_PROTO_ALLOW_SYSCALL](#) as message tag, the kernel handles the next IPC or exception of the alien thread in a regular way. If the exception handler leaves certain thread state unchanged (in particular the instruction pointer), this will be the IPC or exception that caused the call of the exception handler. For a regularly processed IPC or exception of the alien thread the kernel also performs an exception IPC on kernel exit.

This feature can be used to attach a debugger to a thread and trace all object invocations and their results. It could also be used to handle other systems that use the same syscall instruction as [L4Re](#).

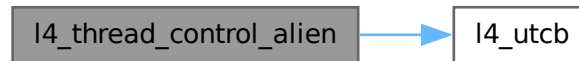
Examples

[examples/sys/aliens/main.c](#), and [examples/sys/singlestep/main.c](#).

Definition at line 981 of file [thread.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



l4_thread_control_bind()

```
void l4_thread_control_bind (
    l4_utcb_t * thread_utcb,
    l4_cap_idx_t task) [inline]
```

Bind the thread to a task.

Parameters

| | |
|--------------------|---|
| <i>thread_utcb</i> | The thread's UTCB address within the task it shall be bound to. The address must be aligned (architecture dependent; at least word aligned) and it must point to at least L4_UTCB_OFFSET bytes of kernel-user memory. |
| <i>task</i> | The task the thread shall be bound to. |

Precondition

The thread must not be bound to a task yet.

The capability `task` must have the permission [L4_CAP_FPAGE_S](#), otherwise the later call to [l4_thread_control_commit\(\)](#) will fail with [L4_EPERM](#).

A thread may execute code in the context of a task if and only if the thread is bound to the task. To actually start execution, [l4_thread_ex_regs\(\)](#) needs to be used. Execution in the context of the task means that the code has access to all the task's resources (and only those). The executed code itself must be one of those resources. A thread can be bound at most once to a task.

Note

The UTCBs of different threads in the same task should not overlap in order to prevent data corruption.

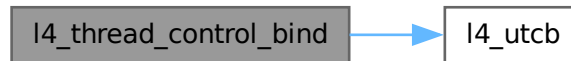
Examples

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 975 of file [thread.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



l4_thread_control_commit()

```
l4_msgtag_t l4_thread_control_commit (
    l4_cap_idx_t thread) [inline]
```

Commit the thread control parameters.

Parameters

| | |
|---------------|--|
| <i>thread</i> | Capability selector of target thread to commit to. |
|---------------|--|

Returns

Syscall return tag containing one of the following return codes.

Return values

| | |
|-------------------|---|
| <i>L4_EOK</i> | Operation successful. |
| <i>-L4_EPERM</i> | Insufficient permissions; see precondition. |
| <i>-L4_EINVAL</i> | Malformed thread control parameters. |

Precondition

The capability `thread` must have the permission [L4_CAP_FPAGE_S](#). When using [l4_thread_control_bind\(\)](#), also the respective task capability must have the permission [L4_CAP_FPAGE_S](#).

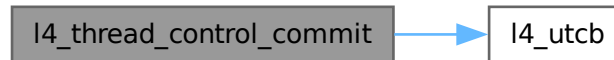
Examples

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 987 of file [thread.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



`l4_thread_control_exc_handler()`

```
void l4_thread_control_exc_handler (
    l4_cap_idx_t exc_handler) [inline]
```

Set the exception handler.

Parameters

| | |
|--------------------|---|
| <i>exc_handler</i> | Capability selector invoked to send an exception IPC. |
|--------------------|---|

Note

The exception-handler capability selector is interpreted in the task the thread is bound to (executes in).

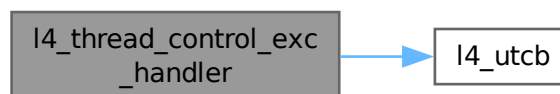
Examples

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 968 of file [thread.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



l4_thread_control_pager()

```
void l4_thread_control_pager (
    l4_cap_idx_t pager) [inline]
```

Set the pager.

Parameters

| | |
|--------------|---|
| <i>pager</i> | Capability selector invoked to send a page-fault IPC. |
|--------------|---|

Note

The pager capability selector is interpreted in the task the thread is bound to (executes in).

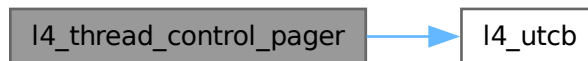
Examples

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 962 of file [thread.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:

**l4_thread_control_start()**

```
void l4_thread_control_start (
    void ) [inline]
```

Start a thread control API sequence.

This function starts a sequence of thread control API functions. After this functions any of following functions may be called in any order.

- [l4_thread_control_pager\(\)](#)
- [l4_thread_control_exc_handler\(\)](#)
- [l4_thread_control_bind\(\)](#)
- [l4_thread_control_alien\(\)](#)

To commit the changes to the thread [l4_thread_control_commit\(\)](#) must be called in the end.

Note

The thread control API calls store the parameters for the thread in the UTCB of the caller (see [l4_utcb\(\)](#)), this means between [l4_thread_control_start\(\)](#) and [l4_thread_control_commit\(\)](#) no functions that modify the UTCB contents must be called.

Examples

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 956 of file [thread.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:

**13.1.10.13.5 vCPU API**

vCPU API.

Collaboration diagram for vCPU API:

**Data Structures**

- struct [l4_vcpu_state_t](#)
State of a vCPU.
- struct [l4_vcpu_regs_t](#)
vCPU registers.
- struct [l4_vcpu_ipc_regs_t](#)
vCPU message registers.

Typedefs

- typedef struct l4_vcpu_state_t **l4_vcpu_state_t**
State of a vCPU.
- typedef struct l4_vcpu_regs_t **l4_vcpu_regs_t**
vCPU registers.
- typedef struct l4_vcpu_ipc_regs_t **l4_vcpu_ipc_regs_t**
vCPU message registers.
- typedef [l4_exc_regs_t](#) **l4_vcpu_regs_t**
vCPU registers.
- typedef struct l4_vcpu_ipc_regs_t **l4_vcpu_ipc_regs_t**
vCPU message registers.
- typedef struct l4_vcpu_regs_t **l4_vcpu_regs_t**
vCPU registers.
- typedef struct l4_vcpu_ipc_regs_t **l4_vcpu_ipc_regs_t**
vCPU message registers.
- typedef struct l4_vcpu_regs_t **l4_vcpu_regs_t**
vCPU registers.
- typedef struct l4_vcpu_ipc_regs_t **l4_vcpu_ipc_regs_t**
vCPU message registers.
- typedef [l4_exc_regs_t](#) **l4_vcpu_regs_t**
vCPU registers.
- typedef struct l4_vcpu_ipc_regs_t **l4_vcpu_ipc_regs_t**
vCPU message registers.

Enumerations

- enum [L4_vcpu_state_flags](#) {
 [L4_VCPU_F_IRQ](#) = 0x01 , [L4_VCPU_F_PAGE_FAULTS](#) = 0x02 , [L4_VCPU_F_EXCEPTIONS](#) = 0x04 ,
 [L4_VCPU_F_USER_MODE](#) = 0x20 ,
 [L4_VCPU_F_FPU_ENABLED](#) = 0x80 }
State flags of a vCPU.
- enum [L4_vcpu_sticky_flags](#) { [L4_VCPU_SF_IRQ_PENDING](#) = 0x01 }
Sticky flags of a vCPU.
- enum [L4_vcpu_state_offset](#) { [L4_VCPU_OFFSET_EXT_STATE](#) = 0x180 , [L4_VCPU_OFFSET_EXT_INFOS](#) = 0x100 }
Offsets for vCPU state layouts.
- enum [L4_vcpu_state_offset](#) { [L4_VCPU_OFFSET_EXT_STATE](#) = 0x280 , [L4_VCPU_OFFSET_EXT_INFOS](#) = 0x200 }
Offsets for vCPU state layouts.
- enum [L4_vcpu_state_offset](#) { [L4_VCPU_OFFSET_EXT_STATE](#) = 0x400 , [L4_VCPU_OFFSET_EXT_INFOS](#) = 0x200 }
Offsets for vCPU state layouts.
- enum [L4_vcpu_state_offset](#) { [L4_VCPU_OFFSET_EXT_STATE](#) = 0x400 , [L4_VCPU_OFFSET_EXT_INFOS](#) = 0x200 }
Offsets for vCPU state layouts.
- enum [L4_vcpu_state_offset](#) { [L4_VCPU_OFFSET_EXT_STATE](#) = 0x400 , [L4_VCPU_OFFSET_EXT_INFOS](#) = 0x200 }
Offsets for vCPU state layouts.

13.1.10.13.5.1 Detailed Description

vCPU API.

The vCPU API in [L4Re](#) implements virtual processors (vCPUs) on top of [L4::Thread](#). This API can be used for user level threading, operating system rehosting (see [L4Linux](#)) and virtualization.

You switch a thread into vCPU operation with [L4::Thread::vcpu_control](#).

In vCPU mode, incoming IPC can be redirected to a handler function. If an IPC is sent to the vCPU, the thread's normal execution is interrupted and the handler called. Which kind of IPC is redirected is specified by the [L4_vcpu_state_flags](#) set in the [l4_vcpu_state_t::state](#) field of [vcpu_state](#). All events enabled in the [vcpu_state](#) field are redirected to the handler. The handler is set via [l4_vcpu_state_t::entry_ip](#) and [l4_vcpu_state_t::entry_sp](#). IPC redirection works independent of "kernel" and "user" mode, but see [l4_vcpu_state_t::entry_sp](#). When the entry handler is called, the UTCB contains the result of the IPC and content normally found in CPU registers is in [l4_vcpu_state_t::i](#).

Furthermore, the thread can execute in the context of different tasks, called the "kernel" and the "user" mode. The kernel task is the one to which the thread was originally bound via [L4::Thread::control\(\)](#). Execution starts in the kernel task and it is always switched to when the asynchronous IPC handler is invoked. When returning from the handler via [l4_thread_vcpu_resume_start\(\)](#) and [l4_thread_vcpu_resume_commit\(\)](#), a different user task can be specified by setting [l4_vcpu_state_t::user_task](#) and enabling the [L4_VCPU_F_USER_MODE](#) flag in [l4_vcpu_state_t::state](#). Note that the kernel may cache the user task internally, see [l4_thread_vcpu_resume_commit\(\)](#).

If the [L4_VCPU_F_USER_MODE](#) flag is enabled, the following flags will be automatically enabled in [l4_vcpu_state_t::state](#) on [L4::Thread::vcpu_resume_commit\(\)](#):

- [L4_VCPU_F_IRQ](#)
- [L4_VCPU_F_PAGE_FAULTS](#)
- [L4_VCPU_F_EXCEPTIONS](#)

When the kernel mode is entered, the following flags will be automatically disabled in [l4_vcpu_state_t::state](#):

- [L4_VCPU_F_IRQ](#)
- [L4_VCPU_F_PAGE_FAULTS](#)
- [L4_VCPU_F_USER_MODE](#)

Extended vCPU operation is used for hardware CPU virtualization. It can be enabled with [L4::Thread::vcpu_control_ext\(\)](#).

[vCPU Support Library](#) defines a convenience API for working with vCPUs.

See also

[vCPU Support Library](#)

13.1.10.13.5.2 Enumeration Type Documentation

L4_vcpu_state_flags

enum [L4_vcpu_state_flags](#)

State flags of a vCPU.

Enumerator

| | |
|-----------------------|---|
| L4_VCPU_F_IRQ | <p>Receiving of IRQs and IPC enabled. While this flag is not set, the corresponding vCPU thread will not receive any IPC and threads attempting to send an IPC to this thread will block (according to the selected send timeout).</p> <p>Note</p> <p>On L4::Thread::vcpu_resume_commit() this flag is automatically enabled in l4_vcpu_state_t::state if L4_VCPU_F_USER_MODE is enabled.</p> <p>When the kernel mode is entered, this flag is automatically disabled in l4_vcpu_state_t::state.</p> |
| L4_VCPU_F_PAGE_FAULTS | <p>Page faults enabled. If this flag is set, a page fault switches to kernel mode (potentially causing a VM exit) and calls the entry handler. If this flag is not set, a page fault generates a page fault IPC to the pager of the vCPU thread.</p> <p>Note</p> <p>IPC redirection for page faults controlled by this flag works independent of "kernel" and "user" mode.</p> <p>On L4::Thread::vcpu_resume_commit() this flag is automatically enabled in l4_vcpu_state_t::state if L4_VCPU_F_USER_MODE is enabled.</p> <p>When the kernel mode is entered, this flag is automatically disabled in l4_vcpu_state_t::state.</p> |
| L4_VCPU_F_EXCEPTIONS | <p>Exceptions enabled. If this flag is set, then, on the event of an exception, the vCPU switches to kernel mode (potentially causing a VM exit) and calls the entry handler. If this flag is not set, an exception generates an exception IPC to the exception handler of the vCPU thread.</p> <p>Note</p> <p>IPC redirection for exceptions controlled by this flag works independent of "kernel" and "user" mode.</p> <p>On L4::Thread::vcpu_resume_commit() this flag is automatically enabled in l4_vcpu_state_t::state if L4_VCPU_F_USER_MODE is enabled.</p> |
| L4_VCPU_F_USER_MODE | <p>User task will be used. If set, the vCPU switches to user mode on next L4::Thread::vcpu_resume_commit(). If clear, the vCPU stays in "kernel" mode.</p> <p>Note</p> <p>When the kernel mode is entered, this flag is automatically disabled in l4_vcpu_state_t::state.</p> |

| | |
|-----------------------|--|
| L4_VCPU_F_FPU_ENABLED | FPU enabled. This flag is only relevant if L4_VCPU_F_USER_MODE is set. Setting this flag allows code in vCPU mode to use the FPU. If this flag is not set, any FPU operation will trigger a corresponding exception (FPU fault). |
|-----------------------|--|

Definition at line 101 of file [vcpu.h](#).

L4_vcpu_state_offset [1/5]

enum [L4_vcpu_state_offset](#)

Offsets for vCPU state layouts.

Enumerator

| | |
|--------------------------|-------------------------------------|
| L4_VCPU_OFFSET_EXT_STATE | Offset where extended state begins. |
| L4_VCPU_OFFSET_EXT_INFOS | Offset where extended infos begin. |

Definition at line 34 of file [__vcpu-arch.h](#).

L4_vcpu_state_offset [2/5]

enum [L4_vcpu_state_offset](#)

Offsets for vCPU state layouts.

Enumerator

| | |
|--------------------------|-------------------------------------|
| L4_VCPU_OFFSET_EXT_STATE | Offset where extended state begins. |
| L4_VCPU_OFFSET_EXT_INFOS | Offset where extended infos begin. |

Definition at line 35 of file [__vcpu-arch.h](#).

L4_vcpu_state_offset [3/5]

enum [L4_vcpu_state_offset](#)

Offsets for vCPU state layouts.

Enumerator

| | |
|--------------------------|-------------------------------------|
| L4_VCPU_OFFSET_EXT_STATE | Offset where extended state begins. |
| L4_VCPU_OFFSET_EXT_INFOS | Offset where extended infos begin. |

Definition at line 36 of file [__vcpu-arch.h](#).

L4_vcpu_state_offset [4/5]

```
enum L4_vcpu_state_offset
```

Offsets for vCPU state layouts.

Enumerator

| | |
|--------------------------|-------------------------------------|
| L4_VCPU_OFFSET_EXT_STATE | Offset where extended state begins. |
| L4_VCPU_OFFSET_EXT_INFOS | Offset where extended infos begin. |

Definition at line 34 of file [__vcpu-arch.h](#).

L4_vcpu_state_offset [5/5]

```
enum L4_vcpu_state_offset
```

Offsets for vCPU state layouts.

Enumerator

| | |
|--------------------------|-------------------------------------|
| L4_VCPU_OFFSET_EXT_STATE | Offset where extended state begins. |
| L4_VCPU_OFFSET_EXT_INFOS | Offset where extended infos begin. |

Definition at line 34 of file [__vcpu-arch.h](#).

L4_vcpu_sticky_flags

```
enum L4_vcpu_sticky_flags
```

Sticky flags of a vCPU.

Enumerator

| | |
|------------------------|---|
| L4_VCPU_SF_IRQ_PENDING | An event is pending: Either an IRQ or another thread attempts to send an IPC to this vCPU thread. |
|------------------------|---|

Definition at line 167 of file [vcpu.h](#).

13.1.10.14 Thread groups

C thread group interface, see [L4::Thread_group](#) for the C++ interface.

Collaboration diagram for Thread groups:



Functions

- [l4_msgtag_t l4_thread_group_add](#) ([l4_cap_idx_t](#) tg, [l4_cap_idx_t](#) thread) [L4_NOTHROW](#)
Add thread to a thread group.
- [l4_msgtag_t l4_thread_group_remove](#) ([l4_cap_idx_t](#) tg, [l4_cap_idx_t](#) thread) [L4_NOTHROW](#)
Remove thread from a thread group.

13.1.10.14.1 Detailed Description

C thread group interface, see [L4::Thread_group](#) for the C++ interface.

An [L4](#) thread group is a collection of threads used as indirection for IPC gate and IRQ objects such that these objects can have multiple receivers, from which the kernel selects one according to a policy.

The primary use case for thread groups are multi-threaded servers and CPU core local IRQ / IPC delivery.

A thread can be bound to at most one thread group. Before binding a thread to a thread group, the thread must be bound to a task. All threads bound to the same thread group must belong to the same task.

13.1.10.14.2 Function Documentation

13.1.10.14.2.1 l4_thread_group_add()

```
l4\_msgtag\_t l4_thread_group_add (
    l4\_cap\_idx\_t tg,
    l4\_cap\_idx\_t thread) [inline]
```

Add thread to a thread group.

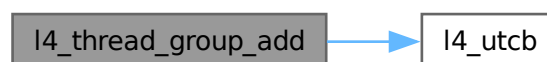
Parameters

| | |
|---------------|------------------------------------|
| <i>tg</i> | Thread group object. |
| <i>thread</i> | Thread to add to the thread group. |

Definition at line 114 of file [thread_group.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



13.1.10.14.2.2 l4_thread_group_remove()

```
l4_msgtag_t l4_thread_group_remove (
    l4_cap_idx_t tg,
    l4_cap_idx_t thread) [inline]
```

Remove thread from a thread group.

Parameters

| | |
|---------------|---|
| <i>tg</i> | Thread group object. |
| <i>thread</i> | Thread to remove from the thread group. |

Definition at line 121 of file [thread_group.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



13.1.10.15 Virtual Console

C Virtual console interface for simple character based input and output, see [L4::Vcon](#) for the C++ interface.

Collaboration diagram for Virtual Console:



Data Structures

- struct [l4_vcon_attr_t](#)
Vcon attribute structure.

Typedefs

- typedef struct l4_vcon_attr_t l4_vcon_attr_t
Vcon attribute structure.

Enumerations

- enum L4_vcon_size_consts { L4_VCON_WRITE_SIZE = (L4_UTCB_GENERIC_DATA_SIZE - 2) * sizeof(l4_umword_t) , L4_VCON_READ_SIZE = (L4_UTCB_GENERIC_DATA_SIZE - 1) * sizeof(l4_umword_t) }
Size constants.
- enum L4_vcon_i_flags { L4_VCON_INLCR = 000100 , L4_VCON_IGNCR = 000200 , L4_VCON_ICRNL = 000400 }
Input flags.
- enum L4_vcon_o_flags { L4_VCON_ONLCR = 000004 , L4_VCON_OCRNL = 000010 , L4_VCON_ONLRET = 000040 }
Output flags.
- enum L4_vcon_l_flags { L4_VCON_ICANON = 000002 , L4_VCON_ECHO = 000010 }
Local flags.

Functions

- l4_msgtag_t l4_vcon_send (l4_cap_idx_t vcon, char const *buf, unsigned size) L4_NOTHROW
Send data to virtual console.
- l4_msgtag_t l4_vcon_send_u (l4_cap_idx_t vcon, char const *buf, unsigned size, l4_utcb_t *utcb) L4_NOTHROW
*Send data to *this* virtual console.*
- long l4_vcon_write (l4_cap_idx_t vcon, char const *buf, unsigned size) L4_NOTHROW
Write data to virtual console.
- long l4_vcon_write_u (l4_cap_idx_t vcon, char const *buf, unsigned size, l4_utcb_t *utcb) L4_NOTHROW
*Write data to *this* virtual console.*
- int l4_vcon_read (l4_cap_idx_t vcon, char *buf, unsigned size) L4_NOTHROW
Read data from virtual console.
- int l4_vcon_read_u (l4_cap_idx_t vcon, char *buf, unsigned size, l4_utcb_t *utcb) L4_NOTHROW
*Read data from *this* virtual console.*
- int l4_vcon_read_with_flags (l4_cap_idx_t vcon, char *buf, unsigned size) L4_NOTHROW
Read data from virtual console, extended version including flags.
- l4_msgtag_t l4_vcon_set_attr (l4_cap_idx_t vcon, l4_vcon_attr_t const *attr) L4_NOTHROW
Set attributes of a Vcon.
- l4_msgtag_t l4_vcon_set_attr_u (l4_cap_idx_t vcon, l4_vcon_attr_t const *attr, l4_utcb_t *utcb) L4_NOTHROW
*Set the attributes of *this* virtual console.*
- l4_msgtag_t l4_vcon_get_attr (l4_cap_idx_t vcon, l4_vcon_attr_t *attr) L4_NOTHROW
Get attributes of a Vcon.
- l4_msgtag_t l4_vcon_get_attr_u (l4_cap_idx_t vcon, l4_vcon_attr_t *attr, l4_utcb_t *utcb) L4_NOTHROW
*Get attributes of *this* virtual console.*
- void l4_vcon_set_attr_raw (l4_vcon_attr_t *attr) L4_NOTHROW
Set terminal attributes to disable all special processing.

13.1.10.15.1 Detailed Description

C Virtual console interface for simple character based input and output, see [L4::Vcon](#) for the C++ interface.

The interrupt for read events is provided by the virtual key interrupt which, in contrast to hardware IRQs, implements a limited functionality:

- Only IRQ line 0 is supported, no MSI vectors.
- The IRQ is edge-triggered and the IRQ mode cannot be changed.
- As the IRQ is edge-triggered, it does not have to be explicitly unmasked.

A server implementing the virtual console protocol has a queue for input events. When the first input event is added to the empty queue, the virtual key interrupt is triggered. Further events are added to the queue without generating further interrupts. The queue is emptied when a client reads all queued input events.

Include File

```
#include <l4/sys/vcon.h>
```

See [L4::Vcon](#) for the C++ interface.

13.1.10.15.2 Typedef Documentation

13.1.10.15.2.1 l4_vcon_attr_t

```
typedef struct l4_vcon_attr_t l4_vcon_attr_t
```

Vcon attribute structure.

The flags members can be a combination of their respective enums.

See also

[L4_vcon_i_flags](#)

[L4_vcon_o_flags](#)

[L4_vcon_l_flags](#)

13.1.10.15.3 Enumeration Type Documentation

13.1.10.15.3.1 L4_vcon_i_flags

enum [L4_vcon_i_flags](#)

Input flags.

Enumerator

| | |
|---------------|---|
| L4_VCON_INLCR | Translate NL to CR. |
| L4_VCON_IGNCR | Ignore CR. |
| L4_VCON_ICRNL | Translate CR to NL if L4_VCON_IGNCR is not set. |

Definition at line [208](#) of file [vcon.h](#).

13.1.10.15.3.2 L4_vcon_l_flags

enum [L4_vcon_l_flags](#)

Local flags.

Enumerator

| | |
|----------------|-----------------|
| L4_VCON_ICANON | Canonical mode. |
| L4_VCON_ECHO | Echo input. |

Definition at line [230](#) of file [vcon.h](#).

13.1.10.15.3.3 L4_vcon_o_flags

enum [L4_vcon_o_flags](#)

Output flags.

Enumerator

| | |
|----------------|------------------------|
| L4_VCON_ONLCR | Translate NL to CR-NL. |
| L4_VCON_OCRNL | Translate CR to NL. |
| L4_VCON_ONLRET | Do not output CR. |

Definition at line [219](#) of file [vcon.h](#).

13.1.10.15.3.4 L4_vcon_size_consts

```
enum L4_vcon_size_consts
```

Size constants.

Enumerator

| | |
|--------------------|---|
| L4_VCON_WRITE_SIZE | Maximum size that can be written with one l4_vcon_write call. |
| L4_VCON_READ_SIZE | Maximum size that can be read with one l4_vcon_read* call. |

Definition at line 95 of file [vcon.h](#).

13.1.10.15.4 Function Documentation

13.1.10.15.4.1 l4_vcon_get_attr()

```
l4_msgtag_t l4_vcon_get_attr (
    l4_cap_idx_t vcon,
    l4_vcon_attr_t * attr) [inline]
```

Get attributes of a Vcon.

Parameters

| | | |
|-----|-------------|----------------------|
| | <i>vcon</i> | Vcon object. |
| out | <i>attr</i> | Attribute structure. |

Returns

Syscall return tag

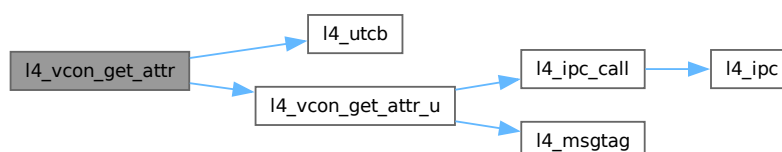
Examples

[examples/sys/isr/main.c](#).

Definition at line 435 of file [vcon.h](#).

References [L4_NOTHROW](#), [l4_utcb\(\)](#), and [l4_vcon_get_attr_u\(\)](#).

Here is the call graph for this function:



13.1.10.15.4.2 `l4_vcon_get_attr_u()`

```
l4_msgtag_t l4_vcon_get_attr_u (
    l4_cap_idx_t vcon,
    l4_vcon_attr_t * attr,
    l4_utcb_t * utcb) [inline]
```

Get attributes of this virtual console.

Parameters

| | | |
|-----|-------------|--|
| | <i>vcon</i> | Capability index of the vcon object. |
| out | <i>attr</i> | Attribute structure. Contains the attributes after a successful call of this function. |
| | <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb . |

Returns

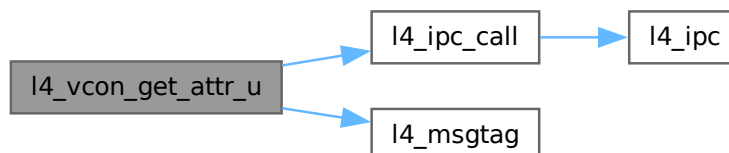
Syscall return tag.

Definition at line 417 of file [vcon.h](#).

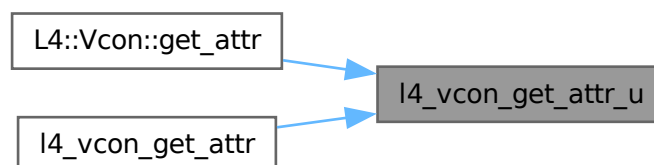
References [l4_ipc_call\(\)](#), [L4_IPC_NEVER](#), [l4_msgtag\(\)](#), [L4_NOTHROW](#), [L4_PROTO_LOG](#), [L4_VCON_GET_ATTR_OP](#), and [l4_msg_regs_t::mr](#).

Referenced by [L4::Vcon::get_attr\(\)](#), and [l4_vcon_get_attr\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.1.10.15.4.3 l4_vcon_read()

```
int l4_vcon_read (
    l4_cap_idx_t vcon,
    char * buf,
    unsigned size) [inline]
```

Read data from virtual console.

Parameters

| | | |
|-----|-------------|--------------------------|
| | <i>vcon</i> | Vcon object. |
| out | <i>buf</i> | Pointer to data buffer. |
| | <i>size</i> | Size of buffer in bytes. |

Return values

| | |
|------------------------|--|
| <code>-L4_EPERM</code> | Insufficient permissions; see precondition. |
| <code>>size</code> | More bytes to read, <code>size</code> bytes are in the buffer <code>buf</code> . |
| <code><=size</code> | Number of bytes read. |

Precondition

The capability `vcon` must have the permission `L4_CAP_FPAGE_W`.

Note

Size must not exceed `L4_VCON_READ_SIZE`.

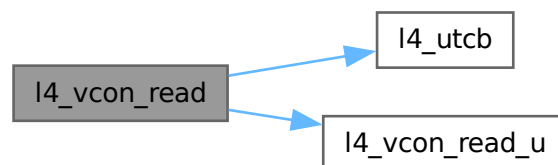
Examples

[examples/sys/isr/main.c](#).

Definition at line 391 of file [vcon.h](#).

References [L4_NOTHROW](#), [l4_utcb\(\)](#), and [l4_vcon_read_u\(\)](#).

Here is the call graph for this function:



13.1.10.15.4.4 l4_vcon_read_u()

```
int l4_vcon_read_u (
    l4_cap_idx_t vcon,
    char * buf,
    unsigned size,
    l4_utcb_t * utcb) [inline]
```

Read data from this virtual console.

Parameters

| | | |
|-----|-------------|--|
| | <i>vcon</i> | Capability index of the vcon object. |
| out | <i>buf</i> | Pointer to data buffer. |
| | <i>size</i> | Size of the data buffer in bytes. |
| | <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb . |

Return values

| | |
|------------------|--|
| <i>-L4_EPERM</i> | Insufficient permissions; see precondition. |
| <i>>size</i> | More bytes to read, <i>size</i> bytes are in the buffer <i>buf</i> . |
| <i><=size</i> | Number of bytes read. |

Precondition

The invoked Vcon capability must have the permission [L4_CAP_FPAGE_W](#).

Note

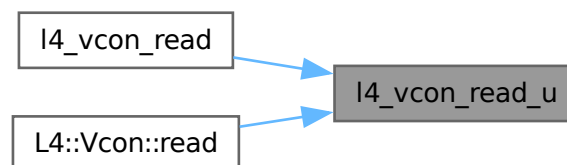
Size must not exceed [L4_VCON_READ_SIZE](#).

Definition at line 381 of file [vcon.h](#).

References [L4_NOTHROW](#), and [L4_VCON_READ_SIZE_MASK](#).

Referenced by [l4_vcon_read\(\)](#), and [L4::Vcon::read\(\)](#).

Here is the caller graph for this function:



13.1.10.15.4.5 l4_vcon_read_with_flags()

```
int l4_vcon_read_with_flags (
    l4_cap_idx_t vcon,
    char * buf,
    unsigned size) [inline]
```

Read data from virtual console, extended version including flags.

Parameters

| | | |
|-----|-------------|--------------------------|
| | <i>vcon</i> | Vcon object. |
| out | <i>buf</i> | Pointer to data buffer. |
| | <i>size</i> | Size of buffer in bytes. |

If this function returns a positive value the caller can check the [L4_VCON_READ_STAT_BREAK](#) flag bit for a break condition. The bytes read can be obtained by masking the return value with [L4_VCON_READ_SIZE_MASK](#).

If a break condition is signaled, it is always the first event in the transmitted content, i.e. all characters supplied by this read call follow the break condition.

buf might be a NULL, in this case the input data will be dropped.

Note

Size must not exceed [L4_VCON_READ_SIZE](#).

Return values

| | |
|------------------|--|
| <i>-L4_EPERM</i> | Insufficient permissions; see precondition. |
| <i>>size</i> | More bytes to read, <i>size</i> bytes are in the buffer <i>buf</i> . |
| <i><=size</i> | Number of bytes read. |

Precondition

The capability *vcon* must have the permission [L4_CAP_FPAGE_W](#).

Definition at line [375](#) of file [vcon.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



13.1.10.15.4.6 `l4_vcon_send()`

```
l4_msgtag_t l4_vcon_send (
    l4_cap_idx_t vcon,
    char const * buf,
    unsigned size) [inline]
```

Send data to virtual console.

Parameters

| | |
|-------------|--------------------------|
| <i>vcon</i> | Vcon object. |
| <i>buf</i> | Pointer to data buffer. |
| <i>size</i> | Size of buffer in bytes. |

Returns

Syscall return tag

Note

Size must not exceed `L4_VCON_WRITE_SIZE`, a proper value of the `size` parameter is NOT checked. Also, this function is a send only operation, this means there is no return value except for a failed send operation. Use `l4_ipc_error()` to check for send errors, and **do not** use `l4_error()`.

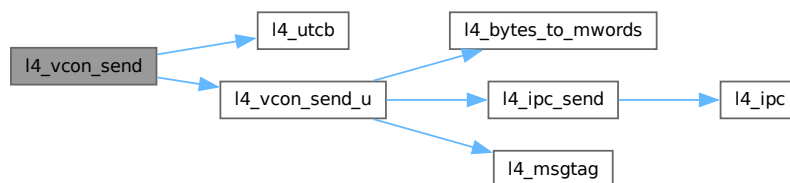
Examples

[examples/sys/utcb-ipc/main.c](#).

Definition at line 315 of file [vcon.h](#).

References [L4_NOTHROW](#), [l4_utcb\(\)](#), and [l4_vcon_send_u\(\)](#).

Here is the call graph for this function:



13.1.10.15.4.7 l4_vcon_send_u()

```
l4_msgtag_t l4_vcon_send_u (
    l4_cap_idx_t vcon,
    char const * buf,
    unsigned size,
    l4_utcb_t * utcb) [inline]
```

Send data to this virtual console.

Parameters

| | |
|-------------|--|
| <i>vcon</i> | Capability index of the Vcon object. |
| <i>buf</i> | Pointer to the data buffer. |
| <i>size</i> | Size of the data buffer in bytes. |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb . |

Returns

Syscall return tag

Note

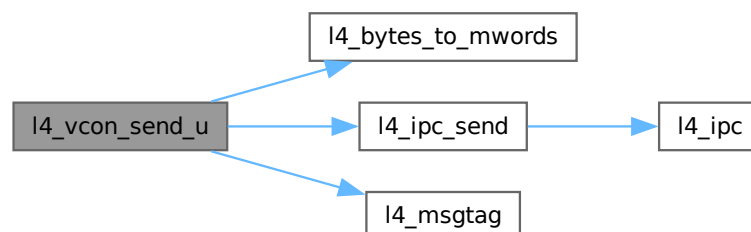
Size must not exceed [L4_VCON_WRITE_SIZE](#), a proper value of the `size` parameter is NOT checked. Also, this function is a send only operation, this means there is no return value except for a failed send operation. Use [l4_ipc_error\(\)](#) to check for send errors, do not use [l4_error\(\)](#), as [l4_error\(\)](#) will always return an error.

Definition at line [302](#) of file [vcon.h](#).

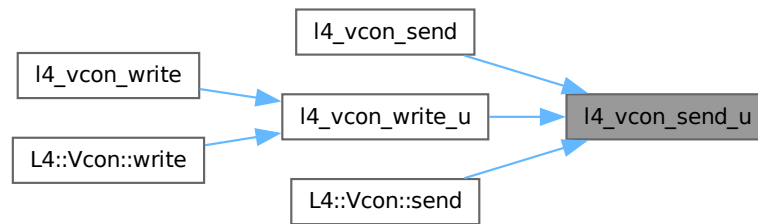
References [l4_bytes_to_mwords\(\)](#), [L4_IPC_NEVER](#), [l4_ipc_send\(\)](#), [l4_msgtag\(\)](#), [L4_MSGTAG_SCHEDULE](#), [L4_NOTHROW](#), [L4_PROTO_LOG](#), [L4_VCON_WRITE_OP](#), and [l4_msg_regs_t::mr](#).

Referenced by [l4_vcon_send\(\)](#), [l4_vcon_write_u\(\)](#), and [L4::Vcon::send\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.1.10.15.4.8 l4_vcon_set_attr()

```

l4_msgtag_t l4_vcon_set_attr (
    l4_cap_idx_t vcon,
    l4_vcon_attr_t const * attr) [inline]
  
```

Set attributes of a Vcon.

Parameters

| | |
|-------------|----------------------|
| <i>vcon</i> | Vcon object. |
| <i>attr</i> | Attribute structure. |

Returns

Syscall return tag

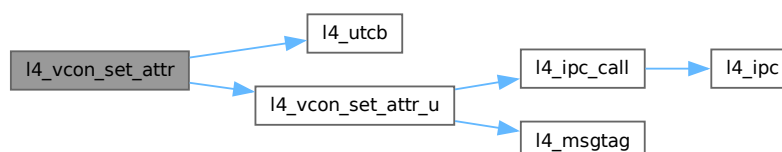
Examples

[examples/sys/isr/main.c](#).

Definition at line 411 of file [vcon.h](#).

References [L4_NOTHROW](#), [l4_utcb\(\)](#), and [l4_vcon_set_attr_u\(\)](#).

Here is the call graph for this function:



13.1.10.15.4.9 l4_vcon_set_attr_raw()

```
void l4_vcon_set_attr_raw (
    l4_vcon_attr_t * attr) [inline]
```

Set terminal attributes to disable all special processing.

Removes all flags that would mangle the read or written characters. Also disables echoing and any special processing of characters.

Parameters

| | | |
|----------------|-------------|--------------------------------|
| <i>in, out</i> | <i>attr</i> | Attribute structure to update. |
|----------------|-------------|--------------------------------|

Definition at line 441 of file [vcon.h](#).

References [L4_NOTHROW](#).

Referenced by [l4_vcon_attr_t::set_raw\(\)](#).

Here is the caller graph for this function:



13.1.10.15.4.10 l4_vcon_set_attr_u()

```
l4_msgtag_t l4_vcon_set_attr_u (
    l4_cap_idx_t vcon,
    l4_vcon_attr_t const * attr,
    l4_utcb_t * utcb) [inline]
```

Set the attributes of `this` virtual console.

Parameters

| | |
|-------------|--|
| <i>vcon</i> | Capability index of the vcon object. |
| <i>attr</i> | Attribute structure with the attributes for the virtual console. |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb . |

Returns

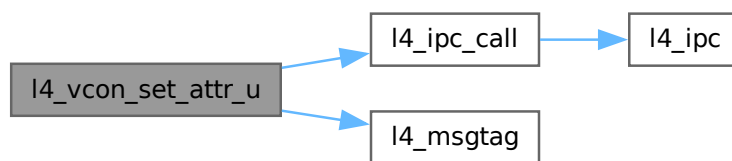
Syscall return tag.

Definition at line 397 of file [vcon.h](#).

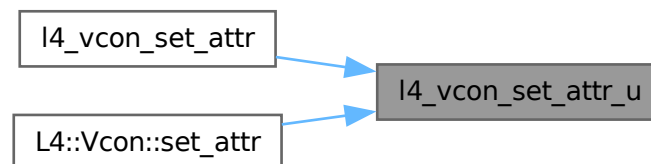
References [l4_ipc_call\(\)](#), [L4_IPC_NEVER](#), [l4_msgtag\(\)](#), [L4_NOTHROW](#), [L4_PROTO_LOG](#), [L4_VCON_SET_ATTR_OP](#), and [l4_msg_regs_t::mr](#).

Referenced by [l4_vcon_set_attr\(\)](#), and [L4::Vcon::set_attr\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**13.1.10.15.4.11 l4_vcon_write()**

```

long l4_vcon_write (
    l4_cap_idx_t vcon,
    char const * buf,
    unsigned size) [inline]
  
```

Write data to virtual console.

Parameters

| | |
|-------------|--------------------------|
| <i>vcon</i> | Vcon object. |
| <i>buf</i> | Pointer to data buffer. |
| <i>size</i> | Size of buffer in bytes. |

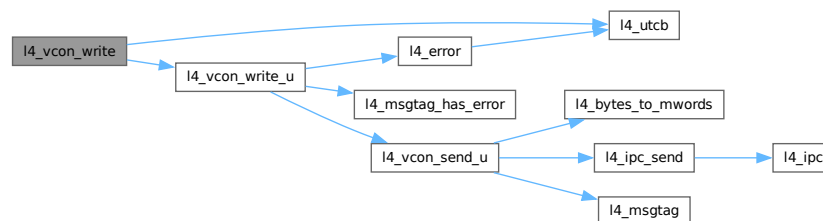
Return values

| | |
|----------|--|
| <0 | Error. |
| ≥ 0 | Number of bytes written to the virtual console |

Definition at line 336 of file [vcon.h](#).

References [L4_NOTHROW](#), [l4_utcb\(\)](#), and [l4_vcon_write_u\(\)](#).

Here is the call graph for this function:



13.1.10.15.4.12 l4_vcon_write_u()

```

long l4_vcon_write_u (
    l4_cap_idx_t vcon,
    char const * buf,
    unsigned size,
    l4_utcb_t * utcb) [inline]

```

Write data to `this` virtual console.

Parameters

| | |
|-------------|--|
| <i>vcon</i> | Capability index of the vcon object. |
| <i>buf</i> | Pointer to the data buffer. |
| <i>size</i> | Size of the data buffer in bytes. |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb . |

Return values

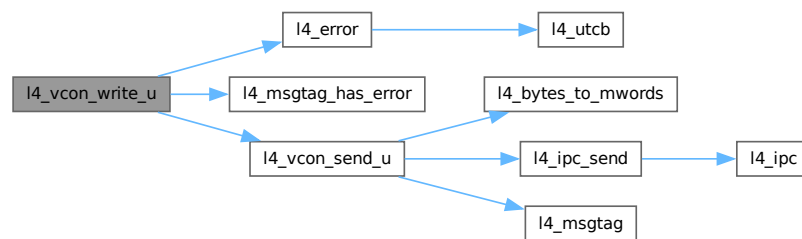
| | |
|----------|---|
| < 0 | Error. |
| ≥ 0 | Number of bytes written to the virtual console. |

Definition at line 321 of file [vcon.h](#).

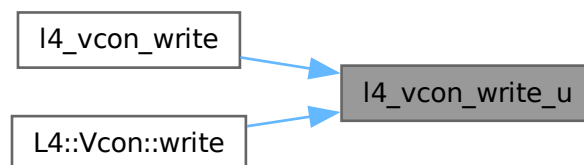
References [l4_error\(\)](#), [l4_msgtag_has_error\(\)](#), [L4_NOTHROW](#), [l4_vcon_send_u\(\)](#), and [L4_VCON_WRITE_SIZE](#).

Referenced by [l4_vcon_write\(\)](#), and [L4::Vcon::write\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.1.11 Kernel Interface Page

Kernel Interface Page.

Collaboration diagram for Kernel Interface Page:



Topics

- [Memory descriptors \(C version\)](#) 440
C Interface for KIP memory descriptors.

Data Structures

- class [L4::Kip::Mem_desc](#)
Memory descriptors stored in the kernel interface page.
- struct [l4_kernel_info_t](#)
[L4](#) Kernel Interface Page.

Macros

- `#define L4_KERNEL_INFO_MAGIC (0x4BE6344CL) /* "L4μK" */`
Kernel Info Page identifier ("L4μK").

Typedefs

- typedef struct [l4_kernel_info_t](#) [l4_kernel_info_t](#)
[L4](#) Kernel Interface Page.

Enumerations

- enum { [L4_KIP_OFFS_READ_US](#) = 0x900 , [L4_KIP_OFFS_READ_NS](#) = 0x980 }

Functions

- [l4_kernel_info_t](#) const * [l4_kip](#) (void) [L4_NOTHROW](#)
Get Kernel Info Page.
- [l4_umword_t](#) [l4_kip_version](#) ([l4_kernel_info_t](#) const *kip) [L4_NOTHROW](#)
Get the kernel version.
- const char * [l4_kip_version_string](#) ([l4_kernel_info_t](#) const *kip) [L4_NOTHROW](#)
Get the kernel version string.
- int [l4_kernel_info_version_offset](#) ([l4_kernel_info_t](#) const *kip) [L4_NOTHROW](#)
Return offset in bytes of version_strings relative to the KIP base.
- [l4_cpu_time_t](#) [l4_kip_clock](#) ([l4_kernel_info_t](#) const *kip) [L4_NOTHROW](#)
Return clock value from the KIP.
- [l4_umword_t](#) [l4_kip_clock_lw](#) ([l4_kernel_info_t](#) const *kip) [L4_NOTHROW](#)
Return least significant machine word of clock value from the KIP.
- [l4_uint64_t](#) [l4_kip_clock_ns](#) ([l4_kernel_info_t](#) const *kip) [L4_NOTHROW](#)
Return current clock using the KIP in nanoseconds.

13.1.11.1 Detailed Description

Kernel Interface Page.

C interface for the Kernel Interface Page:

C++ interface for the Kernel Interface Page:

Include File

```
#include <l4/sys/kip>
```

Include File

```
#include <l4/sys/kip.h>
```

13.1.11.2 Typedef Documentation

13.1.11.2.1 l4_kernel_info_t

```
typedef struct l4_kernel_info_t l4_kernel_info_t
```

[L4](#) Kernel Interface Page.

32-bit architecture may assume that the upper 32 bits of addresses is 0

13.1.11.3 Enumeration Type Documentation

13.1.11.3.1 anonymous enum

```
anonymous enum
```

Enumerator

| | |
|---------------------|---|
| L4_KIP_OFFS_READ_US | Offset of KIP code (provided by the kernel) for reading the KIP clock in microseconds. If the kernel is configured for a fine-grained KIP clock (CONFIG_SYNC↔TSC enabled for IA32, ARM_SYNC_CLOCK for ARM), this code provides the KIP clock with microseconds granularity and accuracy by reading the hardware clock used by the kernel and transforming this value into microseconds. Otherwise this code just reads the KIP clock value. |
| L4_KIP_OFFS_READ_NS | Offset of KIP code (provided by the kernel) for reading the time stamp counter and transforming this value into nanoseconds. If the kernel is configured for fine-grained KIP clock (CONFIG_SYNC enabled for IA32, ARM_SYNC_CLOCK for ARM), this code provides the KIP clock with nanoseconds granularity and accuracy by reading the hardware clock used by the kernel and transforming this value into nanoseconds. Otherwise this code just reads the KIP clock value and multiplies it by 1000. |

Definition at line 95 of file [kip.h](#).

13.1.11.4 Function Documentation

13.1.11.4.1 l4_kernel_info_version_offset()

```
int l4_kernel_info_version_offset (
    l4_kernel_info_t const * kip) [inline]
```

Return offset in bytes of version_strings relative to the KIP base.

Parameters

| | |
|------------|--|
| <i>kip</i> | Pointer to the kernel info page (KIP). |
|------------|--|

Returns

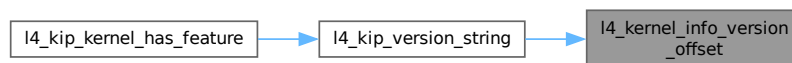
offset of version_strings relative to the KIP base address, in bytes.

Definition at line 238 of file [kip.h](#).

References [L4_NOTHROW](#).

Referenced by [l4_kip_version_string\(\)](#).

Here is the caller graph for this function:



13.1.11.4.2 l4_kip()

```
l4_kernel_info_t const * l4_kip (
    void ) [inline]
```

Get Kernel Info Page.

Returns

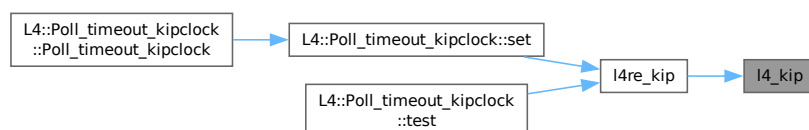
Pointer to Kernel Info Page (KIP) structure.

Definition at line 226 of file [kip.h](#).

References [L4_NOTHROW](#).

Referenced by [l4re_kip\(\)](#).

Here is the caller graph for this function:



13.1.11.4.3 l4_kip_clock()

```
l4_cpu_time_t l4_kip_clock (
    l4_kernel_info_t const * kip) [inline]
```

Return clock value from the KIP.

Parameters

| | |
|------------|--|
| <i>kip</i> | Pointer to the kernel info page (KIP). |
|------------|--|

Returns

Value of the clock field in the KIP.

The KIP clock always contains the current (relative) time in micro seconds independently of the CPU frequency. The clock is only guaranteed to be accurate within the scheduling granularity announced in the KIP.

This function basically calls the KIP code for reading the KIP clock with microseconds resolution. The accuracy depends on the platform and the kernel configuration.

See also

[L4_KIP_OFFS_READ_US](#).

Examples

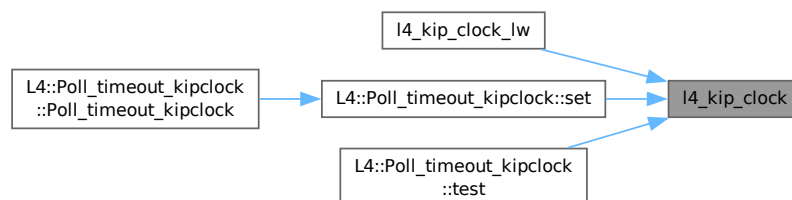
[examples/libs/shmc/prodcons.c](#).

Definition at line 242 of file [kip.h](#).

References [L4_KIP_OFFS_READ_US](#), and [L4_NOTHROW](#).

Referenced by [l4_kip_clock_lw\(\)](#), [L4::Poll_timeout_kipclock::set\(\)](#), and [L4::Poll_timeout_kipclock::test\(\)](#).

Here is the caller graph for this function:



13.1.11.4.4 l4_kip_clock_lw()

```
l4_umword_t l4_kip_clock_lw (
    l4_kernel_info_t const * kip) [inline]
```

Return least significant machine word of clock value from the KIP.

Parameters

| | |
|------------|--|
| <i>kip</i> | Pointer to the kernel info page (KIP). |
|------------|--|

Returns

Lower machine word of clock value from the KIP.

Deprecated Use [l4_kip_clock\(\)](#) instead.

This function will always provide the least significant machine word of the clock value from the KIP, regardless of the kernel configuration.

Definition at line 261 of file [kip.h](#).

References [l4_kip_clock\(\)](#), and [L4_NOTHROW](#).

Here is the call graph for this function:



13.1.11.4.5 l4_kip_clock_ns()

```
l4_cpu_time_t l4_kip_clock_ns (
    l4_kernel_info_t const * kip) [inline]
```

Return current clock using the KIP in nanoseconds.

Parameters

| | |
|------------|--|
| <i>kip</i> | Pointer to the kernel info page (KIP). |
|------------|--|

Returns

Value of the current clock in nanoseconds.

This function basically calls the KIP code for reading the KIP clock with nanoseconds resolution. The accuracy depends on the platform and the kernel configuration.

See also

[L4_KIP_OFFS_READ_NS](#).

Definition at line 252 of file [kip.h](#).

References [L4_KIP_OFFS_READ_NS](#), and [L4_NOTHROW](#).

13.1.11.4.6 l4_kip_version()

```
l4_umword_t l4_kip_version (
    l4_kernel_info_t const * kip) [inline]
```

Get the kernel version.

Parameters

| | |
|------------|-------------------|
| <i>kip</i> | Kernel Info Page. |
|------------|-------------------|

Returns

Kernel version string. 0 if KIP could not be mapped.

Definition at line 230 of file [kip.h](#).

References [L4_NOTHROW](#).

13.1.11.4.7 l4_kip_version_string()

```
const char * l4_kip_version_string (
    l4_kernel_info_t const * kip) [inline]
```

Get the kernel version string.

Parameters

| | |
|------------|-------------------|
| <i>kip</i> | Kernel Info Page. |
|------------|-------------------|

Returns

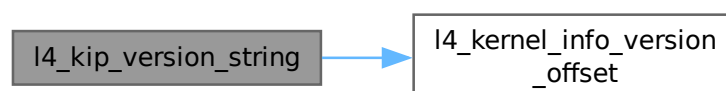
Kernel version string.

Definition at line 234 of file [kip.h](#).

References [l4_kernel_info_version_offset\(\)](#), and [L4_NOTHROW](#).

Referenced by [l4_kip_kernel_has_feature\(\)](#).

Here is the call graph for this function:



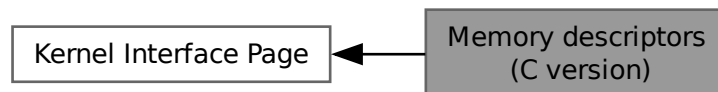
Here is the caller graph for this function:



13.1.11.5 Memory descriptors (C version)

C Interface for KIP memory descriptors.

Collaboration diagram for Memory descriptors (C version):



Data Structures

- struct `l4_kernel_info_mem_desc_t`
Memory descriptor data structure.

Typedefs

- typedef struct `l4_kernel_info_mem_desc_t` `l4_kernel_info_mem_desc_t`
Memory descriptor data structure.

Enumerations

- enum `l4_mem_type_t` {
`l4_mem_type_undefined` = 0x0 , `l4_mem_type_conventional` = 0x1 , `l4_mem_type_reserved` = 0x2 ,
`l4_mem_type_dedicated` = 0x3 ,
`l4_mem_type_shared` = 0x4 , `l4_mem_type_info` = 0xd , `l4_mem_type_bootloader` = 0xe , `l4_mem_type_archspecific`
= 0xf }
Type of a memory descriptor.
- enum `l4_mem_info_sub_type_t` { `l4_mem_info_acpi_rsdp` = 0 , `l4_mem_reserved_kernel` = 0 ,
`l4_mem_reserved_heap` = 1 , `l4_mem_reserved_mmio` = 2 }
Memory sub types for l4_mem_type_info descriptors.
- enum `l4_mem_archspecific_sub_type_common_t` { `l4_mem_archspecific_acpi_tables` = 3 , `l4_mem_archspecific_acpi_nvs`
= 4 }
Memory sub types for l4_mem_type_archspecific descriptors.

Functions

- `l4_kernel_info_mem_desc_t * l4_kernel_info_get_mem_descs (l4_kernel_info_t *kip) L4_NOTHROW`
Get pointer to memory descriptors from KIP.
- `unsigned l4_kernel_info_get_num_mem_descs (l4_kernel_info_t *kip) L4_NOTHROW`
Get number of memory descriptors in KIP.
- `void l4_kernel_info_set_mem_desc (l4_kernel_info_mem_desc_t *md, l4_addr_t start, l4_addr_t end, unsigned type, unsigned virt, unsigned sub_type) L4_NOTHROW`
Populate a memory descriptor.
- `l4_umword_t l4_kernel_info_get_mem_desc_start (l4_kernel_info_mem_desc_t *md) L4_NOTHROW`
Get start address of the region described by the memory descriptor.
- `l4_umword_t l4_kernel_info_get_mem_desc_end (l4_kernel_info_mem_desc_t *md) L4_NOTHROW`
Get end address of the region described by the memory descriptor.
- `l4_umword_t l4_kernel_info_get_mem_desc_type (l4_kernel_info_mem_desc_t *md) L4_NOTHROW`
Get type of the memory region.
- `l4_umword_t l4_kernel_info_get_mem_desc_subtype (l4_kernel_info_mem_desc_t *md) L4_NOTHROW`
Get sub-type of memory region.
- `l4_umword_t l4_kernel_info_get_mem_desc_is_virtual (l4_kernel_info_mem_desc_t *md) L4_NOTHROW`
Get virtual flag of the memory descriptor.

13.1.11.5.1 Detailed Description

C Interface for KIP memory descriptors.

Include File

```
#include <l4/sys/memdesc.h>
```

This module contains the C functions to access the memory descriptor in the kernel interface page (KIP).

13.1.11.5.2 Typedef Documentation

13.1.11.5.2.1 l4_kernel_info_mem_desc_t

```
typedef struct l4_kernel_info_mem_desc_t l4_kernel_info_mem_desc_t
```

Memory descriptor data structure.

Note

This data type is opaque, and must be accessed by the accessor functions defined in this module.

13.1.11.5.3 Enumeration Type Documentation

13.1.11.5.3.1 I4_mem_archspecific_sub_type_common_t

```
enum I4_mem_archspecific_sub_type_common_t
```

Memory sub types for I4_mem_type_archspecific descriptors.

Enumerator

| | |
|---------------------------------|----------------------------------|
| I4_mem_archspecific_acpi_tables | Firmware ACPI tables. |
| I4_mem_archspecific_acpi_nvs | Firmware reserved address space. |

Definition at line 63 of file [memdesc.h](#).

13.1.11.5.3.2 I4_mem_info_sub_type_t

```
enum I4_mem_info_sub_type_t
```

Memory sub types for I4_mem_type_info descriptors.

Enumerator

| | |
|------------------------|--|
| I4_mem_info_acpi_rsdp | Physical address of the ACPI root pointer. |
| I4_mem_reserved_kernel | Kernel image. |
| I4_mem_reserved_heap | Kernel heap. |
| I4_mem_reserved_mmio | MMIO range reserved by kernel. |

Definition at line 50 of file [memdesc.h](#).

13.1.11.5.3.3 I4_mem_type_t

```
enum I4_mem_type_t
```

Type of a memory descriptor.

Enumerator

| | |
|--------------------------|--|
| I4_mem_type_undefined | Undefined, unused descriptor. |
| I4_mem_type_conventional | Conventional memory. |
| I4_mem_type_reserved | Reserved memory for kernel etc. |
| I4_mem_type_dedicated | Dedicated memory (some device memory). |
| I4_mem_type_shared | Shared memory (not implemented). |
| I4_mem_type_info | Info from the boot loader. |
| I4_mem_type_bootloader | Memory owned by the boot loader. |

| | |
|---------------------------------------|---|
| <code>l4_mem_type_archspecific</code> | Architecture specific memory (e.g., ACPI memory). |
|---------------------------------------|---|

Definition at line 33 of file [memdesc.h](#).

13.1.11.5.4 Function Documentation

13.1.11.5.4.1 `l4_kernel_info_get_mem_desc_end()`

```
l4_umword_t l4_kernel_info_get_mem_desc_end (  
    l4_kernel_info_mem_desc_t * md) [inline]
```

Get end address of the region described by the memory descriptor.

Returns

End address.

Definition at line 219 of file [memdesc.h](#).

References [L4_NOTHROW](#).

13.1.11.5.4.2 `l4_kernel_info_get_mem_desc_is_virtual()`

```
l4_umword_t l4_kernel_info_get_mem_desc_is_virtual (  
    l4_kernel_info_mem_desc_t * md) [inline]
```

Get virtual flag of the memory descriptor.

Returns

1 if region is virtual memory, 0 if region is physical memory

Definition at line 240 of file [memdesc.h](#).

References [L4_NOTHROW](#).

13.1.11.5.4.3 `l4_kernel_info_get_mem_desc_start()`

```
l4_umword_t l4_kernel_info_get_mem_desc_start (  
    l4_kernel_info_mem_desc_t * md) [inline]
```

Get start address of the region described by the memory descriptor.

Returns

Start address.

Definition at line 212 of file [memdesc.h](#).

References [L4_NOTHROW](#).

13.1.11.5.4.4 l4_kernel_info_get_mem_desc_subtype()

```
l4_umword_t l4_kernel_info_get_mem_desc_subtype (  
    l4_kernel_info_mem_desc_t * md) [inline]
```

Get sub-type of memory region.

Returns

Sub-type.

The sub type is defined for architecture specific memory descriptors (see [l4_mem_type_archspecific](#)) and has architecture specific meaning.

Definition at line 233 of file [memdesc.h](#).

References [L4_NOTHROW](#).

13.1.11.5.4.5 l4_kernel_info_get_mem_desc_type()

```
l4_umword_t l4_kernel_info_get_mem_desc_type (  
    l4_kernel_info_mem_desc_t * md) [inline]
```

Get type of the memory region.

Returns

Type of the region (see [l4_mem_type_t](#)).

Definition at line 226 of file [memdesc.h](#).

References [L4_NOTHROW](#).

13.1.11.5.4.6 l4_kernel_info_get_num_mem_descs()

```
unsigned l4_kernel_info_get_num_mem_descs (  
    l4_kernel_info_t * kip) [inline]
```

Get number of memory descriptors in KIP.

Returns

Number of memory descriptors.

Definition at line 190 of file [memdesc.h](#).

References [L4_NOTHROW](#).

13.1.11.5.4.7 l4_kernel_info_set_mem_desc()

```
void l4_kernel_info_set_mem_desc (
    l4_kernel_info_mem_desc_t * md,
    l4_addr_t start,
    l4_addr_t end,
    unsigned type,
    unsigned virt,
    unsigned sub_type) [inline]
```

Populate a memory descriptor.

Parameters

| | |
|-----------------|---|
| <i>md</i> | Pointer to memory descriptor |
| <i>start</i> | Start of region |
| <i>end</i> | End of region |
| <i>type</i> | Type of region |
| <i>virt</i> | 1 if virtual region, 0 if physical region |
| <i>sub_type</i> | Sub type. |

Definition at line 197 of file [memdesc.h](#).

References [L4_NOTHROW](#).

13.1.12 Capabilities

C interface for capabilities.

Collaboration diagram for Capabilities:



Macros

- **#define L4_CAP_SHIFT**
Capability index shift.
- **#define L4_CAP_OFFSET**
Offset of two consecutive capability selectors.
- **#define L4_CAP_MASK**
Mask to get only the relevant bits of an [l4_cap_idx_t](#).

Typedefs

- typedef unsigned long [l4_cap_idx_t](#)
Capability selector type.

Enumerations

- enum [l4_default_caps_t](#) {
[L4_BASE_TASK_CAP](#) , [L4_BASE_FACTORY_CAP](#) , [L4_BASE_THREAD_CAP](#) , [L4_BASE_PAGER_CAP](#) ,
[L4_BASE_LOG_CAP](#) , [L4_BASE_ICU_CAP](#) , [L4_BASE_SCHEDULER_CAP](#) , [L4_BASE_IOMMU_CAP](#) ,
[L4_BASE_DEBUGGER_CAP](#) , [L4_BASE_ARM_SMCCC_CAP](#) , [L4_BASE_CAPS_LAST_P1](#) , [L4_BASE_CAPS_LAST](#)
= [L4_BASE_CAPS_LAST_P1](#) - 1 }
Default capabilities setup for the initial tasks.

Functions

- unsigned [l4_is_invalid_cap](#) ([l4_cap_idx_t](#) c) [L4_NOTHROW](#)
Test if a capability selector is the invalid capability.
- unsigned [l4_is_valid_cap](#) ([l4_cap_idx_t](#) c) [L4_NOTHROW](#)
Test if a capability selector is a valid selector.
- unsigned [l4_capability_equal](#) ([l4_cap_idx_t](#) c1, [l4_cap_idx_t](#) c2) [L4_NOTHROW](#)
Test if the capability indices of two capability selectors are equal.

13.1.12.1 Detailed Description

C interface for capabilities.

Add

```
#include <l4/sys/types.h>
#include <l4/sys/consts.h>
```

to your code to use the functions and definitions explained here.

13.1.12.2 Typedef Documentation

13.1.12.2.1 [l4_cap_idx_t](#)

```
typedef unsigned long l4\_cap\_idx\_t
```

Capability selector type.

A capability selector is either a (shifted) capability index, a (shifted) reply capability index or the invalid capability selector [L4_INVALID_CAP](#).

Usage of the invalid capability selector is defined only for invoking IPC (see [Object Invocation](#)): When IPC is invoked on [L4_INVALID_CAP](#), then it is resolved to a capability for the current thread with full permissions.

Reply capability selectors are denoted by the presence of the [L4_REPLY_CAP_BIT](#). A reply capability selector is an index into the reply capability space of the current task. The [L4_INVALID_REPLY_CAP](#) selector is resolved to the implicit reply capability of the current thread.

Otherwise, the API assumes that each argument of type [l4_cap_idx_t](#) is a capability index, i.e., $idx \ll \text{L4_CAP_SHIFT}$ for arbitrary idx . The behavior for other arguments is then undefined.

Examples

[examples/libs/shmc/prodcons.c](#), [examples/sys/aliens/main.c](#), [examples/sys/isr/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 357 of file [types.h](#).

13.1.12.3 Enumeration Type Documentation

13.1.12.3.1 l4_default_caps_t

```
enum l4_default_caps_t
```

Default capabilities setup for the initial tasks.

These capability selectors are setup per default by the micro kernel for the two initial tasks, the Root-Pager (Sigma0) and the Root-Task (Moe).

Attention

These constants do not have any particular meaning for applications started by Moe, see [Initial Environment](#) for this kind of information.

See also

[Initial Environment](#) for information useful for normal user applications.

Enumerator

| | |
|-----------------------|--|
| L4_BASE_TASK_CAP | Capability selector for the current task. |
| L4_BASE_FACTORY_CAP | Capability selector for the factory. |
| L4_BASE_THREAD_CAP | Capability selector for the first thread. |
| L4_BASE_PAGER_CAP | Capability selector for the pager gate. For Sigma0, the pager is not present since it never raises page faults. For Moe, the pager is set to Sigma0. |
| L4_BASE_LOG_CAP | Capability selector for the log object. Present if the corresponding feature is turned on in the microkernel configuration. |
| L4_BASE_ICU_CAP | Capability selector for the base icu object. |
| L4_BASE_SCHEDULER_CAP | Capability selector for the scheduler cap. |
| L4_BASE_IOMMU_CAP | Capability selector for the IO-MMU cap. Present if the microkernel detected an IO-MMU. |
| L4_BASE_DEBUGGER_CAP | Capability selector for the debugger cap. Present if the corresponding feature is turned on in the microkernel configuration. |
| L4_BASE_ARM_SMCCC_CAP | Capability selector for the ARM SMCCC cap. Present if the microkernel detected an ARM SMC capable trusted execution environment. |
| L4_BASE_CAPS_LAST | Last capability index used for base capabilities. |

Definition at line 343 of file [consts.h](#).

13.1.12.4 Function Documentation

13.1.12.4.1 l4_capability_equal()

```
unsigned l4_capability_equal (
    l4_cap_idx_t c1,
    l4_cap_idx_t c2) [inline]
```

Test if the capability indices of two capability selectors are equal.

Parameters

| | |
|-----------|----------------------|
| <i>c1</i> | Capability selector. |
| <i>c2</i> | Capability selector. |

Return values

| | |
|---|--|
| 0 | The index parts of the capability selectors differ. |
| 1 | The index parts of the capability selectors are equal. |

Precondition

Both capability selectors must be valid (cf. [l4_is_valid_cap\(\)](#)) otherwise the return value is undefined.

Definition at line 418 of file [types.h](#).

References [L4_CAP_SHIFT](#), and [L4_NOTHROW](#).

13.1.12.4.2 l4_is_invalid_cap()

```
unsigned l4_is_invalid_cap (
    l4_cap_idx_t c) [inline]
```

Test if a capability selector is the invalid capability.

Parameters

| | |
|----------|---------------------|
| <i>c</i> | Capability selector |
|----------|---------------------|

Return values

| | |
|----|--|
| 0 | The capability selector is not the invalid capability. |
| >0 | The capability selector is the invalid capability. |

Examples

[examples/libs/l4re/c/ma+rm.c](#), [examples/sys/aliens/main.c](#), [examples/sys/isr/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 410 of file [types.h](#).

References [L4_NOTHROW](#).

13.1.12.4.3 l4_is_valid_cap()

```
unsigned l4_is_valid_cap (
    l4_cap_idx_t c) [inline]
```

Test if a capability selector is a valid selector.

Parameters

| | |
|----------|---------------------|
| <i>c</i> | Capability selector |
|----------|---------------------|

Return values

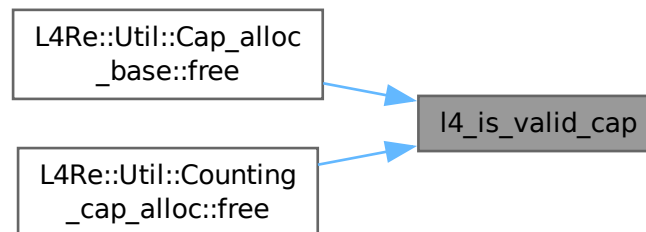
| | |
|----|---------------------------------------|
| 0 | The capability selector is not valid. |
| >0 | The capability selector is valid. |

Definition at line 414 of file [types.h](#).

References [L4_NOTHROW](#).

Referenced by [L4Re::Util::Cap_alloc_base::free\(\)](#), and [L4Re::Util::Counting_cap_alloc< COUNTERTYPE, Dbg >::free\(\)](#).

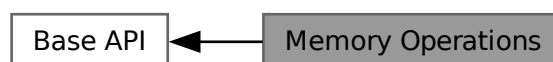
Here is the caller graph for this function:



13.1.13 Memory Operations

Operations for memory access.

Collaboration diagram for Memory Operations:



Enumerations

- enum [L4_mem_op_widths](#) { [L4_MEM_WIDTH_1BYTE](#) = 0 , [L4_MEM_WIDTH_2BYTE](#) = 1 , [L4_MEM_WIDTH_4BYTE](#) = 2 }

Memory access width definitions.

Functions

- unsigned long [l4_mem_read](#) (unsigned long virtaddress, unsigned width)
Read user task memory from kernel privilege level.
- void [l4_mem_write](#) (unsigned long virtaddress, unsigned width, unsigned long value)
Write user task memory from kernel privilege level.

13.1.13.1 Detailed Description

Operations for memory access.

This module provides functionality to access user task memory from the kernel. This is needed for some devices that are only accessible from privileged processor mode. Only use this when absolutely required. This functionality is only available on the ARM architecture.

```
#include <l4/sys/mem_op.h>
```

13.1.13.2 Enumeration Type Documentation

13.1.13.2.1 L4_mem_op_widths

```
enum L4\_mem\_op\_widths
```

Memory access width definitions.

Enumerator

| | |
|------------------------------------|-----------------------------------|
| L4_MEM_WIDTH_1BYTE | Access one byte (8-bit width). |
| L4_MEM_WIDTH_2BYTE | Access two bytes (16-bit width). |
| L4_MEM_WIDTH_4BYTE | Access four bytes (32-bit width). |

Definition at line [40](#) of file [mem_op.h](#).

13.1.13.3 Function Documentation

13.1.13.3.1 l4_mem_read()

```
unsigned long l4_mem_read (  
    unsigned long virtaddress,  
    unsigned width)    [inline]
```

Read user task memory from kernel privilege level.

Parameters

| | |
|--------------------|--------------------------------------|
| <i>virtaddress</i> | Virtual address in the calling task. |
| <i>width</i> | Width of access in bytes in log2, |

See also

[L4_mem_op_widths](#)

Returns

Read value.

Upon a given invalid address or invalid width value the function does nothing.

Definition at line 130 of file [mem_op.h](#).

References [l4_mem_arm_op_call\(\)](#).

Here is the call graph for this function:



13.1.13.3.2 l4_mem_write()

```
void l4_mem_write (  
    unsigned long virtaddress,  
    unsigned width,  
    unsigned long value)    [inline]
```

Write user task memory from kernel privilege level.

Parameters

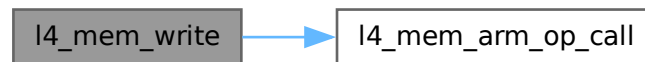
| | |
|--------------------|---|
| <i>virtaddress</i> | Virtual address in the calling task. |
| <i>width</i> | Width of access in bytes in log2 (i.e. allowed values: 0, 1, 2) |
| <i>value</i> | Value to write. |

Upon a given invalid address or invalid width value the function does nothing.

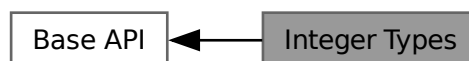
Definition at line 136 of file [mem_op.h](#).

References [l4_mem_arm_op_call\(\)](#).

Here is the call graph for this function:

**13.1.14 Integer Types**

Collaboration diagram for Integer Types:

**Files**

- file [l4int.h](#)
Fixed sized integer types, generic version.
- file [l4int.h](#)
Fixed sized integer types, arm version.
- file [l4int.h](#)
Fixed sized integer types, arm version.
- file [l4int.h](#)
Fixed sized integer types, AMD64 version.
- file [l4int.h](#)
Fixed sized integer types, x86 version.
- file [l4int.h](#)
Fixed sized integer types, RISC-V version.

Macros

- `#define L4_MWORD_BITS 32`
Size of machine words in bits.
- `#define L4_MWORD_BITS 64`
Size of machine words in bits.
- `#define L4_MWORD_BITS 64`
Size of machine words in bits.
- `#define L4_MWORD_BITS 32`
Size of machine words in bits.
- `#define L4_MWORD_BITS __riscv_xlen`
Size of machine words in bits.

Typedefs

- `typedef signed char l4_int8_t`
Signed 8bit value.
- `typedef unsigned char l4_uint8_t`
Unsigned 8bit value.
- `typedef signed short int l4_int16_t`
Signed 16bit value.
- `typedef unsigned short int l4_uint16_t`
Unsigned 16bit value.
- `typedef signed int l4_int32_t`
Signed 32bit value.
- `typedef unsigned int l4_uint32_t`
Unsigned 32bit value.
- `typedef signed long long l4_int64_t`
Signed 64bit value.
- `typedef unsigned long long l4_uint64_t`
Unsigned 64bit value.
- `typedef unsigned long l4_addr_t`
Address type.
- `typedef signed long l4_mword_t`
Signed machine word.
- `typedef unsigned long l4_umword_t`
Unsigned machine word.
- `typedef l4_uint64_t l4_cpu_time_t`
CPU clock type.
- `typedef l4_uint64_t l4_kernel_clock_t`
Kernel clock type.
- `typedef unsigned int l4_size_t`
Unsigned size type.
- `typedef signed int l4_ssize_t`
Signed size type.
- `typedef unsigned long l4_size_t`
Unsigned size type.
- `typedef signed long l4_ssize_t`
Signed size type.
- `typedef unsigned long l4_size_t`

- *Unsigned size type.*
- typedef signed long **l4_ssize_t**
- *Signed size type.*
- typedef unsigned int **l4_size_t**
- *Unsigned size type.*
- typedef signed int **l4_ssize_t**
- *Signed size type.*
- typedef unsigned int **l4_size_t**
- *Unsigned size type.*
- typedef signed int **l4_ssize_t**
- *Signed size type.*

13.1.14.1 Detailed Description

Include File

```
#include <l4/sys/l4int.h>
```

13.2 EDID parsing functionality

Enumerations

- enum **Libedid_consts** { **Libedid_block_size** = 128 }
- *EDID constants.*

Functions

- int **libedid_check_header** (const unsigned char *edid)
- *Check for valid EDID header.*
- int **libedid_checksum** (const unsigned char *edid)
- *Calculates the EDID checksum.*
- unsigned **libedid_version** (const unsigned char *edid)
- *Returns the EDID version number.*
- unsigned **libedid_revision** (const unsigned char *edid)
- *Returns the EDID revision number.*
- void **libedid_pnp_id** (const unsigned char *edid, unsigned char *id)
- *Extracts the display's PnP ID.*
- void **libedid_preferred_resolution** (const unsigned char *edid, unsigned *w, unsigned *h)
- *Extract the display's preferred mode.*
- unsigned **libedid_num_ext_blocks** (const unsigned char *edid)
- *Get the number of EDID extension blocks.*
- unsigned **libedid_dump_standard_timings** (const unsigned char *edid)
- *Dump the standard timings to stdout.*
- void **libedid_dump** (const unsigned char *edid)
- *Dump raw EDID data to stdout.*

13.2.1 Detailed Description

13.2.2 Enumeration Type Documentation

13.2.2.1 Libedid_consts

enum [Libedid_consts](#)

EDID constants.

Enumerator

| | |
|---------------------------------|----------------------------------|
| <code>Libedid_block_size</code> | Size of one EDID block in bytes. |
|---------------------------------|----------------------------------|

Definition at line 23 of file [edid.h](#).

13.2.3 Function Documentation

13.2.3.1 libedid_check_header()

```
int libedid_check_header (  
    const unsigned char * edid)
```

Check for valid EDID header.

Parameters

| | |
|-------------|---------------------------------|
| <i>edid</i> | Pointer to a 128byte EDID block |
|-------------|---------------------------------|

Returns

0 if the header is correct, -EINVAL otherwise

13.2.3.2 libedid_checksum()

```
int libedid_checksum (  
    const unsigned char * edid)
```

Calculates the EDID checksum.

Parameters

| | |
|-------------|---------------------------------|
| <i>edid</i> | Pointer to a 128byte EDID block |
|-------------|---------------------------------|

Returns

0 if checksum is correct, -EINVAL otherwise

13.2.3.3 libedid_dump()

```
void libedid_dump (
    const unsigned char * edid)
```

Dump raw EDID data to stdout.

Parameters

| | |
|-------------|---------------------------------|
| <i>edid</i> | Pointer to a 128byte EDID block |
|-------------|---------------------------------|

13.2.3.4 libedid_dump_standard_timings()

```
unsigned libedid_dump_standard_timings (
    const unsigned char * edid)
```

Dump the standard timings to stdout.

Parameters

| | |
|-------------|---------------------------------|
| <i>edid</i> | Pointer to a 128byte EDID block |
|-------------|---------------------------------|

Returns

Number of standard timings stored in EDID

13.2.3.5 libedid_num_ext_blocks()

```
unsigned libedid_num_ext_blocks (
    const unsigned char * edid)
```

Get the number of EDID extension blocks.

Parameters

| | |
|-------------|---------------------------------|
| <i>edid</i> | Pointer to a 128byte EDID block |
|-------------|---------------------------------|

Returns

Number of EDID extension blocks

13.2.3.6 libedid_pnp_id()

```
void libedid_pnp_id (
    const unsigned char * edid,
    unsigned char * id)
```

Extracts the display's PnP ID.

Parameters

| | | |
|-----|-------------|---|
| | <i>edid</i> | Pointer to a 128byte EDID block |
| out | <i>id</i> | Return the PnP id. Must point to 4 bytes. |

13.2.3.7 libedid_preferred_resolution()

```
void libedid_preferred_resolution (
    const unsigned char * edid,
    unsigned * w,
    unsigned * h)
```

Extract the display's preferred mode.

Parameters

| | | |
|-----|-------------|---|
| | <i>edid</i> | Pointer to a 128byte EDID block |
| out | <i>w</i> | X resolution of preferred video mode in pixels. |
| out | <i>h</i> | Y resolution of preferred video mode in pixels. |

13.2.3.8 libedid_revision()

```
unsigned libedid_revision (
    const unsigned char * edid)
```

Returns the EDID revision number.

Parameters

| | |
|-------------|-----------------------------|
| <i>edid</i> | Pointer to a 128 EDID block |
|-------------|-----------------------------|

Returns

Revision number

13.2.3.9 libedid_version()

```
unsigned libedid_version (
    const unsigned char * edid)
```

Returns the EDID version number.

Parameters

| | |
|-------------|---------------------------------|
| <i>edid</i> | Pointer to a 128byte EDID block |
|-------------|---------------------------------|

Returns

Version number

13.3 IO interface

Typedefs

- typedef [l4vbus_resource_t](#) [l4io_resource_t](#)
Resource descriptor.
- typedef [l4vbus_device_t](#) [l4io_device_t](#)
Device descriptor.

Enumerations

- enum [l4io_iomem_flags_t](#) {
[L4IO_MEM_NONCACHED](#) = 0 , [L4IO_MEM_CACHED](#) = 1 , [L4IO_MEM_USE_MTRR](#) = 2 , [L4IO_MEM_ATTR_MASK](#) = 0xf ,
[L4IO_MEM_WRITE_COMBINED](#) = [L4IO_MEM_USE_MTRR](#) | [L4IO_MEM_CACHED](#) , [L4IO_MEM_USE_RESERVED_AREA](#) = 0x40 << 8 , [L4IO_MEM_EAGER_MAP](#) = 0x80 << 8 }
Flags for IO memory.
- enum [l4io_device_types_t](#) {
[L4IO_DEVICE_INVALID](#) = 0 , [L4IO_DEVICE_PCI](#) , [L4IO_DEVICE_USB](#) , [L4IO_DEVICE_OTHER](#) ,
[L4IO_DEVICE_ANY](#) = ~0 }
Device types.
- enum [l4io_resource_types_t](#) {
[L4IO_RESOURCE_INVALID](#) = [L4VBUS_RESOURCE_INVALID](#) , [L4IO_RESOURCE_IRQ](#) = [L4VBUS_RESOURCE_IRQ](#) , [L4IO_RESOURCE_MEM](#) = [L4VBUS_RESOURCE_MEM](#) , [L4IO_RESOURCE_PORT](#) = [L4VBUS_RESOURCE_PORT](#) ,
[L4IO_RESOURCE_ANY](#) = ~0 }
Resource types.

Functions

- long [l4io_request_iomem](#) ([l4_addr_t](#) phys, unsigned long size, int flags, [l4_addr_t](#) *virt)
Request an IO memory region.
- long [l4io_request_iomem_region](#) ([l4_addr_t](#) phys, [l4_addr_t](#) virt, unsigned long size, int flags)
Request an IO memory region and map it to a specified region.
- long [l4io_release_iomem](#) ([l4_addr_t](#) virt, unsigned long size)
Release an IO memory region.
- long [l4io_request_ioport](#) (unsigned portnum, unsigned len)
Request an IO port region.
- long [l4io_release_ioport](#) (unsigned portnum, unsigned len)
Release an IO port region.
- int [l4io_lookup_device](#) (const char *devname, [l4io_device_handle_t](#) *dev_handle, [l4io_device_t](#) *dev, [l4io_resource_handle_t](#) *res_handle)
Find a device by name.
- int [l4io_lookup_resource](#) ([l4io_device_handle_t](#) devhandle, enum [l4io_resource_types_t](#) type, [l4io_resource_handle_t](#) *reshandle, [l4io_resource_t](#) *res)
Request a specific resource from a device description.
- [l4_addr_t](#) [l4io_request_resource_iomem](#) ([l4io_device_handle_t](#) devhandle, [l4io_resource_handle_t](#) *reshandle)
Request IO memory.
- int [l4io_has_resource](#) (enum [l4io_resource_types_t](#) type, [l4vbus_paddr_t](#) start, [l4vbus_paddr_t](#) end)
Check if a resource is available.

13.3.1 Detailed Description

13.3.2 Typedef Documentation

13.3.2.1 [l4io_resource_t](#)

```
typedef l4vbus\_resource\_t l4io\_resource\_t
```

Resource descriptor.

For IRQ types, the end field is not used, i.e. only a single interrupt can be described with a [l4io_resource_t](#)

Definition at line 67 of file [types.h](#).

13.3.3 Enumeration Type Documentation

13.3.3.1 [l4io_device_types_t](#)

```
enum l4io\_device\_types\_t
```

Device types.

Enumerator

| | |
|---------------------|--------------------------------------|
| L4IO_DEVICE_INVALID | Invalid type. |
| L4IO_DEVICE_PCI | PCI device. |
| L4IO_DEVICE_USB | USB device. |
| L4IO_DEVICE_OTHER | Any other device without unique IDs. |
| L4IO_DEVICE_ANY | any type |

Definition at line 36 of file [types.h](#).

13.3.3.2 l4io_iomem_flags_t

enum [l4io_iomem_flags_t](#)

Flags for IO memory.

Enumerator

| | |
|----------------------------|---|
| L4IO_MEM_NONCACHED | Non-cache memory. |
| L4IO_MEM_CACHED | Cache memory. |
| L4IO_MEM_USE_MTRR | Use MTRR. |
| L4IO_MEM_USE_RESERVED_AREA | Use reserved area for mapping I/O memory. Flag only valid for l4io_request_iomem_region() |
| L4IO_MEM_EAGER_MAP | Eagerly map the I/O memory. Passthrough to the l4re-rm. |

Definition at line 14 of file [types.h](#).

13.3.3.3 l4io_resource_types_t

enum [l4io_resource_types_t](#)

Resource types.

Enumerator

| | |
|-----------------------|-------------------------------|
| L4IO_RESOURCE_INVALID | Invalid type. |
| L4IO_RESOURCE_IRQ | Interrupt resource. |
| L4IO_RESOURCE_MEM | I/O memory resource. |
| L4IO_RESOURCE_PORT | I/O port resource (x86 only). |
| L4IO_RESOURCE_ANY | any type |

Definition at line 48 of file [types.h](#).

13.3.4 Function Documentation

13.3.4.1 l4io_has_resource()

```
int l4io_has_resource (
    enum l4io_resource_types_t type,
    l4vbus_paddr_t start,
    l4vbus_paddr_t end)
```

Check if a resource is available.

Parameters

| | |
|--------------|------------------|
| <i>type</i> | Type of resource |
| <i>start</i> | Minimal value. |
| <i>end</i> | Maximum value. |

References [L4_INLINE](#).

13.3.4.2 l4io_lookup_device()

```
int l4io_lookup_device (
    const char * devname,
    l4io_device_handle_t * dev_handle,
    l4io_device_t * dev,
    l4io_resource_handle_t * res_handle)
```

Find a device by name.

Parameters

| | | |
|-----|-------------------|--|
| | <i>devname</i> | Name of device. |
| out | <i>dev_handle</i> | Device handle for found device, can be NULL. |
| out | <i>dev</i> | Device information, filled by the function, can be NULL. |
| out | <i>res_handle</i> | Resource handle, can be NULL. |

Returns

0 on success, error code otherwise

References [L4_CV](#), and [L4_EXPORT](#).

13.3.4.3 l4io_lookup_resource()

```
int l4io_lookup_resource (
    l4io_device_handle_t devhandle,
    enum l4io_resource_types_t type,
    l4io_resource_handle_t * reshandle,
    l4io_resource_t * res)
```

Request a specific resource from a device description.

Parameters

| | | |
|---------|------------------|---|
| | <i>devhandle</i> | Device handle. |
| | <i>type</i> | Type of resource to request (see l4io_resource_types_t). |
| in, out | <i>reshandle</i> | Resource handle, start with handle returned by device functions. The next resource handle is returned here. |
| out | <i>res</i> | Device descriptor. |

Returns

0 on success, error code otherwise, esp. -L4_ENOENT if no more resources found

References [L4_CV](#), and [L4_EXPORT](#).

13.3.4.4 l4io_release_iomem()

```
long l4io_release_iomem (
    l4_addr_t virt,
    unsigned long size)
```

Release an IO memory region.

Parameters

| | |
|-------------|---|
| <i>virt</i> | Virtual address of region to free, see l4io_request_iomem |
| <i>size</i> | Size of the region to release. |

Returns

0 on success, <0 on error

References [L4_CV](#), and [L4_EXPORT](#).

13.3.4.5 `l4io_release_ioport()`

```
long l4io_release_ioport (
    unsigned portnum,
    unsigned len)
```

Release an IO port region.

Parameters

| | |
|----------------|--------------------------------|
| <i>portnum</i> | Start of port range to release |
| <i>len</i> | Length of range to request |

Returns

0 on success, <0 on error

Note

X86 architecture only

References [L4_CV](#), [L4_EXPORT](#), and [L4_INLINE](#).

13.3.4.6 `l4io_request_iomem()`

```
long l4io_request_iomem (
    l4_addr_t phys,
    unsigned long size,
    int flags,
    l4_addr_t * virt)
```

Request an IO memory region.

Parameters

| | | |
|----------------|--------------|--|
| | <i>phys</i> | Physical address of the I/O memory region |
| | <i>size</i> | Size of the region in Bytes, granularity pages. |
| | <i>flags</i> | See l4io_iomem_flags_t |
| <i>in, out</i> | <i>virt</i> | Virtual address where the IO memory region should be mapped to. If the caller passes '0' a region in the caller's address space is searched and the virtual address is returned. |

Return values

| | |
|------------|---|
| 0 | Success. |
| -L4_ENOENT | No area in the caller's address space could be found to map the IO memory region. |
| -L4_EPERM | Operation not allowed. |
| -L4_EINVAL | Invalid value. |

| | |
|--------------------------------|--|
| <code>-L4_EADDRNOTAVAIL</code> | The requested virtual address is not available. |
| <code>-L4_ENOMEM</code> | The requested IO memory region could not be allocated. |
| <code><0</code> | IPC errors. |

Note

This function uses [L4Re](#) functionality to reserve a part of the virtual address space of the caller.

References [L4_CV](#), and [L4_EXPORT](#).

13.3.4.7 l4io_request_iomem_region()

```
long l4io_request_iomem_region (
    l4_addr_t phys,
    l4_addr_t virt,
    unsigned long size,
    int flags)
```

Request an IO memory region and map it to a specified region.

Parameters

| | |
|--------------|---|
| <i>phys</i> | Physical address of the I/O memory region |
| <i>virt</i> | Virtual address. |
| <i>size</i> | Size of the region in Bytes, granularity pages. |
| <i>flags</i> | See l4io_iomem_flags_t |

Return values

| | |
|--------------------------------|--|
| <code>0</code> | Success. |
| <code>-L4_ENOENT</code> | No area could be found to map the IO memory region. |
| <code>-L4_EPERM</code> | Operation not allowed. |
| <code>-L4_EINVAL</code> | Invalid value. |
| <code>-L4_EADDRNOTAVAIL</code> | The requested virtual address is not available. |
| <code>-L4_ENOMEM</code> | The requested IO memory region could not be allocated. |
| <code><0</code> | IPC errors. |

Note

This function uses [L4Re](#) functionality to reserve a part of the virtual address space of the caller.

References [L4_CV](#), and [L4_EXPORT](#).

13.3.4.8 l4io_request_ioport()

```
long l4io_request_ioport (
    unsigned portnum,
    unsigned len)
```

Request an IO port region.

Parameters

| | |
|----------------|--------------------------------|
| <i>portnum</i> | Start of port range to request |
| <i>len</i> | Length of range to request |

Returns

0 on success, <0 on error

Note

X86 architecture only

References [L4_CV](#), and [L4_EXPORT](#).

13.3.4.9 l4io_request_resource_iomem()

```
l4_addr_t l4io_request_resource_iomem (
    l4io_device_handle_t devhandle,
    l4io_resource_handle_t * reshandle)
```

Request IO memory.

Parameters

| | | |
|----------------|------------------|--|
| | <i>devhandle</i> | Device handle. |
| <i>in, out</i> | <i>reshandle</i> | Resource handle from which IO memory should be requested. Upon successful completion 'reshandle' points to the device's next resource. |

Return values

| | |
|----|---|
| 0 | An error occurred. The value of 'reshandle' is undefined. |
| >0 | The virtual address of the IO memory mapping. |

References [L4_CV](#), and [L4_EXPORT](#).

13.4 IPC Helpers

Functions

- void [L4::throw_ipc_exception](#) ([L4::Cap](#)< void > const &o, [l4_msgtag_t](#) const &err, [l4_utcb_t](#) *utcb)
Throw an [L4](#) IPC error as exception.
- void [L4::throw_ipc_exception](#) (void const *o, [l4_msgtag_t](#) const &err, [l4_utcb_t](#) *utcb)
Throw an [L4](#) IPC error as exception.

13.4.1 Detailed Description

13.4.2 Function Documentation

13.4.2.1 throw_ipc_exception() [1/2]

```
void L4::throw_ipc_exception (
    L4::Cap< void > const & o,
    l4\_msgtag\_t const & err,
    l4\_utcb\_t * utcb) [inline]
```

Throw an [L4](#) IPC error as exception.

Parameters

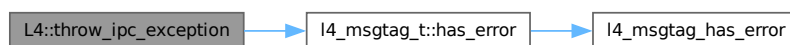
| | |
|-------------|--|
| <i>o</i> | The client side object, for which the IPC was invoked. |
| <i>err</i> | The IPC result code (error code). |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |

Definition at line [34](#) of file [ipc_helper](#).

References [l4_msgtag_t::has_error\(\)](#).

Referenced by [throw_ipc_exception\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.4.2.2 throw_ipc_exception() [2/2]

```
void L4::throw_ipc_exception (
    void const * o,
    l4_msgtag_t const & err,
    l4_utcb_t * utcb) [inline]
```

Throw an [L4](#) IPC error as exception.

Parameters

| | |
|-------------|--|
| <i>o</i> | The client side object, for which the IPC was invoked. |
| <i>err</i> | The IPC result code (error code). |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |

Definition at line 50 of file [ipc_helper](#).

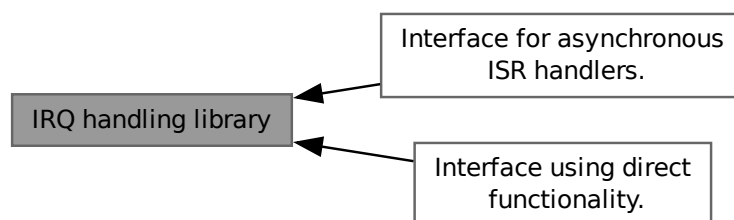
References [throw_ipc_exception\(\)](#).

Here is the call graph for this function:



13.5 IRQ handling library

Collaboration diagram for IRQ handling library:



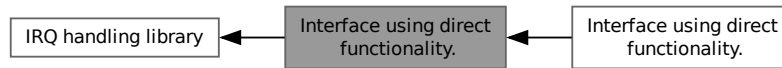
Topics

- [Interface using direct functionality.](#) 468
- [Interface for asynchronous ISR handlers.](#) 475
This interface has just two (main) functions.

13.5.1 Detailed Description

13.5.2 Interface using direct functionality.

Collaboration diagram for Interface using direct functionality.:



Topics

- [Interface using direct functionality.](#) 473

Functions

- `l4irq_t * l4irq_attach (int irqnum)`
Attach/connect to IRQ.
- `l4irq_t * l4irq_attach_ft (int irqnum, unsigned mode)`
Attach/connect to IRQ using given type.
- `l4irq_t * l4irq_attach_thread (int irqnum, l4_cap_idx_t to_thread)`
Attach/connect to IRQ.
- `l4irq_t * l4irq_attach_thread_ft (int irqnum, l4_cap_idx_t to_thread, unsigned mode)`
Attach/connect to IRQ using given type.
- `long l4irq_wait (l4irq_t *irq)`
Wait for specified IRQ.
- `long l4irq_unmask_and_wait_any (l4irq_t *unmask_irq, l4irq_t **ret_irq)`
Unmask a specific IRQ and wait for any attached IRQ.
- `long l4irq_wait_any (l4irq_t **irq)`
Wait for any attached IRQ.
- `long l4irq_unmask (l4irq_t *irq)`
Unmask a specific IRQ.
- `long l4irq_detach (l4irq_t *irq)`
Detach from IRQ.

13.5.2.1 Detailed Description

13.5.2.2 Function Documentation

13.5.2.2.1 `l4irq_attach()`

```
l4irq_t * l4irq_attach (  
    int irqnum)
```

Attach/connect to IRQ.

Parameters

| | |
|---------------|-----------------------|
| <i>irqnum</i> | IRQ number to request |
|---------------|-----------------------|

Returns

Pointer to `l4irq_t` structure, 0 on error

This `l4irq_attach` has to be called in the same thread as `l4irq_wait` and caller has to be a pthread thread.

Examples

[examples/libs/libirq/loop.c](#).

References [L4_CV](#).

13.5.2.2.2 `l4irq_attach_ft()`

```
l4irq_t * l4irq_attach_ft (  
    int irqnum,  
    unsigned mode)
```

Attach/connect to IRQ using given type.

Parameters

| | |
|---------------|-----------------------|
| <i>irqnum</i> | IRQ number to request |
| <i>mode</i> | Interrupt type, |

See also

[L4_irq_mode](#)

Returns

Pointer to `l4irq_t` structure, 0 on error

This `l4irq_attach` has to be called in the same thread as `l4irq_wait` and caller has to be a pthread thread.

References [L4_CV](#).

13.5.2.2.3 l4irq_attach_thread()

```
l4irq_t * l4irq_attach_thread (
    int irqnum,
    l4_cap_idx_t to_thread)
```

Attach/connect to IRQ.

Parameters

| | |
|------------------|--------------------------------------|
| <i>irqnum</i> | IRQ number to request |
| <i>to_thread</i> | Attach IRQ to this specified thread. |

Returns

Pointer to l4irq_t structure, 0 on error

The pointer to the IRQ structure is used as a label in the IRQ object.

References [L4_CV](#).

13.5.2.2.4 l4irq_attach_thread_ft()

```
l4irq_t * l4irq_attach_thread_ft (
    int irqnum,
    l4_cap_idx_t to_thread,
    unsigned mode)
```

Attach/connect to IRQ using given type.

Parameters

| | |
|------------------|--------------------------------------|
| <i>irqnum</i> | IRQ number to request |
| <i>to_thread</i> | Attach IRQ to this specified thread. |
| <i>mode</i> | Interrupt type, |

See also

[L4_irq_mode](#)

Returns

Pointer to l4irq_t structure, 0 on error

The pointer to the IRQ structure is used as a label in the IRQ object.

References [L4_CV](#).

13.5.2.2.5 l4irq_detach()

```
long l4irq_detach (  
    l4irq_t * irq)
```

Detach from IRQ.

Parameters

| | |
|------------|--------------------|
| <i>irq</i> | IRQ data structure |
|------------|--------------------|

Returns

0 on success, != 0 on error

References [L4_CV](#).

13.5.2.2.6 l4irq_unmask()

```
long l4irq_unmask (  
    l4irq_t * irq)
```

Unmask a specific IRQ.

Parameters

| | |
|------------|--------------------|
| <i>irq</i> | IRQ data structure |
|------------|--------------------|

Returns

0 on success, != 0 on error

This function is useful if a thread wants to wait for multiple IRQs using `l4_ipc_wait`.

References [L4_CV](#).

13.5.2.2.7 l4irq_unmask_and_wait_any()

```
long l4irq_unmask_and_wait_any (  
    l4irq_t * unmask_irq,  
    l4irq_t ** ret_irq)
```

Unmask a specific IRQ and wait for any attached IRQ.

Parameters

| | | |
|--|-------------------|--------------------------------|
| | <i>unmask_irq</i> | IRQ data structure for unmask. |
|--|-------------------|--------------------------------|

| | | |
|-----|----------------|---------------------|
| out | <i>ret_irq</i> | Received interrupt. |
|-----|----------------|---------------------|

Returns

0 on success, != 0 on error

References [L4_CV](#).

13.5.2.2.8 l4irq_wait()

```
long l4irq_wait (
    l4irq_t * irq)
```

Wait for specified IRQ.

Parameters

| | |
|------------|--------------------|
| <i>irq</i> | IRQ data structure |
|------------|--------------------|

Returns

0 on success, != 0 on error

Examples

[examples/libs/libirq/loop.c](#).

References [L4_CV](#).

13.5.2.2.9 l4irq_wait_any()

```
long l4irq_wait_any (
    l4irq_t ** irq)
```

Wait for any attached IRQ.

Return values

| | |
|------------|---------------------|
| <i>irq</i> | Received interrupt. |
|------------|---------------------|

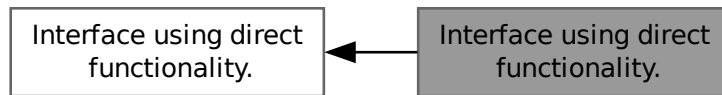
Returns

0 on success, != 0 on error

References [L4_CV](#).

13.5.2.3 Interface using direct functionality.

Collaboration diagram for Interface using direct functionality.:



Functions

- `l4irq_t * l4irq_attach_cap (l4_cap_idx_t irqcap)`
Attach/connect to IRQ.
- `l4irq_t * l4irq_attach_cap_ft (l4_cap_idx_t irqcap, unsigned mode)`
Attach/connect to IRQ using given type.
- `l4irq_t * l4irq_attach_thread_cap (l4_cap_idx_t irqcap, l4_cap_idx_t to_thread)`
Attach/connect to IRQ.
- `l4irq_t * l4irq_attach_thread_cap_ft (l4_cap_idx_t irqcap, l4_cap_idx_t to_thread, unsigned mode)`
Attach/connect to IRQ using given type.

13.5.2.3.1 Detailed Description

13.5.2.3.2 Function Documentation

13.5.2.3.2.1 l4irq_attach_cap()

```
l4irq_t * l4irq_attach_cap (
    l4_cap_idx_t irqcap)
```

Attach/connect to IRQ.

Parameters

| | |
|---------------------|----------------|
| <code>irqcap</code> | IRQ capability |
|---------------------|----------------|

Returns

Pointer to `l4irq_t` structure, 0 on error

This `l4irq_attach` has to be called in the same thread as `l4irq_wait` and caller has to be a pthread thread.

References [L4_CV](#).

13.5.2.3.2.2 l4irq_attach_cap_ft()

```
l4irq_t * l4irq_attach_cap_ft (
    l4_cap_idx_t irqcap,
    unsigned mode)
```

Attach/connect to IRQ using given type.

Parameters

| | |
|---------------|-----------------|
| <i>irqcap</i> | IRQ capability |
| <i>mode</i> | Interrupt type, |

See also

[L4_irq_mode](#)

Returns

Pointer to l4irq_t structure, 0 on error

This l4irq_attach has to be called in the same thread as l4irq_wait and caller has to be a pthread thread.

References [L4_CV](#).

13.5.2.3.2.3 l4irq_attach_thread_cap()

```
l4irq_t * l4irq_attach_thread_cap (
    l4_cap_idx_t irqcap,
    l4_cap_idx_t to_thread)
```

Attach/connect to IRQ.

Parameters

| | |
|------------------|----------------------------|
| <i>irqcap</i> | IRQ capability |
| <i>to_thread</i> | Attach IRQ to this thread. |

Returns

Pointer to l4irq_t structure, 0 on error

The pointer to the IRQ structure is used as a label in the IRQ object.

References [L4_CV](#).

13.5.2.3.2.4 l4irq_attach_thread_cap_ft()

```
l4irq_t * l4irq_attach_thread_cap_ft (
    l4_cap_idx_t irqcap,
    l4_cap_idx_t to_thread,
    unsigned mode)
```

Attach/connect to IRQ using given type.

Parameters

| | |
|------------------|----------------------------|
| <i>irqcap</i> | IRQ capability |
| <i>to_thread</i> | Attach IRQ to this thread. |
| <i>mode</i> | Interrupt type, |

See also

[L4_irq_mode](#)

Returns

Pointer to l4irq_t structure, 0 on error

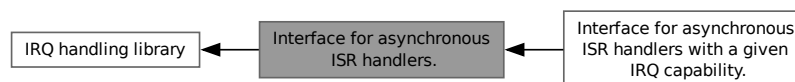
The pointer to the IRQ structure is used as a label in the IRQ object.

References [L4_CV](#).

13.5.3 Interface for asynchronous ISR handlers.

This interface has just two (main) functions.

Collaboration diagram for Interface for asynchronous ISR handlers.:



Topics

- [Interface for asynchronous ISR handlers with a given IRQ capability.](#) 477
This group is just an enhanced version to [l4irq_request\(\)](#) which takes a capability object instead of a plain number.

Functions

- `l4irq_t * l4irq_request` (int irqnum, void(*isr_handler)(void *), void *isr_data, int irq_thread_prio, unsigned mode)
Attach asynchronous ISR handler to IRQ.
- `long l4irq_release` (l4irq_t *irq)
Release asynchronous ISR handler and free resources.

13.5.3.1 Detailed Description

This interface has just two (main) functions.

`l4irq_request` to install a handler for an interrupt and `l4irq_release` to uninstall the handler again and release all resources associated with it.

13.5.3.2 Function Documentation

13.5.3.2.1 l4irq_release()

```
long l4irq_release (
    l4irq_t * irq)
```

Release asynchronous ISR handler and free resources.

Parameters

| | |
|------------|--------------------|
| <i>irq</i> | IRQ data structure |
|------------|--------------------|

Returns

0 success, != 0 failure

Examples

[examples/libs/libirq/async_isr.c](#).

References [L4_CV](#).

13.5.3.2.2 l4irq_request()

```
l4irq_t * l4irq_request (
    int irqnum,
    void(* isr_handler ) (void *),
    void * isr_data,
    int irq_thread_prio,
    unsigned mode)
```

Attach asynchronous ISR handler to IRQ.

Parameters

| | |
|------------------------|--|
| <i>irqnum</i> | IRQ number to request |
| <i>isr_handler</i> | Handler routine that is called when an interrupt triggers |
| <i>isr_data</i> | Pointer given as argument to isr_handler |
| <i>irq_thread_prio</i> | L4 thread priority of the ISR handler. Give -1 for same priority as creator. |
| <i>mode</i> | Interrupt type, |

See also

[L4_irq_mode](#)

Returns

Pointer to l4irq_t structure, 0 on error

Examples

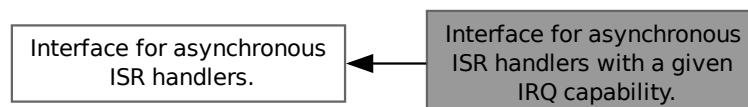
[examples/libs/libirq/async_isr.c](#).

References [L4_CV](#).

13.5.3.3 Interface for asynchronous ISR handlers with a given IRQ capability.

This group is just an enhanced version to [l4irq_request\(\)](#) which takes a capability object instead of a plain number.

Collaboration diagram for Interface for asynchronous ISR handlers with a given IRQ capability.:



Functions

- `l4irq_t * l4irq_request_cap (l4_cap_idx_t irqcap, void(*isr_handler)(void *), void *isr_data, int irq_thread_prio, unsigned mode)`

Attach asynchronous ISR handler to IRQ.

13.5.3.3.1 Detailed Description

This group is just an enhanced version to `l4irq_request()` which takes a capability object instead of a plain number.

13.5.3.3.2 Function Documentation

13.5.3.3.2.1 `l4irq_request_cap()`

```
l4irq_t * l4irq_request_cap (
    l4_cap_idx_t irqcap,
    void(* isr_handler ) (void *),
    void * isr_data,
    int irq_thread_prio,
    unsigned mode)
```

Attach asynchronous ISR handler to IRQ.

Parameters

| | |
|------------------------|--|
| <i>irqcap</i> | IRQ capability |
| <i>isr_handler</i> | Handler routine that is called when an interrupt triggers |
| <i>isr_data</i> | Pointer given as argument to <code>isr_handler</code> |
| <i>irq_thread_prio</i> | L4 thread priority of the ISR handler. Give -1 for same priority as creator. |
| <i>mode</i> | Interrupt type, |

See also

[L4_irq_mode](#)

Returns

Pointer to `l4irq_t` structure, 0 on error

13.6 L4 IPC Opcodes

List of protocol specific opcodes used for communication with [L4Re](#) and Kernel objects.

Enumerations

- enum `L4_icu_opcode` {
`L4_ICU_OP_BIND` , `L4_ICU_OP_UNBIND` , `L4_ICU_OP_INFO` , `L4_ICU_OP_MSI_INFO` ,
`L4_ICU_OP_UNMASK` , `L4_ICU_OP_MASK` , `L4_ICU_OP_SET_MODE` }
Opcodes to the ICU interface.
- enum `L4_ipc_gate_ops` { `L4_IPC_GATE_BIND_OP` = 0x10 , `L4_IPC_GATE_GET_INFO_OP` = 0x11 }
Operations on the IPC-gate.
- enum `L4_platform_ctl_ops` {
`L4_PLATFORM_CTL_SYS_SUSPEND_OP` = 0UL , `L4_PLATFORM_CTL_SYS_SHUTDOWN_OP` = 1UL ,
`L4_PLATFORM_CTL_CPU_ALLOW_SHUTDOWN_OP` = 2UL , `L4_PLATFORM_CTL_CPU_ENABLE_OP` =
3UL ,
`L4_PLATFORM_CTL_CPU_DISABLE_OP` = 4UL , `L4_PLATFORM_CTL_SET_TASK_ASID_OP` = 0x10UL }
Operations on platform-control objects.
- enum `L4_task_ops` {
`L4_TASK_MAP_OP` = 0UL , `L4_TASK_UNMAP_OP` = 1UL , `L4_TASK_CAP_INFO_OP` = 2UL ,
`L4_TASK_ADD_KU_MEM_OP` = 3UL ,
`L4_TASK_LDT_SET_X86_OP` = 0x11UL , `L4_TASK_MAP_VGICC_ARM_OP` = 0x12UL }
Operations on task objects.
- enum `L4_thread_ops` {
`L4_THREAD_CONTROL_OP` = 0UL , `L4_THREAD_EX_REGS_OP` = 1UL , `L4_THREAD_SWITCH_OP` =
2UL , `L4_THREAD_STATS_OP` = 3UL ,
`L4_THREAD_VCPU_RESUME_OP` = 4UL , `L4_THREAD_REGISTER_DELETE_IRQ_OP` = 5UL ,
`L4_THREAD_MODIFY_SENDER_OP` = 6UL , `L4_THREAD_VCPU_CONTROL_OP` = 7UL ,
`L4_THREAD_VCPU_CONTROL_EXT_OP` = `L4_THREAD_VCPU_CONTROL_OP` | 0x10000 , `L4_THREAD_REGISTER_DO`
= 8UL , `L4_THREAD_X86_GDT_OP` = 0x10UL , `L4_THREAD_ARM_TPIDRURO_OP` = 0x10UL ,
`L4_THREAD_AMD64_SET_SEGMENT_BASE_OP` = 0x12UL , `L4_THREAD_AMD64_GET_SEGMENT_INFO_OP`
= 0x13UL , `L4_THREAD_OPCODE_MASK` = 0xffff }
Operations on thread objects.
- enum `L4_vcon_ops` { `L4_VCON_WRITE_OP` = 0UL , `L4_VCON_READ_OP` = 1UL , `L4_VCON_SET_ATTR_OP`
= 2UL , `L4_VCON_GET_ATTR_OP` = 3UL }
Operations on vcon objects.

13.6.1 Detailed Description

List of protocol specific opcodes used for communication with [L4Re](#) and Kernel objects.

13.6.2 Enumeration Type Documentation

13.6.2.1 L4_icu_opcode

```
enum L4_icu_opcode
```

Opcodes to the ICU interface.

Enumerator

| | |
|--------------------|---|
| L4_ICU_OP_BIND | Bind opcode. See also l4_icu_bind() |
| L4_ICU_OP_UNBIND | Unbind opcode. See also l4_icu_unbind() |
| L4_ICU_OP_INFO | Info opcode. See also l4_icu_info() |
| L4_ICU_OP_MSI_INFO | Msi-info opcode. See also l4_icu_msi_info() |
| L4_ICU_OP_UNMASK | Unmask opcode. See also l4_icu_unmask() |
| L4_ICU_OP_MASK | Mask opcode. See also l4_icu_mask() |
| L4_ICU_OP_SET_MODE | Set-mode opcode. See also l4_icu_set_mode() |

Definition at line 97 of file [icu.h](#).

13.6.2.2 L4_ipc_gate_ops

enum [L4_ipc_gate_ops](#)

Operations on the IPC-gate.

Enumerator

| | |
|-------------------------|-----------------|
| L4_IPC_GATE_BIND_OP | Bind operation. |
| L4_IPC_GATE_GET_INFO_OP | Info operation. |

Definition at line 108 of file [ipc_gate.h](#).

13.6.2.3 L4_platform_ctl_ops

enum [L4_platform_ctl_ops](#)

Operations on platform-control objects.

See [L4_PROTO_PLATFORM_CTL](#) for the protocol type to use for messages to platform-control objects.

Enumerator

| | |
|---------------------------------------|-----------------------|
| L4_PLATFORM_CTL_SYS_SUSPEND_OP | Suspend. |
| L4_PLATFORM_CTL_SYS_SHUTDOWN_OP | shutdown/reboot |
| L4_PLATFORM_CTL_CPU_ALLOW_SHUTDOWN_OP | allow CPU shutdown |
| L4_PLATFORM_CTL_CPU_ENABLE_OP | enable an offline CPU |
| L4_PLATFORM_CTL_CPU_DISABLE_OP | disable an online CPU |
| L4_PLATFORM_CTL_SET_TASK_ASID_OP | Arm: set task ASID. |

Definition at line 159 of file [platform_control.h](#).

13.6.2.4 L4_task_ops

enum [L4_task_ops](#)

Operations on task objects.

Enumerator

| | |
|--------------------------|-----------------------------|
| L4_TASK_MAP_OP | Map. |
| L4_TASK_UNMAP_OP | Unmap. |
| L4_TASK_CAP_INFO_OP | Cap info. |
| L4_TASK_ADD_KU_MEM_OP | Add kernel-user memory. |
| L4_TASK_LDT_SET_X86_OP | x86: LDT set |
| L4_TASK_MAP_VGICC_ARM_OP | Arm: Map virtual GICC area. |

Definition at line 341 of file [task.h](#).

13.6.2.5 L4_thread_ops

enum [L4_thread_ops](#)

Operations on thread objects.

Enumerator

| | |
|----------------------|--------------------|
| L4_THREAD_CONTROL_OP | Control operation. |
|----------------------|--------------------|

| | |
|-------------------------------------|--|
| L4_THREAD_EX_REGS_OP | Exchange registers operation. |
| L4_THREAD_SWITCH_OP | Do a thread switch. |
| L4_THREAD_STATS_OP | Thread statistics. |
| L4_THREAD_VCPU_RESUME_OP | VCPU resume. |
| L4_THREAD_REGISTER_DELETE_IRQ_OP | Register an IPC-gate deletion IRQ. |
| L4_THREAD_MODIFY_SENDER_OP | Modify all senders IDs that match the given pattern. |
| L4_THREAD_VCPU_CONTROL_OP | Enable / disable VCPU feature. |
| L4_THREAD_REGISTER_DOORBELL_IRQ_OP | Register direct IRQ injection doorbell IRQ. |
| L4_THREAD_X86_GDT_OP | Gdt. |
| L4_THREAD_ARM_TPIDRURO_OP | Set TPIDRURO register. |
| L4_THREAD_AMD64_SET_SEGMENT_BASE_OP | Set segment base. |
| L4_THREAD_AMD64_GET_SEGMENT_INFO_OP | Get segment information. |
| L4_THREAD_OPCODE_MASK | Mask for opcodes. |

Definition at line 732 of file [thread.h](#).

13.6.2.6 L4_vcon_ops

enum [L4_vcon_ops](#)

Operations on vcon objects.

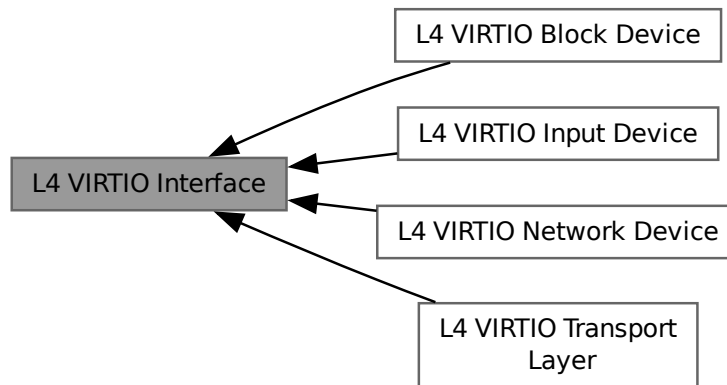
Enumerator

| | |
|---------------------|-------------------------|
| L4_VCON_WRITE_OP | Write. |
| L4_VCON_READ_OP | Read. |
| L4_VCON_SET_ATTR_OP | Get console attributes. |
| L4_VCON_GET_ATTR_OP | Set console attributes. |

Definition at line 291 of file [vcon.h](#).

13.7 L4 VIRTIO Interface

Collaboration diagram for L4 VIRTIO Interface:



Topics

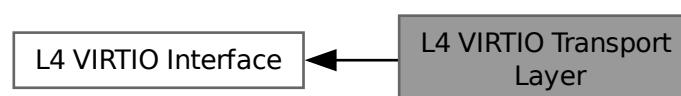
- L4 VIRTIO Transport Layer 483
[L4 specific VIRTIO Transport layer.](#)
- L4 VIRTIO Block Device 492
- L4 VIRTIO Input Device 494
- L4 VIRTIO Network Device 494

13.7.1 Detailed Description

13.7.2 L4 VIRTIO Transport Layer

[L4 specific VIRTIO Transport layer.](#)

Collaboration diagram for L4 VIRTIO Transport Layer:



Namespaces

- namespace [L4virtio](#)
L4-VIRTIO Transport C++ API.

Data Structures

- struct [l4virtio_config_hdr_t](#)
L4-VIRTIO config header, provided in shared data space.
- struct [l4virtio_config_queue_t](#)
Queue configuration entry.

Typedefs

- typedef struct l4virtio_config_hdr_t [l4virtio_config_hdr_t](#)
L4-VIRTIO config header, provided in shared data space.
- typedef struct l4virtio_config_queue_t [l4virtio_config_queue_t](#)
Queue configuration entry.

Enumerations

- enum [L4_virtio_protocol](#)
L4-VIRTIO protocol number.
- enum [L4_virtio_opcodes](#) {
[L4VIRTIO_OP_SET_STATUS](#) = 0 , [L4VIRTIO_OP_CONFIG_QUEUE](#) = 1 , [L4VIRTIO_OP_REGISTER_DS](#) = 3 , [L4VIRTIO_OP_DEVICE_CONFIG](#) = 4 ,
[L4VIRTIO_OP_GET_DEVICE_IRQ](#) = 5 }
Opcodes to setup and configure a device.
- enum [L4virtio_device_ids](#) {
[L4VIRTIO_ID_NET](#) = 1 , [L4VIRTIO_ID_BLOCK](#) = 2 , [L4VIRTIO_ID_CONSOLE](#) = 3 , [L4VIRTIO_ID_RNG](#) = 4 ,
[L4VIRTIO_ID_BALLOON](#) = 5 , [L4VIRTIO_ID_RPMMSG](#) = 7 , [L4VIRTIO_ID_SCSI](#) = 8 , [L4VIRTIO_ID_9P](#) = 9 ,
[L4VIRTIO_ID_RPROC_SERIAL](#) = 11 , [L4VIRTIO_ID_CAIF](#) = 12 , [L4VIRTIO_ID_GPU](#) = 16 , [L4VIRTIO_ID_INPUT](#) = 18 ,
[L4VIRTIO_ID_VSOCK](#) = 19 , [L4VIRTIO_ID_CRYPTIO](#) = 20 , [L4VIRTIO_ID_FS](#) = 26 , [L4VIRTIO_ID_SCMI](#) = 32 ,
[L4VIRTIO_ID_I2C](#) = 34 , [L4VIRTIO_ID_WATCHDOG](#) = 35 , [L4VIRTIO_ID_CAN](#) = 36 , [L4VIRTIO_ID_GPIO](#) = 41 ,
[L4VIRTIO_ID_SPI](#) = 45 , [L4VIRTIO_ID_SOCKET](#) = 0x9999 }
Virtio device IDs as reported in the driver's config space.
- enum [L4virtio_device_status](#) {
[L4VIRTIO_STATUS_ACKNOWLEDGE](#) = 1 , [L4VIRTIO_STATUS_DRIVER](#) = 2 , [L4VIRTIO_STATUS_DRIVER_OK](#) = 4 , [L4VIRTIO_STATUS_FEATURES_OK](#) = 8 ,
[L4VIRTIO_STATUS_DEVICE_NEEDS_RESET](#) = 0x40 , [L4VIRTIO_STATUS_FAILED](#) = 0x80 }
Virtio device status bits.
- enum [L4virtio_feature_bits](#) { [L4VIRTIO_FEATURE_VERSION_1](#) = 32 , [L4VIRTIO_FEATURE_CMD_CONFIG](#) = 160 }
L4virtio-specific feature bits.
- enum [L4_virtio_irq_status](#) { [L4VIRTIO_IRQ_STATUS_VRING](#) = 1 , [L4VIRTIO_IRQ_STATUS_CONFIG](#) = 2 }
VIRTIO IRQ status codes (l4virtio_config_hdr_t::irq_status).
- enum [L4_virtio_cmd](#) {
[L4VIRTIO_CMD_NONE](#) = 0x00000000 , [L4VIRTIO_CMD_SET_STATUS](#) = 0x01000000 , [L4VIRTIO_CMD_CFG_QUEUE](#) = 0x02000000 , [L4VIRTIO_CMD_CFG_CHANGED](#) = 0x04000000 ,
[L4VIRTIO_CMD_NOTIFY_QUEUE](#) = 0x08000000 , [L4VIRTIO_CMD_MASK](#) = 0xff000000 }
Virtio commands for device configuration.

Functions

- [L4_BEGIN_DECLS](#) [l4virtio_config_queue_t](#) * [l4virtio_config_queues](#) ([l4virtio_config_hdr_t](#) const *cfg)
Get the pointer to the first queue config.
- void * [l4virtio_device_config](#) ([l4virtio_config_hdr_t](#) const *cfg)
Get the pointer to the device configuration.
- void [l4virtio_set_feature](#) ([l4_uint32_t](#) *feature_map, unsigned feat)
Set the given feature bit in a feature map.
- void [l4virtio_clear_feature](#) ([l4_uint32_t](#) *feature_map, unsigned feat)
Clear the given feature bit in a feature map.
- unsigned [l4virtio_get_feature](#) ([l4_uint32_t](#) *feature_map, unsigned feat)
Check if the given bit in a feature map is set.
- int [l4virtio_set_status](#) ([l4_cap_idx_t](#) cap, unsigned status) [L4_NOTHROW](#)
- int [l4virtio_config_queue](#) ([l4_cap_idx_t](#) cap, unsigned queue) [L4_NOTHROW](#)
- int [l4virtio_register_ds](#) ([l4_cap_idx_t](#) cap, [l4_cap_idx_t](#) ds_cap, [l4_uint64_t](#) base, [l4_umword_t](#) offset, [l4_umword_t](#) size) [L4_NOTHROW](#)
- int [l4virtio_device_config_ds](#) ([l4_cap_idx_t](#) cap, [l4_cap_idx_t](#) config_ds, [l4_addr_t](#) *ds_offset) [L4_NOTHROW](#)
- int [l4virtio_device_notification_irq](#) ([l4_cap_idx_t](#) cap, unsigned index, [l4_cap_idx_t](#) irq) [L4_NOTHROW](#)

13.7.2.1 Detailed Description

[L4](#) specific VIRTIO Transport layer.

The [L4](#) specific VIRTIO Transport layer is based on [L4Re::Dataspace](#) as shared memory and [L4::Irq](#) for signaling. The VIRTIO configuration space is mostly based on a shared memory implementation too and accompanied by two IPC functions to synchronize the configuration between device and driver.

13.7.2.2 Typedef Documentation

13.7.2.2.1 l4virtio_config_queue_t

```
typedef struct l4virtio_config_queue_t l4virtio_config_queue_t
```

Queue configuration entry.

An array of such entries is available at the [l4virtio_config_hdr_t::queues_offset](#) in the config data space.

Consistency rules for the queue config are:

- A driver might read `num_max` at any time.
- A driver must write to `num`, `desc_addr`, `avail_addr`, and `used_addr` only when `ready` is zero (0). Values in these fields are validated and used by the device only after successfully setting `ready` to one (1), either by the IPC or by `L4VIRTIO_CMD_CFG_QUEUE`.
- The value of `device_notify_index` is valid only when `ready` is one.
- The driver might write to `device_notify_index` at any time, however the change is guaranteed to take effect after a successful `L4VIRTIO_CMD_CFG_QUEUE` or after a `config_queue` IPC. Note, the change might also have immediate effect, depending on the device implementation.

13.7.2.3 Enumeration Type Documentation

13.7.2.3.1 L4_virtio_cmd

enum [L4_virtio_cmd](#)

Virtio commands for device configuration.

Enumerator

| | |
|---------------------------|---------------------------|
| L4VIRTIO_CMD_NONE | No command pending. |
| L4VIRTIO_CMD_SET_STATUS | Set the status register. |
| L4VIRTIO_CMD_CFG_QUEUE | Configure a queue. |
| L4VIRTIO_CMD_CFG_CHANGED | Device config changed. |
| L4VIRTIO_CMD_NOTIFY_QUEUE | Configure a queue. |
| L4VIRTIO_CMD_MASK | Mask to get command bits. |

Definition at line 122 of file [virtio.h](#).

13.7.2.3.2 L4_virtio_irq_status

enum [L4_virtio_irq_status](#)

VIRTIO IRQ status codes (l4virtio_config_hdr_t::irq_status).

Note

l4virtio_config_hdr_t::irq_status is currently unused.

Enumerator

| | |
|----------------------------|--------------------------|
| L4VIRTIO_IRQ_STATUS_VRING | VRING IRQ pending flag. |
| L4VIRTIO_IRQ_STATUS_CONFIG | CONFIG IRQ pending flag. |

Definition at line 113 of file [virtio.h](#).

13.7.2.3.3 L4_virtio_opcodes

enum [L4_virtio_opcodes](#)

Opcodes to setup and configure a device.

Enumerator

| | |
|------------------------|-------------------------------|
| L4VIRTIO_OP_SET_STATUS | Write device status register. |
|------------------------|-------------------------------|

| | |
|----------------------------|-------------------------------------|
| L4VIRTIO_OP_CONFIG_QUEUE | Configure queue. |
| L4VIRTIO_OP_REGISTER_DS | Register shared memory with device. |
| L4VIRTIO_OP_DEVICE_CONFIG | Get device config page. |
| L4VIRTIO_OP_GET_DEVICE_IRQ | Retrieve device notification IRQ. |

Definition at line 52 of file [virtio.h](#).

13.7.2.3.4 L4virtio_device_ids

enum [L4virtio_device_ids](#)

Virtio device IDs as reported in the driver's config space.

Enumerator

| | |
|--------------------------|--------------------------------------|
| L4VIRTIO_ID_NET | Virtual ethernet card. |
| L4VIRTIO_ID_BLOCK | General block device. |
| L4VIRTIO_ID_CONSOLE | Simple device for data IO via ports. |
| L4VIRTIO_ID_RNG | Entropy source. |
| L4VIRTIO_ID_BALLOON | Memory ballooning device. |
| L4VIRTIO_ID_RPMMSG | Device using rpmsg protocol. |
| L4VIRTIO_ID_SCSI | SCSI host device. |
| L4VIRTIO_ID_9P | Device using 9P transport protocol. |
| L4VIRTIO_ID_RPROC_SERIAL | Rproc serial device. |
| L4VIRTIO_ID_CAIF | Device using CAIF network protocol. |
| L4VIRTIO_ID_GPU | GPU. |
| L4VIRTIO_ID_INPUT | Input. |
| L4VIRTIO_ID_VSOCK | Vsock transport. |
| L4VIRTIO_ID_CRYPT | Crypto. |
| L4VIRTIO_ID_FS | FS. |
| L4VIRTIO_ID_SCM | Scmi device. |
| L4VIRTIO_ID_I2C | I2C device. |
| L4VIRTIO_ID_WATCHDOG | Watchdog device. |
| L4VIRTIO_ID_CAN | CAN device. |
| L4VIRTIO_ID_GPIO | Gpio device. |
| L4VIRTIO_ID_SPI | SPI device. |
| L4VIRTIO_ID_SOCKET | Unofficial socket device. |

Definition at line 62 of file [virtio.h](#).

13.7.2.3.5 L4virtio_device_status

enum [L4virtio_device_status](#)

Virtio device status bits.

Enumerator

| | |
|------------------------------------|--------------------------------------|
| L4VIRTIO_STATUS_ACKNOWLEDGE | Guest OS has found device. |
| L4VIRTIO_STATUS_DRIVER | Guest OS knows how to drive device. |
| L4VIRTIO_STATUS_DRIVER_OK | Driver is set up. |
| L4VIRTIO_STATUS_FEATURES_OK | Driver has acknowledged feature set. |
| L4VIRTIO_STATUS_DEVICE_NEEDS_RESET | Device detected fatal error. |
| L4VIRTIO_STATUS_FAILED | Driver detected fatal error. |

Definition at line 90 of file [virtio.h](#).

13.7.2.3.6 L4virtio_feature_bits

enum [L4virtio_feature_bits](#)

L4virtio-specific feature bits.

Enumerator

| | |
|-----------------------------|---|
| L4VIRTIO_FEATURE_VERSION_1 | Virtio protocol version 1 supported. Must be 1 for L4virtio . |
| L4VIRTIO_FEATURE_CMD_CONFIG | Status and queue config are set via cmd field instead of via IPC. |

Definition at line 101 of file [virtio.h](#).

13.7.2.4 Function Documentation

13.7.2.4.1 l4virtio_config_queue()

```
int l4virtio_config_queue (
    l4\_cap\_idx\_t cap,
    unsigned queue)
```

Parameters

| | |
|------------|--------------------------------|
| <i>cap</i> | Capability to the VIRTIO host. |
|------------|--------------------------------|

Trigger queue configuration of the given queue.

Usually all queues are configured when the status is written to running. However, in some cases queues shall be disabled or enabled dynamically, in this case this function triggers a reconfiguration from the shared memory register of the queue config.

Parameters

| | |
|--------------|---|
| <i>queue</i> | Queue index for the queue to be configured. |
|--------------|---|

Return values

| | |
|-------------------|---|
| <i>0</i> | on success. |
| <i>-L4_EIO</i> | The queue's status is invalid. |
| <i>-L4_ERANGE</i> | The queue index exceeds the number of queues. |
| <i>-L4_EINVAL</i> | Otherwise. |

References [L4_CV](#), and [L4_NOTHROW](#).

13.7.2.4.2 l4virtio_config_queues()

```
L4_BEGIN_DECLS l4virtio_config_queue_t * l4virtio_config_queues (
    l4virtio_config_hdr_t const * cfg) [inline]
```

Get the pointer to the first queue config.

Parameters

| | |
|------------|-------------------------------|
| <i>cfg</i> | Pointer to the config header. |
|------------|-------------------------------|

Returns

pointer to queue config of queue 0.

Definition at line 255 of file [virtio.h](#).

References [l4virtio_config_hdr_t::queues_offset](#).

13.7.2.4.3 l4virtio_device_config()

```
void * l4virtio_device_config (
    l4virtio_config_hdr_t const * cfg) [inline]
```

Get the pointer to the device configuration.

Parameters

| | |
|------------|-------------------------------|
| <i>cfg</i> | Pointer to the config header. |
|------------|-------------------------------|

Returns

pointer to device configuration structure.

Definition at line 266 of file [virtio.h](#).

13.7.2.4.4 l4virtio_device_config_ds()

```
int l4virtio_device_config_ds (
    l4_cap_idx_t cap,
    l4_cap_idx_t config_ds,
    l4_addr_t * ds_offset)
```

Parameters

| | |
|------------|----------------------------------|
| <i>cap</i> | Capability to the L4-VIRTIO host |
|------------|----------------------------------|

Get the dataspace with the [L4virtio](#) configuration page.

Parameters

| | |
|------------------|---|
| <i>config_ds</i> | Capability for receiving the dataspace capability for the shared L4-VIRTIO config data space. |
| <i>ds_offset</i> | Offset into the dataspace where the device configuration structure starts. |

References [L4_CV](#), and [L4_NOTHROW](#).

13.7.2.4.5 l4virtio_device_notification_irq()

```
int l4virtio_device_notification_irq (
    l4_cap_idx_t cap,
    unsigned index,
    l4_cap_idx_t irq)
```

Parameters

| | |
|------------|----------------------------------|
| <i>cap</i> | Capability to the L4-VIRTIO host |
|------------|----------------------------------|

Get the notification interrupt corresponding to the given index.

Parameters

| | | |
|-----|--------------|----------------------------------|
| | <i>index</i> | Index of the interrupt. |
| out | <i>irq</i> | Triggerable for the given index. |

Return values

| | |
|------------------|---|
| <i>L4_EOK</i> | Success. |
| <i>L4_ENOSYS</i> | IRQ notification not supported by device. |
| <i><0</i> | Other error. |

An index is only guaranteed to return an IRQ object when the index is set in one of the device notify index fields. The device must return the same interrupt for a given index as long as the index is in use. If an index disappears as a result of a configuration change and then is reused later, the interrupt is not guaranteed to be the same.

Interrupts must always be rerequested after a device reset.

References [L4_END_DECLS](#), and [L4_NOTHROW](#).

13.7.2.4.6 l4virtio_register_ds()

```
int l4virtio_register_ds (
    l4_cap_idx_t cap,
    l4_cap_idx_t ds_cap,
    l4_uint64_t base,
    l4_umword_t offset,
    l4_umword_t size)
```

Parameters

| | |
|------------|-------------------------------|
| <i>cap</i> | Capability to the VIRTIO host |
|------------|-------------------------------|

Register a shared data space with VIRTIO host.

Parameters

| | |
|---------------|--|
| <i>ds_cap</i> | Dataspace capability to register. The lower 8 bits determine the rights mask with which the guest's rights are masked during the registration of the dataspace at the VIRTIO host. |
| <i>base</i> | VIRTIO guest physical start address of shared memory region |
| <i>offset</i> | Offset within the data space that is attached to the given <i>base</i> in the guest physical memory. |
| <i>size</i> | Size of the memory region in the guest |

Return values

| | |
|-------------------|--|
| <i>L4_EOK</i> | Operation successful. |
| <i>-L4_EINVAL</i> | The <i>ds_cap</i> capability is invalid, does not refer to a valid dataspace, is not a trusted dataspace if trusted dataspace validation is enabled, or <i>size</i> and <i>offset</i> specify an invalid region. |
| <i>-L4_ENOMEM</i> | The limit of dataspaces that can be registered has been reached or no capability slot could be allocated. |
| <i>-L4_ERANGE</i> | <i>offset</i> is larger than the size of the dataspace. |
| <i>< 0</i> | Any error returned by the dataspace when queried for information during setup or any error returned by the region manager from attaching the dataspace. |

References [L4_CV](#), and [L4_NOTHROW](#).

13.7.2.4.7 l4virtio_set_status()

```
int l4virtio_set_status (
    l4_cap_idx_t cap,
    unsigned status)
```

Parameters

| | |
|------------|-------------------------------|
| <i>cap</i> | Capability to the VIRTIO host |
|------------|-------------------------------|

Write the VIRTIO status register.

Parameters

| | |
|---------------|--|
| <i>status</i> | Status word to write to the VIRTIO status. |
|---------------|--|

Return values

| | |
|---|-------------|
| 0 | on success. |
|---|-------------|

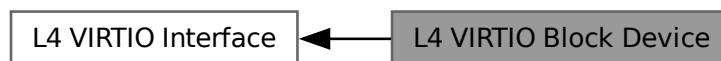
Note

All other registers are accessed via shared memory.

References [L4_CV](#), and [L4_NOTHROW](#).

13.7.3 L4 VIRTIO Block Device

Collaboration diagram for L4 VIRTIO Block Device:



Data Structures

- struct [l4virtio_block_header_t](#)
Header structure of a request for a block device.
- struct [l4virtio_block_discard_t](#)
Structure used for the write zeroes and discard commands.
- struct [l4virtio_block_config_t](#)
Device configuration for block devices.

Typedefs

- typedef struct [l4virtio_block_header_t](#) **[l4virtio_block_header_t](#)**
Header structure of a request for a block device.
- typedef struct [l4virtio_block_discard_t](#) **[l4virtio_block_discard_t](#)**
Structure used for the write zeroes and discard commands.
- typedef struct [l4virtio_block_config_t](#) **[l4virtio_block_config_t](#)**
Device configuration for block devices.

Enumerations

- enum `L4virtio_block_operations` {
`L4VIRTIO_BLOCK_T_IN` = 0 , `L4VIRTIO_BLOCK_T_OUT` = 1 , `L4VIRTIO_BLOCK_T_FLUSH` = 4 ,
`L4VIRTIO_BLOCK_T_GET_ID` = 8 ,
`L4VIRTIO_BLOCK_T_DISCARD` = 11 , `L4VIRTIO_BLOCK_T_WRITE_ZEROES` = 13 }
Kinds of operation over a block device.
- enum `L4virtio_block_status` { `L4VIRTIO_BLOCK_S_OK` = 0 , `L4VIRTIO_BLOCK_S_IOERR` = 1 ,
`L4VIRTIO_BLOCK_S_UNSUPP` = 2 }
Status of a finished block request.
- enum `L4virtio_block_feature_bits`
Block device feature bits.

13.7.3.1 Detailed Description

13.7.3.2 Enumeration Type Documentation

13.7.3.2.1 L4virtio_block_operations

enum `L4virtio_block_operations`

Kinds of operation over a block device.

Enumerator

| | |
|--|-------------------------------------|
| <code>L4VIRTIO_BLOCK_T_IN</code> | Read from device. |
| <code>L4VIRTIO_BLOCK_T_OUT</code> | Write to device. |
| <code>L4VIRTIO_BLOCK_T_FLUSH</code> | Flush data to disk. |
| <code>L4VIRTIO_BLOCK_T_GET_ID</code> | Get device ID. |
| <code>L4VIRTIO_BLOCK_T_DISCARD</code> | Discard a range of sectors. |
| <code>L4VIRTIO_BLOCK_T_WRITE_ZEROES</code> | Write zeroes to a range of sectors. |

Definition at line 19 of file `virtio_block.h`.

13.7.3.2.2 L4virtio_block_status

enum `L4virtio_block_status`

Status of a finished block request.

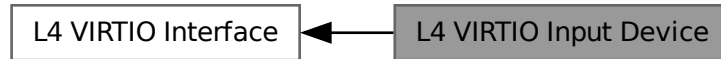
Enumerator

| | |
|--------------------------------------|--------------------------------|
| <code>L4VIRTIO_BLOCK_S_OK</code> | Request finished successfully. |
| <code>L4VIRTIO_BLOCK_S_IOERR</code> | IO error on device. |
| <code>L4VIRTIO_BLOCK_S_UNSUPP</code> | Operation is not supported. |

Definition at line 32 of file `virtio_block.h`.

13.7.4 L4 VIRTIO Input Device

Collaboration diagram for L4 VIRTIO Input Device:



Data Structures

- struct [l4virtio_input_absinfo_t](#)
Information about the absolute axis in the underlying evdev implementation.
- struct [l4virtio_input_devids_t](#)
Device ID information for the device.
- struct [l4virtio_input_config_t](#)
Device configuration for input devices.
- struct [l4virtio_input_event_t](#)
Single event in event or status queue.

Typedefs

- typedef struct [l4virtio_input_absinfo_t](#) [l4virtio_absinfo_t](#)
Information about the absolute axis in the underlying evdev implementation.
- typedef struct [l4virtio_input_devids_t](#) [l4virtio_input_devids_t](#)
Device ID information for the device.
- typedef struct [l4virtio_input_config_t](#) [l4virtio_input_config_t](#)
Device configuration for input devices.
- typedef struct [l4virtio_input_event_t](#) [l4virtio_input_event_t](#)
Single event in event or status queue.

Enumerations

- enum [L4virtio_input_config_select](#)
Device information selectors.

13.7.4.1 Detailed Description

13.7.5 L4 VIRTIO Network Device

Collaboration diagram for L4 VIRTIO Network Device:



Data Structures

- struct [l4virtio_net_header_t](#)
Header structure of a request for a network device.
- struct [l4virtio_net_config_t](#)
Device configuration for network devices.

Typedefs

- typedef struct l4virtio_net_header_t **l4virtio_net_header_t**
Header structure of a request for a network device.
- typedef struct l4virtio_net_config_t **l4virtio_net_config_t**
Device configuration for network devices.

Enumerations

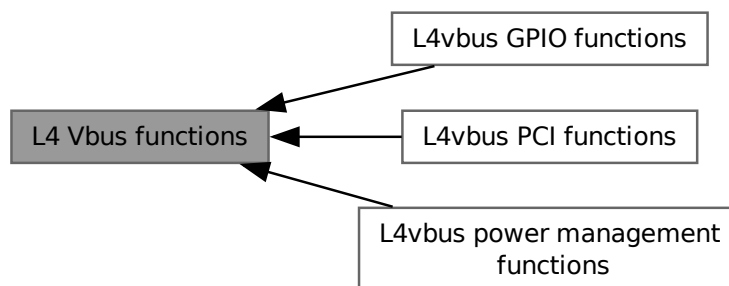
- enum [L4virtio_net_feature_bits](#)
Network device feature bits.

13.7.5.1 Detailed Description

13.8 L4 Vbus functions

C interface of the Vbus API.

Collaboration diagram for L4 Vbus functions:



Topics

- [L4vbus GPIO functions](#) 505
- [L4vbus PCI functions](#) 516
- [L4vbus power management functions](#) 522

Enumerations

- enum `L4vbus_dma_domain_assign_flags` { `L4VBUS_DMAD_UNBIND` = 0 , `L4VBUS_DMAD_BIND` = 1 , `L4VBUS_DMAD_L4RE_DMA_SPACE` = 0 , `L4VBUS_DMAD_KERNEL_DMA_SPACE` = 2 }

Flags for `l4vbus_assign_dma_domain()`.

Functions

- `L4_BEGIN_DECLS` int `l4vbus_get_device_by_hid` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` parent, `l4vbus_device_handle_t` *child, char const *hid, int depth, `l4vbus_device_t` *devinfo)
Find a device by the hardware interface identifier (HID).
- int `l4vbus_get_next_device` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` parent, `l4vbus_device_handle_t` *child, int depth, `l4vbus_device_t` *devinfo)
Find next child following `child`.
- int `l4vbus_get_device` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` dev, `l4vbus_device_t` *devinfo)
Obtain detailed information about a Vbus device.
- int `l4vbus_get_resource` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` dev, unsigned res_idx, `l4vbus_resource_t` *res)
Obtain the resource description of an individual device resource.
- int `l4vbus_is_compatible` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` dev, char const *cid)
Check if the given device has a compatibility ID (CID) or HID that matches `cid`.
- int `l4vbus_get_hid` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` dev, char *hid, unsigned long max_len)
Get the HID (hardware identifier) of a device.
- int `l4vbus_get_adr` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` dev, `l4_uint32_t` *adr)
Get the bus-specific address of a device.
- int `l4vbus_request_ioport` (`l4_cap_idx_t` vbus, `l4vbus_resource_t` const *res)
Request an IO port resource.
- int `l4vbus_assign_dma_domain` (`l4_cap_idx_t` vbus, unsigned domain_id, unsigned flags, `l4_cap_idx_t` dma_space)
Bind or unbind a kernel [DMA space](#) or a `L4Re::Dma_space` to a DMA domain.
- int `l4vbus_release_ioport` (`l4_cap_idx_t` vbus, `l4vbus_resource_t` const *res)
Release a previously requested IO port resource.
- int `l4vbus_vicu_get_cap` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` icu, `l4_cap_idx_t` cap)
Get capability of ICU.

13.8.1 Detailed Description

C interface of the Vbus API.

The virtual bus (Vbus) is a hierarchical (tree) structure of device nodes where each device has a set of resources attached to it. Each virtual bus provides an Icu ([Interrupt controller](#)) for interrupt handling.

The Vbus interface allows a client to find and query devices present on his virtual bus. After obtaining a device handle for a specific device the client can enumerate its resources.

Include File

```
#include <l4/vbus/vbus.h>
```

Refer to [L4vbus](#) for the C++ API.

13.8.2 Enumeration Type Documentation

13.8.2.1 L4vbus_dma_domain_assign_flags

enum [L4vbus_dma_domain_assign_flags](#)

Flags for [l4vbus_assign_dma_domain\(\)](#).

Enumerator

| | |
|------------------------------|---|
| L4VBUS_DMAD_UNBIND | Unbind the given DMA space from the DMA domain. |
| L4VBUS_DMAD_BIND | Bind the given DMA space to the DMA domain. |
| L4VBUS_DMAD_L4RE_DMA_SPACE | The given DMA space is an L4Re::Dma_space . |
| L4VBUS_DMAD_KERNEL_DMA_SPACE | The given DMA space is a kernel DMA space (L4::Task). |

Definition at line 174 of file [vbus.h](#).

13.8.3 Function Documentation

13.8.3.1 l4vbus_assign_dma_domain()

```
int l4vbus_assign_dma_domain (
    l4_cap_idx_t vbus,
    unsigned domain_id,
    unsigned flags,
    l4_cap_idx_t dma_space)
```

Bind or unbind a kernel [DMA space](#) or a [L4Re::Dma_space](#) to a DMA domain.

Parameters

| | |
|------------------|---|
| <i>vbus</i> | Capability of the system bus |
| <i>domain_id</i> | DMA domain ID (resource address of DMA domain found on the vBUS). If the value is ~0U the DMA space of the whole vBUS is used. |
| <i>flags</i> | A combination of L4vbus_dma_domain_assign_flags . |
| <i>dma_space</i> | The DMA space capability to bind or unbind, this must either be an L4Re::Dma_space or a kernel DMA space (L4::Task created with L4_PROTO_DMA_SPACE) and the type must be reflected in the <i>flags</i> . |

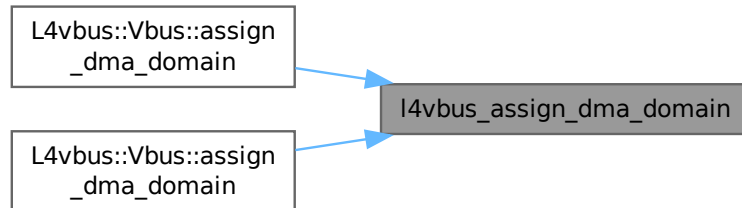
Return values

| | |
|------------|---|
| 0 | Operation completed successfully. |
| -L4_ENOENT | The vbus does not support a global DMA domain or no DMA domain could be found. |
| -L4_EINVAL | Invalid argument used. |
| -L4_EBUSY | DMA domain is already active, this means another DMA space is already assigned. |

References [L4_CV](#).

Referenced by [L4vbus::Vbus::assign_dma_domain\(\)](#), and [L4vbus::Vbus::assign_dma_domain\(\)](#).

Here is the caller graph for this function:



13.8.3.2 l4vbus_get_adr()

```
int l4vbus_get_adr (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t dev,
    l4_uint32_t * adr)
```

Get the bus-specific address of a device.

Parameters

| | | |
|-----|-------------|------------------------------|
| | <i>vbus</i> | Capability of the system bus |
| | <i>dev</i> | Handle of the device |
| out | <i>adr</i> | Address |

Return values

| | |
|-------------------|------------------------------|
| <i>L4_EOK</i> | Success. |
| <i>-L4_ENOSYS</i> | Device has no valid address. |

References [L4_CV](#).

13.8.3.3 l4vbus_get_device()

```
int l4vbus_get_device (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t dev,
    l4vbus_device_t * devinfo)
```

Obtain detailed information about a Vbus device.

Parameters

| | | |
|-----|----------------|---|
| | <i>vbus</i> | Capability of the vbus to which the device is connected. |
| | <i>dev</i> | Device handle of the device from which to retrieve the details. |
| out | <i>devinfo</i> | Information structure which contains details about the device. The pointer might be NULL. |

Return values

| | |
|-------------------|---|
| <i>0</i> | Success. |
| <i>-L4_ENODEV</i> | No device with the given device handle <i>dev</i> could be found. |

References [L4_CV](#).

Referenced by [L4vbus::Device::device\(\)](#).

Here is the caller graph for this function:

**13.8.3.4 l4vbus_get_device_by_hid()**

```

L4_BEGIN_DECLS int l4vbus_get_device_by_hid (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t parent,
    l4vbus_device_handle_t * child,
    char const * hid,
    int depth,
    l4vbus_device_t * devinfo)
  
```

Find a device by the hardware interface identifier (HID).

Parameters

| | |
|---------------|--|
| <i>vbus</i> | Capability of the system bus |
| <i>parent</i> | Handle to the parent to start the search |

This function searches the vbus for a device with the given HID and returns a handle to the first matching device. The HID usually conforms to an ACPI HID or a Linux device tree compatible identifier.

It is possible to have multiple devices with the same HID on a vbus. In order to find all matching devices this function has to be called repeatedly with *child* pointing to the device found in the previous iteration. The iteration starts at *child* that might be any device node in the tree.

Parameters

| | | |
|----------------|----------------|---|
| <i>in, out</i> | <i>child</i> | Handle of the device from where in the device tree the search should start. To start searching from the beginning <i>child</i> must be initialized using the default (L4VBUS_NULL). If a matching device is found, its handle is returned through this parameter. |
| | <i>hid</i> | HID of the device |
| | <i>depth</i> | Maximum depth for the recursive lookup |
| <i>out</i> | <i>devinfo</i> | Device information structure (might be NULL) |

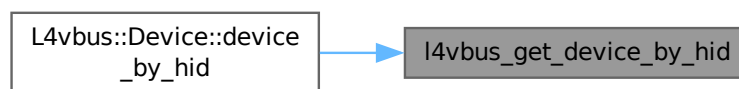
Return values

| | |
|-------------------|--|
| <i>>=0</i> | A device with the given HID was found. |
| <i>-L4_ENOENT</i> | No device with the given HID could be found on the vbus. |
| <i>-L4_EINVAL</i> | Invalid or no HID provided. |
| <i>-L4_ENODEV</i> | Function called on a non-existing device. |

References [L4_CV](#).

Referenced by [L4vbus::Device::device_by_hid\(\)](#).

Here is the caller graph for this function:

**13.8.3.5 l4vbus_get_hid()**

```

int l4vbus_get_hid (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t dev,
    char * hid,
    unsigned long max_len)
  
```

Get the HID (hardware identifier) of a device.

Parameters

| | |
|-------------|--|
| <i>vbus</i> | Capability of the system bus |
| <i>dev</i> | Handle of the device |
| <i>hid</i> | Pointer to a buffer for the HID string |

| | |
|----------------|---------------------------------------|
| <i>max_len</i> | The size of the buffer (<i>hid</i>) |
|----------------|---------------------------------------|

Returns

the length of the HID string on success, else failure

References [L4_CV](#).

13.8.3.6 l4vbus_get_next_device()

```
int l4vbus_get_next_device (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t parent,
    l4vbus_device_handle_t * child,
    int depth,
    l4vbus_device_t * devinfo)
```

Find next child following *child*.

Parameters

| | | |
|---------|----------------|---|
| | <i>vbus</i> | Capability of the system bus |
| | <i>parent</i> | Handle to the parent device (use L4VBUS_ROOT_BUS for the system bus) |
| in, out | <i>child</i> | Handle of the device that precedes the device that shall be returned. To start from the beginning, <i>child</i> must be initialized with L4VBUS_NULL . If a device is found, its handle is returned through this parameter. |
| | <i>depth</i> | Depth to look for |
| out | <i>devinfo</i> | Device information (might be NULL) |

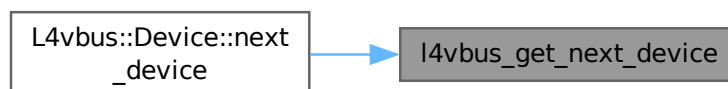
Returns

0 on success, else failure

References [L4_CV](#).

Referenced by [L4vbus::Device::next_device\(\)](#).

Here is the caller graph for this function:



13.8.3.7 l4vbus_get_resource()

```
int l4vbus_get_resource (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t dev,
    unsigned res_idx,
    l4vbus_resource_t * res)
```

Obtain the resource description of an individual device resource.

Parameters

| | | |
|-----|----------------|---|
| | <i>vbus</i> | Capability of the vbus to which the device is connected. |
| | <i>dev</i> | Device handle of the device on the vbus. The device handle can be obtained by using the l4vbus_get_device_by_hid() and l4vbus_get_next_device() functions. |
| | <i>res_idx</i> | Index of the resource for which the resource description should be returned. The total number of resources for a device is available in the l4vbus_device_t structure that is returned by L4vbus::Device::device_by_hid() and L4vbus::Device::next_device() . |
| out | <i>res</i> | Descriptor of the resource. |

This function returns the resource descriptor of an individual device resource selected by the *res_idx* parameter.

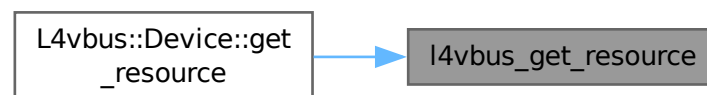
Return values

| | |
|------------|---|
| 0 | Success. |
| -L4_ENOENT | Invalid resource index <i>res_idx</i> . |

References [L4_CV](#).

Referenced by [L4vbus::Device::get_resource\(\)](#).

Here is the caller graph for this function:



13.8.3.8 l4vbus_is_compatible()

```
int l4vbus_is_compatible (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t dev,
    char const * cid)
```


Check if the given device has a compatibility ID (CID) or HID that matches *cid*.

Parameters

| | |
|-------------|---|
| <i>vbus</i> | Capability of the system bus |
| <i>dev</i> | device handle for which the CID shall be tested |
| <i>cid</i> | the compatibility ID to test |

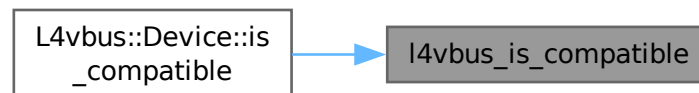
Returns

1 when the given ID (*cid*) matches this device, 0 when the given ID does not match, <0 on error.

References [L4_CV](#).

Referenced by [L4vbus::Device::is_compatible\(\)](#).

Here is the caller graph for this function:



13.8.3.9 l4vbus_release_ioport()

```
int l4vbus_release_ioport (  
    l4_cap_idx_t vbus,  
    l4vbus_resource_t const * res)
```

Release a previously requested IO port resource.

Parameters

| | | |
|----|-------------|---|
| | <i>vbus</i> | Capability of the system bus. |
| in | <i>res</i> | The IO port resource to be released from the bus. |

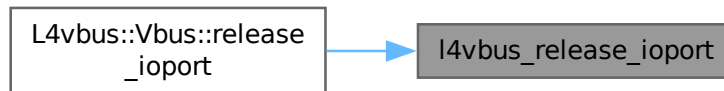
Returns

≥ 0 on success, < 0 on error.

References [L4_CV](#).

Referenced by [L4vbus::Vbus::release_ioport\(\)](#).

Here is the caller graph for this function:

**13.8.3.10 l4vbus_request_ioport()**

```
int l4vbus_request_ioport (
    l4_cap_idx_t vbus,
    l4vbus_resource_t const * res)
```

Request an IO port resource.

Parameters

| | | |
|----|-------------|--|
| | <i>vbus</i> | Capability of the system bus. |
| in | <i>res</i> | The IO port resource to be requested from the bus. |

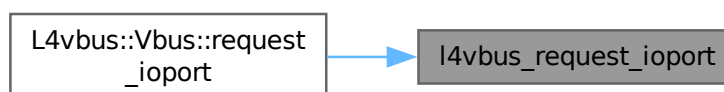
Return values

| | |
|------------|--------------------------------------|
| 0 | Success. |
| -L4_EINVAL | Resource is not an IO port resource. |
| -L4_ENOENT | No matching IO port resource found. |

If any IO port resource is found that contains the requested IO port range the IO ports are obtained.

Referenced by [L4vbus::Vbus::request_ioport\(\)](#).

Here is the caller graph for this function:



13.8.3.11 l4vbus_vicu_get_cap()

```
int l4vbus_vicu_get_cap (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t icu,
    l4_cap_idx_t cap)
```

Get capability of ICU.

Parameters

| | |
|-------------|-------------------------------------|
| <i>vbus</i> | Capability of the system bus. |
| <i>icu</i> | ICU device handle. |
| <i>cap</i> | Capability slot for the capability. |

Returns

0 on success, else failure

References [L4_END_DECLS](#).

Referenced by [L4vbus::l4vbus_vicu\(\)](#).

Here is the caller graph for this function:



13.8.4 L4vbus GPIO functions

Collaboration diagram for L4vbus GPIO functions:



Enumerations

- enum `L4vbus_gpio_generic_func` { `L4VBUS_GPIO_SETUP_INPUT` = 0x100 , `L4VBUS_GPIO_SETUP_OUTPUT` = 0x200 , `L4VBUS_GPIO_SETUP_IRQ` = 0x300 }

Constants for generic GPIO functions.

- enum `L4vbus_gpio_pull_modes` { `L4VBUS_GPIO_PIN_PULL_NONE` = 0x100 , `L4VBUS_GPIO_PIN_PULL_UP` = 0x200 , `L4VBUS_GPIO_PIN_PULL_DOWN` = 0x300 }

Constants for generic GPIO pull up/down resistor configuration.

Functions

- int `l4vbus_gpio_setup` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` handle, unsigned pin, unsigned mode, int value)

Configure the function of a GPIO pin.

- int `l4vbus_gpio_config_pull` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` handle, unsigned pin, unsigned mode)

Generic function to set pull up/down mode.

- int `l4vbus_gpio_config_pad` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` handle, unsigned pin, unsigned func, unsigned value)

Hardware specific configuration function.

- int `l4vbus_gpio_config_get` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` handle, unsigned pin, unsigned func, unsigned *value)

Read hardware specific configuration.

- int `l4vbus_gpio_get` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` handle, unsigned pin)

Read value of GPIO input pin.

- int `l4vbus_gpio_set` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` handle, unsigned pin, int value)

Set GPIO output pin.

- int `l4vbus_gpio_multi_setup` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` handle, unsigned offset, unsigned mask, unsigned mode, unsigned value)

Configure function of multiple GPIO pins at once.

- int `l4vbus_gpio_multi_config_pad` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` handle, unsigned offset, unsigned mask, unsigned func, unsigned value)

Hardware specific configuration function for multiple GPIO pins.

- int `l4vbus_gpio_multi_get` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` handle, unsigned offset, unsigned *data)

Read values of multiple GPIO pins at once.

- int `l4vbus_gpio_multi_set` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` handle, unsigned offset, unsigned mask, unsigned data)

Set multiple GPIO output pins at once.

- int `l4vbus_gpio_to_irq` (`l4_cap_idx_t` vbus, `l4vbus_device_handle_t` handle, unsigned pin)

Create IRQ for GPIO pin.

13.8.4.1 Detailed Description

13.8.4.2 Enumeration Type Documentation

13.8.4.2.1 L4vbus_gpio_generic_func

enum `L4vbus_gpio_generic_func`

Constants for generic GPIO functions.

Enumerator

| | |
|--------------------------|-------------------------|
| L4VBUS_GPIO_SETUP_INPUT | Set GPIO pin to input. |
| L4VBUS_GPIO_SETUP_OUTPUT | Set GPIO pin to output. |
| L4VBUS_GPIO_SETUP_IRQ | Set GPIO pin to IRQ. |

Definition at line 24 of file [vbus_gpio.h](#).

13.8.4.2.2 L4vbus_gpio_pull_modes

enum [L4vbus_gpio_pull_modes](#)

Constants for generic GPIO pull up/down resistor configuration.

Enumerator

| | |
|---------------------------|------------------------------------|
| L4VBUS_GPIO_PIN_PULL_NONE | No pull up or pull down resistors. |
| L4VBUS_GPIO_PIN_PULL_UP | enable pull up resistor |
| L4VBUS_GPIO_PIN_PULL_DOWN | enable pull down resistor |

Definition at line 34 of file [vbus_gpio.h](#).

13.8.4.3 Function Documentation

13.8.4.3.1 l4vbus_gpio_config_get()

```
int l4vbus_gpio_config_get (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    unsigned pin,
    unsigned func,
    unsigned * value)
```

Read hardware specific configuration.

Parameters

| | | |
|-----|---------------|---|
| | <i>vbus</i> | V-BUS capability |
| | <i>handle</i> | Device handle for the GPIO chip |
| | <i>pin</i> | GPIO pin number |
| | <i>func</i> | Hardware specific configuration register to read from. Usually this is an offset to the GPIO chip's base address. |
| out | <i>value</i> | The configuration value. |

Returns

0 if OK, error code otherwise

References [L4_CV](#).

Referenced by [L4vbus::Gpio_pin::config_get\(\)](#).

Here is the caller graph for this function:

**13.8.4.3.2 l4vbus_gpio_config_pad()**

```

int l4vbus_gpio_config_pad (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    unsigned pin,
    unsigned func,
    unsigned value)
  
```

Hardware specific configuration function.

Parameters

| | |
|---------------|--|
| <i>vbus</i> | V-BUS capability |
| <i>handle</i> | Device handle for the GPIO chip |
| <i>pin</i> | GPIO pin number |
| <i>func</i> | Hardware specific configuration register, usually offset to the GPIO chip's base address |
| <i>value</i> | Value which is written into the hardware specific configuration register for the specified pin |

Returns

0 if OK, error code otherwise

References [L4_CV](#).

Referenced by [L4vbus::Gpio_pin::config_pad\(\)](#).

Here is the caller graph for this function:



13.8.4.3.3 l4vbus_gpio_config_pull()

```
int l4vbus_gpio_config_pull (  
    l4_cap_idx_t vbus,  
    l4vbus_device_handle_t handle,  
    unsigned pin,  
    unsigned mode)
```

Generic function to set pull up/down mode.

Parameters

| | |
|---------------|---|
| <i>vbus</i> | V-BUS capability |
| <i>handle</i> | Device handle for the GPIO chip |
| <i>pin</i> | GPIO pin number |
| <i>mode</i> | mode for pull up/down resistors, see L4vbus_gpio_pull_modes |

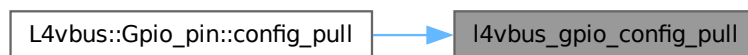
Returns

0 if OK, error code otherwise

References [L4_CV](#).

Referenced by [L4vbus::Gpio_pin::config_pull\(\)](#).

Here is the caller graph for this function:



13.8.4.3.4 l4vbus_gpio_get()

```
int l4vbus_gpio_get (  
    l4_cap_idx_t vbus,  
    l4vbus_device_handle_t handle,  
    unsigned pin)
```

Read value of GPIO input pin.

Parameters

| | |
|-------------|------------------|
| <i>vbus</i> | V-BUS capability |
|-------------|------------------|

| | |
|---------------|---------------------------------|
| <i>handle</i> | Device handle for the GPIO chip |
| <i>pin</i> | GPIO pin number to read from |

Returns

Value of GPIO pin (usually 0 or 1), negative error code otherwise.

References [L4_CV](#).

Referenced by [L4vbus::Gpio_pin::get\(\)](#).

Here is the caller graph for this function:



13.8.4.3.5 l4vbus_gpio_multi_config_pad()

```

int l4vbus_gpio_multi_config_pad (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    unsigned offset,
    unsigned mask,
    unsigned func,
    unsigned value)
  
```

Hardware specific configuration function for multiple GPIO pins.

Parameters

| | |
|---------------|--|
| <i>vbus</i> | V-BUS capability |
| <i>handle</i> | Device handle for the GPIO chip |
| <i>offset</i> | Pin corresponding to the LSB in <i>mask</i> . Note: allowed may be hardware specific. |
| <i>mask</i> | Mask of GPIO pins to configure. A bit set to 1 configures this pin. A maximum of 32 pins can be configured at once. The real number depends on the hardware and the driver implementation. |
| <i>func</i> | Hardware specific configuration register, usually offset to the GPIO chip's base address. |
| <i>value</i> | Value which is written into the hardware specific configuration register for the specified pins |

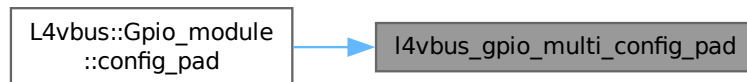
Returns

0 if OK, error code otherwise

References [L4_CV](#).

Referenced by [L4vbus::Gpio_module::config_pad\(\)](#).

Here is the caller graph for this function:

**13.8.4.3.6 l4vbus_gpio_multi_get()**

```

int l4vbus_gpio_multi_get (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    unsigned offset,
    unsigned * data)
  
```

Read values of multiple GPIO pins at once.

Parameters

| | | |
|-----|---------------|---|
| | <i>vbus</i> | V-BUS capability |
| | <i>handle</i> | Device handle for the GPIO chip |
| | <i>offset</i> | Pin corresponding to the LSB in <i>data</i> . Note: allowed may be hardware specific. |
| out | <i>data</i> | Each bit returns the value (0 or 1) for the corresponding GPIO pin. The value of pins that are not accessible is undefined. |

Returns

0 if OK, error code otherwise

References [L4_CV](#).

Referenced by [L4vbus::Gpio_module::get\(\)](#).

Here is the caller graph for this function:



13.8.4.3.7 l4vbus_gpio_multi_set()

```
int l4vbus_gpio_multi_set (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    unsigned offset,
    unsigned mask,
    unsigned data)
```

Set multiple GPIO output pins at once.

Parameters

| | |
|---------------|--|
| <i>vbus</i> | V-BUS capability |
| <i>handle</i> | Device handle for the GPIO chip |
| <i>offset</i> | Pin corresponding to the LSB in <i>data</i> . Note: allowed may be hardware specific. |
| <i>mask</i> | Mask of GPIO pins to set. A bit set to 1 selects this pin. A maximum of 32 pins can be set at once. The real number depends on the hardware and the driver implementation. |
| <i>data</i> | Each bit corresponds to the GPIO pin in <i>mask</i> . The value of each bit is written to the GPIO pin if its bit in <i>mask</i> is set. |

Returns

0 if OK, error code otherwise

References [L4_CV](#).

Referenced by [L4vbus::Gpio_module::set\(\)](#).

Here is the caller graph for this function:



13.8.4.3.8 l4vbus_gpio_multi_setup()

```
int l4vbus_gpio_multi_setup (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    unsigned offset,
    unsigned mask,
    unsigned mode,
    unsigned value)
```

Configure function of multiple GPIO pins at once.

Parameters

| | |
|---------------|--|
| <i>vbus</i> | V-BUS capability |
| <i>handle</i> | Device handle for the GPIO chip |
| <i>offset</i> | Pin corresponding to the LSB in <i>mask</i> . Note: allowed may be hardware specific. |
| <i>mask</i> | Mask of GPIO pins to configure. A bit set to 1 configures this pin. A maximum of 32 pins can be configured at once. The real number depends on the hardware and the driver implementation. |
| <i>mode</i> | GPIO function, see L4vbus_gpio_generic_func for generic functions. Hardware specific functions must be provided in the lower 8 bits. |
| <i>value</i> | Optional value to set the GPIO pins to if they are configured as output pins |

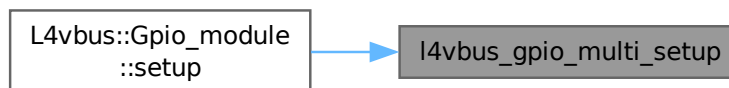
Returns

0 if OK, error code otherwise

References [L4_CV](#).

Referenced by [L4vbus::Gpio_module::setup\(\)](#).

Here is the caller graph for this function:



13.8.4.3.9 l4vbus_gpio_set()

```

int l4vbus_gpio_set (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    unsigned pin,
    int value)
  
```

Set GPIO output pin.

Parameters

| | |
|---------------|---------------------------------|
| <i>vbus</i> | V-BUS capability |
| <i>handle</i> | Device handle for the GPIO chip |
| <i>pin</i> | GPIO pin number to write to |

| | |
|--------------|---|
| <i>value</i> | Value to write to the GPIO pin (usually 0 or 1) |
|--------------|---|

Returns

0 if OK, error code otherwise

References [L4_CV](#).

Referenced by [L4vbus::Gpio_pin::set\(\)](#).

Here is the caller graph for this function:

**13.8.4.3.10 l4vbus_gpio_setup()**

```

int l4vbus_gpio_setup (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    unsigned pin,
    unsigned mode,
    int value)
  
```

Configure the function of a GPIO pin.

Parameters

| | |
|---------------|--|
| <i>vbus</i> | V-BUS capability |
| <i>handle</i> | Device handle for the GPIO chip |
| <i>pin</i> | GPIO pin number |
| <i>mode</i> | GPIO function, see L4vbus_gpio_generic_func for generic functions. Hardware specific functions must be provided in the lower 8 bits. |
| <i>value</i> | Optional value to set the GPIO pin to if it is configured as an output pin |

Returns

0 if OK, error code otherwise

References [L4_CV](#).

Referenced by [L4vbus::Gpio_pin::setup\(\)](#).

Here is the caller graph for this function:

**13.8.4.3.11 l4vbus_gpio_to_irq()**

```
int l4vbus_gpio_to_irq (  
    l4_cap_idx_t vbus,  
    l4vbus_device_handle_t handle,  
    unsigned pin)
```

Create IRQ for GPIO pin.

Parameters

| | |
|---------------|---------------------------------|
| <i>vbus</i> | V-BUS capability |
| <i>handle</i> | Device handle for the GPIO chip |
| <i>pin</i> | GPIO pin to create an IRQ for. |

Returns

IRQ number if OK, negative error code otherwise

References [L4_END_DECLS](#).

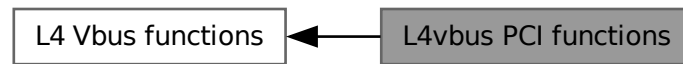
Referenced by [L4vbus::Gpio_pin::to_irq\(\)](#).

Here is the caller graph for this function:



13.8.5 L4vbus PCI functions

Collaboration diagram for L4vbus PCI functions:



Functions

- `L4_BEGIN_DECLS int l4vbus_pci_cfg_read (l4_cap_idx_t vbus, l4vbus_device_handle_t handle, l4_uint32_t bus, l4_uint32_t devfn, l4_uint32_t reg, l4_uint32_t *value, l4_uint32_t width)`
Read from the vPCI configuration space using the PCI root bridge.
- `int l4vbus_pci_cfg_write (l4_cap_idx_t vbus, l4vbus_device_handle_t handle, l4_uint32_t bus, l4_uint32_t devfn, l4_uint32_t reg, l4_uint32_t value, l4_uint32_t width)`
Write to the vPCI configuration space using the PCI root bridge.
- `int l4vbus_pci_irq_enable (l4_cap_idx_t vbus, l4vbus_device_handle_t handle, l4_uint32_t bus, l4_uint32_t devfn, int pin, unsigned char *trigger, unsigned char *polarity)`
Enable PCI interrupt for a specific device using the PCI root bridge.
- `int l4vbus_pcidev_cfg_read (l4_cap_idx_t vbus, l4vbus_device_handle_t handle, l4_uint32_t reg, l4_uint32_t *value, l4_uint32_t width)`
Read from the device's vPCI configuration space.
- `int l4vbus_pcidev_cfg_write (l4_cap_idx_t vbus, l4vbus_device_handle_t handle, l4_uint32_t reg, l4_uint32_t value, l4_uint32_t width)`
Write to the device's vPCI configuration space.
- `int l4vbus_pcidev_irq_enable (l4_cap_idx_t vbus, l4vbus_device_handle_t handle, unsigned char *trigger, unsigned char *polarity)`
Enable the device's PCI interrupt.

13.8.5.1 Detailed Description

13.8.5.2 Function Documentation

13.8.5.2.1 l4vbus_pci_cfg_read()

```

L4_BEGIN_DECLS int l4vbus_pci_cfg_read (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    l4_uint32_t bus,
    l4_uint32_t devfn,
    l4_uint32_t reg,
    l4_uint32_t * value,
    l4_uint32_t width)
  
```

Read from the vPCI configuration space using the PCI root bridge.

Parameters

| | | |
|-----|---------------|--|
| | <i>vbus</i> | Capability of the system bus |
| | <i>handle</i> | Device handle of the PCI root bridge |
| | <i>bus</i> | Bus number |
| | <i>devfn</i> | Device id (upper 16bit) and function (lower 16bit) |
| | <i>reg</i> | Register in configuration space to read |
| out | <i>value</i> | Value that has been read |
| | <i>width</i> | Width to read in bits (e.g. 8, 16, 32) |

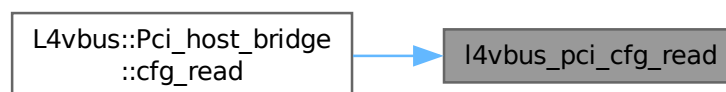
Returns

0 on success, else failure

References [L4_CV](#).

Referenced by [L4vbus::Pci_host_bridge::cfg_read\(\)](#).

Here is the caller graph for this function:

**13.8.5.2.2 l4vbus_pci_cfg_write()**

```

int l4vbus_pci_cfg_write (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    l4_uint32_t bus,
    l4_uint32_t devfn,
    l4_uint32_t reg,
    l4_uint32_t value,
    l4_uint32_t width)
  
```

Write to the vPCI configuration space using the PCI root bridge.

Parameters

| | |
|---------------|--------------------------------------|
| <i>vbus</i> | Capability of the system bus |
| <i>handle</i> | Device handle of the PCI root bridge |

| | |
|--------------|--|
| <i>bus</i> | Bus number |
| <i>devfn</i> | Device id (upper 16bit) and function (lower 16bit) |
| <i>reg</i> | Register in configuration space to write |
| <i>value</i> | Value to write |
| <i>width</i> | Width to write in bits (e.g. 8, 16, 32) |

Returns

0 on success, else failure

References [L4_CV](#).

Referenced by [L4vbus::Pci_host_bridge::cfg_write\(\)](#).

Here is the caller graph for this function:



13.8.5.2.3 l4vbus_pci_irq_enable()

```

int l4vbus_pci_irq_enable (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    l4_uint32_t bus,
    l4_uint32_t devfn,
    int pin,
    unsigned char * trigger,
    unsigned char * polarity)
  
```

Enable PCI interrupt for a specific device using the PCI root bridge.

Parameters

| | | |
|-----|----------------|---|
| | <i>vbus</i> | Capability of the system bus |
| | <i>handle</i> | Device handle of the PCI root bridge |
| | <i>bus</i> | Bus number |
| | <i>devfn</i> | Device id (upper 16bit) and function (lower 16bit) |
| | <i>pin</i> | Interrupt pin (normally as reported in configuration register INTR) |
| out | <i>trigger</i> | False if interrupt is level-triggered |

| | | |
|-----|-----------------|--------------------------------------|
| out | <i>polarity</i> | True if interrupt is of low polarity |
|-----|-----------------|--------------------------------------|

Returns

On success: Interrupt line to be used, else failure

References [L4_CV](#).

Referenced by [L4vbus::Pci_host_bridge::irq_enable\(\)](#).

Here is the caller graph for this function:

**13.8.5.2.4 l4vbus_pciddev_cfg_read()**

```

int l4vbus_pciddev_cfg_read (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    l4_uint32_t reg,
    l4_uint32_t * value,
    l4_uint32_t width)

```

Read from the device's vPCI configuration space.

Parameters

| | | |
|-----|---------------|---|
| | <i>vbus</i> | Capability of the system bus |
| | <i>handle</i> | Device handle of the PCI device |
| | <i>reg</i> | Register in configuration space to read |
| out | <i>value</i> | Value that has been read |
| | <i>width</i> | Width to read in bits (e.g. 8, 16, 32) |

Returns

0 on success, else failure

References [L4_CV](#).

Referenced by [L4vbus::Pci_dev::cfg_read\(\)](#).

Here is the caller graph for this function:

**13.8.5.2.5 l4vbus_pciddev_cfg_write()**

```
int l4vbus_pciddev_cfg_write (  
    l4_cap_idx_t vbus,  
    l4vbus_device_handle_t handle,  
    l4_uint32_t reg,  
    l4_uint32_t value,  
    l4_uint32_t width)
```

Write to the device's vPCI configuration space.

Parameters

| | |
|---------------|--|
| <i>vbus</i> | Capability of the system bus |
| <i>handle</i> | Device handle of the PCI device |
| <i>reg</i> | Register in configuration space to write |
| <i>value</i> | Value to write |
| <i>width</i> | Width to write in bits (e.g. 8, 16, 32) |

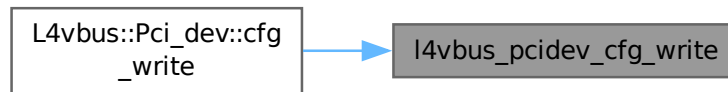
Returns

0 on success, else failure

References [L4_CV](#).

Referenced by [L4vbus::Pci_dev::cfg_write\(\)](#).

Here is the caller graph for this function:



13.8.5.2.6 l4vbus_pcidev_irq_enable()

```

int l4vbus_pcidev_irq_enable (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle,
    unsigned char * trigger,
    unsigned char * polarity)
  
```

Enable the device's PCI interrupt.

Parameters

| | | |
|-----|-----------------|---------------------------------------|
| | <i>vbus</i> | Capability of the system bus |
| | <i>handle</i> | Device handle of the PCI device |
| out | <i>trigger</i> | False if interrupt is level-triggered |
| out | <i>polarity</i> | True if interrupt is of low polarity |

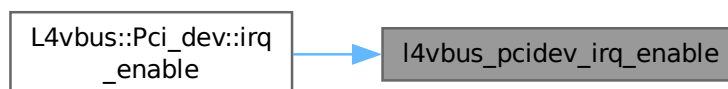
Returns

On success: Interrupt line to be used, else failure

References [L4_END_DECLS](#).

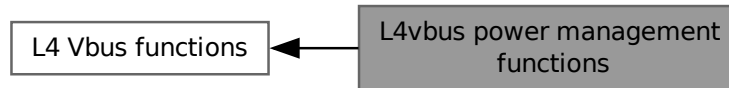
Referenced by [L4vbus::Pci_dev::irq_enable\(\)](#).

Here is the caller graph for this function:



13.8.6 L4vbus power management functions

Collaboration diagram for L4vbus power management functions:



Functions

- [L4_BEGIN_DECLS](#) `int l4vbus_pm_suspend (l4_cap_idx_t vbus, l4vbus_device_handle_t handle)`
Suspend the device.
- `int l4vbus_pm_resume (l4_cap_idx_t vbus, l4vbus_device_handle_t handle)`
Resume the device.

13.8.6.1 Detailed Description

13.8.6.2 Function Documentation

13.8.6.2.1 l4vbus_pm_resume()

```
int l4vbus_pm_resume (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t handle)
```

Resume the device.

Parameters

| | |
|---------------|---|
| <i>vbus</i> | V-BUS capability |
| <i>handle</i> | Handle for the device to be resumed Switches the device from low-power mode to normal operation and restores the saved state. |

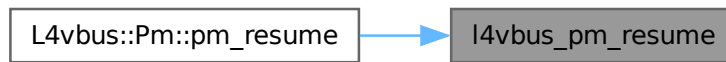
Return values

| | |
|---|----------|
| 0 | Success. |
|---|----------|

References [L4_END_DECLS](#).

Referenced by [L4vbus::Pm< DEC >::pm_resume\(\)](#).

Here is the caller graph for this function:



13.8.6.2.2 l4vbus_pm_suspend()

```
L4_BEGIN_DECLS int l4vbus_pm_suspend (  
    l4_cap_idx_t vbus,  
    l4vbus_device_handle_t handle)
```

Suspend the device.

Parameters

| | |
|---------------|--|
| <i>vbus</i> | V-BUS capability |
| <i>handle</i> | Handle for the device to be suspended Saves the state of the device and puts it into a low-power mode. |

Return values

| | |
|---|----------|
| 0 | Success. |
|---|----------|

References [L4_CV](#).

Referenced by [L4vbus::Pm< DEC >::pm_suspend\(\)](#).

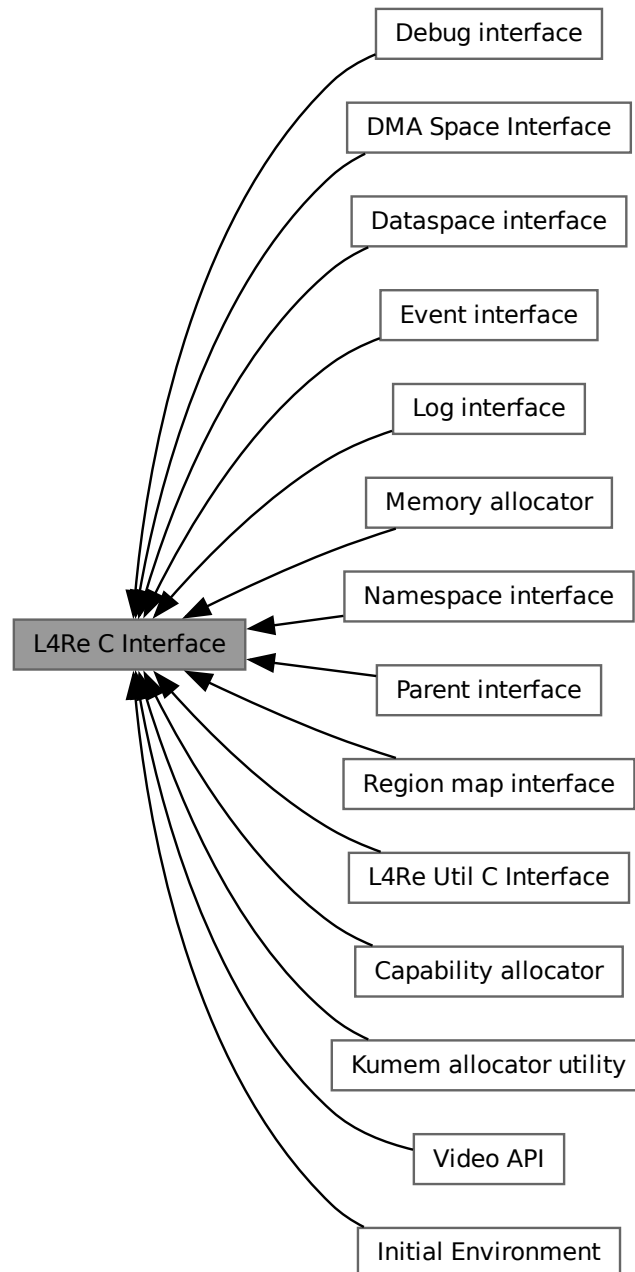
Here is the caller graph for this function:



13.9 L4Re C Interface

Documentation for the [L4Re C Interface](#).

Collaboration diagram for L4Re C Interface:



Topics

•

| | |
|--|-----|
| L4Re Util C Interface | 526 |
| <i>Documentation of the L4 Runtime Environment utility functionality in C.</i> | |
| • | |
| Dataspace interface | 526 |
| <i>Dataspace C interface.</i> | |
| • | |
| Debug interface | 532 |
| • | |
| DMA Space Interface | 533 |
| <i>DMA Space C interface.</i> | |
| • | |
| Event interface | 537 |
| <i>Event C interface.</i> | |
| • | |
| Log interface | 540 |
| <i>Log C interface.</i> | |
| • | |
| Memory allocator | 543 |
| <i>Memory allocator C interface.</i> | |
| • | |
| Namespace interface | 548 |
| <i>Namespace C interface.</i> | |
| • | |
| Parent interface | 552 |
| • | |
| Region map interface | 552 |
| <i>Region map C interface.</i> | |
| • | |
| Capability allocator | 570 |
| <i>Capability allocator C interface.</i> | |
| • | |
| Kumem allocator utility | 571 |
| <i>Kumem allocator utility C interface.</i> | |
| • | |
| Video API | 572 |
| • | |
| Initial Environment | 580 |
| <i>C interface of the initial environment that is provided to an L4 task.</i> | |

Files

- file [inhibitor.h](#)
Inhibitor C interface.

13.9.1 Detailed Description

Documentation for the [L4Re C Interface](#).

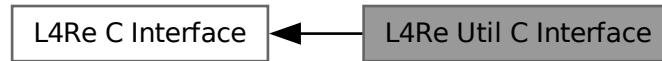
The interface functions closely align with the C++ functions and add no further functionalities.

For new programs it is advised to use the C++ interface.

13.9.2 L4Re Util C Interface

Documentation of the [L4](#) Runtime Environment utility functionality in C.

Collaboration diagram for L4Re Util C Interface:



Documentation of the [L4](#) Runtime Environment utility functionality in C.

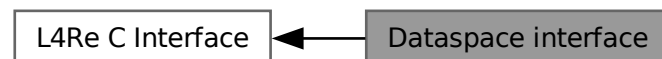
The interface functions closely align with the C++ functions and add no further functionalities.

For new programs it is advised to use the C++ interface.

13.9.3 Dataspace interface

Dataspace C interface.

Collaboration diagram for Dataspace interface:



Data Structures

- struct [l4re_ds_stats_t](#)
Information about the data space.

Enumerations

- enum [l4re_ds_map_flags](#) { }
Flags to specify the memory mapping type of a request.

Functions

- [l4_ret_t l4re_ds_clear](#) ([l4re_ds_t](#) ds, [l4re_ds_offset_t](#) offset, [l4re_ds_size_t](#) size) [L4_NOTHROW](#)
Clear parts of a dataspace.
- [l4_ret_t l4re_ds_allocate](#) ([l4re_ds_t](#) ds, [l4re_ds_offset_t](#) offset, [l4re_ds_size_t](#) size) [L4_NOTHROW](#)
Allocate a range in the dataspace.
- [l4_ret_t l4re_ds_copy_in](#) ([l4re_ds_t](#) ds, [l4re_ds_offset_t](#) dst_offs, [l4re_ds_t](#) src, [l4re_ds_offset_t](#) src_offs, [l4re_ds_size_t](#) size) [L4_NOTHROW](#)
Copy contents from another dataspace.
- [l4re_ds_size_t l4re_ds_size](#) ([l4re_ds_t](#) ds) [L4_NOTHROW](#)
Get size of a dataspace.
- [l4re_ds_flags_t l4re_ds_flags](#) ([l4re_ds_t](#) ds) [L4_NOTHROW](#)
Get flags of the dataspace.
- [l4_ret_t l4re_ds_info](#) ([l4re_ds_t](#) ds, [l4re_ds_stats_t](#) *stats) [L4_NOTHROW](#)
Get information on the dataspace.
- [l4_ret_t l4re_ds_map_info](#) ([l4re_ds_t](#) ds, [l4_addr_t](#) *start_addr, [l4_addr_t](#) *end_addr) [L4_NOTHROW](#)
Get mapping range of dataspace.

Variables

- [L4_BEGIN_DECLS](#) typedef [l4_cap_idx_t](#) [l4re_ds_t](#)
Dataspace type.

13.9.3.1 Detailed Description

Dataspace C interface.

13.9.3.2 Enumeration Type Documentation

13.9.3.2.1 l4re_ds_map_flags

```
enum l4re_ds_map_flags
```

Flags to specify the memory mapping type of a request.

Enumerator

| | |
|-------------------------|--|
| L4RE_DS_F_NORMAL | request normal memory mapping |
| L4RE_DS_F_CACHEABLE | request normal memory mapping |
| L4RE_DS_F_BUFFERABLE | request bufferable (write buffered) mappings |
| L4RE_DS_F_UNCACHEABLE | request uncacheable memory mappings |
| L4RE_DS_F_CACHING_MASK | mask for caching flags |
| L4RE_DS_F_CACHING_SHIFT | shift value for caching flags |

Definition at line 48 of file [dataspace.h](#).

13.9.3.3 Function Documentation

13.9.3.3.1 `l4re_ds_allocate()`

```
l4_ret_t l4re_ds_allocate (
    l4re_ds_t ds,
    l4re_ds_offset_t offset,
    l4re_ds_size_t size)
```

Allocate a range in the dataspace.

Parameters

| | |
|---------------|------------------------------------|
| <i>ds</i> | Dataspace capability. |
| <i>offset</i> | Offset in the dataspace, in bytes. |
| <i>size</i> | Size of the range, in bytes. |

Return values

| | |
|-------------------|--|
| <i>L4_EOK</i> | Success |
| <i>-L4_ERANGE</i> | Given range is outside the dataspace. (A dataspace provider may also silently ignore areas outside the dataspace.) |
| <i>-L4_ENOMEM</i> | Not enough memory available. |
| <i><0</i> | IPC errors |

On success, at least the given range is guaranteed to be allocated. The dataspace manager may also allocate more memory due to page granularity.

The memory is allocated with the same rights as the dataspace capability.

References [L4_CV](#), [L4_NOTHROW](#), and [l4re_ds_t](#).

13.9.3.3.2 `l4re_ds_clear()`

```
l4_ret_t l4re_ds_clear (
    l4re_ds_t ds,
    l4re_ds_offset_t offset,
    l4re_ds_size_t size)
```

Clear parts of a dataspace.

Parameters

| | |
|---------------|-------------------------------------|
| <i>ds</i> | Dataspace capability. |
| <i>offset</i> | Offset within dataspace (in bytes). |
| <i>size</i> | Size of region to clear (in bytes). |

Return values

| | |
|--------------------------|--|
| ≥ 0 | Success. |
| <code>-L4_ERANGE</code> | Given range is outside the dataspace. (A dataspace provider may also silently ignore areas outside the dataspace.) |
| <code>-L4_EACCESS</code> | No L4_CAP_FPAGE_W right on dataspace capability. |
| < 0 | IPC errors |

Zeroes out the memory. Depending on the type of memory the memory could also be deallocated and replaced by a shared zero-page.

References [L4_CV](#), [L4_NOTHROW](#), and [l4re_ds_t](#).

13.9.3.3.3 l4re_ds_copy_in()

```
l4_ret_t l4re_ds_copy_in (
    l4re_ds_t ds,
    l4re_ds_offset_t dst_offs,
    l4re_ds_t src,
    l4re_ds_offset_t src_offs,
    l4re_ds_size_t size)
```

Copy contents from another dataspace.

Parameters

| | |
|-----------------|----------------------------------|
| <i>ds</i> | Destination dataspace. |
| <i>dst_offs</i> | Offset in destination dataspace. |
| <i>src</i> | Source dataspace to copy from. |
| <i>src_offs</i> | Offset in the source dataspace. |
| <i>size</i> | Size to copy (in bytes). |

Return values

| | |
|--------------------------|---|
| <code>L4_EOK</code> | Success |
| <code>-L4_EACCESS</code> | No L4_CAP_FPAGE_W right on the destination dataspace. |
| <code>-L4_EINVAL</code> | Invalid parameter supplied. |
| < 0 | IPC errors |

The copy operation may use copy-on-write mechanisms. The operation may also fail if both dataspaces are not from the same dataspace manager or the dataspace managers do not cooperate.

References [L4_CV](#), [L4_NOTHROW](#), and [l4re_ds_t](#).

13.9.3.3.4 l4re_ds_flags()

```
l4re_ds_flags_t l4re_ds_flags (
    l4re_ds_t ds)
```

Get flags of the dataspace.

Parameters

| | |
|-----------|-----------------------|
| <i>ds</i> | Dataspace capability. |
|-----------|-----------------------|

Return values

| | |
|----------|------------------------|
| ≥ 0 | Flags of the dataspace |
| < 0 | IPC errors |

See also

[L4Re::Dataspace::F::Flags](#)

References [L4_CV](#), [L4_NOTHROW](#), and [l4re_ds_t](#).

13.9.3.3.5 l4re_ds_info()

```
l4_ret_t l4re_ds_info (
    l4re_ds_t ds,
    l4re_ds_stats_t * stats)
```

Get information on the dataspace.

Parameters

| | | |
|------------|--------------|-----------------------|
| | <i>ds</i> | Dataspace capability. |
| <i>out</i> | <i>stats</i> | Dataspace information |

Return values

| | |
|-------|------------|
| 0 | Success |
| < 0 | IPC errors |

References [L4_CV](#), [L4_NOTHROW](#), and [l4re_ds_t](#).

13.9.3.3.6 l4re_ds_map_info()

```
l4_ret_t l4re_ds_map_info (
    l4re_ds_t ds,
    l4_addr_t * start_addr,
    l4_addr_t * end_addr)
```

Get mapping range of dataspace.

Parameters

| | |
|-----------|---|
| <i>ds</i> | Dataspace capability. In case of a MMU-less system, the dataspace must be mapped at the correct address in the task because virtual and physical address must match. This method returns the start and end address of the physically contiguous buffer backing the dataspace. |
|-----------|---|

On MMU-enabled system any page aligned address is permissible. On such systems the method is just a stub.

Parameters

| | | |
|-----|-------------------|---------------------------------------|
| out | <i>start_addr</i> | Start address of dataspace. |
| out | <i>end_addr</i> | End address (inclusive) of dataspace. |

Return values

| | |
|-----------|--|
| >0 | Start/end address have been set and need to be obeyed. |
| 0 | No constraint of mapping address. |
| -L4_EPERM | Cannot infer mapping address. Dataspace not mappable. |
| <0 | IPC errors. |

References [L4_END_DECLS](#), [L4_NOTHROW](#), and [l4re_ds_t](#).

13.9.3.3.7 l4re_ds_size()

```
l4re_ds_size_t l4re_ds_size (
    l4re_ds_t ds)
```

Get size of a dataspace.

Parameters

| | |
|-----------|-----------------------|
| <i>ds</i> | Dataspace capability. |
|-----------|-----------------------|

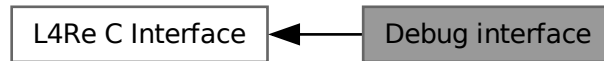
Returns

Size of the dataspace in bytes.

References [L4_CV](#), [L4_NOTHROW](#), and [l4re_ds_t](#).

13.9.4 Debug interface

Collaboration diagram for Debug interface:



Functions

- [L4_BEGIN_DECLS](#) [l4_ret_t](#) [l4re_debug_obj_debug](#) ([l4_cap_idx_t](#) *srv*, unsigned long *function*) [L4_NOTHROW](#)
Call debug function of [L4Re](#) service.

13.9.4.1 Detailed Description

13.9.4.2 Function Documentation

13.9.4.2.1 [l4re_debug_obj_debug\(\)](#)

```

L4_BEGIN_DECLS l4_ret_t l4re_debug_obj_debug (
    l4_cap_idx_t srv,
    unsigned long function)
  
```

Call debug function of [L4Re](#) service.

Parameters

| | |
|-----------------|-------------------|
| <i>srv</i> | Object to call. |
| <i>function</i> | Function to call. |

See also

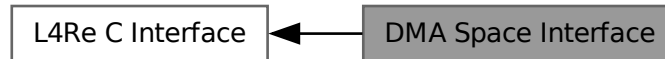
[L4Re::Debug_obj::debug](#)

References [L4_END_DECLS](#), and [L4_NOTHROW](#).

13.9.5 DMA Space Interface

DMA Space C interface.

Collaboration diagram for DMA Space Interface:



Typedefs

- typedef [l4_cap_idx_t](#) [l4re_dma_space_t](#)
DMA space capability type.

Functions

- [l4_ret_t](#) [l4re_dma_space_map](#) ([l4re_dma_space_t](#) dma, [l4re_ds_t](#) src, [l4re_ds_offset_t](#) offset, [l4_size_t](#) *size, unsigned long attrs, enum [l4re_dma_space_direction](#) dir, [l4re_dma_space_dma_addr_t](#) *dma_addr) [L4_NOTHROW](#)
Map the given part of this data space into the DMA address space.
- [l4_ret_t](#) [l4re_dma_space_unmap](#) ([l4re_dma_space_t](#) dma, [l4re_dma_space_dma_addr_t](#) dma_addr, [l4_size_t](#) size, unsigned long attrs, enum [l4re_dma_space_direction](#) dir) [L4_NOTHROW](#)
Unmap the given part of this data space from the DMA address space.
- [l4_ret_t](#) [l4re_dma_space_associate](#) ([l4re_dma_space_t](#) dma, [l4_cap_idx_t](#) dma_task, unsigned long attr) [L4_NOTHROW](#)
Associate a (kernel) [DMA space](#) for a device to this *Dma_space*.
- [l4_ret_t](#) [l4re_dma_space_disassociate](#) ([l4re_dma_space_t](#) dma)
Disassociate the (kernel) [DMA space](#) from this *Dma_space*.

13.9.5.1 Detailed Description

DMA Space C interface.

13.9.5.2 Typedef Documentation

13.9.5.2.1 l4re_dma_space_t

```
typedef l4_cap_idx_t l4re_dma_space_t
```

DMA space capability type.

Managed DMA Address Space.

A managed `Dma_space` represents the [L4Re](#) abstraction of an DMA address space of one or several devices. Devices are assigned to a managed `Dma_space` by binding the `Dma_space` to the respective DMA domain (see [L4vbus::Vbus::assign_dma_domain\(\)](#)), which might link the `Dma_space` with a kernel [DMA space](#). Note that several DMA domains can be bound to the same `Dma_space`. Whenever a device needs direct access to parts of an [L4Re::Dataspace](#), that part of the data space must be mapped to the managed `Dma_space` that is assigned to that device. Binding to DMA domains must happen before mapping. After the DMA accesses to the memory are finished the memory must be unmapped from the device's DMA address space.

Mapping to a managed DMA address space, using `map()`, makes the given parts of the data space visible to the associated device at the returned DMA address. As long as the memory is mapped into a DMA space it is 'pinned' and cannot be subject to dynamic memory management such as swapping. Additionally, `map()` is responsible for the necessary syncing operations before the DMA.

`unmap()` is the reverse operation to `map()` and unmaps the given data-space part for the DMA address space. `unmap()` is responsible for the necessary sync operations after the DMA.

Definition at line 49 of file [dma_space.h](#).

13.9.5.3 Function Documentation

13.9.5.3.1 l4re_dma_space_associate()

```
l4_ret_t l4re_dma_space_associate (
    l4re_dma_space_t dma,
    l4_cap_idx_t dma_task,
    unsigned long attr)
```

Associate a (kernel) [DMA space](#) for a device to this `Dma_space`.

Parameters

| | | |
|----|-----------------|---|
| | <i>dma</i> | DMA space capability |
| in | <i>dma_task</i> | The (kernel) DMA space used for the device that shall be associated with this DMA space. In case no IOMMU is present or configured, the <code>dma_task</code> might be an invalid capability when L4Re::Dma_space::Phys_space is set in <code>attr</code> , in this case the CPUs physical memory is used as DMA address space. |
| in | <i>attr</i> | Attributes for this DMA space. See L4Re::Dma_space::Space_attrb . |

Return values

| | |
|-------------------|---|
| <i>L4_EOK</i> | Operation successful. |
| <i>-L4_EPERM</i> | Insufficient permissions; see precondition. |
| <i>-L4_EINVAL</i> | |
| <i>-L4_ENOENT</i> | |

Precondition

The invoked `Dma_space` capability must have the permission [L4_CAP_FPAGE_W](#).

References [L4_CV](#), and [L4_NOTHROW](#).

13.9.5.3.2 l4re_dma_space_disassociate()

```
l4_ret_t l4re_dma_space_disassociate (
    l4re_dma_space_t dma)
```

Disassociate the (kernel) [DMA space](#) from this `Dma_space`.

Parameters

| | |
|------------|----------------------|
| <i>dma</i> | DMA space capability |
|------------|----------------------|

Return values

| | |
|-------------------|---|
| <i>L4_EOK</i> | Operation successful. |
| <i>-L4_EPERM</i> | Insufficient permissions; see precondition. |
| <i>-L4_ENOENT</i> | |

Precondition

The invoked `Dma_space` capability must have the permission [L4_CAP_FPAGE_W](#).

References [L4_END_DECLS](#).

13.9.5.3.3 l4re_dma_space_map()

```
l4_ret_t l4re_dma_space_map (
    l4re_dma_space_t dma,
    l4re_ds_t src,
    l4re_ds_offset_t offset,
    l4_size_t * size,
    unsigned long attrs,
    enum l4re_dma_space_direction dir,
    l4re_dma_space_dma_addr_t * dma_addr)
```

Map the given part of this data space into the DMA address space.

Parameters

| | | |
|---------|-----------------|--|
| | <i>dma</i> | DMA space capability |
| in | <i>src</i> | Source data space (that describes the memory). Caller needs write right to the data space. |
| in | <i>offset</i> | The offset (bytes) within <i>src</i> . |
| in, out | <i>size</i> | The size (bytes) of the region to be mapped for DMA, after successful mapping the size returned is the size mapped for DMA as a single block. This size might be smaller than the original input size, in this case the caller might call <code>map()</code> again with a new offset and the remaining size. |
| in | <i>attrs</i> | The attributes used for this DMA mapping (a combination of <code>Dma_space::Attribute</code> values). |
| in | <i>dir</i> | The direction of the DMA transfer issued with this mapping. The same value must later be passed to <code>unmap()</code> . |
| out | <i>dma_addr</i> | The DMA address to use for DMA with the associated device. |

Return values

| | |
|-------------------|--|
| <i>L4_EOK</i> | Operation successful. |
| <i>-L4_EPERM</i> | Insufficient permissions; see precondition. |
| <i>-L4_EINVAL</i> | The capability <i>src</i> is invalid or does not refer to a valid dataspace. |
| <i>-L4_EEXIST</i> | The specified region overlaps an existing mapping. |
| <i>-L4_ENOMEM</i> | Not enough memory to allocate internal datastructures. |
| <i>-L4_ERANGE</i> | <i>offset</i> is larger than the size of the dataspace. |

Precondition

The capability *src* must have the permission [L4_CAP_FPAGE_W](#).

Note

`associate()` must be called prior to mapping memory. Usually this is done implicitly when binding the managed `Dma_space` to a DMA domain (see [L4vbus::Vbus::assign_dma_domain\(\)](#)).

References [L4_CV](#), [L4_NOTHROW](#), and [l4re_ds_t](#).

13.9.5.3.4 l4re_dma_space_unmap()

```
l4_ret_t l4re_dma_space_unmap (
    l4re_dma_space_t dma,
    l4re_dma_space_dma_addr_t dma_addr,
    l4_size_t size,
    unsigned long attrs,
    enum l4re_dma_space_direction dir)
```

Unmap the given part of this data space from the DMA address space.

Parameters

| | |
|-----------------|---|
| <i>dma</i> | DMA space capability |
| <i>dma_addr</i> | The DMA address (returned by <code>Dma_space::map()</code>). |
| <i>size</i> | The size (bytes) of the memory region to unmap. |
| <i>attrs</i> | The attributes for the unmap (currently none). |
| <i>dir</i> | The direction of the finished DMA operation. |

Returns

0 in the case of success, a negative error code otherwise.

References [L4_CV](#), and [L4_NOTHROW](#).

13.9.6 Event interface

Event C interface.

Collaboration diagram for Event interface:

**Functions**

- [l4_ret_t l4re_event_get_buffer](#) (const [l4_cap_idx_t](#) server, const [l4re_ds_t](#) ds) [L4_NOTHROW](#)
Get an event signal buffer.
- [l4_ret_t l4re_event_get_num_streams](#) (const [l4_cap_idx_t](#) server) [L4_NOTHROW](#)
Get number of streams.
- [l4_ret_t l4re_event_get_stream_info](#) (const [l4_cap_idx_t](#) server, int idx, [l4re_event_stream_info_t](#) *info) [L4_NOTHROW](#)
Get information on a stream.
- [l4_ret_t l4re_event_get_stream_info_for_id](#) (const [l4_cap_idx_t](#) server, [l4_umword_t](#) stream_id, [l4re_event_stream_info_t](#) *info) [L4_NOTHROW](#)
Get info for a stream given a stream id.
- [l4_ret_t l4re_event_get_axis_info](#) (const [l4_cap_idx_t](#) server, [l4_umword_t](#) id, unsigned naxes, unsigned const *axis, [l4re_event_absinfo_t](#) *info) [L4_NOTHROW](#)
Get Axis information for a stream.

13.9.6.1 Detailed Description

Event C interface.

13.9.6.2 Function Documentation

13.9.6.2.1 l4re_event_get_axis_info()

```
l4_ret_t l4re_event_get_axis_info (
    const l4_cap_idx_t server,
    l4_umword_t id,
    unsigned naxes,
    unsigned const * axis,
    l4re_event_absinfo_t * info)
```

Get Axis information for a stream.

Parameters

| | | |
|-----|---------------|--|
| | <i>server</i> | Server to talk to. |
| | <i>id</i> | Id of the stream to get information from. |
| | <i>naxes</i> | Number of axes in <i>axis</i> array. |
| in | <i>axis</i> | Array of axis IDs whose information should be retrieved. |
| out | <i>info</i> | Information buffer to store the retrieved axis infos. |

Return values

| | |
|----|---------|
| 0 | Success |
| <0 | Error |

See also

[L4Re::Event::get_axis_info](#)

References [L4_END_DECLS](#), and [L4_NOTHROW](#).

13.9.6.2.2 l4re_event_get_buffer()

```
l4_ret_t l4re_event_get_buffer (
    const l4_cap_idx_t server,
    const l4re_ds_t ds)
```

Get an event signal buffer.

Parameters

| | |
|---------------|---------------------------------------|
| <i>server</i> | Server to talk to. |
| <i>ds</i> | Buffer to event data. |

Returns

0 for success, <0 on error

See also

[L4Re::Event::get_buffer](#)

References [L4_CV](#), [L4_NOTHROW](#), and [l4re_ds_t](#).

13.9.6.2.3 l4re_event_get_num_streams()

```
l4_ret_t l4re_event_get_num_streams (
    const l4_cap_idx_t server)
```

Get number of streams.

Parameters

| | |
|---------------|--------------------|
| <i>server</i> | Server to talk to. |
|---------------|--------------------|

Returns

0 for success, <0 on error

See also

[L4Re::Event::get_num_streams](#)

References [L4_CV](#), and [L4_NOTHROW](#).

13.9.6.2.4 l4re_event_get_stream_info()

```
l4_ret_t l4re_event_get_stream_info (
    const l4_cap_idx_t server,
    int idx,
    l4re_event_stream_info_t * info)
```

Get information on a stream.

Parameters

| | | |
|-----|---------------|---------------------|
| | <i>server</i> | Server to talk to. |
| | <i>idx</i> | Index value. |
| out | <i>info</i> | Information buffer. |

Returns

0 for success, <0 on error

See also

[L4Re::Event::get_stream_info](#)

References [L4_CV](#), and [L4_NOTHROW](#).

13.9.6.2.5 l4re_event_get_stream_info_for_id()

```
l4_ret_t l4re_event_get_stream_info_for_id (
    const l4_cap_idx_t server,
    l4_umword_t stream_id,
    l4re_event_stream_info_t * info)
```

Get info for a stream given a stream id.

Parameters

| | | |
|-----|------------------|---------------------|
| | <i>server</i> | Server to talk to. |
| | <i>stream_id</i> | Stream ID. |
| out | <i>info</i> | Information buffer. |

Returns

0 for success, <0 on error

See also

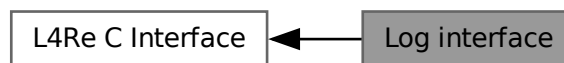
[L4Re::Event::get_stream_info_for_id](#)

References [L4_CV](#), and [L4_NOTHROW](#).

13.9.7 Log interface

Log C interface.

Collaboration diagram for Log interface:



Functions

- [L4_BEGIN_DECLS](#) void [l4re_log_print](#) (char const *string) [L4_NOTHROW](#)
Write a null terminated string to the default log.
- void [l4re_log_printn](#) (char const *string, int len) [L4_NOTHROW](#)
Write a string of a given length to the default log.
- void [l4re_log_print_srv](#) (const [l4_cap_idx_t](#) logcap, char const *string) [L4_NOTHROW](#)
Write a null terminated string to a log.
- void [l4re_log_printn_srv](#) (const [l4_cap_idx_t](#) logcap, char const *string, int len) [L4_NOTHROW](#)
Write a string of a given length to a log.

13.9.7.1 Detailed Description

Log C interface.

13.9.7.2 Function Documentation

13.9.7.2.1 `l4re_log_print()`

```
void l4re_log_print (
    char const * string) [inline]
```

Write a null terminated string to the default log.

Parameters

| | |
|---------------|---------------------------------|
| <i>string</i> | Text to print, null terminated. |
|---------------|---------------------------------|

See also

[L4Re::Log::print](#)

Definition at line 81 of file [log.h](#).

References [L4_NOTHROW](#), and [l4re_log_print_srv\(\)](#).

Here is the call graph for this function:



13.9.7.2.2 `l4re_log_print_srv()`

```
void l4re_log_print_srv (
    const l4\_cap\_idx\_t logcap,
    char const * string)
```

Write a null terminated string to a log.

Parameters

| | |
|---------------|---------------------------------|
| <i>logcap</i> | Log capability (service). |
| <i>string</i> | Text to print, null terminated. |

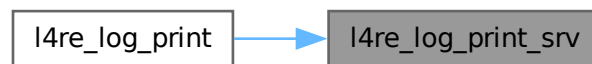
See also

[L4Re::Log::print](#)

References [L4_CV](#), and [L4_NOTHROW](#).

Referenced by [l4re_log_print\(\)](#).

Here is the caller graph for this function:



13.9.7.2.3 l4re_log_printn()

```
void l4re_log_printn (
    char const * string,
    int len) [inline]
```

Write a string of a given length to the default log.

Parameters

| | |
|---------------|---------------------------------|
| <i>string</i> | Text to print, null terminated. |
| <i>len</i> | Length of string in bytes. |

See also

[L4Re::Log::printn](#)

Definition at line 87 of file [log.h](#).

References [L4_NOTHROW](#), and [l4re_log_printn_srv\(\)](#).

Here is the call graph for this function:



13.9.7.2.4 l4re_log_printn_srv()

```
void l4re_log_printn_srv (  
    const l4_cap_idx_t logcap,  
    char const * string,  
    int len)
```

Write a string of a given length to a log.

Parameters

| | |
|---------------|---------------------------------|
| <i>logcap</i> | Log capability (service). |
| <i>string</i> | Text to print, null terminated. |
| <i>len</i> | Length of string in bytes. |

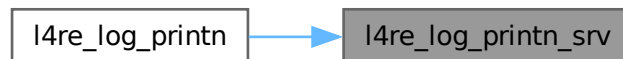
See also

[L4Re::Log::printn](#)

References [L4_CV](#), [L4_INLINE](#), and [L4_NOTHROW](#).

Referenced by [l4re_log_printn\(\)](#).

Here is the caller graph for this function:



13.9.8 Memory allocator

Memory allocator C interface.

Collaboration diagram for Memory allocator:



Enumerations

- enum [l4re_ma_flags](#)
Flags for requesting memory at the memory allocator.

Functions

- [l4_ret_t l4re_ma_alloc](#) (long size, [l4re_ds_t](#) const mem, unsigned long flags) [L4_NOTHROW](#)
Allocate memory.
- [l4_ret_t l4re_ma_alloc_align](#) (long size, [l4re_ds_t](#) const mem, unsigned long flags, unsigned long align) [L4_NOTHROW](#)
Allocate memory.
- [l4_ret_t l4re_ma_alloc_align_srv](#) ([l4_cap_idx_t](#) srv, long size, [l4re_ds_t](#) const mem, unsigned long flags, unsigned long align) [L4_NOTHROW](#)
Allocate memory.

13.9.8.1 Detailed Description

Memory allocator C interface.

13.9.8.2 Enumeration Type Documentation

13.9.8.2.1 l4re_ma_flags

```
enum l4re\_ma\_flags
```

Flags for requesting memory at the memory allocator.

See also

[L4Re::Mem_alloc::Mem_alloc_flags](#)

Definition at line 32 of file [mem_alloc.h](#).

13.9.8.3 Function Documentation

13.9.8.3.1 l4re_ma_alloc()

```
l4\_ret\_t l4re_ma_alloc (  
    long size,  
    l4re\_ds\_t const mem,  
    unsigned long flags) [inline]
```

Allocate memory.

Parameters

| | |
|--------------|---|
| <i>size</i> | Size in bytes to be requested. Allocation granularity is (super)pages, however, the allocator will store the byte-granular given size as the size of the dataspace and consecutively will use this byte-granular size for servicing the dataspace. Allocators may optionally also implement a maximum allocation strategy: if <i>size</i> is a negative value and <i>flags</i> set the <code>Mem_alloc_flags::Continuous</code> bit, the allocator tries to allocate as much memory as possible leaving an amount of at least <code>-size</code> bytes within the associated quota. |
| <i>mem</i> | Capability slot where the capability to the dataspace is received. |
| <i>flags</i> | Special dataspace properties, see l4re_ma_flags |

Return values

| | |
|-------------------|------------------------------|
| <i>0</i> | Success |
| <i>-L4_ERANGE</i> | Given size not supported. |
| <i>-L4_ENOMEM</i> | Not enough memory available. |
| <i>< 0</i> | IPC error |

See also

[L4Re::Mem_alloc::alloc](#)

The memory allocator returns a dataspace.

Note

This function is using the [L4Re::Env::env\(\)](#)->`mem_alloc()` service.

Examples

[examples/libs/l4re/c/ma+rm.c](#).

Definition at line 136 of file [mem_alloc.h](#).

References [L4_NOTHROW](#), [l4re_ds_t](#), and [l4re_ma_alloc_align_srv\(\)](#).

Here is the call graph for this function:



13.9.8.3.2 l4re_ma_alloc_align()

```
l4_ret_t l4re_ma_alloc_align (
    long size,
    l4re_ds_t const mem,
    unsigned long flags,
    unsigned long align) [inline]
```

Allocate memory.

Parameters

| | |
|--------------|---|
| <i>size</i> | Size in bytes to be requested. Allocation granularity is (super)pages, however, the allocator will store the byte-granular given size as the size of the dataspace and consecutively will use this byte-granular size for servicing the dataspace. Allocators may optionally also implement a maximum allocation strategy: if <i>size</i> is a negative value and <i>flags</i> set the <code>Mem_alloc_flags::Continuous</code> bit, the allocator tries to allocate as much memory as possible leaving an amount of at least <code>-size</code> bytes within the associated quota. |
| <i>mem</i> | Capability slot where the capability to the dataspace is received. |
| <i>flags</i> | Special dataspace properties, see l4re_ma_flags |
| <i>align</i> | Log2 alignment of dataspace if supported by allocator, will be at least <code>L4_PAGESHIFT</code> , with <code>Super_pages</code> flag set at least <code>L4_SUPERPAGESHIFT</code> |

Return values

| | |
|-------------------------|------------------------------|
| <code>0</code> | Success |
| <code>-L4_ERANGE</code> | Given size not supported. |
| <code>-L4_ENOMEM</code> | Not enough memory available. |
| <code><0</code> | IPC error |

See also

[L4Re::Mem_alloc::alloc](#) and
[l4re_ma_alloc](#)

The memory allocator returns a dataspace.

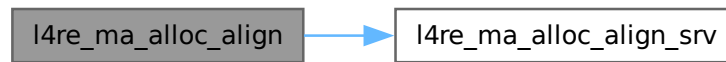
Note

This function is using the [L4Re::Env::env\(\)](#)->`mem_alloc()` service.

Definition at line 144 of file [mem_alloc.h](#).

References [L4_NOTHROW](#), [l4re_ds_t](#), and [l4re_ma_alloc_align_srv\(\)](#).

Here is the call graph for this function:



13.9.8.3.3 l4re_ma_alloc_align_srv()

```

l4_ret_t l4re_ma_alloc_align_srv (
    l4_cap_idx_t srv,
    long size,
    l4re_ds_t const mem,
    unsigned long flags,
    unsigned long align)
  
```

Allocate memory.

Parameters

| | |
|--------------|---|
| <i>srv</i> | Memory allocator service. |
| <i>size</i> | Size to be requested. |
| <i>mem</i> | Capability slot to put the requested dataspace in |
| <i>flags</i> | Flags, see l4re_ma_flags |
| <i>align</i> | Log2 alignment of dataspace if supported by allocator, will be at least L4_PAGESHIFT, with Super_↔ pages flag set at least L4_SUPERPAGESHIFT, default 0 |

Returns

0 on success, <0 on error

See also

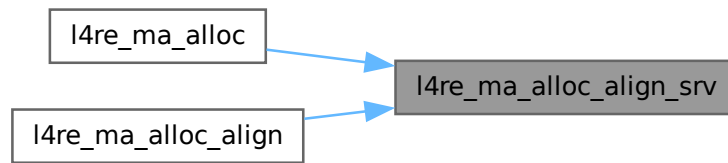
[L4Re::Mem_alloc::alloc](#)

The memory allocator returns a dataspace.

References [L4_CV](#), [L4_INLINE](#), [L4_NOTHROW](#), and [l4re_ds_t](#).

Referenced by [l4re_ma_alloc\(\)](#), and [l4re_ma_alloc_align\(\)](#).

Here is the caller graph for this function:



13.9.9 Namespace interface

Namespace C interface.

Collaboration diagram for Namespace interface:



Enumerations

- enum [l4re_ns_register_flags](#)
Namespace register flags.

Functions

- [l4_ret_t](#) [l4re_ns_query_to_srv](#) ([l4re_namespace_t](#) srv, char const *name, [l4_cap_idx_t](#) const cap, int timeout)
[L4_NOTHROW](#)
Query the name space for the object named by name.
- [l4_ret_t](#) [l4re_ns_query_srv](#) ([l4re_namespace_t](#) srv, char const *name, [l4_cap_idx_t](#) const cap)
[L4_NOTHROW](#)
Query the name space for the object named by name.
- [l4_ret_t](#) [l4re_ns_register_obj_srv](#) ([l4re_namespace_t](#) srv, char const *name, [l4_cap_idx_t](#) const obj, unsigned flags) [L4_NOTHROW](#)
Register an object with a name.

Variables

- [L4_BEGIN_DECLS](#) typedef [l4_cap_idx_t](#) [l4re_namespace_t](#)
Namespace type.

13.9.9.1 Detailed Description

Namespace C interface.

13.9.9.2 Enumeration Type Documentation

13.9.9.2.1 l4re_ns_register_flags

```
enum l4re_ns_register_flags
```

Namespace register flags.

See also

[L4Re::Namespace::Register_flags](#)

Definition at line 29 of file [namespace.h](#).

13.9.9.3 Function Documentation

13.9.9.3.1 l4re_ns_query_srv()

```
l4_ret_t l4re_ns_query_srv (
    l4re_namespace_t srv,
    char const * name,
    l4_cap_idx_t const cap) [inline]
```

Query the name space for the object named by *name*.

Parameters

| | |
|-------------|---|
| <i>srv</i> | Name space server to use for the query. |
| <i>name</i> | String to query. |
| <i>cap</i> | Capability slot where the received capability will be stored. |

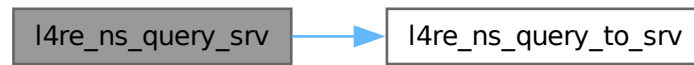
Return values

| | |
|-------------------|---|
| <i>0</i> | Name could be fully resolved. |
| <i>>0</i> | Name could only be partly resolved. The number of remaining characters is returned. |
| <i>-L4_ENOENT</i> | Entry could not be found. |
| <i>-L4_EAGAIN</i> | Entry exists but no object is yet attached. Try again later. |
| <i><0</i> | IPC errors, see l4_error_code_t . |

Definition at line 95 of file [namespace.h](#).

References [L4_NOTHROW](#), [l4re_namespace_t](#), and [l4re_ns_query_to_srv\(\)](#).

Here is the call graph for this function:



13.9.9.3.2 l4re_ns_query_to_srv()

```

l4_ret_t l4re_ns_query_to_srv (
    l4re_namespace_t srv,
    char const * name,
    l4_cap_idx_t const cap,
    int timeout)
  
```

Query the name space for the object named by `name`.

Parameters

| | |
|----------------|--|
| <i>timeout</i> | Timeout of query in milliseconds. The client will only wait if a name already has been registered with the server but no object has been attached yet. |
| <i>srv</i> | Name space server to use for the query. |
| <i>name</i> | String to query. |
| <i>cap</i> | Capability slot where the received capability will be stored. |

Return values

| | |
|-------------------|---|
| <i>0</i> | Name could be fully resolved. |
| <i>>0</i> | Name could only be partly resolved. The number of remaining characters is returned. |
| <i>-L4_ENOENT</i> | Entry could not be found. |
| <i>-L4_EAGAIN</i> | Entry exists but no object is yet attached. Try again later. |
| <i><0</i> | IPC errors, see l4_error_code_t . |

References [L4_CV](#), [L4_INLINE](#), [L4_NOTHROW](#), and [l4re_namespace_t](#).

Referenced by [l4re_ns_query_srv\(\)](#).

Here is the caller graph for this function:



13.9.9.3.3 l4re_ns_register_obj_srv()

```

l4_ret_t l4re_ns_register_obj_srv (
    l4re_namespace_t srv,
    char const * name,
    l4_cap_idx_t const obj,
    unsigned flags)
  
```

Register an object with a name.

Parameters

| | |
|--------------|---|
| <i>srv</i> | Name space server to use for the query. |
| <i>name</i> | Name under which the object should be registered. |
| <i>obj</i> | Capability to object to register. An invalid capability may be given to only reserve the name for later use. |
| <i>flags</i> | Flags to assign to the entry, see L4Re::Namespace::Register_flags . Note that the rights that are assigned to a capability are not only determined by the rights given in these flags but also by the rights with which the <code>obj</code> capability was mapped to the name space. |

Return values

| | |
|-------------------|---|
| <i>0</i> | Object was successfully registered with <i>name</i> . |
| <i>-L4_EEXIST</i> | Name already registered. |
| <i>-L4_EPERM</i> | Insufficient permissions; see precondition. |
| <i>-L4_ENOMEM</i> | Server has insufficient resources. |
| <i>-L4_EINVAL</i> | Invalid parameter. |
| <i><0</i> | IPC errors, see l4_error_code_t . |

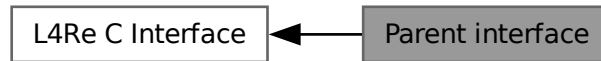
Precondition

The invoked Namespace capability must have the permission [L4_CAP_FPAGE_W](#).

References [L4_CV](#), [L4_INLINE](#), [L4_NOTHROW](#), and [l4re_namespace_t](#).

13.9.10 Parent interface

Collaboration diagram for Parent interface:



13.9.11 Region map interface

Region map C interface.

Collaboration diagram for Region map interface:



Enumerations

- enum `l4re_rm_flags_values` {
`L4RE_RM_F_R` = `L4RE_DS_F_R` , `L4RE_RM_F_W` = `L4RE_DS_F_W` , `L4RE_RM_F_X` = `L4RE_DS_F_X`
 , `L4RE_RM_F_RX` = `L4RE_DS_F_RX` ,
`L4RE_RM_F_RW` = `L4RE_DS_F_RW` , `L4RE_RM_F_RWX` = `L4RE_DS_F_RWX` , `L4RE_RM_F_KERNEL`
 = `0x100` , `L4RE_RM_F_DETACH_FREE` = `0x200` ,
`L4RE_RM_F_PAGER` = `0x400` , `L4RE_RM_F_RESERVED` = `0x800` , `L4RE_RM_CACHING_SHIFT` = `4` ,
`L4RE_RM_F_CACHING` = `L4RE_DS_F_CACHING_MASK` ,
`L4RE_RM_REGION_FLAGS` = `0xffff` , `L4RE_RM_F_CACHE_NORMAL` = `L4RE_DS_F_NORMAL` ,
`L4RE_RM_F_CACHE_BUFFERED` = `L4RE_DS_F_BUFFERABLE` , `L4RE_RM_F_CACHE_UNCACHED`
 = `L4RE_DS_F_UNCACHEABLE` ,
`L4RE_RM_F_SEARCH_ADDR` = `0x020000` , `L4RE_RM_F_IN_AREA` = `0x040000` , `L4RE_RM_F_EAGER_MAP`
 = `0x080000` , `L4RE_RM_F_NO_EAGER_MAP` = `0x100000` ,
`L4RE_RM_F_ATTACH_FLAGS` = `0x1f0000` }

Flags for region operations.

Functions

- `int l4re_rm_reserve_area (l4_addr_t *start, unsigned long size, l4re_rm_flags_t flags, unsigned char align) L4_NOTHROW`
Reserve the given area in the region map.
- `int l4re_rm_free_area (l4_addr_t addr) L4_NOTHROW`
Free an area from the region map.
- `int l4re_rm_attach (void **start, unsigned long size, l4re_rm_flags_t flags, l4re_ds_t mem, l4re_rm_offset_t offs, unsigned char align) L4_NOTHROW`
Attach a data space to a region.
- `int l4re_rm_attach_w_info (void **start, unsigned long size, l4re_rm_flags_t flags, l4re_ds_t mem, l4re_rm_offset_t offs, unsigned char align, char const *name, l4re_ds_offset_t backing_offset) L4_NOTHROW`
Attach a data space to a region.
- `int l4re_rm_detach (void *addr) L4_NOTHROW`
Detach and unmap a region from the address space in the current task.
- `int l4re_rm_detach_ds (void *addr, l4re_ds_t *ds) L4_NOTHROW`
Detach and unmap a region and return affected dataspace in the current task.
- `int l4re_rm_detach_unmap (l4_addr_t addr, l4_cap_idx_t task) L4_NOTHROW`
Detach and unmap in specified task.
- `int l4re_rm_detach_ds_unmap (void *addr, l4re_ds_t *ds, l4_cap_idx_t task) L4_NOTHROW`
Detach and unmap in specified task.
- `int l4re_rm_find (l4_addr_t *addr, unsigned long *size, l4re_rm_offset_t *offset, l4re_rm_flags_t *flags, l4re_ds_t *m) L4_NOTHROW`
Find a region given an address and size.
- `int l4re_rm_get_info (l4_addr_t addr, char *name, unsigned int len, l4re_rm_offset_t *backing_offset) L4_NOTHROW`
Return auxiliary information of a region.
- `void l4re_rm_show_lists (void) L4_NOTHROW`
Dump region map internal data structures.
- `int l4re_rm_reserve_area_srv (l4_cap_idx_t rm, l4_addr_t *start, unsigned long size, l4re_rm_flags_t flags, unsigned char align) L4_NOTHROW`
- `int l4re_rm_free_area_srv (l4_cap_idx_t rm, l4_addr_t addr) L4_NOTHROW`
- `int l4re_rm_attach_srv (l4_cap_idx_t rm, void **start, unsigned long size, l4re_rm_flags_t flags, l4re_ds_t mem, l4re_rm_offset_t offs, unsigned char align) L4_NOTHROW`
- `int l4re_rm_attach_w_info_srv (l4_cap_idx_t rm, void **start, unsigned long size, l4re_rm_flags_t flags, l4re_ds_t mem, l4re_rm_offset_t offs, unsigned char align, char const *name, l4re_ds_offset_t backing_offset) L4_NOTHROW`
- `int l4re_rm_detach_srv (l4_cap_idx_t rm, l4_addr_t addr, l4re_ds_t *ds, l4_cap_idx_t task) L4_NOTHROW`
- `int l4re_rm_find_srv (l4_cap_idx_t rm, l4_addr_t *addr, unsigned long *size, l4re_rm_offset_t *offset, l4re_rm_flags_t *flags, l4re_ds_t *m) L4_NOTHROW`
- `int l4re_rm_get_info_srv (l4_cap_idx_t rm, l4_addr_t addr, char *name, unsigned int len, l4re_rm_offset_t *backing_offset) L4_NOTHROW`
- `void l4re_rm_show_lists_srv (l4_cap_idx_t rm) L4_NOTHROW`
Dump region map internal data structures.

13.9.11.1 Detailed Description

Region map C interface.

13.9.11.2 Enumeration Type Documentation

13.9.11.2.1 l4re_rm_flags_values

enum `l4re_rm_flags_values`

Flags for region operations.

Enumerator

| | |
|--------------------------|---|
| L4RE_RM_F_R | Region is read-only. |
| L4RE_RM_F_KERNEL | Kernel-provided memory (KUMEM). |
| L4RE_RM_F_DETACH_FREE | Free the portion of the data space after detach. |
| L4RE_RM_F_PAGER | Region has a pager. |
| L4RE_RM_F_RESERVED | Region is reserved (blocked). |
| L4RE_RM_CACHING_SHIFT | Start of region mapper cache bits. |
| L4RE_RM_F_CACHING | Mask of all region manager cache bits. |
| L4RE_RM_REGION_FLAGS | Mask of all region flags. |
| L4RE_RM_F_CACHE_NORMAL | Cache bits for normal cacheable memory. |
| L4RE_RM_F_CACHE_BUFFERED | Cache bits for buffered (write combining) memory. |
| L4RE_RM_F_CACHE_UNCACHED | Cache bits for uncached memory. |
| L4RE_RM_F_SEARCH_ADDR | Search for a suitable address range. |
| L4RE_RM_F_IN_AREA | Search only in area, or map into area. |
| L4RE_RM_F_EAGER_MAP | Eagerly map the attached data space in. |
| L4RE_RM_F_NO_EAGER_MAP | Prevent eager mapping of the attached data space. |
| L4RE_RM_F_ATTACH_FLAGS | Mask of all attach flags. |

Definition at line 30 of file `rm.h`.

13.9.11.3 Function Documentation

13.9.11.3.1 l4re_rm_attach()

```
int l4re_rm_attach (
    void ** start,
    unsigned long size,
    l4re_rm_flags_t flags,
    l4re_ds_t mem,
    l4re_rm_offset_t offs,
    unsigned char align) [inline]
```

Attach a data space to a region.

Parameters

| | | |
|----------------|--------------|---|
| <i>in, out</i> | <i>start</i> | Virtual start address where the region manager shall attach the data space. Will be rounded down to the nearest start of a page. If L4Re::Rm::F::Search_addr is given this value is used as the start address to search for a free virtual memory region and the resulting address is returned here. If L4Re::Rm::F::In_area is given the value is used as a selector for the area (see L4Re::Rm::reserve_area) to attach the data space to. |
| | <i>size</i> | Size of the data space to attach (in bytes). Will be rounded up to the nearest multiple of the page size. |
| | <i>flags</i> | The flags control how and with which rights the dataspace is attached to the region. See L4Re::Rm::F::Attach_flags and L4Re::Rm::F::Region_flags . The caller must specify the desired rights of the attached region explicitly. The default set of rights is empty. If the <code>Eager_map</code> flag is set this function may also return L4Re::Dataspace::map error codes if the mapping fails. |
| | <i>mem</i> | Data space. |
| | <i>offs</i> | Offset into the data space to use. |
| | <i>align</i> | Alignment of the virtual region, log2-size, default: a page (L4_PAGESHIFT). This is only meaningful if the L4Re::Rm::F::Search_addr flag is used. |

Return values

| | |
|-------------------|--|
| 0 | Success |
| -L4_ENOENT | No area could be found (see L4Re::Rm::F::In_area) |
| -L4_EPERM | Operation not allowed. |
| -L4_EINVAL | |
| -L4_EADDRNOTAVAIL | The given address is not available. |
| <0 | IPC errors |

Makes the whole or parts of a data space visible in the virtual memory of the corresponding task. The corresponding region in the virtual address space is backed with the contents of the dataspace.

Note

When searching for a free place in the virtual address space, the space between *start* and the end of the virtual address space is

See also

[L4Re::Rm::attach](#)

This function is using the `L4::Env::env()->rm()` service.

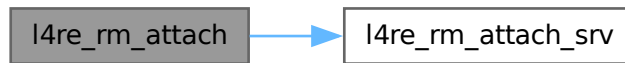
Examples

[examples/libs/l4re/c/ma+rm.c](#).

Definition at line 414 of file [rm.h](#).

References [L4_NOTHROW](#), [l4re_ds_t](#), and [l4re_rm_attach_srv\(\)](#).

Here is the call graph for this function:



13.9.11.3.2 l4re_rm_attach_srv()

```
int l4re_rm_attach_srv (  
    l4_cap_idx_t rm,  
    void ** start,  
    unsigned long size,  
    l4re_rm_flags_t flags,  
    l4re_ds_t mem,  
    l4re_rm_offset_t offs,  
    unsigned char align)
```

See also

[L4Re::Rm::attach](#)

References [L4_CV](#), [L4_NOTHROW](#), and [l4re_ds_t](#).

Referenced by [l4re_rm_attach\(\)](#).

Here is the caller graph for this function:



13.9.11.3.3 l4re_rm_attach_w_info()

```
int l4re_rm_attach_w_info (  
    void ** start,  
    unsigned long size,  
    l4re_rm_flags_t flags,  
    l4re_ds_t mem,  
    l4re_rm_offset_t offs,
```

```

unsigned char align,
char const * name,
l4re_ds_offset_t backing_offset) [inline]

```

Attach a data space to a region.

Parameters

| | | |
|----------------|-----------------------|---|
| <i>in, out</i> | <i>start</i> | Virtual start address where the region manager shall attach the data space. Will be rounded down to the nearest start of a page. If L4Re::Rm::F::Search_addr is given this value is used as the start address to search for a free virtual memory region and the resulting address is returned here. If L4Re::Rm::F::In_area is given the value is used as a selector for the area (see L4Re::Rm::reserve_area) to attach the data space to. |
| | <i>size</i> | Size of the data space to attach (in bytes). Will be rounded up to the nearest multiple of the page size. |
| | <i>flags</i> | The flags control how and with which rights the dataspace is attached to the region. See L4Re::Rm::F::Attach_flags and L4Re::Rm::F::Region_flags . The caller must specify the desired rights of the attached region explicitly. The default set of rights is empty. If the <code>F::Eager_map</code> flag is set this function may also return L4Re::Dataspace::map error codes if the mapping fails. |
| | <i>mem</i> | Data space. |
| | <i>offs</i> | Offset into the data space to use. |
| | <i>align</i> | Alignment of the virtual region, log2-size, default: a page (L4_PAGESHIFT). This is only meaningful if the L4Re::Rm::F::Search_addr flag is used. |
| | <i>name</i> | Optional name of the region. |
| | <i>backing_offset</i> | Optional value describing an offset into the backing store of this region. |

Return values

| | |
|-------------------|--|
| 0 | Success |
| -L4_ENOENT | No area could be found (see L4Re::Rm::F::In_area) |
| -L4_EPERM | Operation not allowed. |
| -L4_EINVAL | |
| -L4_EADDRNOTAVAIL | The given address is not available. |
| <0 | IPC errors |

Makes the whole or parts of a data space visible in the virtual memory of the corresponding task. The corresponding region in the virtual address space is backed with the contents of the dataspace.

Note

When searching for a free place in the virtual address space, the space between *start* and the end of the virtual address space is

See also

[L4Re::Rm::attach](#)

This function is using the `L4::Env::env()->rm()` service.

Definition at line 423 of file `rm.h`.

References [L4_NOTHROW](#), [l4re_ds_t](#), and [l4re_rm_attach_w_info_srv\(\)](#).

Here is the call graph for this function:



13.9.11.3.4 l4re_rm_attach_w_info_srv()

```

int l4re_rm_attach_w_info_srv (
    l4_cap_idx_t rm,
    void ** start,
    unsigned long size,
    l4re_rm_flags_t flags,
    l4re_ds_t mem,
    l4re_rm_offset_t offs,
    unsigned char align,
    char const * name,
    l4re_ds_offset_t backing_offset)
  
```

See also

[L4Re::Rm::attach](#)

References [L4_CV](#), [L4_NOTHROW](#), and [l4re_ds_t](#).

Referenced by [l4re_rm_attach_w_info\(\)](#).

Here is the caller graph for this function:



13.9.11.3.5 l4re_rm_detach()

```
int l4re_rm_detach (  
    void * addr) [inline]
```

Detach and unmap a region from the address space in the current task.

Parameters

| | |
|-------------|----------------------------------|
| <i>addr</i> | Address of the region to detach. |
|-------------|----------------------------------|

Return values

| | |
|---|------------------|
| L4Re::Rm::Detach_result | On success. |
| <code>-L4_ENOENT</code> | No region found. |
| <code><0</code> | IPC errors |

Frees a region in the virtual address space given by *addr*. The corresponding part of the address space is now available again.

Also

See also

[L4Re::Rm::detach](#)

This function is using the `L4::Env::env()->rm()` service.

Definition at line 436 of file [rm.h](#).

References [L4_BASE_TASK_CAP](#), [L4_NOTHROW](#), and [l4re_rm_detach_srv\(\)](#).

Here is the call graph for this function:



13.9.11.3.6 l4re_rm_detach_ds()

```
int l4re_rm_detach_ds (
    void * addr,
    l4re_ds_t * ds) [inline]
```

Detach and unmap a region and return affected dataspace in the current task.

Parameters

| | | |
|-----|-------------|-------------------------------------|
| | <i>addr</i> | Address of the region to detach. |
| out | <i>ds</i> | Returns dataspace that is affected. |

Return values

| | |
|---|------------------|
| L4Re::Rm::Detach_result | On success. |
| <code>-L4_ENOENT</code> | No region found. |
| <code><0</code> | IPC errors |

Frees a region in the virtual address space given by *addr*. The corresponding part of the address space is now available again.

Also

See also

[L4Re::Rm::detach](#)

This function is using the `L4::Env::env()->rm()` service.

Examples

[examples/libs/l4re/c/ma+rm.c](#).

Definition at line 449 of file [rm.h](#).

References [L4_BASE_TASK_CAP](#), [L4_NOTHROW](#), [l4re_ds_t](#), and [l4re_rm_detach_srv\(\)](#).

Here is the call graph for this function:



13.9.11.3.7 l4re_rm_detach_ds_unmap()

```
int l4re_rm_detach_ds_unmap (
    void * addr,
    l4re_ds_t * ds,
    l4_cap_idx_t task) [inline]
```

Detach and unmap in specified task.

Parameters

| | | |
|-----|-------------|---|
| | <i>addr</i> | Address of the region to detach. |
| out | <i>ds</i> | Returns dataspace that is affected. |
| | <i>task</i> | Task to unmap pages from, specify L4_INVALID_CAP to not unmap |

Returns

0 on success, <0 on error

Also

See also

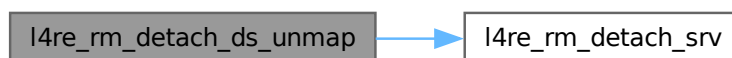
[L4Re::Rm::detach](#)

This function is using the L4::Env::env()->rm() service.

Definition at line 456 of file [rm.h](#).

References [L4_NOTHROW](#), [l4re_ds_t](#), and [l4re_rm_detach_srv\(\)](#).

Here is the call graph for this function:



13.9.11.3.8 l4re_rm_detach_srv()

```
int l4re_rm_detach_srv (
    l4_cap_idx_t rm,
    l4_addr_t addr,
    l4re_ds_t * ds,
    l4_cap_idx_t task)
```

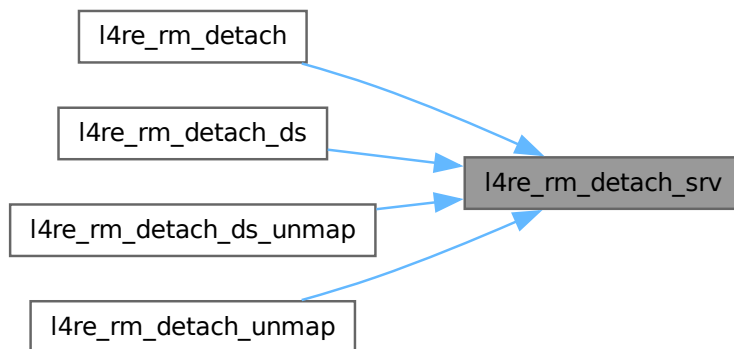
See also

[L4Re::Rm::detach](#)

References [L4_CV](#), [L4_NOTHROW](#), and [l4re_ds_t](#).

Referenced by [l4re_rm_detach\(\)](#), [l4re_rm_detach_ds\(\)](#), [l4re_rm_detach_ds_unmap\(\)](#), and [l4re_rm_detach_unmap\(\)](#).

Here is the caller graph for this function:



13.9.11.3.9 l4re_rm_detach_unmap()

```

int l4re_rm_detach_unmap (
    l4_addr_t addr,
    l4_cap_idx_t task) [inline]
  
```

Detach and unmap in specified task.

Parameters

| | |
|-------------|---|
| <i>addr</i> | Address of the region to detach. |
| <i>task</i> | Task to unmap pages from, specify L4_INVALID_CAP to not unmap |

Returns

0 on success, <0 on error

Also

See also

[L4Re::Rm::detach](#)

This function is using the `L4::Env::env()->rm()` service.

Definition at line 443 of file [rm.h](#).

References [L4_NOTHROW](#), and [l4re_rm_detach_srv\(\)](#).

Here is the call graph for this function:



13.9.11.3.10 l4re_rm_find()

```

int l4re_rm_find (
    l4_addr_t * addr,
    unsigned long * size,
    l4re_rm_offset_t * offset,
    l4re_rm_flags_t * flags,
    l4re_ds_t * m) [inline]
  
```

Find a region given an address and size.

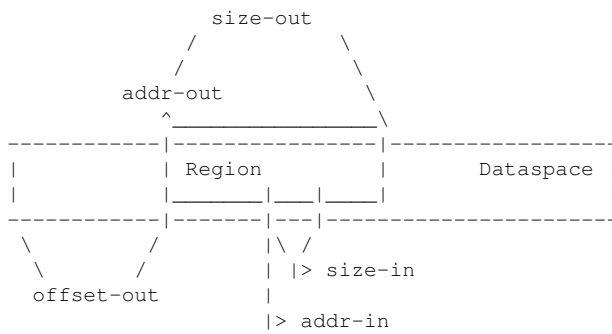
Parameters

| | | |
|---------|---------------|---|
| in, out | <i>addr</i> | Address to look for. Returns the start address of the found region. |
| in, out | <i>size</i> | Size of the area to look for (in bytes). Returns the size of the found region (in bytes). |
| out | <i>offset</i> | Offset at the beginning of the region within the associated dataspace. |
| out | <i>flags</i> | Region flags, see <code>F::Region_flags</code> (and <code>F::In_area</code>). |
| out | <i>m</i> | Associated dataspace or paging service. |

Return values

| | |
|------------|------------------------|
| 0 | Success |
| -L4_EPERM | Operation not allowed. |
| -L4_ENOENT | No region found. |
| <0 | IPC errors |

This function returns the properties of the region that contains the area described by the `addr` and `size` parameter. If no such region is found but a reserved area, the area is returned and `F::In_area` is set in `flags`. Note, in the case of an area the `offset` and `m` return values are invalid.

**Note**

The value of the size input parameter should be 1 to assure that a region can be determined unambiguously.

See also

[L4Re::Rm::find](#)

Definition at line 463 of file [rm.h](#).

References [L4_NOTHROW](#), [l4re_ds_t](#), and [l4re_rm_find_srv\(\)](#).

Here is the call graph for this function:

**13.9.11.3.11 l4re_rm_find_srv()**

```
int l4re_rm_find_srv (
    l4_cap_idx_t rm,
    l4_addr_t * addr,
    unsigned long * size,
    l4re_rm_offset_t * offset,
    l4re_rm_flags_t * flags,
    l4re_ds_t * m)
```

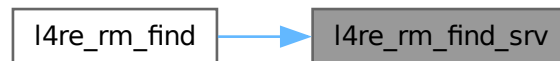
See also

[L4Re::Rm::find](#)

References [L4_CV](#), [L4_NOTHROW](#), and [l4re_ds_t](#).

Referenced by [l4re_rm_find\(\)](#).

Here is the caller graph for this function:



13.9.11.3.12 l4re_rm_free_area()

```
int l4re_rm_free_area (  
    l4_addr_t addr) [inline]
```

Free an area from the region map.

Parameters

| | |
|-------------|-------------------------------------|
| <i>addr</i> | An address within the area to free. |
|-------------|-------------------------------------|

Return values

| | |
|------------|----------------|
| 0 | Success |
| -L4_ENOENT | No area found. |
| <0 | IPC errors |

Note

The data spaces that are attached to that area are not detached by this operation.

See also

`reserve_area()` for more information about areas.

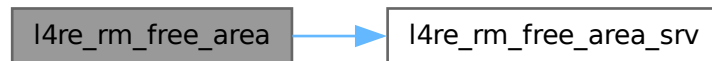
[L4Re::Rm::free_area](#)

This function is using the `L4::Env::env()->rm()` service.

Definition at line 408 of file `rm.h`.

References [L4_NOTHROW](#), and [l4re_rm_free_area_srv\(\)](#).

Here is the call graph for this function:



13.9.11.3.13 `l4re_rm_free_area_srv()`

```
int l4re_rm_free_area_srv (  
    l4_cap_idx_t rm,  
    l4_addr_t addr)
```

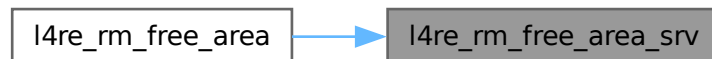
See also

[L4Re::Rm::free_area](#)

References [L4_CV](#), and [L4_NOTHROW](#).

Referenced by [l4re_rm_free_area\(\)](#).

Here is the caller graph for this function:



13.9.11.3.14 l4re_rm_get_info()

```
int l4re_rm_get_info (
    l4_addr_t addr,
    char * name,
    unsigned int len,
    l4re_rm_offset_t * backing_offset) [inline]
```

Return auxiliary information of a region.

This is a debugging feature and might not be available.

Parameters

| | | |
|-----|-----------------------|--------------------------------|
| | <i>addr</i> | Virtual address of the region. |
| out | <i>name</i> | Name of the region. |
| out | <i>backing_offset</i> | Backing offset information. |

Return values

| | |
|------------|-------------------------|
| 0 | Success |
| -L4_ENOENT | Region not found. |
| -L4_ENOSYS | Function not available. |
| <0 | IPC errors |

Parameters

| | |
|------------|--|
| <i>len</i> | Length of the name given in name argument, in bytes. |
|------------|--|

See also

[L4Re::Rm::get_info](#)

Definition at line 479 of file [rm.h](#).

References [L4_NOTHROW](#), and [l4re_rm_get_info_srv\(\)](#).

Here is the call graph for this function:



13.9.11.3.15 l4re_rm_get_info_srv()

```
int l4re_rm_get_info_srv (
    l4_cap_idx_t rm,
    l4_addr_t addr,
    char * name,
    unsigned int len,
    l4re_rm_offset_t * backing_offset)
```

See also

[L4Re::Rm::get_info](#)

References [L4_CV](#), and [L4_NOTHROW](#).

Referenced by [l4re_rm_get_info\(\)](#).

Here is the caller graph for this function:



13.9.11.3.16 l4re_rm_reserve_area()

```
int l4re_rm_reserve_area (
    l4_addr_t * start,
    unsigned long size,
    l4re_rm_flags_t flags,
    unsigned char align) [inline]
```

Reserve the given area in the region map.

Parameters

| | | |
|---------|--------------|---|
| in, out | <i>start</i> | The virtual start address of the area to reserve. Returns the start address of the area. |
| | <i>size</i> | The size of the area to reserve (in bytes). |
| | <i>flags</i> | Flags for the reserved area (see L4Re::Rm::F::Region_flags and L4Re::Rm::F::Attach_flags). |
| | <i>align</i> | Alignment of area if searched as bits (log2 value). |

Return values

| | |
|-------------------|------------------------------------|
| 0 | Success |
| -L4_EADDRNOTAVAIL | The given area cannot be reserved. |
| <0 | IPC errors |

This function reserves an area within the virtual address space managed by the region map. There are two kinds of areas available:

- Reserved areas (*flags* = [L4Re::Rm::F::Reserved](#)), where no data spaces can be attached
- Special purpose areas (*flags* = 0), where data spaces can be attached to the area via the [L4Re::Rm::F::In_area](#) flag and a start address within the area itself.

Note

When searching for a free place in the virtual address space (with *flags* = [L4Re::Rm::F::Search_addr](#)), the space between *start* and the end of the virtual address space is searched.

See also

[L4Re::Rm::reserve_area](#)

This function is using the `L4::Env::env()->rm()` service.

Definition at line 400 of file [rm.h](#).

References [L4_NOTHROW](#), and [l4re_rm_reserve_area_srv\(\)](#).

Here is the call graph for this function:



13.9.11.3.17 l4re_rm_reserve_area_srv()

```

int l4re_rm_reserve_area_srv (
    l4_cap_idx_t rm,
    l4_addr_t * start,
    unsigned long size,
    l4re_rm_flags_t flags,
    unsigned char align)

```

See also

[L4Re::Rm::reserve_area](#)

References [L4_CV](#), and [L4_NOTHROW](#).

Referenced by [l4re_rm_reserve_area\(\)](#).

Here is the caller graph for this function:



13.9.11.3.18 l4re_rm_show_lists()

```
void l4re_rm_show_lists (
    void ) [inline]
```

Dump region map internal data structures.

This function is using the `L4::Env::env()->rm()` service.

Definition at line 471 of file [rm.h](#).

References [L4_NOTHROW](#), and [l4re_rm_show_lists_srv\(\)](#).

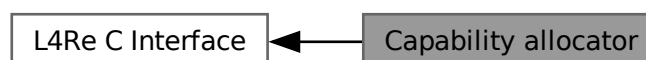
Here is the call graph for this function:



13.9.12 Capability allocator

Capability allocator C interface.

Collaboration diagram for Capability allocator:



Functions

- [L4_BEGIN_DECLS](#) [l4_cap_idx_t](#) **l4re_util_cap_alloc** (void) [L4_NOTHROW](#)
Get free capability index at capability allocator.
- void **l4re_util_cap_free** ([l4_cap_idx_t](#) cap) [L4_NOTHROW](#)
Return capability index to capability allocator.
- void **l4re_util_cap_free_um** ([l4_cap_idx_t](#) cap) [L4_NOTHROW](#)
Return capability index to capability allocator, and unmaps the object.
- long **l4re_util_cap_last** (void) [L4_NOTHROW](#)
Return last capability index the allocator can return.

13.9.12.1 Detailed Description

Capability allocator C interface.

13.9.12.2 Function Documentation

13.9.12.2.1 l4re_util_cap_last()

```
long l4re_util_cap_last (
    void )
```

Return last capability index the allocator can return.

Returns

last/biggest capability index the allocator can return

References [L4_END_DECLS](#), and [L4_NOTHROW](#).

13.9.13 Kumem allocator utility

Kumem allocator utility C interface.

Collaboration diagram for Kumem allocator utility:



Kumem allocator utility C interface.

13.9.14 Video API

Collaboration diagram for Video API:



Data Structures

- struct [l4re_video_color_component_t](#)
Color component structure.
- struct [l4re_video_pixel_info_t](#)
Pixel_info structure.
- struct [l4re_video_goos_info_t](#)
Goos information structure.
- struct [l4re_video_view_info_t](#)
View information structure.
- struct [l4re_video_view_t](#)
C representation of a goos view.

Typedefs

- typedef struct [l4re_video_color_component_t](#) [l4re_video_color_component_t](#)
Color component structure.
- typedef struct [l4re_video_pixel_info_t](#) [l4re_video_pixel_info_t](#)
Pixel_info structure.
- typedef [l4_cap_idx_t](#) [l4re_video_goos_t](#)
Goos object type.
- typedef struct [l4re_video_view_info_t](#) [l4re_video_view_info_t](#)
View information structure.
- typedef struct [l4re_video_view_t](#) [l4re_video_view_t](#)
C representation of a goos view.

Enumerations

- enum [l4re_video_goos_info_flags_t](#) { [F_l4re_video_goos_auto_refresh](#) = 0x01 , [F_l4re_video_goos_pointer](#) = 0x02 , [F_l4re_video_goos_dynamic_views](#) = 0x04 , [F_l4re_video_goos_dynamic_buffers](#) = 0x08 }
Flags of information on the goos.
- enum [l4re_video_view_info_flags_t](#) {
[F_l4re_video_view_none](#) = 0x00 , [F_l4re_video_view_set_buffer](#) = 0x01 , [F_l4re_video_view_set_buffer_offset](#) = 0x02 , [F_l4re_video_view_set_bytes_per_line](#) = 0x04 ,
[F_l4re_video_view_set_pixel](#) = 0x08 , [F_l4re_video_view_set_position](#) = 0x10 , [F_l4re_video_view_dyn_allocated](#) = 0x20 , [F_l4re_video_view_set_background](#) = 0x40 ,
[F_l4re_video_view_set_flags](#) = 0x80 , [F_l4re_video_view_fully_dynamic](#) , [F_l4re_video_view_above](#) = 0x01000 , [F_l4re_video_view_flags_mask](#) = 0xff000 }
Flags of information on a view.

Functions

- `L4_BEGIN_DECLS` `int l4re_video_goos_info (l4re_video_goos_t goos, l4re_video_goos_info_t *ginfo)`
`L4_NOTHROW`
Get information on a goos.
- `int l4re_video_goos_refresh (l4re_video_goos_t goos, int x, int y, int w, int h) L4_NOTHROW`
Flush a rectangle of pixels of the goos screen.
- `int l4re_video_goos_create_buffer (l4re_video_goos_t goos, unsigned long size, l4_cap_idx_t buffer)`
`L4_NOTHROW`
Create a new buffer (memory buffer) for pixel data.
- `int l4re_video_goos_delete_buffer (l4re_video_goos_t goos, unsigned idx) L4_NOTHROW`
Delete a pixel buffer.
- `int l4re_video_goos_get_static_buffer (l4re_video_goos_t goos, unsigned idx, l4_cap_idx_t buffer)`
`L4_NOTHROW`
Get the data-space capability of the static pixel buffer.
- `int l4re_video_goos_create_view (l4re_video_goos_t goos, l4re_video_view_t *view) L4_NOTHROW`
Create a new view (.
- `int l4re_video_goos_delete_view (l4re_video_goos_t goos, l4re_video_view_t *view) L4_NOTHROW`
Delete a view.
- `int l4re_video_goos_get_view (l4re_video_goos_t goos, unsigned idx, l4re_video_view_t *view)`
`L4_NOTHROW`
Get a view for the given index.
- `L4_BEGIN_DECLS` `int l4re_video_view_refresh (l4re_video_view_t *view, int x, int y, int w, int h)`
`L4_NOTHROW`
Flush the given rectangle of pixels of the given view.
- `int l4re_video_view_get_info (l4re_video_view_t *view, l4re_video_view_info_t *info) L4_NOTHROW`
Retrieve information about the given view.
- `int l4re_video_view_set_info (l4re_video_view_t *view, l4re_video_view_info_t *info) L4_NOTHROW`
Set properties of the view.
- `int l4re_video_view_set_viewport (l4re_video_view_t *view, int x, int y, int w, int h, unsigned long bofs)`
`L4_NOTHROW`
Set the viewport parameters of a view.
- `int l4re_video_view_stack (l4re_video_view_t *view, l4re_video_view_t *pivot, int behind) L4_NOTHROW`
Change the stacking order in the stack of visible views.

13.9.14.1 Detailed Description

13.9.14.2 Typedef Documentation

13.9.14.2.1 l4re_video_view_t

```
typedef struct l4re_video_view_t l4re_video_view_t
```

C representation of a goos view.

A view is a visible rectangle that provides a view to the contents of a buffer (frame buffer) memory object and is placed on a real screen.

13.9.14.3 Enumeration Type Documentation

13.9.14.3.1 l4re_video_goos_info_flags_t

```
enum l4re_video_goos_info_flags_t
```

Flags of information on the goos.

Enumerator

| | |
|-----------------------------------|--|
| F_l4re_video_goos_auto_refresh | The graphics display is automatically refreshed. |
| F_l4re_video_goos_pointer | We have a mouse pointer. |
| F_l4re_video_goos_dynamic_views | Supports dynamically allocated views. |
| F_l4re_video_goos_dynamic_buffers | Supports dynamically allocated buffers. |

Definition at line 29 of file [goos.h](#).

13.9.14.3.2 l4re_video_view_info_flags_t

```
enum l4re_video_view_info_flags_t
```

Flags of information on a view.

Enumerator

| | |
|--------------------------------------|---|
| F_l4re_video_view_none | everything for this view is static (the VESA-FB case) |
| F_l4re_video_view_set_buffer | buffer object for this view can be changed |
| F_l4re_video_view_set_buffer_offset | buffer offset can be set |
| F_l4re_video_view_set_bytes_per_line | bytes per line can be set |
| F_l4re_video_view_set_pixel | pixel type can be set |
| F_l4re_video_view_set_position | position on screen can be set |
| F_l4re_video_view_dyn_allocated | View is dynamically allocated. |
| F_l4re_video_view_set_background | Set view as background for session. |
| F_l4re_video_view_set_flags | Set view property flags. |
| F_l4re_video_view_above | Flag the view as stay on top. |
| F_l4re_video_view_flags_mask | Mask containing all possible property flags. |

Definition at line 23 of file [view.h](#).

13.9.14.4 Function Documentation

13.9.14.4.1 l4re_video_goos_create_buffer()

```
int l4re_video_goos_create_buffer (
    l4re_video_goos_t goos,
```



```
unsigned long size,  
l4_cap_idx_t buffer)
```

Create a new buffer (memory buffer) for pixel data.

Parameters

| | |
|---------------|---|
| <i>goos</i> | the target object for the operation. |
| <i>size</i> | the size in bytes for the pixel buffer. |
| <i>buffer</i> | a capability index to receive the data-space capability for the buffer. |

Returns

≥ 0 : The index of the created buffer (used to assign views and for deletion). < 0 : on error

References [L4_CV](#), and [L4_NOTHROW](#).

13.9.14.4.2 l4re_video_goos_create_view()

```
int l4re_video_goos_create_view (  
    l4re_video_goos_t goos,  
    l4re_video_view_t * view)
```

Create a new view (.

See also

[l4re_video_view_t](#))

Parameters

| | | |
|-----|-------------|--|
| | <i>goos</i> | the goos session to use. |
| out | <i>view</i> | structure initialized to the new view. |

References [L4_CV](#), and [L4_NOTHROW](#).

13.9.14.4.3 l4re_video_goos_delete_buffer()

```
int l4re_video_goos_delete_buffer (  
    l4re_video_goos_t goos,  
    unsigned idx)
```

Delete a pixel buffer.

Parameters

| | |
|-------------|-------------------------|
| <i>goos</i> | the target goos object. |
|-------------|-------------------------|

| | |
|------------|---|
| <i>idx</i> | the buffer index of the buffer to delete (the return value of l4re_video_goos_create_buffer()) |
|------------|---|

References [L4_CV](#), and [L4_NOTHROW](#).

13.9.14.4.4 l4re_video_goos_delete_view()

```
int l4re_video_goos_delete_view (
    l4re_video_goos_t goos,
    l4re_video_view_t * view)
```

Delete a view.

Parameters

| | |
|-------------|---|
| <i>goos</i> | the goos session to use. |
| <i>view</i> | the view to delete, the given data-structure is invalid afterwards. |

References [L4_CV](#), and [L4_NOTHROW](#).

13.9.14.4.5 l4re_video_goos_get_static_buffer()

```
int l4re_video_goos_get_static_buffer (
    l4re_video_goos_t goos,
    unsigned idx,
    l4_cap_idx_t buffer)
```

Get the data-space capability of the static pixel buffer.

Parameters

| | |
|---------------|--|
| <i>goos</i> | The target goos object. |
| <i>idx</i> | Index of the static buffer. |
| <i>buffer</i> | A capability index to receive the data-space capability. |

This function allows access to static, preexisting pixel buffers. Such static buffers exist for static configurations, such as the VESA framebuffer.

References [L4_CV](#), and [L4_NOTHROW](#).

13.9.14.4.6 l4re_video_goos_get_view()

```
int l4re_video_goos_get_view (
    l4re_video_goos_t goos,
    unsigned idx,
    l4re_video_view_t * view)
```

Get a view for the given index.

Parameters

| | | |
|-----|-------------|---|
| | <i>goos</i> | the target goos session. |
| | <i>idx</i> | the index of the view to retrieve. |
| out | <i>view</i> | structure initialized to the view with the given index. |

This function allows to access static views as provided by the VESA framebuffer (the monitor). However, it also allows to access dynamic views created with [l4re_video_goos_create_view\(\)](#).

References [L4_END_DECLS](#), and [L4_NOTHROW](#).

13.9.14.4.7 l4re_video_goos_info()

```
L4_BEGIN_DECLS int l4re_video_goos_info (
    l4re_video_goos_t goos,
    l4re_video_goos_info_t * ginfo)
```

Get information on a goos.

Parameters

| | | |
|-----|--------------|--|
| | <i>goos</i> | Goos object |
| out | <i>ginfo</i> | Pointer to goos information structure. |

Returns

0 for success, <0 on error

- [-L4_ENODEV](#)
- IPC errors

References [L4_CV](#), and [L4_NOTHROW](#).

13.9.14.4.8 l4re_video_goos_refresh()

```
int l4re_video_goos_refresh (
    l4re_video_goos_t goos,
    int x,
    int y,
    int w,
    int h)
```

Flush a rectangle of pixels of the goos screen.

Parameters

| | |
|-------------|--|
| <i>goos</i> | the target object of the operation. |
| <i>x</i> | the x-coordinate of the upper left corner of the rectangle |
| <i>y</i> | the y-coordinate of the upper left corner of the rectangle |
| <i>w</i> | the width of the rectangle to be flushed |
| <i>h</i> | the height of the rectangle |

References [L4_CV](#), and [L4_NOTHROW](#).

13.9.14.4.9 l4re_video_view_get_info()

```
int l4re_video_view_get_info (
    l4re_video_view_t * view,
    l4re_video_view_info_t * info)
```

Retrieve information about the given *view*.

Parameters

| | | |
|-----|-------------|--|
| | <i>view</i> | the target view for the operation. |
| out | <i>info</i> | a buffer receiving the information about the view. |

References [L4_CV](#), and [L4_NOTHROW](#).

13.9.14.4.10 l4re_video_view_refresh()

```
L4_BEGIN_DECLS int l4re_video_view_refresh (
    l4re_video_view_t * view,
    int x,
    int y,
    int w,
    int h)
```

Flush the given rectangle of pixels of the given *view*.

Parameters

| | |
|-------------|---------------------------------------|
| <i>view</i> | the target view of the operation. |
| <i>x</i> | x-coordinate of the upper left corner |
| <i>y</i> | y-coordinate of the upper left corner |
| <i>w</i> | the width of the rectangle |
| <i>h</i> | the height of the rectangle |

References [L4_CV](#), and [L4_NOTHROW](#).

13.9.14.4.11 l4re_video_view_set_info()

```
int l4re_video_view_set_info (
    l4re_video_view_t * view,
    l4re_video_view_info_t * info)
```

Set properties of the view.

Parameters

| | |
|-------------|---------------------------------------|
| <i>view</i> | the target view of the operation. |
| <i>info</i> | the parameters to be set on the view. |

Which parameters can be manipulated on a given view can be figured out with [l4re_video_view_get_info\(\)](#) and this depends on the concrete instance the view object.

References [L4_CV](#), and [L4_NOTHROW](#).

13.9.14.4.12 l4re_video_view_set_viewport()

```
int l4re_video_view_set_viewport (
    l4re_video_view_t * view,
    int x,
    int y,
    int w,
    int h,
    unsigned long bofs)
```

Set the viewport parameters of a view.

Parameters

| | |
|-------------|--|
| <i>view</i> | the target view of the operation. |
| <i>x</i> | the x-coordinate of the upper left corner on the screen. |
| <i>y</i> | the y-coordinate of the upper left corner on the screen. |
| <i>w</i> | the width of the view. |

| | |
|-------------|--|
| <i>h</i> | the height of the view. |
| <i>bofs</i> | the offset (in bytes) of the upper left pixel in the memory buffer |

This function is a convenience wrapper for [l4re_video_view_set_info\(\)](#), just setting the often changed parameters of a dynamic view. With this function a view can be placed on the real screen and at the same time on its backing buffer.

References [L4_CV](#), and [L4_NOTHROW](#).

13.9.14.4.13 l4re_video_view_stack()

```
int l4re_video_view_stack (
    l4re_video_view_t * view,
    l4re_video_view_t * pivot,
    int behind)
```

Change the stacking order in the stack of visible views.

Parameters

| | |
|---------------|--|
| <i>view</i> | the target view for the operation. |
| <i>pivot</i> | the neighbor view, relative to which <i>view</i> shall be stacked. a NULL value allows top (<i>behind</i> = 1) and bottom (<i>behind</i> = 0) placement of the view. |
| <i>behind</i> | describes the placement of the view relative to the <i>pivot</i> view. |

References [L4_END_DECLS](#), and [L4_NOTHROW](#).

13.9.15 Initial Environment

C interface of the initial environment that is provided to an [L4](#) task.

Collaboration diagram for Initial Environment:



Data Structures

- struct [l4re_env_cap_entry_t](#)

Entry in the [L4Re](#) environment array for the named initial objects.

Typedefs

- typedef struct l4re_env_cap_entry_t **l4re_env_cap_entry_t**
Entry in the [L4Re](#) environment array for the named initial objects.

Functions

- [l4re_env_t](#) * [l4re_env](#) (void) [L4_NOTHROW](#)
Get [L4Re](#) initial environment.
- [l4_kernel_info_t](#) const * [l4re_kip](#) (void) [L4_NOTHROW](#)
Get Kernel Info Page.
- [l4_cap_idx_t](#) [l4re_env_get_cap](#) (char const *name) [L4_NOTHROW](#)
Get the capability selector for the object named name.
- [l4_cap_idx_t](#) [l4re_env_get_cap_e](#) (char const *name, [l4re_env_t](#) const *e) [L4_NOTHROW](#)
Get the capability selector for the object named name.
- [l4re_env_cap_entry_t](#) const * [l4re_env_get_cap_l](#) (char const *name, unsigned l, [l4re_env_t](#) const *e) [L4_NOTHROW](#)
Get the full [l4re_env_cap_entry_t](#) for the object named name.

13.9.15.1 Detailed Description

C interface of the initial environment that is provided to an [L4](#) task.

Include File

```
#include <l4/re/env.h>
```

For an explanation of the default task capabilities see [l4_default_caps_t](#).

For the C++ interface refer to [L4Re::Env](#).

13.9.15.2 Function Documentation

13.9.15.2.1 l4re_env()

```
l4re\_env\_t * l4re\_env (  
    void ) [inline]
```

Get [L4Re](#) initial environment.

Returns

Pointer to [L4Re](#) initial environment.

Examples

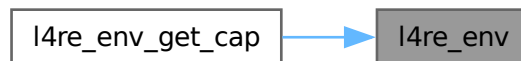
[examples/sys/aliens/main.c](#), [examples/sys/isr/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 195 of file [env.h](#).

References [L4_NOTHROW](#).

Referenced by [l4re_env_get_cap\(\)](#).

Here is the caller graph for this function:

**13.9.15.2.2 l4re_env_get_cap()**

```
l4_cap_idx_t l4re_env_get_cap (
    char const * name) [inline]
```

Get the capability selector for the object named *name*.

Parameters

| | |
|-------------|---|
| <i>name</i> | is the name of the object to lookup in the initial objects. |
|-------------|---|

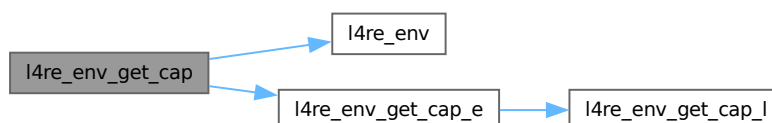
Returns

A valid capability selector if the object exists or an invalid capability selector if not ([l4_is_invalid_cap\(\)](#)).

Definition at line 234 of file [env.h](#).

References [L4_NOTHROW](#), [l4re_env\(\)](#), and [l4re_env_get_cap_e\(\)](#).

Here is the call graph for this function:



13.9.15.2.3 l4re_env_get_cap_e()

```
l4_cap_idx_t l4re_env_get_cap_e (  
    char const * name,  
    l4re_env_t const * e) [inline]
```

Get the capability selector for the object named *name*.

Parameters

| | |
|-------------|---|
| <i>name</i> | is the name of the object to lookup in the initial objects. |
| <i>e</i> | is the environment structure to use for the operation. |

Returns

A valid capability selector if the object exists or an invalid capability selector if not ([l4_is_invalid_cap\(\)](#)).

Definition at line 221 of file [env.h](#).

References [l4re_env_cap_entry_t::cap](#), [L4_INVALID_CAP](#), [L4_NOTHROW](#), and [l4re_env_get_cap_l\(\)](#).

Referenced by [l4re_env_get_cap\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.9.15.2.4 l4re_env_get_cap_l()

```
l4re_env_cap_entry_t const * l4re_env_get_cap_l (
    char const * name,
    unsigned l,
    l4re_env_t const * e) [inline]
```

Get the full [l4re_env_cap_entry_t](#) for the object named *name*.

Parameters

| | |
|-------------|--|
| <i>name</i> | is the name of the object to lookup in the initial objects. |
| <i>l</i> | is the length of the name string, thus <i>name</i> might not be zero terminated. |
| <i>e</i> | is the environment structure to use for the operation. |

Returns

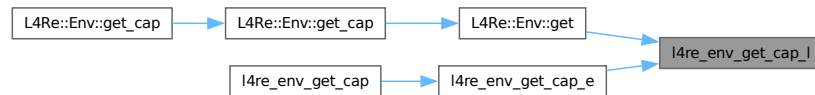
A pointer to an [l4re_env_cap_entry_t](#) if the object exists or NULL if not.

Definition at line 203 of file [env.h](#).

References [l4re_env_cap_entry_t::flags](#), [L4_NOTHROW](#), and [l4re_env_cap_entry_t::name](#).

Referenced by [L4Re::Env::get\(\)](#), and [l4re_env_get_cap_e\(\)](#).

Here is the caller graph for this function:



13.9.15.2.5 l4re_kip()

```
l4_kernel_info_t const * l4re_kip (
    void ) [inline]
```

Get Kernel Info Page.

Returns

Pointer to Kernel Info Page (KIP) structure.

Examples

[examples/libs/shmc/prodcons.c](#), and [examples/sys/aliens/main.c](#).

Definition at line 199 of file [env.h](#).

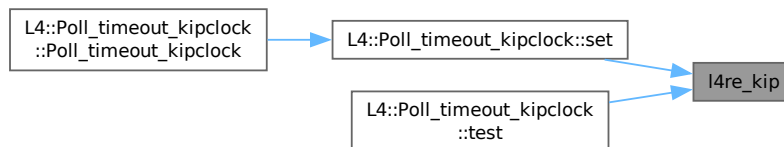
References [l4_kip\(\)](#), and [L4_NOTHROW](#).

Referenced by [L4::Poll_timeout_kipclock::set\(\)](#), and [L4::Poll_timeout_kipclock::test\(\)](#).

Here is the call graph for this function:



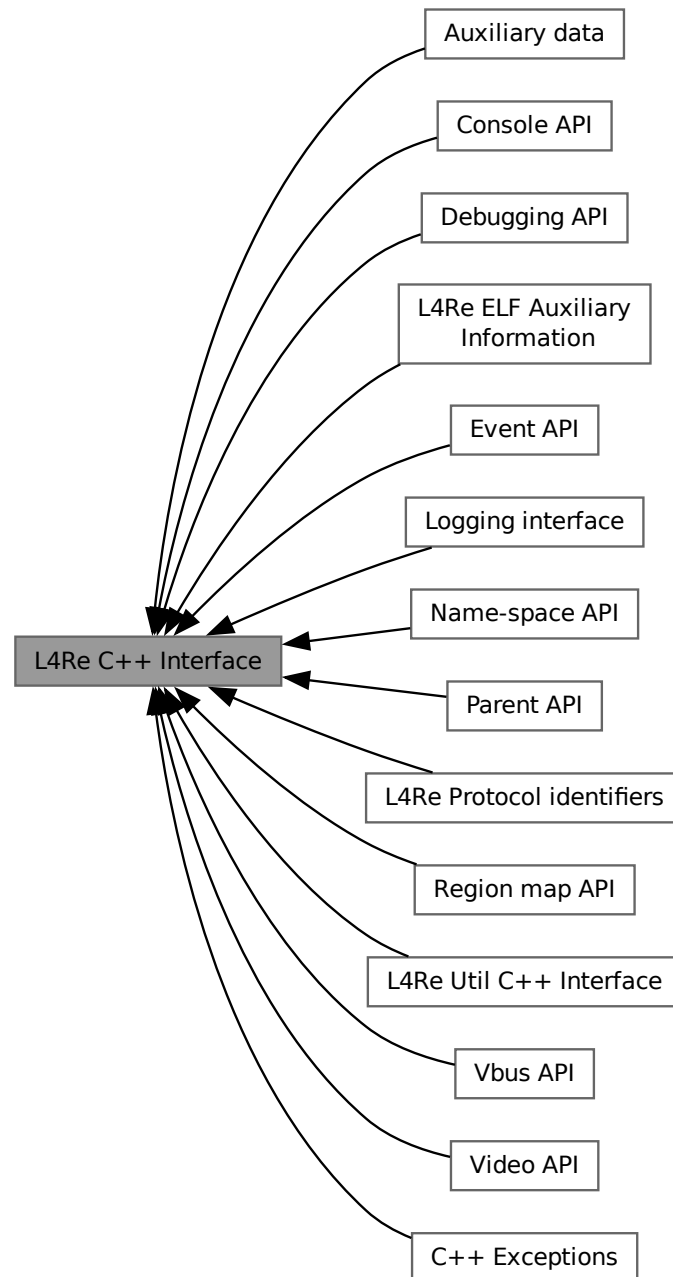
Here is the caller graph for this function:



13.10 L4Re C++ Interface

Documentation of the [L4](#) Runtime Environment C++ API.

Collaboration diagram for L4Re C++ Interface:



Topics

- L4Re Util C++ Interface [587](#)
Documentation of the [L4](#) Runtime Environment utility functionality in C++.
- Console API [592](#)

| | | |
|---|--|-----|
| • | <i>Console interface.</i> | |
| • | Debugging API | 593 |
| | <i>Debugging Interface.</i> | |
| • | L4Re ELF Auxiliary Information | 593 |
| | <i>API for embedding auxiliary information into binary programs.</i> | |
| • | Event API | 596 |
| | <i>Event API.</i> | |
| • | Auxiliary data | 596 |
| • | Logging interface | 597 |
| | <i>Interface for log output.</i> | |
| • | Name-space API | 598 |
| | <i>API for name spaces that store capabilities.</i> | |
| • | Parent API | 598 |
| | <i>Parent interface.</i> | |
| • | L4Re Protocol identifiers | 599 |
| | <i>Fix L4Re Protocol Constants.</i> | |
| • | Region map API | 600 |
| | <i>Virtual address-space management.</i> | |
| • | Video API | 601 |
| | <i>API for framebuffer based graphics.</i> | |
| • | C++ Exceptions | 602 |
| • | Vbus API | 603 |
| | <i>C++ interface of the Vbus API.</i> | |

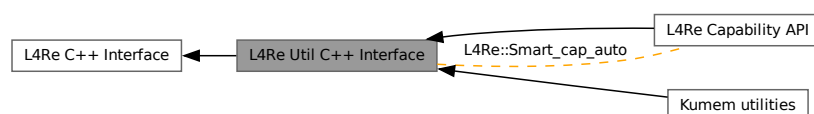
13.10.1 Detailed Description

Documentation of the [L4](#) Runtime Environment C++ API.

13.10.2 L4Re Util C++ Interface

Documentation of the [L4](#) Runtime Environment utility functionality in C++.

Collaboration diagram for L4Re Util C++ Interface:



Topics

- L4Re Capability API 588
- Kmem•utilities 591

Data Structures

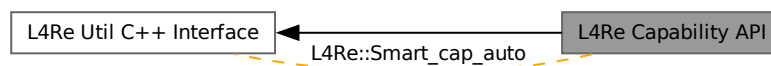
- class [L4Re::Smart_cap_auto< Unmap_flags >](#)
Helper for [Unique_cap](#) and [Unique_del_cap](#).
- class [L4Re::Util::Cap_alloc_base](#)
Capability allocator.
- class [L4Re::Util::Br_manager](#)
Buffer-register (BR) manager for [L4::Server](#).
- class [L4Re::Util::Counting_cap_alloc< COUNTERTYPE, Dbg >](#)
Internal reference-counting cap allocator.
- class [L4Re::Util::Event_buffer_t< PAYLOAD >](#)
Event_buffer utility class.
- class [L4Re::Util::Event_buffer_consumer_t< PAYLOAD >](#)
An event buffer consumer.
- class [L4Re::Util::Vcon_svr< SVR >](#)
[Console](#) server template class.
- class [L4Re::Util::Video::Goos_svr](#)
Goos server class.

13.10.2.1 Detailed Description

Documentation of the [L4](#) Runtime Environment utility functionality in C++.

13.10.2.2 L4Re Capability API

Collaboration diagram for L4Re Capability API:



Data Structures

- class [L4Re::Cap_alloc](#)
Capability allocator interface.
- class [L4Re::Smart_cap_auto< Unmap_flags >](#)
Helper for [Unique_cap](#) and [Unique_del_cap](#).
- class [L4Re::Smart_count_cap< Unmap_flags >](#)
Helper for [Ref_cap](#) and [Ref_del_cap](#).
- class [L4Re::Util::Smart_cap_auto< Unmap_flags >](#)
Helper for [Unique_cap](#) and [Unique_del_cap](#).
- class [L4Re::Util::Smart_count_cap< Unmap_flags >](#)
Helper for [Ref_cap](#) and [Ref_del_cap](#).
- struct [L4Re::Util::Ref_cap< T >](#)
Automatic capability that implements automatic free and unmap of the capability selector.
- struct [L4Re::Util::Ref_del_cap< T >](#)
Automatic capability that implements automatic free and unmap+delete of the capability selector.

Functions

- `template<typename T>
Ref_cap< T >::Cap L4Re::Util::make_ref_cap ()`
Allocate a capability slot and wrap it in a [Ref_cap](#).
- `template<typename T>
Ref_del_cap< T >::Cap L4Re::Util::make_ref_del_cap ()`
Allocate a capability slot and wrap it in a [Ref_del_cap](#).

Variables

- `_Cap_alloc L4Re::Util::cap_alloc`
Capability allocator.
- `Def_reply_cap_alloc L4Re::Util::reply_cap_alloc`
Reply capability allocator.

13.10.2.2.1 Detailed Description

13.10.2.2.2 Function Documentation

13.10.2.2.2.1 [make_ref_cap\(\)](#)

```
template<typename T>
Ref\_cap< T >::Cap L4Re::Util::make\_ref\_cap ()
```

Allocate a capability slot and wrap it in a [Ref_cap](#).

Template Parameters

| | |
|----------|--|
| <i>T</i> | Type of capability the slot is used for. |
|----------|--|

Definition at line 237 of file [cap_alloc](#).

References [cap_alloc](#).

13.10.2.2.2.2 make_ref_del_cap()

```
template<typename T>
Ref_del_cap< T >::Cap L4Re::Util::make_ref_del_cap ()
```

Allocate a capability slot and wrap it in a [Ref_del_cap](#).

Template Parameters

| | |
|----------|--|
| <i>T</i> | Type of capability the slot is used for. |
|----------|--|

Definition at line 246 of file [cap_alloc](#).

References [cap_alloc](#).

13.10.2.2.3 Variable Documentation

13.10.2.2.3.1 cap_alloc

```
_Cap_alloc L4Re::Util::cap_alloc [extern]
```

Capability allocator.

This is the instance of the capability allocator that is used by usual applications.

The capability allocator uses the [Counting_cap_alloc](#), a reference-counting thread-safe capability allocator, that keeps a reference counter for each managed capability selector.

Examples

[examples/libs/l4re/c++/mem_alloc/ma+rm.cc](#), [examples/libs/l4re/c++/shared_ds/ds_clnt.cc](#), [examples/libs/l4re/c++/shared_ds/ds_d](#) and [examples/libs/l4re/streammap/client.cc](#).

Referenced by [L4Re::Util::Br_manager::alloc_buffer_demand\(\)](#), [L4Re::Util::Smart_count_cap< Unmap_flags >::copy\(\)](#), [L4Re::Util::Smart_cap_auto< Unmap_flags >::free\(\)](#), [L4Re::Util::Smart_count_cap< Unmap_flags >::free\(\)](#), [L4Re::Util::Event_t< Default_event_payload >::init\(\)](#), [L4Re::Util::Event_t< Default_event_payload >::init_poll\(\)](#), [make_ref_cap\(\)](#), [make_ref_del_cap\(\)](#), [make_shared_cap\(\)](#), [make_shared_del_cap\(\)](#), [make_unique_cap\(\)](#), [make_unique_del_cap\(\)](#), [L4Re::Util::Br_manager::realloc_rcv_cap\(\)](#), and [L4Re::Util::Object_registry::unregister_obj\(\)](#).

13.10.2.2.3.2 reply_cap_alloc

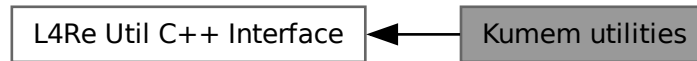
```
Def_reply_cap_alloc L4Re::Util::reply_cap_alloc [extern]
```

Reply capability allocator.

This is the default reply capability allocator that is used by usual applications.

13.10.2.3 Kumem utilities

Collaboration diagram for Kumem utilities:



Functions

- `int L4Re::Util::kumem_alloc (l4_addr_t *mem, unsigned pages_order, L4::Cap< L4::Task > task=L4Re::Env::env() ->task(), L4::Cap< L4Re::Rm > rm=L4Re::Env::env() ->rm()) noexcept`
Allocate state area.

13.10.2.3.1 Detailed Description

13.10.2.3.2 Function Documentation

13.10.2.3.2.1 kumem_alloc()

```

int L4Re::Util::kumem_alloc (
    l4_addr_t * mem,
    unsigned pages_order,
    L4::Cap< L4::Task > task = L4Re::Env::env() ->task(),
    L4::Cap< L4Re::Rm > rm = L4Re::Env::env() ->rm())    [noexcept]
  
```

Allocate state area.

Parameters

| | | |
|-----|--------------------|--|
| out | <i>mem</i> | Pointer to memory that has been allocated. |
| | <i>pages_order</i> | Size to allocate, in log2 pages. |
| | <i>task</i> | Task to use for allocation. |
| | <i>rm</i> | Region manager to use for allocation. |

Return values

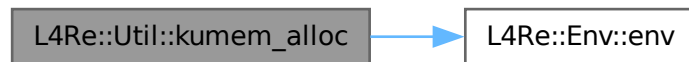
| | |
|----|-----------------------|
| 0 | for success |
| <0 | error code on failure |

Note

The amount of kernel-user memory that can be allocated at once is limited by the used kernel implementation. The minimum allocatable amount is one page. A portable implementation should not depend on allocations greater than 16KiB to succeed.

References [L4Re::Env::env\(\)](#).

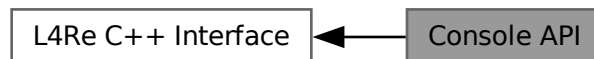
Here is the call graph for this function:



13.10.3 Console API

[Console](#) interface.

Collaboration diagram for Console API:



Data Structures

- class [L4Re::Console](#)
Console class.

13.10.3.1 Detailed Description

[Console](#) interface.

13.10.4 Debugging API

Debugging Interface.

Collaboration diagram for Debugging API:



Data Structures

- class [L4Re::Debug_obj](#)
Debug interface.

13.10.4.1 Detailed Description

Debugging Interface.

The debugging interface can be provided to retrieve, or log debugging information for an object. Each class may realize the debug interface to provide debugging functionality. For example, the region map objects provide a facility to dump the currently established memory regions.

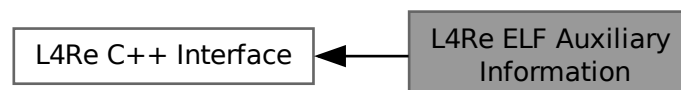
See also

[L4::Debug_obj](#) for more information.

13.10.5 L4Re ELF Auxiliary Information

API for embedding auxiliary information into binary programs.

Collaboration diagram for L4Re ELF Auxiliary Information:



Data Structures

- struct `l4re_elf_aux_t`
Generic header for each auxiliary vector element.
- struct `l4re_elf_aux_vma_t`
Auxiliary vector element for a reserved virtual memory area.
- struct `l4re_elf_aux_mword_t`
Auxiliary vector element for a single unsigned data word.

Macros

- `#define L4RE_ELF_AUX_ELEM const __attribute__((used, section(".ro.l4re_elf_aux"), aligned(sizeof(l4_umword_t))))`
Define an auxiliary vector element.
- `#define L4RE_ELF_AUX_ELEM_T(type, id, tag, val...)`
Define an auxiliary vector element.

Typedefs

- `typedef struct l4re_elf_aux_t l4re_elf_aux_t`
Generic header for each auxiliary vector element.
- `typedef struct l4re_elf_aux_vma_t l4re_elf_aux_vma_t`
Auxiliary vector element for a reserved virtual memory area.
- `typedef struct l4re_elf_aux_mword_t l4re_elf_aux_mword_t`
Auxiliary vector element for a single unsigned data word.

Enumerations

- enum {
`L4RE_ELF_AUX_T_NONE = 0 , L4RE_ELF_AUX_T_VMA , L4RE_ELF_AUX_T_STACK_SIZE ,`
`L4RE_ELF_AUX_T_STACK_ADDR ,`
`L4RE_ELF_AUX_T_KIP_ADDR , L4RE_ELF_AUX_T_EX_REGS_FLAGS }`

13.10.5.1 Detailed Description

API for embedding auxiliary information into binary programs.

This API allows information for the binary loader to be embedded into a binary application. This information can be reserved areas in the virtual memory of an application and things such as the stack size to be allocated for the first application thread.

13.10.5.2 Macro Definition Documentation

13.10.5.2.1 L4RE_ELF_AUX_ELEM

```
#define L4RE_ELF_AUX_ELEM const __attribute__((used, section(".ro.l4re_elf_aux"), aligned(sizeof(l4_umword_t))))
```

Define an auxiliary vector element.

This is the generic method for defining auxiliary vector elements. A more convenient way is to use `L4RE_ELF_AUX_ELEM_T`.

Usage:

```
L4RE_ELF_AUX_ELEM l4re_elf_aux_vma_t decl_name =  
{ L4RE_ELF_AUX_T_VMA, sizeof(l4re_elf_aux_vma_t), 0x2000, 0x4000 };
```

Definition at line 41 of file `elf_aux.h`.

13.10.5.2.2 L4RE_ELF_AUX_ELEM_T

```
#define L4RE_ELF_AUX_ELEM_T(  
    type,  
    id,  
    tag,  
    val...)
```

Value:

```
static L4RE_ELF_AUX_ELEM type id = {tag, sizeof(type), val}
```

Define an auxiliary vector element.

Parameters

| | |
|-------------|--|
| <i>type</i> | is the data type for the element (e.g., l4re_elf_aux_vma_t) |
| <i>id</i> | is the identifier (variable name) for the declaration (the variable is defined with <code>static</code> storage class) |
| <i>tag</i> | is the tag value for the element e.g., L4RE_ELF_AUX_T_VMA |
| <i>val</i> | are the values to be set in the descriptor |

Usage:

```
L4RE_ELF_AUX_ELEM_T(l4re_elf_aux_vma_t, decl_name, L4RE_ELF_AUX_T_VMA, 0x2000, 0x4000 );
```

Definition at line 56 of file [elf_aux.h](#).

13.10.5.3 Enumeration Type Documentation

13.10.5.3.1 anonymous enum

```
anonymous enum
```

Enumerator

| | |
|------------------------------|---|
| L4RE_ELF_AUX_T_NONE | Tag for an invalid element in the auxiliary vector. |
| L4RE_ELF_AUX_T_VMA | Tag for descriptor for a reserved virtual memory area. |
| L4RE_ELF_AUX_T_STACK_SIZE | Tag for descriptor that defines the stack size for the first application thread. |
| L4RE_ELF_AUX_T_STACK_ADDR | Tag for descriptor that defines the stack address for the first application thread. |
| L4RE_ELF_AUX_T_KIP_ADDR | Tag for descriptor that defines the KIP address for the binary's address space. |
| L4RE_ELF_AUX_T_EX_REGS_FLAGS | Tag for descriptor to override <code>ex_regs()</code> flags. |

Definition at line 59 of file [elf_aux.h](#).

13.10.6 Event API

Event API.

Collaboration diagram for Event API:



Data Structures

- class [L4Re::Event](#)
Event class.
- struct [L4Re::Default_event_payload](#)
Default event stream payload.
- class [L4Re::Event_buffer_t< PAYLOAD >](#)
Event buffer class.

13.10.6.1 Detailed Description

Event API.

On top of a shared [L4Re::Dataspace](#) (and optionally using [L4::Triggerable](#)), the event API implements asynchronous event transmission from an event provider (server) to an event receiver (client). Events are put into an [Event_buffer_t](#) residing on the shared [L4Re::Dataspace](#).

This interface is usually not used directly. Instead use [L4Re::Util::Event_t](#) for clients. An example server portion is implemented in [L4Re::Util::Event_svr](#).

This interface is usually used with [L4Re::Default_event_payload](#) which delivers HID events modeled on the Linux evdev API, and the interface's methods allow further querying of information about the HID event streams.

13.10.7 Auxiliary data

Collaboration diagram for Auxiliary data:



Data Structures

- struct [l4re_aux_t](#)
Auxiliary descriptor.

Typedefs

- typedef struct l4re_aux_t **l4re_aux_t**
Auxiliary descriptor.

Enumerations

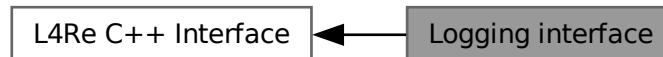
- enum [l4re_aux_ldr_flags_t](#)
Flags for program loading.

13.10.7.1 Detailed Description

13.10.8 Logging interface

Interface for log output.

Collaboration diagram for Logging interface:



Data Structures

- class [L4Re::Log](#)
Log interface class.

13.10.8.1 Detailed Description

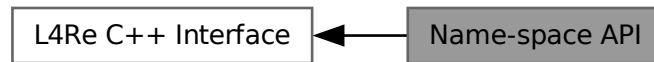
Interface for log output.

The logging interface provides a facility sending log output. One purpose of the interface is to serialize the output and provide the possibility to tag output sent to a specific log object.

13.10.9 Name-space API

API for name spaces that store capabilities.

Collaboration diagram for Name-space API:



Data Structures

- class [L4Re::Namespace](#)
Name-space interface.

13.10.9.1 Detailed Description

API for name spaces that store capabilities.

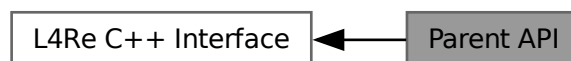
This is a basic abstraction for managing a mapping from human-readable names to capabilities. In particular, a name can also be mapped to a capability that refers to another name space object. By this means name spaces can be constructed hierarchically.

Name spaces play a central role in [L4Re](#), because the implementation of the name space objects determines the policy which capabilities (which objects) are accessible to a client of a name space.

13.10.10 Parent API

[Parent](#) interface.

Collaboration diagram for Parent API:



Data Structures

- class [L4Re::Parent](#)
Parent interface.

13.10.10.1 Detailed Description

[Parent](#) interface.

The parent interface provides means for an [L4](#) task to signal changes in its execution state. The main purpose is to signal program termination to the program that started it, so that its resources can be reclaimed. In a typical [L4Re](#) system, this program will be Moe or Ned.

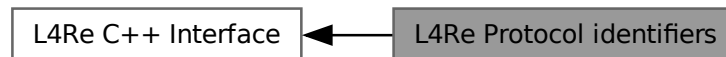
See also

[L4Re::Parent](#) for information about the concrete interface.

13.10.11 L4Re Protocol identifiers

Fix [L4Re](#) Protocol Constants.

Collaboration diagram for L4Re Protocol identifiers:



Enumerations

- enum [L4Re::Dataspace_::Opcodes](#)
Data-space communication-protocol opcodes.
- enum [L4Re::Event_::Opcodes](#)
Event communication-protocol opcodes.
- enum [L4Re::Inhibitor_::Opcodes](#)
Inhibitor communication-protocol opcodes.
- enum [L4Re::Log_::Opcodes](#)
Logging-service communication-protocol opcodes.
- enum [L4Re::Mem_alloc_::Opcodes](#)
Memory-allocator communication-protocol opcodes.
- enum [L4Re::Namespace_::Opcodes](#)
Name-space communication-protocol opcodes.
- enum [L4Re::Parent_::Opcodes](#)
Parent communication-protocol opcodes.
- enum [L4re_protocols](#) {
[L4RE_PROTO_DATASPACE](#) = 0x4000 , [L4RE_PROTO_NAMESPACE](#) , [L4RE_PROTO_PARENT](#) ,
[L4RE_PROTO_GOOS](#) ,
[L4RE_PROTO_RSVD_1](#) , [L4RE_PROTO_RM](#) , [L4RE_PROTO_EVENT](#) , [L4RE_PROTO_INHIBITOR](#) ,
[L4RE_PROTO_DMA_SPACE](#) , [L4RE_PROTO_MMIO_SPACE](#) , [L4RE_PROTO_ITAS](#) , [L4RE_PROTO_MEM_ALLOC](#)
 ,
[L4RE_PROTO_REMOTE_ACCESS](#) , [L4RE_PROTO_DEBUG](#) = ~0x7fffL }
Common L4Re Protocol Constants.
- enum [L4Re::Rm_::Opcodes](#)
Region-map communication-protocol opcodes.
- enum [L4Re::Video::Goos_::Opcodes](#)
Frame buffer communication-protocol opcodes.

13.10.11.1 Detailed Description

Fix [L4Re](#) Protocol Constants.

13.10.11.2 Enumeration Type Documentation

13.10.11.2.1 L4re_protocols

enum [L4re_protocols](#)

Common [L4Re](#) Protocol Constants.

Enumerator

| | |
|--------------------------|--|
| L4RE_PROTO_DATASPACE | ID for L4Re::Dataspace RPCs. |
| L4RE_PROTO_NAMESPACE | ID for L4Re::Namespace RPCs. |
| L4RE_PROTO_PARENT | ID for L4Re::Parent RPCs. |
| L4RE_PROTO_GOOS | ID for L4Re::Video::Goos RPCs. |
| L4RE_PROTO_RSVD_1 | Reserved ID. |
| L4RE_PROTO_RM | ID for L4Re::Rm RPCs. |
| L4RE_PROTO_EVENT | ID for L4Re::Event RPCs. |
| L4RE_PROTO_INHIBITOR | ID for L4Re::Inhibitor RPCs. |
| L4RE_PROTO_DMA_SPACE | ID for L4Re::Dma_space RPCs. |
| L4RE_PROTO_MMIO_SPACE | ID for L4Re::Mmio_space . |
| L4RE_PROTO_ITAS | ID for L4Re::Itas . |
| L4RE_PROTO_MEM_ALLOC | ID for L4Re::Mem_alloc . |
| L4RE_PROTO_REMOTE_ACCESS | ID for L4Re::Remote_access . |
| L4RE_PROTO_DEBUG | ID for debugging RPCs. |

Definition at line [24](#) of file [protocols.h](#).

13.10.12 Region map API

Virtual address-space management.

Collaboration diagram for Region map API:



Data Structures

- class [L4Re::Rm](#)
Region map.

13.10.12.1 Detailed Description

Virtual address-space management.

A region map object implements two protocols. The first protocol is the kernel page-fault protocol, to resolve page faults for threads running in an [L4](#) task. The second protocol is the region map protocol itself, which allows managing the virtual memory address space of an [L4](#) task.

There are two basic concepts provided by the region map abstraction:

- **Areas** are reserved ranges in the virtual memory address space.
- **Regions** are ranges that are backed by (part of) a dataspace, i.e. accessing them results in access to the physical memory the dataspace manages.

Note that regions may live outside of areas and that an area does not necessarily contain any region.

Areas can be reserved for special use or for attaching a dataspace at a later time. When attaching a dataspace, the user can instruct the region map to search for an appropriate range to attach to. Regions are skipped in this search since they already have dataspace attached to them, and, depending on [L4Re::Rm::F::In_area](#), areas are skipped because they are reserved. Amongst others, areas can be used to attach several dataspace inside a certain range of addresses without interference from other threads.

When a region map receives a page fault IPC, the region map will check if the faulting virtual address lies in a region. If yes, it will answer the page fault IPC with a mapping from the backing dataspace. If not, an error is returned.

Depending on the system type, an attached dataspace might or might not be mapped eagerly. MMU-based systems resort to lazy mapping while systems without MMU do eager mappings by default. The [L4Re::Rm::F::Eager_map](#) and [L4Re::Rm::F::No_eager_map](#) flags can be used to force the respective behaviour, independent of the underlying system. In case both flags are given, the [L4Re::Rm::F::No_eager_map](#) flag wins.

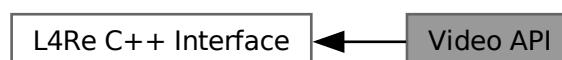
See also

[L4Re::Dataspace](#), [L4Re::Rm](#),
[Memory management - Data Spaces and the Region Map](#)

13.10.13 Video API

API for framebuffer based graphics.

Collaboration diagram for Video API:



Data Structures

- class [L4Re::Video::Color_component](#)
A color component.
- class [L4Re::Video::Pixel_info](#)
Pixel information.

13.10.13.1 Detailed Description

API for framebuffer based graphics.

Contains the basic APIs that abstract framebuffers and views into them for [L4Re](#) applications.

13.10.14 C++ Exceptions

Collaboration diagram for C++ Exceptions:



Files

- file [exceptions](#)
Base exceptions.

Data Structures

- class [L4::Exception_tracer](#)
Back-trace support for exceptions.
- class [L4::Base_exception](#)
Base class for all exceptions, thrown by the [L4Re](#) framework.
- class [L4::Runtime_error](#)
Exception for an abstract runtime error.
- class [L4::Out_of_memory](#)
Exception signalling insufficient memory.
- class [L4::Element_already_exists](#)
Exception for duplicate element insertions.
- class [L4::Unknown_error](#)
Exception for an unknown condition.
- class [L4::Element_not_found](#)
Exception for a failed lookup (element not found).
- class [L4::Invalid_capability](#)
Indicates that an invalid object was invoked.
- class [L4::Com_error](#)
Error conditions during IPC.
- class [L4::Bounds_error](#)
Access out of bounds.

Macros

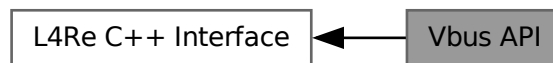
- `#define L4_CXX_EXCEPTION_BACKTRACE 20`
Number of instruction pointers in backtrace.

13.10.14.1 Detailed Description

13.10.15 Vbus API

C++ interface of the Vbus API.

Collaboration diagram for Vbus API:



Data Structures

- class `L4vbus::Pm< DEC >`
Power-management API mixin.
- class `L4vbus::Device`
Device on a `L4vbus::Vbus`.
- class `L4vbus::Icu`
Vbus Interrupt controller API.
- class `L4vbus::Vbus`
The virtual bus (`Vbus`) interface.
- class `L4vbus::Gpio_pin`
A GPIO pin.
- class `L4vbus::Gpio_module`
A `Gpio_module` groups multiple GPIO pins together.
- class `L4vbus::Pci_host_bridge`
A Pci host bridge.
- class `L4vbus::Pci_dev`
A PCI device.

13.10.15.1 Detailed Description

C++ interface of the Vbus API.

The virtual bus (Vbus) is a hierarchical (tree) structure of device nodes where each device has a set of resources attached to it. Each virtual bus provides an Icu ([Interrupt controller](#)) for interrupt handling.

The Vbus interface allows a client to find and query devices present on his virtual bus. After obtaining a device handle for a specific device the client can enumerate its resources.

Refer to [L4 Vbus functions](#) for the C API.

Include File

```
#include <l4/vbus/vbus>
```

Include File

```
#include <l4/vbus/vbus_gpio>
```

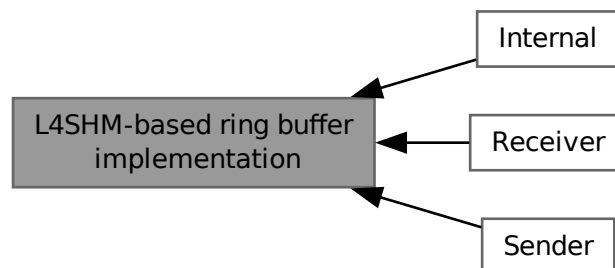
Include File

```
#include <l4/vbus/vbus_pci>
```

13.11 L4SHM-based ring buffer implementation

The library provides a non-locking (strictly 1:1) shared-memory-based ring buffer implementation based on the L4SHM library.

Collaboration diagram for L4SHM-based ring buffer implementation:



Topics

- Sender • [605](#)
- Receiver [605](#)
- Internal • [605](#)

13.11.1 Detailed Description

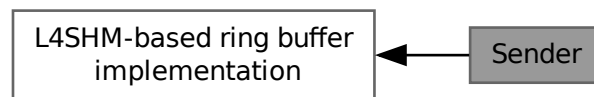
The library provides a non-locking (strictly 1:1) shared-memory-based ring buffer implementation based on the L4SHM library.

It requires an already allocated L4SHM area to be attached to sender and receiver. It will allocate an SHM chunk within this area and provides functions to produce data and consume data in FIFO order from the ring buffer.

The sender side of the buffer needs to be initialized *before* the receiver side, because allocation of the SHM chunk and the necessary signals is done on the sender side and the receiver initialization tries to attach to these objects.

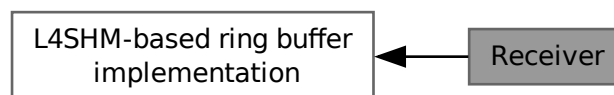
13.11.2 Sender

Collaboration diagram for Sender:



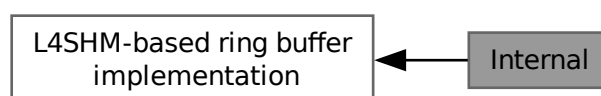
13.11.3 Receiver

Collaboration diagram for Receiver:



13.11.4 Internal

Collaboration diagram for Internal:



Data Structures

- struct [l4shmc_ringbuf_head_t](#)
Head field of a ring buffer.
- struct [l4shmc_ringbuf_t](#)
Ring buffer.

Macros

- #define [L4SHMC_RINGBUF_HEAD](#)(ringbuf)
Get ring buffer head pointer.
- #define [L4SHMC_RINGBUF_DATA](#)(ringbuf)
Get ring buffer data pointer.
- #define [L4SHMC_RINGBUF_DATA_SIZE](#)(ringbuf)
Get size of data area.

13.11.4.1 Detailed Description

13.11.4.2 Macro Definition Documentation

13.11.4.2.1 L4SHMC_RINGBUF_DATA

```
#define L4SHMC_RINGBUF_DATA(  
    ringbuf)
```

Value:

```
(L4SHMC\_RINGBUF\_HEAD(ringbuf)->data)
```

Get ring buffer data pointer.

Parameters

| | |
|----------------|---|
| <i>ringbuf</i> | l4shmc_ringbuf_t struct |
|----------------|---|

Definition at line 114 of file [ringbuf.h](#).

13.11.4.2.2 L4SHMC_RINGBUF_DATA_SIZE

```
#define L4SHMC_RINGBUF_DATA_SIZE(  
    ringbuf)
```

Value:

```
((ringbuf)->_size - sizeof(l4shmc\_ringbuf\_head\_t))
```

Get size of data area.

Parameters

| | |
|----------------|---|
| <i>ringbuf</i> | l4shmc_ringbuf_t struct |
|----------------|---|

Definition at line 123 of file [ringbuf.h](#).

13.11.4.2.3 L4SHMC_RINGBUF_HEAD

```
#define L4SHMC_RINGBUF_HEAD(  
    ringbuf)
```

Value:

```
((l4shmc_ringbuf_head_t*) ((ringbuf)->_addr))
```

Get ring buffer head pointer.

Parameters

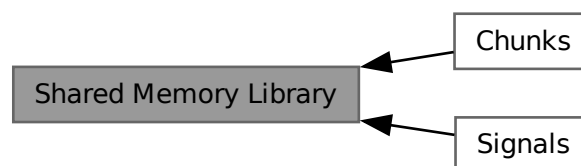
| | |
|----------------|---|
| <i>ringbuf</i> | l4shmc_ringbuf_t struct |
|----------------|---|

Definition at line 105 of file [ringbuf.h](#).

13.12 Shared Memory Library

L4SHM provides a shared memory infrastructure that establishes a shared memory area between multiple parties and uses a fast notification mechanism.

Collaboration diagram for Shared Memory Library:

**Topics**

- [Chunks](#) 613
- [Signals](#) 626

Functions

- [L4_BEGIN_DECLS](#) long [l4shmc_create](#) (char const *shmc_name)
Create a shared memory area.
- long [l4shmc_attach](#) (char const *shmc_name, l4shmc_area_t *shmarea)
Attach to a shared memory area.
- long [l4shmc_get_client_nr](#) (l4shmc_area_t const *shmarea)
Determine the client number of the shared memory region.
- long [l4shmc_mark_client_initialized](#) (l4shmc_area_t *shmarea)
Mark this shared memory client as 'initialized'.
- long [l4shmc_get_initialized_clients](#) (l4shmc_area_t *shmarea, l4_umword_t *bitmask)
Fetch the `_clients_init_done` bitmask of the shared memory area.
- long [l4shmc_connect_chunk_signal](#) (l4shmc_chunk_t *chunk, l4shmc_signal_t *signal)
Connect a signal with a chunk.
- long [l4shmc_area_size](#) (l4shmc_area_t const *shmarea)
Get size of shared memory area.
- long [l4shmc_area_size_free](#) (l4shmc_area_t const *shmarea)
Get free size of shared memory area.
- long [l4shmc_area_overhead](#) (void)
Get memory overhead per area that is not available for chunks.
- long [l4shmc_chunk_overhead](#) (void)
Get memory overhead required in addition to the chunk capacity for adding one chunk.

13.12.1 Detailed Description

L4SHM provides a shared memory infrastructure that establishes a shared memory area between multiple parties and uses a fast notification mechanism.

A shared memory area consists of chunks and signals. A chunk is a defined chunk of memory within the memory area with a maximum size. A chunk is filled (written) by a producer and read by a consumer. When a producer has finished writing to the chunk it signals a data ready notification to the consumer.

A consumer attaches to a chunk and waits for the producer to fill the chunk. After reading out the chunk it marks the chunk free again.

A shared memory area can have multiple chunks.

The interface is divided in three roles.

- The master role, responsible for setting up a shared memory area.
- A producer, generating data into a chunk
- A consumer, receiving data.

A signal can be connected with a chunk or can be used independently (e.g. for multiple chunks).

13.12.2 Function Documentation

13.12.2.1 `l4shmc_area_overhead()`

```
long l4shmc_area_overhead (  
    void )
```

Get memory overhead per area that is not available for chunks.

Returns

Size of the overhead in bytes.

References [L4_CV](#).

13.12.2.2 `l4shmc_area_size()`

```
long l4shmc_area_size (  
    l4shmc_area_t const * shmarea)
```

Get size of shared memory area.

Parameters

| | |
|----------------|---------------------|
| <i>shmarea</i> | Shared memory area. |
|----------------|---------------------|

Return values

| | |
|------|---------------------------------|
| >0 | Size of the shared memory area. |
| <0 | Error. |

References [L4_CV](#).

13.12.2.3 `l4shmc_area_size_free()`

```
long l4shmc_area_size_free (  
    l4shmc_area_t const * shmarea)
```

Get free size of shared memory area.

To get the max size to pass to `l4shmc_add_chunk`, subtract [l4shmc_chunk_overhead\(\)](#).

Parameters

| | |
|----------------|---------------------|
| <i>shmarea</i> | Shared memory area. |
|----------------|---------------------|

Returns

Size of the shared memory area.

References [L4_CV](#).

13.12.2.4 l4shmc_attach()

```
long l4shmc_attach (
    char const * shmc_name,
    l4shmc_area_t * shmarea)
```

Attach to a shared memory area.

Parameters

| | | |
|-----|------------------|--|
| | <i>shmc_name</i> | Name of the shared memory area. |
| out | <i>shmarea</i> | Pointer to shared memory area descriptor to be filled with information for the shared memory area. |

On success, the data in 'shmarea' contains a client number which can be used to mutual agree about client initialization:

- [l4shmc_get_client_nr\(\)](#) returns the client number stored in 'shmarea'. The first attached client will get 0 and this number is increased for each attached client.
- [l4shmc_mark_client_initialized\(\)](#) tells other clients that this client has finished its initialization.
- [l4shmc_get_initialized_clients\(\)](#) returns the bitmap of initialized clients attached to this shared memory.

Return values

| | |
|----|----------|
| 0 | Success. |
| <0 | Error. |

Examples

[examples/libs/shmc/prodcons.c](#).

References [L4_CV](#).

13.12.2.5 l4shmc_chunk_overhead()

```
long l4shmc_chunk_overhead (
    void )
```

Get memory overhead required in addition to the chunk capacity for adding one chunk.

Returns

Size of the overhead in bytes.

References [L4_END_DECLS](#).

13.12.2.6 l4shmc_connect_chunk_signal()

```
long l4shmc_connect_chunk_signal (  
    l4shmc_chunk_t * chunk,  
    l4shmc_signal_t * signal)
```

Connect a signal with a chunk.

Parameters

| | |
|---------------|--------------------------------|
| <i>chunk</i> | Chunk to attach the signal to. |
| <i>signal</i> | Signal to attach. |

Return values

| | |
|----|----------|
| 0 | Success. |
| <0 | Error. |

Examples

[examples/libs/shmc/prodcons.c](#).

References [L4_CV](#), and [L4_INLINE](#).

13.12.2.7 l4shmc_create()

```
L4_BEGIN_DECLS long l4shmc_create (  
    char const * shmc_name)
```

Create a shared memory area.

Parameters

| | |
|------------------|---------------------------------|
| <i>shmc_name</i> | Name of the shared memory area. |
|------------------|---------------------------------|

Return values

| | |
|------------|--|
| 0 | Success. |
| -L4_ENOMEM | The requested size is too big. |
| -L4_ENOENT | No valid capability with the name of the shared memory area found. |
| <0 | Errors from l4re_rm_attach or l4re_ns_register_obj_srv. |

Examples

[examples/libs/shmc/prodcons.c](#).

References [L4_CV](#).

13.12.2.8 l4shmc_get_client_nr()

```
long l4shmc_get_client_nr (
    l4shmc_area_t const * shmarea)
```

Determine the client number of the shared memory region.

Parameters

| | |
|----------------|-------------------------|
| <i>shmarea</i> | The shared memory area. |
|----------------|-------------------------|

Returns

client number.

References [L4_CV](#).

13.12.2.9 l4shmc_get_initialized_clients()

```
long l4shmc_get_initialized_clients (
    l4shmc_area_t * shmarea,
    l4_umword_t * bitmask)
```

Fetch the `_clients_init_done` bitmask of the shared memory area.

Parameters

| | | |
|-----|----------------|---|
| | <i>shmarea</i> | The shared memory area. |
| out | <i>bitmask</i> | The bitmask describing which clients are initialized. |

Return values

| | |
|----|----------|
| 0 | Success. |
| <0 | Error. |

See also

[l4shmc_mark_client_initialized\(\)](#), [l4shmc_get_client_nr\(\)](#)

Examples

[examples/libs/shmc/prodcons.c](#).

References [L4_CV](#).

13.12.2.10 l4shmc_mark_client_initialized()

```
long l4shmc_mark_client_initialized (
    l4shmc_area_t * shmarea)
```

Mark this shared memory client as 'initialized'.

The corresponding bit is set in the `_clients_init_done` bitmask. The bitmask can be fetched with [l4shmc_get_initialized_clients\(\)](#).

Parameters

| | |
|----------------|-------------------------|
| <i>shmarea</i> | The shared memory area. |
|----------------|-------------------------|

Return values

| | |
|----|----------|
| 0 | Success. |
| <0 | Error. |

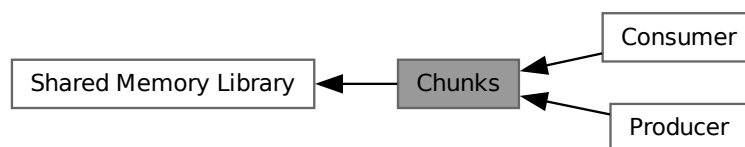
Examples

[examples/libs/shmc/prodcons.c](#).

References [L4_CV](#).

13.12.3 Chunks

Collaboration diagram for Chunks:

**Topics**

- [Producer](#) 618
- [Consumer](#) 621

Functions

- long [l4shmc_add_chunk](#) (l4shmc_area_t *shmarea, char const *chunk_name, [l4_umword_t](#) chunk_capacity, l4shmc_chunk_t *chunk)
Add a chunk in the shared memory area.
- long [l4shmc_get_chunk](#) (l4shmc_area_t *shmarea, char const *chunk_name, l4shmc_chunk_t *chunk)
Get chunk out of shared memory area.
- long [l4shmc_get_chunk_to](#) (l4shmc_area_t *shmarea, char const *chunk_name, [l4_umword_t](#) timeout_ms, l4shmc_chunk_t *chunk)
Get chunk out of shared memory area, with timeout.
- long [l4shmc_iterate_chunk](#) (l4shmc_area_t const *shmarea, char const **chunk_name, long offs)
Iterate over names of all existing chunks.
- void * [l4shmc_chunk_ptr](#) (l4shmc_chunk_t const *chunk)
Get data pointer to chunk.
- long [l4shmc_chunk_capacity](#) (l4shmc_chunk_t const *chunk)
Get capacity of a chunk.
- l4shmc_signal_t * [l4shmc_chunk_signal](#) (l4shmc_chunk_t const *chunk)
Get the registered signal of a chunk.

13.12.3.1 Detailed Description

13.12.3.2 Function Documentation

13.12.3.2.1 l4shmc_add_chunk()

```
long l4shmc_add_chunk (
    l4shmc_area_t * shmarea,
    char const * chunk_name,
    l4\_umword\_t chunk_capacity,
    l4shmc_chunk_t * chunk)
```

Add a chunk in the shared memory area.

Parameters

| | | |
|-----|-----------------------|---|
| | <i>shmarea</i> | The shared memory area to put the chunk in. |
| | <i>chunk_name</i> | Name of the chunk. |
| | <i>chunk_capacity</i> | Capacity for payload of the chunk in bytes. |
| out | <i>chunk</i> | Chunk structure to fill in. |

Return values

| | |
|----|----------|
| 0 | Success. |
| <0 | Error. |

Examples

[examples/libs/shmc/prodcons.c](#).

References [L4_CV](#).

13.12.3.2.2 l4shmc_chunk_capacity()

```
long l4shmc_chunk_capacity (
    l4shmc_chunk_t const * chunk) [inline]
```

Get capacity of a chunk.

Parameters

| | |
|--------------|--------|
| <i>chunk</i> | Chunk. |
|--------------|--------|

Returns

Capacity of the chunk in bytes.

References [L4_CV](#), and [L4_INLINE](#).

13.12.3.2.3 l4shmc_chunk_ptr()

```
void * l4shmc_chunk_ptr (
    l4shmc_chunk_t const * chunk) [inline]
```

Get data pointer to chunk.

Parameters

| | |
|--------------|--------|
| <i>chunk</i> | Chunk. |
|--------------|--------|

Returns

Chunk pointer.

Examples

[examples/libs/shmc/prodcons.c](#).

References [L4_CV](#), and [L4_INLINE](#).

13.12.3.2.4 l4shmc_chunk_signal()

```
l4shmc_signal_t * l4shmc_chunk_signal (
    l4shmc_chunk_t const * chunk) [inline]
```

Get the registered signal of a chunk.

Parameters

| | |
|--------------|--------|
| <i>chunk</i> | Chunk. |
|--------------|--------|

Return values

| | |
|-----|--|
| 0 | No signal has been registered with this chunk. |
| !=0 | Pointer to signal otherwise. |

References [L4_CV](#), and [L4_INLINE](#).

13.12.3.2.5 l4shmc_get_chunk()

```
long l4shmc_get_chunk (
    l4shmc_area_t * shmbarea,
    char const * chunk_name,
    l4shmc_chunk_t * chunk) [inline]
```

Get chunk out of shared memory area.

Parameters

| | | |
|-----|-------------------|-------------------------------|
| | <i>shmbarea</i> | Shared memory area. |
| | <i>chunk_name</i> | Name of the chunk. |
| out | <i>chunk</i> | Chunk data structure to fill. |

Return values

| | |
|----|----------|
| 0 | Success. |
| <0 | Error. |

Examples

[examples/libs/shmc/prodcons.c](#).

References [L4_CV](#).

13.12.3.2.6 l4shmc_get_chunk_to()

```
long l4shmc_get_chunk_to (
    l4shmc_area_t * shmarea,
    char const * chunk_name,
    l4_umword_t timeout_ms,
    l4shmc_chunk_t * chunk)
```

Get chunk out of shared memory area, with timeout.

Parameters

| | | |
|-----|-------------------|--|
| | <i>shmarea</i> | Shared memory area. |
| | <i>chunk_name</i> | Name of the chunk. |
| | <i>timeout_ms</i> | Timeout in milliseconds to wait for the chunk to appear in the shared memory area. |
| out | <i>chunk</i> | Chunk data structure to fill. |

Return values

| | |
|----|----------|
| 0 | Success. |
| <0 | Error. |

References [L4_CV](#).

13.12.3.2.7 l4shmc_iterate_chunk()

```
long l4shmc_iterate_chunk (
    l4shmc_area_t const * shmarea,
    char const ** chunk_name,
    long offs)
```

Iterate over names of all existing chunks.

Parameters

| | |
|-------------------|---|
| <i>shmarea</i> | Shared memory area. |
| <i>chunk_name</i> | Where the name of the current chunk will be stored |
| <i>offs</i> | 0 to start iteration, return value of previous call to l4shmc_iterate_chunk() to get next chunk |

Return values

| | |
|----|-----------------------------------|
| 0 | No more chunks available. |
| <0 | Error. |
| >0 | Iterator value for the next call. |

References [L4_CV](#).

13.12.3.3 Producer

Collaboration diagram for Producer:



Functions

- long [l4shmc_chunk_try_to_take](#) (l4shmc_chunk_t *chunk)
Try to mark chunk busy.
- long [l4shmc_chunk_try_to_take_for_writing](#) (l4shmc_chunk_t *chunk)
Try to mark chunk busy writing.
- long [l4shmc_chunk_try_to_take_for_overwriting](#) (l4shmc_chunk_t *chunk)
Try to mark the chunk busy writing after it was ready for reading.
- long [l4shmc_chunk_ready](#) (l4shmc_chunk_t *chunk, [l4_umword_t](#) size)
Mark chunk as filled (ready).
- long [l4shmc_chunk_ready_sig](#) (l4shmc_chunk_t *chunk, [l4_umword_t](#) size)
Mark chunk as filled (ready) and signal consumer.
- long [l4shmc_is_chunk_clear](#) (l4shmc_chunk_t const *chunk)
Check whether chunk is free.

13.12.3.3.1 Detailed Description

13.12.3.3.2 Function Documentation

13.12.3.3.2.1 l4shmc_chunk_ready()

```

long l4shmc_chunk_ready (
    l4shmc_chunk_t * chunk,
    l4_umword_t size) [inline]
  
```

Mark chunk as filled (ready).

Parameters

| | |
|--------------|--------------------------------------|
| <i>chunk</i> | chunk. |
| <i>size</i> | Size of data in the chunk, in bytes. |

Return values

| | |
|----|----------|
| 0 | Success. |
| <0 | Error. |

References [L4_CV](#), and [L4_INLINE](#).

13.12.3.3.2.2 l4shmc_chunk_ready_sig()

```
long l4shmc_chunk_ready_sig (  
    l4shmc_chunk_t * chunk,  
    l4_umword_t size) [inline]
```

Mark chunk as filled (ready) and signal consumer.

Parameters

| | |
|--------------|--------------------------------------|
| <i>chunk</i> | chunk. |
| <i>size</i> | Size of data in the chunk, in bytes. |

Return values

| | |
|----|----------|
| 0 | Success. |
| <0 | Error. |

Examples

[examples/libs/shmc/prodcons.c](#).

References [L4_CV](#), and [L4_INLINE](#).

13.12.3.3.2.3 l4shmc_chunk_try_to_take()

```
long l4shmc_chunk_try_to_take (  
    l4shmc_chunk_t * chunk) [inline]
```

Try to mark chunk busy.

Parameters

| | |
|--------------|----------------|
| <i>chunk</i> | chunk to mark. |
|--------------|----------------|

Return values

| | |
|---|-----------------------|
| 0 | Chunk could be taken. |
|---|-----------------------|

| | |
|-------|--------------------------------------|
| < 0 | Chunk could not be taken, try again. |
|-------|--------------------------------------|

Examples

[examples/libs/shmc/prodcons.c](#).

References [L4_CV](#), and [L4_INLINE](#).

13.12.3.3.2.4 l4shmc_chunk_try_to_take_for_overwriting()

```
long l4shmc_chunk_try_to_take_for_overwriting (
    l4shmc_chunk_t * chunk) [inline]
```

Try to mark the chunk busy writing after it was ready for reading.

Parameters

| | |
|--------------|-----------------------------|
| <i>chunk</i> | chunk to mark busy writing. |
|--------------|-----------------------------|

This function is used by the producer to overwrite a message if the consumer did not read the message within an expected time. This function can only be used if the consumer uses [l4shmc_chunk_try_to_take_for_reading\(\)](#) before reading the chunk.

Return values

| | |
|-------|--|
| 0 | Chunk could be taken and can be written. |
| < 0 | Chunk could not be taken, try again. |

References [L4_CV](#), and [L4_INLINE](#).

13.12.3.3.2.5 l4shmc_chunk_try_to_take_for_writing()

```
long l4shmc_chunk_try_to_take_for_writing (
    l4shmc_chunk_t * chunk) [inline]
```

Try to mark chunk busy writing.

This function is actually an alias for [l4shmc_chunk_try_to_take\(\)](#).

Parameters

| | |
|--------------|-----------------------------|
| <i>chunk</i> | chunk to mark busy writing. |
|--------------|-----------------------------|

Return values

| | |
|--------------------|--|
| <code>0</code> | Chunk could be taken and can be written. |
| <code><0</code> | Chunk could not be taken, try again. |

References [L4_CV](#), and [L4_INLINE](#).

13.12.3.3.2.6 `l4shmc_is_chunk_clear()`

```
long l4shmc_is_chunk_clear (
    l4shmc_chunk_t const * chunk) [inline]
```

Check whether chunk is free.

Parameters

| | |
|--------------------|-----------------|
| <code>chunk</code> | Chunk to check. |
|--------------------|-----------------|

Return values

| | |
|------------------|---------------------|
| <code>!=0</code> | Chunk is clear. |
| <code>0</code> | Chunk is not clear. |

References [L4_CV](#), and [L4_INLINE](#).

13.12.3.4 Consumer

Collaboration diagram for Consumer:



Functions

- long [l4shmc_chunk_try_to_take_for_reading](#) (l4shmc_chunk_t *chunk)
Try to mark chunk busy reading.
- long [l4shmc_enable_chunk](#) (l4shmc_chunk_t *chunk)
Enable a signal connected with a chunk.
- long [l4shmc_wait_chunk](#) (l4shmc_chunk_t *chunk)
Wait on a specific chunk.
- long [l4shmc_wait_chunk_to](#) (l4shmc_chunk_t *chunk, [l4_timeout_t](#) timeout)

- *Check whether a specific chunk has an event pending, with timeout.*
 • long [l4shmc_wait_chunk_try](#) (l4shmc_chunk_t *chunk)
Check whether a specific chunk has an event pending.
- long [l4shmc_chunk_consumed](#) (l4shmc_chunk_t *chunk)
Mark a chunk as free.
- long [l4shmc_is_chunk_ready](#) (l4shmc_chunk_t const *chunk)
Check whether data is available.
- long [l4shmc_chunk_size](#) (l4shmc_chunk_t const *chunk)
Get current size of a chunk.

13.12.3.4.1 Detailed Description

13.12.3.4.2 Function Documentation

13.12.3.4.2.1 l4shmc_chunk_consumed()

```
long l4shmc_chunk_consumed (
    l4shmc_chunk_t * chunk) [inline]
```

Mark a chunk as free.

Parameters

| | |
|--------------|------------------------|
| <i>chunk</i> | Chunk to mark as free. |
|--------------|------------------------|

Return values

| | |
|----|----------|
| 0 | Success. |
| <0 | Error. |

Examples

[examples/libs/shmc/prodcons.c](#).

References [L4_CV](#).

13.12.3.4.2.2 l4shmc_chunk_size()

```
long l4shmc_chunk_size (
    l4shmc_chunk_t const * chunk) [inline]
```

Get current size of a chunk.

Parameters

| | |
|--------------|--------|
| <i>chunk</i> | Chunk. |
|--------------|--------|

Returns

Current size of the chunk in bytes.

Examples

[examples/libs/shmc/prodcons.c](#).

References [L4_CV](#), and [L4_INLINE](#).

13.12.3.4.2.3 l4shmc_chunk_try_to_take_for_reading()

```
long l4shmc_chunk_try_to_take_for_reading (
    l4shmc_chunk_t * chunk) [inline]
```

Try to mark chunk busy reading.

Parameters

| | |
|--------------|-----------------------------|
| <i>chunk</i> | chunk to mark busy reading. |
|--------------|-----------------------------|

Return values

| | |
|----|---------------------------------------|
| 0 | Chunk could be taken and can be read. |
| <0 | Chunk could not be taken, try again. |

References [L4_CV](#), and [L4_INLINE](#).

13.12.3.4.2.4 l4shmc_enable_chunk()

```
long l4shmc_enable_chunk (
    l4shmc_chunk_t * chunk)
```

Enable a signal connected with a chunk.

Parameters

| | |
|--------------|------------------|
| <i>chunk</i> | Chunk to enable. |
|--------------|------------------|

Return values

| | |
|----|----------|
| 0 | Success. |
| <0 | Error. |

A signal must be enabled before waiting when the consumer waits on any signal. Enabling is not needed if the consumer waits for a specific signal or chunk.

References [L4_CV](#), and [L4_INLINE](#).

13.12.3.4.2.5 l4shmc_is_chunk_ready()

```
long l4shmc_is_chunk_ready (
    l4shmc_chunk_t const * chunk) [inline]
```

Check whether data is available.

Parameters

| | |
|--------------|-----------------|
| <i>chunk</i> | Chunk to check. |
|--------------|-----------------|

Return values

| | |
|------------|--------------------|
| <i>!=0</i> | Data is available. |
| <i>0</i> | No data available. |

References [L4_CV](#), and [L4_INLINE](#).

13.12.3.4.2.6 l4shmc_wait_chunk()

```
long l4shmc_wait_chunk (
    l4shmc_chunk_t * chunk) [inline]
```

Wait on a specific chunk.

Parameters

| | |
|--------------|--------------------|
| <i>chunk</i> | Chunk to wait for. |
|--------------|--------------------|

Return values

| | |
|--------------|----------|
| <i>0</i> | Success. |
| <i><0</i> | Error. |

Examples

[examples/libs/shmc/prodcons.c](#).

References [L4_CV](#).

13.12.3.4.2.7 l4shmc_wait_chunk_to()

```
long l4shmc_wait_chunk_to (
    l4shmc_chunk_t * chunk,
    l4_timeout_t timeout)
```

Check whether a specific chunk has an event pending, with timeout.

Parameters

| | |
|----------------|-----------------|
| <i>chunk</i> | Chunk to check. |
| <i>timeout</i> | Timeout. |

Return values

| | |
|----|----------|
| 0 | Success. |
| <0 | Error. |

The return code indicates whether an event was pending or not. Success means an event was pending, if an receive timeout error is returned no event was pending.

References [L4_CV](#), and [L4_INLINE](#).

13.12.3.4.2.8 l4shmc_wait_chunk_try()

```
long l4shmc_wait_chunk_try (
    l4shmc_chunk_t * chunk) [inline]
```

Check whether a specific chunk has an event pending.

Parameters

| | |
|--------------|-----------------|
| <i>chunk</i> | Chunk to check. |
|--------------|-----------------|

Return values

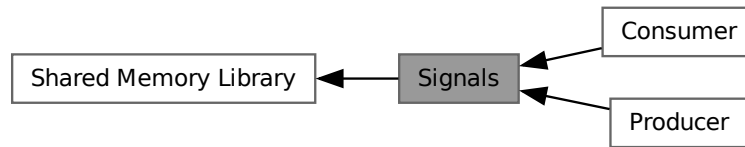
| | |
|----|----------|
| 0 | Success. |
| <0 | Error. |

The return code indicates whether an event was pending or not. Success means an event was pending, if an receive timeout error is returned no event was pending.

References [L4_CV](#), and [L4_INLINE](#).

13.12.4 Signals

Collaboration diagram for Signals:



Topics

- [Producer](#) 629
- [Consumer](#) 630

Functions

- [long l4shmc_add_signal](#) (l4shmc_area_t *shmarea, char const *signal_name, l4shmc_signal_t *signal)
Add a signal for the shared memory area.
- [long l4shmc_attach_signal](#) (l4shmc_area_t *shmarea, char const *signal_name, [l4_cap_idx_t](#) thread, l4shmc_signal_t *signal)
Attach to signal.
- [long l4shmc_get_signal](#) (l4shmc_area_t *shmarea, char const *signal_name, l4shmc_signal_t *signal)
Get signal object from the shared memory area.
- [l4_cap_idx_t l4shmc_signal_cap](#) (l4shmc_signal_t const *signal)
Get the signal capability of a signal.
- [long l4shmc_check_magic](#) (l4shmc_chunk_t const *chunk)
Check magic value of a chunk.

13.12.4.1 Detailed Description

13.12.4.2 Function Documentation

13.12.4.2.1 l4shmc_add_signal()

```

long l4shmc_add_signal (
    l4shmc_area_t * shmarea,
    char const * signal_name,
    l4shmc_signal_t * signal)
  
```

Add a signal for the shared memory area.

Parameters

| | | |
|-----|--------------------|------------------------------|
| | <i>shmbarea</i> | The shared memory area. |
| | <i>signal_name</i> | Name of the signal. |
| out | <i>signal</i> | Signal structure to fill in. |

Return values

| | |
|----|----------|
| 0 | Success. |
| <0 | Error. |

Examples

[examples/libs/shmc/prodcons.c](#).

References [L4_CV](#), and [L4_INLINE](#).

13.12.4.2.2 l4shmc_attach_signal()

```
long l4shmc_attach_signal (  
    l4shmc_area_t * shmbarea,  
    char const * signal_name,  
    l4_cap_idx_t thread,  
    l4shmc_signal_t * signal)
```

Attach to signal.

Parameters

| | | |
|-----|--------------------|--|
| | <i>shmbarea</i> | Shared memory area. |
| | <i>signal_name</i> | Name of the signal. |
| | <i>thread</i> | Thread capability index to attach the signal to. |
| out | <i>signal</i> | Signal data structure to fill. |

Return values

| | |
|----|----------|
| 0 | Success. |
| <0 | Error. |

Examples

[examples/libs/shmc/prodcons.c](#).

References [L4_CV](#).

13.12.4.2.3 l4shmc_check_magic()

```
long l4shmc_check_magic (
    l4shmc_chunk_t const * chunk) [inline]
```

Check magic value of a chunk.

Parameters

| | |
|--------------|--------|
| <i>chunk</i> | Chunk. |
|--------------|--------|

Return values

| | |
|----|--|
| 0 | Magic value is not valid. |
| >0 | Chunk is OK, the magic value is valid. |

References [L4_CV](#).

13.12.4.2.4 l4shmc_get_signal()

```
long l4shmc_get_signal (
    l4shmc_area_t * shmbarea,
    char const * signal_name,
    l4shmc_signal_t * signal)
```

Get signal object from the shared memory area.

Parameters

| | | |
|-----|--------------------|--------------------------------|
| | <i>shmbarea</i> | Shared memory area. |
| | <i>signal_name</i> | Name of the signal. |
| out | <i>signal</i> | Signal data structure to fill. |

Return values

| | |
|----|----------|
| 0 | Success. |
| <0 | Error. |

References [L4_CV](#).

13.12.4.2.5 l4shmc_signal_cap()

```
l4\_cap\_idx\_t l4shmc_signal_cap (
    l4shmc_signal_t const * signal) [inline]
```

Get the signal capability of a signal.

Parameters

| | |
|---------------|---------|
| <i>signal</i> | Signal. |
|---------------|---------|

Returns

Capability of the signal object.

References [L4_CV](#), and [L4_INLINE](#).

13.12.4.3 Producer

Collaboration diagram for Producer:

**Functions**

- long [l4shmc_trigger](#) (l4shmc_signal_t *signal)
Trigger a signal.

13.12.4.3.1 Detailed Description**13.12.4.3.2 Function Documentation****13.12.4.3.2.1 l4shmc_trigger()**

```

long l4shmc_trigger (
    l4shmc_signal_t * signal) [inline]
  
```

Trigger a signal.

Parameters

| | |
|---------------|--------------------|
| <i>signal</i> | Signal to trigger. |
|---------------|--------------------|

Return values

| | |
|----|----------|
| 0 | Success. |
| <0 | Error. |

Examples

[examples/libs/shmc/prodcons.c](#).

References [L4_CV](#), and [L4_INLINE](#).

13.12.4.4 Consumer

Collaboration diagram for Consumer:

**Functions**

- long [l4shmc_enable_signal](#) (l4shmc_signal_t *signal)
Enable a signal.
- long [l4shmc_wait_any](#) (l4shmc_signal_t **retsignal)
Wait on any signal.
- long [l4shmc_wait_any_try](#) (l4shmc_signal_t **retsignal)
Check whether any waited signal has an event pending.
- long [l4shmc_wait_any_to](#) (l4_timeout_t timeout, l4shmc_signal_t **retsignal)
Wait for any signal with timeout.
- long [l4shmc_wait_signal](#) (l4shmc_signal_t *signal)
Wait on a specific signal.
- long [l4shmc_wait_signal_to](#) (l4shmc_signal_t *signal, l4_timeout_t timeout)
Wait on a specific signal, with timeout.
- long [l4shmc_wait_signal_try](#) (l4shmc_signal_t *signal)
Check whether a specific signal has an event pending.

13.12.4.4.1 Detailed Description

13.12.4.4.2 Function Documentation

13.12.4.4.2.1 l4shmc_enable_signal()

```
long l4shmc_enable_signal (  
    l4shmc_signal_t * signal)
```

Enable a signal.

Parameters

| | |
|---------------|-------------------|
| <i>signal</i> | Signal to enable. |
|---------------|-------------------|

Return values

| | |
|----|----------|
| 0 | Success. |
| <0 | Error. |

A signal must be enabled before waiting when the consumer waits on any signal. Enabling is not needed if the consumer waits for a specific signal or chunk.

References [L4_CV](#).

13.12.4.4.2.2 l4shmc_wait_any()

```
long l4shmc_wait_any (  
    l4shmc_signal_t ** retsignal) [inline]
```

Wait on any signal.

Parameters

| | | |
|-----|------------------|------------------|
| out | <i>retsignal</i> | Signal received. |
|-----|------------------|------------------|

Return values

| | |
|----|----------|
| 0 | Success. |
| <0 | Error. |

References [L4_CV](#), and [L4_INLINE](#).

13.12.4.4.2.3 l4shmc_wait_any_to()

```
long l4shmc_wait_any_to (
    l4_timeout_t timeout,
    l4shmc_signal_t ** retsignal)
```

Wait for any signal with timeout.

Parameters

| | | |
|-----|------------------|---|
| | <i>timeout</i> | Timeout. |
| out | <i>retsignal</i> | Signal that has the event pending if any. |

Return values

| | |
|----|----------|
| 0 | Success. |
| <0 | Error. |

The return code indicates whether an event was pending or not. Success means an event was pending, if an receive timeout error is returned no event was pending.

References [L4_CV](#), and [L4_INLINE](#).

13.12.4.4.2.4 l4shmc_wait_any_try()

```
long l4shmc_wait_any_try (
    l4shmc_signal_t ** retsignal) [inline]
```

Check whether any waited signal has an event pending.

Parameters

| | | |
|-----|------------------|---|
| out | <i>retsignal</i> | Signal that has the event pending if any. |
|-----|------------------|---|

Return values

| | |
|----|----------|
| 0 | Success. |
| <0 | Error. |

The return code indicates whether an event was pending or not. Success means an event was pending, if an receive timeout error is returned no event was pending.

References [L4_CV](#).

13.12.4.4.2.5 l4shmc_wait_signal()

```
long l4shmc_wait_signal (  
    l4shmc_signal_t * signal)    [inline]
```

Wait on a specific signal.

Parameters

| | |
|---------------|---------------------|
| <i>signal</i> | Signal to wait for. |
|---------------|---------------------|

Return values

| | |
|----|----------|
| 0 | Success. |
| <0 | Error. |

Examples

[examples/libs/shmc/prodcons.c](#).

References [L4_CV](#).

13.12.4.4.2.6 l4shmc_wait_signal_to()

```
long l4shmc_wait_signal_to (  
    l4shmc_signal_t * signal,  
    l4_timeout_t timeout)
```

Wait on a specific signal, with timeout.

Parameters

| | |
|----------------|---------------------|
| <i>signal</i> | Signal to wait for. |
| <i>timeout</i> | Timeout. |

Return values

| | |
|----|----------|
| 0 | Success. |
| <0 | Error. |

References [L4_CV](#), and [L4_INLINE](#).

13.12.4.4.2.7 l4shmc_wait_signal_try()

```
long l4shmc_wait_signal_try (
    l4shmc_signal_t * signal) [inline]
```

Check whether a specific signal has an event pending.

Parameters

| | |
|---------------|------------------|
| <i>signal</i> | Signal to check. |
|---------------|------------------|

Return values

| | |
|----|----------|
| 0 | Success. |
| <0 | Error. |

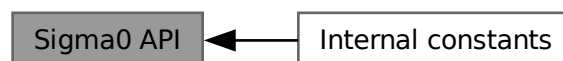
The return code indicates whether an event was pending or not. Success means an event was pending, if an receive timeout error is returned no event was pending.

References [L4_CV](#), and [L4_INLINE](#).

13.13 Sigma0 API

Sigma0 API bindings.

Collaboration diagram for Sigma0 API:



Topics

- [Internal constants](#) 639
Internal sigma0 definitions.

Files

- file [sigma0.h](#)
Sigma0 interface.

Enumerations

- enum `l4sigma0_return_flags_t` {
`L4SIGMA0_OK` , `L4SIGMA0_NOTALIGNED` , `L4SIGMA0_IPCERROR` , `L4SIGMA0_NOFPAGE` ,
`L4SIGMA0_4` , `L4SIGMA0_5` , `L4SIGMA0_SMALLERFPAGE` }

Return flags of libsigma0 functions.

Functions

- `L4_BEGIN_DECLS l4_kernel_info_t * l4sigma0_map_kip (l4_cap_idx_t sigma0, void *addr, unsigned log2↵_size)`
Map the kernel info page from sigma0 to addr.
- int `l4sigma0_map_mem (l4_cap_idx_t sigma0, l4_addr_t phys, l4_addr_t virt, l4_addr_t size)`
Request a memory mapping from sigma0.
- int `l4sigma0_map_iomem (l4_cap_idx_t sigma0, l4_addr_t phys, l4_addr_t virt, l4_addr_t size, int cached)`
Request IO memory from sigma0.
- int `l4sigma0_map_anypage (l4_cap_idx_t sigma0, l4_addr_t map_area, unsigned log2_map_size, l4_addr_t *base, unsigned sz)`
Request an arbitrary free page of RAM.
- void `l4sigma0_debug_dump (l4_cap_idx_t sigma0)`
Request sigma0 to dump internal debug information.
- char const * `l4sigma0_map_errstr (int err)`
Get user readable error messages for the return codes.

13.13.1 Detailed Description

Sigma0 API bindings.

Convenience bindings for the Sigma0 protocol.

13.13.2 Enumeration Type Documentation

13.13.2.1 l4sigma0_return_flags_t

enum `l4sigma0_return_flags_t`

Return flags of libsigma0 functions.

Enumerator

| | |
|------------------------------------|--|
| <code>L4SIGMA0_OK</code> | Ok. |
| <code>L4SIGMA0_NOTALIGNED</code> | Phys, virt or size not aligned. |
| <code>L4SIGMA0_IPCERROR</code> | IPC error. |
| <code>L4SIGMA0_NOFPAGE</code> | No fpage received. |
| <code>L4SIGMA0_SMALLERFPAGE</code> | Superpage requested but smaller flexpage received. |

Definition at line 76 of file [sigma0.h](#).

13.13.3 Function Documentation

13.13.3.1 `l4sigma0_debug_dump()`

```
void l4sigma0_debug_dump (
    l4_cap_idx_t sigma0)
```

Request sigma0 to dump internal debug information.

Parameters

| | |
|---------------|--|
| <i>sigma0</i> | Capability selector for the sigma0 gate. |
|---------------|--|

The debug information, such as internal memory maps, as well as statistics about the internal allocators is dumped to the kernel debugger.

References [L4_CV](#), and [L4_INLINE](#).

13.13.3.2 `l4sigma0_map_anypage()`

```
int l4sigma0_map_anypage (
    l4_cap_idx_t sigma0,
    l4_addr_t map_area,
    unsigned log2_map_size,
    l4_addr_t * base,
    unsigned sz)
```

Request an arbitrary free page of RAM.

Parameters

| | | |
|-----|----------------------|--|
| | <i>sigma0</i> | Capability selector for the sigma0 gate. |
| | <i>map_area</i> | The base address of the local virtual memory area where the page should be mapped. |
| | <i>log2_map_size</i> | The size of the requested page log 2 (the size in bytes is $2^{\text{log2_map_size}}$). This must be at least the minimal page size. By specifying larger sizes the largest possible hardware page size will be used. |
| out | <i>base</i> | Physical address of the page received (i.e. the send base of the received mapping if any). |
| | <i>sz</i> | Size to map by the server in 2^{sz} bytes. |

Return values

| | |
|---------------------------|--------------------|
| <i>0</i> | Success. |
| <i>-L4SIGMA0_IPCERROR</i> | IPC error. |
| <i>-L4SIGMA0_NOFPAGE</i> | No fpage received. |

This function requests arbitrary free memory from sigma0. It should be used whenever spare memory is needed, instead of requesting specific physical memory with [l4sigma0_map_mem\(\)](#).

See [l4sigma0_map_errstr\(\)](#) to get a description of the return value.

References [L4_CV](#).

13.13.3.3 l4sigma0_map_errstr()

```
char const * l4sigma0_map_errstr (
    int err) [inline]
```

Get user readable error messages for the return codes.

Parameters

| | |
|------------|--|
| <i>err</i> | The error code reported by the <i>map</i> functions. |
|------------|--|

Returns

A string containing the error message.

Definition at line 210 of file [sigma0.h](#).

References [L4_INLINE](#).

13.13.3.4 l4sigma0_map_iomem()

```
int l4sigma0_map_iomem (
    l4_cap_idx_t sigma0,
    l4_addr_t phys,
    l4_addr_t virt,
    l4_addr_t size,
    int cached)
```

Request IO memory from sigma0.

Parameters

| | |
|---------------|--|
| <i>sigma0</i> | Capability selector for the sigma0 gate. |
| <i>phys</i> | The physical address to be requested (page aligned). |
| <i>virt</i> | The virtual address where the memory should be mapped to (page aligned). |
| <i>size</i> | The size of the IO memory area to be mapped (multiple of page size) |
| <i>cached</i> | Requests cacheable IO memory if 1 and uncached if 0. |

Return values

| | |
|----------------------|---|
| 0 | Success. |
| -L4SIGMA0_NOTALIGNED | <i>phys</i> , <i>virt</i> , or <i>size</i> are not aligned. |
| -L4SIGMA0_IPCERROR | IPC error. |
| -L4SIGMA0_NOFPAGE | No fpage received. |

This function is similar to [l4sigma0_map_mem\(\)](#), the difference is that it requests IO memory. IO memory is everything that is not known to be normal RAM. Also ACPI tables or the BIOS memory is treated as IO memory.

See [l4sigma0_map_errstr\(\)](#) to get a description of the return value.

References [L4_CV](#).

13.13.3.5 l4sigma0_map_kip()

```
L4_BEGIN_DECLS l4_kernel_info_t * l4sigma0_map_kip (
    l4_cap_idx_t sigma0,
    void * addr,
    unsigned log2_size)
```

Map the kernel info page from sigma0 to addr.

Parameters

| | |
|------------------|--|
| <i>sigma0</i> | Capability selector for the sigma0 gate. |
| <i>addr</i> | Start of the receive window to receive KIP in. |
| <i>log2_size</i> | Size of the receive window to receive KIP in. |

Returns

Address KIP was mapped to, 0 indicates an error.

13.13.3.6 l4sigma0_map_mem()

```
int l4sigma0_map_mem (
    l4_cap_idx_t sigma0,
    l4_addr_t phys,
    l4_addr_t virt,
    l4_addr_t size)
```

Request a memory mapping from sigma0.

Parameters

| | |
|---------------|--|
| <i>sigma0</i> | Capability selector for the sigma0 gate. |
| <i>phys</i> | The physical address of the requested page (must be at least aligned to the minimum page size). |
| <i>virt</i> | The virtual address where the paged should be mapped in the local address space (must be at least aligned to the minimum page size). |
| <i>size</i> | The size of the requested page, this must be a multiple of the minimum page size. |

Return values

| | |
|-----------------------------|---|
| <i>0</i> | Success. |
| <i>-L4SIGMA0_NOTALIGNED</i> | <i>phys</i> , <i>virt</i> , or <i>size</i> are not aligned. |
| <i>-L4SIGMA0_IPCERROR</i> | IPC error. |
| <i>-L4SIGMA0_NOFPAGE</i> | No fpage received. |

This function only maps normal RAM. To map other memory, use [l4sigma0_map_iomem\(\)](#). See also there for the distinction between both memory types.

This is the direct method to request memory from sigma0. There is also the indirect method where sigma0 will answer page faults with a mapping that is one-to-one between the faulting virtual page and the backing physical page. See [L4::Pager::page_fault\(\)](#). For an overview of the memory hierarchy, see [Memory management - Data Spaces and the Region Map](#).

See [l4sigma0_map_errstr\(\)](#) to get a description of the return value.

References [L4_CV](#).

13.13.4 Internal constants

Internal sigma0 definitions.

Collaboration diagram for Internal constants:



Macros

- **#define SIGMA0_REQ_MAGIC** ~0xFFUL
Request magic.
- **#define SIGMA0_REQ_MASK** ~0xFFUL
Request mask.
- **#define SIGMA0_REQ_ID_MASK** 0xF0
ID mask.
- **#define SIGMA0_REQ_ID_FPAGE_RAM** 0x60
RAM.
- **#define SIGMA0_REQ_ID_FPAGE_IOMEM** 0x70
I/O memory.
- **#define SIGMA0_REQ_ID_FPAGE_IOMEM_CACHED** 0x80
Cached I/O memory.
- **#define SIGMA0_REQ_ID_FPAGE_ANY** 0x90
Any.
- **#define SIGMA0_REQ_ID_KIP** 0xA0
KIP.
- **#define SIGMA0_REQ_ID_DEBUG_DUMP** 0xC0
Debug dump.
- **#define SIGMA0_REQ_ID_COV** 0xE0
Trigger cov dump.
- **#define SIGMA0_IS_MAGIC_REQ**(d1)
Check if magic.
- **#define SIGMA0_REQ**(x)
Construct.

- `#define SIGMA0_REQ_FPAGE_RAM (SIGMA0_REQ(FPAGE_RAM))`
RAM.
- `#define SIGMA0_REQ_FPAGE_IOMEM (SIGMA0_REQ(FPAGE_IOMEM))`
I/O memory.
- `#define SIGMA0_REQ_FPAGE_IOMEM_CACHED (SIGMA0_REQ(FPAGE_IOMEM_CACHED))`
Cache I/O memory.
- `#define SIGMA0_REQ_FPAGE_ANY (SIGMA0_REQ(FPAGE_ANY))`
Any.
- `#define SIGMA0_REQ_KIP (SIGMA0_REQ(KIP))`
KIP.
- `#define SIGMA0_REQ_DEBUG_DUMP (SIGMA0_REQ(DEBUG_DUMP))`
Debug dump.
- `#define SIGMA0_REQ_COV (SIGMA0_REQ(COV))`
Cov.

13.13.4.1 Detailed Description

Internal sigma0 definitions.

13.14 Small C++ Template Library

Namespaces

- namespace `cxx`
Our C++ library.

Data Structures

- class `L4::Alloc_list`
A simple list-based allocator.
- class `cxx::List_item`
Basic list item.
- struct `cxx::Pair< First, Second >`
Pair of two values.
- class `cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >`
Basic slab allocator.
- class `cxx::Slab< Type, Slab_size, Max_free, Alloc >`
Slab allocator for object of type `Type`.
- class `cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >`
Merged slab allocator (allocators for objects of the same size are merged together).
- class `cxx::Slab_static< Type, Slab_size, Max_free, Alloc >`
Merged slab allocator (allocators for objects of the same size are merged together).
- class `cxx::Nothrow`
Helper type to distinguish the `operator new` version that does not throw exceptions.
- class `cxx::New_allocator< _Type >`
Standard allocator based on `operator new` ().
- class `L4::String`
A null-terminated string container class.

Functions

- `template<typename A, typename ... ARGS>`
`constexpr A const & cxx::min (A const &a1, A const &a2, ARGS const &...a)`
Get the minimum of a_1 and a_2 up to a_N .
- `template<typename A, typename ... ARGS>`
`constexpr A const & cxx::min (cxx::identity_t< A > const &a1, cxx::identity_t< A > const &a2, ARGS const &...a)`
Get the minimum of a_1 and a_2 up to a_N .
- `template<typename A, typename ... ARGS>`
`constexpr A const & cxx::max (A const &a1, A const &a2, ARGS const &...a)`
Get the maximum of a_1 and a_2 up to a_N .
- `template<typename A, typename ... ARGS>`
`constexpr A const & cxx::max (cxx::identity_t< A > const &a1, cxx::identity_t< A > const &a2, ARGS const &...a)`
Get the maximum of a_1 and a_2 up to a_N .
- `template<typename T1>`
`T1 cxx::clamp (T1 v, T1 lo, T1 hi)`
Limit v to the range given by lo and hi .
- `void * operator new (size_t, void *mem, cxx::Nothrow const &) noexcept`
Simple placement new operator.
- `void * operator new (size_t, cxx::Nothrow const &) noexcept`
New operator that does not throw exceptions.
- `void operator delete (void *, cxx::Nothrow const &) noexcept`
Delete operator complementing the new operator not throwing exceptions.

13.14.1 Detailed Description

13.14.2 Function Documentation

13.14.2.1 `clamp()`

```
template<typename T1>
T1 cxx::clamp (
    T1 v,
    T1 lo,
    T1 hi) [inline]
```

Limit v to the range given by lo and hi .

Parameters

| | |
|------|-------------------------------------|
| v | The value to clamp. |
| lo | The lower boundary to clamp v to. |
| hi | The upper boundary to clamp v to. |

Definition at line [109](#) of file [minmax](#).

13.14.2.2 max() [1/2]

```
template<typename A, typename ... ARGS>
A const & cxx::max (
    A const & a1,
    A const & a2,
    ARGS const &... a) [constexpr]
```

Get the maximum of a1 and a2 upt to aN.

Parameters

| | |
|-------------------|--|
| <i>a1</i> | The first value. |
| <i>a2</i> | The second value. |
| <i>...↔ a</i> | Arbitrary number of additional parameters. |

Matches with automatic argument type deduction.

Definition at line 78 of file [minmax](#).

13.14.2.3 max() [2/2]

```
template<typename A, typename ... ARGS>
A const & cxx::max (
    cxx::identity_t< A > const & a1,
    cxx::identity_t< A > const & a2,
    ARGS const &... a) [constexpr]
```

Get the maximum of a1 and a2 upt to aN.

Parameters

| | |
|-------------------|--|
| <i>a1</i> | The first value. |
| <i>a2</i> | The second value. |
| <i>...↔ a</i> | Arbitrary number of additional parameters. |

Matches with explicit template type A.

Definition at line 93 of file [minmax](#).

13.14.2.4 min() [1/2]

```
template<typename A, typename ...  ARGS>
A const & cxx::min (
    A const & a1,
    A const & a2,
    ARGS const &...  a) [constexpr]
```

Get the minimum of a1 and a2 up to aN.

Parameters

| | |
|-------------------|--|
| <i>a1</i> | The first value. |
| <i>a2</i> | The second value. |
| <i>...↔ a</i> | Arbitrary number of additional parameters. |

Matches with automatic argument type deduction.

Definition at line 36 of file [minmax](#).

13.14.2.5 min() [2/2]

```
template<typename A, typename ...  ARGS>
A const & cxx::min (
    cxx::identity_t< A > const & a1,
    cxx::identity_t< A > const & a2,
    ARGS const &...  a) [constexpr]
```

Get the minimum of a1 and a2 up to aN.

Parameters

| | |
|-------------------|--|
| <i>a1</i> | The first value. |
| <i>a2</i> | The second value. |
| <i>...↔ a</i> | Arbitrary number of additional parameters. |

Matches with explicit template type A.

Definition at line 53 of file [minmax](#).

13.14.2.6 operator new()

```
void * operator new (
    size_t ,
    void * mem,
    cxx::Nothrow const & ) [inline], [noexcept]
```

Simple placement new operator.

Parameters

| | |
|------------|--|
| <i>mem</i> | the address of the memory block to place the new object. |
|------------|--|

Returns

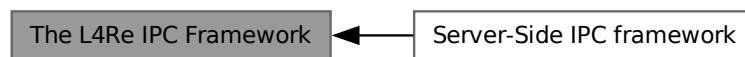
the address given by *mem*.

Definition at line 28 of file [std_alloc](#).

13.15 The L4Re IPC Framework

The mechanisms for IPC communication between [L4Re](#) applications.

Collaboration diagram for The L4Re IPC Framework:

**Topics**

- [Server-Side IPC framework](#) 644
Server-Side framework for implementing object-oriented servers.

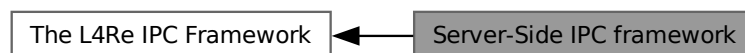
13.15.1 Detailed Description

The mechanisms for IPC communication between [L4Re](#) applications.

13.15.2 Server-Side IPC framework

Server-Side framework for implementing object-oriented servers.

Collaboration diagram for Server-Side IPC framework:



Namespaces

- namespace `L4::lpc_svr`
Helper classes for `L4::Server` instantiation.

Data Structures

- class `L4::lpc_svr::Server_iface`
Interface for server-loop related functions.
- class `L4::Basic_registry`
This registry returns the corresponding server object based on the label of an `lpc_gate`.
- struct `L4::lpc_svr::Ignore_errors`
Mix in for `LOOP_HOOKS` to ignore IPC errors.
- struct `L4::lpc_svr::Default_timeout`
Mix in for `LOOP_HOOKS` to use a 0 send and an infinite receive timeout.
- struct `L4::lpc_svr::Compound_reply`
Mix in for `LOOP_HOOKS` to always use compound reply and wait.
- struct `L4::lpc_svr::Default_setup_wait`
Mix in for `LOOP_HOOKS` for `setup_wait` no op.
- class `L4::lpc_svr::Br_manager_no_buffers`
Empty implementation of `Server_iface`.
- struct `L4::lpc_svr::Default_loop_hooks`
Default `LOOP_HOOKS`.
- class `L4::Server< LOOP_HOOKS >`
Basic server loop for handling client requests.
- class `L4::Server_object`
Abstract server object to be used with `L4::Server` and `L4::Basic_registry`.
- struct `L4::Server_object_t< IFACE, BASE >`
Base class (template) for server implementing server objects.
- struct `L4::Server_object_x< Derived, IFACE, BASE >`
Helper class to implement `p_dispatch` based server objects.
- struct `L4::lrq_handler_object`
Server object base class for handling IRQ messages.
- class `L4::lpc_svr::Timeout`
Callback interface for `Timeout_queue`.
- class `L4::lpc_svr::Timeout_queue`
Timeout queue to be used in `l4re` server loop.
- class `L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >`
Loop hooks mixin for integrating a timeout queue into the server loop.

Enumerations

- enum `L4::lpc_svr::Reply_mode` { `L4::lpc_svr::Reply_compound`, `L4::lpc_svr::Reply_separate` }
Reply mode for server loop.

13.15.2.1 Detailed Description

Server-Side framework for implementing object-oriented servers.

13.15.2.2 Enumeration Type Documentation

13.15.2.2.1 Reply_mode

enum [L4::Ipc_svr::Reply_mode](#)

Reply mode for server loop.

The reply mode specifies if the server loop shall do a compound reply and wait operation ([Reply_compound](#)), which is the most performant method. Note, `setup_wait()` is called before the reply. The other way is to call reply and wait separately and call `setup_wait` in between.

The actual mode is determined by the return value of the `before_reply()` hook in the `LOOP_HOOKS` of [L4::Server](#).

Enumerator

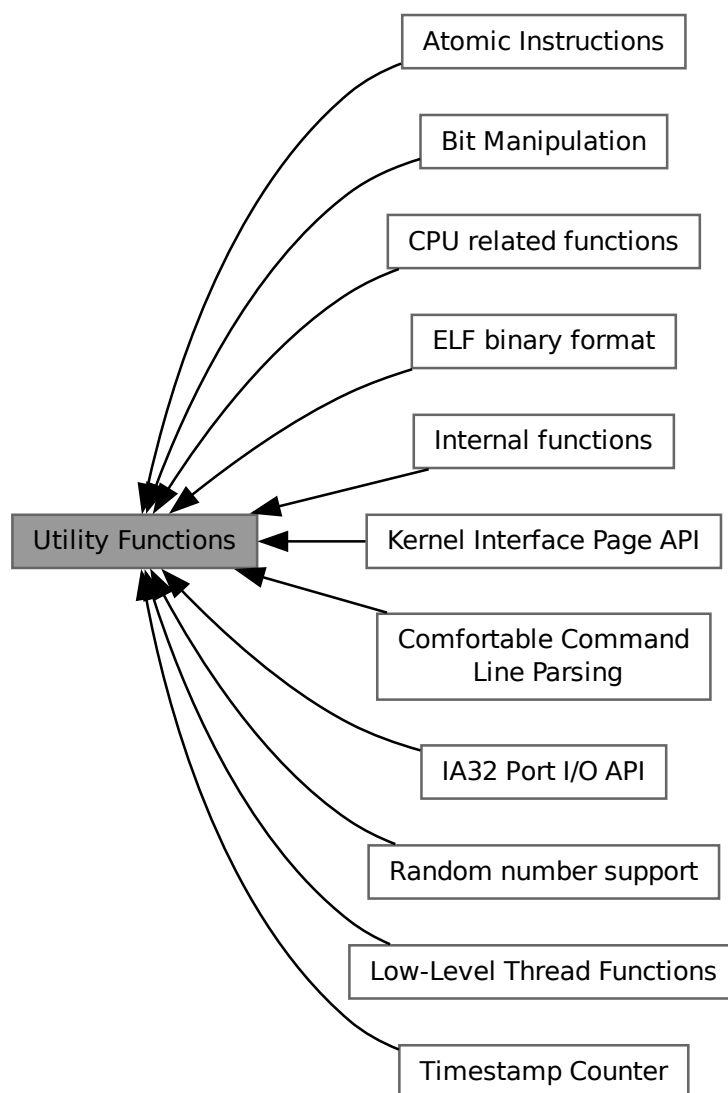
| | |
|----------------|--|
| Reply_compound | Server shall use a compound reply and wait (fast). |
| Reply_separate | Server shall call reply and wait separately. |

Definition at line [46](#) of file [ipc_server_loop](#).

13.16 Utility Functions

Utilities, generic file.

Collaboration diagram for Utility Functions:



Topics

- CPU related functions [653](#)
- IA32 Port I/O API [656](#)
- Timestamp Counter [662](#)
-

| | |
|---|-----|
| Atomic Instructions | 669 |
| • | |
| Internal functions | 687 |
| • | |
| Bit Manipulation | 687 |
| • | |
| ELF binary format | 694 |
| <i>Functions and types related to ELF binaries.</i> | |
| • | |
| Kernel Interface Page API | 719 |
| • | |
| Comfortable Command Line Parsing | 721 |
| • | |
| Random number support | 723 |
| • | |
| Low-Level Thread Functions | 724 |

Files

- file [rand.h](#)
 Simple Pseudo-Random Number Generator.

Functions

- [L4_BEGIN_DECLS](#) long [l4util_splitlog2_hdl](#) ([l4_addr_t](#) start, [l4_addr_t](#) end, long(*handler)([l4_addr_t](#) s, [l4_addr_t](#) e, int log2size))
 Split a range into log2 base and size aligned chunks.
- [l4_addr_t](#) [l4util_splitlog2_size](#) ([l4_addr_t](#) start, [l4_addr_t](#) end)
 Return log2 base and size aligned length of a range.
- [L4_BEGIN_DECLS](#) [l4_timeout_s](#) [l4util_micros2l4to](#) ([l4_uint64_t](#) us) [L4_NOTHROW](#)
 Calculate l4 timeouts.
- void [l4_sleep](#) ([l4_uint32_t](#) ms) [L4_NOTHROW](#)
 Suspend thread for a period of ms milliseconds.
- void [l4_usleep](#) ([l4_uint64_t](#) us) [L4_NOTHROW](#)
 Suspend thread for a period of us microseconds.
- void [l4_sleep_forever](#) (void) [L4_NOTHROW](#) [L4_NORETURN](#)
 Go sleep and never wake up.
- void [l4_touch_ro](#) (const void *addr, unsigned size) [L4_NOTHROW](#)
 Touch data area to force mapping (read-only).
- void [l4_touch_rw](#) (const void *addr, unsigned size) [L4_NOTHROW](#)
 Touch data areas to force mapping (read-write).

13.16.1 Detailed Description

Utilities, generic file.

13.16.2 Function Documentation

13.16.2.1 l4_sleep()

```
void l4_sleep (  
    l4_uint32_t ms)
```

Suspend thread for a period of *ms* milliseconds.

Parameters

| | |
|-----------|----------------------|
| <i>ms</i> | Time in milliseconds |
|-----------|----------------------|

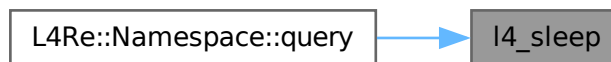
Examples

[examples/libs/libirq/async_isr.c](#), [examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), and [examples/sys/start-with-exc/main.c](#).

References [L4_CV](#), and [L4_NOTHROW](#).

Referenced by [L4Re::Namespace::query\(\)](#).

Here is the caller graph for this function:



13.16.2.2 l4_touch_ro()

```
void l4_touch_ro (  
    const void * addr,  
    unsigned size) [inline]
```

Touch data area to force mapping (read-only).

Parameters

| | |
|-------------|--------------------------------|
| <i>addr</i> | Start of memory area to touch. |
| <i>size</i> | Size of area to touch. |

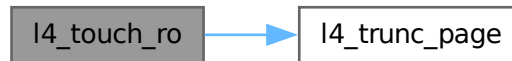
Examples

[examples/sys/singlestep/main.c](#).

Definition at line 92 of file [util.h](#).

References [L4_NOTHROW](#), [L4_PAGESIZE](#), and [l4_trunc_page\(\)](#).

Here is the call graph for this function:



13.16.2.3 l4_touch_rw()

```
void l4_touch_rw (  
    const void * addr,  
    unsigned size) [inline]
```

Touch data areas to force mapping (read-write).

Parameters

| | |
|-------------|--------------------------------|
| <i>addr</i> | Start of memory area to touch. |
| <i>size</i> | Size of area to touch. |

Examples

[examples/sys/aliens/main.c](#), and [examples/sys/singlestep/main.c](#).

Definition at line 105 of file [util.h](#).

References [L4_NOTHROW](#), [L4_PAGESIZE](#), and [l4_trunc_page\(\)](#).

Here is the call graph for this function:



13.16.2.4 l4_usleep()

```
void l4_usleep (  
    l4_uint64_t us)
```

Suspend thread for a period of *us* microseconds.

Parameters

| | |
|-----------|----------------------|
| <i>us</i> | Time in microseconds |
|-----------|----------------------|

Note

The timer resolution of [L4](#) kernels is usually 1ms.

References [L4_CV](#), [L4_INLINE](#), [L4_NORETURN](#), and [L4_NOTHROW](#).

13.16.2.5 l4util_micros2l4to()

```
L4_BEGIN_DECLS l4_timeout_s l4util_micros2l4to (  
    l4_uint64_t us)
```

Calculate l4 timeouts.

Parameters

| | |
|-----------|--|
| <i>us</i> | time in microseconds. Special cases: <ul style="list-style-type: none">• 0 -> timeout 0• ~0U -> timeout NEVER |
|-----------|--|

Returns

the corresponding l4_timeout value

Deprecated Use [l4_timeout_from_us\(\)](#).

References [L4_CV](#), and [L4_NOTHROW](#).

13.16.2.6 l4util_splitlog2_hdl()

```
L4_END_DECLS long l4util_splitlog2_hdl (
    l4_addr_t start,
    l4_addr_t end,
    long(* handler ) (l4_addr_t s, l4_addr_t e, int log2size)) [inline]
```

Split a range into log2 base and size aligned chunks.

Parameters

| | |
|----------------|--|
| <i>start</i> | Start of range |
| <i>end</i> | End of range (inclusive) (e.g. 2-4 is len 3) |
| <i>handler</i> | Handler function that is called with start and end (both inclusive) of the chunk. On success, the handler must return 0, if it returns !=0 the function will immediately return with the return code of the handler. |

Returns

0 on success, != 0 otherwise

Definition at line 51 of file [splitlog2.h](#).

References [L4_EINVAL](#), and [l4util_splitlog2_size\(\)](#).

Here is the call graph for this function:



13.16.2.7 l4util_splitlog2_size()

```
l4_addr_t l4util_splitlog2_size (
    l4_addr_t start,
    l4_addr_t end) [inline]
```

Return log2 base and size aligned length of a range.

Parameters

| | |
|--------------|--|
| <i>start</i> | Start of range |
| <i>end</i> | End of range (inclusive) (e.g. 2-4 is len 3) |

Returns

length of elements in log2size (length is $1 \ll \log_2 \text{size}$)

Definition at line 70 of file [splitlog2.h](#).

Referenced by [l4util_splitlog2_hdl\(\)](#).

Here is the caller graph for this function:

**13.16.3 CPU related functions**

Collaboration diagram for CPU related functions:

**Functions**

- int [l4util_cpu_has_cpuid](#) (void)
Check whether the CPU supports the "cpuid" instruction.
- unsigned int [l4util_cpu_capabilities](#) (void)
Returns the CPU capabilities if the "cpuid" instruction is available.
- unsigned int [l4util_cpu_capabilities_nocheck](#) (void)
Returns the CPU capabilities.
- void **`l4util_cpu_cpuid`** (unsigned long mode, unsigned long *eax, unsigned long *ebx, unsigned long *ecx, unsigned long *edx)
Generic CPUID access function.

13.16.3.1 Detailed Description

13.16.3.2 Function Documentation

13.16.3.2.1 l4util_cpu_capabilities()

```
unsigned int l4util_cpu_capabilities (  
    void ) [inline]
```

Returns the CPU capabilities if the "cpuid" instruction is available.

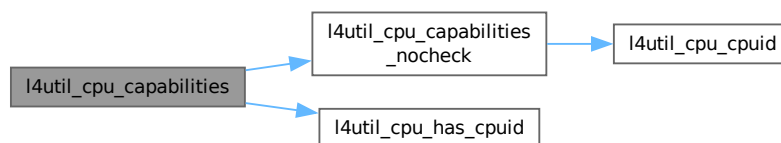
Returns

CPU capabilities if the "cpuid" instruction is available, 0 if the "cpuid" instruction is not supported.

Definition at line 95 of file [cpu.h](#).

References [l4util_cpu_capabilities_nocheck\(\)](#), and [l4util_cpu_has_cpuid\(\)](#).

Here is the call graph for this function:



13.16.3.2.2 l4util_cpu_capabilities_nocheck()

```
unsigned int l4util_cpu_capabilities_nocheck (  
    void ) [inline]
```

Returns the CPU capabilities.

Returns

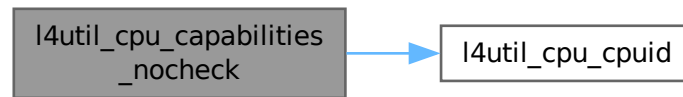
CPU capabilities.

Definition at line 84 of file [cpu.h](#).

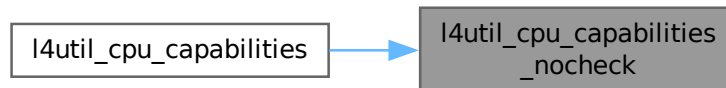
References [l4util_cpu_cpuid\(\)](#).

Referenced by [l4util_cpu_capabilities\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.16.3.2.3 l4util_cpu_has_cpuid()

```
int l4util_cpu_has_cpuid (  
    void ) [inline]
```

Check whether the CPU supports the "cpuid" instruction.

Returns

1 if it has, 0 if it has not

Definition at line 64 of file [cpu.h](#).

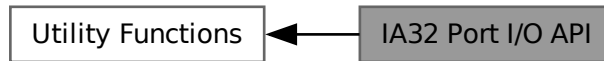
Referenced by [l4util_cpu_capabilities\(\)](#).

Here is the caller graph for this function:



13.16.4 IA32 Port I/O API

Collaboration diagram for IA32 Port I/O API:



Functions

- [l4_uint8_t l4util_in8](#) ([l4_uint16_t](#) port)
Read byte from I/O port.
- [l4_uint16_t l4util_in16](#) ([l4_uint16_t](#) port)
Read 16-bit-value from I/O port.
- [l4_uint32_t l4util_in32](#) ([l4_uint16_t](#) port)
Read 32-bit-value from I/O port.
- void [l4util_ins8](#) ([l4_uint16_t](#) port, [l4_umword_t](#) addr, [l4_umword_t](#) count)
Read a block of 8-bit-values from I/O ports.
- void [l4util_ins16](#) ([l4_uint16_t](#) port, [l4_umword_t](#) addr, [l4_umword_t](#) count)
Read a block of 16-bit-values from I/O ports.
- void [l4util_ins32](#) ([l4_uint16_t](#) port, [l4_umword_t](#) addr, [l4_umword_t](#) count)
Read a block of 32-bit-values from I/O ports.
- void [l4util_out8](#) ([l4_uint8_t](#) value, [l4_uint16_t](#) port)
Write byte to I/O port.
- void [l4util_out16](#) ([l4_uint16_t](#) value, [l4_uint16_t](#) port)
Write 16-bit-value to I/O port.
- void [l4util_out32](#) ([l4_uint32_t](#) value, [l4_uint16_t](#) port)
Write 32-bit-value to I/O port.
- void [l4util_outs8](#) ([l4_uint16_t](#) port, [l4_umword_t](#) addr, [l4_umword_t](#) count)
Write a block of bytes to I/O port.
- void [l4util_outs16](#) ([l4_uint16_t](#) port, [l4_umword_t](#) addr, [l4_umword_t](#) count)
Write a block of 16-bit-values to I/O port.
- void [l4util_outs32](#) ([l4_uint16_t](#) port, [l4_umword_t](#) addr, [l4_umword_t](#) count)
Write block of 32-bit-values to I/O port.
- void [l4util_iodelay](#) (void)
delay I/O port access by writing to port 0x80
- int [l4util_ioport_map](#) ([l4_cap_idx_t](#) sigma0id, unsigned port_start, unsigned log2size)
Map a range of I/O ports.

13.16.4.1 Detailed Description

13.16.4.2 Function Documentation

13.16.4.2.1 l4util_in16()

```
l4_uint16_t l4util_in16 (  
    l4_uint16_t port) [inline]
```

Read 16-bit-value from I/O port.

Parameters

| | |
|-------------|------------------|
| <i>port</i> | I/O port address |
|-------------|------------------|

Returns

value

Definition at line 195 of file [port_io.h](#).

13.16.4.2.2 l4util_in32()

```
l4_uint32_t l4util_in32 (  
    l4_uint16_t port) [inline]
```

Read 32-bit-value from I/O port.

Parameters

| | |
|-------------|------------------|
| <i>port</i> | I/O port address |
|-------------|------------------|

Returns

value

Definition at line 203 of file [port_io.h](#).

13.16.4.2.3 l4util_in8()

```
l4_uint8_t l4util_in8 (  
    l4_uint16_t port) [inline]
```

Read byte from I/O port.

Parameters

| | |
|-------------|------------------|
| <i>port</i> | I/O port address |
|-------------|------------------|

Returns

value

Definition at line 187 of file [port_io.h](#).

13.16.4.2.4 l4util_ins16()

```
void l4util_ins16 (  
    l4_uint16_t port,  
    l4_umword_t addr,  
    l4_umword_t count) [inline]
```

Read a block of 16-bit-values from I/O ports.

Parameters

| | |
|--------------|--------------------------|
| <i>port</i> | I/O port address |
| <i>addr</i> | address of buffer |
| <i>count</i> | number of I/O operations |

Definition at line 220 of file [port_io.h](#).

13.16.4.2.5 l4util_ins32()

```
void l4util_ins32 (  
    l4_uint16_t port,  
    l4_umword_t addr,  
    l4_umword_t count) [inline]
```

Read a block of 32-bit-values from I/O ports.

Parameters

| | |
|-------------|-------------------|
| <i>port</i> | I/O port address |
| <i>addr</i> | address of buffer |

| | |
|--------------|--------------------------|
| <i>count</i> | number of I/O operations |
|--------------|--------------------------|

Definition at line 229 of file [port_io.h](#).

13.16.4.2.6 l4util_ins8()

```
void l4util_ins8 (
    l4_uint16_t port,
    l4_umword_t addr,
    l4_umword_t count) [inline]
```

Read a block of 8-bit-values from I/O ports.

Parameters

| | |
|--------------|--------------------------|
| <i>port</i> | I/O port address |
| <i>addr</i> | address of buffer |
| <i>count</i> | number of I/O operations |

Definition at line 211 of file [port_io.h](#).

13.16.4.2.7 l4util_ioport_map()

```
int l4util_ioport_map (
    l4_cap_idx_t sigma0id,
    unsigned port_start,
    unsigned log2size) [inline]
```

Map a range of I/O ports.

Parameters

| | |
|-------------------|-------------------------------|
| <i>sigma0id</i> | I/O port service (sigma0). |
| <i>port_start</i> | (Start) Port to request. |
| <i>log2size</i> | Log2size of range to request. |

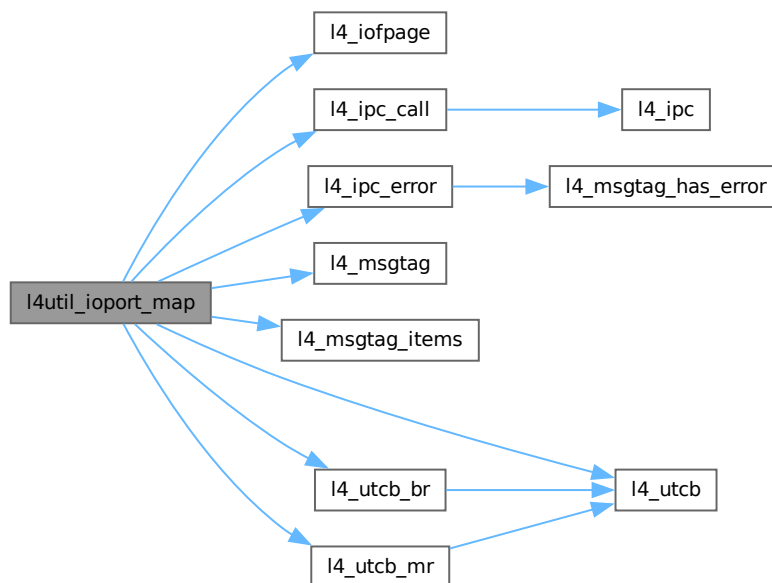
Returns

IPC result: 0 if the range could be successfully mapped on error: IPC failure, or -L4_ENOENT if nothing mapped

Definition at line 289 of file [port_io.h](#).

References [l4_buf_regs_t::bdr](#), [l4_buf_regs_t::br](#), [L4_ENOENT](#), [l4_iofpage\(\)](#), [l4_ipc_call\(\)](#), [l4_ipc_error\(\)](#), [L4_IPC_NEVER](#), [L4_ITEM_MAP](#), [l4_msgtag\(\)](#), [l4_msgtag_items\(\)](#), [L4_PROTO_IO_PAGE_FAULT](#), [l4_utcb\(\)](#), [l4_utcb_br\(\)](#), [l4_utcb_mr\(\)](#), [l4_msg_regs_t::mr](#), and [l4_fpage_t::raw](#).

Here is the call graph for this function:



13.16.4.2.8 l4util_out16()

```
void l4util_out16 (
    l4_uint16_t value,
    l4_uint16_t port) [inline]
```

Write 16-bit-value to I/O port.

Parameters

| | |
|--------------|------------------|
| <i>port</i> | I/O port address |
| <i>value</i> | value to write |

Definition at line 244 of file [port_io.h](#).

13.16.4.2.9 l4util_out32()

```
void l4util_out32 (
    l4_uint32_t value,
    l4_uint16_t port) [inline]
```

Write 32-bit-value to I/O port.

Parameters

| | |
|--------------|------------------|
| <i>port</i> | I/O port address |
| <i>value</i> | value to write |

Definition at line 250 of file [port_io.h](#).

13.16.4.2.10 l4util_out8()

```
void l4util_out8 (  
    l4_uint8_t value,  
    l4_uint16_t port) [inline]
```

Write byte to I/O port.

Parameters

| | |
|--------------|------------------|
| <i>port</i> | I/O port address |
| <i>value</i> | value to write |

Definition at line 238 of file [port_io.h](#).

13.16.4.2.11 l4util_outs16()

```
void l4util_outs16 (  
    l4_uint16_t port,  
    l4_umword_t addr,  
    l4_umword_t count) [inline]
```

Write a block of 16-bit-values to I/O port.

Parameters

| | |
|--------------|--------------------------|
| <i>port</i> | I/O port address |
| <i>addr</i> | address of buffer |
| <i>count</i> | number of I/O operations |

Definition at line 265 of file [port_io.h](#).

13.16.4.2.12 l4util_outs32()

```
void l4util_outs32 (
    l4_uint16_t port,
    l4_umword_t addr,
    l4_umword_t count) [inline]
```

Write block of 32-bit-values to I/O port.

Parameters

| | |
|--------------|--------------------------|
| <i>port</i> | I/O port address |
| <i>addr</i> | address of buffer |
| <i>count</i> | number of I/O operations |

Definition at line 274 of file [port_io.h](#).

13.16.4.2.13 l4util_outs8()

```
void l4util_outs8 (
    l4_uint16_t port,
    l4_umword_t addr,
    l4_umword_t count) [inline]
```

Write a block of bytes to I/O port.

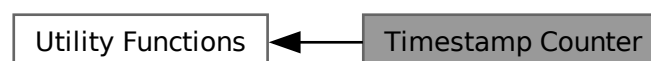
Parameters

| | |
|--------------|--------------------------|
| <i>port</i> | I/O port address |
| <i>addr</i> | address of buffer |
| <i>count</i> | number of I/O operations |

Definition at line 256 of file [port_io.h](#).

13.16.5 Timestamp Counter

Collaboration diagram for Timestamp Counter:



Files

- file [rdtsc.h](#)
Timestamp counter related functions.
- file [rdtsc.h](#)
Timestamp counter related functions.

Functions

- [l4_cpu_time_t l4_rdtsc](#) (void)
Read current value of CPU-internal timestamp counter.
- [l4_uint32_t l4_rdtsc_32](#) (void)
Read the least significant 32 bit of the TSC.
- [l4_uint64_t l4_rdpmc](#) (int ecx)
Return current value of CPU-internal performance measurement counter.
- [l4_uint32_t l4_rdpmc_32](#) (int ecx)
Return the least significant 32 bit of a performance counter.
- [l4_uint64_t l4_tsc_to_ns](#) ([l4_cpu_time_t](#) tsc)
Convert timestamp to ns value.
- [l4_uint64_t l4_tsc_to_us](#) ([l4_cpu_time_t](#) tsc)
Convert timestamp into micro seconds value.
- void [l4_tsc_to_s_and_ns](#) ([l4_cpu_time_t](#) tsc, [l4_uint32_t](#) *s, [l4_uint32_t](#) *ns)
Convert timestamp to s.ns value.
- [l4_cpu_time_t l4_ns_to_tsc](#) ([l4_uint64_t](#) ns)
Convert nano seconds into CPU ticks.
- void [l4_busy_wait_ns](#) ([l4_uint64_t](#) ns)
Wait busy for a small amount of time.
- void [l4_busy_wait_us](#) ([l4_uint64_t](#) us)
Wait busy for a small amount of time.
- [l4_uint32_t l4_calibrate_tsc](#) ([l4_kernel_info_t](#) const *kip)
Determine scalers for timestamp calculations.
- [l4_uint32_t l4_tsc_init](#) ([l4_kernel_info_t](#) const *kip)
Initialize scaler for TSC calibrations from the kernel.
- [l4_uint32_t l4_get_hz](#) (void)
Get CPU frequency in Hz.

13.16.5.1 Detailed Description

13.16.5.2 Function Documentation

13.16.5.2.1 [l4_busy_wait_ns\(\)](#)

```
void l4_busy_wait_ns (
    l4\_uint64\_t ns) [inline]
```

Wait busy for a small amount of time.

Parameters

| | |
|-----------|----------------------|
| <i>ns</i> | nano seconds to wait |
|-----------|----------------------|

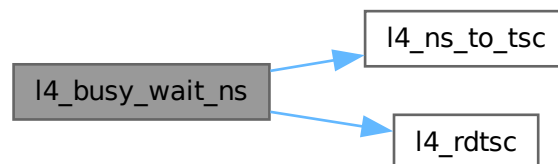
Attention

Not intended for any use!

Definition at line 262 of file [rdtsc.h](#).

References [l4_ns_to_tsc\(\)](#), and [l4_rdtsc\(\)](#).

Here is the call graph for this function:



13.16.5.2.2 l4_busy_wait_us()

```
void l4_busy_wait_us (  
    l4_uint64_t us) [inline]
```

Wait busy for a small amount of time.

Parameters

| | |
|-----------|-----------------------|
| <i>us</i> | micro seconds to wait |
|-----------|-----------------------|

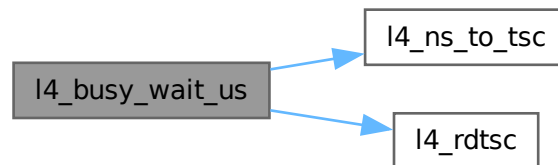
Attention

Not intended for any use!

Definition at line 272 of file [rdtsc.h](#).

References [l4_ns_to_tsc\(\)](#), and [l4_rdtsc\(\)](#).

Here is the call graph for this function:



13.16.5.2.3 l4_calibrate_tsc()

```
l4_uint32_t l4_calibrate_tsc (  
    l4_kernel_info_t const * kip) [inline]
```

Determine scalars for timestamp calculations.

Determine some scalars to be able to convert between real time and CPU ticks. Just calls [l4_tsc_init\(\)](#).

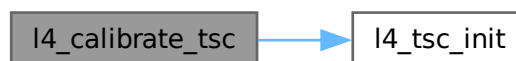
Examples

[examples/sys/aliens/main.c](#).

Definition at line 159 of file [rdtsc.h](#).

References [l4_tsc_init\(\)](#).

Here is the call graph for this function:



13.16.5.2.4 l4_get_hz()

```
l4_uint32_t l4_get_hz (  
    void )
```

Get CPU frequency in Hz.

Returns

frequency in Hz

References [L4_END_DECLS](#), and [L4_INLINE](#).

13.16.5.2.5 l4_ns_to_tsc()

```
l4_cpu_time_t l4_ns_to_tsc (  
    l4_uint64_t ns) [inline]
```

Convert nano seconds into CPU ticks.

Parameters

| | |
|-----------|--------------|
| <i>ns</i> | nano seconds |
|-----------|--------------|

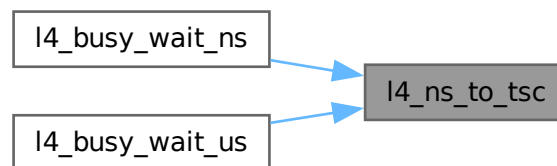
Returns

CPU ticks

Definition at line 248 of file [rdtsc.h](#).

Referenced by [l4_busy_wait_ns\(\)](#), and [l4_busy_wait_us\(\)](#).

Here is the caller graph for this function:



13.16.5.2.6 l4_rdpmc()

```
l4_uint64_t l4_rdpmc (  
    int ecx) [inline]
```

Return current value of CPU-internal performance measurement counter.

Parameters

| | |
|------------|---|
| <i>ecx</i> | ECX value for the rdpmc instruction. For details see the Intel IA-32 Architectures Software Developer's Manual. |
|------------|---|

Returns

64-bit PMC

Definition at line 175 of file [rdtsc.h](#).

13.16.5.2.7 l4_rdpmc_32()

```
l4_uint32_t l4_rdpmc_32 (  
    int ecx) [inline]
```

Return the least significant 32 bit of a performance counter.

Useful for smaller differences, needs less cycles.

Definition at line 195 of file [rdtsc.h](#).

13.16.5.2.8 l4_rdtsc()

```
l4_cpu_time_t l4_rdtsc (  
    void ) [inline]
```

Read current value of CPU-internal timestamp counter.

Returns

64-bit timestamp

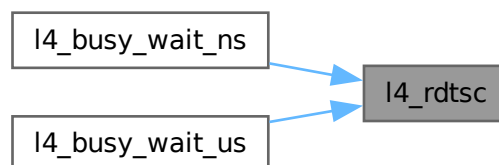
Examples

[examples/sys/aliens/main.c](#).

Definition at line 165 of file [rdtsc.h](#).

Referenced by [l4_busy_wait_ns\(\)](#), and [l4_busy_wait_us\(\)](#).

Here is the caller graph for this function:



13.16.5.2.9 l4_rdtsc_32()

```
l4_uint32_t l4_rdtsc_32 (  
    void ) [inline]
```

Read the least significant 32 bit of the TSC.

Useful for smaller differences, needs less cycles.

Definition at line 185 of file [rdtsc.h](#).

13.16.5.2.10 l4_tsc_init()

```
l4_uint32_t l4_tsc_init (
    l4_kernel_info_t const * kip)
```

Initialize scaler for TSC calibrations from the kernel.

Initialize the scalers needed by [l4_tsc_to_ns\(\)](#)/[l4_ns_to_tsc\(\)](#) and so on. Use the kernel-provided frequency.

Parameters

| | |
|------------|-------------|
| <i>kip</i> | KIP pointer |
|------------|-------------|

Returns

0 on error (no scalers exported by kernel) otherwise returns ($2^{32} / (\text{tsc per } \mu\text{sec})$). This value has the same semantics as the value returned by the `calibrate_delay_loop()` function of the Linux kernel.

References [L4_CV](#).

Referenced by [l4_calibrate_tsc\(\)](#).

Here is the caller graph for this function:



13.16.5.2.11 l4_tsc_to_ns()

```
l4_uint64_t l4_tsc_to_ns (
    l4_cpu_time_t tsc) [inline]
```

Convert timestamp to ns value.

Parameters

| | |
|------------|-------------------------|
| <i>tsc</i> | time value in CPU ticks |
|------------|-------------------------|

Returns

time value in ns

Examples

[examples/sys/aliens/main.c](#).

Definition at line 205 of file [rdtsc.h](#).

13.16.5.2.12 l4_tsc_to_s_and_ns()

```
void l4_tsc_to_s_and_ns (
    l4_cpu_time_t tsc,
    l4_uint32_t * s,
    l4_uint32_t * ns) [inline]
```

Convert timestamp to s.ns value.

Parameters

| | | |
|-----|------------|-------------------------|
| | <i>tsc</i> | time value in CPU ticks |
| out | <i>s</i> | seconds |
| out | <i>ns</i> | nano seconds |

Definition at line 233 of file [rdtsc.h](#).

13.16.5.2.13 l4_tsc_to_us()

```
l4_uint64_t l4_tsc_to_us (
    l4_cpu_time_t tsc) [inline]
```

Convert timestamp into micro seconds value.

Parameters

| | |
|------------|-------------------------|
| <i>tsc</i> | time value in CPU ticks |
|------------|-------------------------|

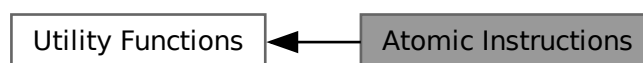
Returns

time value in micro seconds

Definition at line 219 of file [rdtsc.h](#).

13.16.6 Atomic Instructions

Collaboration diagram for Atomic Instructions:



Files

- file [atomic.h](#)
atomic operations header and generic implementations

Functions

- int [l4util_cmpxchg32](#) (volatile [l4_uint32_t](#) *dest, [l4_uint32_t](#) cmp_val, [l4_uint32_t](#) new_val)
Atomic compare and exchange (32 bit version).
- int [l4util_cmpxchg16](#) (volatile [l4_uint16_t](#) *dest, [l4_uint16_t](#) cmp_val, [l4_uint16_t](#) new_val)
Atomic compare and exchange (16 bit version).
- int [l4util_cmpxchg8](#) (volatile [l4_uint8_t](#) *dest, [l4_uint8_t](#) cmp_val, [l4_uint8_t](#) new_val)
Atomic compare and exchange (8 bit version).
- int [l4util_cmpxchg](#) (volatile [l4_umword_t](#) *dest, [l4_umword_t](#) cmp_val, [l4_umword_t](#) new_val)
Atomic compare and exchange (machine wide fields).
- [l4_uint32_t](#) [l4util_xchg32](#) (volatile [l4_uint32_t](#) *dest, [l4_uint32_t](#) val)
Atomic exchange (32 bit version).
- [l4_uint16_t](#) [l4util_xchg16](#) (volatile [l4_uint16_t](#) *dest, [l4_uint16_t](#) val)
Atomic exchange (16 bit version).
- [l4_uint8_t](#) [l4util_xchg8](#) (volatile [l4_uint8_t](#) *dest, [l4_uint8_t](#) val)
Atomic exchange (8 bit version).
- [l4_umword_t](#) [l4util_xchg](#) (volatile [l4_umword_t](#) *dest, [l4_umword_t](#) val)
Atomic exchange (machine wide fields).
- void [l4util_atomic_add](#) (volatile long *dest, long val)
Atomic add.
- void [l4util_atomic_inc](#) (volatile long *dest)
Atomic increment.

Atomic add/sub/and/or (8,16,32 bit version) without result

- void [l4util_add8](#) (volatile [l4_uint8_t](#) *dest, [l4_uint8_t](#) val)
- void [l4util_add16](#) (volatile [l4_uint16_t](#) *dest, [l4_uint16_t](#) val)
- void [l4util_add32](#) (volatile [l4_uint32_t](#) *dest, [l4_uint32_t](#) val)
- void [l4util_sub8](#) (volatile [l4_uint8_t](#) *dest, [l4_uint8_t](#) val)
- void [l4util_sub16](#) (volatile [l4_uint16_t](#) *dest, [l4_uint16_t](#) val)
- void [l4util_sub32](#) (volatile [l4_uint32_t](#) *dest, [l4_uint32_t](#) val)
- void [l4util_and8](#) (volatile [l4_uint8_t](#) *dest, [l4_uint8_t](#) val)
- void [l4util_and16](#) (volatile [l4_uint16_t](#) *dest, [l4_uint16_t](#) val)
- void [l4util_and32](#) (volatile [l4_uint32_t](#) *dest, [l4_uint32_t](#) val)
- void [l4util_or8](#) (volatile [l4_uint8_t](#) *dest, [l4_uint8_t](#) val)
- void [l4util_or16](#) (volatile [l4_uint16_t](#) *dest, [l4_uint16_t](#) val)
- void [l4util_or32](#) (volatile [l4_uint32_t](#) *dest, [l4_uint32_t](#) val)

Atomic add/sub/and/or operations (8,16,32 bit) with result

- [l4_uint8_t](#) [l4util_add8_res](#) (volatile [l4_uint8_t](#) *dest, [l4_uint8_t](#) val)
- [l4_uint16_t](#) [l4util_add16_res](#) (volatile [l4_uint16_t](#) *dest, [l4_uint16_t](#) val)
- [l4_uint32_t](#) [l4util_add32_res](#) (volatile [l4_uint32_t](#) *dest, [l4_uint32_t](#) val)
- [l4_uint8_t](#) [l4util_sub8_res](#) (volatile [l4_uint8_t](#) *dest, [l4_uint8_t](#) val)
- [l4_uint16_t](#) [l4util_sub16_res](#) (volatile [l4_uint16_t](#) *dest, [l4_uint16_t](#) val)
- [l4_uint32_t](#) [l4util_sub32_res](#) (volatile [l4_uint32_t](#) *dest, [l4_uint32_t](#) val)
- [l4_uint8_t](#) [l4util_and8_res](#) (volatile [l4_uint8_t](#) *dest, [l4_uint8_t](#) val)
- [l4_uint16_t](#) [l4util_and16_res](#) (volatile [l4_uint16_t](#) *dest, [l4_uint16_t](#) val)
- [l4_uint32_t](#) [l4util_and32_res](#) (volatile [l4_uint32_t](#) *dest, [l4_uint32_t](#) val)
- [l4_uint8_t](#) [l4util_or8_res](#) (volatile [l4_uint8_t](#) *dest, [l4_uint8_t](#) val)
- [l4_uint16_t](#) [l4util_or16_res](#) (volatile [l4_uint16_t](#) *dest, [l4_uint16_t](#) val)
- [l4_uint32_t](#) [l4util_or32_res](#) (volatile [l4_uint32_t](#) *dest, [l4_uint32_t](#) val)

Atomic inc/dec (8,16,32 bit) without result

- void [l4util_inc8](#) (volatile [l4_uint8_t](#) *dest)
- void [l4util_inc16](#) (volatile [l4_uint16_t](#) *dest)
- void [l4util_inc32](#) (volatile [l4_uint32_t](#) *dest)
- void [l4util_dec8](#) (volatile [l4_uint8_t](#) *dest)
- void [l4util_dec16](#) (volatile [l4_uint16_t](#) *dest)
- void [l4util_dec32](#) (volatile [l4_uint32_t](#) *dest)

Atomic inc/dec (8,16,32 bit) with result

- [l4_uint8_t](#) [l4util_inc8_res](#) (volatile [l4_uint8_t](#) *dest)
- [l4_uint16_t](#) [l4util_inc16_res](#) (volatile [l4_uint16_t](#) *dest)
- [l4_uint32_t](#) [l4util_inc32_res](#) (volatile [l4_uint32_t](#) *dest)
- [l4_uint8_t](#) [l4util_dec8_res](#) (volatile [l4_uint8_t](#) *dest)
- [l4_uint16_t](#) [l4util_dec16_res](#) (volatile [l4_uint16_t](#) *dest)
- [l4_uint32_t](#) [l4util_dec32_res](#) (volatile [l4_uint32_t](#) *dest)

13.16.6.1 Detailed Description**13.16.6.2 Function Documentation****13.16.6.2.1 l4util_add16()**

```
void l4util_add16 (
    volatile l4\_uint16\_t * dest,
    l4\_uint16\_t val) [inline]
```

Parameters

| | |
|-------------|-------------------------|
| <i>dest</i> | destination operand |
| <i>val</i> | value to add/sub/and/or |

Definition at line [472](#) of file [atomic.h](#).

13.16.6.2.2 l4util_add16_res()

```
l4\_uint16\_t l4util_add16_res (
    volatile l4\_uint16\_t * dest,
    l4\_uint16\_t val) [inline]
```

Parameters

| | |
|-------------|-------------------------|
| <i>dest</i> | destination operand |
| <i>val</i> | value to add/sub/and/or |

Returns

res

Definition at line [524](#) of file [atomic.h](#).

13.16.6.2.3 l4util_add32()

```
void l4util_add32 (  
    volatile l4_uint32_t * dest,  
    l4_uint32_t val) [inline]
```

Parameters

| | |
|-------------|-------------------------|
| <i>dest</i> | destination operand |
| <i>val</i> | value to add/sub/and/or |

Definition at line 476 of file [atomic.h](#).

13.16.6.2.4 l4util_add32_res()

```
l4_uint32_t l4util_add32_res (  
    volatile l4_uint32_t * dest,  
    l4_uint32_t val) [inline]
```

Parameters

| | |
|-------------|-------------------------|
| <i>dest</i> | destination operand |
| <i>val</i> | value to add/sub/and/or |

Returns

res

Definition at line 528 of file [atomic.h](#).

13.16.6.2.5 l4util_add8()

```
void l4util_add8 (  
    volatile l4_uint8_t * dest,  
    l4_uint8_t val) [inline]
```

Parameters

| | |
|-------------|-------------------------|
| <i>dest</i> | destination operand |
| <i>val</i> | value to add/sub/and/or |

Definition at line 468 of file [atomic.h](#).

13.16.6.2.6 l4util_add8_res()

```
l4_uint8_t l4util_add8_res (
    volatile l4_uint8_t * dest,
    l4_uint8_t val) [inline]
```

Parameters

| | |
|-------------|-------------------------|
| <i>dest</i> | destination operand |
| <i>val</i> | value to add/sub/and/or |

Returns

res

Definition at line 520 of file [atomic.h](#).

13.16.6.2.7 l4util_and16()

```
void l4util_and16 (
    volatile l4_uint16_t * dest,
    l4_uint16_t val) [inline]
```

Parameters

| | |
|-------------|-------------------------|
| <i>dest</i> | destination operand |
| <i>val</i> | value to add/sub/and/or |

Definition at line 500 of file [atomic.h](#).

13.16.6.2.8 l4util_and16_res()

```
l4_uint16_t l4util_and16_res (
    volatile l4_uint16_t * dest,
    l4_uint16_t val) [inline]
```

Parameters

| | |
|-------------|-------------------------|
| <i>dest</i> | destination operand |
| <i>val</i> | value to add/sub/and/or |

Returns

res

Definition at line 548 of file [atomic.h](#).

13.16.6.2.9 l4util_and32()

```
void l4util_and32 (  
    volatile l4_uint32_t * dest,  
    l4_uint32_t val) [inline]
```

Parameters

| | |
|-------------|-------------------------|
| <i>dest</i> | destination operand |
| <i>val</i> | value to add/sub/and/or |

Definition at line 504 of file [atomic.h](#).

13.16.6.2.10 l4util_and32_res()

```
l4_uint32_t l4util_and32_res (  
    volatile l4_uint32_t * dest,  
    l4_uint32_t val) [inline]
```

Parameters

| | |
|-------------|-------------------------|
| <i>dest</i> | destination operand |
| <i>val</i> | value to add/sub/and/or |

Returns

res

Definition at line 552 of file [atomic.h](#).

13.16.6.2.11 l4util_and8()

```
void l4util_and8 (  
    volatile l4_uint8_t * dest,  
    l4_uint8_t val) [inline]
```

Parameters

| | |
|-------------|-------------------------|
| <i>dest</i> | destination operand |
| <i>val</i> | value to add/sub/and/or |

Definition at line 496 of file [atomic.h](#).

13.16.6.2.12 l4util_and8_res()

```
l4_uint8_t l4util_and8_res (
    volatile l4_uint8_t * dest,
    l4_uint8_t val) [inline]
```

Parameters

| | |
|-------------|-------------------------|
| <i>dest</i> | destination operand |
| <i>val</i> | value to add/sub/and/or |

Returns

res

Definition at line 544 of file [atomic.h](#).

13.16.6.2.13 l4util_atomic_add()

```
void l4util_atomic_add (
    volatile long * dest,
    long val) [inline]
```

Atomic add.

Parameters

| | |
|-------------|---------------------|
| <i>dest</i> | destination operand |
| <i>val</i> | value to add |

Definition at line 480 of file [atomic.h](#).

13.16.6.2.14 l4util_atomic_inc()

```
void l4util_atomic_inc (
    volatile long * dest) [inline]
```

Atomic increment.

Parameters

| | |
|-------------|---------------------|
| <i>dest</i> | destination operand |
|-------------|---------------------|

Definition at line 423 of file [atomic.h](#).

13.16.6.2.15 l4util_cmpxchg()

```
int l4util_cmpxchg (  
    volatile l4_umword_t * dest,  
    l4_umword_t cmp_val,  
    l4_umword_t new_val) [inline]
```

Atomic compare and exchange (machine wide fields).

Parameters

| | |
|----------------|---------------------|
| <i>dest</i> | destination operand |
| <i>cmp_val</i> | compare value |
| <i>new_val</i> | new value for dest |

Returns

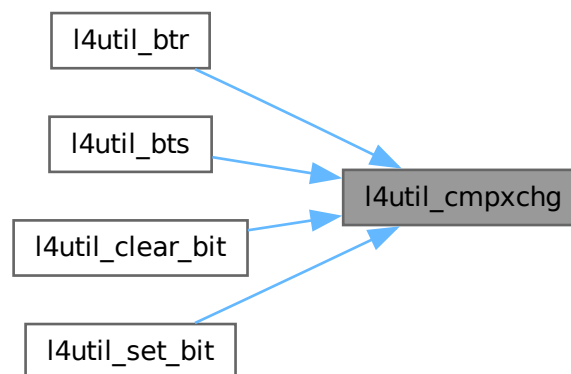
0 if comparison failed, 1 otherwise

Compare the value in *dest* with *cmp_val*, if equal set *dest* to *new_val*

Definition at line 379 of file [atomic.h](#).

Referenced by [l4util_btr\(\)](#), [l4util_bts\(\)](#), [l4util_clear_bit\(\)](#), and [l4util_set_bit\(\)](#).

Here is the caller graph for this function:



13.16.6.2.16 l4util_cmpxchg16()

```
int l4util_cmpxchg16 (  
    volatile l4_uint16_t * dest,  
    l4_uint16_t cmp_val,  
    l4_uint16_t new_val) [inline]
```

Atomic compare and exchange (16 bit version).

Parameters

| | |
|----------------|---------------------|
| <i>dest</i> | destination operand |
| <i>cmp_val</i> | compare value |
| <i>new_val</i> | new value for dest |

Returns

0 if comparison failed, !=0 otherwise

Compare the value in *dest* with *cmp_val*, if equal set *dest* to *new_val*

Definition at line 363 of file [atomic.h](#).

13.16.6.2.17 l4util_cmpxchg32()

```
L4_END_DECLS int l4util_cmpxchg32 (  
    volatile l4_uint32_t * dest,  
    l4_uint32_t cmp_val,  
    l4_uint32_t new_val) [inline]
```

Atomic compare and exchange (32 bit version).

Parameters

| | |
|----------------|---------------------|
| <i>dest</i> | destination operand |
| <i>cmp_val</i> | compare value |
| <i>new_val</i> | new value for dest |

Returns

0 if comparison failed, !=0 otherwise

Compare the value in *dest* with *cmp_val*, if equal set *dest* to *new_val*

Definition at line 355 of file [atomic.h](#).

13.16.6.2.18 l4util_cmpxchg8()

```
int l4util_cmpxchg8 (  
    volatile l4_uint8_t * dest,  
    l4_uint8_t cmp_val,  
    l4_uint8_t new_val) [inline]
```

Atomic compare and exchange (8 bit version).

Parameters

| | |
|----------------|---------------------|
| <i>dest</i> | destination operand |
| <i>cmp_val</i> | compare value |
| <i>new_val</i> | new value for dest |

Returns

0 if comparison failed, !=0 otherwise

Compare the value in *dest* with *cmp_val*, if equal set *dest* to *new_val*

Definition at line 371 of file [atomic.h](#).

13.16.6.2.19 l4util_dec16()

```
void l4util_dec16 (  
    volatile l4_uint16_t * dest) [inline]
```

Parameters

| | |
|-------------|---------------------|
| <i>dest</i> | destination operand |
|-------------|---------------------|

Definition at line 431 of file [atomic.h](#).

13.16.6.2.20 l4util_dec16_res()

```
l4_uint16_t l4util_dec16_res (  
    volatile l4_uint16_t * dest) [inline]
```

Parameters

| | |
|-------------|---------------------|
| <i>dest</i> | destination operand |
|-------------|---------------------|

Returns

res

Definition at line 456 of file [atomic.h](#).

13.16.6.2.21 l4util_dec32()

```
void l4util_dec32 (
    volatile l4_uint32_t * dest) [inline]
```

Parameters

| | |
|-------------|---------------------|
| <i>dest</i> | destination operand |
|-------------|---------------------|

Definition at line 435 of file [atomic.h](#).

13.16.6.2.22 l4util_dec32_res()

```
l4_uint32_t l4util_dec32_res (
    volatile l4_uint32_t * dest) [inline]
```

Parameters

| | |
|-------------|---------------------|
| <i>dest</i> | destination operand |
|-------------|---------------------|

Returns

res

Definition at line 460 of file [atomic.h](#).

13.16.6.2.23 l4util_dec8()

```
void l4util_dec8 (
    volatile l4_uint8_t * dest) [inline]
```

Parameters

| | |
|-------------|---------------------|
| <i>dest</i> | destination operand |
|-------------|---------------------|

Definition at line 427 of file [atomic.h](#).

13.16.6.2.24 l4util_dec8_res()

```
l4_uint8_t l4util_dec8_res (
    volatile l4_uint8_t * dest) [inline]
```

Parameters

| | |
|-------------|---------------------|
| <i>dest</i> | destination operand |
|-------------|---------------------|

Returns

res

Definition at line 452 of file [atomic.h](#).

13.16.6.2.25 l4util_inc16()

```
void l4util_inc16 (  
    volatile l4_uint16_t * dest) [inline]
```

Parameters

| | |
|-------------|---------------------|
| <i>dest</i> | destination operand |
|-------------|---------------------|

Definition at line 415 of file [atomic.h](#).

13.16.6.2.26 l4util_inc16_res()

```
l4_uint16_t l4util_inc16_res (  
    volatile l4_uint16_t * dest) [inline]
```

Parameters

| | |
|-------------|---------------------|
| <i>dest</i> | destination operand |
|-------------|---------------------|

Returns

res

Definition at line 444 of file [atomic.h](#).

13.16.6.2.27 l4util_inc32()

```
void l4util_inc32 (  
    volatile l4_uint32_t * dest) [inline]
```

Parameters

| | |
|-------------|---------------------|
| <i>dest</i> | destination operand |
|-------------|---------------------|

Definition at line 419 of file [atomic.h](#).

13.16.6.2.28 l4util_inc32_res()

```
l4_uint32_t l4util_inc32_res (  
    volatile l4_uint32_t * dest) [inline]
```

Parameters

| | |
|-------------|---------------------|
| <i>dest</i> | destination operand |
|-------------|---------------------|

Returns

res

Definition at line 448 of file [atomic.h](#).**13.16.6.2.29 l4util_inc8()**

```
void l4util_inc8 (  
    volatile l4_uint8_t * dest) [inline]
```

Parameters

| | |
|-------------|---------------------|
| <i>dest</i> | destination operand |
|-------------|---------------------|

Definition at line 411 of file [atomic.h](#).**13.16.6.2.30 l4util_inc8_res()**

```
l4_uint8_t l4util_inc8_res (  
    volatile l4_uint8_t * dest) [inline]
```

Parameters

| | |
|-------------|---------------------|
| <i>dest</i> | destination operand |
|-------------|---------------------|

Returns

res

Definition at line 440 of file [atomic.h](#).**13.16.6.2.31 l4util_or16()**

```
void l4util_or16 (  
    volatile l4_uint16_t * dest,  
    l4_uint16_t val) [inline]
```

Parameters

| | |
|-------------|---------------------|
| <i>dest</i> | destination operand |
|-------------|---------------------|

| | |
|------------|-------------------------|
| <i>val</i> | value to add/sub/and/or |
|------------|-------------------------|

Definition at line 512 of file [atomic.h](#).

13.16.6.2.32 l4util_or16_res()

```
l4_uint16_t l4util_or16_res (  
    volatile l4_uint16_t * dest,  
    l4_uint16_t val) [inline]
```

Parameters

| | |
|-------------|-------------------------|
| <i>dest</i> | destination operand |
| <i>val</i> | value to add/sub/and/or |

Returns

res

Definition at line 560 of file [atomic.h](#).

13.16.6.2.33 l4util_or32()

```
void l4util_or32 (  
    volatile l4_uint32_t * dest,  
    l4_uint32_t val) [inline]
```

Parameters

| | |
|-------------|-------------------------|
| <i>dest</i> | destination operand |
| <i>val</i> | value to add/sub/and/or |

Definition at line 516 of file [atomic.h](#).

13.16.6.2.34 l4util_or32_res()

```
l4_uint32_t l4util_or32_res (  
    volatile l4_uint32_t * dest,  
    l4_uint32_t val) [inline]
```

Parameters

| | |
|-------------|-------------------------|
| <i>dest</i> | destination operand |
| <i>val</i> | value to add/sub/and/or |

Returns

res

Definition at line 564 of file [atomic.h](#).

13.16.6.2.35 l4util_or8()

```
void l4util_or8 (
    volatile l4_uint8_t * dest,
    l4_uint8_t val) [inline]
```

Parameters

| | |
|-------------|-------------------------|
| <i>dest</i> | destination operand |
| <i>val</i> | value to add/sub/and/or |

Definition at line 508 of file [atomic.h](#).

13.16.6.2.36 l4util_or8_res()

```
l4_uint8_t l4util_or8_res (
    volatile l4_uint8_t * dest,
    l4_uint8_t val) [inline]
```

Parameters

| | |
|-------------|-------------------------|
| <i>dest</i> | destination operand |
| <i>val</i> | value to add/sub/and/or |

Returns

res

Definition at line 556 of file [atomic.h](#).

13.16.6.2.37 l4util_sub16()

```
void l4util_sub16 (
    volatile l4_uint16_t * dest,
    l4_uint16_t val) [inline]
```

Parameters

| | |
|-------------|-------------------------|
| <i>dest</i> | destination operand |
| <i>val</i> | value to add/sub/and/or |

Definition at line 488 of file [atomic.h](#).

13.16.6.2.38 l4util_sub16_res()

```
l4_uint16_t l4util_sub16_res (  
    volatile l4_uint16_t * dest,  
    l4_uint16_t val) [inline]
```

Parameters

| | |
|-------------|-------------------------|
| <i>dest</i> | destination operand |
| <i>val</i> | value to add/sub/and/or |

Returns

res

Definition at line 536 of file [atomic.h](#).

13.16.6.2.39 l4util_sub32()

```
void l4util_sub32 (  
    volatile l4_uint32_t * dest,  
    l4_uint32_t val) [inline]
```

Parameters

| | |
|-------------|-------------------------|
| <i>dest</i> | destination operand |
| <i>val</i> | value to add/sub/and/or |

Definition at line 492 of file [atomic.h](#).

13.16.6.2.40 l4util_sub32_res()

```
l4_uint32_t l4util_sub32_res (  
    volatile l4_uint32_t * dest,  
    l4_uint32_t val) [inline]
```

Parameters

| | |
|-------------|-------------------------|
| <i>dest</i> | destination operand |
| <i>val</i> | value to add/sub/and/or |

Returns

res

Definition at line 540 of file [atomic.h](#).

13.16.6.2.41 l4util_sub8()

```
void l4util_sub8 (
    volatile l4_uint8_t * dest,
    l4_uint8_t val) [inline]
```

Parameters

| | |
|-------------|-------------------------|
| <i>dest</i> | destination operand |
| <i>val</i> | value to add/sub/and/or |

Definition at line 484 of file [atomic.h](#).

13.16.6.2.42 l4util_sub8_res()

```
l4_uint8_t l4util_sub8_res (
    volatile l4_uint8_t * dest,
    l4_uint8_t val) [inline]
```

Parameters

| | |
|-------------|-------------------------|
| <i>dest</i> | destination operand |
| <i>val</i> | value to add/sub/and/or |

Returns

res

Definition at line 532 of file [atomic.h](#).

13.16.6.2.43 l4util_xchg()

```
l4_umword_t l4util_xchg (
    volatile l4_umword_t * dest,
    l4_umword_t val) [inline]
```

Atomic exchange (machine wide fields).

Parameters

| | |
|-------------|---------------------|
| <i>dest</i> | destination operand |
| <i>val</i> | new value for dest |

Returns

old value at destination

Definition at line 405 of file [atomic.h](#).

13.16.6.2.44 l4util_xchg16()

```
l4_uint16_t l4util_xchg16 (  
    volatile l4_uint16_t * dest,  
    l4_uint16_t val) [inline]
```

Atomic exchange (16 bit version).

Parameters

| | |
|-------------|---------------------|
| <i>dest</i> | destination operand |
| <i>val</i> | new value for dest |

Returns

old value at destination

Definition at line 393 of file [atomic.h](#).

13.16.6.2.45 l4util_xchg32()

```
l4_uint32_t l4util_xchg32 (  
    volatile l4_uint32_t * dest,  
    l4_uint32_t val) [inline]
```

Atomic exchange (32 bit version).

Parameters

| | |
|-------------|---------------------|
| <i>dest</i> | destination operand |
| <i>val</i> | new value for dest |

Returns

old value at destination

Definition at line 387 of file [atomic.h](#).

13.16.6.2.46 l4util_xchg8()

```
l4_uint8_t l4util_xchg8 (  
    volatile l4_uint8_t * dest,  
    l4_uint8_t val) [inline]
```

Atomic exchange (8 bit version).

Parameters

| | |
|-------------|---------------------|
| <i>dest</i> | destination operand |
| <i>val</i> | new value for dest |

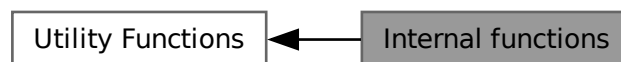
Returns

old value at destination

Definition at line [399](#) of file [atomic.h](#).

13.16.7 Internal functions

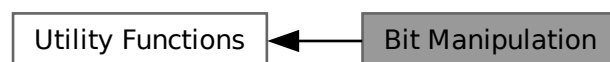
Collaboration diagram for Internal functions:

**Functions**

- void **base64_encode** (const char *infile, unsigned int in_size, char **outfile)
base-64-encode string infile
- void **base64_decode** (const char *infile, unsigned int in_size, char **outfile)
decode base-64-encoded string infile

13.16.7.1 Detailed Description**13.16.8 Bit Manipulation**

Collaboration diagram for Bit Manipulation:



Files

- file [bitops_arch.h](#)
amd64 bit manipulation functions
- file [bitops.h](#)
bit manipulation functions
- file [bitops_arch.h](#)
x86 bit manipulation functions

Functions

- void [l4util_set_bit](#) (int b, volatile [l4_umword_t](#) *dest)
Set bit in memory.
- void [l4util_clear_bit](#) (int b, volatile [l4_umword_t](#) *dest)
Clear bit in memory.
- void [l4util_complement_bit](#) (int b, volatile [l4_umword_t](#) *dest)
Complement bit in memory.
- int [l4util_test_bit](#) (int b, const volatile [l4_umword_t](#) *dest)
Test bit (return value of bit).
- int [l4util_bts](#) (int b, volatile [l4_umword_t](#) *dest)
Bit test and set.
- int [l4util_btr](#) (int b, volatile [l4_umword_t](#) *dest)
Bit test and reset.
- int [l4util_btc](#) (int b, volatile [l4_umword_t](#) *dest)
Bit test and complement.
- int [l4util_bsr](#) ([l4_umword_t](#) word)
Bit scan reverse.
- int [l4util_bsf](#) ([l4_umword_t](#) word)
Bit scan forward.
- int [l4util_find_first_set_bit](#) (const void *dest, [l4_size_t](#) size)
Find the first set bit in a memory region.
- int [l4util_find_first_zero_bit](#) (const void *dest, [l4_size_t](#) size)
Find the first zero bit in a memory region.
- int [l4util_next_power2](#) (unsigned long val)
Find the next power of 2 for a given number.

13.16.8.1 Detailed Description

13.16.8.2 Function Documentation

13.16.8.2.1 l4util_bsf()

```
int l4util_bsf (
    l4\_umword\_t word) [inline]
```

Bit scan forward.

Parameters

| | |
|-------------|----------------------|
| <i>word</i> | value (machine size) |
|-------------|----------------------|

Returns

index of least significant bit set in word, -1 if no bit is set (word == 0)

"bit scan forward", find least significant bit set in word.

Definition at line 316 of file [bitops.h](#).

13.16.8.2.2 l4util_bsr()

```
int l4util_bsr (  
    l4_umword_t word) [inline]
```

Bit scan reverse.

Parameters

| | |
|-------------|----------------------|
| <i>word</i> | value (machine size) |
|-------------|----------------------|

Returns

index of most significant set bit in word, -1 if no bit is set (word == 0)

"bit scan reverse", find most significant set bit in word (-> LOG2(word))

Definition at line 299 of file [bitops.h](#).

13.16.8.2.3 l4util_btc()

```
int l4util_btc (  
    int b,  
    volatile l4_umword_t * dest) [inline]
```

Bit test and complement.

Parameters

| | |
|-------------|---------------------|
| <i>b</i> | bit position |
| <i>dest</i> | destination operand |

Returns

Old value of bit *b*.

Complement bit *b* and return old value.

Definition at line 394 of file [bitops.h](#).

13.16.8.2.4 l4util_btr()

```
int l4util_btr (  
    int b,  
    volatile l4_umword_t * dest) [inline]
```

Bit test and reset.

Parameters

| | |
|-------------|---------------------|
| <i>b</i> | bit position |
| <i>dest</i> | destination operand |

Returns

Old value of bit *b*.

Reset bit *b* and return old value.

Definition at line 278 of file [bitops.h](#).

References [l4util_cmpxchg\(\)](#).

Here is the call graph for this function:



13.16.8.2.5 l4util_bts()

```
int l4util_bts (  
    int b,  
    volatile l4_umword_t * dest) [inline]
```

Bit test and set.

Parameters

| | |
|-------------|---------------------|
| <i>b</i> | bit position |
| <i>dest</i> | destination operand |

Returns

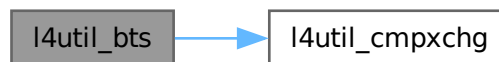
Old value of bit *b*.

Set the *b* bit of *dest* to 1 and return the old value.

Definition at line 256 of file [bitops.h](#).

References [l4util_cmpxchg\(\)](#).

Here is the call graph for this function:

**13.16.8.2.6 l4util_clear_bit()**

```
void l4util_clear_bit (  
    int b,  
    volatile l4_umword_t * dest) [inline]
```

Clear bit in memory.

Parameters

| | |
|-------------|---------------------|
| <i>b</i> | bit position |
| <i>dest</i> | destination operand |

Definition at line 226 of file [bitops.h](#).

References [l4util_cmpxchg\(\)](#).

Here is the call graph for this function:



13.16.8.2.7 l4util_complement_bit()

```
void l4util_complement_bit (
    int b,
    volatile l4_umword_t * dest) [inline]
```

Complement bit in memory.

Parameters

| | |
|-------------|---------------------|
| <i>b</i> | bit position |
| <i>dest</i> | destination operand |

Definition at line 359 of file [bitops.h](#).

13.16.8.2.8 l4util_find_first_set_bit()

```
int l4util_find_first_set_bit (
    const void * dest,
    l4_size_t size) [inline]
```

Find the first set bit in a memory region.

Parameters

| | |
|-------------|---|
| <i>dest</i> | bit string |
| <i>size</i> | size of string in bits (must be a multiple of L4_MWORD_BITS!) |

Returns

number of the first set bit, >= size if no bit is set

Definition at line 400 of file [bitops.h](#).

13.16.8.2.9 l4util_find_first_zero_bit()

```
int l4util_find_first_zero_bit (
    const void * dest,
    l4_size_t size) [inline]
```

Find the first zero bit in a memory region.

Parameters

| | |
|-------------|---|
| <i>dest</i> | bit string |
| <i>size</i> | size of string in bits (must be a multiple of L4_MWORD_BITS!) |

Returns

number of the first zero bit, >= size if no bit is set

Definition at line 333 of file [bitops.h](#).

13.16.8.2.10 l4util_next_power2()

```
int l4util_next_power2 (  
    unsigned long val) [inline]
```

Find the next power of 2 for a given number.

Parameters

| | |
|------------|---------------|
| <i>val</i> | initial value |
|------------|---------------|

Returns

next-highest power of 2

Definition at line 373 of file [bitops.h](#).

13.16.8.2.11 l4util_set_bit()

```
void l4util_set_bit (  
    int b,  
    volatile l4_umword_t * dest) [inline]
```

Set bit in memory.

Parameters

| | |
|-------------|---------------------|
| <i>b</i> | bit position |
| <i>dest</i> | destination operand |

Definition at line 207 of file [bitops.h](#).

References [l4util_cmpxchg\(\)](#).

Here is the call graph for this function:



13.16.8.2.12 l4util_test_bit()

```
int l4util_test_bit (
    int b,
    const volatile l4_umword_t * dest) [inline]
```

Test bit (return value of bit).

Parameters

| | |
|-------------|---------------------|
| <i>b</i> | bit position |
| <i>dest</i> | destination operand |

Returns

Value of bit *b*.

Definition at line 244 of file [bitops.h](#).

13.16.9 ELF binary format

Functions and types related to ELF binaries.

Collaboration diagram for ELF binary format:



Files

- file [elf.h](#)
ELF definition.

Data Structures

- struct [Elf32_Ehdr](#)
ELF32 header.
- struct [Elf64_Ehdr](#)
ELF64 header.
- struct [Elf32_Shdr](#)
ELF32 section header.
- struct [Elf64_Shdr](#)

- *ELF64 section header.*
- struct [Elf32_Phdr](#)
- *ELF32 program header.*
- struct [Elf64_Phdr](#)
- *ELF64 program header.*
- struct [Elf32_Dyn](#)
- *ELF32 dynamic entry.*
- struct [Elf64_Dyn](#)
- *ELF64 dynamic entry.*
- struct [Elf32_Rel](#)
- *ELF32 relocation entry w/o addend.*
- struct [Elf32_Rela](#)
- *ELF32 relocation entry w/ addend.*
- struct [Elf64_Rel](#)
- *ELF64 relocation entry w/o addend.*
- struct [Elf64_Rela](#)
- *ELF64 relocation entry w/ addend.*
- struct [Elf32_Sym](#)
- *ELF32 symbol table entry.*
- struct [Elf64_Sym](#)
- *ELF64 symbol table entry.*
- struct [Elf32_Auxv](#)
- *Auxiliary vector (32-bit).*
- struct [Elf64_Auxv](#)
- *Auxiliary vector (64-bit).*

Macros

- #define **ElfW**(type)
- *Use 64 or 32 bits types depending on the target architecture.*
- #define **ELF32_R_SYM**(i)
- *Symbol table index.*
- #define **ELF32_R_TYPE**(i)
- #define **ELF32_R_INFO**(s, t)
- *Create info from symbol table index + type.*
- #define **ELF64_R_SYM**(i)
- *Symbol table index.*
- #define **ELF64_R_TYPE**(i)
- #define **ELF64_R_INFO**(s, t)
- *Create info from symbol table index + type.*
- #define **ELF32_ST_BIND**(i)
- #define **ELF32_ST_TYPE**(i)
- #define **ELF32_ST_INFO**(b, t)
- *Make info from bind + type.*
- #define **ELF64_ST_BIND**(i)
- #define **ELF64_ST_TYPE**(i)
- #define **ELF64_ST_INFO**(b, t)
- *Make info from bind + type.*

Typedefs

- typedef struct Elf32_Auxv **Elf32_Auxv**
Auxiliary vector (32-bit).
- typedef struct Elf64_Auxv **Elf64_Auxv**
Auxiliary vector (64-bit).

Enumerations

- enum { **EI_NIDENT** = 16 }
- enum **Elf_ETs** {
 ET_NONE = 0 , **ET_REL** = 1 , **ET_EXEC** = 2 , **ET_DYN** = 3 ,
 ET_CORE = 4 , **ET_LOPROC** = 0xff00 , **ET_HIPROC** = 0xffff }
- Object file type.*
- enum **Elf_EMs** {
 EM_NONE = 0 , **EM_M32** = 1 , **EM_SPARC** = 2 , **EM_386** = 3 ,
 EM_68K = 4 , **EM_88K** = 5 , **EM_860** = 7 , **EM_MIPS** = 8 ,
 EM_MIPS_RS4_BE = 10 , **EM_SPARC64** = 11 , **EM_PARISC** = 15 , **EM_VPP500** = 17 ,
 EM_SPARC32PLUS = 18 , **EM_960** = 19 , **EM_PPC** = 20 , **EM_V800** = 36 ,
 EM_FR20 = 37 , **EM_RH32** = 38 , **EM_RCE** = 39 , **EM_ARM** = 40 ,
 EM_ALPHA = 41 , **EM_SH** = 42 , **EM_SPARCV9** = 43 , **EM_TRICORE** = 44 ,
 EM_ARC = 45 , **EM_H8_300** = 46 , **EM_H8_300H** = 47 , **EM_H8S** = 48 ,
 EM_H8_500 = 49 , **EM_IA_64** = 50 , **EM_MIPS_X** = 51 , **EM_COLDFIRE** = 52 ,
 EM_68HC12 = 53 , **EM_X86_64** = 62 , **EM_PDSP** = 63 , **EM_FX66** = 66 ,
 EM_ST9PLUS = 67 , **EM_ST7** = 68 , **EM_68HC16** = 69 , **EM_68HC11** = 70 ,
 EM_68HC08 = 71 , **EM_68HC05** = 72 , **EM_SVX** = 73 , **EM_ST19** = 74 ,
 EM_VAX = 75 , **EM_CRIS** = 76 , **EM_JAVELIN** = 77 , **EM_FIREPATH** = 78 ,
 EM_ZSP = 79 , **EM_MMIX** = 80 , **EM_HUANY** = 81 , **EM_PRISM** = 82 ,
 EM_AVR = 83 , **EM_FR30** = 84 , **EM_D10V** = 85 , **EM_D30V** = 86 ,
 EM_V850 = 87 , **EM_M32R** = 88 , **EM_MN10300** = 89 , **EM_MN10200** = 90 ,
 EM_PJ = 91 , **EM_OPENRISC** = 92 , **EM_ARC_A5** = 93 , **EM_XTENSA** = 94 ,
 EM_ALTERA_NIOS2 = 113 , **EM_AARCH64** = 183 , **EM_TILEPRO** = 188 , **EM_MICROBLAZE** = 189 ,
 EM_TILEGX = 191 , **EM_RISCV** = 243 , **EM_NUM** = 244 }
- Required architecture.*
- enum **Elf_EVs** { **EV_NONE** = 0 , **EV_CURRENT** = 1 }
- Object file version.*
- enum **Elf_EIs** {
 EI_MAG0 = 0 , **EI_MAG1** = 1 , **EI_MAG2** = 2 , **EI_MAG3** = 3 ,
 EI_CLASS = 4 , **EI_DATA** = 5 , **EI_VERSION** = 6 , **EI_OSABI** = 7 ,
 EI_ABIVERSION = 8 , **EI_PAD** = 9 }
- Identification Indices.*
- enum **Elf_MAGs** { **ELFMAG0** = 0x7f , **ELFMAG1** = 'E' , **ELFMAG2** = 'L' , **ELFMAG3** = 'F' }
- Magic number.*
- enum **Elf_CLASSs** { **ELFCLASSNONE** = 0 , **ELFCLASS32** = 1 , **ELFCLASS64** = 2 , **ELFCLASSNUM** = 3 }
- File class or capacity.*
- enum **Elf_DATAs** { **ELFDATANONE** = 0 , **ELFDATA2LSB** = 1 , **ELFDATA2MSB** = 2 , **ELFDATANUM** = 3 }
- Data encoding.*
- enum **Elf_OSABIs** {
 ELFOSABI_NONE = 0 , **ELFOSABI_SYSV** = 0 , **ELFOSABI_HPUX** = 1 , **ELFOSABI_NETBSD** = 2 ,
 ELFOSABI_LINUX = 3 , **ELFOSABI_SOLARIS** = 6 , **ELFOSABI_AIX** = 7 , **ELFOSABI_IRIX** = 8 ,
 ELFOSABI_FREEBSD = 9 , **ELFOSABI_TRU64** = 10 , **ELFOSABI_MODESTO** = 11 , **ELFOSABI_OPENBSD** = 12 ,
 ELFOSABI_ARM = 97 , **ELFOSABI_STANDALONE** = 255 }
- Identify operating system and ABI to which the object is targeted.*

- enum `Elf_SHNs` {
`SHN_UNDEF` = 0 , `SHN_LORESERVE` = 0xff00 , `SHN_LOPROC` = 0xff00 , `SHN_HIPROC` = 0xff1f ,
`SHN_ABS` = 0xffff1 , `SHN_COMMON` = 0xffff2 , `SHN_HIRESERVE` = 0xffff }
Special section indexes.
- enum `Elf_SHTs` {
`SHT_NULL` = 0 , `SHT_PROGBITS` = 1 , `SHT_SYMTAB` = 2 , `SHT_STRTAB` = 3 ,
`SHT_RELA` = 4 , `SHT_HASH` = 5 , `SHT_DYNAMIC` = 6 , `SHT_NOTE` = 7 ,
`SHT_NOBITS` = 8 , `SHT_REL` = 9 , `SHT_SHLIB` = 10 , `SHT_DYNSYM` = 11 ,
`SHT_INIT_ARRAY` = 14 , `SHT_FINI_ARRAY` = 15 , `SHT_PREINIT_ARRAY` = 16 , `SHT_GROUP` = 17 ,
`SHT_SYMTAB_SHNDX` = 18 , `SHT_NUM` = 19 , `SHT_LOOS` = 0x60000000 , `SHT_HIOS` = 0x6fffffff ,
`SHT_LOPROC` = 0x70000000 , `SHT_HIPROC` = 0x7fffffff , `SHT_LOUSER` = 0x80000000 , `SHT_HIUSER` =
0xffffffff }
Section type.
- enum `Elf_SHFs` {
`SHF_WRITE` = 0x1 , `SHF_ALLOC` = 0x2 , `SHF_EXECINSTR` = 0x4 , `SHF_MERGE` = 0x10 ,
`SHF_STRINGS` = 0x20 , `SHF_INFO_LINK` = 0x40 , `SHF_LINK_ORDER` = 0x80 , `SHF_OS_NONCONFORMING`
= 0x100 ,
`SHF_GROUP` = 0x200 , `SHF_TLS` = 0x400 , `SHF_MASKOS` = 0x0ff00000 , `SHF_MASKPROC` = 0xf0000000
}
Section attribute flags.
- enum `Elf_PTs` {
`PT_NULL` = 0 , `PT_LOAD` = 1 , `PT_DYNAMIC` = 2 , `PT_INTERP` = 3 ,
`PT_NOTE` = 4 , `PT_SHLIB` = 5 , `PT_PHDR` = 6 , `PT_TLS` = 7 ,
`PT_NUM` = 8 , `PT_LOOS` = 0x60000000 , `PT_HIOS` = 0x6fffffff , `PT_LOPROC` = 0x70000000 ,
`PT_HIPROC` = 0x7fffffff , `PT_GNU_EH_FRAME` = `PT_LOOS` + 0x474e550 , `PT_GNU_STACK` = `PT_LOOS`
+ 0x474e551 , `PT_GNU_RELRO` = `PT_LOOS` + 0x474e552 ,
`PT_L4_STACK` = `PT_LOOS` + 0x12 , `PT_L4_AUX` = `PT_LOOS` + 0x14 }
Segment types.
- enum `Elf_PFs` {
`PF_X` = 0x1 , `PF_W` = 0x2 , `PF_R` = 0x4 , `PF_MASKOS` = 0x0ff00000 ,
`PF_MASKPROC` = 0x7fffffff }
Segment permissions.
- enum `Elf_NTscore` {
`NT_PRSTATUS` = 1 , `NT_FPREGSET` = 2 , `NT_PRPSINFO` = 3 , `NT_PRXREG` = 4 ,
`NT_TASKSTRUCT` = 4 , `NT_PLATFORM` = 5 , `NT_AUXV` = 6 , `NT_GWINDOWS` = 7 ,
`NT_ASRS` = 8 , `NT_PSTATUS` = 10 , `NT_PSINFO` = 13 , `NT_PRCRED` = 14 ,
`NT_UTSNAME` = 15 , `NT_LWPSTATUS` = 16 , `NT_LWPSINFO` = 17 , `NT_PRFPXREG` = 20 }
Legal values for note segment descriptor types for core files.
- enum `Elf_NTsobj` { `NT_VERSION` = 1 }
Legal values for the note segment descriptor types for object files.
- enum `Elf_DTs` {
`DT_NULL` = 0 , `DT_NEEDED` = 1 , `DT_PLTRELSZ` = 2 , `DT_PLTGOT` = 3 ,
`DT_HASH` = 4 , `DT_STRTAB` = 5 , `DT_SYMTAB` = 6 , `DT_RELA` = 7 ,
`DT_RELASZ` = 8 , `DT_RELAENT` = 9 , `DT_STRSZ` = 10 , `DT_SYMENT` = 11 ,
`DT_INIT` = 12 , `DT_FINI` = 13 , `DT_SONAME` = 14 , `DT_RPATH` = 15 ,
`DT_SYMBOLIC` = 16 , `DT_REL` = 17 , `DT_RELSZ` = 18 , `DT_RELENT` = 19 ,
`DT_PTRREL` = 20 , `DT_DEBUG` = 21 , `DT_TEXTREL` = 22 , `DT_JMPREL` = 23 ,
`DT_BIND_NOW` = 24 , `DT_INIT_ARRAY` = 25 , `DT_FINI_ARRAY` = 26 , `DT_INIT_ARRAYSZ` = 27 ,
`DT_FINI_ARRAYSZ` = 28 , `DT_RUNPATH` = 29 , `DT_FLAGS` = 30 , `DT_ENCODING` = 32 ,
`DT_PREINIT_ARRAY` = 32 , `DT_PREINIT_ARRAYSZ` = 33 , `DT_NUM` = 34 , `DT_LOOS` = 0x6000000d ,
`DT_HIOS` = 0x6ffff000 , `DT_LOPROC` = 0x70000000 , `DT_HIPROC` = 0x7fffffff }
Dynamic Array Tags.
- enum `Elf_DFs` {
`DF_ORIGIN` = 0x00000001 , `DF_SYMBOLIC` = 0x00000002 , `DF_TEXTREL` = 0x00000004 ,
`DF_BIND_NOW` = 0x00000008 ,
`DF_STATIC_TLS` = 0x00000010 }

Values of Elf32_Dyn.d_un.d_val, Elf64_Dyn.d_un.d_val in the DT_FLAGS entry.

- enum [Elf_DF_1s](#) {
[DF_1_NOW](#) = 0x00000001 , [DF_1_GLOBAL](#) = 0x00000002 , [DF_1_GROUP](#) = 0x00000004 ,
[DF_1_NODELETE](#) = 0x00000008 ,
[DF_1_LOADFLTR](#) = 0x00000010 , [DF_1_INITFIRST](#) = 0x00000020 , [DF_1_NOOPEN](#) = 0x00000040 ,
[DF_1_ORIGIN](#) = 0x00000080 ,
[DF_1_DIRECT](#) = 0x00000100 , [DF_1_TRANS](#) = 0x00000200 , [DF_1_INTERPOSE](#) = 0x00000400 ,
[DF_1_NODEFLIB](#) = 0x00000800 ,
[DF_1_NODUMP](#) = 0x00001000 , [DF_1_CONFALT](#) = 0x00002000 , [DF_1_ENDFILTEE](#) = 0x00004000 ,
[DF_1_DISPRELDNE](#) = 0x00008000 ,
[DF_1_DISPRELPND](#) = 0x00010000 }

State flags selectable in the Elf32_Dyn.d_un.d_val / Elf64_Dyn.d_un.d_val element of the DT_FLAGS_1 entry in the dynamic section.

- enum [Elf_DTF_1s](#)

Flags for the feature selection in DT_FEATURE_1.

- enum [Elf_DF_P1s](#) { [DF_P1_LAZYLOAD](#) = 0x00000001 , [DF_P1_GROUPPERM](#) = 0x00000002 }

Flags in the DT_POSFLAG_1 entry effecting only the next DT_ entry.*

- enum [Elf_R_386_s](#) {
[R_386_NONE](#) = 0 , [R_386_32](#) = 1 , [R_386_PC32](#) = 2 , [R_386_GOT32](#) = 3 ,
[R_386_PLT32](#) = 4 , [R_386_COPY](#) = 5 , [R_386_GLOB_DAT](#) = 6 , [R_386_JMP_SLOT](#) = 7 ,
[R_386_RELATIVE](#) = 8 , [R_386_GOTOFF](#) = 9 , [R_386_GOTPC](#) = 10 , [R_386_32PLT](#) = 11 ,
[R_386_TLS_TPOFF](#) = 14 , [R_386_TLS_IE](#) = 15 , [R_386_TLS_GOTIE](#) = 16 , [R_386_TLS_LE](#) = 17 ,
[R_386_TLS_GD](#) = 18 , [R_386_TLS_LDM](#) = 19 , [R_386_16](#) = 20 , [R_386_PC16](#) = 21 ,
[R_386_8](#) = 22 , [R_386_PC8](#) = 23 , [R_386_TLS_GD_32](#) = 24 , [R_386_TLS_GD_PUSH](#) = 25 ,
[R_386_TLS_GD_CALL](#) = 26 , [R_386_TLS_GD_POP](#) = 27 , [R_386_TLS_LDM_32](#) = 28 , [R_386_TLS_LDM_PUSH](#)
= 29 ,
[R_386_TLS_LDM_CALL](#) = 30 , [R_386_TLS_LDM_POP](#) = 31 , [R_386_TLS_LDO_32](#) = 32 , [R_386_TLS_IE_32](#)
= 33 ,
[R_386_TLS_LE_32](#) = 34 , [R_386_TLS_DTPMOD32](#) = 35 , [R_386_TLS_DTPOFF32](#) = 36 , [R_386_TLS_TPOFF32](#)
= 37 ,
[R_386_NUM](#) = 38 }

Relocation types (processor specific).

- enum [Elf_EF_ARM_s](#) { }

ARM specific declarations.

- enum [Elf_STT_ARM_s](#)

Additional symbol types for Thumb.

- enum [Elf_SHF_s_ARM](#) { [SHF_ARM_ENTRYSECT](#) = 0x10000000 , [SHF_ARM_COMDEF](#) = 0x80000000 }

ARM-specific values for Elf32_Shdr.sh_flags / Elf64_Shdr.sh_flags.

- enum [Elf_ARM_SBs](#) { [PF_ARM_SB](#) = 0x10000000 }

ARM-specific program header flags.

- enum [Elf_R_ARM_s](#) {
[R_ARM_NONE](#) = 0 , [R_ARM_PC24](#) = 1 , [R_ARM_ABS32](#) = 2 , [R_ARM_REL32](#) = 3 ,
[R_ARM_PC13](#) = 4 , [R_ARM_ABS16](#) = 5 , [R_ARM_ABS12](#) = 6 , [R_ARM_THM_ABS5](#) = 7 ,
[R_ARM_ABS8](#) = 8 , [R_ARM_SBREL32](#) = 9 , [R_ARM_THM_PC22](#) = 10 , [R_ARM_THM_PC8](#) = 11 ,
[R_ARM_AMP_VCALL9](#) = 12 , [R_ARM_SWI24](#) = 13 , [R_ARM_THM_SWI8](#) = 14 , [R_ARM_XPC25](#) = 15 ,
[R_ARM_THM_XPC22](#) = 16 , [R_ARM_COPY](#) = 20 , [R_ARM_GLOB_DAT](#) = 21 , [R_ARM_JUMP_SLOT](#) = 22 ,
[R_ARM_RELATIVE](#) = 23 , [R_ARM_GOTOFF](#) = 24 , [R_ARM_GOTPC](#) = 25 , [R_ARM_GOT32](#) = 26 ,
[R_ARM_PLT32](#) = 27 , [R_ARM_ALU_PCREL_7_0](#) = 32 , [R_ARM_ALU_PCREL_15_8](#) = 33 , [R_ARM_↵](#)
[ALU_PCREL_23_15](#) = 34 ,
[R_ARM_LDR_SBREL_11_0](#) = 35 , [R_ARM_ALU_SBREL_19_12](#) = 36 , [R_ARM_ALU_SBREL_27_20](#) =
37 , [R_ARM_GNU_VTENTRY](#) = 100 ,
[R_ARM_GNU_VTINHERIT](#) = 101 , [R_ARM_THM_PC11](#) = 102 , [R_ARM_THM_PC9](#) = 103 , [R_ARM_↵](#)
[RXPC25](#) = 249 ,
[R_ARM_RSBREL32](#) = 250 , [R_ARM_THM_RPC22](#) = 251 , [R_ARM_RREL32](#) = 252 , [R_ARM_RABS22](#) =
253 ,
[R_ARM_RPC24](#) = 254 , [R_ARM_RBASE](#) = 255 , [R_ARM_NUM](#) = 256 }

ARM relocations.

- enum `Elf_R_AARCH64_s` { `R_AARCH64_NONE` = 0 , `R_AARCH64_RELATIVE` = 1027 }

AARCH64 relocations.

- enum `Elf_R_X86_64_s` {
`R_X86_64_NONE` = 0 , `R_X86_64_64` = 1 , `R_X86_64_PC32` = 2 , `R_X86_64_GOT32` = 3 ,
`R_X86_64_PLT32` = 4 , `R_X86_64_COPY` = 5 , `R_X86_64_GLOB_DAT` = 6 , `R_X86_64_JUMP_SLOT` = 7 ,
`R_X86_64_RELATIVE` = 8 , `R_X86_64_GOTPCREL` = 9 , `R_X86_64_32` = 10 , `R_X86_64_32S` = 11 ,
`R_X86_64_16` = 12 , `R_X86_64_PC16` = 13 , `R_X86_64_8` = 14 , `R_X86_64_PC8` = 15 ,
`R_X86_64_DTPMOD64` = 16 , `R_X86_64_DTPOFF64` = 17 , `R_X86_64_TPOFF64` = 18 , `R_X86_64_TLSGD`
= 19 ,
`R_X86_64_TLSLD` = 20 , `R_X86_64_DTPOFF32` = 21 , `R_X86_64_GOTTPOFF` = 22 , `R_X86_64_TPOFF32`
= 23 ,
`R_X86_64_NUM` = 24 }

AMD x86-64 relocations.

- enum `Elf_STNs`
Symbol Table Entry.
- enum `Elf_STBs` {
`STB_LOCAL` = 0 , `STB_GLOBAL` = 1 , `STB_WEAK` = 2 , `STB_LOOS` = 10 ,
`STB_HIOS` = 12 , `STB_LOPROC` = 13 , `STB_HIPROC` = 15 }

Symbol Binding.

- enum `Elf_STTs` {
`STT_NOTYPE` = 0 , `STT_OBJECT` = 1 , `STT_FUNC` = 2 , `STT_SECTION` = 3 ,
`STT_FILE` = 4 , `STT_LOOS` = 10 , `STT_HIOS` = 12 , `STT_LOPROC` = 13 ,
`STT_HIPROC` = 15 }

Symbol Types.

- enum `Elf_ATs` {
`AT_NULL` = 0 , `AT_IGNORE` = 1 , `AT_EXECFD` = 2 , `AT_PHDR` = 3 ,
`AT_PHENT` = 4 , `AT_PHNUM` = 5 , `AT_PAGESZ` = 6 , `AT_BASE` = 7 ,
`AT_FLAGS` = 8 , `AT_ENTRY` = 9 , `AT_NOTELF` = 10 , `AT_UID` = 11 ,
`AT_EUID` = 12 , `AT_GID` = 13 , `AT_EGID` = 14 , `AT_L4_AUX` = 0xf0 ,
`AT_L4_ENV` = 0xf1 , `AT_L4_KIP` = 0xf2 }

Legal values for `Elf32_Auxv.atype` / `Elf64_Auxv.atype`.

ELF types

- typedef `l4_uint32_t` `Elf32_Addr`
size 4 align 4
- typedef `l4_uint32_t` `Elf32_Off`
size 4 align 4
- typedef `l4_uint16_t` `Elf32_Half`
size 2 align 2
- typedef `l4_uint32_t` `Elf32_Word`
size 4 align 4
- typedef `l4_int32_t` `Elf32_Sword`
size 4 align 4
- typedef `l4_uint64_t` `Elf64_Addr`
size 8 align 8
- typedef `l4_uint64_t` `Elf64_Off`
size 8 align 8
- typedef `l4_uint16_t` `Elf64_Half`
size 2 align 2
- typedef `l4_uint32_t` `Elf64_Word`
size 4 align 4

- typedef [l4_int32_t](#) Elf64_Sword
size 4 align 4
- typedef [l4_uint64_t](#) Elf64_Xword
size 8 align 8
- typedef [l4_int64_t](#) Elf64_Sxword
size 8 align 8

13.16.9.1 Detailed Description

Functions and types related to ELF binaries.

13.16.9.2 Macro Definition Documentation

13.16.9.2.1 ELF32_R_TYPE

```
#define ELF32_R_TYPE(  
    i)
```

Value:

```
((unsigned char)(i))
```

See also

[Elf_R_386s](#).

Definition at line [663](#) of file [elf.h](#).

13.16.9.2.2 ELF32_ST_BIND

```
#define ELF32_ST_BIND(  
    i)
```

Value:

```
((i) >> 4)
```

See also

[Elf_STBs](#).

Definition at line [893](#) of file [elf.h](#).

13.16.9.2.3 ELF32_ST_TYPE

```
#define ELF32_ST_TYPE(  
    i)
```

Value:

```
((i) & 0xf)
```

See also

[Elf_STTs](#).

Definition at line [896](#) of file [elf.h](#).

13.16.9.2.4 ELF64_R_TYPE

```
#define ELF64_R_TYPE(  
    i)
```

Value:

```
((i) & 0xffffffffL)
```

See also

[Elf_R_386s](#).

Definition at line [671](#) of file [elf.h](#).

13.16.9.2.5 ELF64_ST_BIND

```
#define ELF64_ST_BIND(  
    i)
```

Value:

```
((i) >> 4)
```

See also

[Elf_STBs](#)

Definition at line [902](#) of file [elf.h](#).

13.16.9.2.6 ELF64_ST_TYPE

```
#define ELF64_ST_TYPE(  
    i)
```

Value:

```
((i) & 0xf)
```

See also

[Elf_STTs](#)

Definition at line [905](#) of file [elf.h](#).

13.16.9.3 Enumeration Type Documentation

13.16.9.3.1 anonymous enum

anonymous enum

Enumerator

| | |
|-----------|-----------------------|
| EI_NIDENT | Number of characters. |
|-----------|-----------------------|

Definition at line 117 of file [elf.h](#).

13.16.9.3.2 Elf_ARM_SBs

enum [Elf_ARM_SBs](#)

ARM-specific program header flags.

Enumerator

| | |
|-----------|---|
| PF_ARM_SB | Segment contains the location addressed by the static base. |
|-----------|---|

Definition at line 770 of file [elf.h](#).

13.16.9.3.3 Elf_ATs

enum [Elf_ATs](#)

Legal values for [Elf32_Auxv.atype](#) / [Elf64_Auxv.atype](#).

Enumerator

| | |
|-----------|-------------------------------|
| AT_NULL | End of vector. |
| AT_IGNORE | Entry should be ignored. |
| AT_EXECD | File descriptor of program. |
| AT_PHDR | Program headers for program. |
| AT_PHEM | Size of program header entry. |
| AT_PHNUM | Number of program headers. |
| AT_PAGESZ | System page size. |
| AT_BASE | Base address of interpreter. |
| AT_FLAGS | Flags. |
| AT_ENTRY | Entry point of program. |
| AT_NOTELF | Program is not ELF. |
| AT_UID | Real UID. |

| | |
|-----------|-----------------------------------|
| AT_EUID | Effective UID. |
| AT_GID | Real GID. |
| AT_EGID | Effective GID. |
| AT_L4_AUX | L4Re AUX section. |
| AT_L4_ENV | L4Re ENV section. |
| AT_L4_KIP | L4Re KIP section. |

Definition at line [939](#) of file [elf.h](#).

13.16.9.3.4 Elf_CLASSES

enum [Elf_CLASSES](#)

File class or capacity.

Enumerator

| | |
|--------------|----------------------------------|
| ELFCLASSNONE | Invalid class. |
| ELFCLASS32 | 32-bit object |
| ELFCLASS64 | 64-bit object |
| ELFCLASSNUM | Mask for 32-bit or 64-bit class. |

Definition at line [298](#) of file [elf.h](#).

13.16.9.3.5 Elf_DATAs

enum [Elf_DATAs](#)

Data encoding.

Enumerator

| | |
|-------------|---------------------------------------|
| ELFDATANONE | invalid data encoding |
| ELFDATA2LSB | 0x01020304 => [0x04 0x03 0x02 0x01] |
| ELFDATA2MSB | 0x01020304 => [0x01 0x02 0x03 0x04] |
| ELFDATANUM | Mask for valid data encoding. |

Definition at line [307](#) of file [elf.h](#).

13.16.9.3.6 Elf_DF_1s

enum [Elf_DF_1s](#)

State flags selectable in the Elf32_Dyn.d_un.d_val / Elf64_Dyn.d_un.d_val element of the DT_FLAGS_1 entry in the dynamic section.

Enumerator

| | |
|-----------------|-------------------------------------|
| DF_1_NOW | Set RTLD_NOW for this object. |
| DF_1_GLOBAL | Set RTLD_GLOBAL for this object. |
| DF_1_GROUP | Set RTLD_GROUP for this object. |
| DF_1_NODELETE | Set RTLD_NODELETE for this object. |
| DF_1_LOADFLTR | Trigger filtee loading at runtime. |
| DF_1_INITFIRST | Set RTLD_INITFIRST for this object. |
| DF_1_NOOPEN | Set RTLD_NOOPEN for this object. |
| DF_1_ORIGIN | \$ORIGIN must be handled. |
| DF_1_DIRECT | Direct binding enabled. |
| DF_1_INTERPOSE | Object is used to interpose. |
| DF_1_NODEFLIB | Ignore default lib search path. |
| DF_1_NODUMP | Object can't be dldump'ed. |
| DF_1_CONFALT | Configuration alternative created. |
| DF_1_ENDFILTEE | Filtee terminates filters search. |
| DF_1_DISPRELDNE | Disp reloc applied at build time. |
| DF_1_DISPRELPND | Disp reloc applied at run-time. |

Definition at line [594](#) of file [elf.h](#).

13.16.9.3.7 Elf_DF_P1s

enum [Elf_DF_P1s](#)

Flags in the DT_POSFLAG_1 entry effecting only the next DT_* entry.

Enumerator

| | |
|-----------------|---|
| DF_P1_LAZYLOAD | Lazyload following object. |
| DF_P1_GROUPPERM | Symbols from next object are not generally available. |

Definition at line [623](#) of file [elf.h](#).

13.16.9.3.8 Elf_DFs

enum [Elf_DFs](#)

Values of Elf32_Dyn.d_un.d_val, Elf64_Dyn.d_un.d_val in the DT_FLAGS entry.

Enumerator

| | |
|---------------|-----------------------------------|
| DF_ORIGIN | Object may use DF_ORIGIN. |
| DF_SYMBOLIC | Symbol resolutions starts here. |
| DF_TEXTREL | Object contains text relocations. |
| DF_BIND_NOW | No lazy binding for this object. |
| DF_STATIC_TLS | Module uses the static TLS model. |

Definition at line [581](#) of file [elf.h](#).

13.16.9.3.9 Elf_DTs

enum [Elf_DTs](#)

Dynamic Array Tags.

See also

[Elf32_Dyn.d_tag](#), [Elf64_Dyn.d_tag](#).

Enumerator

| | |
|-------------|------------------------------------|
| DT_NULL | end of _DYNAMIC array |
| DT_NEEDED | name of a needed library |
| DT_PLTRELSZ | total size of relocation entry |
| DT_PLTGOT | address assoc with prog link table |
| DT_HASH | address of symbol hash table |
| DT_STRTAB | address of string table |
| DT_SYMTAB | address of symbol table |
| DT_RELA | address of relocation table |
| DT_RELASZ | total size of relocation table |
| DT_RELAENT | size of DT_RELA relocation entry |
| DT_STRSZ | size of the string table |
| DT_SYMENT | size of a symbol table entry |
| DT_INIT | address of initialization function |
| DT_FINI | address of termination function |
| DT_SONAME | name of the shared object |
| DT_RPATH | search library path |
| DT_SYMBOLIC | alter symbol resolution algorithm |

| | |
|--------------------|---------------------------------------|
| DT_REL | address of relocation table |
| DT_RELSZ | total size of DT_REL relocation table |
| DT_RELENT | size of the DT_REL relocation entry |
| DT_PTRREL | type of relocation entry |
| DT_DEBUG | for debugging purposes |
| DT_TEXTREL | at least on entry changes r/o section |
| DT_JMPREL | address of relocation entries |
| DT_BIND_NOW | Process relocations of object. |
| DT_INIT_ARRAY | Array with addresses of init fct. |
| DT_FINI_ARRAY | Array with addresses of fini fct. |
| DT_INIT_ARRAYSZ | Size in bytes of DT_INIT_ARRAY. |
| DT_FINI_ARRAYSZ | Size in bytes of DT_FINI_ARRAY. |
| DT_RUNPATH | Library search path. |
| DT_FLAGS | Flags for the object being loaded. |
| DT_ENCODING | Start of encoded range. |
| DT_PREINIT_ARRAY | Array with addresses of preinit fct. |
| DT_PREINIT_ARRAYSZ | size in bytes of DT_PREINIT_ARRAY |
| DT_NUM | Number used. |
| DT_LOOS | Start of OS-specific. |
| DT_HIOS | End of OS-specific. |
| DT_LOPROC | processor-specific |
| DT_HIPROC | processor-specific |

Definition at line 535 of file [elf.h](#).

13.16.9.3.10 Elf_EF_ARM_s

enum [Elf_EF_ARM_s](#)

ARM specific declarations.

Processor specific flags for the ELF header e_flags field.

Enumerator

| | |
|---------------|-------------------------------------|
| EF_ARM_ALIGN8 | 8-bit structure alignment is in use |
|---------------|-------------------------------------|

Definition at line 730 of file [elf.h](#).

13.16.9.3.11 Elf_EIs

enum [Elf_EIs](#)

Identification Indices.

See also

[Elf32_Ehdr.e_ident](#), [Elf64_Ehdr.e_ident](#)

Enumerator

| | |
|---------------|--|
| EI_MAG0 | file id 0 |
| EI_MAG1 | file id 1 |
| EI_MAG2 | file id 2 |
| EI_MAG3 | file id 3 |
| EI_CLASS | file class |
| EI_DATA | data encoding |
| EI_VERSION | file version |
| EI_OSABI | Operating system / ABI identification. |
| EI_ABIVERSION | ABI version. |
| EI_PAD | start of padding bytes |

Definition at line 274 of file [elf.h](#).

13.16.9.3.12 Elf_EMs

enum [Elf_EMs](#)

Required architecture.

See also

[Elf32_Ehdr.e_machine](#), [Elf64_Ehdr.e_machine](#)

Enumerator

| | |
|----------------|-------------------------|
| EM_NONE | no machine |
| EM_M32 | AT&T WE 32100. |
| EM_SPARC | SPARC. |
| EM_386 | Intel 80386. |
| EM_68K | Motorola 68000. |
| EM_88K | Motorola 88000. |
| EM_860 | Intel 80860. |
| EM_MIPS | MIPS RS3000 big-endian. |
| EM_MIPS_RS4_BE | MIPS RS4000 big-endian. |
| EM_SPARC64 | SPARC 64-bit. |
| EM_PARISC | HP PA-RISC. |
| EM_VPP500 | Fujitsu VPP500. |
| EM_SPARC32PLUS | Sun's V8plus. |
| EM_960 | Intel 80960. |

| | |
|-------------|--|
| EM_PPC | PowerPC. |
| EM_V800 | NEC V800. |
| EM_FR20 | Fujitsu FR20. |
| EM_RH32 | TRW RH-32. |
| EM_RCE | Motorola RCE. |
| EM_ARM | Advanced RISC Machines ARM. |
| EM_ALPHA | Digital Alpha. |
| EM_SH | Hitachi SuperH. |
| EM_SPARCV9 | SPARC v9 64-bit. |
| EM_TRICORE | Siemens Tricore embedded processor. |
| EM_ARC | Argonaut RISC Core, Argonaut Techn Inc. |
| EM_H8_300 | Hitachi H8/300. |
| EM_H8_300H | Hitachi H8/300H. |
| EM_H8S | Hitachi H8/S. |
| EM_H8_500 | Hitachi H8/500. |
| EM_IA_64 | HP/Intel IA-64. |
| EM_MIPS_X | Stanford MIPS-X. |
| EM_COLDFIRE | Motorola Coldfire. |
| EM_68HC12 | Motorola M68HC12. |
| EM_X86_64 | Advanced Micro Devices x86-64. |
| EM_PDSP | Sony DSP Processor. |
| EM_FX66 | Siemens FX66 microcontroller. |
| EM_ST9PLUS | STMicroelectronics ST9+ 8/16 mc. |
| EM_ST7 | STmicroelectronics ST7 8 bit mc. |
| EM_68HC16 | Motorola MC68HC16 microcontroller. |
| EM_68HC11 | Motorola MC68HC11 microcontroller. |
| EM_68HC08 | Motorola MC68HC08 microcontroller. |
| EM_68HC05 | Motorola MC68HC05 microcontroller. |
| EM_SVX | Silicon Graphics SVx. |
| EM_ST19 | STMicroelectronics ST19 8 bit mc. |
| EM_VAX | Digital VAX. |
| EM_CRIS | Axis Communications 32-bit embedded processor. |
| EM_JAVELIN | Infineon Technologies 32-bit embedded processor. |
| EM_FIREPATH | Element 14 64-bit DSP Processor. |
| EM_ZSP | LSI Logic 16-bit DSP Processor. |
| EM_MMIX | Donald Knuth's educational 64-bit processor. |
| EM_HUANY | Harvard University machine-independent object files. |
| EM_PRISM | SiTera Prism. |
| EM_AVR | Atmel AVR 8-bit microcontroller. |
| EM_FR30 | Fujitsu FR30. |
| EM_D10V | Mitsubishi D10V. |
| EM_D30V | Mitsubishi D30V. |

| | |
|-----------------|-------------------------------------|
| EM_V850 | NEC v850. |
| EM_M32R | Mitsubishi M32R. |
| EM_MN10300 | Matsushita MN10300. |
| EM_MN10200 | Matsushita MN10200. |
| EM_PJ | picoJava |
| EM_OPENRISC | OpenRISC 32-bit embedded processor. |
| EM_ARC_A5 | ARC Cores Tangent-A5. |
| EM_XTENSA | Tensilica Xtensa Architecture. |
| EM_ALTERA_NIOS2 | Altera Nios II. |
| EM_AARCH64 | ARM AARCH64. |
| EM_TILEPRO | Tilera TILEPro. |
| EM_MICROBLAZE | Xilinx MicroBlaze. |
| EM_TILEGX | Tilera TILE-Gx. |
| EM_RISCV | RISC-V. |

Definition at line 183 of file [elf.h](#).

13.16.9.3.13 Elf_ETs

enum [Elf_ETs](#)

Object file type.

See also

[Elf32_Ehdr.e_type](#), [Elf64_Ehdr.e_type](#)

Enumerator

| | |
|-----------|--------------------|
| ET_NONE | no file type |
| ET_REL | relocatable file |
| ET_EXEC | executable file |
| ET_DYN | shared object file |
| ET_CORE | core file |
| ET_LOPROC | processor-specific |
| ET_HIPROC | processor-specific |

Definition at line 168 of file [elf.h](#).

13.16.9.3.14 Elf_EVs

enum [Elf_EVs](#)

Object file version.

See also

[Elf32_Ehdr.e_version](#), [Elf64_Ehdr.e_version](#)

Enumerator

| | |
|------------|------------------|
| EV_NONE | Invalid version. |
| EV_CURRENT | Current version. |

Definition at line 266 of file [elf.h](#).

13.16.9.3.15 Elf_MAGs

enum [Elf_MAGs](#)

Magic number.

Enumerator

| | |
|---------|------------------|
| ELFMAG0 | e_ident[EI_MAG0] |
| ELFMAG1 | e_ident[EI_MAG1] |
| ELFMAG2 | e_ident[EI_MAG2] |
| ELFMAG3 | e_ident[EI_MAG3] |

Definition at line 289 of file [elf.h](#).

13.16.9.3.16 Elf_NT_s_core

enum [Elf_NT_s_core](#)

Legal values for note segment descriptor types for core files.

Enumerator

| | |
|---------------|------------------------------------|
| NT_PRSTATUS | Contains copy of prstatus struct. |
| NT_FPREGSET | Contains copy of fpregset struct. |
| NT_PRPSINFO | Contains copy of prpsinfo struct. |
| NT_PRXREG | Contains copy of prxregset struct. |
| NT_TASKSTRUCT | Contains copy of task structure. |
| NT_PLATFORM | String from sysinfo(SI_PLATFORM). |
| NT_AUXV | Contains copy of auxv array. |
| NT_GWINDOWS | Contains copy of gwindows struct. |
| NT_ASRS | Contains copy of asrset struct. |
| NT_PSTATUS | Contains copy of pstatus struct. |
| NT_PSINFO | Contains copy of psinfo struct. |

| | |
|--------------|-------------------------------------|
| NT_PRCRED | Contains copy of prcred struct. |
| NT_UTSNAME | Contains copy of utsname struct. |
| NT_LWPSTATUS | Contains copy of lwpstatus struct. |
| NT_LWPSINFO | Contains copy of lwpinfo struct. |
| NT_PRFPXREG | Contains copy of fprxregset struct. |

Definition at line 486 of file [elf.h](#).

13.16.9.3.17 Elf_NTs_obj

enum [Elf_NTs_obj](#)

Legal values for the note segment descriptor types for object files.

Enumerator

| | |
|------------|----------------------------|
| NT_VERSION | Contains a version string. |
|------------|----------------------------|

Definition at line 507 of file [elf.h](#).

13.16.9.3.18 Elf_OSABIs

enum [Elf_OSABIs](#)

Identify operating system and ABI to which the object is targeted.

Enumerator

| | |
|---------------------|------------------------------------|
| ELFOSABI_NONE | UNIX System V ABI. |
| ELFOSABI_SYSV | Alias. |
| ELFOSABI_HPUX | HP-UX. |
| ELFOSABI_NETBSD | NetBSD. |
| ELFOSABI_LINUX | Linux. |
| ELFOSABI_SOLARIS | Sun Solaris. |
| ELFOSABI_AIX | IBM AIX. |
| ELFOSABI_IRIX | SGI Irix. |
| ELFOSABI_FREEBSD | FreeBSD. |
| ELFOSABI_TRU64 | Compaq TRU64 UNIX. |
| ELFOSABI_MODESTO | Novell Modesto. |
| ELFOSABI_OPENBSD | OpenBSD. |
| ELFOSABI_ARM | ARM. |
| ELFOSABI_STANDALONE | Standalone (embedded) application. |

Definition at line 316 of file [elf.h](#).

13.16.9.3.19 ELF_PFs

enum [ELF_PFs](#)

Segment permissions.

Enumerator

| | |
|-------------|---------------------|
| PF_X | Executable. |
| PF_W | Write. |
| PF_R | Read. |
| PF_MASKOS | OS-specific. |
| PF_MASKPROC | Processor-specific. |

Definition at line [476](#) of file [elf.h](#).

13.16.9.3.20 Elf_PTs

enum [Elf_PTs](#)

Segment types.

Enumerator

| | |
|-----------------|--------------------------------|
| PT_NULL | array is unused |
| PT_LOAD | loadable |
| PT_DYNAMIC | dynamic linking information |
| PT_INTERP | path to interpreter |
| PT_NOTE | auxiliary information |
| PT_SHLIB | reserved |
| PT_PHDR | location of the pht itself |
| PT_TLS | Thread-local storage segment. |
| PT_NUM | Number of defined types. |
| PT_LOOS | OS-specific. |
| PT_HIOS | OS-specific. |
| PT_LOPROC | processor-specific |
| PT_HIPROC | processor-specific |
| PT_GNU_EH_FRAME | EH frame information. |
| PT_GNU_STACK | Flags for stack. |
| PT_GNU_RELRO | Read only after reloc. |
| PT_L4_STACK | Address of the stack. |
| PT_L4_AUX | Address of the AUX structures. |

Definition at line [451](#) of file [elf.h](#).

13.16.9.3.21 Elf_R_386_s

enum [Elf_R_386_s](#)

Relocation types (processor specific).

Enumerator

| | |
|--------------------|--|
| R_386_NONE | none |
| R_386_32 | S + A. |
| R_386_PC32 | S + A - P. |
| R_386_GOT32 | G + A - P. |
| R_386_PLT32 | L + A - P. |
| R_386_COPY | none |
| R_386_GLOB_DAT | S. |
| R_386_JMP_SLOT | S. |
| R_386_RELATIVE | B + A. |
| R_386_GOTOFF | S + A - GOT. |
| R_386_GOTPC | GOT + A - P. |
| R_386_TLS_TPOFF | Offset in static TLS block. |
| R_386_TLS_IE | Address of GOT entry for static TLS block offset. |
| R_386_TLS_GOTIE | GOT entry for static TLS block offset. |
| R_386_TLS_LE | Offset relative to static TLS block. |
| R_386_TLS_GD | Direct 32 bit for GNU version of general dynamic thread local data. |
| R_386_TLS_LDM | Direct 32 bit for GNU version of local dynamic thread local data in LE code. |
| R_386_TLS_GD_32 | Direct 32 bit for general dynamic thread local data. |
| R_386_TLS_GD_PUSH | Tag for pushl in GD TLS code. |
| R_386_TLS_GD_CALL | Relocation for call to <code>__tls_get_addr()</code> . |
| R_386_TLS_GD_POP | Tag for popl in GD TLS code. |
| R_386_TLS_LDM_32 | Direct 32 bit for local dynamic thread local data in LE code. |
| R_386_TLS_LDM_PUSH | Tag for pushl in LDM TLS code. |
| R_386_TLS_LDM_CALL | Relocation for call to <code>__tls_get_addr()</code> in LDM code. |
| R_386_TLS_LDM_POP | Tag for popl in LDM TLS code. |
| R_386_TLS_LDO_32 | Offset relative to TLS block. |
| R_386_TLS_IE_32 | GOT entry for negated static TLS block offset. |
| R_386_TLS_LE_32 | Negated offset relative to static TLS block. |
| R_386_TLS_DTPMOD32 | ID of module containing symbol. |
| R_386_TLS_DTPOFF32 | Offset in TLS block. |
| R_386_TLS_TPOFF32 | Negated offset in static TLS block. |
| R_386_NUM | Keep this the last entry. |

Definition at line 677 of file [elf.h](#).

13.16.9.3.22 Elf_R_AARCH64_s

enum [Elf_R_AARCH64_s](#)

AARCH64 relocations.

Enumerator

| | |
|----------------|-----------|
| R_AARCH64_NONE | No reloc. |
|----------------|-----------|

Definition at line [825](#) of file [elf.h](#).

13.16.9.3.23 Elf_R_ARM_s

enum [Elf_R_ARM_s](#)

ARM relocations.

Enumerator

| | |
|-----------------|----------------------------------|
| R_ARM_NONE | No reloc. |
| R_ARM_PC24 | PC relative 26 bit branch. |
| R_ARM_ABS32 | Direct 32 bit. |
| R_ARM_REL32 | PC relative 32 bit. |
| R_ARM_ABS16 | Direct 16 bit. |
| R_ARM_ABS12 | Direct 12 bit. |
| R_ARM_ABS8 | Direct 8 bit. |
| R_ARM_COPY | Copy symbol at runtime. |
| R_ARM_GLOB_DAT | Create GOT entry. |
| R_ARM_JUMP_SLOT | Create PLT entry. |
| R_ARM_RELATIVE | Adjust by program base. |
| R_ARM_GOTOFF | 32 bit offset to GOT |
| R_ARM_GOTPC | 32 bit PC relative offset to GOT |
| R_ARM_GOT32 | 32 bit GOT entry |
| R_ARM_PLT32 | 32 bit PLT address |
| R_ARM_THM_PC11 | thumb unconditional branch |
| R_ARM_THM_PC9 | thumb conditional branch |
| R_ARM_NUM | Keep this the last entry. |

Definition at line [777](#) of file [elf.h](#).

13.16.9.3.24 Elf_R_X86_64_s

enum [Elf_R_X86_64_s](#)

AMD x86-64 relocations.

Enumerator

| | |
|--------------------|---|
| R_X86_64_NONE | No reloc. |
| R_X86_64_64 | Direct 64 bit. |
| R_X86_64_PC32 | PC relative 32 bit signed. |
| R_X86_64_GOT32 | 32 bit GOT entry |
| R_X86_64_PLT32 | 32 bit PLT address |
| R_X86_64_COPY | Copy symbol at runtime. |
| R_X86_64_GLOB_DAT | Create GOT entry. |
| R_X86_64_JUMP_SLOT | Create PLT entry. |
| R_X86_64_RELATIVE | Adjust by program base. |
| R_X86_64_GOTPCREL | 32 bit signed PC relative offset to GOT |
| R_X86_64_32 | Direct 32 bit zero extended. |
| R_X86_64_32S | Direct 32 bit sign extended. |
| R_X86_64_16 | Direct 16 bit zero extended. |
| R_X86_64_PC16 | 16 bit sign extended pc relative |
| R_X86_64_8 | Direct 8 bit sign extended. |
| R_X86_64_PC8 | 8 bit sign extended pc relative |
| R_X86_64_DTPOFF64 | ID of module containing symbol. |
| R_X86_64_DTPOFF64 | Offset in module's TLS block. |
| R_X86_64_TPOFF64 | Offset in initial TLS block. |
| R_X86_64_TLSGD | 32 bit signed PC relative offset to two GOT entries for GD symbol |
| R_X86_64_TLSLD | 32 bit signed PC relative offset to two GOT entries for LD symbol |
| R_X86_64_DTPOFF32 | Offset in TLS block. |
| R_X86_64_GOTTPOFF | 32 bit signed PC relative offset to GOT entry for IE symbol |
| R_X86_64_TPOFF32 | Offset in initial TLS block. |

Definition at line [832](#) of file [elf.h](#).

13.16.9.3.25 Elf_SHF_s_ARM

enum [Elf_SHF_s_ARM](#)

ARM-specific values for [Elf32_Shdr.sh_flags](#) / [Elf64_Shdr.sh_flags](#).

Enumerator

| | |
|-------------------|----------------------------------|
| SHF_ARM_ENTRYSECT | Section contains an entry point. |
|-------------------|----------------------------------|

| | |
|----------------|--|
| SHF_ARM_COMDEF | Section may be multiply defined in the input to a link step. |
|----------------|--|

Definition at line 762 of file [elf.h](#).

13.16.9.3.26 Elf_SHFs

enum [Elf_SHFs](#)

Section attribute flags.

Enumerator

| | |
|----------------------|---|
| SHF_WRITE | writeable during execution |
| SHF_ALLOC | section occupies virt memory |
| SHF_EXECINSTR | code section |
| SHF_MERGE | Might be merged. |
| SHF_STRINGS | Contains nul-terminated strings. |
| SHF_INFO_LINK | 'sh_info' contains SHT index |
| SHF_LINK_ORDER | Preserve order after combining. |
| SHF_OS_NONCONFORMING | Non-standard OS-specific handling required. |
| SHF_GROUP | Section is member of a group. |
| SHF_TLS | Section hold thread-local data. |
| SHF_MASKOS | OS-specific. |
| SHF_MASKPROC | processor-specific mask |

Definition at line 406 of file [elf.h](#).

13.16.9.3.27 Elf_SHNs

enum [Elf_SHNs](#)

Special section indexes.

Enumerator

| | |
|---------------|---------------------------------|
| SHN_UNDEF | undefined section header entry |
| SHN_LORESERVE | lower bound of reserved indexes |
| SHN_LOPROC | lower bound of proc spec entr |
| SHN_HIPROC | upper bound of proc spec entr |
| SHN_ABS | absolute values for ref |
| SHN_COMMON | common symbols |
| SHN_HIRESERVE | upper bound of reserved indexes |

Definition at line 335 of file [elf.h](#).

13.16.9.3.28 Elf_SHTs

enum [Elf_SHTs](#)

Section type.

Enumerator

| | |
|-------------------|------------------------------------|
| SHT_NULL | inactive section header |
| SHT_PROGBITS | information defined by program |
| SHT_SYMTAB | symbol table |
| SHT_STRTAB | string table |
| SHT_RELA | reloc entries w/ explicit addends |
| SHT_HASH | symbol hash table |
| SHT_DYNAMIC | information for dynamic linking |
| SHT_NOTE | information that marks the file |
| SHT_NOBITS | occupies no space in the file |
| SHT_REL | reloc entries w/o explicit addends |
| SHT_SHLIB | reserved + unspecified semantics |
| SHT_DYNSYM | symbol table (dynamic) |
| SHT_INIT_ARRAY | Array of constructors. |
| SHT_FINI_ARRAY | Array of destructors. |
| SHT_PREINIT_ARRAY | Array of pre-constructors. |
| SHT_GROUP | Section group. |
| SHT_SYMTAB_SHNDX | Extended section indices. |
| SHT_NUM | Number of defined types. |
| SHT_LOOS | Start OS-specific. |
| SHT_HIOS | End OS-specific. |
| SHT_LOPROC | Start processor-specific. |
| SHT_HIPROC | End processor-specific. |
| SHT_LOUSER | Start application-specific. |
| SHT_HIUSER | End application-specific. |

Definition at line [377](#) of file [elf.h](#).

13.16.9.3.29 Elf_STBs

enum [Elf_STBs](#)

Symbol Binding.

See also

[ELF32_ST_BIND](#), [ELF64_ST_BIND](#)

Enumerator

| | |
|------------|---------------------------------------|
| STB_LOCAL | not visible outside object file |
| STB_GLOBAL | visible to all objects being combined |
| STB_WEAK | resemble global symbols |
| STB_LOOS | OS-specific. |
| STB_HIOS | OS-specific. |
| STB_LOPROC | Processor-specific. |
| STB_HIPROC | Processor-specific. |

Definition at line 912 of file [elf.h](#).

13.16.9.3.30 Elf_STTs

enum [Elf_STTs](#)

Symbol Types.

See also

[ELF32_ST_TYPE](#), [ELF64_ST_TYPE](#)

Enumerator

| | |
|-------------|--|
| STT_NOTYPE | symbol's type not specified |
| STT_OBJECT | associated with a data object |
| STT_FUNC | associated with a function or other code |
| STT_SECTION | associated with a section |
| STT_FILE | source file name associated with object |
| STT_LOOS | OS-specific. |
| STT_HIOS | OS-specific. |
| STT_LOPROC | processor-specific |
| STT_HIPROC | processor-specific |

Definition at line 925 of file [elf.h](#).

13.16.10 Kernel Interface Page API

Collaboration diagram for Kernel Interface Page API:



Files

- file [kip.h](#)

Macros

- `#define l4util_kip_for_each_feature(s)`
Cycle through kernel features given in the KIP.

Functions

- `L4_BEGIN_DECLS int l4util_kip_kernel_has_feature (l4_kernel_info_t const *k, char const *str)`
Check if kernel supports a feature.
- `unsigned long l4util_kip_kernel_abi_version (l4_kernel_info_t const *k)`
Return kernel ABI version.

13.16.10.1 Detailed Description

13.16.10.2 Macro Definition Documentation

13.16.10.2.1 l4util_kip_for_each_feature

```
#define l4util_kip_for_each_feature(  
    s)
```

Value:

`l4_kip_for_each_feature(s)`

Cycle through kernel features given in the KIP.

Cycles through all KIP kernel feature strings. `s` must be a character pointer (`char const *`) initialized with `l4_kip_version_string()`.

Deprecated Use `l4_kip_for_each_feature()`.

Definition at line 58 of file [kip.h](#).

13.16.10.3 Function Documentation

13.16.10.3.1 l4util_kip_kernel_abi_version()

```
unsigned long l4util_kip_kernel_abi_version (  
    l4_kernel_info_t const * k)
```

Return kernel ABI version.

Parameters

| | |
|----------|--|
| <i>k</i> | Pointer to the kernel info page (KIP). |
|----------|--|

Returns

Kernel ABI version.

References [L4_CV](#), and [L4_END_DECLS](#).

13.16.10.3.2 l4util_kip_kernel_has_feature()

```
L4_BEGIN_DECLS int l4util_kip_kernel_has_feature (  
    l4_kernel_info_t const * k,  
    char const * str)
```

Check if kernel supports a feature.

Parameters

| | |
|------------|--|
| <i>k</i> | Pointer to the kernel info page (KIP). |
| <i>str</i> | Feature name to check. |

Returns

1 if the kernel supports the feature, 0 if not.

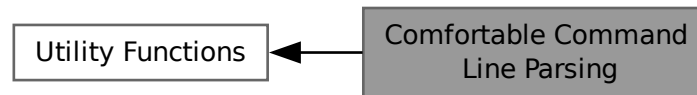
Checks the feature field in the KIP for the given string.

Deprecated Use [l4_kip_kernel_has_feature\(\)](#).

References [L4_CV](#).

13.16.11 Comfortable Command Line Parsing

Collaboration diagram for Comfortable Command Line Parsing:



Typedefs

- typedef void(* **parse_cmd_fn_t**) (int)
Function type for PARSE_CMD_FN.
- typedef void(* **parse_cmd_fn_arg_t**) (int, const char *, int)
Function type for PARSE_CMD_FN_ARG.

Enumerations

- enum [parse_cmd_type](#)
Types for parsing.

Functions

- [L4_BEGIN_DECLS](#) int [parse_cmdline](#) (int *argc, const char ***argv, int arg0,...)
Parse the command-line for specified arguments and store the values into variables.

13.16.11.1 Detailed Description

13.16.11.2 Function Documentation

13.16.11.2.1 parse_cmdline()

```

L4_BEGIN_DECLS int parse_cmdline (
    int * argc,
    const char *** argv,
    int arg0,
    ...)
  
```

Parse the command-line for specified arguments and store the values into variables.

This Functions gets the command-line, and a list of command-descriptors. Then, the command-line is parsed according to the given descriptors, storing strings, switches and numeric arguments at given addresses, and possibly calling specified functions. A default help descriptor is added. Its purpose is to present a short command overview in the case the given command-line does not fit to the descriptors.

Each command-descriptor has the following form:

short option char, long option name, comment, type, val, addr.

The *short option char* specifies the short form of the described option. The short form will be recognized after a single dash, or in a group of short options preceeded by a single dash. Specify '' if no short form should be used.

The *long option name* specifies the long form of the described option. The long form will be recognized after two dashes. Specify 0 if no long form should be used for this option.

The *comment* is a string that will be used when presenting the short command-line help.

The *type* specifies, if the option should be recognized as

- a number (PARSE_CMD_INT),
- a switch (PARSE_CMD_SWITCH),
- a string (PARSE_CMD_STRING),
- a function call (PARSE_CMD_FN, PARSE_CMD_FN_ARG),
- an increment/decrement operator (PARSE_CMD_INC, PARSE_CMD_DEC).

If *type* is PARSE_CMD_INT, the option requires a second argument on the command-line after the option. This argument is parsed as a number. It can be preceeded by 0x to present a hex-value or by 0 to present an octal form. *addr* is interpreted as an int-pointer. The scanned argument from the command-line is stored in this pointer.

If *type* is PARSE_CMD_SWITCH, *addr* must be a pointer to int, and the value from *val* is stored at this pointer.

With PARSE_CMD_STRING, an additional argument is expected at the cmdline. *addr* must be a pointer to const char*, and a pointer to the argument on the command line is stored at this pointer. The value in *val* is a default value, which is stored at *addr* if the corresponding option is not given on the command line.

With PARSE_CMD_FN_ARG, *addr* is interpreted as a function pointer of type [parse_cmd_fn_t](#). It will be called with *val* as argument if the corresponding option is found.

If *type* is PARSE_CMD_FN_ARG, *addr* is as a function pointer of type [parse_cmd_fn_arg_t](#), and handled similar to PARSE_CMD_FN. An additional argument is expected at the command line, however. It is given to the called function as 2nd argument, and parsed as an integer as with PARSE_CMD_INT as a third argument.

If *type* is PARSE_CMD_INC or PARSE_CMD_DEC, *addr* is interpreted as an int-pointer. The value of *val* is stored to this pointer first. For every occurrence of the option in the command line, the integer referenced by *addr* is incremented or decremented, respectively.

The list of command-descriptors is terminated by specifying a binary 0 for the short option char.

Note: The short option char 'h' and the long option name "help" must not be specified. They are used for the default help descriptor and produce a short command-options help when specified on the command-line.

Parameters

| | |
|-------------|--|
| <i>argc</i> | pointer to number of command line parameters as passed to main |
| <i>argv</i> | pointer to array of command line parameters as passed to main |
| <i>arg0</i> | format list describing the command line options to parse for |

Returns

0 if the command-line was successfully parsed, otherwise:

- -1 if the given descriptors are somehow wrong.
- -2 if not enough memory was available to hold temporary structs.
- -3 if the given command-line args did not meet the specified set.
- -4 if the help-option was given.

Upon return, argc and argv point to a list of arguments that were not scanned as arguments. See `getoptlong` for details on scanning.

References [L4_CV](#), and [L4_END_DECLS](#).

13.16.12 Random number support

Collaboration diagram for Random number support:



Functions

- [l4_uint32_t](#) `l4util_rand` (void)
Deliver next random number.
- void `l4util_srand` ([l4_uint32_t](#) seed)
Initialize random number generator.

13.16.12.1 Detailed Description

13.16.12.2 Function Documentation

13.16.12.2.1 l4util_rand()

```
l4_uint32_t l4util_rand (  
    void )
```

Deliver next random number.

Returns

A new random number

References [L4_CV](#).

13.16.12.2 l4util_srand()

```
void l4util_srand (
    l4_uint32_t seed)
```

Initialize random number generator.

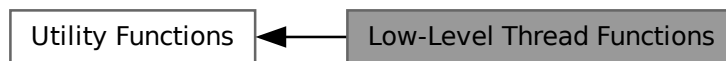
Parameters

| | |
|-------------|---------------------|
| <i>seed</i> | Value to initialize |
|-------------|---------------------|

References [L4_END_DECLS](#).

13.16.13 Low-Level Thread Functions

Collaboration diagram for Low-Level Thread Functions:



13.17 Virtio Net Switch

A virtual network switch that can be used as defined in the virtio protocol.

Data Structures

- class [Mac_addr](#)
A wrapper class around the value of a MAC address.
- class [Mac_table](#)< [Size](#) >
Mac_table manages a 1:n association between ports and MAC addresses.
- class [Switch_factory](#)
The IPC interface for creating ports.
- class [L4virtio_port](#)
A Port on the Virtio Net Switch.
- class [Net_transfer](#)
A network request to only a single destination.
- class [Virtio_net_request](#)
Abstraction for a network request.
- class [Virtio_switch](#)
The Virtio switch contains all ports and processes network requests.
- struct [Buffer](#)
Data buffer used to transfer packets.
- class [Virtio_vlan_mangle](#)
Class for VLAN packet rewriting.

13.17.1 Detailed Description

A virtual network switch that can be used as defined in the virtio protocol.

The abstraction of a single connection with a network device (also called client) from the switch's perspective is a port. A client can register multiple ports on the switch. The communication between a client and the switch happens via IRQs, MMIO and shared memory as defined by the Virtio protocol. The switch supports VLANs and ports can be either 'access' or 'trunk' ports. The optionally available monitor port receives network traffic from all ports, and the monitor can not send.

13.18 vCPU Support Library

vCPU handling functionality.

Collaboration diagram for vCPU Support Library:



Topics

- [Extended vCPU support](#) [732](#)
Extended vCPU handling functionality.

Data Structures

- class [L4vcpu::State](#)
C++ implementation of state word in the vCPU area.
- class [L4vcpu::Vcpu](#)
C++ implementation of the vCPU save state area.

Functions

- void [l4vcpu_irq_disable](#) ([l4_vcpu_state_t](#) *vcpu) [L4_NOTHROW](#)
Disable a vCPU for event delivery.
- unsigned [l4vcpu_irq_disable_save](#) ([l4_vcpu_state_t](#) *vcpu) [L4_NOTHROW](#)
Disable a vCPU for event delivery and return previous state.
- void [l4vcpu_irq_enable](#) ([l4_vcpu_state_t](#) *vcpu, [l4_utcb_t](#) *utcb, [l4vcpu_event_hndl_t](#) do_event_work_cb, [l4vcpu_setup_ipc_t](#) setup_ipc) [L4_NOTHROW](#)
Enable a vCPU for event delivery.
- void [l4vcpu_irq_restore](#) ([l4_vcpu_state_t](#) *vcpu, unsigned s, [l4_utcb_t](#) *utcb, [l4vcpu_event_hndl_t](#) do_event_work_cb, [l4vcpu_setup_ipc_t](#) setup_ipc) [L4_NOTHROW](#)

Restore a previously saved IRQ/event state.

- void [l4vcpu_wait_for_event](#) ([l4_vcpu_state_t](#) *vcpu, [l4_utcb_t](#) *utcb, [l4vcpu_event_hndl_t](#) do_event_work←_cb, [l4vcpu_setup_ipc_t](#) setup_ipc) [L4_NOTHROW](#)

Wait for event.

- void [l4vcpu_print_state](#) (const [l4_vcpu_state_t](#) *vcpu, const char *prefix) [L4_NOTHROW](#)

Print the state of a vCPU.

- int [l4vcpu_is_irq_entry](#) ([l4_vcpu_state_t](#) const *vcpu) [L4_NOTHROW](#)

Return whether the entry reason was an IRQ/IPC message.

- int [l4vcpu_is_page_fault_entry](#) ([l4_vcpu_state_t](#) const *vcpu) [L4_NOTHROW](#)

Return whether the entry reason was a page fault.

13.18.1 Detailed Description

vCPU handling functionality.

This library provides convenience functionality on top of the l4sys vCPU interface to ease programming. It wraps commonly used code and abstracts architecture depends parts as far as reasonable.

13.18.2 Function Documentation

13.18.2.1 l4vcpu_irq_disable()

```
void l4vcpu_irq_disable (
    l4\_vcpu\_state\_t * vcpu) [inline]
```

Disable a vCPU for event delivery.

Parameters

| | |
|-------------|-----------------------|
| <i>vcpu</i> | Pointer to vCPU area. |
|-------------|-----------------------|

Definition at line 201 of file [vcpu.h](#).

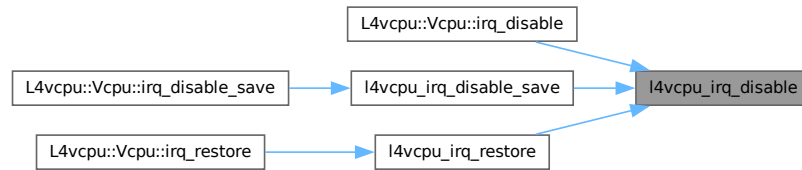
References [l4_barrier\(\)](#), [L4_NOTHROW](#), and [L4_VCPU_F_IRQ](#).

Referenced by [L4vcpu::Vcpu::irq_disable\(\)](#), [l4vcpu_irq_disable_save\(\)](#), and [l4vcpu_irq_restore\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.18.2.2 l4vcpu_irq_disable_save()

```
unsigned l4vcpu_irq_disable_save (
    l4_vcpu_state_t * vcpu) [inline]
```

Disable a vCPU for event delivery and return previous state.

Parameters

| | |
|-------------|-----------------------|
| <i>vcpu</i> | Pointer to vCPU area. |
|-------------|-----------------------|

Returns

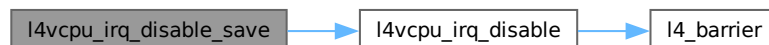
IRQ state before disabling IRQs.

Definition at line 209 of file [vcpu.h](#).

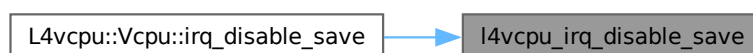
References [L4_NOTHROW](#), and [l4vcpu_irq_disable\(\)](#).

Referenced by [L4vcpu::Vcpu::irq_disable_save\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.18.2.3 l4vcpu_irq_enable()

```
void l4vcpu_irq_enable (
    l4_vcpu_state_t * vcpu,
    l4_utcb_t * utcb,
    l4vcpu_event_hndl_t do_event_work_cb,
    l4vcpu_setup_ipc_t setup_ipc) [inline]
```

Enable a vCPU for event delivery.

Parameters

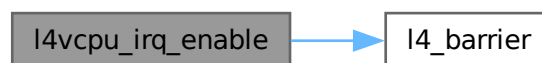
| | |
|-------------------------|---|
| <i>vcpu</i> | Pointer to vCPU area. |
| <i>utcb</i> | Utcbl pointer of the calling vCPU. |
| <i>do_event_work_cb</i> | Call-back function that is called in case an event (such as an interrupt) is pending. |
| <i>setup_ipc</i> | Function call-back that is called right before any IPC operation, and before event delivery is enabled. |

Definition at line 232 of file [vcpu.h](#).

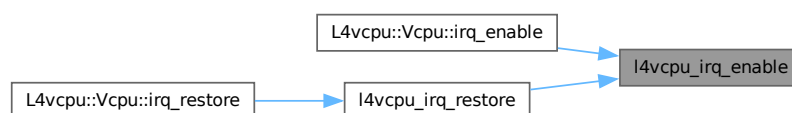
References [l4_barrier\(\)](#), [L4_IPC_BOTH_TIMEOUT_0](#), [L4_LIKELY](#), [L4_NOTHROW](#), [L4_VCPU_F_IRQ](#), and [L4_VCPU_SF_IRQ_PENDING](#).

Referenced by [L4vcpu::Vcpu::irq_enable\(\)](#), and [l4vcpu_irq_restore\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.18.2.4 l4vcpu_irq_restore()

```
void l4vcpu_irq_restore (
    l4_vcpu_state_t * vcpu,
    unsigned s,
    l4_utcb_t * utcb,
    l4vcpu_event_hndl_t do_event_work_cb,
    l4vcpu_setup_ipc_t setup_ipc) [inline]
```

Restore a previously saved IRQ/event state.

Parameters

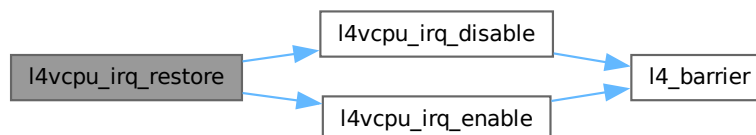
| | |
|-------------------------|---|
| <i>vcpu</i> | Pointer to vCPU area. |
| <i>s</i> | IRQ state to be restored. |
| <i>utcb</i> | Utcbl pointer of the calling vCPU. |
| <i>do_event_work_cb</i> | Call-back function that is called in case an event (such as an interrupt) is pending after enabling. |
| <i>setup_ipc</i> | Function call-back that is called right before any IPC operation, and before event delivery is enabled. |

Definition at line 257 of file [vcpu.h](#).

References [L4_NOTHROW](#), [L4_VCPU_F_IRQ](#), [l4vcpu_irq_disable\(\)](#), and [l4vcpu_irq_enable\(\)](#).

Referenced by [L4vcpu::Vcpu::irq_restore\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.18.2.5 l4vcpu_is_irq_entry()

```
int l4vcpu_is_irq_entry (
    l4_vcpu_state_t const * vcpu) [inline]
```

Return whether the entry reason was an IRQ/IPC message.

Parameters

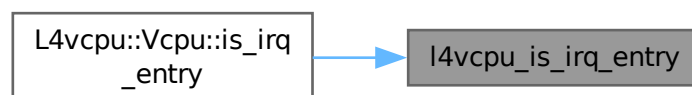
| | |
|-------------|-----------------------|
| <i>vcpu</i> | Pointer to vCPU area. |
|-------------|-----------------------|

return 0 if not, !=0 otherwise.

References [L4_CV](#), [L4_INLINE](#), and [L4_NOTHROW](#).

Referenced by [L4vcpu::Vcpu::is_irq_entry\(\)](#).

Here is the caller graph for this function:



13.18.2.6 l4vcpu_is_page_fault_entry()

```
int l4vcpu_is_page_fault_entry (
    l4_vcpu_state_t const * vcpu) [inline]
```

Return whether the entry reason was a page fault.

Parameters

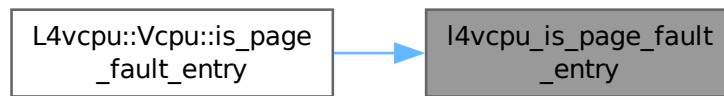
| | |
|-------------|-----------------------|
| <i>vcpu</i> | Pointer to vCPU area. |
|-------------|-----------------------|

return 0 if not, !=0 otherwise.

References [L4_CV](#), and [L4_NOTHROW](#).

Referenced by [L4vcpu::Vcpu::is_page_fault_entry\(\)](#).

Here is the caller graph for this function:



13.18.2.7 l4vcpu_print_state()

```
void l4vcpu_print_state (
    const l4_vcpu_state_t * vcpu,
    const char * prefix)
```

Print the state of a vCPU.

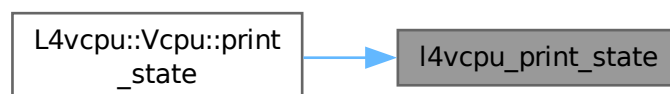
Parameters

| | |
|---------------|---------------------------------|
| <i>vcpu</i> | Pointer to vCPU area. |
| <i>prefix</i> | A prefix for each line printed. |

References [L4_CV](#), [L4_INLINE](#), and [L4_NOTHROW](#).

Referenced by [L4vcpu::Vcpu::print_state\(\)](#).

Here is the caller graph for this function:



13.18.2.8 l4vcpu_wait_for_event()

```
void l4vcpu_wait_for_event (
    l4_vcpu_state_t * vcpu,
    l4_utcb_t * utcb,
    l4vcpu_event_hndl_t do_event_work_cb,
    l4vcpu_setup_ipc_t setup_ipc) [inline]
```

Wait for event.

Parameters

| | |
|-------------------------|--|
| <i>vcpu</i> | Pointer to vCPU area. |
| <i>utcb</i> | Utc b pointer of the calling vCPU. |
| <i>do_event_work_cb</i> | Call-back function that is called when the vCPU awakes and needs to handle an event/↔ IRQ. |
| <i>setup_ipc</i> | Function call-back that is called right before any IPC operation. |

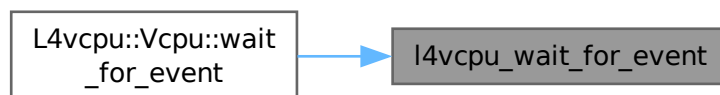
Note that event delivery remains disabled after this function returns.

Definition at line 270 of file [vcpu.h](#).

References [L4_IPC_NEVER](#), and [L4_NOTHROW](#).

Referenced by [L4vcpu::Vcpu::wait_for_event\(\)](#).

Here is the caller graph for this function:



13.18.3 Extended vCPU support

Extended vCPU handling functionality.

Collaboration diagram for Extended vCPU support:



Functions

- `int l4vcpu_ext_alloc (l4_vcpu_state_t **vcpu, l4_addr_t *ext_state, l4_cap_idx_t task, l4_cap_idx_t regmgr)`
[L4_NOTHROW](#)
Allocate state area for an extended vCPU.

13.18.3.1 Detailed Description

Extended vCPU handling functionality.

13.18.3.2 Function Documentation

13.18.3.2.1 l4vcpu_ext_alloc()

```
int l4vcpu_ext_alloc (
    l4_vcpu_state_t ** vcpu,
    l4_addr_t * ext_state,
    l4_cap_idx_t task,
    l4_cap_idx_t regmgr)
```

Allocate state area for an extended vCPU.

Parameters

| | | |
|-----|------------------|---------------------------------------|
| out | <i>vcpu</i> | Allocated vcpu-state area. |
| out | <i>ext_state</i> | Allocated extended vcpu-state area. |
| | <i>task</i> | Task to use for allocation. |
| | <i>regmgr</i> | Region manager to use for allocation. |

Returns

0 for success, error code otherwise

References [L4_CV](#), [L4_INLINE](#), and [L4_NOTHROW](#).

Chapter 14

Namespace Documentation

14.1 cxx Namespace Reference

Our C++ library.

Namespaces

- namespace [Bits](#)
Internal helpers for the cxx package.

Data Structures

- class [Avl_map](#)
AVL tree based associative container.
- class [Avl_set](#)
AVL set for simple comparable items.
- class [Avl_tree_node](#)
Node of an AVL tree.
- class [Avl_tree](#)
A generic AVL tree.
- class [Bitfield](#)
Definition for a member (part) of a bit field.
- class [Bitmap_base](#)
Basic bitmap abstraction.
- class [Bitmap](#)
A static bitmap.
- class [Elide_dtor](#)
Wrapper class to remove destructor calls.
- class [H_list_item_t](#)
Basic element type for a double-linked [H_list](#).
- class [H_list](#)
General double-linked list of unspecified [cxx::H_list_item](#) elements.
- struct [H_list_t](#)
Double-linked list of typed [H_list_item_t](#) elements.

- class [List_item](#)
Basic list item.
- class [List](#)
Doubly linked list, with internal allocation.
- class [List_alloc](#)
Standard list-based allocator.
- struct [Pair](#)
Pair of two values.
- class [Pair_first_compare](#)
Comparison functor for [Pair](#).
- class [Ref_ptr](#)
A reference-counting pointer with automatic cleanup.
- struct [Ref_obj_list_item](#)
Item for list linked via [cxx::Ref_ptr](#) with default reference counting.
- class [Error](#)
Error value.
- class [Result](#)
A result of a function call.
- class [Base_slab](#)
Basic slab allocator.
- class [Slab](#)
Slab allocator for object of type `Type`.
- class [Base_slab_static](#)
Merged slab allocator (allocators for objects of the same size are merged together).
- class [Slab_static](#)
Merged slab allocator (allocators for objects of the same size are merged together).
- class [S_list](#)
Simple single-linked list.
- class [static_vector](#)
Simple encapsulation for a dynamically allocated array.
- class [Nothrow](#)
Helper type to distinguish the `operator new` version that does not throw exceptions.
- class [New_allocator](#)
Standard allocator based on `operator new ()`.
- struct [Lt_functor](#)
Generic comparator class that defaults to the less-than operator.
- class [String](#)
Allocation free string class with explicit length field.
- class [Weak_ref_base](#)
Generic (base) weak reference to some object.
- class [Weak_ref](#)
Typed weak reference to an object of type `T`.

Typedefs

- typedef [H_list_item_t](#)< void > [H_list_item](#)
Untyped list item.
- template<typename T>
using [Ref_ptr_list_item](#) = [Bits::Smart_ptr_list_item](#)<T, [cxx::Ref_ptr](#)<T> >
Item for list linked with [cxx::Ref_ptr](#).

- `template<typename T>`
`using Ref_ptr_list = Bits::Smart_ptr_list<Ref_ptr_list_item<T> >`
Single-linked list where elements are connected via a `cxx::Ref_ptr`.
- `template<typename T>`
`using Unique_ptr_list_item = Bits::Smart_ptr_list_item<T, cxx::unique_ptr<T> >`
Item for list linked with `cxx::unique_ptr`.
- `template<typename T>`
`using Unique_ptr_list = Bits::Smart_ptr_list<Unique_ptr_list_item<T> >`
Single-linked list where elements are connected with a `cxx::unique_ptr`.

Functions

- `template<typename A, typename ... ARGS>`
`constexpr A const & min (A const &a1, A const &a2, ARGS const &...a)`
Get the minimum of `a1` and `a2` up to `aN`.
- `template<typename A, typename ... ARGS>`
`constexpr A const & min (cxx::identity_t< A > const &a1, cxx::identity_t< A > const &a2, ARGS const &...a)`
Get the minimum of `a1` and `a2` up to `aN`.
- `template<typename A, typename ... ARGS>`
`constexpr A const & max (A const &a1, A const &a2, ARGS const &...a)`
Get the maximum of `a1` and `a2` up to `aN`.
- `template<typename A, typename ... ARGS>`
`constexpr A const & max (cxx::identity_t< A > const &a1, cxx::identity_t< A > const &a2, ARGS const &...a)`
Get the maximum of `a1` and `a2` up to `aN`.
- `template<typename T1>`
`T1 clamp (T1 v, T1 lo, T1 hi)`
Limit `v` to the range given by `lo` and `hi`.
- `template<typename T>`
`constexpr T gcd (T a, T b)`
Compute the greatest common divisor of two unsigned values.
- `template<typename T>`
`constexpr T lcm (T a, T b)`
Compute the least common multiple of two unsigned values.
- `template<typename T>`
`T access_once (T const *a)`
Read the value at an address at most once.
- `template<typename T, typename VAL>`
`void write_now (T *a, VAL &&val)`
Write a value at an address exactly once.

14.1.1 Detailed Description

Our C++ library.

Small Low-Level C++ Library.

Strings.

Various kinds of C++ utilities.

14.1.2 Function Documentation

14.1.2.1 access_once()

```
template<typename T>
T cxx::access_once (
    T const * a) [inline]
```

Read the value at an address at most once.

The read might be omitted if the result is not used by any code unless `typename` contains `volatile`. If the read operation has side effects and must not be omitted, use different means like [L4drivers::Mmio_register_block](#) or similar.

The compiler is disallowed to reuse a previous read at the same address, for example:

```
val1 = *a;
val2 = access_once(a); // compiler may not replace this by val2 = val1;
```

The compiler is also disallowed to repeat the read, for example:

```
val1 = access_once(a);
val2 = val1; // compiler may not replace this by val2 = *a;
```

The above implies that the compiler is also disallowed to move the read out of or into loops.

Note

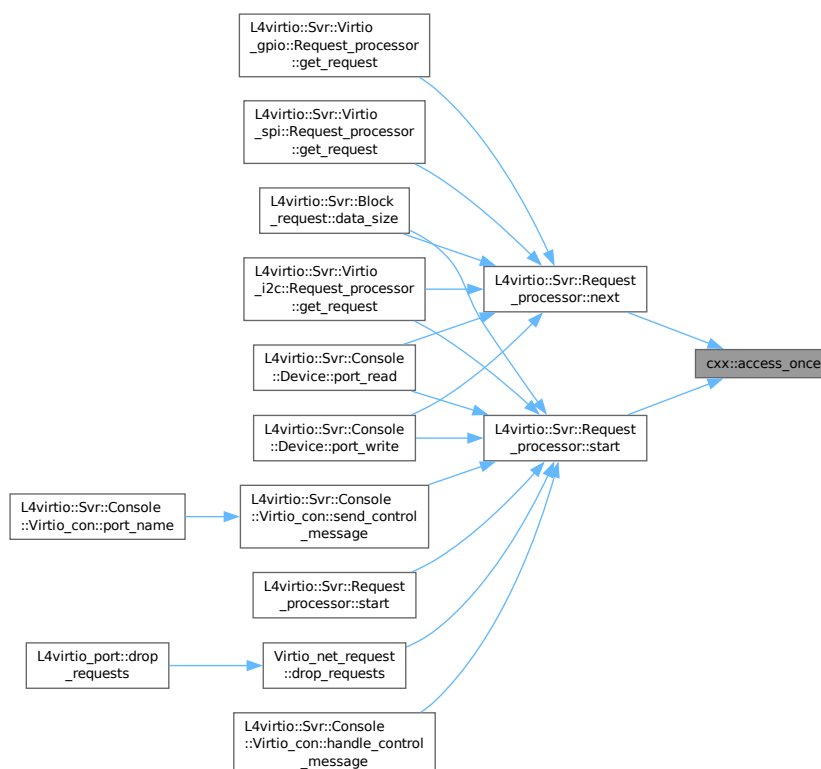
The read might still be moved relative to other code.

The value might be read from a hardware cache, not from RAM.

Definition at line 40 of file [utils](#).

Referenced by [L4virtio::Svr::Request_processor::next\(\)](#), and [L4virtio::Svr::Request_processor::start\(\)](#).

Here is the caller graph for this function:



14.1.2.2 gcd()

```
template<typename T>
T cxx::gcd (
    T a,
    T b) [constexpr]
```

Compute the greatest common divisor of two unsigned values.

This uses the Euclidean modulo algorithm.

Note

Contrary to the C++17 specification, this implementation requires the same unsigned type of both arguments and returns the same type (not a common type). This should be fine for most practical use cases.

Contrary to the C++17 specification, this implementation does not accept signed arguments and negative literals.

Template Parameters

| | |
|----------|--------------------------------------|
| <i>T</i> | Unsigned integer type of the values. |
|----------|--------------------------------------|

Parameters

| | |
|----------|---------------|
| <i>a</i> | First input. |
| <i>b</i> | Second input. |

Returns

Greatest common divisor of the input values.

Definition at line 34 of file [numeric](#).

Referenced by [lcm\(\)](#).

Here is the caller graph for this function:



14.1.2.3 lcm()

```
template<typename T>
T cxx::lcm (
    T a,
    T b) [constexpr]
```

Compute the least common multiple of two unsigned values.

This uses the greatest common divisor to compute the least common multiple.

Note

Contrary to the C++17 specification, this implementation requires the same unsigned type of both arguments and returns the same type (not a common type). This should be fine for most practical use cases.

Contrary to the C++17 specification, this implementation does not accept signed arguments and negative literals.

Template Parameters

| | |
|----------|--------------------------------------|
| <i>T</i> | Unsigned integer type of the values. |
|----------|--------------------------------------|

Parameters

| | |
|----------|---------------|
| <i>a</i> | First input. |
| <i>b</i> | Second input. |

Returns

Least common multiple of the input values.

Definition at line 68 of file [numeric](#).

References [gcd\(\)](#).

Here is the call graph for this function:



14.1.2.4 write_now()

```
template<typename T, typename VAL>
void cxx::write_now (
    T * a,
    VAL && val) [inline]
```

Write a value at an address exactly once.

The compiler is disallowed to skip the write, for example:

```
*a = val;
write_now(a, val); // compiler may not skip this line
```

The compiler is also disallowed to repeat the write.

The above implies that the compiler is also disallowed to move the write out of or into loops.

Note

The write might still be moved relative to other code.

The value might be written just to a hardware cache for the moment, not immediately to RAM.

Definition at line 71 of file [utils](#).

14.2 cxx::Bits Namespace Reference

Internal helpers for the cxx package.

Data Structures

- struct [Avl_map_get_key](#)
Key-getter for [Avl_map](#).
- class [Avl_set_iter](#)
Generic iterator for the AVL-tree based set.
- struct [Avl_set_get_key](#)
Internal, key-getter for [Avl_set](#) nodes.
- class [Base_avl_set](#)
AVL set with internally managed nodes.
- class [Bst](#)
Basic binary search tree (BST).
- struct [Direction](#)
The direction to go in a binary search tree.
- class [Bst_node](#)
Basic type of a node in a binary search tree (BST).
- class [Basic_list](#)
Internal: Common functions for all head-based list implementations.
- class [Smart_ptr_list_item](#)
List item for an arbitrary item in a [Smart_ptr_list](#).
- class [Smart_ptr_list](#)
List of smart-pointer-managed objects.

14.2.1 Detailed Description

Internal helpers for the cxx package.

14.3 Enum_bitops Namespace Reference

Mechanism to opt-in for enum bitwise operators.

Data Structures

- struct [Has_marker](#)
Marker for the opt-in ADL function.
- struct [Has_marker](#)< T, Void< decltype(enum_bitops_enable(declval< T >()))> >
Marker for the opt-in ADL function.
- struct [Enable](#)
Check whether the given enum type opts in for the bitwise operators.

14.3.1 Detailed Description

Mechanism to opt-in for enum bitwise operators.

For an enumeration type T to opt-in for bitwise operators, there needs to be a declaration of a function

```
void enum_bitops_enable(T)
```

available for the argument-dependent lookup (the function is never actually called, thus no definition is required and no code is generated).

14.4 Enum_bitops_impl Namespace Reference

Bitwise operators on enumeration types.

Functions

- template<typename T, typename = L4::Types::Enable_if_t<L4::Types::Is_enum<T>::value>>
constexpr [L4::Types::Underlying_type_t](#)< T > **to_underlying** (T const arg) noexcept
Convert enum value to its underlying type value.
- template<typename T, typename = L4::Types::Enable_if_t<Enum_bitops::Enable<T>::value>>
constexpr T **operator~** (T const a) noexcept
Negate enum value.
- template<typename T, typename = L4::Types::Enable_if_t<Enum_bitops::Enable<T>::value>>
constexpr T **operator&** (T l, T r) noexcept
Intersect enum values.
- template<typename T, typename = L4::Types::Enable_if_t<Enum_bitops::Enable<T>::value>>
constexpr T & **operator&=** (T &a, T const b) noexcept
Intersect and assign enum values.

- `template<typename T, typename = L4::Types::Enable_if_t<Enum_bitops::Enable<T>::value>>`
`constexpr T operator| (T l, T r) noexcept`
Union enum values.
- `template<typename T, typename = L4::Types::Enable_if_t<Enum_bitops::Enable<T>::value>>`
`constexpr T & operator|= (T &a, T const b) noexcept`
Union and assign enum values.
- `template<typename T, typename = L4::Types::Enable_if_t<Enum_bitops::Enable<T>::value>>`
`constexpr T operator- (T l, T r) noexcept`
Difference (intersect with negation, clear bits) enum values.
- `template<typename T, typename = L4::Types::Enable_if_t<Enum_bitops::Enable<T>::value>>`
`constexpr T & operator-= (T &a, T const b) noexcept`
Difference (intersect with negation, clear bits) and assign enum values.

14.4.1 Detailed Description

Bitwise operators on enumeration types.

These operators allow to use the enum type as a bitmask type with '~', '&' and '|' operators that keep the enum type as result. The compound assignment operators '&=' and '|=' are also provided.

The set-difference '-' and '-=' operators (a shorthand for intersection with a negated second argument) can be used to clear specific bits from the enum value. There is also the [to_underlying\(\)](#) helper function.

14.5 L4 Namespace Reference

[L4](#) low-level kernel interface.

Namespaces

- namespace [Typeid](#)
Definition of interface data-type helpers.
- namespace [lpc](#)
IPC related functionality.
- namespace [lpc_svr](#)
Helper classes for [L4::Server](#) instantiation.
- namespace [Types](#)
[L4](#) basic type helpers for C++.

Data Structures

- struct [Type_info](#)
Dynamic Type Information for [L4Re](#) Interfaces.
- struct [Kobject_typeid](#)
Meta object for handling access to type information of Kobjects.
- struct [Kobject_typeid< void >](#)
Minimalistic ID for `void` interface.
- class [Kobject_t](#)
Helper class to create an [L4Re](#) interface class that is derived from a single base class.
- class [Kobject_2t](#)
Helper class to create an [L4Re](#) interface class that is derived from two base classes (see [L4::Kobject_t](#)).
- struct [Kobject_3t](#)
Helper class to create an [L4Re](#) interface class that is derived from three base classes (see [L4::Kobject_t](#)).
- struct [Kobject_demand](#)
Get the combined server-side resource requirements for all type `T...`
- struct [Proto_t](#)
Data type for defining protocol numbers.
- struct [Kobject_x](#)
Generic [Kobject](#) inheritance template.
- class [Arm_smccc](#)
Wrapper for function calls that follow the ARM SMC/HVC calling convention.
- class [Cap](#)
C++ interface for capabilities.
- class [Cap_base](#)
Base class for all kinds of capabilities.
- class [Reply_cap_idx](#)
Value class for a reply capability index.
- class [Reply_cap_alloc](#)
Interface to a reply capability allocator.
- class [Reply_cap](#)
An explicit reply capability.
- struct [Epiface](#)
Base class for interface implementations.
- struct [Epiface_t0](#)
[Epiface](#) mixin for generic Kobject-based interfaces.
- struct [Irqep_t](#)
[Epiface](#) implementation for interrupt handlers.
- class [Registry_iface](#)
Abstract interface for object registries.
- struct [Epiface_t](#)
[Epiface](#) implementation for Kobject-based interface implementations.
- class [Basic_registry](#)
This registry returns the corresponding server object based on the label of an [lpc_gate](#).
- class [Server](#)
Basic server loop for handling client requests.
- class [Debugger](#)
C++ kernel debugger API.
- class [Exception](#)
[Exception](#) interface.
- class [Factory](#)

- C++ Factory interface, see [Factory](#) for the C interface.*
- class [lommu](#)
 - Interface for IO-MMUs used for DMA remapping.*
- class [lpc_gate](#)
 - The C++ IPC gate interface, see [IPC-Gate API](#) for the C interface.*
- class [lrq_eoi](#)
 - Interface for sending an unmask message to an object.*
- struct [Triggerable](#)
 - Interface that allows an object to be triggered by some source.*
- class [lrq](#)
 - C++ [lrq](#) interface, see [IRQs](#) for the C interface.*
- class [lcu](#)
 - C++ [lcu](#) interface, see [Interrupt controller](#) for the C interface.*
- class [Kobject](#)
 - Base class for all kinds of kernel objects and remote objects, referenced by capabilities.*
- class [Meta](#)
 - [Meta](#) interface that shall be implemented by each [L4Re](#) object and gives access to the dynamic type information for [L4Re](#) objects.*
- class [lo_pager](#)
 - [lo_pager](#) interface.*
- class [Pager](#)
 - [Pager](#) interface including the [lo_pager](#) interface.*
- class [Platform_control](#)
 - [L4](#) C++ interface for controlling platform-wide properties, see [Platform Control C API](#) for the C interface.*
- class [Rcv_endpoint](#)
 - Interface for kernel objects that allow to receive IPC from them.*
- class [Scheduler](#)
 - C++ interface of the [Scheduler](#) kernel object, see [Scheduler](#) for the C interface.*
- struct [Semaphore](#)
 - C++ Kernel-provided semaphore interface, see [Kernel-provided semaphore](#) for the C interface.*
- class [Smart_cap](#)
 - Smart capability class.*
- class [Task](#)
 - C++ interface of the [Task](#) kernel object, see [Task](#) for the C interface.*
- class [Thread](#)
 - C++ [L4](#) kernel thread interface, see [Thread](#) for the C interface.*
- class [Thread_group](#)
 - C++ [L4](#) kernel thread group interface, see [Thread groups](#) for the C interface.*
- class [Vcon](#)
 - C++ [L4 Vcon](#) interface, see [Virtual Console](#) for the C interface.*
- class [Vm](#)
 - Virtual machine host address space.*
- class [Poll_timeout_kipclock](#)
 - A polling timeout based on the [L4Re](#) clock.*
- class [Alloc_list](#)
 - A simple list-based allocator.*
- class [IOModifier](#)
 - Modifier class for the IO stream.*
- class [Exception_tracer](#)
 - Back-trace support for exceptions.*
- class [Base_exception](#)

- Base class for all exceptions, thrown by the [L4Re](#) framework.*

 - class [Runtime_error](#)
Exception for an abstract runtime error.
 - class [Out_of_memory](#)
Exception signalling insufficient memory.
 - class [Element_already_exists](#)
Exception for duplicate element insertions.
 - class [Unknown_error](#)
Exception for an unknown condition.
 - class [Element_not_found](#)
Exception for a failed lookup (element not found).
 - class [Invalid_capability](#)
Indicates that an invalid object was invoked.
 - class [Com_error](#)
Error conditions during IPC.
 - class [Bounds_error](#)
Access out of bounds.
 - class [Server_object](#)
Abstract server object to be used with [L4::Server](#) and [L4::Basic_registry](#).
 - struct [Server_object_t](#)
Base class (template) for server implementing server objects.
 - struct [Server_object_x](#)
Helper class to implement p_dispatch based server objects.
 - struct [Irq_handler_object](#)
Server object base class for handling IRQ messages.
 - class [Lock_guard](#)
Basic lock guard implementation that prevents forgotten unlocks on exit paths from a method or a block of code.
 - class [String](#)
A null-terminated string container class.
 - class [Poll_timeout_counter](#)
Evaluate an expression for a maximum number of times.
 - class [Uart_apb](#)
Driver for the Advanced Peripheral Bus (APB) UART from the Cortex-M System Design Kit (CMSDK).
 - class [Uart](#)
Uart driver abstraction.

Typedefs

- typedef int **Opcode**
Data type for RPC opcodes.

Enumerations

- enum : [l4_proto_t](#) { [PROTO_ANY](#) = 0 , [PROTO_EMPTY](#) = -19 }

Functions

- `template<typename T>`
`Type_info` const * `kobject_typeid` () noexcept
Get the `L4::Type_info` for the `L4Re` interface given in `T`.
- `template<typename T, typename F>`
`Cap< T >` `cap_dynamic_cast` (`Cap< F >` const &c) noexcept
`dynamic_cast` for capabilities.
- `template<typename T, typename F>`
`Cap< T >` `cap_cast` (`Cap< F >` const &c) noexcept
`static_cast` for capabilities.
- `template<typename T, typename F>`
`Cap< T >` `cap_reinterpret_cast` (`Cap< F >` const &c) noexcept
`reinterpret_cast` for capabilities.
- `template<typename T>`
`constexpr T` `trunc_order` (T val, unsigned char order)
Round a value down so the given number of lsb is zero.
- `template<typename T>`
`constexpr T` `round_order` (T val, unsigned char order)
Round a value up so the given number of lsb is zero.
- `template<typename T, typename F, typename SMART>`
`Smart_cap< T, SMART >` `cap_cast` (`Smart_cap< F, SMART >` const &c) noexcept
`static_cast` for (smart) capabilities.
- `template<typename T, typename F, typename SMART>`
`Smart_cap< T, SMART >` `cap_reinterpret_cast` (`Smart_cap< F, SMART >` const &c) noexcept
`reinterpret_cast` for (smart) capabilities.
- `void` `throw_ipc_exception` (`L4::Cap< void >` const &o, `l4_msgtag_t` const &err, `l4_utcb_t` *utcb)
Throw an `L4` IPC error as exception.
- `void` `throw_ipc_exception` (void const *o, `l4_msgtag_t` const &err, `l4_utcb_t` *utcb)
Throw an `L4` IPC error as exception.

Variables

- `IOModifier` const `hex`
Modifies the stream to print numbers as hexadecimal values.
- `IOModifier` const `dec`
Modifies the stream to print numbers as decimal values.
- `BasicOStream` `cout`
Standard output stream.
- `BasicOStream` `cerr`
Standard error stream.

14.5.1 Detailed Description

`L4` low-level kernel interface.

14.5.2 Enumeration Type Documentation

14.5.2.1 anonymous enum

anonymous enum : [l4_proto_t](#)

Enumerator

| | |
|-------------|--|
| PROTO_ANY | Default protocol used by Kobject_t and Kobject_x . |
| PROTO_EMPTY | Empty protocol for empty APIs. |

Definition at line 44 of file [__typeinfo.h](#).

14.5.3 Function Documentation

14.5.3.1 cap_cast() [1/2]

```
template<typename T, typename F>
Cap< T > L4::cap_cast (
    Cap< F > const & c) [inline], [noexcept]
```

static_cast for capabilities.

Template Parameters

| | |
|----------|---|
| <i>T</i> | The target type of the capability |
| <i>F</i> | The source type (and is usually implicitly set) |

Parameters

| | |
|----------|--|
| <i>c</i> | The source capability that shall be casted |
|----------|--|

Returns

A capability typed to the interface T.

The use of this cast operator is similar to the `static_cast<>()` for C++ pointers. It does the same type checking and adjustments like C++ does on pointers.

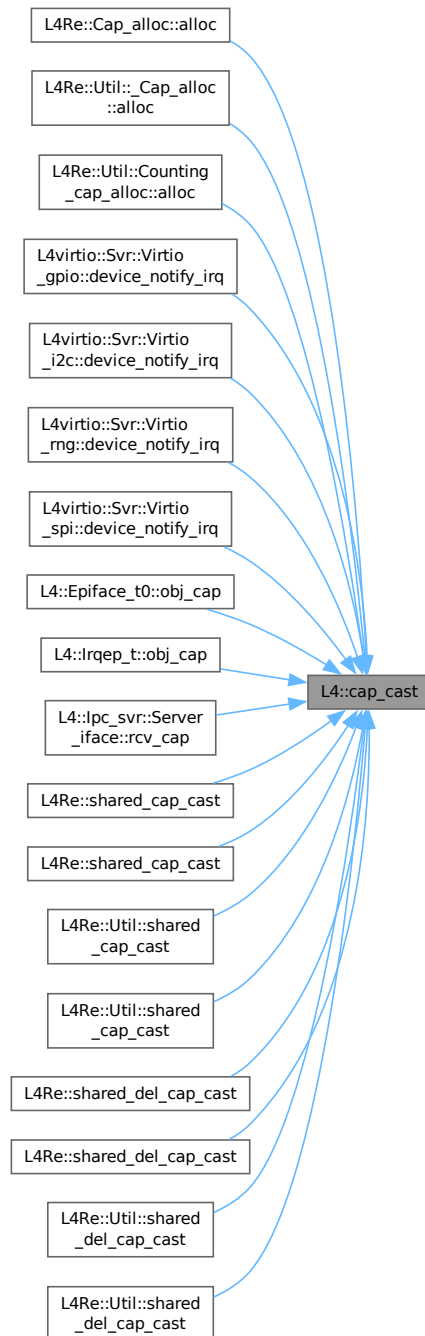
Example code:

```
L4::Cap<L4::Kobject> obj = ... ;
L4::Cap<L4::Icu> icu = L4::cap_cast<L4::Icu>(obj);
```

Definition at line 416 of file [capability.h](#).

Referenced by [L4Re::Cap_alloc::alloc\(\)](#), [L4Re::Util::_Cap_alloc::alloc\(\)](#), [L4Re::Util::Counting_cap_alloc< COUNTERTYPE, Dbg >::a](#), [L4virtio::Svr::Virtio_gpio< Request_handler, Epiface >::device_notify_irq\(\)](#), [L4virtio::Svr::Virtio_i2c< Request_handler, Epiface >::d](#), [L4virtio::Svr::Virtio_rng< Rnd_state, Epiface >::device_notify_irq\(\)](#), [L4virtio::Svr::Virtio_spi< Spi_request_handler, Epiface >::device](#), [L4::Epiface_t0< RPC_IFACE, BASE >::obj_cap\(\)](#), [L4::Irqep_t< Derived, BASE, bool >::obj_cap\(\)](#), [L4::lpc_svr::Server_iface::rcv_ca](#), [L4Re::shared_cap_cast\(\)](#), [L4Re::shared_cap_cast\(\)](#), [L4Re::Util::shared_cap_cast\(\)](#), [L4Re::Util::shared_cap_cast\(\)](#), [L4Re::shared_del_cap_cast\(\)](#), [L4Re::shared_del_cap_cast\(\)](#), [L4Re::Util::shared_del_cap_cast\(\)](#), and [L4Re::Util::shared_del_cap_ca](#).

Here is the caller graph for this function:



14.5.3.2 `cap_cast()` [2/2]

```
template<typename T, typename F, typename SMART>
Smart_cap< T, SMART > L4::cap_cast (
    Smart_cap< F, SMART > const & c) [inline], [noexcept]
```

`static_cast` for (smart) capabilities.

Template Parameters

| | |
|--------------|--|
| <i>T</i> | Type to cast the capability to. |
| <i>F</i> | (implicit) Type of the passed capability. |
| <i>SMART</i> | (implicit) Class implementing the Smart_cap interface. |

Parameters

| | |
|----------|--------------------------|
| <i>c</i> | Capability to be casted. |
|----------|--------------------------|

Returns

A smart capability with new type *T*.

Definition at line [192](#) of file [smart_capability](#).

14.5.3.3 `cap_dynamic_cast()`

```
template<typename T, typename F>
Cap< T > L4::cap_dynamic_cast (
    Cap< F > const & c) [inline], [noexcept]
```

`dynamic_cast` for capabilities.

Template Parameters

| | |
|----------|--|
| <i>T</i> | The target type of the capability. |
| <i>F</i> | The source type (is usually implicitly set). |

Parameters

| | |
|----------|---|
| <i>c</i> | The source capability that shall be casted. |
|----------|---|

Return values

| | |
|------------------------------|---|
| Cap<T> | Capability of target interface <i>T</i> . |
|------------------------------|---|

| | |
|-----------------------------|---|
| <code>L4_INVALID_CAP</code> | <code>c</code> does not support the target interface <code>T</code> or the <code>L4::Meta</code> interface. |
|-----------------------------|---|

The use of this cast operator is similar to the `dynamic_cast<>()` for C++ pointers. It also induces overhead, because it uses the meta interface (`L4::Meta`) to do runtime type checking.

Example code:

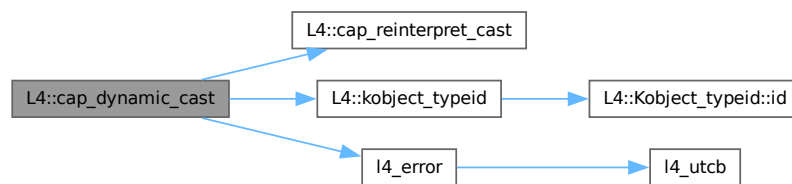
```
L4::Cap<L4::Kobject> obj = ... ;
L4::Cap<L4::Icu> icu = L4::cap_dynamic_cast<L4::Icu>(obj);
```

Definition at line 116 of file [capability](#).

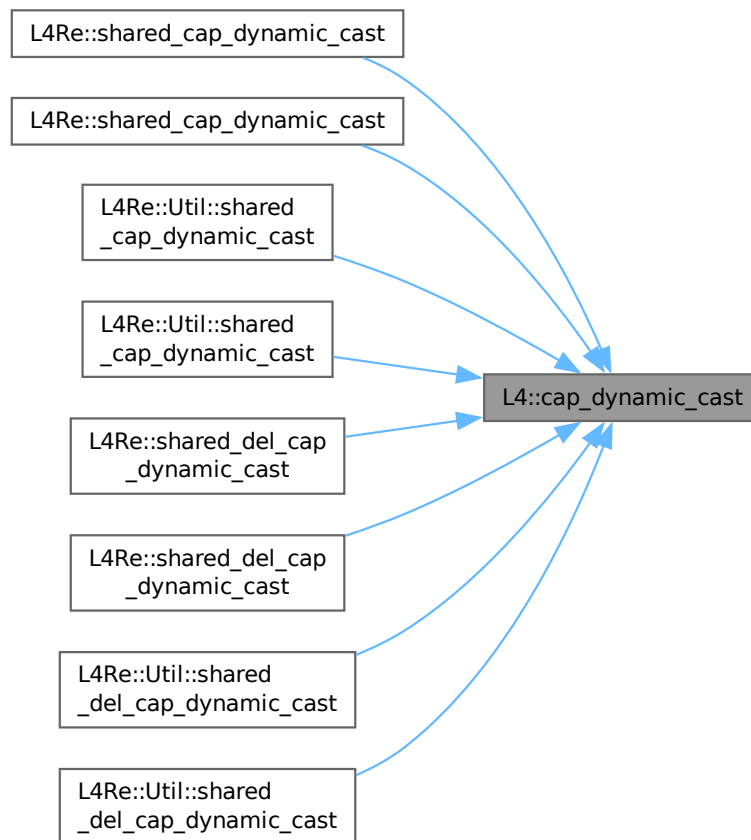
References [cap_reinterpret_cast\(\)](#), [L4::Cap_base::Invalid](#), [kobject_typeid\(\)](#), and [l4_error\(\)](#).

Referenced by [L4Re::shared_cap_dynamic_cast\(\)](#), [L4Re::shared_cap_dynamic_cast\(\)](#), [L4Re::Util::shared_cap_dynamic_cast\(\)](#), [L4Re::Util::shared_cap_dynamic_cast\(\)](#), [L4Re::shared_del_cap_dynamic_cast\(\)](#), [L4Re::shared_del_cap_dynamic_cast\(\)](#), [L4Re::Util::shared_del_cap_dynamic_cast\(\)](#), and [L4Re::Util::shared_del_cap_dynamic_cast\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



14.5.3.4 `cap_reinterpret_cast()` [1/2]

```

template<typename T, typename F>
Cap< T > L4::cap_reinterpret_cast (
    Cap< F > const & c) [inline], [noexcept]

```

`reinterpret_cast` for capabilities.

Template Parameters

| | |
|----------|---|
| <i>T</i> | The target type of the capability |
| <i>F</i> | The source type (and is usually implicitly set) |

Parameters

| | |
|----------|--|
| <i>c</i> | The source capability that shall be casted |
|----------|--|

Returns

A capability typed to the interface T.

The use of this cast operator is similar to the `reinterpret_cast<>()` for C++ pointers. It does not do any type checking or type adjustment.

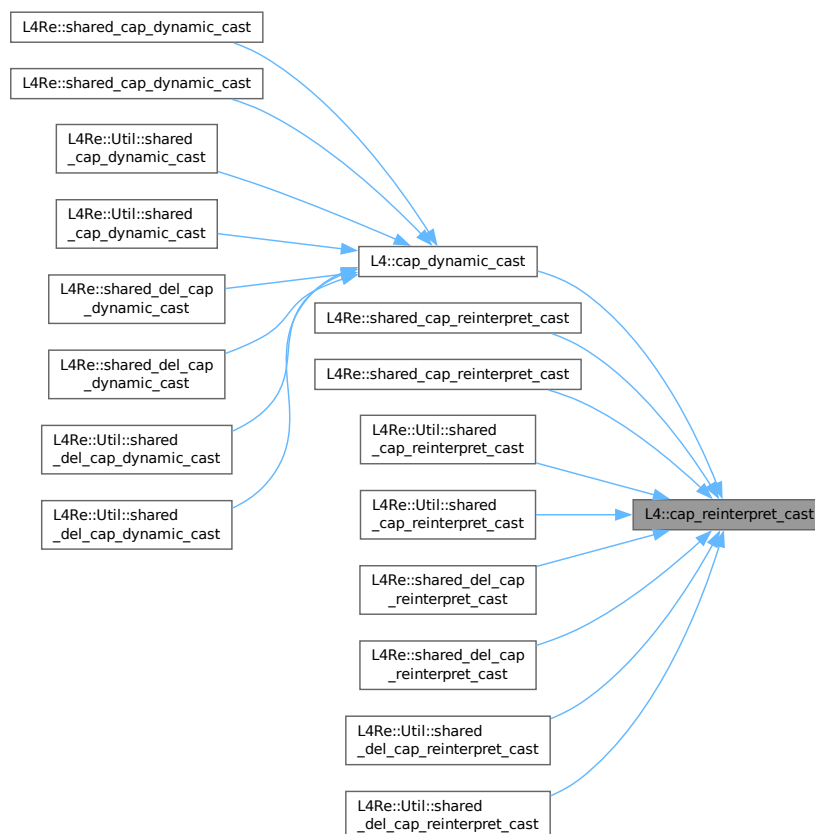
Example code:

```
L4::Cap<L4::Kobject> obj = ... ;
L4::Cap<L4::Icu> icu = L4::cap_reinterpret_cast<L4::Icu>(obj);
```

Definition at line 447 of file [capability.h](#).

Referenced by [cap_dynamic_cast\(\)](#), [L4Re::shared_cap_reinterpret_cast\(\)](#), [L4Re::shared_cap_reinterpret_cast\(\)](#), [L4Re::Util::shared_cap_reinterpret_cast\(\)](#), [L4Re::Util::shared_cap_reinterpret_cast\(\)](#), [L4Re::shared_del_cap_reinterpret_cast\(\)](#), [L4Re::shared_del_cap_reinterpret_cast\(\)](#), [L4Re::Util::shared_del_cap_reinterpret_cast\(\)](#), and [L4Re::Util::shared_del_cap_reinterpret_cast\(\)](#).

Here is the caller graph for this function:



14.5.3.5 `cap_reinterpret_cast()` [2/2]

```
template<typename T, typename F, typename SMART>
Smart_cap< T, SMART > L4::cap_reinterpret_cast (
    Smart_cap< F, SMART > const & c) [inline], [noexcept]
```

`reinterpret_cast` for (smart) capabilities.

Template Parameters

| | |
|--------------|--|
| <i>T</i> | Type to cast the capability to. |
| <i>F</i> | (implicit) Type of the passed capability. |
| <i>SMART</i> | (implicit) Class implementing the Smart_cap interface. |

Parameters

| | |
|----------|--------------------------|
| <i>c</i> | Capability to be casted. |
|----------|--------------------------|

Returns

A smart capability with new type *T*.

Definition at line 211 of file [smart_capability](#).

14.5.3.6 `round_order()`

```
template<typename T>
T L4::round_order (
    T val,
    unsigned char order) [constexpr]
```

Round a value up so the given number of lsb is zero.

Template Parameters

| | |
|----------|--|
| <i>T</i> | The type of the value (shall be some integral type). |
|----------|--|

Parameters

| | |
|--------------|--|
| <i>val</i> | The value to round up to the next multiple of 2^{order} . |
| <i>order</i> | order (2^{order}) to round up to. |

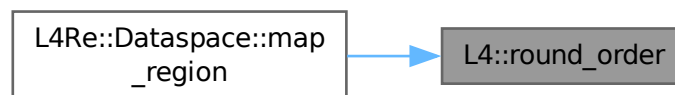
Returns

val rounded up to the next 2^{order} .

Definition at line 32 of file [consts](#).

Referenced by [L4Re::Dataspace::map_region\(\)](#).

Here is the caller graph for this function:

**14.5.3.7 trunc_order()**

```

template<typename T>
T L4::trunc_order (
    T val,
    unsigned char order) [constexpr]
  
```

Round a value down so the given number of lsb is zero.

Template Parameters

| | |
|----------|--|
| <i>T</i> | The type of the value (shall be some integral type). |
|----------|--|

Parameters

| | |
|--------------|---|
| <i>val</i> | The value where the given lsb shall be masked. |
| <i>order</i> | the number of least significant bits (lsb) to mask. |

Returns

val with order lsb masked to zero.

Definition at line 18 of file [consts](#).

Referenced by [L4Re::Dataspace::map_region\(\)](#).

Here is the caller graph for this function:



14.6 L4::lpc Namespace Reference

IPC related functionality.

Namespaces

- namespace [Msg](#)
IPC Message related functionality.

Data Structures

- struct [Array_ref](#)
Array reference data type for arrays located in the message.
- struct [Array](#)
Array data type for dynamically sized arrays in RPCs.
- struct [Array_in_buf](#)
Server-side copy in buffer for [Array](#).
- struct [Call](#)
RPC attribute for a standard RPC call.
- struct [Call_zero_send_timeout](#)
RPC attribute for an RPC call, with zero send timeout.
- struct [Call_t](#)
RPC attribute for an RPC call with required rights.
- struct [Send_only](#)
RPC attribute for a send-only RPC.
- struct [Ret_array](#)
Dynamically sized output array of type T.
- struct [Out](#)
Mark an argument as a output value in an RPC signature.
- struct [In_out](#)
Mark an argument as in-out argument.
- struct [As_value](#)
Pass the argument as plain data value.
- struct [Opt](#)
Attribute for defining an optional RPC argument.
- class [Small_buf](#)

- A receive item for receiving a single object capability.*

 - class [Gen_fpage](#)

Generic RPC base for typed message items.
 - class [Snd_fpage](#)

Send item or return item.
 - class [Rcv_fpage](#)

Non-small receive item.
 - class [Cap](#)

Capability type for RPC interfaces (see [L4 : :Cap<T>](#)).
 - class [Varg](#)

Variably sized RPC argument.
 - class [Varg_list](#)

Self-contained list of variable-sized RPC parameters.
 - class [Varg_list_ref](#)

List of variable-sized RPC parameters as received by the server.
 - class [Str_cp_in](#)

Abstraction for extracting a zero-terminated string from an [lpc::Istream](#).
 - class [Msg_ptr](#)

Pointer to an element of type T in an [lpc::Istream](#).
 - class [Istream](#)

Input stream for IPC unmarshalling.
 - class [Ostream](#)

Output stream for IPC marshalling.
 - class [Iostream](#)

Input/Output stream for IPC [un]marshalling.

Typedefs

- using [Array_len_default](#) = unsigned short

Default type for passing length of an array.

Functions

- template<typename T>
[Cap< T > make_cap](#) ([L4::Cap< T > cap](#), unsigned rights) noexcept

Make an [L4::lpc::Cap<T>](#) for the given capability and rights.
- template<typename T>
[Cap< T > make_cap_rw](#) ([L4::Cap< T > cap](#)) noexcept

Make an [L4::lpc::Cap<T>](#) for the given capability with [L4_CAP_FPAGE_RW](#) rights.
- template<typename T>
[Cap< T > make_cap_rws](#) ([L4::Cap< T > cap](#)) noexcept

Make an [L4::lpc::Cap<T>](#) for the given capability with [L4_CAP_FPAGE_RWS](#) rights.
- template<typename T>
[Cap< T > make_cap_full](#) ([L4::Cap< T > cap](#)) noexcept

Make an [L4::IPC::Cap<T>](#) for the given capability with full fpage and object-specific rights.
- template<typename T>
[Cap< T > make_cap_grant](#) ([L4::Cap< T > cap](#)) noexcept

Make an [L4::IPC::Cap<T>](#) for the given capability that shall be granted with full rights.
- template<typename T>
[Internal::Buf_cp_out< T > buf_cp_out](#) (T const *v, unsigned long size)

- Insert an array into an [lpc::Ostream](#).*

 - `template<typename T>`
`Internal::Buf_cp_in< T > buf_cp_in (T *v, unsigned long &size)`
Extract an array from an [lpc::Istream](#).
 - `template<typename T>`
`Str_cp_in< T > str_cp_in (T *v, unsigned long &size)`
Create a [Str_cp_in](#) for the given values.
 - `template<typename T>`
`Msg_ptr< T > msg_ptr (T *&p)`
Create an [Msg_ptr](#) to adjust the given pointer.
 - `template<typename T>`
`Internal::Buf_in< T > buf_in (T *&v, unsigned long &size)`
Return a pointer to stream array data.
 - `template<typename T>`
`T read (Istream &s)`
Read a value out of a stream.

14.6.1 Detailed Description

IPC related functionality.

14.6.2 Function Documentation

14.6.2.1 [buf_cp_in\(\)](#)

```
template<typename T>
Internal::Buf_cp_in< T > L4::Ipc::buf_cp_in (
    T * v,
    unsigned long & size)
```

Extract an array from an [lpc::Istream](#).

Parameters

| | | |
|----------------|-------------|---|
| | <i>v</i> | Pointer to the array that shall receive the values from the lpc::Istream . |
| <i>in, out</i> | <i>size</i> | Input: the number of elements the array can take at most Output: the number of elements found in the stream. |

[buf_cp_in\(\)](#) can be used to extract an array from an [lpc::Istream](#). This is the counterpart [buf_cp_out\(\)](#). The data from the received message is thereby copied to the given buffer and size is set to the number of elements found in the stream. To avoid the copy operation [buf_in\(\)](#) may be used instead.

See also

[buf_in\(\)](#) and [buf_cp_out\(\)](#).

Definition at line 159 of file [ipc_stream](#).

14.6.2.2 buf_cp_out()

```
template<typename T>
Internal::Buf_cp_out< T > L4::Ipc::buf_cp_out (
    T const * v,
    unsigned long size)
```

Insert an array into an [lpc::Ostream](#).

Parameters

| | |
|-------------|--|
| <i>v</i> | Pointer to the array that shall be inserted into an lpc::Ostream . |
| <i>size</i> | Number of elements in the array. |

This function inserts an array (e.g. a string) into an [lpc::Ostream](#). The data is copied to the stream. On insertion into the [lpc::Ostream](#) exactly the given number of elements of type T are copied to the message buffer, this means the source buffer is no longer referenced after insertion into the stream.

See also

The counterpart is either [buf_cp_in\(\)](#) or [buf_in\(\)](#).

Definition at line 100 of file [ipc_stream](#).

14.6.2.3 buf_in()

```
template<typename T>
Internal::Buf_in< T > L4::Ipc::buf_in (
    T *& v,
    unsigned long & size)
```

Return a pointer to stream array data.

Parameters

| | | |
|-----|-------------|--|
| out | <i>v</i> | Pointer to the array within the lpc::Istream . |
| out | <i>size</i> | The number of elements found in the stream. |

This routine provides a possibility to extract an array from an [lpc::Istream](#), without extra copy overhead. In contrast to [buf_cp_in\(\)](#) the data is not copied to a buffer, but a pointer to the array is returned. The user must make sure the UTCB is not used for other purposes while the returned pointer is still in use.

The mechanism is comparable to that of [Msg_ptr](#), however it handles arrays inserted with [buf_cp_out\(\)](#).

See also

[buf_cp_in\(\)](#) and [buf_cp_out\(\)](#).

Definition at line 310 of file [ipc_stream](#).

14.6.2.4 make_cap()

```
template<typename T>
Cap< T > L4::Ipc::make_cap (
    L4::Cap< T > cap,
    unsigned rights) [noexcept]
```

Make an [L4::Ipc::Cap<T>](#) for the given capability and rights.

Template Parameters

| | |
|----------|---|
| <i>T</i> | (IMPLICIT) type of the referenced interface |
|----------|---|

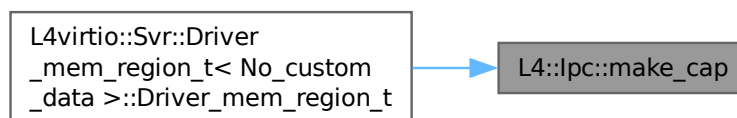
Parameters

| | |
|---------------|--|
| <i>cap</i> | source capability (L4::Cap<T>) |
| <i>rights</i> | rights mask that shall be applied on transfer. |

Definition at line 819 of file [ipc_types](#).

Referenced by [L4virtio::Svr::Driver_mem_region_t< No_custom_data >::Driver_mem_region_t\(\)](#).

Here is the caller graph for this function:



14.6.2.5 make_cap_full()

```
template<typename T>
Cap< T > L4::Ipc::make_cap_full (
    L4::Cap< T > cap) [noexcept]
```

Make an [L4::Ipc::Cap<T>](#) for the given capability with full fpage and object-specific rights.

Template Parameters

| | |
|----------|---|
| <i>T</i> | (implicit) type of the referenced interface |
|----------|---|

Parameters

| | |
|------------|--|
| <i>cap</i> | source capability (L4::Cap<T>) |
|------------|--|

See also

[L4_cap_fpage_rights](#)
[Attributes and additional permissions for object send items](#)

Note

Full rights (including object-specific rights) are required when mapping an IPC gate where the receiver should become the server, i.e. where the receiver wants to call [L4::ipc_gate::bind_thread\(\)](#) or [L4::ipc_gate::bind_snd_destination\(\)](#).

Definition at line 858 of file [ipc_types](#).

References [L4_CAP_FPAGE_RWSD](#), and [L4_FPAGE_C_OBJ_RIGHTS](#).

14.6.2.6 make_cap_grant()

```
template<typename T>
Cap< T > L4::Ipc::make_cap_grant (
    L4::Cap< T > cap) [noexcept]
```

Make an [L4::IPC::Cap<T>](#) for the given capability that shall be granted with full rights.

Template Parameters

| | |
|----------|---|
| <i>T</i> | (implicit) type of the referenced interface |
|----------|---|

Parameters

| | |
|------------|--|
| <i>cap</i> | source capability (L4::Cap<T>) |
|------------|--|

See also

[L4_MAP_ITEM_GRANT](#)

Definition at line 871 of file [ipc_types](#).

References [L4::ipc::Cap< T >::Grant](#), [L4_CAP_FPAGE_RWSD](#), and [L4_FPAGE_C_OBJ_RIGHTS](#).

14.6.2.7 make_cap_rw()

```
template<typename T>
Cap< T > L4::Ipc::make_cap_rw (
    L4::Cap< T > cap) [noexcept]
```

Make an [L4::Ipc::Cap<T>](#) for the given capability with [L4_CAP_FPAGE_RW](#) rights.

Template Parameters

| | |
|----------|---|
| <i>T</i> | (IMPLICIT) type of the referenced interface |
|----------|---|

Parameters

| | |
|------------|--|
| <i>cap</i> | source capability (L4::Cap<T>) |
|------------|--|

Examples

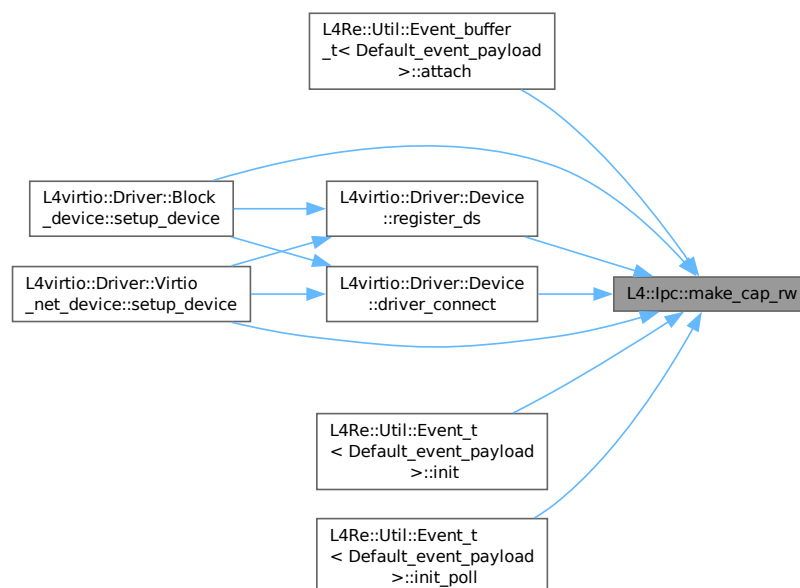
[examples/libs/l4re/c++/mem_alloc/ma+rm.cc](#), [examples/libs/l4re/c++/shared_ds/ds_clnt.cc](#), and [examples/libs/l4re/c++/shared_](#)

Definition at line 829 of file [ipc_types](#).

References [L4_CAP_FPAGE_RW](#).

Referenced by [L4Re::Util::Event_buffer_t< Default_event_payload >::attach\(\)](#), [L4virtio::Driver::Device::driver_connect\(\)](#), [L4Re::Util::Event_t< Default_event_payload >::init\(\)](#), [L4Re::Util::Event_t< Default_event_payload >::init_poll\(\)](#), [L4virtio::Driver::Device::register_ds\(\)](#), [L4virtio::Driver::Block_device::setup_device\(\)](#), and [L4virtio::Driver::Virtio_net_device::setup_de](#)

Here is the caller graph for this function:



14.6.2.8 make_cap_rws()

```
template<typename T>
Cap< T > L4::Ipc::make_cap_rws (
    L4::Cap< T > cap) [noexcept]
```

Make an [L4::ipc::Cap<T>](#) for the given capability with [L4_CAP_FPAGE_RWS](#) rights.

Template Parameters

| | |
|----------|---|
| <i>T</i> | (IMPLICIT) type of the referenced interface |
|----------|---|

Parameters

| | |
|------------|--|
| <i>cap</i> | source capability (L4::Cap<T>) |
|------------|--|

Definition at line 839 of file [ipc_types](#).

References [L4_CAP_FPAGE_RWS](#).

14.6.2.9 msg_ptr()

```
template<typename T>
Msg_ptr< T > L4::Ipc::msg_ptr (
    T *& p)
```

Create an [Msg_ptr](#) to adjust the given pointer.

This function makes it more convenient to extract pointers to data in the message buffer itself from an [ipc::Istream](#). This may be used to avoid copy out of large data structures. (See [Msg_ptr](#).)

Definition at line 252 of file [ipc_stream](#).

14.6.2.10 read()

```
template<typename T>
T L4::Ipc::read (
    Istream & s) [inline]
```

Read a value out of a stream.

Parameters

| | |
|----------|------------------------------|
| <i>s</i> | An Istream . |
|----------|------------------------------|

Returns

The value of type *T*.

The stream position is progressed accordingly.

Definition at line 1289 of file [ipc_stream](#).

14.6.2.11 str_cp_in()

```
template<typename T>
Str_cp_in< T > L4::Ipc::str_cp_in (
    T * v,
    unsigned long & size)
```

Create a [Str_cp_in](#) for the given values.

Parameters

| | | |
|----------------|-------------|---|
| | <i>v</i> | Pointer to the array that shall receive the values from the lpc::lstream . |
| <i>in, out</i> | <i>size</i> | Input: the number of elements the array can take at most Output: the number of elements found in the stream. |

This function makes it more convenient to extract arrays from an [lpc::lstream](#) (

See also

[Str_cp_in](#).)

Definition at line 213 of file [ipc_stream](#).

14.7 L4::lpc::Msg Namespace Reference

IPC Message related functionality.

Data Structures

- struct [Elem< Array< A, LEN > >](#)
Array as input arguments.
- struct [Elem< Array< A, LEN > & >](#)
Array as output argument.
- struct [Elem< Array_ref< A, LEN > & >](#)
Array_ref as output argument.
- struct [Dir_in](#)
Marker type for input values.
- struct [Dir_out](#)
Marker type for output values.
- struct [Cls_data](#)
Marker type for data values.
- struct [Cls_item](#)
Marker type for item values.
- struct [Cls_buffer](#)
Marker type for receive buffer values.
- struct [Do_in_data](#)
Marker for Input data.
- struct [Do_out_data](#)

- Marker for Output data.*
- struct [Do_in_items](#)
 - Marker for Input items.*
- struct [Do_out_items](#)
 - Marker for Output items.*
- struct [Do_rcv_buffers](#)
 - Marker for receive buffers.*
- struct [CInt_val_ops](#)
 - Defines client-side handling of 'MTYPE' as RPC argument.*
- struct [Svr_val_ops](#)
 - Defines server-side handling for MTYPE server arguments.*
- struct [Is_valid_rpc_type](#)
 - Type trait defining a valid RPC parameter type.*
- struct [Svr_arg_pack](#)
 - Server-side RPC arguments data structure used to provide arguments to the server-side implementation of an RPC function.*
- struct [True](#)
 - True meta value.*
- struct [False](#)
 - False meta value.*

Enumerations

- enum {
[Word_bytes](#) = sizeof(l4_umword_t) , [Item_words](#) = 2 , [Item_bytes](#) = Word_bytes * Item_words , [Mr_words](#) = L4_UTCB_GENERIC_DATA_SIZE ,
[Mr_bytes](#) = Word_bytes * Mr_words , [Br_bytes](#) = Word_bytes * L4_UTCB_GENERIC_BUFFERS_SIZE }

Functions

- constexpr unsigned long [align_to](#) (unsigned long bytes, unsigned long align) noexcept
Pad bytes to the given alignment align (in bytes).
- template<typename T>
 constexpr unsigned long [align_to](#) (unsigned long bytes) noexcept
Pad bytes to the alignment of the type T.
- template<typename T>
 constexpr bool [check_size](#) (unsigned offset, unsigned limit) noexcept
Check if there is enough space for T from offset to limit.
- template<typename T, typename CTYPE>
 bool [check_size](#) (unsigned offset, unsigned limit, CTYPE cnt) noexcept
Check if there is enough space for an array of T from offset to limit.
- template<typename T>
 int [msg_add](#) (char *msg, unsigned offs, unsigned limit, T v) noexcept
Add some data to a message at offs.
- template<typename T>
 int [msg_get](#) (char *msg, unsigned offs, unsigned limit, T &v) noexcept
Get some data from a message at offs.

14.7.1 Detailed Description

IPC Message related functionality.

14.7.2 Enumeration Type Documentation

14.7.2.1 anonymous enum

anonymous enum

Enumerator

| | |
|------------|---|
| Word_bytes | number of bytes for one message word |
| Item_words | number of message words for one message item |
| Item_bytes | number of bytes for one message item |
| Mr_words | number of message words available in the UTCB |
| Mr_bytes | number of bytes available in the UTCB message registers |
| Br_bytes | number of bytes available in the UTCB buffer registers |

Definition at line 85 of file [ipc_basics](#).

14.7.3 Function Documentation

14.7.3.1 align_to() [1/2]

```
template<typename T>
unsigned long L4::Ipc::Msg::align_to (
    unsigned long bytes) [constexpr, [noexcept]
```

Pad *bytes* to the alignment of the type *T*.

Template Parameters

| | |
|----------|--------------------------------------|
| <i>T</i> | The data type used for the alignment |
|----------|--------------------------------------|

Parameters

| | |
|--------------|---------------------------------|
| <i>bytes</i> | The value to add the padding to |
|--------------|---------------------------------|

Returns

bytes padded to achieve the alignment of *T*.

Definition at line 40 of file [ipc_basics](#).

References [align_to\(\)](#).

Here is the call graph for this function:



14.7.3.2 align_to() [2/2]

```
unsigned long L4::Ipc::Msg::align_to (  
    unsigned long bytes,  
    unsigned long align) [constexpr], [noexcept]
```

Pad bytes to the given alignment *align* (in bytes).

Parameters

| | |
|--------------|------------------------------|
| <i>bytes</i> | The input value in bytes |
| <i>align</i> | The alignment value in bytes |

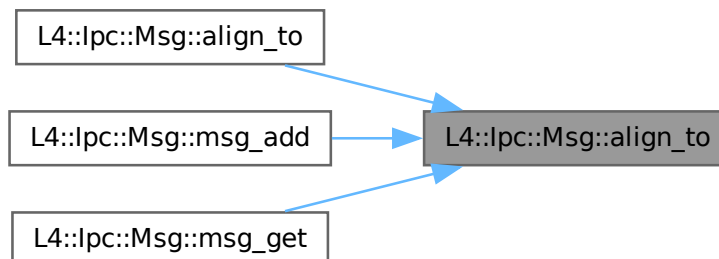
Returns

the result after padding *bytes* to *align*.

Definition at line 30 of file [ipc_basics](#).

Referenced by [align_to\(\)](#), [msg_add\(\)](#), and [msg_get\(\)](#).

Here is the caller graph for this function:



14.7.3.3 `check_size()` [1/2]

```
template<typename T>
bool L4::Ipc::Msg::check_size (
    unsigned offset,
    unsigned limit) [constexpr], [noexcept]
```

Check if there is enough space for T from offset to limit.

Template Parameters

| | |
|----------|---|
| <i>T</i> | The data type that shall be fitted at <i>offset</i> |
|----------|---|

Parameters

| | |
|---------------|--|
| <i>offset</i> | The current offset in bytes (must already be padded if desired). |
| <i>limit</i> | The limit in bytes that must not be exceeded after adding the size of <i>T</i> . |

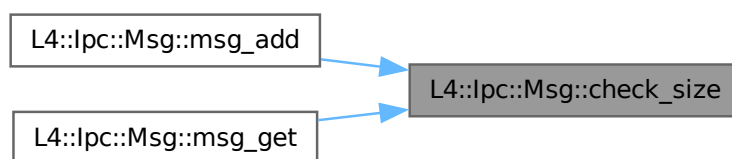
Returns

true if the limit will not be exceeded, false else.

Definition at line 53 of file [ipc_basics](#).

Referenced by [msg_add\(\)](#), and [msg_get\(\)](#).

Here is the caller graph for this function:



14.7.3.4 `check_size()` [2/2]

```
template<typename T, typename CTYPE>
bool L4::Ipc::Msg::check_size (
    unsigned offset,
    unsigned limit,
    CTYPE cnt) [inline], [noexcept]
```

Check if there is enough space for an array of T from offset to limit.

Template Parameters

| | |
|--------------|---|
| <i>T</i> | The data type that shall be fitted at <i>offset</i> |
| <i>CTYPE</i> | Type of the <i>cnt</i> parameter |

Parameters

| | |
|---------------|---|
| <i>offset</i> | The current offset in bytes (must already be padded if desired). |
| <i>limit</i> | The limit in bytes that must not be exceeded after adding <i>cnt</i> times the size of <i>T</i> . |
| <i>cnt</i> | The number of elements of type <i>T</i> that shall be put at <i>offset</i> . |

Returns

true if the limit will not be exceeded, false else.

Definition at line 71 of file [ipc_basics](#).

References [L4_UNLIKELY](#).

14.7.3.5 msg_add()

```
template<typename T>
int L4::IpC::Msg::msg_add (
    char * msg,
    unsigned offs,
    unsigned limit,
    T v) [inline], [noexcept]
```

Add some data to a message at offs.

Template Parameters

| | |
|----------|-----------------------------|
| <i>T</i> | The type of the data to add |
|----------|-----------------------------|

Parameters

| | |
|--------------|--|
| <i>msg</i> | pointer to the start of the message |
| <i>offs</i> | The current offset within the message, this shall be padded to the alignment of <i>T</i> if <i>v</i> is added. |
| <i>limit</i> | The size limit in bytes that offset must not exceed. |
| <i>v</i> | The value to add to the message |

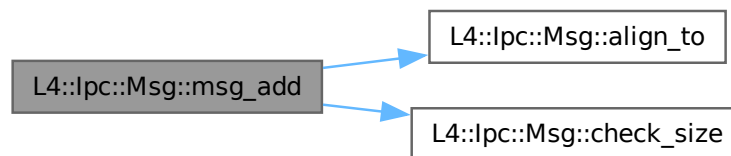
Returns

The new offset when successful, a negative value if the given limit will be exceeded.

Definition at line 114 of file [ipc_basics](#).

References [align_to\(\)](#), [check_size\(\)](#), [L4_MSGTOOLONG](#), and [L4_UNLIKELY](#).

Here is the call graph for this function:

**14.7.3.6 msg_get()**

```

template<typename T>
int L4::Ipc::Msg::msg_get (
    char * msg,
    unsigned offs,
    unsigned limit,
    T & v) [inline], [noexcept]
  
```

Get some data from a message at offs.

Template Parameters

| | |
|----------|------------------------------|
| <i>T</i> | The type of the data to read |
|----------|------------------------------|

Parameters

| | |
|--------------|---|
| <i>msg</i> | Pointer to the start of the message |
| <i>offs</i> | The current offset within the message, this shall be padded to the alignment of <i>T</i> if a <i>v</i> can be read. |
| <i>limit</i> | The size limit in bytes that offset must not exceed. |
| <i>v</i> | A reference to receive the value from the message |

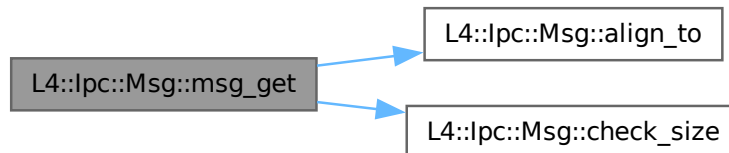
Returns

The new offset when successful, a negative value if the given limit will be exceeded.

Definition at line 135 of file [ipc_basics](#).

References [align_to\(\)](#), [check_size\(\)](#), [L4_EMSGTOOSHORT](#), and [L4_UNLIKELY](#).

Here is the call graph for this function:



14.8 L4::lpc_svr Namespace Reference

Helper classes for [L4::Server](#) instantiation.

Data Structures

- class [Server_iface](#)
Interface for server-loop related functions.
- struct [Ignore_errors](#)
Mix in for LOOP_HOOKS to ignore IPC errors.
- struct [Default_timeout](#)
Mix in for LOOP_HOOKS to use a 0 send and an infinite receive timeout.
- struct [Compound_reply](#)
Mix in for LOOP_HOOKS to always use compound reply and wait.
- struct [Default_setup_wait](#)
Mix in for LOOP_HOOKS for setup_wait no op.
- struct [Direct_dispatch](#)
Direct dispatch helper, for forwarding dispatch calls to a registry R.
- struct [Direct_dispatch< R * >](#)
Direct dispatch helper, for forwarding dispatch calls via a pointer to a registry R.
- struct [Exc_dispatch](#)
Dispatch helper wrapping try {} catch {} around the dispatch call.
- struct [Dbg_dispatch](#)
Dispatch helper that, in addition to what [Exc_dispatch](#) does, prints exception messages.
- class [Br_manager_no_buffers](#)
Empty implementation of [Server_iface](#).
- struct [Default_loop_hooks](#)
Default LOOP_HOOKS.
- class [Timeout](#)
Callback interface for [Timeout_queue](#).
- class [Timeout_queue](#)
[Timeout](#) queue to be used in l4re server loop.
- class [Timeout_queue_hooks](#)
Loop hooks mixin for integrating a timeout queue into the server loop.

Enumerations

- enum [Reply_mode](#) { [Reply_compound](#) , [Reply_separate](#) }
Reply mode for server loop.

14.8.1 Detailed Description

Helper classes for [L4::Server](#) instantiation.

14.9 L4::Typeid Namespace Reference

Definition of interface data-type helpers.

Data Structures

- struct [P_dispatch](#)
Use for protocol based dispatch stage.
- struct [Raw_ipc](#)
RPCs list for passing raw incoming IPC to the server object.
- struct [Rpc](#)
Standard list of RPCs of an interface.
- struct [Rpc_code](#)
List of RPCs of an interface using a special opcode type.
- struct [Rpc_nocode](#)
List of RPCs of an interface using a single operation without an opcode.
- struct [Rpc_sys](#)
List of RPCs typically used for kernel interfaces.

14.9.1 Detailed Description

Definition of interface data-type helpers.

Note

These type helpers are intended for internal use, if you look for standard C++ type traits use the `<type_traits>` header for the standard C++ library or use `<l4/cxx/type_traits>`.

14.10 L4::Types Namespace Reference

[L4](#) basic type helpers for C++.

Data Structures

- class [Flags](#)
Template for defining typical [Flags](#) bitmaps.
- struct [Int_for_size](#)
Metafunction to get an unsigned integral type for the given size.
- struct [Int_for_type](#)
*Metafunction to get an integral type of the same size as *T*.*
- struct [Flags_ops_t](#)
*Mixin class to define a set of friend bitwise operators on *DT*.*
- struct [Flags_t](#)
Template type to define a flags type with bitwise operations.
- struct [__Add_rvalue_reference_helper](#)
Helper template for [Add_rvalue_reference](#).
- struct [__Add_rvalue_reference_helper< T, Void< T && > >](#)
Helper template for [Add_rvalue_reference](#).
- struct [Add_rvalue_reference](#)
Create an rvalue reference of the given type.
- struct [Bool](#)
Boolean meta type.
- struct [False](#)
[False](#) meta value.
- struct [True](#)
[True](#) meta value.
- struct [Integral_constant](#)
Wrapper for a static constant of the given type.
- struct [Is_enum](#)
Check whether the given type is an enumeration type.
- struct [__Underlying_type_helper](#)
Helper template for [Underlying_type](#).
- struct [__Underlying_type_helper< T, false >](#)
Helper template for [Underlying_type](#).
- struct [Underlying_type](#)
Get an underlying type of an enumeration type.
- struct [Same](#)
Compare two data types for equality.
- struct [Same_template](#)
*Check if a type *T* is an instantiation of a given template.*

Typedefs

- `template<typename...>`
using [Void](#) = void
Map a sequence of any types to the void type.
- `template<typename T>`
using [Add_rvalue_reference_t](#) = typename [Add_rvalue_reference](#)<T>::type
Helper type for the [Add_rvalue_reference](#).
- `template<typename T>`
using [Underlying_type_t](#) = typename [Underlying_type](#)<T>::type
Helper type for [Underlying_type](#).
- `template<bool Condition, typename T = void>`
using [Enable_if_t](#) = typename [Enable_if](#)<Condition, T>::type
Helper type for [Enable_if](#).

Functions

- `template<typename T>
Add_rvalue_reference_t< T > declval () noexcept`
Template for writing typed expressions in unevaluated contexts.

14.10.1 Detailed Description

[L4](#) basic type helpers for C++.

14.10.2 Function Documentation

14.10.2.1 declval()

```
template<typename T>
Add_rvalue_reference_t< T > L4::Types::declval () [noexcept]
```

Template for writing typed expressions in unevaluated contexts.

In unevaluated contexts, the template converts a type (possibly an incomplete type) to an expression of that type.

Template Parameters

| | |
|----------|---|
| <i>T</i> | Type to be converted to an expression of that type. |
|----------|---|

14.11 L4Re Namespace Reference

[L4Re](#) C++ Interfaces.

Namespaces

- namespace [Vfs](#)
Virtual file system for interfaces in POSIX libc.
- namespace [Util](#)
Documentation of the [L4](#) Runtime Environment utility functionality in C++.

Data Structures

- class [Cap_alloc](#)
Capability allocator interface.
- class [Smart_cap_auto](#)
Helper for [Unique_cap](#) and [Unique_del_cap](#).
- class [Smart_count_cap](#)
Helper for [Ref_cap](#) and [Ref_del_cap](#).
- class [Console](#)
[Console](#) class.
- class [Dataspace](#)
Interface for memory-like objects.
- class [Debug_obj](#)
Debug interface.
- class [Dma_space](#)
Managed DMA Address Space.
- class [Env](#)
C++ interface of the initial environment that is provided to an [L4](#) task.
- class [Event](#)
[Event](#) class.
- struct [Default_event_payload](#)
Default event stream payload.
- class [Event_buffer_t](#)
[Event](#) buffer class.
- class [Inhibitor](#)
Set of inhibitor locks, which inhibit specific actions when held.
- class [Itas](#)
Interface to the ITAS.
- class [Log](#)
[Log](#) interface class.
- class [Mem_alloc](#)
Memory allocation interface.
- struct [Mmio_space](#)
Interface for memory-like address space accessible via IPC.
- class [Namespace](#)
Name-space interface.
- class [Parent](#)
[Parent](#) interface.
- struct [Random](#)
Low-bandwidth interface for random number generators.
- class [Rm](#)
[Region](#) map.

Typedefs

- template<typename T>
using [Shared_cap](#) = L4::Detail::Shared_cap_impl<T, [L4Re::Smart_count_cap](#)<[L4_FP_ALL_SPACES](#)>>
Shared capability that implements automatic free and unmap of the capability selector.
- template<typename T>
using [Shared_del_cap](#) = L4::Detail::Shared_cap_impl<T, [L4Re::Smart_count_cap](#)<[L4_FP_DELETE_OBJ](#)>>
Shared capability that implements automatic free and unmap+delete of the capability selector.

- `template<typename T>`
`using Unique_cap = L4::Detail::Unique_cap_impl<T, L4Re::Smart_cap_auto<L4_FP_ALL_SPACES>>`
Unique capability that implements automatic free and unmap of the capability selector.
- `template<typename T>`
`using Unique_del_cap = L4::Detail::Unique_cap_impl<T, L4Re::Smart_cap_auto<L4_FP_DELETE_OBJ>>`
Unique capability that implements automatic free and unmap+delete of the capability selector.

Functions

- `void throw_error` (long err, char const *extra="")
Generate C++ exception.
- `l4_ret_t chksys` (l4_ret_t err, char const *extra="", l4_ret_t ret=0)
Generate C++ exception on error.
- `l4_ret_t chksys` (l4_msgtag_t const &t, char const *extra="", l4_utcb_t *utcb=l4_utcb(), l4_ret_t ret=0)
Generate C++ exception on error.
- `l4_ret_t chksys` (l4_msgtag_t const &t, l4_utcb_t *utcb, char const *extra="")
Generate C++ exception on error.
- `template<typename T>`
`T chkcap` (T &&cap, char const *extra="", l4_ret_t err=L4_ENOMEM)
Check for valid capability or raise C++ exception.
- `l4_msgtag_t chkipc` (l4_msgtag_t tag, char const *extra="", l4_utcb_t *utcb=l4_utcb())
Test a message tag for IPC errors.
- `template<typename T>`
`Shared_cap< T > make_shared_cap` (L4Re::Cap_alloc *ca)
Allocate a capability slot and wrap it in a Shared_cap.
- `template<typename T, typename U>`
`Shared_cap< T > shared_cap_cast` (Shared_cap< U > const &from) noexcept
Create a new shared capability by an explicit cast from another shared capability.
- `template<typename T, typename U>`
`Shared_cap< T > shared_cap_cast` (Shared_cap< U > &&from) noexcept
Create a new shared capability by an explicit cast from another shared capability with move semantics.
- `template<typename T, typename U>`
`Shared_cap< T > shared_cap_reinterpret_cast` (Shared_cap< U > const &from) noexcept
Create a new shared capability by a reinterpret cast from another shared capability.
- `template<typename T, typename U>`
`Shared_cap< T > shared_cap_reinterpret_cast` (Shared_cap< U > &&from) noexcept
Create a new shared capability by a reinterpret cast from another shared capability with move semantics.
- `template<typename T, typename U>`
`Shared_cap< T > shared_cap_dynamic_cast` (Shared_cap< U > const &from) noexcept
Create a new shared capability by a dynamic cast from another shared capability.
- `template<typename T, typename U>`
`Shared_cap< T > shared_cap_dynamic_cast` (Shared_cap< U > &&from) noexcept
Create a new shared capability by an dynamic cast from another shared capability with move semantics.
- `template<typename T>`
`Shared_del_cap< T > make_shared_del_cap` (L4Re::Cap_alloc *ca)
Allocate a capability slot and wrap it in a Shared_del_cap.
- `template<typename T, typename U>`
`Shared_del_cap< T > shared_del_cap_cast` (Shared_del_cap< U > const &from) noexcept
Create a new shared capability by an explicit cast from another shared capability.
- `template<typename T, typename U>`
`Shared_del_cap< T > shared_del_cap_cast` (Shared_del_cap< U > &&from) noexcept
Create a new shared capability by an explicit cast from another shared capability with move semantics.

- `template<typename T, typename U>`
`Shared_del_cap< T > shared_del_cap_reinterpret_cast (Shared_del_cap< U > const &from) noexcept`
Create a new shared capability by a reinterpret cast from another shared capability.
- `template<typename T, typename U>`
`Shared_del_cap< T > shared_del_cap_reinterpret_cast (Shared_del_cap< U > &&from) noexcept`
Create a new shared capability by a reinterpret cast from another shared capability with move semantics.
- `template<typename T, typename U>`
`Shared_del_cap< T > shared_del_cap_dynamic_cast (Shared_del_cap< U > const &from) noexcept`
Create a new shared capability by a dynamic cast from another shared capability.
- `template<typename T, typename U>`
`Shared_del_cap< T > shared_del_cap_dynamic_cast (Shared_del_cap< U > &&from) noexcept`
Create a new shared capability by an dynamic cast from another shared capability with move semantics.
- `template<typename T>`
`Unique_cap< T > make_unique_cap (L4Re::Cap_alloc *ca)`
Allocate a capability slot and wrap it in an `Unique_cap`.
- `template<typename T>`
`Unique_del_cap< T > make_unique_del_cap (L4Re::Cap_alloc *ca)`
Allocate a capability slot and wrap it in an `Unique_del_cap`.

14.11.1 Detailed Description

[L4Re C++ Interfaces](#).

[L4 Runtime Environment](#).

14.11.2 Typedef Documentation

14.11.2.1 Shared_cap

```
template<typename T>
using L4Re::Shared_cap = L4::Detail::Shared_cap_impl<T, L4Re::Smart_count_cap<L4_FP_ALL_SPACES>>
```

Shared capability that implements automatic free and unmap of the capability selector.

Template Parameters

| | |
|----------------|--|
| <code>T</code> | Type of the object the capability refers to. |
|----------------|--|

This shared capability implements a counted reference to a capability selector. The capability shall be unmapped and freed when the reference count in the allocator goes to zero.

Note

This type is intended for users who implement a custom capability allocator; otherwise use [L4Re::Util::Shared_cap](#).

Definition at line 34 of file [shared_cap](#).

14.11.2.2 Shared_del_cap

```
template<typename T>
using L4Re::Shared_del_cap = L4::Detail::Shared_cap_impl<T, L4Re::Smart_count_cap<L4_FP_DELETE_OBJ>>
```

Shared capability that implements automatic free and unmap+delete of the capability selector.

Template Parameters

| | |
|----------|--|
| <i>T</i> | Type of the object the capability refers to. |
|----------|--|

This shared capability implements a counted reference to a capability selector. The capability shall be unmapped and freed when the reference count in the allocator goes to zero. The main difference to [Shared_cap](#) is that the unmap is done with the deletion flag enabled and this leads to the deletion of the object if the current task holds appropriate deletion rights.

Note

This type is intended for users who implement a custom capability allocator; otherwise use [L4Re::Util::Shared_del_cap](#).

Definition at line 185 of file [shared_cap](#).

14.11.2.3 Unique_cap

```
template<typename T>
using L4Re::Unique_cap = L4::Detail::Unique_cap_impl<T, L4Re::Smart_cap_auto<L4_FP_ALL_SPACES>>
```

Unique capability that implements automatic free and unmap of the capability selector.

Template Parameters

| | |
|----------|--|
| <i>T</i> | Type of the object the capability refers to. |
|----------|--|

The ownership of the capability is managed in the same way as [unique_ptr](#).

Note

This type is intended for users who implement a custom capability allocator; otherwise use [L4Re::Util::Unique_cap](#).

Definition at line 31 of file [unique_cap](#).

14.11.2.4 Unique_del_cap

```
template<typename T>
using L4Re::Unique_del_cap = L4::Detail::Unique_cap_impl<T, L4Re::Smart_cap_auto<L4_FP_DELETE_OBJ>>
```

Unique capability that implements automatic free and unmap+delete of the capability selector.

Template Parameters

| | |
|----------|--|
| <i>T</i> | Type of the object the capability refers to. |
|----------|--|

The main difference to [Unique_cap](#) is that the unmap is done with the deletion flag enabled and this leads to the deletion of the object if the current task holds appropriate deletion rights.

Note

This type is intended for users who implement a custom capability allocator; otherwise use [L4Re::Util::Unique_del_cap](#).

Definition at line 65 of file [unique_cap](#).

14.11.3 Function Documentation

14.11.3.1 chkcap()

```
template<typename T>
T L4Re::chkcap (
    T && cap,
    char const * extra = "",
    l4_ret_t err = -L4_ENOMEM) [inline]
```

Check for valid capability or raise C++ exception.

Template Parameters

| | |
|----------|---|
| <i>T</i> | Type of object to check, must be capability-like (L4::Cap , L4Re::Util::Unique_cap etc.) |
|----------|---|

Parameters

| | |
|--------------|---|
| <i>cap</i> | Capability value to check. |
| <i>extra</i> | Optional text for exception. |
| <i>err</i> | Error value for exception or 0 if the error code stored in the invalid capability should be used. |

This function checks whether the capability is valid. If the capability is invalid, a C++ exception is generated, using *err* if *err* is not zero, otherwise the capability value is used. A valid capability will just be returned.

Definition at line 149 of file [error_helper](#).

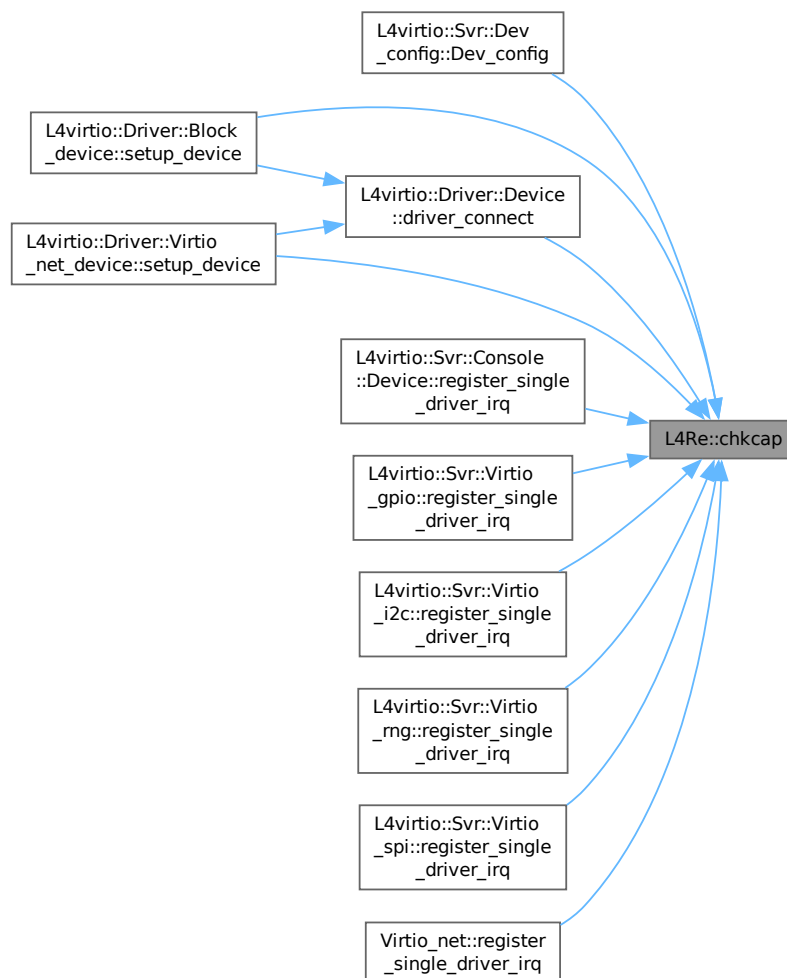
References [L4_ENOMEM](#), [L4_UNLIKELY](#), and [throw_error\(\)](#).

Referenced by [L4virtio::Svr::Dev_config::Dev_config\(\)](#), [L4virtio::Driver::Device::driver_connect\(\)](#), [L4virtio::Svr::Console::Device::register_single_driver_irq\(\)](#), [L4virtio::Svr::Virtio_gpio< Request_handler, Epiface >::register_single_driver_irq\(\)](#), [L4virtio::Svr::Virtio_i2c< Request_handler, Epiface >::register_single_driver_irq\(\)](#), [L4virtio::Svr::Virtio_rng< Rnd_state, Epiface >::register_single_driver_irq\(\)](#), [L4virtio::Svr::Virtio_spi< Spi_request_handler, Epiface >::register_single_driver_irq\(\)](#), [Virtio_net::register_single_driver_irq\(\)](#), [L4virtio::Driver::Block_device::setup_device\(\)](#), and [L4virtio::Driver::Virtio_net_device::setup_device\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



14.11.3.2 chkipc()

```
l4_msgtag_t L4Re::chkipc (
    l4_msgtag_t tag,
    char const * extra = "",
    l4_utcb_t * utcb = l4_utcb()) [inline]
```

Test a message tag for IPC errors.

Parameters

| | |
|--------------|-------------------------------------|
| <i>tag</i> | Message tag returned by the IPC. |
| <i>extra</i> | Exception message in case of error. |
| <i>utcb</i> | The UTCB used in the IPC operation. |

Returns

On IPC error an exception is thrown, otherwise `tag` is returned.

Exceptions

| | |
|------------------------------|------------------------------------|
| <i>L4::Runtime_exception</i> | with the translated IPC error code |
|------------------------------|------------------------------------|

This function does not check the message tag's label value.

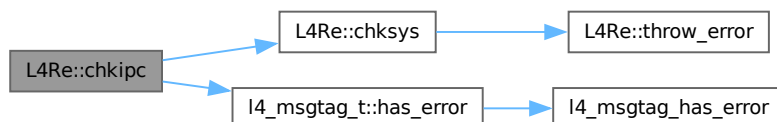
Note

This must be called on a message tag before the UTCB is changed.

Definition at line 180 of file [error_helper](#).

References [chksys\(\)](#), [l4_msgtag_t::has_error\(\)](#), and [L4_UNLIKELY](#).

Here is the call graph for this function:



14.11.3.3 chksys() [1/3]

```
l4_ret_t L4Re::chksys (
    l4_msgtag_t const & t,
    char const * extra = "",
    l4_utcb_t * utcb = l4_utcb(),
    l4_ret_t ret = 0) [inline]
```

Generate C++ exception on error.

Parameters

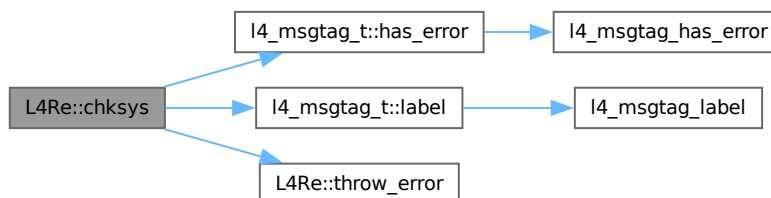
| | |
|--------------|--|
| <i>t</i> | Message tag. |
| <i>extra</i> | Optional text for exception (default "") |
| <i>utcb</i> | Option UTCB |
| <i>ret</i> | Optional value for exception, default is error value (err) |

This function throws an exception if the message tag contains an error or the label in the message tag is negative. Otherwise the label in the message tag is returned.

Definition at line 93 of file [error_helper](#).

References [l4_msgtag_t::has_error\(\)](#), [L4_UNLIKELY](#), [l4_msgtag_t::label\(\)](#), and [throw_error\(\)](#).

Here is the call graph for this function:

**14.11.3.4 chksys()** [2/3]

```
l4_ret_t L4Re::chksys (
    l4_msgtag_t const & t,
    l4_utcb_t * utcb,
    char const * extra = "") [inline]
```

Generate C++ exception on error.

Parameters

| | |
|----------|--------------|
| <i>t</i> | Message tag. |
|----------|--------------|

| | |
|--------------|--|
| <i>utcb</i> | UTCB. |
| <i>extra</i> | Optional text for exception (default "") |

This function throws an exception if the message tag contains an error or the label in the message tag is negative. Otherwise the label in the message tag is returned.

Definition at line 116 of file [error_helper](#).

References [chksys\(\)](#).

Here is the call graph for this function:



14.11.3.5 chksys() [3/3]

```

l4_ret_t L4Re::chksys (
    l4_ret_t err,
    char const * extra = "",
    l4_ret_t ret = 0) [inline]

```

Generate C++ exception on error.

Parameters

| | |
|--------------|--|
| <i>err</i> | Error value, if negative exception will be thrown |
| <i>extra</i> | Optional text for exception (default "") |
| <i>ret</i> | Optional value for exception, default is error value (err) |

This function throws an exception if the `err` is negative and otherwise returns `err`.

Examples

[examples/libs/l4re/c++/shared_ds/ds_srv.cc](#).

Definition at line 72 of file [error_helper](#).

References [L4_UNLIKELY](#), and [throw_error\(\)](#).

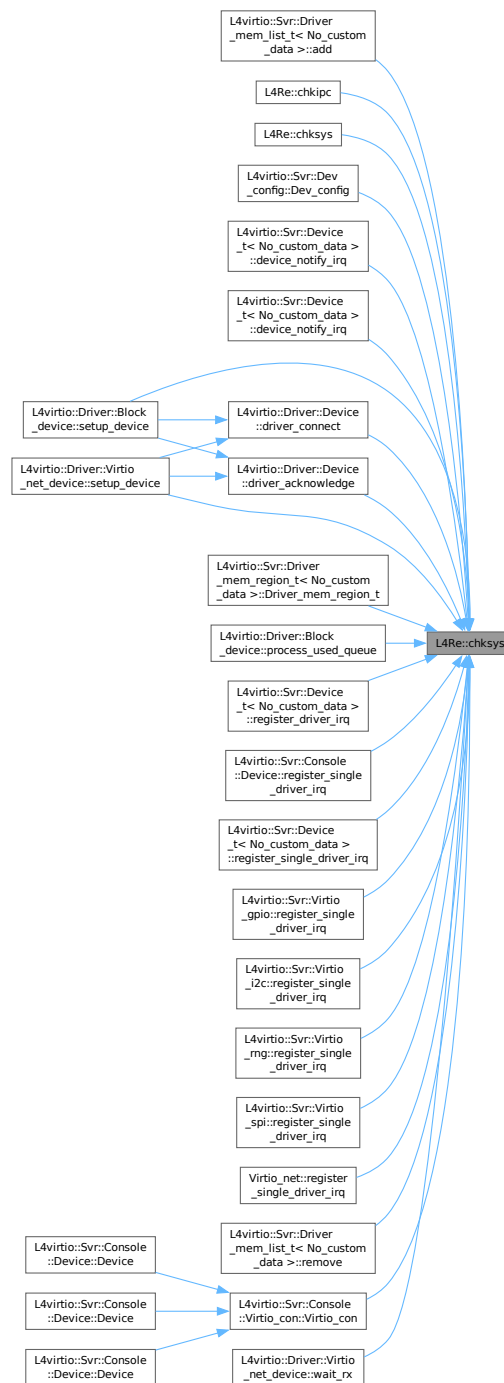
Referenced by [L4virtio::Svr::Driver_mem_list_t< No_custom_data >::add\(\)](#), [chkipc\(\)](#), [chksys\(\)](#), [L4virtio::Svr::Dev_config::Dev_config](#), [L4virtio::Svr::Device_t< No_custom_data >::device_notify_irq\(\)](#), [L4virtio::Svr::Device_t< No_custom_data >::device_notify_irq\(\)](#), [L4virtio::Driver::Device::driver_acknowledge\(\)](#), [L4virtio::Driver::Device::driver_connect\(\)](#), [L4virtio::Svr::Driver_mem_region_t< No_cu](#), [L4virtio::Driver::Block_device::process_used_queue\(\)](#), [L4virtio::Svr::Device_t< No_custom_data >::register_driver_irq\(\)](#), [L4virtio::Svr::Console::Device::register_single_driver_irq\(\)](#), [L4virtio::Svr::Device_t< No_custom_data >::register_single_driver_irq\(\)](#),

[L4virtio::Svr::Virtio_gpio< Request_handler, Epiface >::register_single_driver_irq\(\)](#), [L4virtio::Svr::Virtio_i2c< Request_handler, Epiface >::register_single_driver_irq\(\)](#), [L4virtio::Svr::Virtio_rng< Rnd_state, Epiface >::register_single_driver_irq\(\)](#), [L4virtio::Svr::Virtio_spi< Spi_request_handler, Epiface >::register_single_driver_irq\(\)](#), [L4virtio::Svr::Virtio_net::register_single_driver_irq\(\)](#), [L4virtio::Svr::Driver_mem_list_t< No_custom_data >::remove\(\)](#), [L4virtio::Driver::Block_device::remove\(\)](#), [L4virtio::Driver::Virtio_net_device::setup_device\(\)](#), [L4virtio::Svr::Console::Virtio_con::Virtio_con\(\)](#), and [L4virtio::Driver::Virtio_net_device::setup_device\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



14.11.3.6 make_shared_cap()

```

template<typename T>
Shared_cap< T > L4Re::make_shared_cap (
    L4Re::Cap_alloc * ca)

```

Allocate a capability slot and wrap it in a [Shared_cap](#).

Template Parameters

| | |
|----------|--|
| <i>T</i> | Type of the object the capability refers to. |
|----------|--|

Parameters

| | |
|-----------|------------------------------|
| <i>ca</i> | Capability allocator to use. |
|-----------|------------------------------|

Note

This function is intended for users who implement a custom capability allocator; otherwise use [L4Re::Util::make_shared_cap<T>\(\)](#).

Definition at line 51 of file [shared_cap](#).

References [L4Re::Cap_alloc::alloc\(\)](#).

Here is the call graph for this function:



14.11.3.7 make_shared_del_cap()

```

template<typename T>
Shared\_del\_cap< T > L4Re::make_shared_del_cap (
    L4Re::Cap\_alloc * ca)
  
```

Allocate a capability slot and wrap it in a [Shared_del_cap](#).

Template Parameters

| | |
|----------|--|
| <i>T</i> | Type of the object the capability refers to. |
|----------|--|

Parameters

| | |
|-----------|------------------------------|
| <i>ca</i> | Capability allocator to use. |
|-----------|------------------------------|

Note

This function is intended for users who implement a custom capability allocator; otherwise use [L4Re::Util::make_shared_del_cap<T>\(\)](#).

Definition at line 202 of file [shared_cap](#).

References [L4Re::Cap_alloc::alloc\(\)](#).

Here is the call graph for this function:

**14.11.3.8 make_unique_cap()**

```

template<typename T>
Unique_cap< T > L4Re::make_unique_cap (
    L4Re::Cap_alloc * ca)
  
```

Allocate a capability slot and wrap it in an [Unique_cap](#).

Template Parameters

| | |
|----------|--|
| <i>T</i> | Type of the object the capability refers to. |
|----------|--|

Parameters

| | |
|-----------|------------------------------|
| <i>ca</i> | Capability allocator to use. |
|-----------|------------------------------|

Note

This function is intended for users who implement a custom capability allocator; otherwise use [L4Re::Util::make_unique_cap<T>\(\)](#).

Definition at line 48 of file [unique_cap](#).

References [L4Re::Cap_alloc::alloc\(\)](#).

Here is the call graph for this function:



14.11.3.9 make_unique_del_cap()

```
template<typename T>
Unique_del_cap< T > L4Re::make_unique_del_cap (
    L4Re::Cap_alloc * ca)
```

Allocate a capability slot and wrap it in an [Unique_del_cap](#).

Template Parameters

| | |
|----------|--|
| <i>T</i> | Type of the object the capability refers to. |
|----------|--|

Parameters

| | |
|-----------|------------------------------|
| <i>ca</i> | Capability allocator to use. |
|-----------|------------------------------|

Note

This function is intended for users who implement a custom capability allocator; otherwise use [L4Re::Util::make_unique_del_cap<T>\(\)](#).

Definition at line 82 of file [unique_cap](#).

References [L4Re::Cap_alloc::alloc\(\)](#).

Here is the call graph for this function:



14.11.3.10 shared_cap_cast() [1/2]

```
template<typename T, typename U>
Shared_cap< T > L4Re::shared_cap_cast (
    Shared_cap< U > && from) [noexcept]
```

Create a new shared capability by an explicit cast from another shared capability with move semantics.

Type *U* must be convertible to type *T*. After the call, *from* is empty.

See also

[L4::cap_cast](#)

Template Parameters

| | |
|----------|--|
| <i>T</i> | Type of the object the capability refers to. |
| <i>U</i> | Type of the source shared capability. |

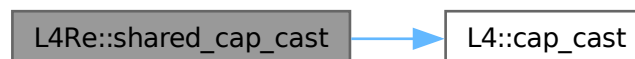
Parameters

| | |
|-------------|---------------------------|
| <i>from</i> | Source shared capability. |
|-------------|---------------------------|

Definition at line 86 of file [shared_cap](#).

References [L4::cap_cast\(\)](#).

Here is the call graph for this function:



14.11.3.11 shared_cap_cast() [2/2]

```

template<typename T, typename U>
Shared_cap< T > L4Re::shared_cap_cast (
    Shared_cap< U > const & from) [noexcept]
  
```

Create a new shared capability by an explicit cast from another shared capability.

Type *U* must be convertible to type *T*.

See also

[L4::cap_cast](#)

Template Parameters

| | |
|----------|--|
| <i>T</i> | Type of the object the capability refers to. |
| <i>U</i> | Type of the source shared capability. |

Parameters

| | |
|-------------|---------------------------|
| <i>from</i> | Source shared capability. |
|-------------|---------------------------|

Definition at line 67 of file [shared_cap](#).

References [L4::cap_cast\(\)](#).

Here is the call graph for this function:

**14.11.3.12 shared_cap_dynamic_cast() [1/2]**

```

template<typename T, typename U>
Shared\_cap< T > L4Re::shared_cap_dynamic_cast (
    Shared\_cap< U > && from) [noexcept]
  
```

Create a new shared capability by an dynamic cast from another shared capability with move semantics.

See also

[L4::cap_dynamic_cast](#)

After the call, *from* is empty, unless the [L4::cap_dynamic_cast](#) fails.

Template Parameters

| | |
|----------|--|
| <i>T</i> | Type of the object the capability refers to. |
| <i>U</i> | Type of the source shared capability. |

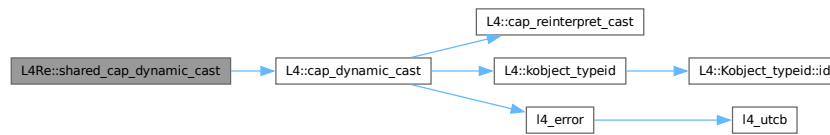
Parameters

| | |
|-------------|---------------------------|
| <i>from</i> | Source shared capability. |
|-------------|---------------------------|

Definition at line 160 of file [shared_cap](#).

References [L4::cap_dynamic_cast\(\)](#).

Here is the call graph for this function:



14.11.3.13 shared_cap_dynamic_cast() [2/2]

```

template<typename T, typename U>
Shared_cap< T > L4Re::shared_cap_dynamic_cast (
    Shared_cap< U > const & from) [noexcept]
  
```

Create a new shared capability by a dynamic cast from another shared capability.

See also

[L4::cap_dynamic_cast](#)

Template Parameters

| | |
|----------|--|
| <i>T</i> | Type of the object the capability refers to. |
| <i>U</i> | Type of the source shared capability. |

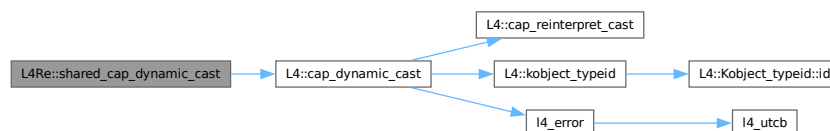
Parameters

| | |
|-------------|---------------------------|
| <i>from</i> | Source shared capability. |
|-------------|---------------------------|

Definition at line 139 of file [shared_cap](#).

References [L4::cap_dynamic_cast\(\)](#).

Here is the call graph for this function:



14.11.3.14 shared_cap_reinterpret_cast() [1/2]

```
template<typename T, typename U>
Shared_cap< T > L4Re::shared_cap_reinterpret_cast (
    Shared_cap< U > && from) [noexcept]
```

Create a new shared capability by a reinterpret cast from another shared capability with move semantics.

After the call, `from` is empty.

See also

[L4::cap_reinterpret_cast](#)

Template Parameters

| | |
|----------|--|
| <i>T</i> | Type of the object the capability refers to. |
| <i>U</i> | Type of the source shared capability. |

Parameters

| | |
|-------------|---------------------------|
| <i>from</i> | Source shared capability. |
|-------------|---------------------------|

Definition at line 122 of file [shared_cap](#).

References [L4::cap_reinterpret_cast\(\)](#).

Here is the call graph for this function:

**14.11.3.15 shared_cap_reinterpret_cast() [2/2]**

```
template<typename T, typename U>
Shared_cap< T > L4Re::shared_cap_reinterpret_cast (
    Shared_cap< U > const & from) [noexcept]
```

Create a new shared capability by a reinterpret cast from another shared capability.

See also

[L4::cap_reinterpret_cast](#)

Template Parameters

| | |
|----------|--|
| <i>T</i> | Type of the object the capability refers to. |
| <i>U</i> | Type of the source shared capability. |

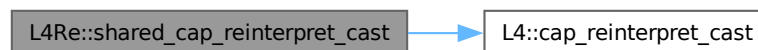
Parameters

| | |
|-------------|---------------------------|
| <i>from</i> | Source shared capability. |
|-------------|---------------------------|

Definition at line 103 of file [shared_cap](#).

References [L4::cap_reinterpret_cast\(\)](#).

Here is the call graph for this function:



14.11.3.16 shared_del_cap_cast() [1/2]

```

template<typename T, typename U>
Shared_del_cap< T > L4Re::shared_del_cap_cast (
    Shared_del_cap< U > && from) [noexcept]
  
```

Create a new shared capability by an explicit cast from another shared capability with move semantics.

Type *U* must be convertible to type *T*. After the call, *from* is empty.

See also

[L4::cap_cast](#)

Template Parameters

| | |
|----------|--|
| <i>T</i> | Type of the object the capability refers to. |
| <i>U</i> | Type of the source shared capability. |

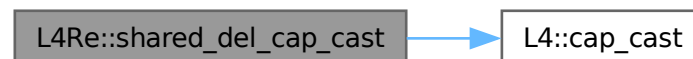
Parameters

| | |
|-------------|---------------------------|
| <i>from</i> | Source shared capability. |
|-------------|---------------------------|

Definition at line 237 of file [shared_cap](#).

References [L4::cap_cast\(\)](#).

Here is the call graph for this function:

**14.11.3.17 shared_del_cap_cast() [2/2]**

```

template<typename T, typename U>
Shared\_del\_cap< T > L4Re::shared_del_cap_cast (
    Shared\_del\_cap< U > const & from)    [noexcept]
  
```

Create a new shared capability by an explicit cast from another shared capability.

Type U must be convertible to type T.

See also

[L4::cap_cast](#)

Template Parameters

| | |
|----------|--|
| <i>T</i> | Type of the object the capability refers to. |
| <i>U</i> | Type of the source shared capability. |

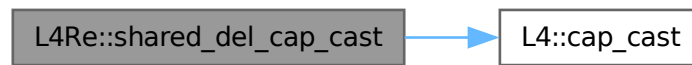
Parameters

| | |
|-------------|---------------------------|
| <i>from</i> | Source shared capability. |
|-------------|---------------------------|

Definition at line 218 of file [shared_cap](#).

References [L4::cap_cast\(\)](#).

Here is the call graph for this function:



14.11.3.18 shared_del_cap_dynamic_cast() [1/2]

```

template<typename T, typename U>
Shared_del_cap< T > L4Re::shared_del_cap_dynamic_cast (
    Shared_del_cap< U > && from) [noexcept]
  
```

Create a new shared capability by an dynamic cast from another shared capability with move semantics.

See also

[L4::cap_dynamic_cast](#)

After the call, `from` is empty, unless the [L4::cap_dynamic_cast](#) fails.

Template Parameters

| | |
|----------|--|
| <i>T</i> | Type of the object the capability refers to. |
| <i>U</i> | Type of the source shared capability. |

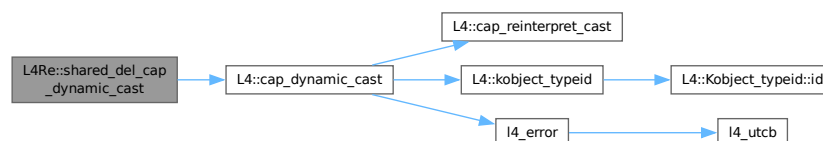
Parameters

| | |
|-------------|---------------------------|
| <i>from</i> | Source shared capability. |
|-------------|---------------------------|

Definition at line 315 of file [shared_cap](#).

References [L4::cap_dynamic_cast\(\)](#).

Here is the call graph for this function:



14.11.3.19 `shared_del_cap_dynamic_cast()` [2/2]

```
template<typename T, typename U>
Shared_del_cap< T > L4Re::shared_del_cap_dynamic_cast (
    Shared_del_cap< U > const & from) [noexcept]
```

Create a new shared capability by a dynamic cast from another shared capability.

See also

[L4::cap_dynamic_cast](#)

Template Parameters

| | |
|----------|--|
| <i>T</i> | Type of the object the capability refers to. |
| <i>U</i> | Type of the source shared capability. |

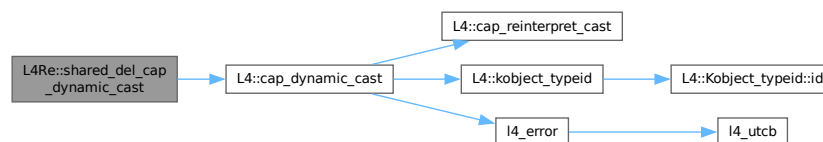
Parameters

| | |
|-------------|---------------------------|
| <i>from</i> | Source shared capability. |
|-------------|---------------------------|

Definition at line 293 of file [shared_cap](#).

References [L4::cap_dynamic_cast\(\)](#).

Here is the call graph for this function:



14.11.3.20 `shared_del_cap_reinterpret_cast()` [1/2]

```
template<typename T, typename U>
Shared_del_cap< T > L4Re::shared_del_cap_reinterpret_cast (
    Shared_del_cap< U > && from) [noexcept]
```

Create a new shared capability by a reinterpret cast from another shared capability with move semantics.

After the call, `from` is empty.

See also

[L4::cap_reinterpret_cast](#)

Template Parameters

| | |
|----------|--|
| <i>T</i> | Type of the object the capability refers to. |
| <i>U</i> | Type of the source shared capability. |

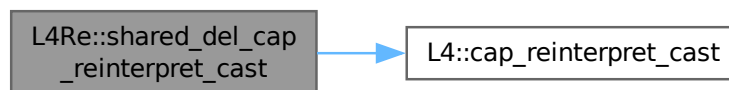
Parameters

| | |
|-------------|---------------------------|
| <i>from</i> | Source shared capability. |
|-------------|---------------------------|

Definition at line 275 of file [shared_cap](#).

References [L4::cap_reinterpret_cast\(\)](#).

Here is the call graph for this function:



14.11.3.21 shared_del_cap_reinterpret_cast() [2/2]

```

template<typename T, typename U>
Shared_del_cap< T > L4Re::shared_del_cap_reinterpret_cast (
    Shared_del_cap< U > const & from) [noexcept]
  
```

Create a new shared capability by a reinterpret cast from another shared capability.

See also

[L4::cap_reinterpret_cast](#)

Template Parameters

| | |
|----------|--|
| <i>T</i> | Type of the object the capability refers to. |
| <i>U</i> | Type of the source shared capability. |

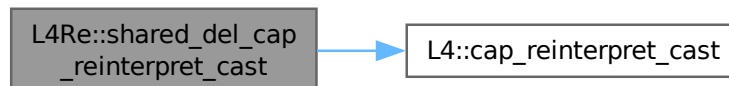
Parameters

| | |
|-------------|---------------------------|
| <i>from</i> | Source shared capability. |
|-------------|---------------------------|

Definition at line 255 of file [shared_cap](#).

References [L4::cap_reinterpret_cast\(\)](#).

Here is the call graph for this function:

**14.11.3.22 throw_error()**

```

void L4Re::throw_error (
    long err,
    char const * extra = "") [inline]
  
```

Generate C++ exception.

Parameters

| | |
|--------------|--|
| <i>err</i> | Error value |
| <i>extra</i> | Optional text for exception (default "") |

This function throws an [L4](#) exception. The exact exception type depends on the error value (err). This function does never return.

Examples

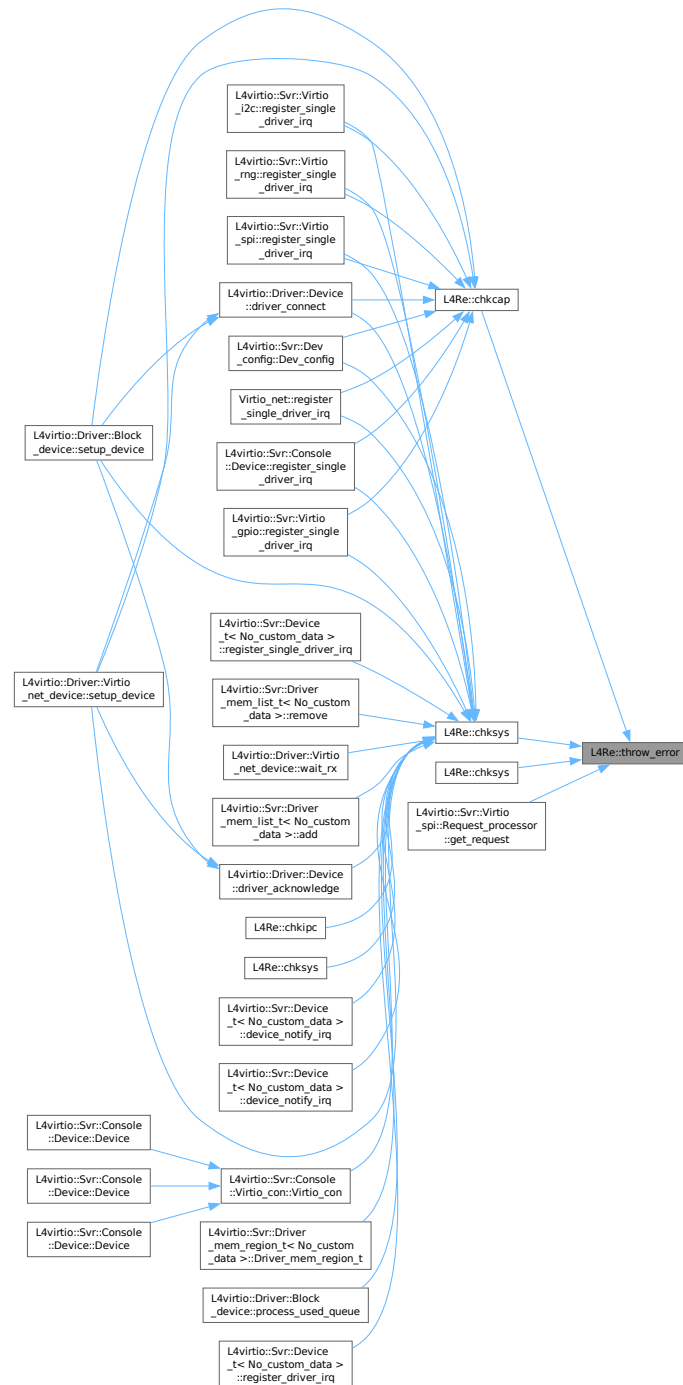
[examples/libs/l4re/c++/shared_ds/ds_srv.cc](#).

Definition at line 37 of file [error_helper](#).

References [L4_EEXIST](#), [L4_ENOENT](#), [L4_ENOMEM](#), and [L4_ERANGE](#).

Referenced by [chkcap\(\)](#), [chksys\(\)](#), [chksys\(\)](#), and [L4virtio::Svr::Virtio_spi< Spi_request_handler, Epiface >::Request_processor::get_](#)

Here is the caller graph for this function:



14.12 L4Re::Util Namespace Reference

Documentation of the [L4](#) Runtime Environment utility functionality in C++.

Data Structures

- class [Cap_alloc_base](#)
Capability allocator.
- class [Br_manager](#)
Buffer-register (BR) manager for [L4::Server](#).
- struct [Br_manager_hooks](#)
Predefined server-loop hooks for a server loop using the [Br_manager](#).
- struct [Br_manager_timeout_hooks](#)
Predefined server-loop hooks for a server with using the [Br_manager](#) and a timeout queue.
- class [Smart_cap_auto](#)
Helper for [Unique_cap](#) and [Unique_del_cap](#).
- class [Smart_count_cap](#)
Helper for [Ref_cap](#) and [Ref_del_cap](#).
- struct [Ref_cap](#)
Automatic capability that implements automatic free and unmap of the capability selector.
- struct [Ref_del_cap](#)
Automatic capability that implements automatic free and unmap+delete of the capability selector.
- class [_Cap_alloc](#)
Adapter to expose the cap allocator implementation as [L4Re::Cap_alloc](#) compatible class.
- struct [Counter](#)
Counter for [Counting_cap_alloc](#) with variable data width.
- struct [Counter_atomic](#)
Thread safe version of counter for [Counting_cap_alloc](#).
- class [Counting_cap_alloc](#)
Internal reference-counting cap allocator.
- class [Dataspace_svr](#)
[Dataspace](#) server class.
- class [Event_t](#)
Convenience wrapper for getting access to an event object.
- class [Event_buffer_t](#)
Event_buffer utility class.
- class [Event_buffer_consumer_t](#)
An event buffer consumer.
- class [Event_svr](#)
Convenience wrapper for implementing an event server.
- class [Item_alloc_base](#)
Item allocator.
- class [Object_registry](#)
A registry that manages server objects and their attached IPC gates for a single server loop for a specific thread.
- class [Registry_server](#)
A server loop object which has a [Object_registry](#) included.
- class [Vcon_svr](#)
[Console](#) server template class.
- class [Bitmap_base](#)
Basic bitmap abstraction.
- class [Bitmap](#)
A static bitmap.

Typedefs

- `template<typename T>`
using `Shared_cap` = `L4::Detail::Shared_cap_impl<T, L4Re::Util::Smart_count_cap<L4_FP_ALL_SPACES>>`
Shared capability that implements automatic free and unmap of the capability selector.
- `template<typename T>`
using `Shared_del_cap` = `L4::Detail::Shared_cap_impl<T, L4Re::Util::Smart_count_cap<L4_FP_DELETE_OBJ>>`
Shared capability that implements automatic free and unmap+delete of the capability selector.
- `template<typename T>`
using `Unique_cap` = `L4::Detail::Unique_cap_impl<T, L4Re::Util::Smart_cap_auto<L4_FP_ALL_SPACES>>`
Unique capability that implements automatic free and unmap of the capability selector.
- `template<typename T>`
using `Unique_del_cap` = `L4::Detail::Unique_cap_impl<T, L4Re::Util::Smart_cap_auto<L4_FP_DELETE_OBJ>>`
Unique capability that implements automatic free and unmap+delete of the capability selector.

Functions

- `template<typename T>`
`Ref_cap< T >::Cap make_ref_cap ()`
Allocate a capability slot and wrap it in a `Ref_cap`.
- `template<typename T>`
`Ref_del_cap< T >::Cap make_ref_del_cap ()`
Allocate a capability slot and wrap it in a `Ref_del_cap`.
- `int kumem_alloc (l4_addr_t *mem, unsigned pages_order, L4::Cap< L4::Task > task=L4Re::Env::env() ->task(), L4::Cap< L4Re::Rm > rm=L4Re::Env::env() ->rm()) noexcept`
Allocate state area.
- `template<typename T>`
`Shared_cap< T > make_shared_cap ()`
Allocate a capability slot and wrap it in a `Shared_cap`.
- `template<typename T, typename U>`
`Shared_cap< T > shared_cap_cast (Shared_cap< U > const &from) noexcept`
Create a new shared capability by an explicit cast from another shared capability.
- `template<typename T, typename U>`
`Shared_cap< T > shared_cap_cast (Shared_cap< U > &&from) noexcept`
Create a new shared capability by an explicit cast from another shared capability with move semantics.
- `template<typename T, typename U>`
`Shared_cap< T > shared_cap_reinterpret_cast (Shared_cap< U > const &from) noexcept`
Create a new shared capability by a reinterpret cast from another shared capability.
- `template<typename T, typename U>`
`Shared_cap< T > shared_cap_reinterpret_cast (Shared_cap< U > &&from) noexcept`
Create a new shared capability by a reinterpret cast from another shared capability with move semantics.
- `template<typename T, typename U>`
`Shared_cap< T > shared_cap_dynamic_cast (Shared_cap< U > const &from) noexcept`
Create a new shared capability by a dynamic cast from another shared capability.
- `template<typename T, typename U>`
`Shared_cap< T > shared_cap_dynamic_cast (Shared_cap< U > &&from) noexcept`
Create a new shared capability by a dynamic cast from another shared capability with move semantics.
- `template<typename T>`
`Shared_del_cap< T > make_shared_del_cap ()`
Allocate a capability slot and wrap it in a `Shared_del_cap`.
- `template<typename T, typename U>`
`Shared_del_cap< T > shared_del_cap_cast (Shared_del_cap< U > const &from) noexcept`

Create a new shared capability by an explicit cast from another shared capability.

- `template<typename T, typename U>`
`Shared_del_cap< T > shared_del_cap_cast (Shared_del_cap< U > &&from) noexcept`

Create a new shared capability by an explicit cast from another shared capability with move semantics.

- `template<typename T, typename U>`
`Shared_del_cap< T > shared_del_cap_reinterpret_cast (Shared_del_cap< U > const &from) noexcept`

Create a new shared capability by a reinterpret cast from another shared capability.

- `template<typename T, typename U>`
`Shared_del_cap< T > shared_del_cap_reinterpret_cast (Shared_del_cap< U > &&from) noexcept`

Create a new shared capability by a reinterpret cast from another shared capability with move semantics.

- `template<typename T, typename U>`
`Shared_del_cap< T > shared_del_cap_dynamic_cast (Shared_del_cap< U > const &from) noexcept`

Create a new shared capability by a dynamic cast from another shared capability.

- `template<typename T, typename U>`
`Shared_del_cap< T > shared_del_cap_dynamic_cast (Shared_del_cap< U > &&from) noexcept`

Create a new shared capability by an dynamic cast from another shared capability with move semantics.

- `template<typename T>`
`Unique_cap< T > make_unique_cap ()`

Allocate a capability slot and wrap it in an `Unique_cap`.

- `template<typename T>`
`Unique_del_cap< T > make_unique_del_cap ()`

Allocate a capability slot and wrap it in an `Unique_del_cap`.

Variables

- `_Cap_alloc` `cap_alloc`
Capability allocator.
- `Def_reply_cap_alloc` `reply_cap_alloc`
Reply capability allocator.

14.12.1 Detailed Description

Documentation of the [L4](#) Runtime Environment utility functionality in C++.

14.12.2 Typedef Documentation

14.12.2.1 Shared_cap

```
template<typename T>
using L4Re::Util::Shared_cap = L4::Detail::Shared_cap_impl<T, L4Re::Util::Smart_count_cap<L4_FP_ALL_SPACES>>
```

Shared capability that implements automatic free and unmap of the capability selector.

Template Parameters

| | |
|----------------|--|
| <code>T</code> | Type of the object the capability refers to. |
|----------------|--|

This shared capability implements a counted reference to a capability selector. The capability shall be unmapped and freed when the reference count in the allocator goes to zero.

Usage:

```

L4Re::Util::Shared_cap<L4Re::Dataspace> global_ds_cap;

{
    L4Re::Util::Shared_cap<L4Re::Dataspace>
        ds_cap = make_shared_cap<L4Re::Dataspace>();
    // reference count for the allocated cap selector is now 1

    // use the dataspace cap
    L4Re::chksys(mem_alloc->alloc(4096, ds_cap.get()));

    global_ds_cap = ds_cap;
    // reference count is now 2
    ...
}
// reference count dropped to 1 (ds_cap is no longer existing).

```

Definition at line 49 of file [shared_cap](#).

14.12.2.2 Shared_del_cap

```

template<typename T>
using L4Re::Util::Shared_del_cap = L4::Detail::Shared_cap_impl<T, L4Re::Util::Smart_count_cap<L4_FP_DELETE_OB

```

Shared capability that implements automatic free and unmap+delete of the capability selector.

Template Parameters

| | |
|----------|--|
| <i>T</i> | Type of the object the capability refers to. |
|----------|--|

This shared capability implements a counted reference to a capability selector. The capability shall be unmapped and freed when the reference count in the allocator goes to zero. The main difference to [Shared_cap](#) is that the unmap is done with the deletion flag enabled and this leads to the deletion of the object if the current task holds appropriate deletion rights.

Usage:

```

L4Re::Util::Shared_del_cap<L4Re::Dataspace> global_ds_cap;

{
    L4Re::Util::Shared_del_cap<L4Re::Dataspace>
        ds_cap = make_shared_del_cap<L4Re::Dataspace>();
    // reference count for the allocated cap selector is now 1

    // use the dataspace cap
    L4Re::chksys(mem_alloc->alloc(4096, ds_cap.get()));

    global_ds_cap = ds_cap;
    // reference count is now 2
    ...
}
// reference count dropped to 1 (ds_cap is no longer existing).
...
global_ds_cap = L4_INVALID_CAP;
// reference count dropped to 0 (data space shall be deleted).

```

Definition at line 214 of file [shared_cap](#).

14.12.2.3 Unique_cap

```
template<typename T>
using L4Re::Util::Unique_cap = L4::Detail::Unique_cap_impl<T, L4Re::Util::Smart_cap_auto<L4_FP_ALL_SPACES>>
```

Unique capability that implements automatic free and unmap of the capability selector.

Template Parameters

| | |
|----------|--|
| <i>T</i> | Type of the object the capability refers to. |
|----------|--|

The ownership of the capability is managed in the same way as `unique_ptr`.

Usage:

```
{
    L4Re::Util::Unique_cap<L4Re::Dataspace>
        ds_cap = L4Re::Util::make_unique_cap<L4Re::Dataspace>();

    // use the dataspace cap
    L4Re::chksys(mem_alloc->alloc(L4_PAGESIZE, ds_cap.get()));

    ...

    // At the end of the scope ds_cap is unmapped and the capability
    // selector is freed.
}
```

Definition at line 43 of file [unique_cap](#).

14.12.2.4 Unique_del_cap

```
template<typename T>
using L4Re::Util::Unique_del_cap = L4::Detail::Unique_cap_impl<T, L4Re::Util::Smart_cap_auto<L4_FP_DELETE_OBJ>
```

Unique capability that implements automatic free and unmap+delete of the capability selector.

Template Parameters

| | |
|----------|--|
| <i>T</i> | Type of the object the capability refers to. |
|----------|--|

The main difference to [Unique_cap](#) is that the unmap is done with the deletion flag enabled and this leads to the deletion of the object if the current task holds appropriate deletion rights.

Usage:

```
{
    L4Re::Util::Unique_del_cap<L4Re::Dataspace>
        ds_cap = make_unique_del_cap<L4Re::Dataspace>();

    // use the dataspace cap
    L4Re::chksys(mem_alloc->alloc(L4_PAGESIZE, ds_cap.get()));

    ...

    // At the end of the scope ds_cap is unmapped and the capability
    // selector is freed. Because the deletion flag is set the data space
    // shall also be deleted (even if there are other references to this
    // data space).
}
```

Definition at line 87 of file [unique_cap](#).

14.12.3 Function Documentation

14.12.3.1 make_shared_cap()

```
template<typename T>  
Shared\_cap< T > L4Re::Util::make_shared_cap ()
```

Allocate a capability slot and wrap it in a [Shared_cap](#).

Template Parameters

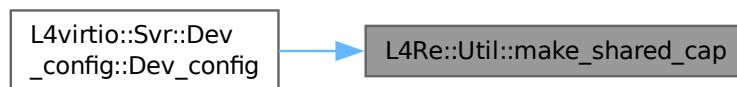
| | |
|----------|--|
| <i>T</i> | Type of the object the capability refers to. |
|----------|--|

Definition at line 62 of file [shared_cap](#).

References [cap_alloc](#).

Referenced by [L4virtio::Svr::Dev_config::Dev_config\(\)](#).

Here is the caller graph for this function:



14.12.3.2 make_shared_del_cap()

```
template<typename T>  
Shared\_del\_cap< T > L4Re::Util::make_shared_del_cap ()
```

Allocate a capability slot and wrap it in a [Shared_del_cap](#).

Template Parameters

| | |
|----------|--|
| <i>T</i> | Type of the object the capability refers to. |
|----------|--|

Definition at line 227 of file [shared_cap](#).

References [cap_alloc](#).

14.12.3.3 make_unique_cap()

```
template<typename T>
Unique_cap< T > L4Re::Util::make_unique_cap ()
```

Allocate a capability slot and wrap it in an [Unique_cap](#).

Template Parameters

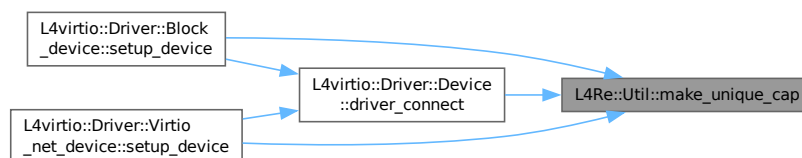
| | |
|----------|--|
| <i>T</i> | Type of the object the capability refers to. |
|----------|--|

Definition at line 56 of file [unique_cap](#).

References [cap_alloc](#).

Referenced by [L4virtio::Driver::Device::driver_connect\(\)](#), [L4virtio::Driver::Block_device::setup_device\(\)](#), and [L4virtio::Driver::Virtio_net_device::setup_device\(\)](#).

Here is the caller graph for this function:



14.12.3.4 make_unique_del_cap()

```
template<typename T>
Unique_del_cap< T > L4Re::Util::make_unique_del_cap ()
```

Allocate a capability slot and wrap it in an [Unique_del_cap](#).

Template Parameters

| | |
|----------|--|
| <i>T</i> | Type of the object the capability refers to. |
|----------|--|

Definition at line 100 of file [unique_cap](#).

References [cap_alloc](#).

14.12.3.5 shared_cap_cast() [1/2]

```
template<typename T, typename U>
Shared_cap< T > L4Re::Util::shared_cap_cast (
    Shared_cap< U > && from) [noexcept]
```

Create a new shared capability by an explicit cast from another shared capability with move semantics.

Type `U` must be convertible to type `T`. After the call, `from` is empty.

See also

[L4::cap_cast](#)

Template Parameters

| | |
|----------|--|
| <i>T</i> | Type of the object the capability refers to. |
| <i>U</i> | Type of the source shared capability. |

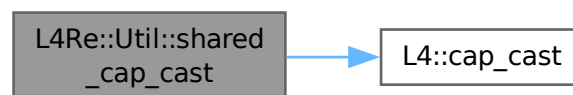
Parameters

| | |
|-------------|---------------------------|
| <i>from</i> | Source shared capability. |
|-------------|---------------------------|

Definition at line 97 of file [shared_cap](#).

References [L4::cap_cast\(\)](#).

Here is the call graph for this function:

**14.12.3.6 shared_cap_cast() [2/2]**

```
template<typename T, typename U>
Shared_cap< T > L4Re::Util::shared_cap_cast (
    Shared_cap< U > const & from) [noexcept]
```

Create a new shared capability by an explicit cast from another shared capability.

Type `U` must be convertible to type `T`.

See also

[L4::cap_cast](#)

Template Parameters

| | |
|----------|--|
| <i>T</i> | Type of the object the capability refers to. |
| <i>U</i> | Type of the source shared capability. |

Parameters

| | |
|-------------|---------------------------|
| <i>from</i> | Source shared capability. |
|-------------|---------------------------|

Definition at line 78 of file [shared_cap](#).

References [L4::cap_cast\(\)](#).

Here is the call graph for this function:



14.12.3.7 `shared_cap_dynamic_cast()` [1/2]

```

template<typename T, typename U>
Shared_cap< T > L4Re::Util::shared_cap_dynamic_cast (
    Shared_cap< U > && from) [noexcept]
  
```

Create a new shared capability by an dynamic cast from another shared capability with move semantics.

See also

[L4::cap_dynamic_cast](#)

After the call, `from` is empty, unless the [L4::cap_dynamic_cast](#) fails.

Template Parameters

| | |
|----------|--|
| <i>T</i> | Type of the object the capability refers to. |
| <i>U</i> | Type of the source shared capability. |

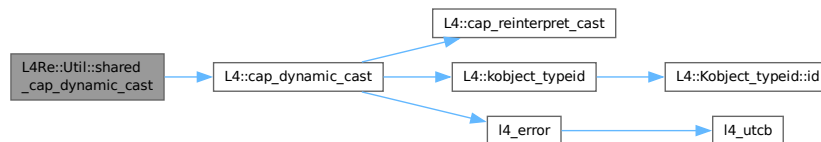
Parameters

| | |
|-------------|---------------------------|
| <i>from</i> | Source shared capability. |
|-------------|---------------------------|

Definition at line 171 of file [shared_cap](#).

References [L4::cap_dynamic_cast\(\)](#).

Here is the call graph for this function:

**14.12.3.8 shared_cap_dynamic_cast() [2/2]**

```

template<typename T, typename U>
Shared_cap< T > L4Re::Util::shared_cap_dynamic_cast (
    Shared_cap< U > const & from) [noexcept]
  
```

Create a new shared capability by a dynamic cast from another shared capability.

See also

[L4::cap_dynamic_cast](#)

Template Parameters

| | |
|----------|--|
| <i>T</i> | Type of the object the capability refers to. |
| <i>U</i> | Type of the source shared capability. |

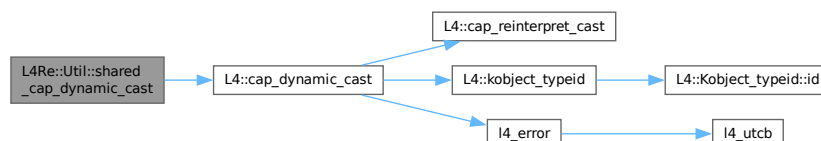
Parameters

| | |
|-------------|---------------------------|
| <i>from</i> | Source shared capability. |
|-------------|---------------------------|

Definition at line 150 of file [shared_cap](#).

References [L4::cap_dynamic_cast\(\)](#).

Here is the call graph for this function:



14.12.3.9 shared_cap_reinterpret_cast() [1/2]

```
template<typename T, typename U>
Shared_cap< T > L4Re::Util::shared_cap_reinterpret_cast (
    Shared_cap< U > && from) [noexcept]
```

Create a new shared capability by a reinterpret cast from another shared capability with move semantics.

After the call, `from` is empty.

See also

[L4::cap_reinterpret_cast](#)

Template Parameters

| | |
|----------|--|
| <i>T</i> | Type of the object the capability refers to. |
| <i>U</i> | Type of the source shared capability. |

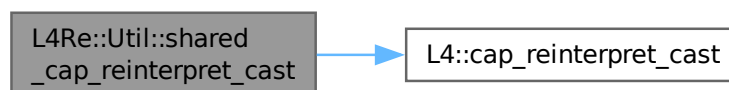
Parameters

| | |
|-------------|---------------------------|
| <i>from</i> | Source shared capability. |
|-------------|---------------------------|

Definition at line 133 of file [shared_cap](#).

References [L4::cap_reinterpret_cast\(\)](#).

Here is the call graph for this function:

**14.12.3.10 shared_cap_reinterpret_cast() [2/2]**

```
template<typename T, typename U>
Shared_cap< T > L4Re::Util::shared_cap_reinterpret_cast (
    Shared_cap< U > const & from) [noexcept]
```

Create a new shared capability by a reinterpret cast from another shared capability.

See also

[L4::cap_reinterpret_cast](#)

Template Parameters

| | |
|----------|--|
| <i>T</i> | Type of the object the capability refers to. |
| <i>U</i> | Type of the source shared capability. |

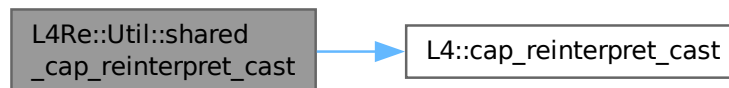
Parameters

| | |
|-------------|---------------------------|
| <i>from</i> | Source shared capability. |
|-------------|---------------------------|

Definition at line 114 of file [shared_cap](#).

References [L4::cap_reinterpret_cast\(\)](#).

Here is the call graph for this function:



14.12.3.11 shared_del_cap_cast() [1/2]

```

template<typename T, typename U>
Shared_del_cap< T > L4Re::Util::shared_del_cap_cast (
    Shared_del_cap< U > && from) [noexcept]
  
```

Create a new shared capability by an explicit cast from another shared capability with move semantics.

Type *U* must be convertible to type *T*. After the call, *from* is empty.

See also

[L4::cap_cast](#)

Template Parameters

| | |
|----------|--|
| <i>T</i> | Type of the object the capability refers to. |
| <i>U</i> | Type of the source shared capability. |

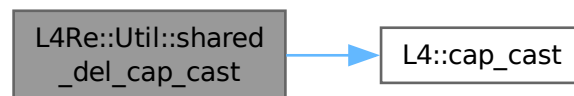
Parameters

| | |
|-------------|---------------------------|
| <i>from</i> | Source shared capability. |
|-------------|---------------------------|

Definition at line 262 of file [shared_cap](#).

References [L4::cap_cast\(\)](#).

Here is the call graph for this function:

**14.12.3.12 shared_del_cap_cast() [2/2]**

```

template<typename T, typename U>
Shared_del_cap< T > L4Re::Util::shared_del_cap_cast (
    Shared_del_cap< U > const & from) [noexcept]
  
```

Create a new shared capability by an explicit cast from another shared capability.

Type `U` must be convertible to type `T`.

See also

[L4::cap_cast](#)

Template Parameters

| | |
|----------|--|
| <i>T</i> | Type of the object the capability refers to. |
| <i>U</i> | Type of the source shared capability. |

Parameters

| | |
|-------------|---------------------------|
| <i>from</i> | Source shared capability. |
|-------------|---------------------------|

Definition at line 243 of file [shared_cap](#).

References [L4::cap_cast\(\)](#).

Here is the call graph for this function:



14.12.3.13 shared_del_cap_dynamic_cast() [1/2]

```

template<typename T, typename U>
Shared_del_cap< T > L4Re::Util::shared_del_cap_dynamic_cast (
    Shared_del_cap< U > && from) [noexcept]
  
```

Create a new shared capability by an dynamic cast from another shared capability with move semantics.

See also

[L4::cap_dynamic_cast](#)

After the call, `from` is empty, unless the [L4::cap_dynamic_cast](#) fails.

Template Parameters

| | |
|----------|--|
| <i>T</i> | Type of the object the capability refers to. |
| <i>U</i> | Type of the source shared capability. |

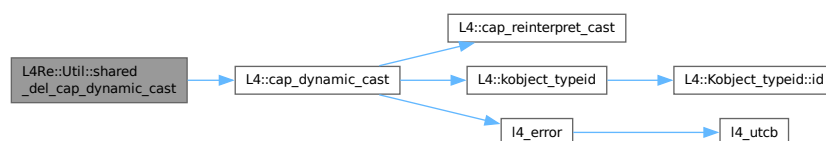
Parameters

| | |
|-------------|---------------------------|
| <i>from</i> | Source shared capability. |
|-------------|---------------------------|

Definition at line 340 of file [shared_cap](#).

References [L4::cap_dynamic_cast](#)().

Here is the call graph for this function:



14.12.3.14 shared_del_cap_dynamic_cast() [2/2]

```
template<typename T, typename U>
Shared_del_cap< T > L4Re::Util::shared_del_cap_dynamic_cast (
    Shared_del_cap< U > const & from) [noexcept]
```

Create a new shared capability by a dynamic cast from another shared capability.

See also

[L4::cap_dynamic_cast](#)

Template Parameters

| | |
|----------|--|
| <i>T</i> | Type of the object the capability refers to. |
| <i>U</i> | Type of the source shared capability. |

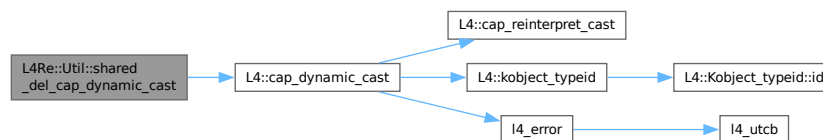
Parameters

| | |
|-------------|---------------------------|
| <i>from</i> | Source shared capability. |
|-------------|---------------------------|

Definition at line 318 of file [shared_cap](#).

References [L4::cap_dynamic_cast\(\)](#).

Here is the call graph for this function:

**14.12.3.15 shared_del_cap_reinterpret_cast() [1/2]**

```
template<typename T, typename U>
Shared_del_cap< T > L4Re::Util::shared_del_cap_reinterpret_cast (
    Shared_del_cap< U > && from) [noexcept]
```

Create a new shared capability by a reinterpret cast from another shared capability with move semantics.

After the call, `from` is empty.

See also

[L4::cap_reinterpret_cast](#)

Template Parameters

| | |
|----------|--|
| <i>T</i> | Type of the object the capability refers to. |
| <i>U</i> | Type of the source shared capability. |

Parameters

| | |
|-------------|---------------------------|
| <i>from</i> | Source shared capability. |
|-------------|---------------------------|

Definition at line 300 of file [shared_cap](#).

References [L4::cap_reinterpret_cast\(\)](#).

Here is the call graph for this function:



14.12.3.16 shared_del_cap_reinterpret_cast() [2/2]

```

template<typename T, typename U>
Shared_del_cap< T > L4Re::Util::shared_del_cap_reinterpret_cast (
    Shared_del_cap< U > const & from) [noexcept]
  
```

Create a new shared capability by a reinterpret cast from another shared capability.

See also

[L4::cap_reinterpret_cast](#)

Template Parameters

| | |
|----------|--|
| <i>T</i> | Type of the object the capability refers to. |
| <i>U</i> | Type of the source shared capability. |

Parameters

| | |
|-------------|---------------------------|
| <i>from</i> | Source shared capability. |
|-------------|---------------------------|

Definition at line 280 of file [shared_cap](#).

References [L4::cap_reinterpret_cast\(\)](#).

Here is the call graph for this function:



14.13 L4Re::Vfs Namespace Reference

Virtual file system for interfaces in POSIX libc.

Data Structures

- class [Be_file](#)
Boiler plate class for implementing an open file for L4Re::Vfs.
- class [Be_file_system](#)
Boilerplate class for implementing a L4Re::Vfs::File_system.
- class [Generic_file](#)
The common interface for an open POSIX file.
- class [Directory](#)
Interface for a POSIX file that is a directory.
- class [Regular_file](#)
Interface for a POSIX file that provides regular file semantics.
- class [Special_file](#)
Interface for a POSIX file that provides special file semantics.
- class [File](#)
The basic interface for an open POSIX file.
- class [Mman](#)
Interface for POSIX memory management.
- class [File_system](#)
Basic interface for an L4Re::Vfs file system.
- class [Fs](#)
POSIX File-system related functionality.
- class [Ops](#)
Interface for the POSIX backends of an application.

Functions

- [L4Re::Vfs::Ops](#) *vfs_ops **asm** ("l4re_env_posix_vfs_ops")
Reference to the applications [L4Re::Vfs::Ops](#) singleton.

14.13.1 Detailed Description

Virtual file system for interfaces in POSIX libc.

14.14 L4vbus Namespace Reference

C++ interface of the [Vbus](#) API.

Data Structures

- class [Pm](#)
Power-management API mixin.
- class [Device](#)
[Device](#) on a [L4vbus::Vbus](#).
- class [Icu](#)
[Vbus](#) Interrupt controller API.
- class [Vbus](#)
The virtual bus ([Vbus](#)) interface.
- class [Gpio_pin](#)
A GPIO pin.
- class [Gpio_module](#)
A [Gpio_module](#) groups multiple GPIO pins together.
- class [Pci_host_bridge](#)
A Pci host bridge.
- class [Pci_dev](#)
A PCI device.

14.14.1 Detailed Description

C++ interface of the [Vbus](#) API.

The virtual bus ([Vbus](#)) is a hierarchical (tree) structure of device nodes where each device has a set of resources attached to it. Each virtual bus provides an [Icu](#) ([Interrupt controller](#)) for interrupt handling.

The [Vbus](#) interface allows a client to find and query devices present on his virtual bus. After obtaining a device handle for a specific device the client can enumerate its resources.

Refer to [L4 Vbus functions](#) for the C API.

Include File

```
#include <l4/vbus/vbus>
```

Include File

```
#include <l4/vbus/vbus_gpio>
```

Include File

```
#include <l4/vbus/vbus_pci>
```

14.15 L4virtio Namespace Reference

L4-VIRTIO Transport C++ API.

Data Structures

- class [Device](#)
IPC interface for virtio over [L4](#) IPC.
- class [Ptr](#)
Pointer used in virtio descriptors.
- class [Virtqueue](#)
Low-level [Virtqueue](#).

14.15.1 Detailed Description

L4-VIRTIO Transport C++ API.

Chapter 15

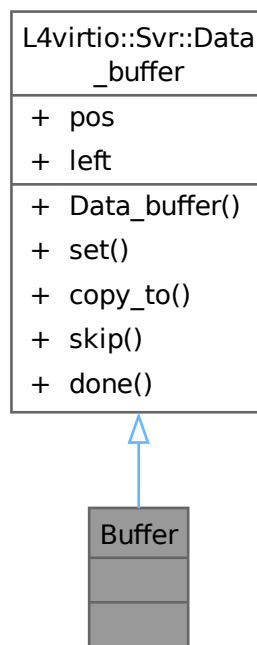
Data Structure Documentation

15.1 Buffer Struct Reference

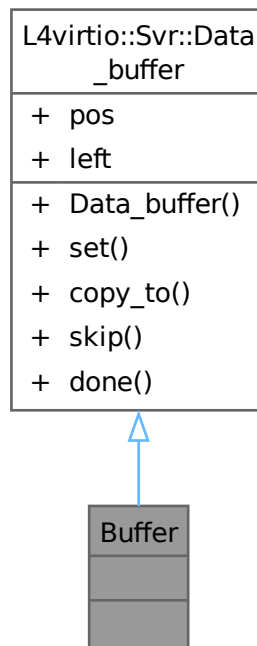
Data buffer used to transfer packets.

```
#include <virtio_net_buffer.h>
```

Inheritance diagram for Buffer:



Collaboration diagram for Buffer:



Additional Inherited Members

Public Member Functions inherited from [L4virtio::Svr::Data_buffer](#)

- `template<typename T>`
`Data_buffer` (`T *p`)
Create buffer for object `p`.
- `template<typename T>`
`void set` (`T *p`)
Set buffer for object `p`.
- `l4_uint32_t copy_to` (`Data_buffer *dst`, `l4_uint32_t max=UINT_MAX`)
Copy contents from this buffer to the destination buffer.
- `l4_uint32_t skip` (`l4_uint32_t bytes`)
Skip given number of bytes in this buffer.
- `bool done` () `const`
Check if there are no more bytes left in the buffer.

Data Fields inherited from [L4virtio::Svr::Data_buffer](#)

- `char * pos`
Current buffer position.
- `l4_uint32_t left`
Bytes left in buffer.

15.1.1 Detailed Description

Data buffer used to transfer packets.

Definition at line 19 of file [virtio_net_buffer.h](#).

The documentation for this struct was generated from the following file:

- `pkg/virtio-net-switch/server/switch/virtio_net_buffer.h`

15.2 `cxx::arith::Ld< V >` Struct Template Reference

Computes the binary logarithm of the given number at compile time.

```
#include <arith>
```

Collaboration diagram for `cxx::arith::Ld< V >`:



15.2.1 Detailed Description

```
template<unsigned long V>
struct cxx::arith::Ld< V >
```

Computes the binary logarithm of the given number at compile time.

Parameters

| | |
|------------|---|
| <i>val</i> | Number whose logarithm to compute, must be greater than zero. |
|------------|---|

Returns

The binary logarithm of `val`.

Definition at line 48 of file [arith](#).

The documentation for this struct was generated from the following file:

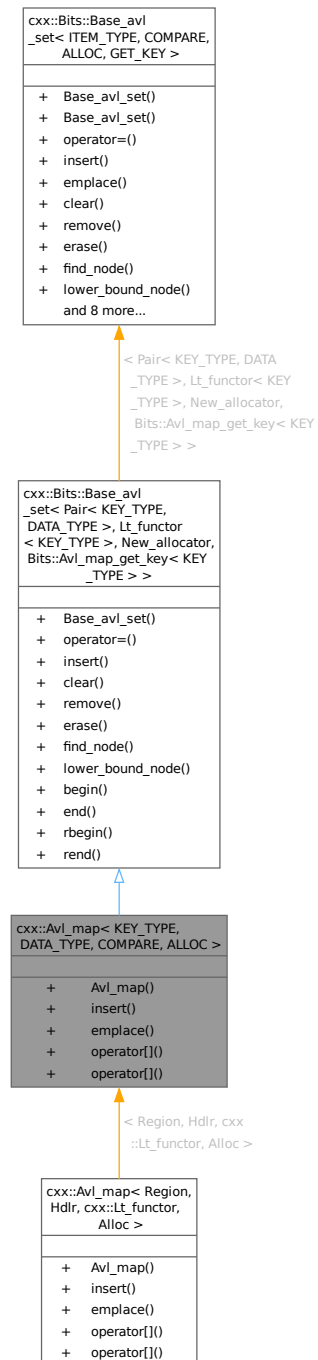
- `l4/cxx/arith`

15.3 cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC > Class Template Reference

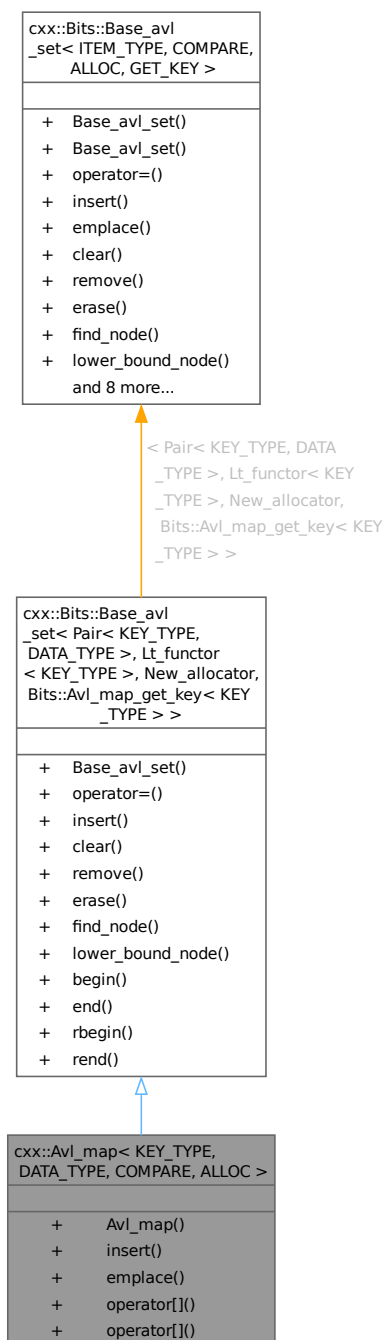
AVL tree based associative container.

```
#include <avl_map>
```

Inheritance diagram for cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >:



Collaboration diagram for `cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >`:



Public Types

- `typedef COMPARE< KEY_TYPE > Key_compare`
Type of the comparison functor.
- `typedef KEY_TYPE Key_type`
Type of the key values.
- `typedef DATA_TYPE Data_type`

Type of the data values.

- typedef Base_type::Node **Node**

Return type for find.

- typedef Base_type::Node_allocator **Node_allocator**

Type of the allocator.

Public Types inherited from

cxx::Bits::Base_avl_set< Pair< KEY_TYPE, DATA_TYPE >, Lt_functor< KEY_TYPE >, New_allocator,

- enum
Return status constants.
- typedef Pair< KEY_TYPE, DATA_TYPE > **Item_type**
Type for the items store in the set.
- typedef Bits::Avl_map_get_key< KEY_TYPE > **Get_key**
Key-getter type to derive the sort key of an internal node.
- typedef Bits::Avl_map_get_key< KEY_TYPE >::Key_type **Key_type**
Type of the sort key used for the items.
- typedef Type_traits< Item_type >::Const_type **Const_item_type**
Type used for const items within the set.
- typedef Lt_functor< KEY_TYPE > **Item_compare**
Type for the comparison functor.
- typedef New_allocator< _Node > **Node_allocator**
Type for the node allocator.
- typedef Avl_set_iter< _Node, Item_type, Fwd > **Iterator**
Forward iterator for the set.
- typedef Avl_set_iter< _Node, Const_item_type, Fwd > **Const_iterator**
Constant forward iterator for the set.
- typedef Avl_set_iter< _Node, Item_type, Rev > **Rev_iterator**
Backward iterator for the set.
- typedef Avl_set_iter< _Node, Const_item_type, Rev > **Const_rev_iterator**
Constant backward iterator for the set.

Public Member Functions

- Avl_map (Node_allocator const &alloc=Node_allocator())
Create an empty AVL-tree based map.
- cxx::Pair< Iterator, int > insert (Key_type const &key, Data_type const &data)
Insert a <key, data> pair into the map.
- template<typename... Args>
cxx::Pair< Iterator, int > emplace (Args &&...args)
Eplace a pair into the map.
- Data_type const & operator[] (Key_type const &key) const
Get the data for the given key.
- Data_type & operator[] (Key_type const &key)
Get or insert data for the given key.

Public Member Functions inherited from**cxx::Bits::Base_avl_set< Pair< KEY_TYPE, DATA_TYPE >, Lt_func< KEY_TYPE >, New_allocator,**

- **Base_avl_set** (**Node_allocator** const &alloc=**Node_allocator**())
Create a AVL-tree based set.
- **Base_avl_set** & **operator=** (**Base_avl_set** const &o)
Copy assignment operator of an AVL-tree based set.
- **cxx::Pair< Iterator, int > insert** (**Item_type** const &item)
Insert an item into the set.
- void **clear** () noexcept
Remove all items from the set.
- int **remove** (**Key_type** const &item)
Remove an item from the set.
- int **erase** (**Key_type** const &item)
Erase the item with the given key.
- **Node find_node** (**Key_type** const &item) const
Lookup a node equal to item.
- **Node lower_bound_node** (**Key_type** const &key) const
Find the first node greater or equal to key.
- **Const_iterator begin** () const
Get the constant forward iterator for the first element in the set.
- **Const_iterator end** () const
Get the end marker for the constant forward iterator.
- **Const_rev_iterator rbegin** () const
Get the constant backward iterator for the last element in the set.
- **Const_rev_iterator rend** () const
Get the end marker for the constant backward iterator.

15.3.1 Detailed Description

```
template<typename KEY_TYPE, typename DATA_TYPE, template< typename A > class COMPARE = Lt_↵
func<, template< typename B > class ALLOC = New_allocator>
class cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >
```

AVL tree based associative container.

Template Parameters

| | |
|------------------|---|
| <i>KEY_TYPE</i> | Type of the key values. |
| <i>DATA_TYPE</i> | Type of the data values. |
| <i>COMPARE</i> | Type comparison functor for the key values. |
| <i>ALLOC</i> | Type of the allocator used for the nodes. |

Definition at line 45 of file [avl_map](#).

15.3.2 Constructor & Destructor Documentation

15.3.2.1 Avl_map()

```
template<typename KEY_TYPE, typename DATA_TYPE, template< typename A > class COMPARE = Lt_↔
functor, template< typename B > class ALLOC = New_allocator>
cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >::Avl_map (
    Node_allocator const & alloc = Node_allocator()) [inline]
```

Create an empty AVL-tree based map.

Parameters

| | |
|--------------|---------------------|
| <i>alloc</i> | The node allocator. |
|--------------|---------------------|

Definition at line 80 of file [avl_map](#).

15.3.3 Member Function Documentation

15.3.3.1 emplace()

```
template<typename KEY_TYPE, typename DATA_TYPE, template< typename A > class COMPARE = Lt_↔
functor, template< typename B > class ALLOC = New_allocator>
template<typename... Args>
cxx::Pair< Iterator, int > cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >::emplace (
    Args &&... args) [inline]
```

Emplace a pair into the map.

Parameters

| | |
|-------------|--|
| <i>args</i> | The cxx::Pair constructor arguments for key & value. |
|-------------|--|

Returns

A pair of iterator (*first*) and return value (*second*). *second* will be 0 if the element was inserted into the set and `−#E_exist` if the key was already in the set and the set was therefore not updated. In both cases, *first* contains an iterator that points to the element. *second* may also be `−#E_nomem` when memory for the new node could not be allocated. *first* is then invalid.

Definition at line 117 of file [avl_map](#).

15.3.3.2 insert()

```
template<typename KEY_TYPE, typename DATA_TYPE, template< typename A > class COMPARE = Lt_↵
functor, template< typename B > class ALLOC = New_allocator>
cxx::Pair< Iterator, int > cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >::insert (
    Key_type const & key,
    Data_type const & data) [inline]
```

Insert a <key, data> pair into the map.

Parameters

| | |
|-------------|---------------------------|
| <i>key</i> | The key value. |
| <i>data</i> | The data value to insert. |

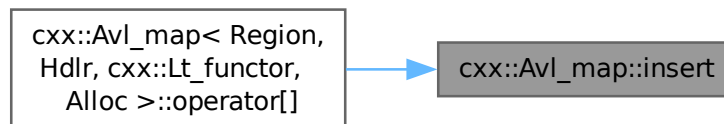
Returns

A pair of iterator (*first*) and return value (*second*). *second* will be 0 if the element was inserted into the set and `-#E_exist` if the key was already in the set and the set was therefore not updated. In both cases, *first* contains an iterator that points to the element. *second* may also be `-#E_nomem` when memory for the new node could not be allocated. *first* is then invalid.

Definition at line 99 of file [avl_map](#).

Referenced by [cxx::Avl_map< Region, Hdlr, cxx::Lt_functor, Alloc >::operator\[\]\(\)](#).

Here is the caller graph for this function:



15.3.3.3 operator[]() [1/2]

```
template<typename KEY_TYPE, typename DATA_TYPE, template< typename A > class COMPARE = Lt_↵
functor, template< typename B > class ALLOC = New_allocator>
Data_type & cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >::operator[] (
    Key_type const & key) [inline]
```

Get or insert data for the given key.

Parameters

| | |
|------------|----------------------------------|
| <i>key</i> | The key value to use for lookup. |
|------------|----------------------------------|

Returns

If the item already exists, a reference to the data item. Otherwise a new data item is default-constructed and inserted under the given key before a reference is returned.

Definition at line 137 of file [avl_map](#).

15.3.3.4 operator[]() [2/2]

```
template<typename KEY_TYPE, typename DATA_TYPE, template< typename A > class COMPARE = Lt_↔
functor, template< typename B > class ALLOC = New_allocator>
Data_type const & cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >::operator[] (
    Key_type const & key) const [inline]
```

Get the data for the given key.

Parameters

| | |
|------------|----------------------------------|
| <i>key</i> | The key value to use for lookup. |
|------------|----------------------------------|

Precondition

A <key, data> pair for the given key value must exist.

Definition at line 125 of file [avl_map](#).

The documentation for this class was generated from the following file:

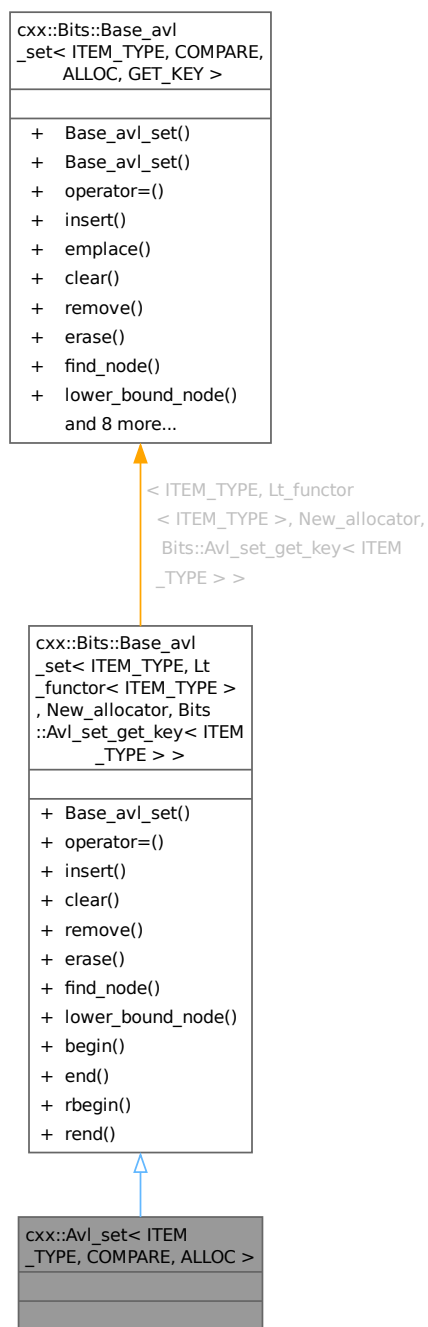
- I4/cxx/[avl_map](#)

15.4 cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC > Class Template Reference

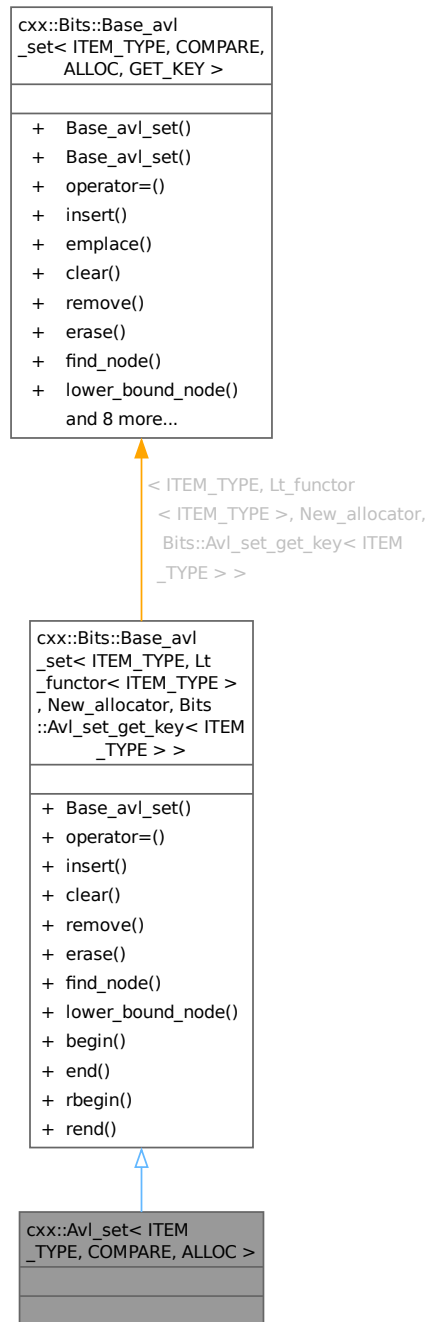
AVL set for simple comparable items.

```
#include <avl_set>
```

Inheritance diagram for cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC >:



Collaboration diagram for `cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC >`:



Additional Inherited Members

Public Types inherited from

`cxx::Bits::Base_avl_set< ITEM_TYPE, Lt_functor< ITEM_TYPE >, New_allocator, Bits::Avl_set_get_key< ITEM_TYPE > >`

- enum

Return status constants.

- typedef `ITEM_TYPE` **Item_type**
Type for the items store in the set.
- typedef `Bits::Avl_set_get_key< ITEM_TYPE >` **Get_key**
Key-getter type to derive the sort key of an internal node.
- typedef `Bits::Avl_set_get_key< ITEM_TYPE >::Key_type` **Key_type**
Type of the sort key used for the items.
- typedef `Type_traits< Item_type >::Const_type` **Const_item_type**
Type used for const items within the set.
- typedef `Lt_func< ITEM_TYPE >` **Item_compare**
Type for the comparison functor.
- typedef `New_allocator< _Node >` **Node_allocator**
Type for the node allocator.
- typedef `Avl_set_iter< _Node, Item_type, Fwd >` **Iterator**
Forward iterator for the set.
- typedef `Avl_set_iter< _Node, Const_item_type, Fwd >` **Const_iterator**
Constant forward iterator for the set.
- typedef `Avl_set_iter< _Node, Item_type, Rev >` **Rev_iterator**
Backward iterator for the set.
- typedef `Avl_set_iter< _Node, Const_item_type, Rev >` **Const_rev_iterator**
Constant backward iterator for the set.

Public Member Functions inherited from

`cxx::Bits::Base_avl_set< ITEM_TYPE, Lt_func< ITEM_TYPE >, New_allocator, Bits::Avl_set_get_key`

- `Base_avl_set` (`Node_allocator` const &alloc=`Node_allocator`())
Create a AVL-tree based set.
- `Base_avl_set` & `operator=` (`Base_avl_set` const &o)
Copy assignment operator of an AVL-tree based set.
- `cxx::Pair< Iterator, int >` `insert` (`Item_type` const &item)
Insert an item into the set.
- void **clear** () noexcept
Remove all items from the set.
- int `remove` (`Key_type` const &item)
Remove an item from the set.
- int `erase` (`Key_type` const &item)
Erase the item with the given key.
- Node `find_node` (`Key_type` const &item) const
Lookup a node equal to `item`.
- Node `lower_bound_node` (`Key_type` const &key) const
Find the first node greater or equal to `key`.
- `Const_iterator` `begin` () const
Get the constant forward iterator for the first element in the set.
- `Const_iterator` `end` () const
Get the end marker for the constant forward iterator.
- `Const_rev_iterator` `rbegin` () const
Get the constant backward iterator for the last element in the set.
- `Const_rev_iterator` `rend` () const
Get the end marker for the constant backward iterator.

15.4.1 Detailed Description

```
template<typename ITEM_TYPE, template< typename T > class COMPARE = Lt_functor, template< type-  
name A > class ALLOC = New_allocator>  
class cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC >
```

AVL set for simple comparable items.

The AVL set can store any kind of items where a partial order is defined. The default relation is defined by the '<' operator.

Template Parameters

| | |
|------------------|---|
| <i>ITEM_TYPE</i> | The type of the items to be stored in the set. |
| <i>COMPARE</i> | The relation to define the partial order, default is to use operator '<'. |
| <i>ALLOC</i> | The allocator to use for the nodes of the AVL set. |

Definition at line 538 of file [avl_set](#).

The documentation for this class was generated from the following file:

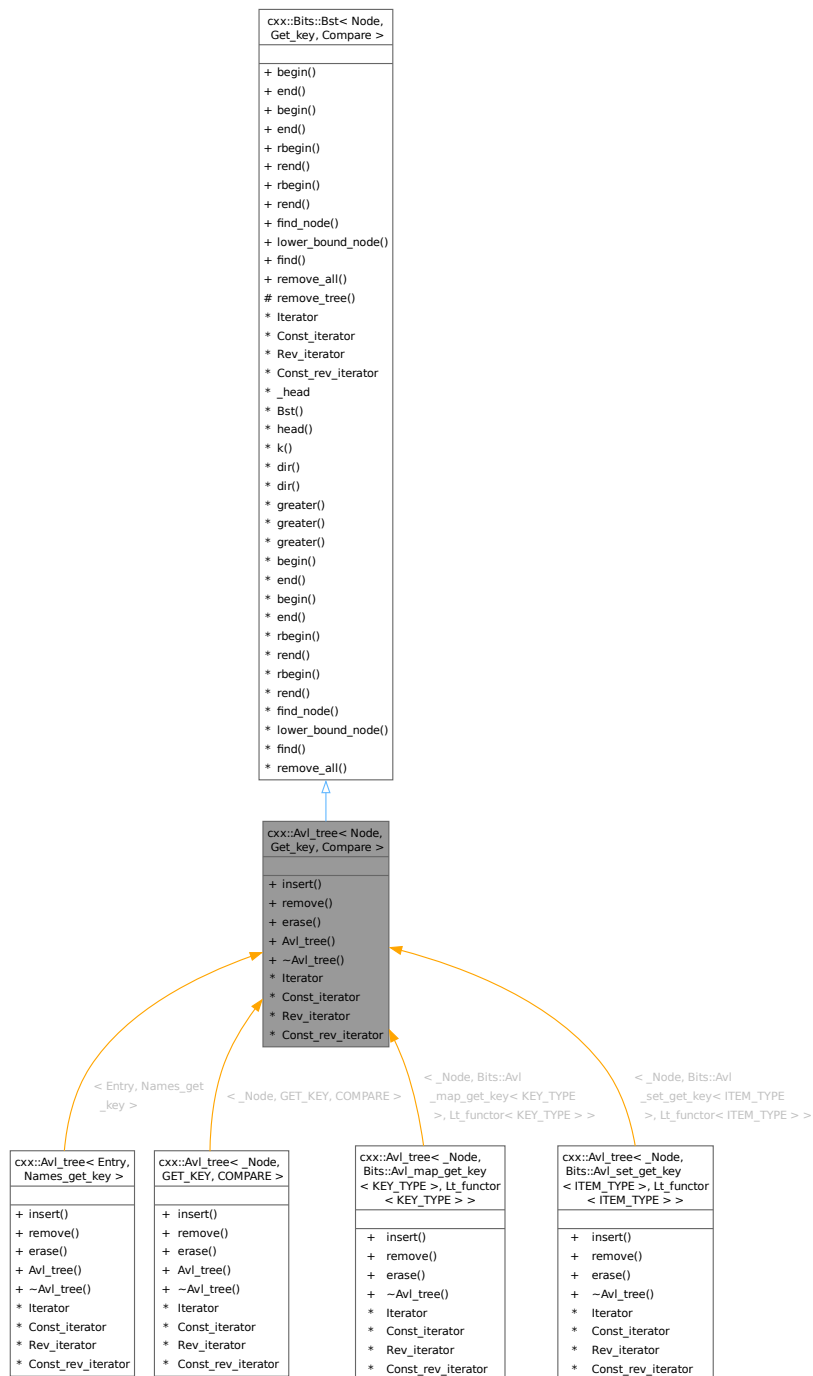
- [l4/cxx/avl_set](#)

15.5 cxx::Avl_tree< Node, Get_key, Compare > Class Template Reference

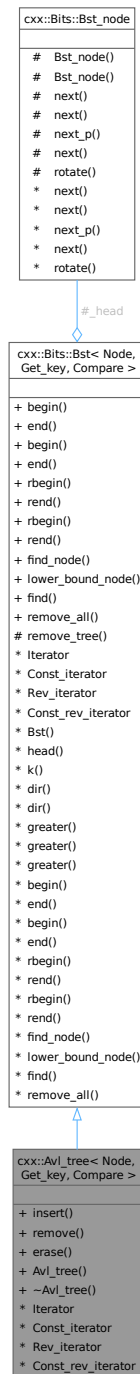
A generic AVL tree.

```
#include <avl_tree>
```

Inheritance diagram for cxx::Avl_tree< Node, Get_key, Compare >:



Collaboration diagram for `cxx::Avl_tree< Node, Get_key, Compare >`:



Public Types

- typedef `Bst::Iterator` `Iterator`

- typedef [Bst::Const_iterator](#) **Const_iterator**
Constant forward iterator for the tree.
- typedef [Bst::Rev_iterator](#) **Rev_iterator**
Backward iterator for the tree.
- typedef [Bst::Const_rev_iterator](#) **Const_rev_iterator**
Constant backward iterator for the tree.

Public Types inherited from [cxx::Bits::Bst< Node, Get_key, Compare >](#)

- typedef [Get_key::Key_type](#) **Key_type**
The type of key values used to generate the total order of the elements.
- typedef [Type_traits< \[Key_type\]\(#\) >::Param_type](#) **Key_param_type**
The type for key parameters.
- typedef [Fwd](#) **Fwd_iter_ops**
Helper for building forward iterators for different wrapper classes.
- typedef [Rev](#) **Rev_iter_ops**
Helper for building reverse iterators for different wrapper classes.
- typedef [__Bst_iter< Node, Node, Fwd >](#) **Iterator**
Forward iterator.
- typedef [__Bst_iter< Node, Node const, Fwd >](#) **Const_iterator**
Constant forward iterator.
- typedef [__Bst_iter< Node, Node, Rev >](#) **Rev_iterator**
Backward iterator.
- typedef [__Bst_iter< Node, Node const, Rev >](#) **Const_rev_iterator**
Constant backward.

Public Member Functions

- [Pair< Node *, bool >](#) [insert](#) (Node *new_node)
Insert a new node into this AVL tree.
- Node * [remove](#) (Key_param_type key)
Remove the node with key from the tree.
- Node * [erase](#) (Key_param_type key)
An alias for [remove\(\)](#).
- [Avl_tree](#) ()=default
Create an empty AVL tree.
- [~Avl_tree](#) () noexcept
Destroy the tree.

Public Member Functions inherited from `cxx::Bits::Bst< Node, Get_key, Compare >`

- `Const_iterator begin ()` const
Get the constant forward iterator for the first element in the set.
- `Const_iterator end ()` const
Get the end marker for the constant forward iterator.
- `Iterator begin ()`
Get the mutable forward iterator for the first element of the set.
- `Iterator end ()`
Get the end marker for the mutable forward iterator.
- `Const_rev_iterator rbegin ()` const
Get the constant backward iterator for the last element in the set.
- `Const_rev_iterator rend ()` const
Get the end marker for the constant backward iterator.
- `Rev_iterator rbegin ()`
Get the mutable backward iterator for the last element of the set.
- `Rev_iterator rend ()`
Get the end marker for the mutable backward iterator.

- `Node * find_node (Key_param_type key)` const
find the node with the given key.
- `Node * lower_bound_node (Key_param_type key)` const
Find the first node with a key not less than the given `key`.
- `Const_iterator find (Key_param_type key)` const
find the node with the given key.
- `template<typename FUNC>`
`void remove_all (FUNC &&callback)`
Clear the tree.

Additional Inherited Members

- `Bst ()`
Create an empty tree.

- `Node * head ()` const
Access the head node as object of type `Node`.

Static Protected Member Functions inherited from `cxx::Bits::Bst< Node, Get_key, Compare >`

- `template<typename FUNC>`
`static void remove_tree (Bst_node *head, FUNC &&callback)`
Remove all elements in the subtree of head.

- `static Key_type k (Bst_node const *n)`
Get the key value of `n`.

- `static Dir dir (Key_param_type l, Key_param_type r)`

Get the direction to go from l to search for r .

- static Dir [dir](#) ([Key_param_type](#) l , [Bst_node](#) const $*r$)

Get the direction to go from l to search for r .

- static bool **greater** ([Key_param_type](#) l , [Key_param_type](#) r)

Is l greater than r .

- static bool **greater** ([Key_param_type](#) l , [Bst_node](#) const $*r$)

Is l greater than r .

- static bool **greater** ([Bst_node](#) const $*l$, [Bst_node](#) const $*r$)

Is l greater than r .

- [Bst_node](#) * **_head**

The head pointer of the tree.

15.5.1 Detailed Description

```
template<typename Node, typename Get_key, typename Compare = Lt_functor<typename Get_key::Key↵
_type>>
class cxx::Avl_tree< Node, Get_key, Compare >
```

A generic AVL tree.

Template Parameters

| | |
|----------------|---|
| <i>Node</i> | The data type of the nodes (must inherit from Avl_tree_node). |
| <i>Get_key</i> | The meta function to get the key value from a node. The implementation uses <code>Get_key::key↵_of(ptr_to_node)</code> . The type of the key values must be defined in <code>Get_key::Key_type</code> . |
| <i>Compare</i> | Binary relation to establish a total order for the nodes of the tree. <code>Compare() (l, r)</code> must return true if the key l is smaller than the key r . |

This implementation does not provide any memory management. It is the responsibility of the caller to allocate nodes before inserting them and to free them when they are removed or when the tree is destroyed. Conversely, the caller must also ensure that nodes are removed from the tree before they are destroyed.

Examples

[tmpfs/lib/src/fs.cc](#).

Definition at line 100 of file [avl_tree](#).

15.5.2 Member Typedef Documentation

15.5.2.1 Iterator

```
template<typename Node, typename Get_key, typename Compare = Lt_functor<typename Get_key::←
Key_type>>>
typedef Bst::Iterator cxx::Avl_tree< Node, Get_key, Compare >::Iterator
```

Forward iterator for the tree.

Definition at line 130 of file [avl_tree](#).

15.5.3 Member Function Documentation

15.5.3.1 insert()

```
template<typename Node, typename Get_key, class Compare>
Pair< Node *, bool > cxx::Avl_tree< Node, Get_key, Compare >::insert (
    Node * new_node)
```

Insert a new node into this AVL tree.

Parameters

| | |
|-----------------|----------------------------|
| <i>new_node</i> | A pointer to the new node. |
|-----------------|----------------------------|

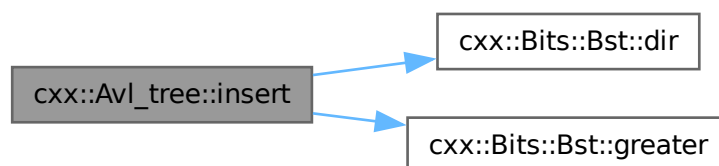
Returns

A pair, with *second* set to `true` and *first* pointing to *new_node*, on success. If there is already a node with the same key then *first* points to this node and *second* is 'false'.

Definition at line 220 of file [avl_tree](#).

References [cxx::Bits::Bst< Node, Get_key, Compare >::dir\(\)](#), [cxx::Bits::Bst< Node, Get_key, Compare >::greater\(\)](#), and [cxx::Bits::Direction::N](#).

Here is the call graph for this function:



15.5.3.2 remove()

```
template<typename Node, typename Get_key, class Compare>
Node * cxx::Avl_tree< Node, Get_key, Compare >::remove (
    Key_param_type key) [inline]
```

Remove the node with *key* from the tree.

Parameters

| | |
|------------|--------------------------------|
| <i>key</i> | The key to the node to remove. |
|------------|--------------------------------|

Returns

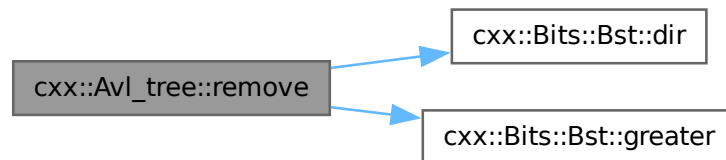
The pointer to the removed node on success, or 0 if no node with the *key* exists.

Definition at line 282 of file [avl_tree](#).

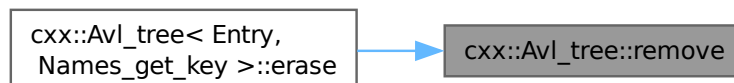
References [cxx::Bits::Bst< Node, Get_key, Compare >::dir\(\)](#), [cxx::Bits::Bst< Node, Get_key, Compare >::greater\(\)](#), [cxx::Bits::Direction::L](#), and [cxx::Bits::Direction::N](#).

Referenced by [cxx::Avl_tree< Entry, Names_get_key >::erase\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

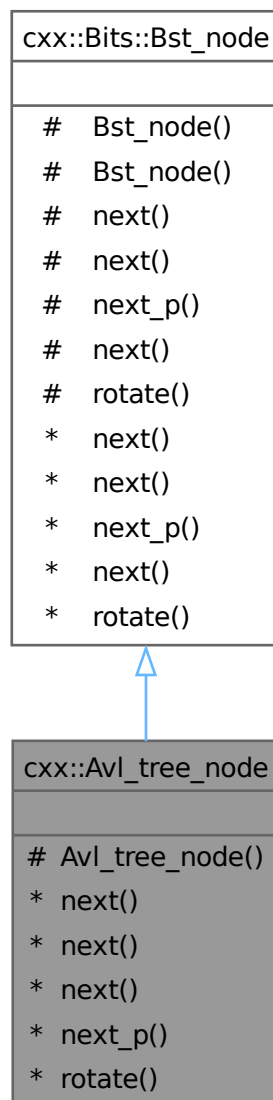
- [I4/cxx/avl_tree](#)

15.6 cxx::Avl_tree_node Class Reference

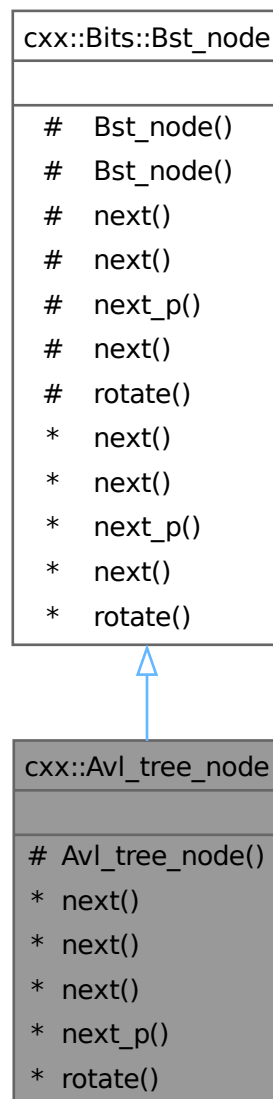
Node of an AVL tree.

```
#include <avl_tree>
```

Inheritance diagram for cxx::Avl_tree_node:



Collaboration diagram for cxx::Avl_tree_node:



Protected Member Functions

- **Avl_tree_node ()**=default
Create an uninitialized node, this is what you should do.

Protected Member Functions inherited from [cxx::Bits::Bst_node](#)

- **Bst_node ()**
Create uninitialized node.
- **Bst_node (bool)**
Create initialized node.

Additional Inherited Members

Static Protected Member Functions inherited from [cxx::Bits::Bst_node](#)

- static [Bst_node](#) * **next** ([Bst_node](#) const *p, [Direction](#) d)
Get next node in direction d.
- static void **next** ([Bst_node](#) *p, [Direction](#) d, [Bst_node](#) *n)
Set next node of p in direction d to n.
- static [Bst_node](#) ** **next_p** ([Bst_node](#) *p, [Direction](#) d)
Get pointer to link in direction d.
- template<typename Node>
static Node * **next** ([Bst_node](#) const *p, [Direction](#) d)
Get next node in direction d as type Node.
- static void **rotate** ([Bst_node](#) **t, [Direction](#) idir)
Rotate subtree t in the opposite direction of idir.

15.6.1 Detailed Description

Node of an AVL tree.

Examples

[tmpfs/lib/src/fs.cc](#).

Definition at line 29 of file [avl_tree](#).

The documentation for this class was generated from the following file:

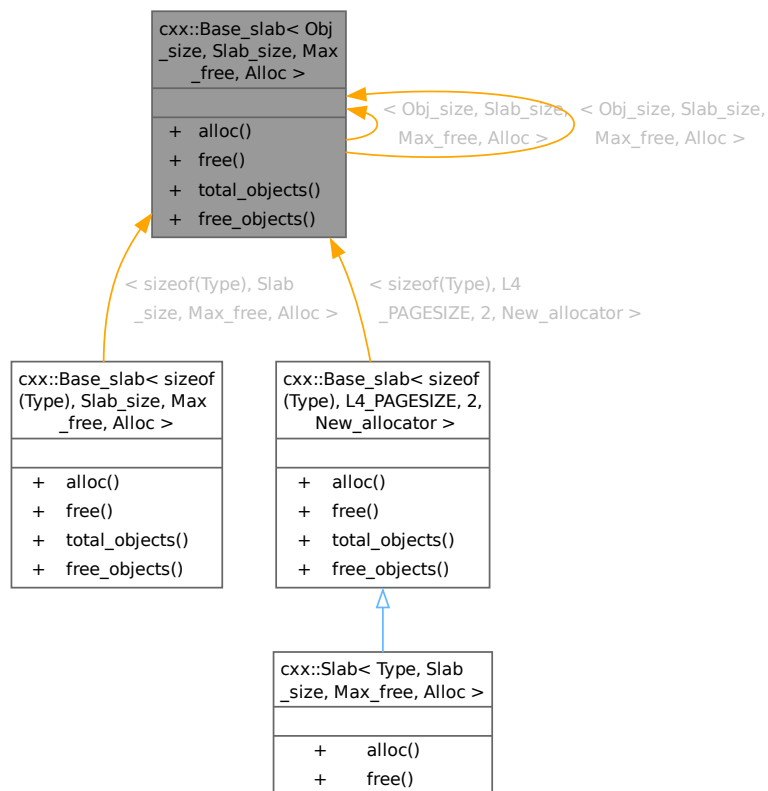
- [l4/cxx/avl_tree](#)

15.7 [cxx::Base_slab](#)< [Obj_size](#), [Slab_size](#), [Max_free](#), [Alloc](#) > Class Template Reference

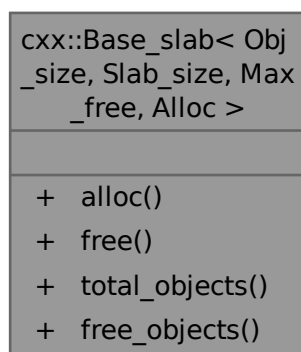
Basic slab allocator.

```
#include <slab_alloc>
```

Inheritance diagram for cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >:



Collaboration diagram for cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >:



Data Structures

- struct [Slab_i](#)

Type of a slab.

Public Types

- enum { [object_size](#) = Obj_size , [slab_size](#) = Slab_size , [objects_per_slab](#) = (Slab_size - sizeof(Slab_head)) / object_size , [max_free_slabs](#) = Max_free }
- typedef Alloc< [Slab_i](#) > **Slab_alloc**
Type of the backend allocator.

Public Member Functions

- void * [alloc](#) () noexcept
Allocate a new object.
- void [free](#) (void * _o) noexcept
Free the given object (_o).
- unsigned [total_objects](#) () const noexcept
Get the total number of objects managed by the slab allocator.
- unsigned [free_objects](#) () const noexcept
Get the number of objects which can be allocated before a new empty slab needs to be added to the slab allocator.

15.7.1 Detailed Description

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A > class Alloc
= New_allocator>
class cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >
```

Basic slab allocator.

Template Parameters

| | |
|------------------|--|
| <i>Obj_size</i> | The size of the objects managed by the allocator (in bytes). |
| <i>Slab_size</i> | The size of a slab (in bytes). |
| <i>Max_free</i> | The maximum number of free slabs. When this limit is reached slabs are freed, provided that the backend allocator supports allocated memory to be freed. |
| <i>Alloc</i> | The backend allocator used to allocate slabs. |

Definition at line 31 of file [slab_alloc](#).

15.7.2 Member Enumeration Documentation

15.7.2.1 anonymous enum

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
anonymous enum
```

Enumerator

| | |
|-------------------------------|-------------------------------|
| <code>object_size</code> | Size of an object. |
| <code>slab_size</code> | Size of a slab. |
| <code>objects_per_slab</code> | Objects per slab. |
| <code>max_free_slabs</code> | Maximum number of free slabs. |

Definition at line 65 of file [slab_alloc](#).

15.7.3 Member Function Documentation

15.7.3.1 `alloc()`

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
void * cxx::Base\_slab< Obj_size, Slab_size, Max_free, Alloc >::alloc () [inline], [noexcept]
```

Allocate a new object.

Returns

A pointer to the new object if the allocation succeeds, or 0 on failure to acquire memory from the backend allocator when the slab cache memory is already exhausted.

Note

The user is responsible for initializing the object.

Definition at line 207 of file [slab_alloc](#).

15.7.3.2 `free()`

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
void cxx::Base\_slab< Obj_size, Slab_size, Max_free, Alloc >::free (
    void * _o) [inline], [noexcept]
```

Free the given object (`_o`).

Precondition

The object must have been allocated with this allocator.

Definition at line 246 of file [slab_alloc](#).

15.7.3.3 free_objects()

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
unsigned cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::free_objects () const [inline],
[noexcept]
```

Get the number of objects which can be allocated before a new empty slab needs to be added to the slab allocator.

Returns

The number of free objects in the slab allocator.

Definition at line 308 of file [slab_alloc](#).

15.7.3.4 total_objects()

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
unsigned cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::total_objects () const [inline],
[noexcept]
```

Get the total number of objects managed by the slab allocator.

Returns

The number of objects managed by the allocator (including the free objects).

Definition at line 299 of file [slab_alloc](#).

The documentation for this class was generated from the following file:

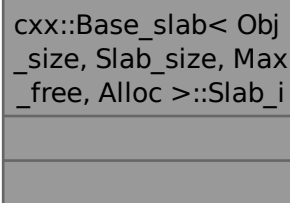
- l4/cxx/slab_alloc

15.8 cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::Slab_i Struct Reference

Type of a slab.

```
#include <slab_alloc>
```

Collaboration diagram for cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::Slab_i:



15.8.1 Detailed Description

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A > class Alloc
= New_allocator>
struct cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >::Slab_i
```

Type of a slab.

Definition at line 86 of file [slab_alloc](#).

The documentation for this struct was generated from the following file:

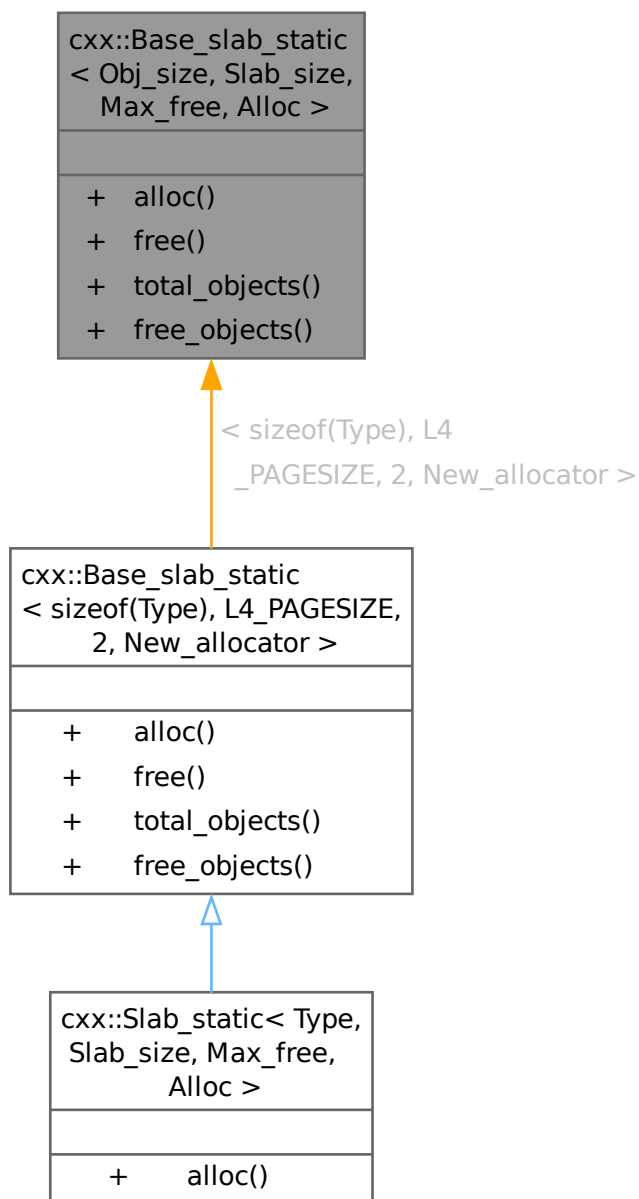
- `l4/cxx/slab_alloc`

15.9 `cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >` Class Template Reference

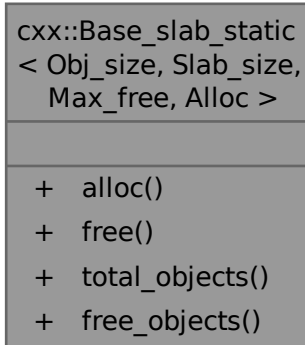
Merged slab allocator (allocators for objects of the same size are merged together).

```
#include <slab_alloc>
```

Inheritance diagram for `cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >`:



Collaboration diagram for cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >:



Public Types

- enum { [object_size](#) = Obj_size , [slab_size](#) = Slab_size , [objects_per_slab](#) = _A::objects_per_slab , [max_free_slabs](#) = Max_free }

Public Member Functions

- void * [alloc](#) () noexcept
Allocate an object.
- void [free](#) (void *p) noexcept
Free the given object (p).
- unsigned [total_objects](#) () const noexcept
Get the total number of objects managed by the slab allocator.
- unsigned [free_objects](#) () const noexcept
Get the number of free objects in the slab allocator.

15.9.1 Detailed Description

```

template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A > class Alloc
= New_allocator>
class cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >
  
```

Merged slab allocator (allocators for objects of the same size are merged together).

Template Parameters

| | |
|------------------|--|
| <i>Obj_size</i> | The size of an object managed by the slab allocator. |
| <i>Slab_size</i> | The size of a slab. |

| | |
|-----------------|-----------------------------------|
| <i>Max_free</i> | The maximum number of free slabs. |
| <i>Alloc</i> | The allocator for the slabs. |

This slab allocator class is useful for merging slab allocators with the same parameters (equal `Obj_size`, `Slab_size`, `Max_free`, and `Alloc` parameters) together and share the overhead for the slab caches among all equal-sized objects.

Definition at line 388 of file [slab_alloc](#).

15.9.2 Member Enumeration Documentation

15.9.2.1 anonymous enum

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
anonymous enum
```

Enumerator

| | |
|-------------------------------|-------------------------------|
| <code>object_size</code> | Size of an object. |
| <code>slab_size</code> | Size of a slab. |
| <code>objects_per_slab</code> | Number of objects per slab. |
| <code>max_free_slabs</code> | Maximum number of free slabs. |

Definition at line 395 of file [slab_alloc](#).

15.9.3 Member Function Documentation

15.9.3.1 `alloc()`

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
void * cxx::Base\_slab\_static< Obj_size, Slab_size, Max_free, Alloc >::alloc () [inline],
[noexcept]
```

Allocate an object.

Note

The user is responsible for initializing the object.

Definition at line 412 of file [slab_alloc](#).

15.9.3.2 free()

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
void cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >::free (
    void * p) [inline], [noexcept]
```

Free the given object (p).

Parameters

| | |
|----------|------------------------------------|
| <i>p</i> | The pointer to the object to free. |
|----------|------------------------------------|

Precondition

p must be a pointer to an object allocated by this allocator.

Definition at line 420 of file [slab_alloc](#).

15.9.3.3 free_objects()

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
unsigned cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >::free_objects () const
[inline], [noexcept]
```

Get the number of free objects in the slab allocator.

Returns

The number of free objects in all free and partially used slabs managed by this allocator.

Note

The value is the merged value for all equal parameterized [Base_slab_static](#) instances.

Definition at line 440 of file [slab_alloc](#).

15.9.3.4 total_objects()

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
unsigned cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >::total_objects () const
[inline], [noexcept]
```

Get the total number of objects managed by the slab allocator.

Returns

The number of objects managed by the allocator (including the free objects).

Note

The value is the merged value for all equal parameterized [Base_slab_static](#) instances.

Definition at line 430 of file [slab_alloc](#).

The documentation for this class was generated from the following file:

- I4/cxx/slab_alloc

15.10 `cxx::Bitfield< T, LSB, MSB >` Class Template Reference

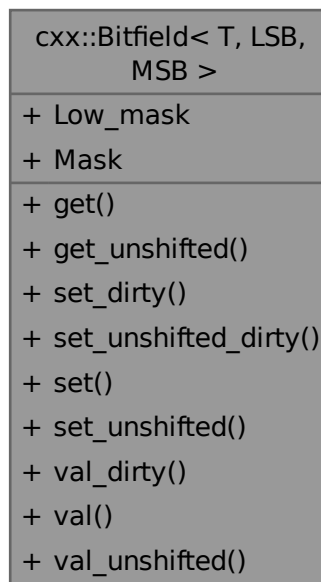
Definition for a member (part) of a bit field.

```
#include <bitfield>
```

Inheritance diagram for `cxx::Bitfield< T, LSB, MSB >`:



Collaboration diagram for `cxx::Bitfield< T, LSB, MSB >`:



Data Structures

- class [Value_base](#)
Internal helper type.
- class [Value](#)
Internal helper type.
- class [Value_unshifted](#)
Internal helper type.

Public Types

- enum { **Bits** = MSB + 1 - LSB , **Lsb** = LSB , **Msb** = MSB }
- typedef Best_type< **Bits** >::Type **Bits_type**
*Type to hold at least **Bits** bits.*
- typedef Best_type< **Bits**+**Lsb** >::Type **Shift_type**
*Type to hold at least **Bits** + **Lsb** bits.*
- typedef **Value**< Base_type & > **Ref**
Reference type to access the bits inside a raw bit field.
- typedef **Value**< Base_type volatile & > **Ref_volatile**
Volatile reference type to access the bits inside a raw bit field.
- typedef **Value**< Base_type const > **Val**
***Value** type to access the bits inside a raw bit field.*
- typedef **Value_unshifted**< Base_type & > **Ref_unshifted**
Reference type to access the bits inside a raw bit field (in place).
- typedef **Value_unshifted**< Base_type volatile & > **Ref_unshifted_volatile**
Volatile reference type to access the bits inside a raw bit field (in place).
- typedef **Value_unshifted**< Base_type const > **Val_unshifted**
***Value** type to access the bits inside a raw bit field (in place).*

Static Public Member Functions

- static constexpr **Bits_type** get (Shift_type val)
*Get the bits out of **val**.*
- static constexpr Base_type get_unshifted (Shift_type val)
*Get the bits in place out of **val**.*
- static constexpr Base_type set_dirty (Base_type dest, Shift_type val)
*Set the bits corresponding to **val**.*
- static constexpr Base_type set_unshifted_dirty (Base_type dest, Shift_type val)
*Set the bits corresponding to **val**.*
- static Base_type set (Base_type dest, **Bits_type** val)
*Set the bits corresponding to **val**.*
- static Base_type set_unshifted (Base_type dest, Shift_type val)
*Set the bits corresponding to **val**.*
- static constexpr Base_type val_dirty (Shift_type val)
*Get the shifted bits for **val**.*
- static constexpr Base_type val (**Bits_type** val)
*Get the shifted bits for **val**.*
- static constexpr Base_type val_unshifted (Shift_type val)
*Get the shifted bits for **val**.*

Static Public Attributes

- static constexpr Base_type **Low_mask** = Base_type(~0ULL) >> (sizeof(Base_type) * 8 - **Bits**)
*Mask value to get **Bits** bits.*
- static constexpr Base_type **Mask** = **Low_mask** << **Lsb**
*Mask value to the bits out of a **T**.*

15.10.1 Detailed Description

```
template<typename T, unsigned LSB, unsigned MSB>
class cxx::Bitfield< T, LSB, MSB >
```

Definition for a member (part) of a bit field.

Parameters

| | |
|------------|--|
| <i>T</i> | The underlying type of the bit field. |
| <i>LSB</i> | The least significant bit of our bits. |
| <i>MSB</i> | The most significant bit of our bits. |

Definition at line 24 of file [bitfield](#).

15.10.2 Member Typedef Documentation

15.10.2.1 Bits_type

```
template<typename T, unsigned LSB, unsigned MSB>
typedef Best_type<Bits>::Type cxx::Bitfield< T, LSB, MSB >::Bits_type
```

Type to hold at least [Bits](#) bits.

This type can handle all values that can be stored in this part of the bit field.

Definition at line 71 of file [bitfield](#).

15.10.2.2 Shift_type

```
template<typename T, unsigned LSB, unsigned MSB>
typedef Best_type<Bits+Lsb>::Type cxx::Bitfield< T, LSB, MSB >::Shift_type
```

Type to hold at least [Bits](#) + [Lsb](#) bits.

This type can handle all values that can be stored in this part of the bit field when they are at the target location ([Lsb](#) bits shifted to the left).

Definition at line 79 of file [bitfield](#).

15.10.3 Member Enumeration Documentation

15.10.3.1 anonymous enum

```
template<typename T, unsigned LSB, unsigned MSB>
anonymous enum
```

Enumerator

| | |
|------|------------------|
| Bits | Number of bits. |
| Lsb | index of the LSB |
| Msb | index of the MSB |

Definition at line 52 of file [bitfield](#).

15.10.4 Member Function Documentation

15.10.4.1 `get()`

```
template<typename T, unsigned LSB, unsigned MSB>
constexpr Bits_type cxx::Bitfield< T, LSB, MSB >::get (
    Shift_type val) [inline], [static], [constexpr]
```

Get the bits out of `val`.

Parameters

| | |
|------------------|---------------------------------------|
| <code>val</code> | The raw value of the whole bit field. |
|------------------|---------------------------------------|

Returns

The bits from `Lsb` to `Msb` shifted to the right.

Definition at line 96 of file [bitfield](#).

15.10.4.2 `get_unshifted()`

```
template<typename T, unsigned LSB, unsigned MSB>
constexpr Base_type cxx::Bitfield< T, LSB, MSB >::get_unshifted (
    Shift_type val) [inline], [static], [constexpr]
```

Get the bits in place out of `val`.

Parameters

| | |
|------------------|---------------------------------------|
| <code>val</code> | The raw value of the whole bit field. |
|------------------|---------------------------------------|

Returns

The bits from `Lsb` to `Msb` (unshifted).

This means other bits are masked out, however the result is not shifted to the right.

Definition at line 109 of file [bitfield](#).

15.10.4.3 set()

```
template<typename T, unsigned LSB, unsigned MSB>
Base_type cxx::Bitfield< T, LSB, MSB >::set (
    Base_type dest,
    Bits_type val) [inline], [static]
```

Set the bits corresponding to `val`.

Parameters

| | |
|-------------|---|
| <i>dest</i> | The current value of the whole bit field. |
| <i>val</i> | The value to set into the bits. |

Returns

The new value of the whole bit field.

Definition at line 158 of file `bitfield`.

15.10.4.4 set_dirty()

```
template<typename T, unsigned LSB, unsigned MSB>
constexpr Base_type cxx::Bitfield< T, LSB, MSB >::set_dirty (
    Base_type dest,
    Shift_type val) [inline], [static], [constexpr]
```

Set the bits corresponding to `val`.

Parameters

| | |
|-------------|---|
| <i>dest</i> | The current value of the whole bit field. |
| <i>val</i> | The value to set into the bits. |

Returns

The new value of the whole bit field.

Precondition

`val` must not contain more than `Bits` bits.

Note

This function does not mask `val` to the right number of bits.

Definition at line 124 of file `bitfield`.

Referenced by `cxx::Bitfield< decltype(raw), 0, 0 >::set()`.

Here is the caller graph for this function:

**15.10.4.5 set_unshifted()**

```

template<typename T, unsigned LSB, unsigned MSB>
Base_type cxx::Bitfield< T, LSB, MSB >::set_unshifted (
    Base_type dest,
    Shift_type val) [inline], [static]
  
```

Set the bits corresponding to `val`.

Parameters

| | |
|-------------|---|
| <i>dest</i> | The current value of the whole bit field. |
| <i>val</i> | The value shifted <code>Lsb</code> bits to the left that shall be set into the bit field. |

Returns

the new value of the whole bit field.

Definition at line 170 of file `bitfield`.

15.10.4.6 set_unshifted_dirty()

```

template<typename T, unsigned LSB, unsigned MSB>
constexpr Base_type cxx::Bitfield< T, LSB, MSB >::set_unshifted_dirty (
    Base_type dest,
    Shift_type val) [inline], [static], [constexpr]
  
```

Set the bits corresponding to `val`.

Parameters

| | |
|-------------|---|
| <i>dest</i> | The current value of the whole bit field. |
| <i>val</i> | The value shifted Lsb bits to the left that shall be set into the bits. |

Returns

The new value of the whole bit field.

Precondition

[val](#) must not contain more than [Bits](#) bits shifted [Lsb](#) bits to the left.

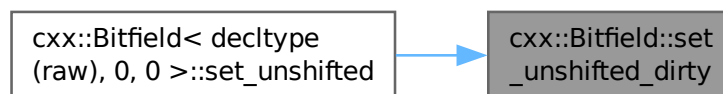
Note

This function does not mask [val](#) to the right number of bits.

Definition at line [144](#) of file [bitfield](#).

Referenced by [cxx::Bitfield< decltype\(raw\), 0, 0 >::set_unshifted\(\)](#).

Here is the caller graph for this function:

**15.10.4.7 val()**

```
template<typename T, unsigned LSB, unsigned MSB>
constexpr Base_type cxx::Bitfield< T, LSB, MSB >::val (
    Bits\_type val) [inline], [static], [constexpr]
```

Get the shifted bits for [val](#).

Parameters

| | |
|------------|---------------------------------|
| <i>val</i> | The value to set into the bits. |
|------------|---------------------------------|

Returns

The raw bit field value.

Definition at line [193](#) of file [bitfield](#).

15.10.4.8 `val_dirty()`

```
template<typename T, unsigned LSB, unsigned MSB>
constexpr Base_type cxx::Bitfield< T, LSB, MSB >::val_dirty (
    Shift_type val) [inline], [static], [constexpr]
```

Get the shifted bits for `val`.

Parameters

| | |
|------------------|---------------------------------|
| <code>val</code> | The value to set into the bits. |
|------------------|---------------------------------|

Returns

The raw bit field value.

Precondition

`val` must not contain more than `Bits` bits.

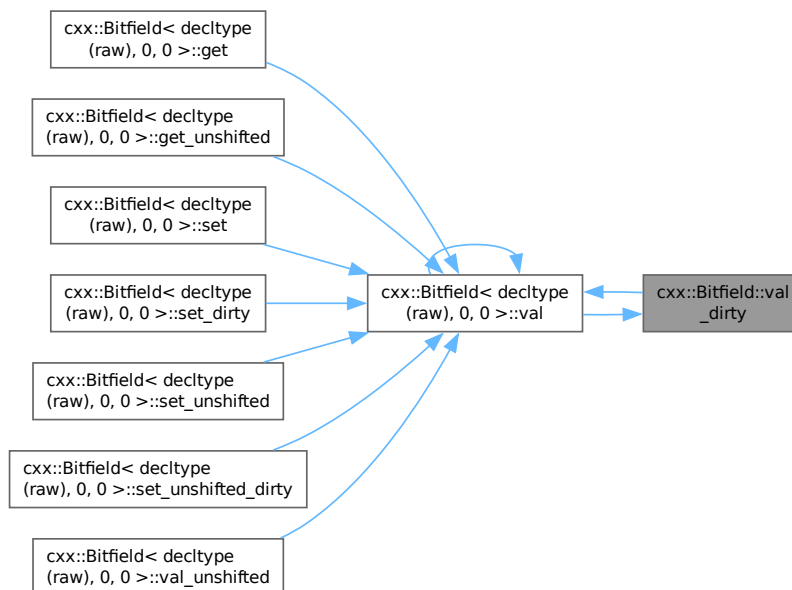
Note

This function does not mask `val` to the right number of bits.

Definition at line 184 of file `bitfield`.

Referenced by `cxx::Bitfield< decltype(raw), 0, 0 >::val()`.

Here is the caller graph for this function:



15.10.4.9 val_unshifted()

```
template<typename T, unsigned LSB, unsigned MSB>
constexpr Base_type cxx::Bitfield< T, LSB, MSB >::val_unshifted (
    Shift_type val) [inline], [static], [constexpr]
```

Get the shifted bits for [val](#).

Parameters

| | |
|------------|---|
| <i>val</i> | The value shifted Lsb bits to the left that shall be set into the bits. |
|------------|---|

Returns

The raw bit field value.

Definition at line [203](#) of file [bitfield](#).

The documentation for this class was generated from the following file:

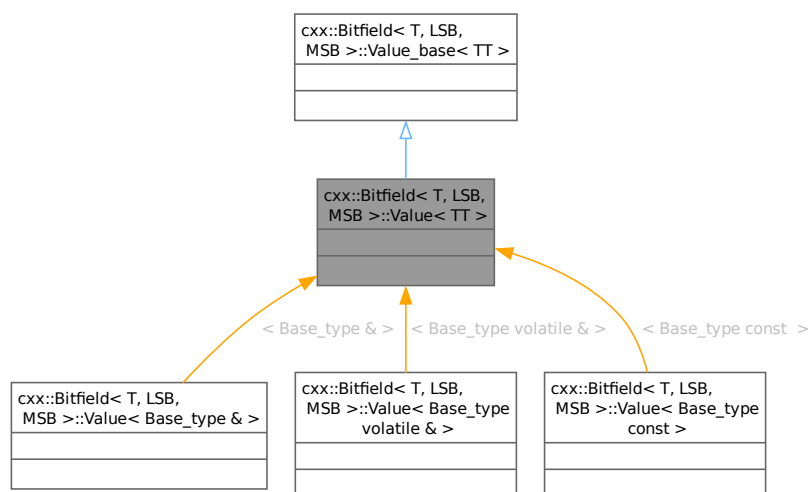
- [l4/cxx/bitfield](#)

15.11 cxx::Bitfield< T, LSB, MSB >::Value< TT > Class Template Reference

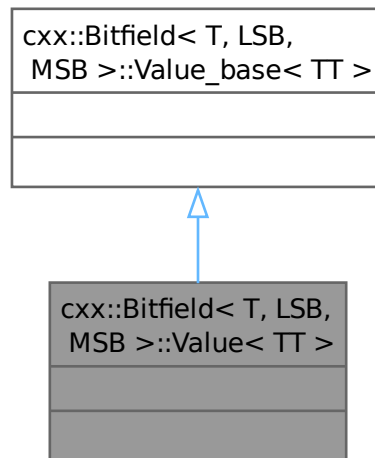
Internal helper type.

```
#include <bitfield>
```

Inheritance diagram for `cxx::Bitfield< T, LSB, MSB >::Value< TT >`:



Collaboration diagram for `cxx::Bitfield< T, LSB, MSB >::Value< TT >`:



15.11.1 Detailed Description

```

template<typename T, unsigned LSB, unsigned MSB>
template<typename TT>
class cxx::Bitfield< T, LSB, MSB >::Value< TT >

```

Internal helper type.

Definition at line [225](#) of file [bitfield](#).

The documentation for this class was generated from the following file:

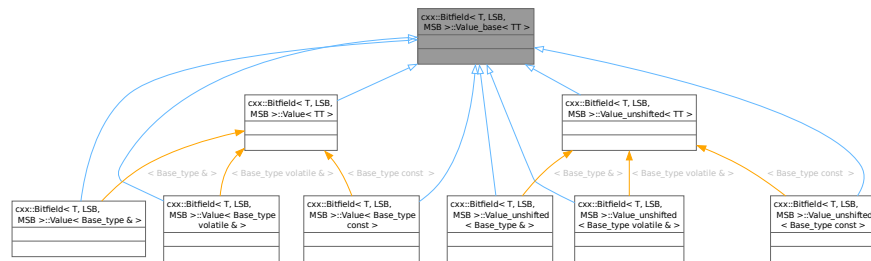
- `I4/cxx/bitfield`

15.12 `cxx::Bitfield< T, LSB, MSB >::Value_base< TT >` Class Template Reference

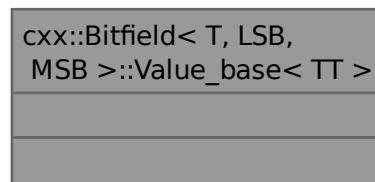
Internal helper type.

```
#include <bitfield>
```

Inheritance diagram for `cxx::Bitfield< T, LSB, MSB >::Value_base< TT >`:



Collaboration diagram for `cxx::Bitfield< T, LSB, MSB >::Value_base< TT >`:



15.12.1 Detailed Description

```

template<typename T, unsigned LSB, unsigned MSB>
template<typename TT>
class cxx::Bitfield< T, LSB, MSB >::Value_base< TT >

```

Internal helper type.

Definition at line 207 of file [bitfield](#).

The documentation for this class was generated from the following file:

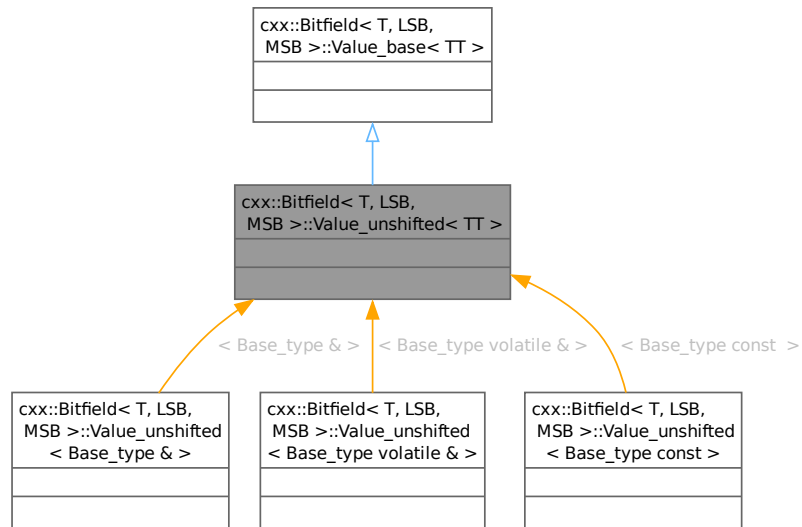
- `I4/cxx/bitfield`

15.13 `cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT >` Class Template Reference

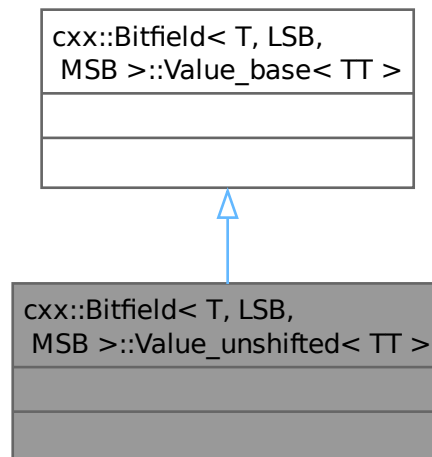
Internal helper type.

```
#include <bitfield>
```


Inheritance diagram for `cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT >`:



Collaboration diagram for `cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT >`:



15.13.1 Detailed Description

```

template<typename T, unsigned LSB, unsigned MSB>
template<typename TT>
class cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT >

```

Internal helper type.

Definition at line 238 of file [bitfield](#).

The documentation for this class was generated from the following file:

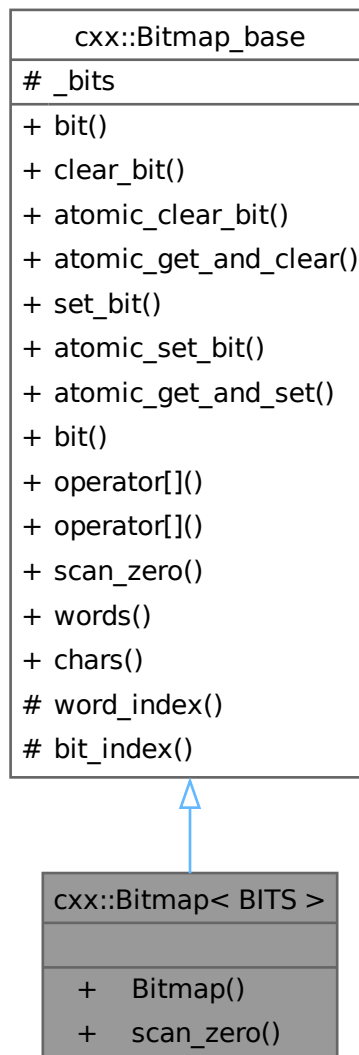
- [l4/cxx/bitfield](#)

15.14 cxx::Bitmap< BITS > Class Template Reference

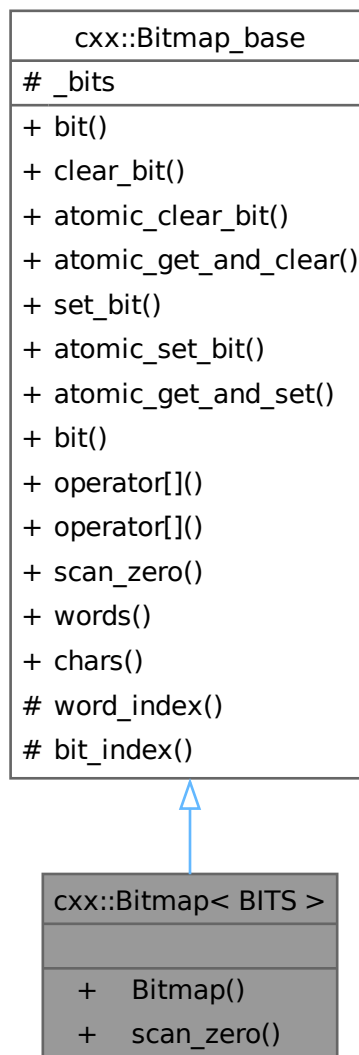
A static bitmap.

```
#include <bitmap>
```

Inheritance diagram for cxx::Bitmap< BITS >:



Collaboration diagram for `cxx::Bitmap< BITS >`:



Public Member Functions

- **Bitmap** () noexcept
Create a bitmap with `BITS` bits.
- long `scan_zero` (long start_bit=0) const noexcept
Scan for the first zero bit.

Public Member Functions inherited from `cxx::Bitmap_base`

- void `bit` (long bit, bool on) noexcept
Set the value of bit `bit` to `on`.

- void `clear_bit` (long `bit`) noexcept
Clear bit `bit`.
- void `atomic_clear_bit` (long `bit`) noexcept
Clear bit `bit` atomically.
- `word_type` `atomic_get_and_clear` (long `bit`) noexcept
Clear bit `bit` atomically and return old state.
- void `set_bit` (long `bit`) noexcept
Set bit `bit`.
- void `atomic_set_bit` (long `bit`) noexcept
Set bit `bit` atomically.
- `word_type` `atomic_get_and_set` (long `bit`) noexcept
Set bit `bit` atomically and return old state.
- `word_type` `bit` (long `bit`) const noexcept
Get the truth value of a bit.
- `word_type` `operator[]` (long `bit`) const noexcept
Get the bit at index `bit`.
- `Bit operator[]` (long `bit`) noexcept
Get the lvalue for the bit at index `bit`.
- long `scan_zero` (long `max_bit`, long `start_bit=0`) const noexcept
Scan for the first zero bit.

Additional Inherited Members

Static Public Member Functions inherited from `cxx::Bitmap_base`

- static long `words` (long `bits`) noexcept
Get the number of `Words` that are used for the bitmap.
- static long `chars` (long `bits`) noexcept
Get the number of chars that are used for the bitmap.

Protected Types inherited from `cxx::Bitmap_base`

- enum { `W_bits` = sizeof(`word_type`) * 8 , `C_bits` = 8 }
- typedef unsigned long `word_type`
Data type for each element of the bit buffer.

Static Protected Member Functions inherited from `cxx::Bitmap_base`

- static unsigned `word_index` (unsigned `bit`)
Get the word index for the given bit.
- static unsigned `bit_index` (unsigned `bit`)
Get the bit index within `word_type` for the given bit.

Protected Attributes inherited from `cxx::Bitmap_base`

- `word_type` * `_bits`
Pointer to the buffer storing the bits.

15.14.1 Detailed Description

```
template<int BITS>
class cxx::Bitmap< BITS >
```

A static bitmap.

Template Parameters

| | |
|-------------|---|
| <i>BITS</i> | The number of bits that shall be in the bitmap. |
|-------------|---|

Definition at line 220 of file [bitmap](#).

15.14.2 Member Function Documentation

15.14.2.1 scan_zero()

```
template<int BITS>
long cxx::Bitmap< BITS >::scan_zero (
    long start_bit = 0) const [inline], [noexcept]
```

Scan for the first zero bit.

Parameters

| | |
|------------------|--|
| <i>start_bit</i> | Hint at the number of the first bit to look at. Zero bits below <i>start_bit</i> may or may not be taken into account by the implementation. |
|------------------|--|

Return values

| | |
|--------------|---|
| <i>>=</i> | 0 Number of first zero bit found. |
| <i>-1</i> | All bits at <i>start_bit</i> or higher are set. |

Compared to [Bitmap_base::scan_zero\(\)](#), the upper bound is set to BITS.

Definition at line 365 of file [bitmap](#).

References [cxx::Bitmap_base::scan_zero\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

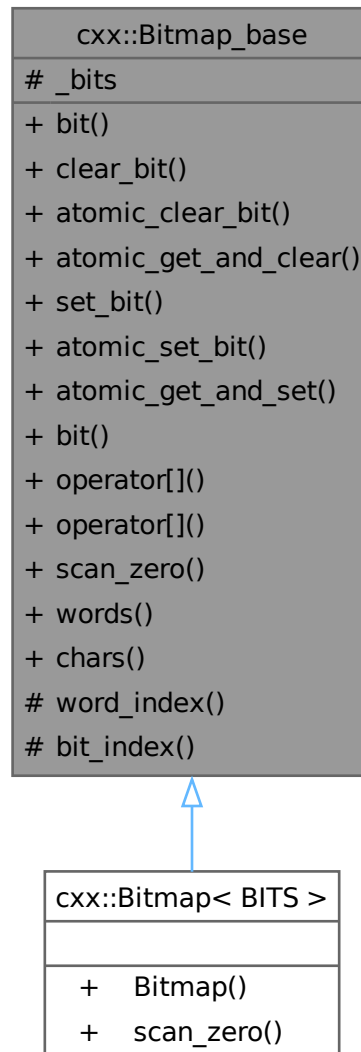
- [l4/cxx/bitmap](#)

15.15 cxx::Bitmap_base Class Reference

Basic bitmap abstraction.

```
#include <bitmap>
```

Inheritance diagram for cxx::Bitmap_base:



Collaboration diagram for cxx::Bitmap_base:

| cxx::Bitmap_base |
|--------------------------|
| # _bits |
| + bit() |
| + clear_bit() |
| + atomic_clear_bit() |
| + atomic_get_and_clear() |
| + set_bit() |
| + atomic_set_bit() |
| + atomic_get_and_set() |
| + bit() |
| + operator[]() |
| + operator[]() |
| + scan_zero() |
| + words() |
| + chars() |
| # word_index() |
| # bit_index() |

Data Structures

- class [Bit](#)
A writable bit in a bitmap.
- class [Word](#)
Helper abstraction for a word contained in the bitmap.
- class [Char](#)
Helper abstraction for a byte contained in the bitmap.

Public Member Functions

- void [bit](#) (long bit, bool on) noexcept
Set the value of bit [bit](#) to on.
- void [clear_bit](#) (long [bit](#)) noexcept
Clear bit [bit](#).
- void [atomic_clear_bit](#) (long [bit](#)) noexcept
Clear bit [bit](#) atomically.
- [word_type](#) [atomic_get_and_clear](#) (long [bit](#)) noexcept
Clear bit [bit](#) atomically and return old state.

- void `set_bit` (long `bit`) noexcept
Set bit `bit`.
- void `atomic_set_bit` (long `bit`) noexcept
Set bit `bit` atomically.
- `word_type` `atomic_get_and_set` (long `bit`) noexcept
Set bit `bit` atomically and return old state.
- `word_type` `bit` (long `bit`) const noexcept
Get the truth value of a bit.
- `word_type` `operator[]` (long `bit`) const noexcept
Get the bit at index `bit`.
- `Bit operator[]` (long `bit`) noexcept
Get the lvalue for the bit at index `bit`.
- long `scan_zero` (long `max_bit`, long `start_bit=0`) const noexcept
Scan for the first zero bit.

Static Public Member Functions

- static long `words` (long `bits`) noexcept
Get the number of `words` that are used for the bitmap.
- static long `chars` (long `bits`) noexcept
Get the number of `chars` that are used for the bitmap.

Protected Types

- enum { `W_bits` = sizeof(`word_type`) * 8 , `C_bits` = 8 }
- typedef unsigned long `word_type`
Data type for each element of the bit buffer.

Static Protected Member Functions

- static unsigned `word_index` (unsigned `bit`)
Get the word index for the given bit.
- static unsigned `bit_index` (unsigned `bit`)
Get the bit index within `word_type` for the given bit.

Protected Attributes

- `word_type` * `_bits`
Pointer to the buffer storing the bits.

15.15.1 Detailed Description

Basic bitmap abstraction.

This abstraction keeps a pointer to a memory area that is used as bitmap.

Definition at line 18 of file `bitmap`.

15.15.2 Member Enumeration Documentation

15.15.2.1 anonymous enum

anonymous enum [protected]

Enumerator

| | |
|--------|---|
| W_bits | number of bits in word_type |
| C_bits | number of bits in char |

Definition at line 26 of file [bitmap](#).

15.15.3 Member Function Documentation

15.15.3.1 atomic_clear_bit()

```
void cxx::Bitmap_base::atomic_clear_bit (
    long bit) [inline], [noexcept]
```

Clear bit [bit](#) atomically.

Use this function for multi-threaded access to the bitmap.

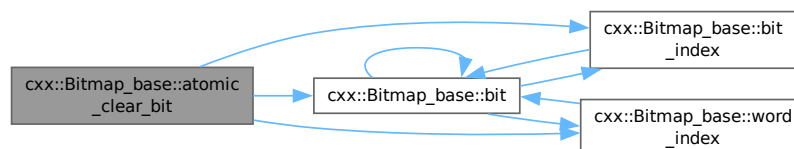
Parameters

| | |
|------------|---------------------------------|
| <i>bit</i> | The number of the bit to clear. |
|------------|---------------------------------|

Definition at line 269 of file [bitmap](#).

References [_bits](#), [bit\(\)](#), [bit_index\(\)](#), and [word_index\(\)](#).

Here is the call graph for this function:



15.15.3.2 atomic_get_and_clear()

```
Bitmap_base::word_type cxx::Bitmap_base::atomic_get_and_clear (
    long bit) [inline], [noexcept]
```

Clear bit `bit` atomically and return old state.

Use this function for multi-threaded access to the bitmap.

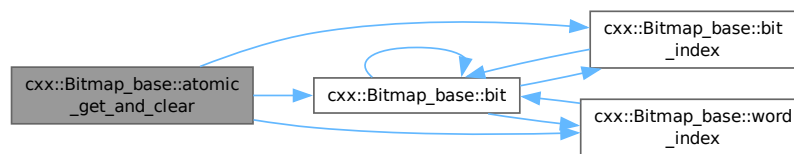
Parameters

| | |
|------------|---------------------------------|
| <i>bit</i> | The number of the bit to clear. |
|------------|---------------------------------|

Definition at line 279 of file [bitmap](#).

References [_bits](#), [bit\(\)](#), [bit_index\(\)](#), and [word_index\(\)](#).

Here is the call graph for this function:



15.15.3.3 atomic_get_and_set()

```
Bitmap_base::word_type cxx::Bitmap_base::atomic_get_and_set (
    long bit) [inline], [noexcept]
```

Set bit `bit` atomically and return old state.

Use this function for multi-threaded access to the bitmap.

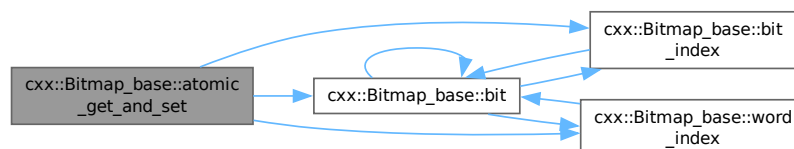
Parameters

| | |
|------------|-------------------------------|
| <i>bit</i> | The number of the bit to set. |
|------------|-------------------------------|

Definition at line 308 of file [bitmap](#).

References [_bits](#), [bit\(\)](#), [bit_index\(\)](#), and [word_index\(\)](#).

Here is the call graph for this function:



15.15.3.4 atomic_set_bit()

```
void cxx::Bitmap_base::atomic_set_bit (
    long bit) [inline], [noexcept]
```

Set bit *bit* atomically.

Use this function for multi-threaded access to the bitmap.

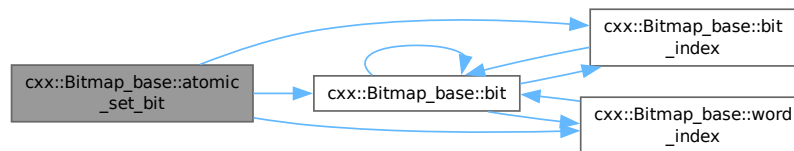
Parameters

| | |
|------------|-------------------------------|
| <i>bit</i> | The number of the bit to set. |
|------------|-------------------------------|

Definition at line 298 of file [bitmap](#).

References [_bits](#), [bit\(\)](#), [bit_index\(\)](#), and [word_index\(\)](#).

Here is the call graph for this function:



15.15.3.5 bit() [1/2]

```
Bitmap_base::word_type cxx::Bitmap_base::bit (
    long bit) const [inline], [noexcept]
```

Get the truth value of a bit.

Parameters

| | |
|------------|--------------------------------|
| <i>bit</i> | The number of the bit to read. |
|------------|--------------------------------|

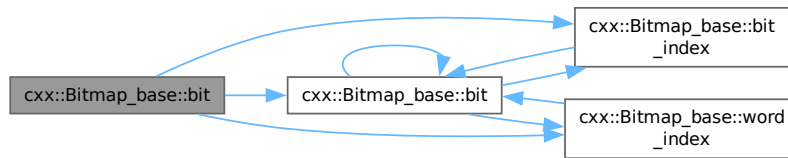
Return values

| | |
|------|-----------------|
| 0 | Bit is not set. |
| != 0 | Bit is set. |

Definition at line 318 of file [bitmap](#).

References [_bits](#), [bit\(\)](#), [bit_index\(\)](#), and [word_index\(\)](#).

Here is the call graph for this function:



15.15.3.6 bit() [2/2]

```
void cxx::Bitmap_base::bit (
    long bit,
    bool on) [inline], [noexcept]
```

Set the value of bit `bit` to `on`.

Parameters

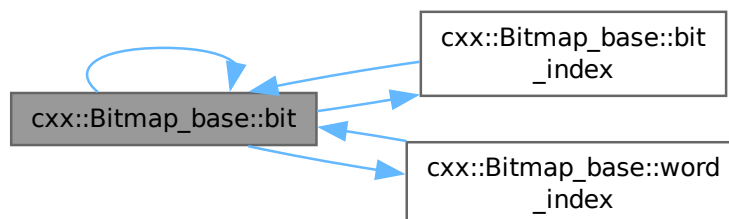
| | |
|------------|--|
| <i>bit</i> | The number of the bit. |
| <i>on</i> | The boolean value that shall be assigned to the bit. |

Definition at line 251 of file `bitmap`.

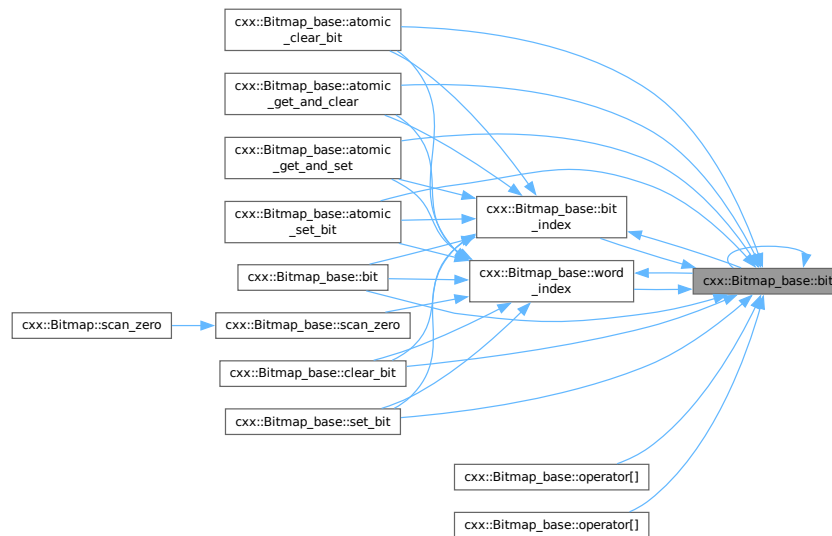
References `_bits`, `bit()`, `bit_index()`, and `word_index()`.

Referenced by `atomic_clear_bit()`, `atomic_get_and_clear()`, `atomic_get_and_set()`, `atomic_set_bit()`, `bit()`, `bit()`, `bit_index()`, `clear_bit()`, `operator[]()`, `operator[]()`, `set_bit()`, and `word_index()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.15.3.7 bit_index()

```
unsigned cxx::Bitmap_base::bit_index (
    unsigned bit) [inline], [static], [protected]
```

Get the bit index within [word_type](#) for the given bit.

Parameters

| | |
|------------|------------------------------|
| <i>bit</i> | The bit index in the bitmap. |
|------------|------------------------------|

Returns

the bit index within [word_type](#) ($\text{bit} \% W_bits$).

Definition at line 53 of file [bitmap](#).

References [bit\(\)](#), and [W_bits](#).

Referenced by [atomic_clear_bit\(\)](#), [atomic_get_and_clear\(\)](#), [atomic_get_and_set\(\)](#), [atomic_set_bit\(\)](#), [bit\(\)](#), [bit\(\)](#), [clear_bit\(\)](#), and [set_bit\(\)](#).

Here is the call graph for this function:



15.15.3.9 operator[]() [1/2]

```
word_type cxx::Bitmap_base::operator[] (
    long bit) const [inline], [noexcept]
```

Get the bit at index `bit`.

Parameters

| | |
|------------|--------------------------------|
| <i>bit</i> | The number of the bit to read. |
|------------|--------------------------------|

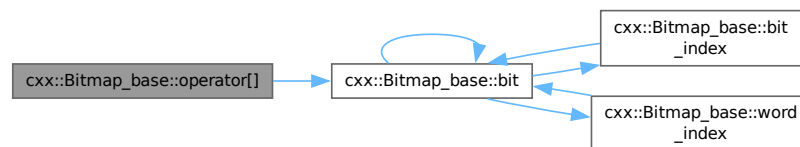
Return values

| | |
|------|-----------------|
| 0 | Bit is not set. |
| != 0 | Bit is set. |

Definition at line 181 of file `bitmap`.

References `bit()`.

Here is the call graph for this function:

**15.15.3.10 operator[]() [2/2]**

```
Bit cxx::Bitmap_base::operator[] (
    long bit) [inline], [noexcept]
```

Get the lvalue for the bit at index `bit`.

Parameters

| | |
|------------|-------------|
| <i>bit</i> | The number. |
|------------|-------------|

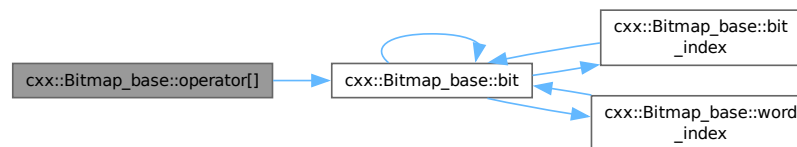
Returns

lvalue for [bit](#)

Definition at line 191 of file [bitmap](#).

References [bit\(\)](#).

Here is the call graph for this function:

**15.15.3.11 scan_zero()**

```
long cxx::Bitmap_base::scan_zero (
    long max_bit,
    long start_bit = 0) const [inline], [noexcept]
```

Scan for the first zero bit.

Parameters

| | |
|------------------|--|
| <i>max_bit</i> | Upper bound (exclusive) for the scanning operation. |
| <i>start_bit</i> | Hint at the number of the first bit to look at. Zero bits below <i>start_bit</i> may or may not be taken into account by the implementation. |

Return values

| | |
|--------------|---|
| <i>>=</i> | 0 Number of first zero bit found. |
| <i>-1</i> | All bits between <i>start_bit</i> and <i>max_bit</i> are set. |

Definition at line 339 of file [bitmap](#).

References [_bits](#), [W_bits](#), and [word_index\(\)](#).

Referenced by [cxx::Bitmap< BITS >::scan_zero\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.15.3.12 set_bit()

```
void cxx::Bitmap_base::set_bit (
    long bit) [inline], [noexcept]
```

Set bit `bit`.

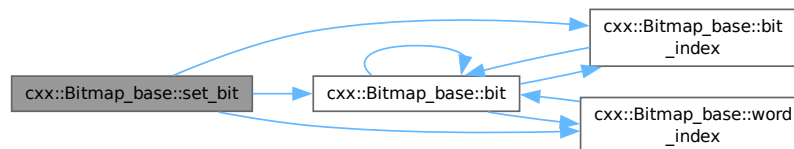
Parameters

| | |
|------------|-------------------------------|
| <i>bit</i> | The number of the bit to set. |
|------------|-------------------------------|

Definition at line 289 of file `bitmap`.

References `_bits`, `bit()`, `bit_index()`, and `word_index()`.

Here is the call graph for this function:



15.15.3.13 word_index()

```
unsigned cxx::Bitmap_base::word_index (
    unsigned bit) [inline], [static], [protected]
```

Get the word index for the given bit.

Parameters

| | |
|------------|-----------------------------------|
| <i>bit</i> | The index of the bit in question. |
|------------|-----------------------------------|

Returns

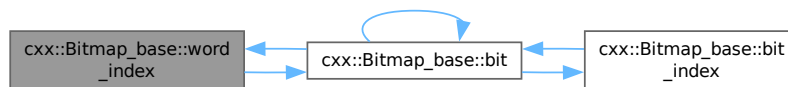
the index in `Bitmap_base::_bits` for the given bit (`bit / W_bits`).

Definition at line 44 of file `bitmap`.

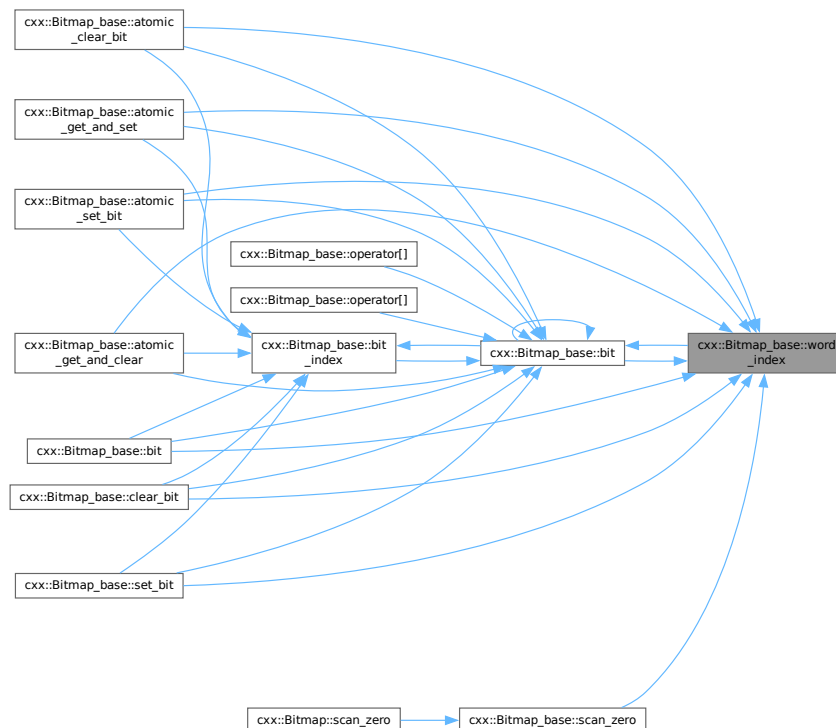
References `bit()`, and `W_bits`.

Referenced by `atomic_clear_bit()`, `atomic_get_and_clear()`, `atomic_get_and_set()`, `atomic_set_bit()`, `bit()`, `bit()`, `clear_bit()`, `scan_zero()`, and `set_bit()`.

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

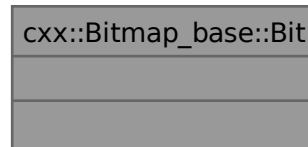
- `I4/cxx/bitmap`

15.16 cxx::Bitmap_base::Bit Class Reference

A writable bit in a bitmap.

```
#include <bitmap>
```

Collaboration diagram for cxx::Bitmap_base::Bit:



15.16.1 Detailed Description

A writable bit in a bitmap.

Definition at line 58 of file [bitmap](#).

The documentation for this class was generated from the following file:

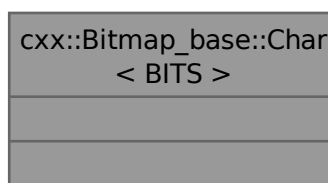
- I4/cxx/bitmap

15.17 cxx::Bitmap_base::Char< BITS > Class Template Reference

Helper abstraction for a byte contained in the bitmap.

```
#include <bitmap>
```

Collaboration diagram for cxx::Bitmap_base::Char< BITS >:



15.17.1 Detailed Description

```
template<long BITS>
class cxx::Bitmap_base::Char< BITS >
```

Helper abstraction for a byte contained in the bitmap.

Definition at line 95 of file [bitmap](#).

The documentation for this class was generated from the following file:

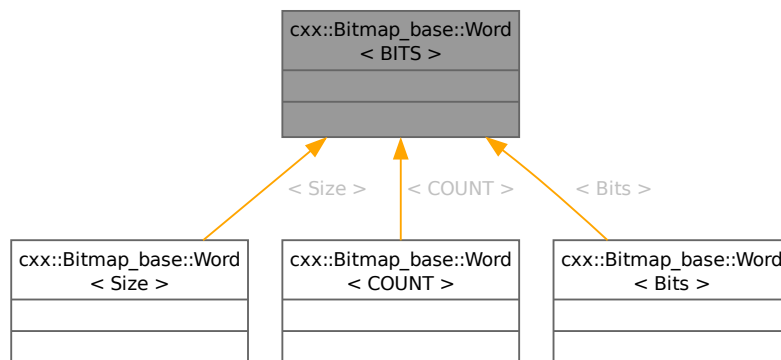
- I4/cxx/bitmap

15.18 cxx::Bitmap_base::Word< BITS > Class Template Reference

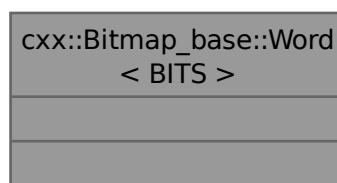
Helper abstraction for a word contained in the bitmap.

```
#include <bitmap>
```

Inheritance diagram for cxx::Bitmap_base::Word< BITS >:



Collaboration diagram for cxx::Bitmap_base::Word< BITS >:



15.18.1 Detailed Description

```
template<long BITS>
class cxx::Bitmap_base::Word< BITS >
```

Helper abstraction for a word contained in the bitmap.

Definition at line 79 of file [bitmap](#).

The documentation for this class was generated from the following file:

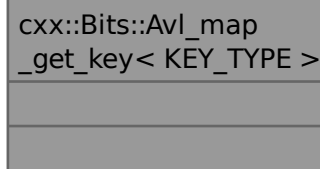
- I4/cxx/bitmap

15.19 cxx::Bits::Avl_map_get_key< KEY_TYPE > Struct Template Reference

Key-getter for [Avl_map](#).

```
#include <avl_map>
```

Collaboration diagram for cxx::Bits::Avl_map_get_key< KEY_TYPE >:



15.19.1 Detailed Description

```
template<typename KEY_TYPE>
struct cxx::Bits::Avl_map_get_key< KEY_TYPE >
```

Key-getter for [Avl_map](#).

Definition at line 25 of file [avl_map](#).

The documentation for this struct was generated from the following file:

- I4/cxx/[avl_map](#)

15.20 `cxx::Bits::Avl_set_get_key< KEY_TYPE >` Struct Template Reference

Internal, key-getter for [Avl_set](#) nodes.

```
#include <avl_set>
```

Collaboration diagram for `cxx::Bits::Avl_set_get_key< KEY_TYPE >`:



15.20.1 Detailed Description

```
template<typename KEY_TYPE>
struct cxx::Bits::Avl_set_get_key< KEY_TYPE >
```

Internal, key-getter for [Avl_set](#) nodes.

Definition at line [103](#) of file [avl_set](#).

The documentation for this struct was generated from the following file:

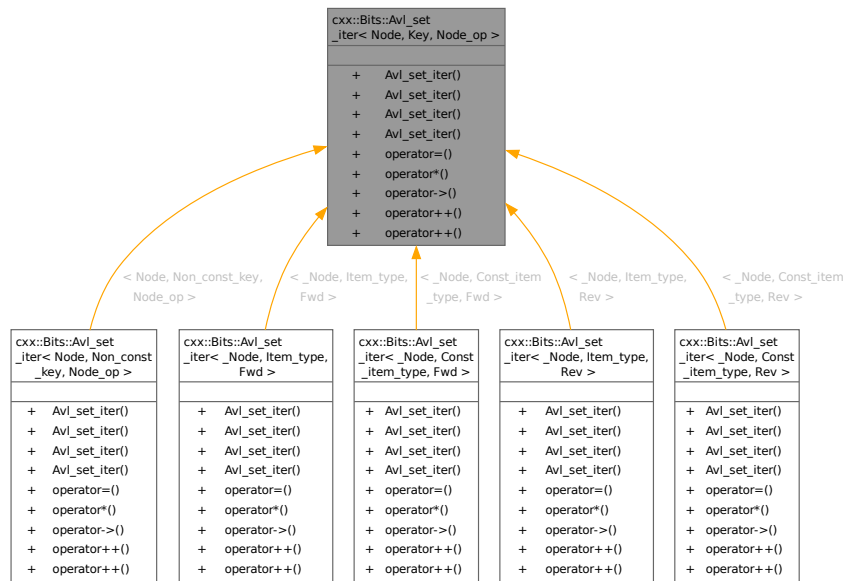
- [l4/cxx/avl_set](#)

15.21 `cxx::Bits::Avl_set_iter< Node, Key, Node_op >` Class Template Reference

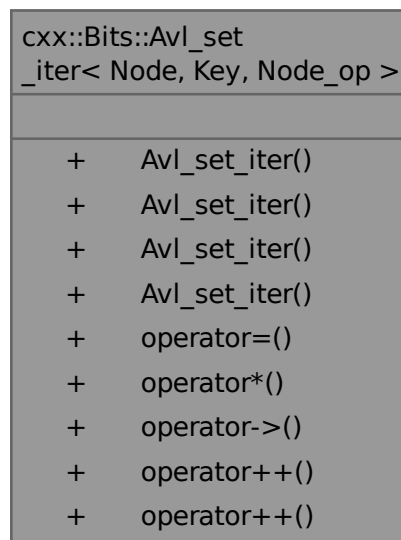
Generic iterator for the AVL-tree based set.

```
#include <avl_set>
```

Inheritance diagram for cxx::Bits::Avl_set_iter< Node, Key, Node_op >:



Collaboration diagram for cxx::Bits::Avl_set_iter< Node, Key, Node_op >:



Public Member Functions

- `Avl_set_iter()`=default
Create an invalid iterator (end marker).

- [Avl_set_iter](#) (Node const *t)
Create an iterator for the given tree.
- [Avl_set_iter](#) (Base const &o)
Create an iterator from a BST iterator.
- [Avl_set_iter](#) (Non_const_iter const &o)
Allow copy of non-const iterator to const iterator versions.
- [Avl_set_iter](#) & **operator=** (Non_const_iter const &o)
Allow assignment of non-const iterator to const iterator versions.
- Key & **operator*** () const
Dereference the iterator and get the item out of the tree.
- Key * **operator->** () const
Member access to the item the iterator points to.
- [Avl_set_iter](#) & **operator++** ()
Set the iterator to the next element (pre increment).
- [Avl_set_iter](#) **operator++** (int)
Set the iterator to the next element (post increment).

15.21.1 Detailed Description

```
template<typename Node, typename Key, typename Node_op>
class cxx::Bits::Avl_set_iter< Node, Key, Node_op >
```

Generic iterator for the AVL-tree based set.

Definition at line 35 of file [avl_set](#).

15.21.2 Constructor & Destructor Documentation

15.21.2.1 Avl_set_iter() [1/2]

```
template<typename Node, typename Key, typename Node_op>
cxx::Bits::Avl_set_iter< Node, Key, Node_op >::Avl_set_iter (
    Node const * t) [inline]
```

Create an iterator for the given tree.

Parameters

| | |
|----------|---------------------------------------|
| <i>t</i> | the root node of the tree to iterate. |
|----------|---------------------------------------|

Definition at line 59 of file [avl_set](#).

15.21.2.2 Avl_set_iter() [2/2]

```
template<typename Node, typename Key, typename Node_op>
cxx::Bits::Avl_set_iter< Node, Key, Node_op >::Avl_set_iter (
    Base const & o) [inline]
```

Create an iterator from a BST iterator.

Parameters

| | |
|----------|--|
| <i>o</i> | The BST iterator that shall be copied. |
|----------|--|

Definition at line 65 of file [avl_set](#).

15.21.3 Member Function Documentation

15.21.3.1 operator*()

```
template<typename Node, typename Key, typename Node_op>
Key & cxx::Bits::Avl_set_iter< Node, Key, Node_op >::operator* () const [inline]
```

Dereference the iterator and get the item out of the tree.

Returns

A reference to the data stored in the AVL tree.

Definition at line 80 of file [avl_set](#).

15.21.3.2 operator->()

```
template<typename Node, typename Key, typename Node_op>
Key * cxx::Bits::Avl_set_iter< Node, Key, Node_op >::operator-> () const [inline]
```

Member access to the item the iterator points to.

Returns

A pointer to the item in the node.

Definition at line 87 of file [avl_set](#).

The documentation for this class was generated from the following file:

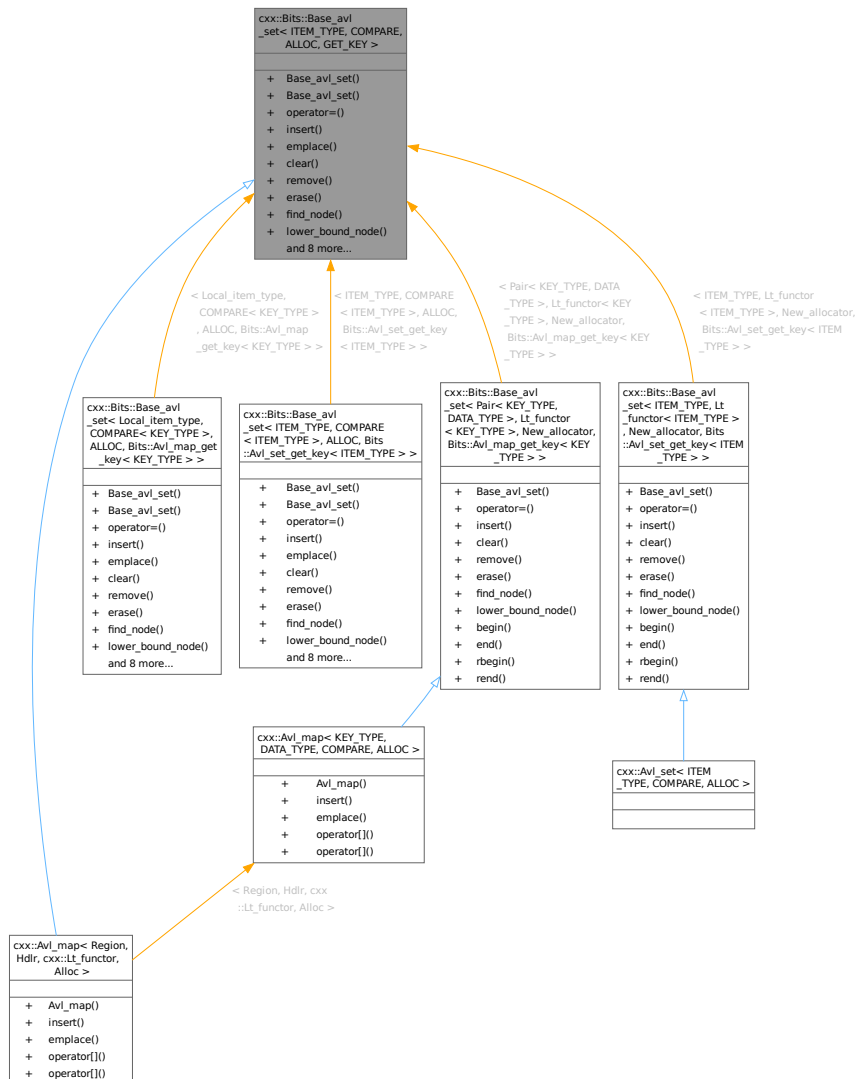
- [I4/cxx/avl_set](#)

15.22 cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY > Class Template Reference

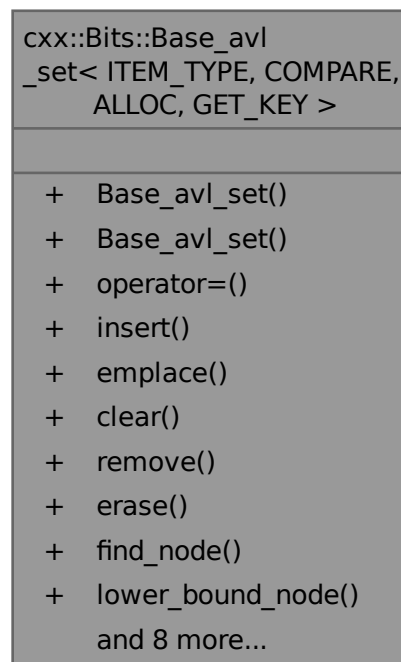
AVL set with internally managed nodes.

```
#include <avl_set>
```

Inheritance diagram for cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >:



Collaboration diagram for `cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >`:



Data Structures

- class [Node](#)
A smart pointer to a tree item.

Public Types

- enum { `E_noent` = 2 , `E_exist` = 17 , `E_nomem` = 12 , `E_inval` = 22 }
Return status constants.
- typedef `ITEM_TYPE` **Item_type**
Type for the items store in the set.
- typedef `GET_KEY` **Get_key**
Key-getter type to derive the sort key of an internal node.
- typedef `GET_KEY::Key_type` **Key_type**
Type of the sort key used for the items.
- typedef `Type_traits< Item_type >::Const_type` **Const_item_type**
Type used for const items within the set.
- typedef `COMPARE` **Item_compare**
Type for the comparison functor.
- typedef `ALLOC< _Node >` **Node_allocator**
Type for the node allocator.
- typedef `Avl_set_iter< _Node, Item_type, Fwd >` **Iterator**

- Forward iterator for the set.*
- typedef `Avl_set_iter`< `_Node`, `Const_item_type`, `Fwd` > **`Const_iterator`**
Constant forward iterator for the set.
- typedef `Avl_set_iter`< `_Node`, `Item_type`, `Rev` > **`Rev_iterator`**
Backward iterator for the set.
- typedef `Avl_set_iter`< `_Node`, `Const_item_type`, `Rev` > **`Const_rev_iterator`**
Constant backward iterator for the set.

Public Member Functions

- `Base_avl_set` (`Node_allocator` const &alloc=`Node_allocator`())
Create a AVL-tree based set.
- `Base_avl_set` (`Base_avl_set` const &o, `Node_allocator` const &alloc=`Node_allocator`())
Create a copy of an AVL-tree based set.
- `Base_avl_set` & `operator=` (`Base_avl_set` const &o)
Copy assignment operator of an AVL-tree based set.
- `cxx::Pair`< `Iterator`, `int` > `insert` (`Item_type` const &item)
Insert an item into the set.
- template<typename... Args>
`cxx::Pair`< `Iterator`, `int` > `emplace` (`Args` &&... args)
Eplace an item into the set.
- void **`clear`** () noexcept
Remove all items from the set.
- int `remove` (`Key_type` const &item)
Remove an item from the set.
- int `erase` (`Key_type` const &item)
Erase the item with the given key.
- `Node` `find_node` (`Key_type` const &item) const
Lookup a node equal to `item`.
- `Node` `lower_bound_node` (`Key_type` const &key) const
Find the first node greater or equal to `key`.
- `Const_iterator` `begin` () const
Get the constant forward iterator for the first element in the set.
- `Const_iterator` `end` () const
Get the end marker for the constant forward iterator.
- `Iterator` `begin` ()
Get the mutable forward iterator for the first element of the set.
- `Iterator` `end` ()
Get the end marker for the mutable forward iterator.
- `Const_rev_iterator` `rbegin` () const
Get the constant backward iterator for the last element in the set.
- `Const_rev_iterator` `rend` () const
Get the end marker for the constant backward iterator.
- `Rev_iterator` `rbegin` ()
Get the mutable backward iterator for the last element of the set.
- `Rev_iterator` `rend` ()
Get the end marker for the mutable backward iterator.

15.22.1 Detailed Description

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename
GET_KEY>
```

```
class cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >
```

AVL set with internally managed nodes.

Definition at line 127 of file [avl_set](#).

15.22.2 Member Enumeration Documentation

15.22.2.1 anonymous enum

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename
GET_KEY>
```

```
anonymous enum
```

Return status constants.

These constants are compatible with the [L4](#) error codes, see [l4_error_code_t](#).

Enumerator

| | |
|---------|---------------------------|
| E_noent | Item does not exist. |
| E_exist | Item exists already. |
| E_nomem | Memory allocation failed. |
| E_inval | Internal error. |

Definition at line 138 of file [avl_set](#).

15.22.3 Constructor & Destructor Documentation

15.22.3.1 Base_avl_set() [1/2]

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename
GET_KEY>
```

```
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Base_avl_set (
    Node_allocator const & alloc = Node_allocator()) [inline], [explicit]
```

Create a AVL-tree based set.

Parameters

| | |
|--------------|---------------------------------|
| <i>alloc</i> | Node allocator. |
|--------------|---------------------------------|

Create an empty set (AVL-tree based).

Definition at line 268 of file [avl_set](#).

15.22.3.2 Base_avl_set() [2/2]

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename
GET_KEY>
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Base_avl_set (
    Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY > const & o,
    Node_allocator const & alloc = Node_allocator()) [inline]
```

Create a copy of an AVL-tree based set.

Parameters

| | |
|--------------|---------------------------------|
| <i>o</i> | The set to copy. |
| <i>alloc</i> | Node allocator. |

Creates a deep copy of the set with all its items.

Definition at line [283](#) of file [avl_set](#).

15.22.4 Member Function Documentation

15.22.4.1 begin() [1/2]

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename
GET_KEY>
Iterator cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::begin () [inline]
```

Get the mutable forward iterator for the first element of the set.

Returns

The mutable forward iterator for the first element of the set.

Definition at line [431](#) of file [avl_set](#).

15.22.4.2 begin() [2/2]

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename
GET_KEY>
Const_iterator cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::begin () const
[inline]
```

Get the constant forward iterator for the first element in the set.

Returns

Constant forward iterator for the first element in the set.

Definition at line [417](#) of file [avl_set](#).

15.22.4.3 `emplace()`

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename
GET_KEY>
template<typename... Args>
cxx::Pair< Iterator, int > cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >←
::emplace (
    Args &&... args)
```

Emplace an item into the set.

Parameters

| | |
|-------------------|---|
| <code>args</code> | Constructor arguments for the item constructor. |
|-------------------|---|

Returns

A pair of iterator (`first`) and return value (`second`). `second` will be 0 if the element was inserted into the set and `-#E_exist` if the element was already in the set and the set was therefore not updated. In both cases, `first` contains an iterator that points to the element. `second` may also be `-#E_nomem` when memory for the node could not be allocated. `first` is then invalid.

The element will always be constructed, even if there is already an existing element in the set. In case of such a collision the newly created element will then be immediately destructed.

15.22.4.4 `end()` [1/2]

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename
GET_KEY>
Iterator cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::end () [inline]
```

Get the end marker for the mutable forward iterator.

Returns

The end marker for mutable forward iterator.

Definition at line 438 of file `avl_set`.

15.22.4.5 `end()` [2/2]

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename
GET_KEY>
Const_iterator cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::end () const
[inline]
```

Get the end marker for the constant forward iterator.

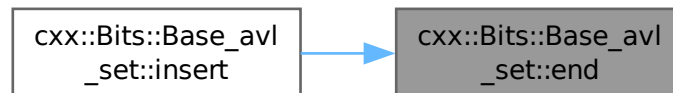
Returns

The end marker for the constant forward iterator.

Definition at line 424 of file [avl_set](#).

Referenced by [insert\(\)](#).

Here is the caller graph for this function:

**15.22.4.6 erase()**

```

template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename
GET_KEY>
int cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::erase (
    Key_type const & item) [inline]
  
```

Erase the item with the given key.

Parameters

| | |
|-------------|--------------------------------|
| <i>item</i> | The key of the item to remove. |
|-------------|--------------------------------|

Definition at line 384 of file [avl_set](#).

15.22.4.7 find_node()

```

template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename
GET_KEY>
Node cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::find_node (
    Key_type const & item) const [inline]
  
```

Lookup a node equal to *item*.

Parameters

| | |
|-------------|--------------------------|
| <i>item</i> | The value to search for. |
|-------------|--------------------------|

Returns

A smart pointer to the element found. If no element was found the smart pointer will be invalid.

Definition at line 395 of file [avl_set](#).

15.22.4.8 insert()

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename
GET_KEY>
Pair< typename Base_avl_set< Item, Compare, Alloc, KEY_TYPE >::Iterator, int > cxx::Bits::Base_avl_set<
Item, Compare, Alloc, KEY_TYPE >::insert (
    Item_type const & item)
```

Insert an item into the set.

Parameters

| | |
|-------------|---------------------|
| <i>item</i> | The item to insert. |
|-------------|---------------------|

Returns

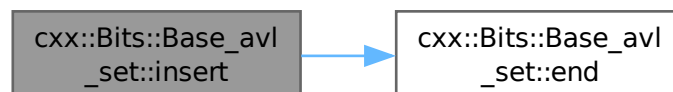
A pair of iterator (*first*) and return value (*second*). *second* will be 0 if the element was inserted into the set and `−#E_exist` if the element was already in the set and the set was therefore not updated. In both cases, *first* contains an iterator that points to the element. *second* may also be `−#E_nomem` when memory for the node could not be allocated. *first* is then invalid.

Insert a new item into the set, each item can only be once in the set.

Definition at line 485 of file [avl_set](#).

References [E_exist](#), [E_nomem](#), [end\(\)](#), [cxx::Pair< First, Second >::first](#), and [cxx::Pair< First, Second >::second](#).

Here is the call graph for this function:



15.22.4.9 lower_bound_node()

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename
GET_KEY>
Node cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::lower_bound_node (
    Key_type const & key) const [inline]
```

Find the first node greater or equal to *key*.

Parameters

| | |
|------------|--------------------------|
| <i>key</i> | Minimum key to look for. |
|------------|--------------------------|

Returns

Smart pointer to the first node greater or equal to *key*. Will be invalid if no such element was found.

Definition at line 406 of file [avl_set](#).

15.22.4.10 operator=()

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename
GET_KEY>
Base_avl_set & cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::operator= (
    Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY > const & o) [inline]
```

Copy assignment operator of an AVL-tree based set.

Parameters

| | |
|----------|-------------------------|
| <i>o</i> | The set to copy-assign. |
|----------|-------------------------|

Creates a deep copy of the set with all its items.

Definition at line 295 of file [avl_set](#).

15.22.4.11 rbegin() [1/2]

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename
GET_KEY>
Rev_iterator cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::rbegin () [inline]
```

Get the mutable backward iterator for the last element of the set.

Returns

The mutable backward iterator for the last element of the set.

Definition at line 459 of file [avl_set](#).

15.22.4.12 rbegin() [2/2]

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename
GET_KEY>
Const_rev_iterator cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::rbegin ()
const [inline]
```

Get the constant backward iterator for the last element in the set.

Returns

The constant backward iterator for the last element in the set.

Definition at line 445 of file [avl_set](#).

15.22.4.13 `remove()`

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename
GET_KEY>
int cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::remove (
    Key_type const & item) [inline]
```

Remove an item from the set.

Parameters

| | |
|-------------|---------------------|
| <i>item</i> | The item to remove. |
|-------------|---------------------|

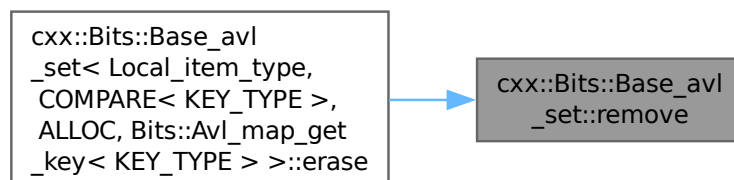
Return values

| | |
|-----------------------|---------------------|
| <code>0</code> | Success |
| <code>-E_noent</code> | Item does not exist |

Definition at line 366 of file `avl_set`.

Referenced by `cxx::Bits::Base_avl_set< Local_item_type, COMPARE< KEY_TYPE >, ALLOC, Bits::Avl_map_get_key< KEY_TYPE >`

Here is the caller graph for this function:



15.22.4.14 `rend()` [1/2]

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename
GET_KEY>
Rev_iterator cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::rend () [inline]
```

Get the end marker for the mutable backward iterator.

Returns

The end marker for mutable backward iterator.

Definition at line 466 of file `avl_set`.

15.22.4.15 `rend()` [2/2]

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename
GET_KEY>
Const_rev_iterator cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::rend ()
const [inline]
```

Get the end marker for the constant backward iterator.

Returns

The end marker for the constant backward iterator.

Definition at line [452](#) of file [avl_set](#).

The documentation for this class was generated from the following file:

- [l4/cxx/avl_set](#)

15.23 `cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node` Class Reference

A smart pointer to a tree item.

```
#include <avl_set>
```

Collaboration diagram for `cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node`:

| cxx::Bits::Base_avl _set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node | |
|--|---------------------------------|
| | |
| + | Node() |
| + | operator*() |
| + | operator->() |
| + | valid() |
| + | operator Item_type const *() |

Public Member Functions

- **Node** ()
Default construction for NIL pointer.
- `Item_type` const & **operator*** ()
Dereference the pointer.
- `Item_type` const * **operator->** ()
Dereferenced member access.
- bool **valid** () const
Validity check.
- **operator** `Item_type` const * ()
Cast to a real item pointer.

15.23.1 Detailed Description

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename
GET_KEY>
class cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node
```

A smart pointer to a tree item.

Definition at line 182 of file `avl_set`.

15.23.2 Member Function Documentation

15.23.2.1 `operator*()`

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename
GET_KEY>
Item_type const & cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node::operator*
() [inline]
```

Dereference the pointer.

Precondition

`Node` is valid.

Definition at line 200 of file `avl_set`.

15.23.2.2 `operator->()`

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename
GET_KEY>
Item_type const * cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node::operator->
() [inline]
```

Dereferenced member access.

Precondition

`Node` is valid.

Definition at line 207 of file `avl_set`.

Collaboration diagram for cxx::Bits::Basic_list< POLICY >:

| cxx::Bits::Basic_list < POLICY > | |
|-------------------------------------|---------|
| # | _f |
| + | empty() |
| + | front() |
| + | clear() |
| + | begin() |
| + | begin() |
| + | end() |
| + | end() |
| + | iter() |

Public Member Functions

- bool **empty** () const
Check if the list is empty.
- Value_type **front** () const
Return the first element in the list.
- void **clear** ()
Remove all elements from the list.
- Iterator **begin** ()
Return an iterator to the beginning of the list.
- Const_iterator **begin** () const
Return a const iterator to the beginning of the list.
- Const_iterator **end** () const
Return a const iterator to the end of the list.
- Iterator **end** ()
Return an iterator to the end of the list.

Static Public Member Functions

- static Const_iterator **iter** (Const_value_type c)
Return a const iterator that begins at the given element.

Protected Attributes

- POLICY::Head_type **_f**
Pointer to front of the list.

15.24.1 Detailed Description

```
template<typename POLICY>
class cxx::Bits::Basic_list< POLICY >
```

Internal: Common functions for all head-based list implementations.

Definition at line 39 of file [list_basics.h](#).

15.24.2 Member Function Documentation

15.24.2.1 clear()

```
template<typename POLICY>
void cxx::Bits::Basic_list< POLICY >::clear () [inline]
```

Remove all elements from the list.

After the operation the state of the elements is undefined.

Definition at line 139 of file [list_basics.h](#).

References [_f](#).

15.24.2.2 iter()

```
template<typename POLICY>
Const_iterator cxx::Bits::Basic_list< POLICY >::iter (
    Const_value_type c) [inline], [static]
```

Return a const iterator that begins at the given element.

Parameters

| | |
|----------|--|
| <i>c</i> | Element where the iterator should start. |
|----------|--|

Precondition

The element *c* must already be in a list.

Definition at line 152 of file [list_basics.h](#).

The documentation for this class was generated from the following file:

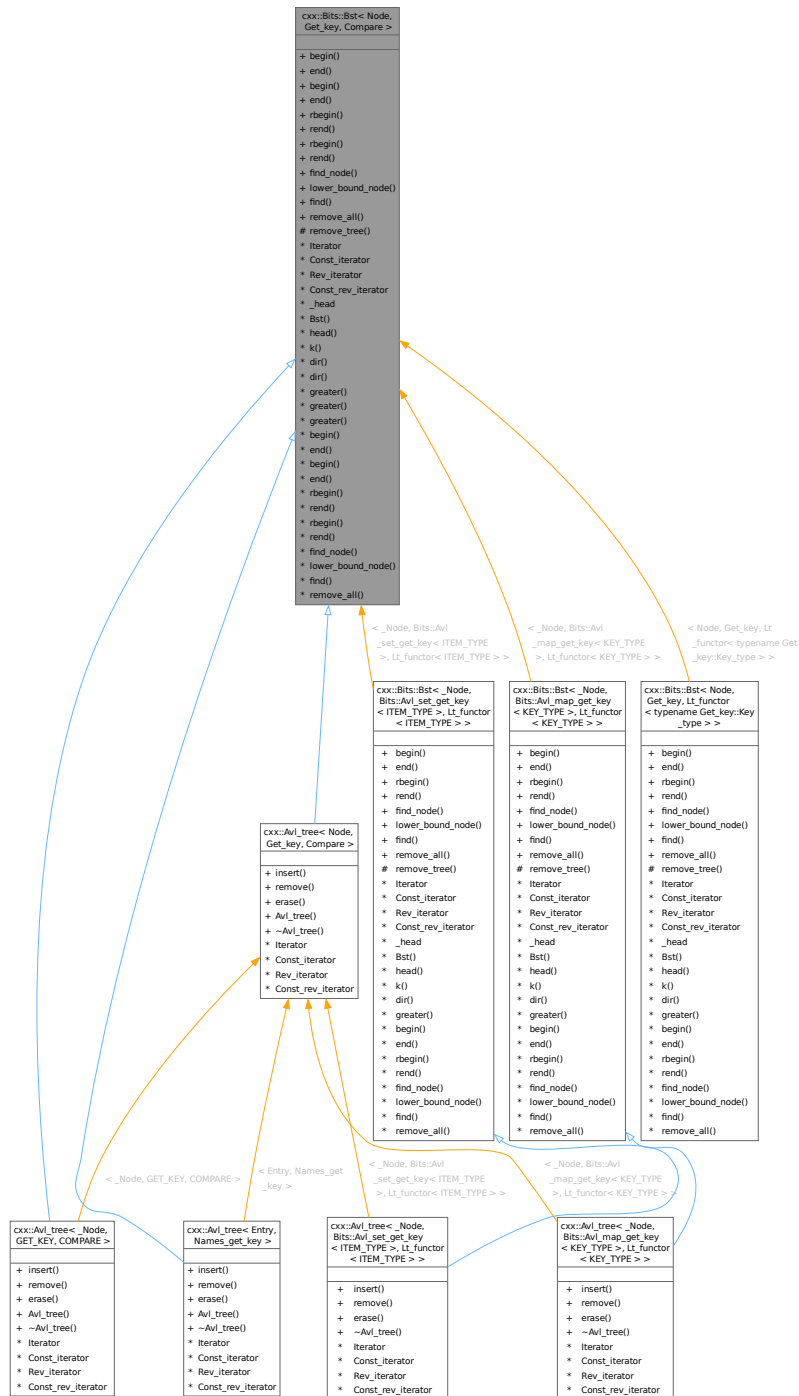
- I4/cxx/bits/list_basics.h

15.25 cxx::Bits::Bst< Node, Get_key, Compare > Class Template Reference

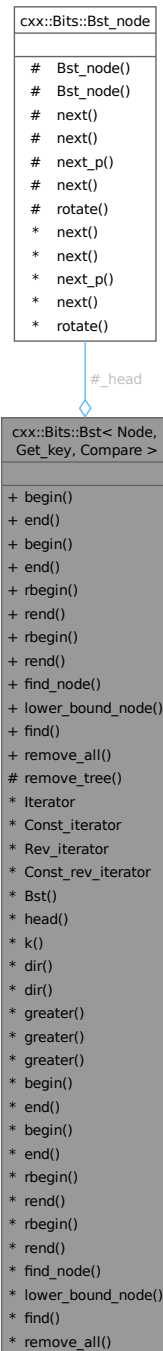
Basic binary search tree (BST).

```
#include <bst.h>
```

Inheritance diagram for cxx::Bits::Bst< Node, Get_key, Compare >:



Collaboration diagram for `cxx::Bits::Bst< Node, Get_key, Compare >`:



Public Types

- `typedef Get_key::Key_type Key_type`
The type of key values used to generate the total order of the elements.
- `typedef Type_traits< Key_type >::Param_type Key_param_type`
The type for key parameters.
- `typedef Fwd Fwd_iter_ops`

Helper for building forward iterators for different wrapper classes.

- typedef Rev **Rev_iter_ops**

Helper for building reverse iterators for different wrapper classes.

Iterators

- typedef __Bst_iter< Node, Node, Fwd > **Iterator**
Forward iterator.
- typedef __Bst_iter< Node, Node const, Fwd > **Const_iterator**
Constant forward iterator.
- typedef __Bst_iter< Node, Node, Rev > **Rev_iterator**
Backward iterator.
- typedef __Bst_iter< Node, Node const, Rev > **Const_rev_iterator**
Constant backward.

Public Member Functions

Get default iterators for the ordered tree.

- [Const_iterator begin](#) () const
Get the constant forward iterator for the first element in the set.
- [Const_iterator end](#) () const
Get the end marker for the constant forward iterator.
- [Iterator begin](#) ()
Get the mutable forward iterator for the first element of the set.
- [Iterator end](#) ()
Get the end marker for the mutable forward iterator.
- [Const_rev_iterator rbegin](#) () const
Get the constant backward iterator for the last element in the set.
- [Const_rev_iterator rend](#) () const
Get the end marker for the constant backward iterator.
- [Rev_iterator rbegin](#) ()
Get the mutable backward iterator for the last element of the set.
- [Rev_iterator rend](#) ()
Get the end marker for the mutable backward iterator.

Lookup functions.

- Node * [find_node](#) (Key_param_type key) const
find the node with the given key.
- Node * [lower_bound_node](#) (Key_param_type key) const
Find the first node with a key not less than the given key.
- [Const_iterator find](#) (Key_param_type key) const
find the node with the given key.
- template<typename FUNC>
void [remove_all](#) (FUNC &&callback)
Clear the tree.

Static Protected Member Functions

- template<typename FUNC>
static void [remove_tree](#) (Bst_node *head, FUNC &&callback)
Remove all elements in the subtree of head.

Interior access for descendants.

As this class is an intended base class we provide protected access to our interior, use 'using' to make this private in concrete implementations.

- `Bst_node * _head`
The head pointer of the tree.
- `Bst ()`
Create an empty tree.
- `Node * head () const`
Access the head node as object of type Node.
- `static Key_type k (Bst_node const *n)`
Get the key value of n.
- `static Dir dir (Key_param_type l, Key_param_type r)`
Get the direction to go from l to search for r.
- `static Dir dir (Key_param_type l, Bst_node const *r)`
Get the direction to go from l to search for r.
- `static bool greater (Key_param_type l, Key_param_type r)`
Is l greater than r.
- `static bool greater (Key_param_type l, Bst_node const *r)`
Is l greater than r.
- `static bool greater (Bst_node const *l, Bst_node const *r)`
Is l greater than r.

15.25.1 Detailed Description

```
template<typename Node, typename Get_key, typename Compare>
class cxx::Bits::Bst< Node, Get_key, Compare >
```

Basic binary search tree (BST).

This class is intended as a base class for concrete binary search trees, such as an AVL tree. This class already provides the basic lookup methods and iterator definitions for a BST.

Definition at line 31 of file [bst.h](#).

15.25.2 Member Function Documentation

15.25.2.1 begin() [1/2]

```
template<typename Node, typename Get_key, typename Compare>
Iterator cxx::Bits::Bst< Node, Get_key, Compare >::begin () [inline]
```

Get the mutable forward iterator for the first element of the set.

Returns

The mutable forward iterator for the first element of the set.

Definition at line 184 of file [bst.h](#).

References [head\(\)](#).

Here is the call graph for this function:

**15.25.2.2 `begin()` [2/2]**

```
template<typename Node, typename Get_key, typename Compare>
Const_iterator cxx::Bits::Bst< Node, Get_key, Compare >::begin () const [inline]
```

Get the constant forward iterator for the first element in the set.

Returns

Constant forward iterator for the first element in the set.

Definition at line 173 of file [bst.h](#).

References [head\(\)](#).

Here is the call graph for this function:



15.25.2.3 dir() [1/2]

```
template<typename Node, typename Get_key, typename Compare>
Dir cxx::Bits::Bst< Node, Get_key, Compare >::dir (
    Key_param_type l,
    Bst_node const * r) [inline], [static], [protected]
```

Get the direction to go from `l` to search for `r`.

Parameters

| | |
|----------------|--------------------------------------|
| <code>l</code> | is the key to look for. |
| <code>r</code> | is the node at the current position. |

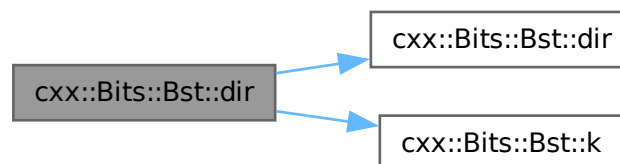
Return values

| | |
|---------------------------|--|
| <code>Direction::L</code> | For left. |
| <code>Direction::R</code> | For right. |
| <code>Direction::N</code> | If <code>l</code> is equal to <code>r</code> . |

Definition at line 129 of file `bst.h`.

References `dir()`, and `k()`.

Here is the call graph for this function:

**15.25.2.4 dir()** [2/2]

```
template<typename Node, typename Get_key, typename Compare>
Dir cxx::Bits::Bst< Node, Get_key, Compare >::dir (
    Key_param_type l,
    Key_param_type r) [inline], [static], [protected]
```

Get the direction to go from `l` to search for `r`.

Parameters

| | |
|----------|-------------------------------------|
| <i>l</i> | is the key to look for. |
| <i>r</i> | is the key at the current position. |

Return values

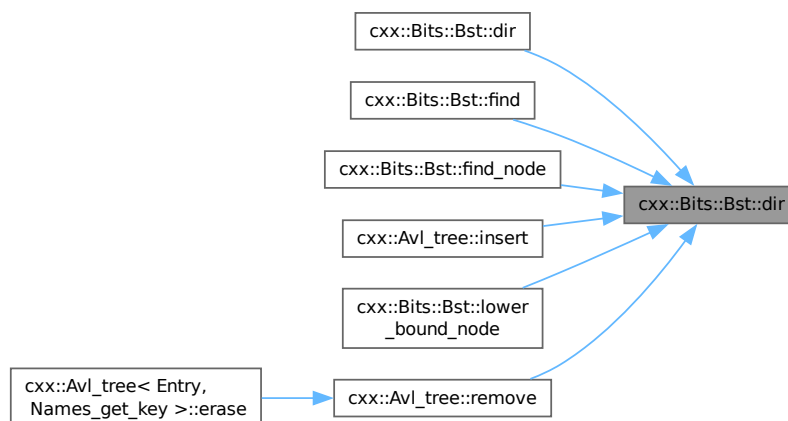
| | |
|-------------------------------------|------------------------------------|
| <i>Direction::L</i> | for left |
| <i>Direction::R</i> | for right |
| <i>Direction::N</i> | if <i>l</i> is equal to <i>r</i> . |

Definition at line 112 of file [bst.h](#).

References [cxx::Bits::Direction::L](#), and [cxx::Bits::Direction::N](#).

Referenced by [dir\(\)](#), [find\(\)](#), [find_node\(\)](#), [cxx::Avl_tree< Node, Get_key, Compare >::insert\(\)](#), [lower_bound_node\(\)](#), and [cxx::Avl_tree< Node, Get_key, Compare >::remove\(\)](#).

Here is the caller graph for this function:

**15.25.2.5 end() [1/2]**

```
template<typename Node, typename Get_key, typename Compare>
Iterator cxx::Bits::Bst< Node, Get_key, Compare >::end () [inline]
```

Get the end marker for the mutable forward iterator.

Returns

The end marker for mutable forward iterator.

Definition at line 189 of file [bst.h](#).

15.25.2.6 end() [2/2]

```
template<typename Node, typename Get_key, typename Compare>
Const_iterator cxx::Bits::Bst< Node, Get_key, Compare >::end () const [inline]
```

Get the end marker for the constant forward iterator.

Returns

The end marker for the constant forward iterator.

Definition at line 178 of file [bst.h](#).

15.25.2.7 find()

```
template<typename Node, typename Get_key, class Compare>
Bst< Node, Get_key, Compare >::Const_iterator cxx::Bits::Bst< Node, Get_key, Compare >::find
(
    Key_param_type key) const [inline]
```

find the node with the given *key*.

Parameters

| | |
|------------|---|
| <i>key</i> | The key value of the element to search. |
|------------|---|

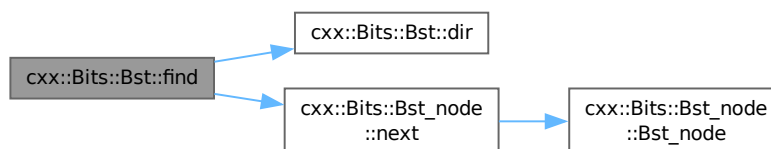
Returns

A valid iterator for the node with the given *key*, or an invalid iterator if *key* was not found.

Definition at line 306 of file [bst.h](#).

References [_head](#), [dir\(\)](#), [cxx::Bits::Direction::L](#), [cxx::Bits::Direction::N](#), and [cxx::Bits::Bst_node::next\(\)](#).

Here is the call graph for this function:



15.25.2.8 find_node()

```
template<typename Node, typename Get_key, class Compare>
Node * cxx::Bits::Bst< Node, Get_key, Compare >::find_node (
    Key_param_type key) const [inline]
```

find the node with the given *key*.

Parameters

| | |
|------------|---|
| <i>key</i> | The key value of the element to search. |
|------------|---|

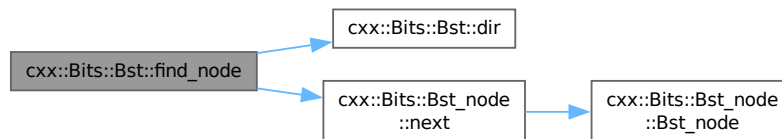
Returns

A pointer to the node with the given *key*, or NULL if *key* was not found.

Definition at line 270 of file [bst.h](#).

References [_head](#), [dir\(\)](#), [cxx::Bits::Direction::N](#), and [cxx::Bits::Bst_node::next\(\)](#).

Here is the call graph for this function:

**15.25.2.9 lower_bound_node()**

```
template<typename Node, typename Get_key, class Compare>
Node * cxx::Bits::Bst< Node, Get_key, Compare >::lower_bound_node (
    Key_param_type key) const [inline]
```

Find the first node with a key not less than the given *key*.

Parameters

| | |
|------------|------------------------------|
| <i>key</i> | The key used for the search. |
|------------|------------------------------|

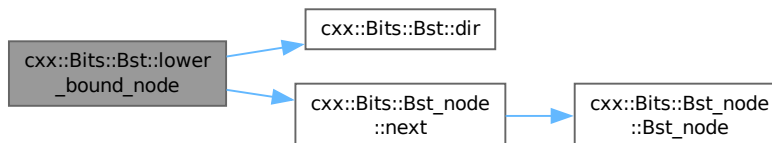
Returns

A pointer to the found node, or `NULL` if no node was found.

Definition at line 286 of file [bst.h](#).

References [_head](#), [dir\(\)](#), [cxx::Bits::Direction::L](#), [cxx::Bits::Direction::N](#), and [cxx::Bits::Bst_node::next\(\)](#).

Here is the call graph for this function:

**15.25.2.10 rbegin() [1/2]**

```
template<typename Node, typename Get_key, typename Compare>
Rev_iterator cxx::Bits::Bst< Node, Get_key, Compare >::rbegin () [inline]
```

Get the mutable backward iterator for the last element of the set.

Returns

The mutable backward iterator for the last element of the set.

Definition at line 206 of file [bst.h](#).

References [head\(\)](#).

Here is the call graph for this function:



15.25.2.11 rbegin() [2/2]

```
template<typename Node, typename Get_key, typename Compare>
Const_rev_iterator cxx::Bits::Bst< Node, Get_key, Compare >::rbegin () const [inline]
```

Get the constant backward iterator for the last element in the set.

Returns

The constant backward iterator for the last element in the set.

Definition at line 195 of file [bst.h](#).

References [head\(\)](#).

Here is the call graph for this function:

**15.25.2.12 remove_all()**

```
template<typename Node, typename Get_key, typename Compare>
template<typename FUNC>
void cxx::Bits::Bst< Node, Get_key, Compare >::remove_all (
    FUNC && callback) [inline]
```

Clear the tree.

Parameters

| | |
|-----------------|---|
| <i>callback</i> | Optional function to be called on each removed element. |
|-----------------|---|

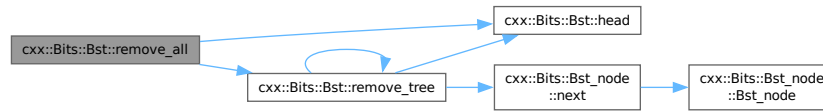
The callback may delete the elements. The function guarantees that the elements are no longer used after the callback has been called.

Definition at line 252 of file [bst.h](#).

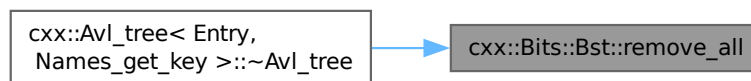
References [_head](#), [head\(\)](#), and [remove_tree\(\)](#).

Referenced by [cxx::Avl_tree< Entry, Names_get_key >::~~Avl_tree\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.25.2.13 remove_tree()

```

template<typename Node, typename Get_key, typename Compare>
template<typename FUNC>
void cxx::Bits::Bst< Node, Get_key, Compare >::remove_tree (
    Bst_node * head,
    FUNC && callback) [inline], [static], [protected]
  
```

Remove all elements in the subtree of head.

Parameters

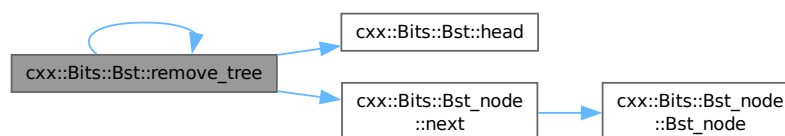
| | |
|-----------------|---|
| <i>head</i> | Head of the the subtree to remove |
| <i>callback</i> | Optional function called on each removed element. |

Definition at line 152 of file [bst.h](#).

References [head\(\)](#), [cxx::Bits::Direction::L](#), [cxx::Bits::Bst_node::next\(\)](#), [cxx::Bits::Direction::R](#), and [remove_tree\(\)](#).

Referenced by [remove_all\(\)](#), and [remove_tree\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.25.2.14 rend() [1/2]

```
template<typename Node, typename Get_key, typename Compare>
Rev_iterator cxx::Bits::Bst< Node, Get_key, Compare >::rend () [inline]
```

Get the end marker for the mutable backward iterator.

Returns

The end marker for mutable backward iterator.

Definition at line 211 of file [bst.h](#).

15.25.2.15 rend() [2/2]

```
template<typename Node, typename Get_key, typename Compare>
Const_rev_iterator cxx::Bits::Bst< Node, Get_key, Compare >::rend () const [inline]
```

Get the end marker for the constant backward iterator.

Returns

The end marker for the constant backward iterator.

Definition at line 200 of file [bst.h](#).

The documentation for this class was generated from the following file:

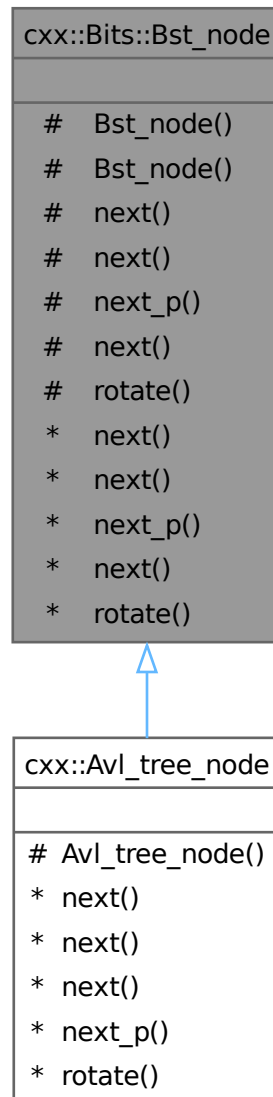
- [l4/cxx/bits/bst.h](#)

15.26 cxx::Bits::Bst_node Class Reference

Basic type of a node in a binary search tree (BST).

```
#include <bst_base.h>
```

Inheritance diagram for cxx::Bits::Bst_node:



Collaboration diagram for cxx::Bits::Bst_node:

| cxx::Bits::Bst_node | |
|---------------------|------------|
| | |
| # | Bst_node() |
| # | Bst_node() |
| # | next() |
| # | next() |
| # | next_p() |
| # | next() |
| # | rotate() |
| * | next() |
| * | next() |
| * | next_p() |
| * | next() |
| * | rotate() |

Protected Member Functions

- **Bst_node** ()
Create uninitialized node.
- **Bst_node** (bool)
Create initialized node.

Static Protected Member Functions

Access to BST linkage.

Provide access to the tree linkage to inherited classes. Inherited nodes, such as AVL nodes should make these methods private via 'using'

- static **Bst_node** * **next** (**Bst_node** const *p, **Direction** d)
Get next node in direction d.
- static void **next** (**Bst_node** *p, **Direction** d, **Bst_node** *n)
Set next node of p in direction d to n.
- static **Bst_node** ** **next_p** (**Bst_node** *p, **Direction** d)
Get pointer to link in direction d.
- template<typename Node>
static Node * **next** (**Bst_node** const *p, **Direction** d)
Get next node in direction d as type Node.
- static void **rotate** (**Bst_node** **t, **Direction** idir)
Rotate subtree t in the opposite direction of idir.

15.26.1 Detailed Description

Basic type of a node in a binary search tree (BST).

Definition at line 70 of file [bst_base.h](#).

The documentation for this class was generated from the following file:

- [l4/cxx/bits/bst_base.h](#)

15.27 cxx::Bits::Direction Struct Reference

The direction to go in a binary search tree.

```
#include <bst_base.h>
```

Collaboration diagram for cxx::Bits::Direction:

| cxx::Bits::Direction |
|---|
| <div> + Direction() + Direction() + Direction() + operator!() + operator==() + operator!=() + operator==() + operator!=() * operator==() * operator!=() * operator==() * operator!=() </div> |

Public Types

- enum [Direction_e](#) { [L](#) = 0 , [R](#) = 1 , [N](#) = 2 }

The literal direction values.

Public Member Functions

- **Direction** ()=default
Uninitialized direction.
- **Direction** ([Direction_e](#) d)
Convert a literal direction ([L](#), [R](#), [N](#)) to an object.
- **Direction** (bool b)
Convert a boolean to a direction (false == [L](#), true == [R](#)).
- **Direction operator!** () const
Negate the direction.

Comparison operators (equality and inequality)

- bool **operator==** ([Direction_e](#) o) const
Compare for equality.
- bool **operator!=** ([Direction_e](#) o) const
Compare for inequality.
- bool **operator==** ([Direction](#) o) const
Compare for equality.
- bool **operator!=** ([Direction](#) o) const
Compare for inequality.

15.27.1 Detailed Description

The direction to go in a binary search tree.

Definition at line 28 of file [bst_base.h](#).

15.27.2 Member Enumeration Documentation

15.27.2.1 Direction_e

```
enum cxx::Bits::Direction::Direction_e
```

The literal direction values.

Enumerator

| | |
|---|------------------------|
| L | Go to the left child. |
| R | Go to the right child. |
| N | Stop. |

Definition at line 31 of file [bst_base.h](#).

15.27.3 Member Function Documentation

15.27.3.1 `operator!()`

```
Direction cxx::Bits::Direction::operator! () const [inline]
```

Negate the direction.

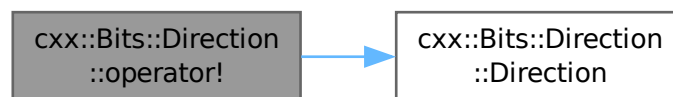
Note

This is only defined for a current value of [L](#) or [R](#)

Definition at line [52](#) of file [bst_base.h](#).

References [Direction\(\)](#).

Here is the call graph for this function:



The documentation for this struct was generated from the following file:

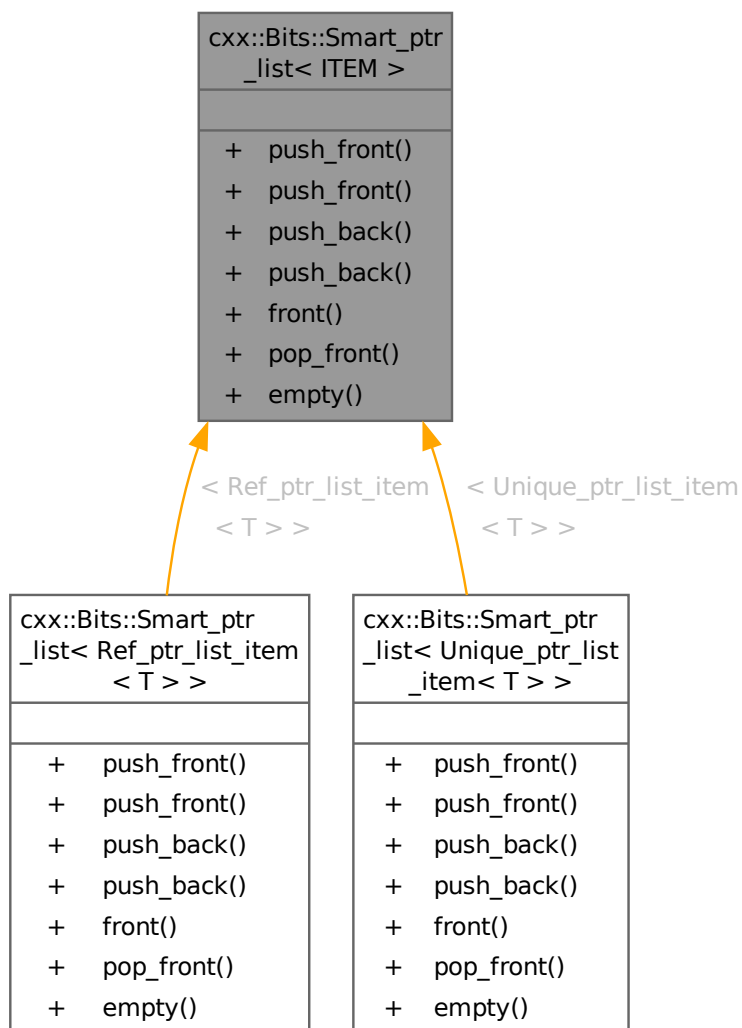
- [I4/cxx/bits/bst_base.h](#)

15.28 `cxx::Bits::Smart_ptr_list< ITEM >` Class Template Reference

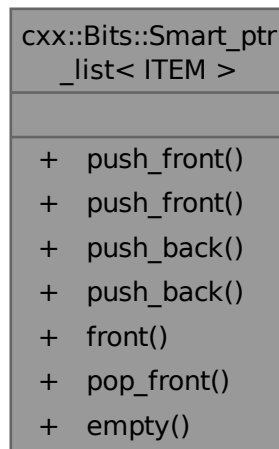
[List](#) of smart-pointer-managed objects.

```
#include <smart_ptr_list.h>
```

Inheritance diagram for cxx::Bits::Smart_ptr_list< ITEM >:



Collaboration diagram for `cxx::Bits::Smart_ptr_list< ITEM >`:



Public Member Functions

- void **push_front** (Next_type &&e)
Add an element to the front of the list.
- void **push_front** (Next_type const &e)
Add an element to the front of the list.
- void **push_back** (Next_type &&e)
Add an element at the end of the list.
- void **push_back** (Next_type const &e)
Add an element at the end of the list.
- Value_type * **front** () const
Return a pointer to the first element in the list.
- Next_type **pop_front** ()
Remove the element in front of the list and return it.
- bool **empty** () const
Check if the list is empty.

15.28.1 Detailed Description

```
template<typename ITEM>
class cxx::Bits::Smart_ptr_list< ITEM >
```

[List](#) of smart-pointer-managed objects.

Template Parameters

| | |
|-------------|-------------------------|
| <i>ITEM</i> | Type of the list items. |
|-------------|-------------------------|

The list is implemented as a single-linked list connected via smart pointers, so that they are automatically cleaned up when they are removed from the list.

Definition at line 46 of file [smart_ptr_list.h](#).

15.28.2 Member Function Documentation

15.28.2.1 `pop_front()`

```
template<typename ITEM>
Next_type cxx::Bits::Smart_ptr_list< ITEM >::pop_front () [inline]
```

Remove the element in front of the list and return it.

Returns

The element that was previously in front of the list as a managed pointer or a nullptr-equivalent when the list was already empty.

Definition at line 149 of file [smart_ptr_list.h](#).

The documentation for this class was generated from the following file:

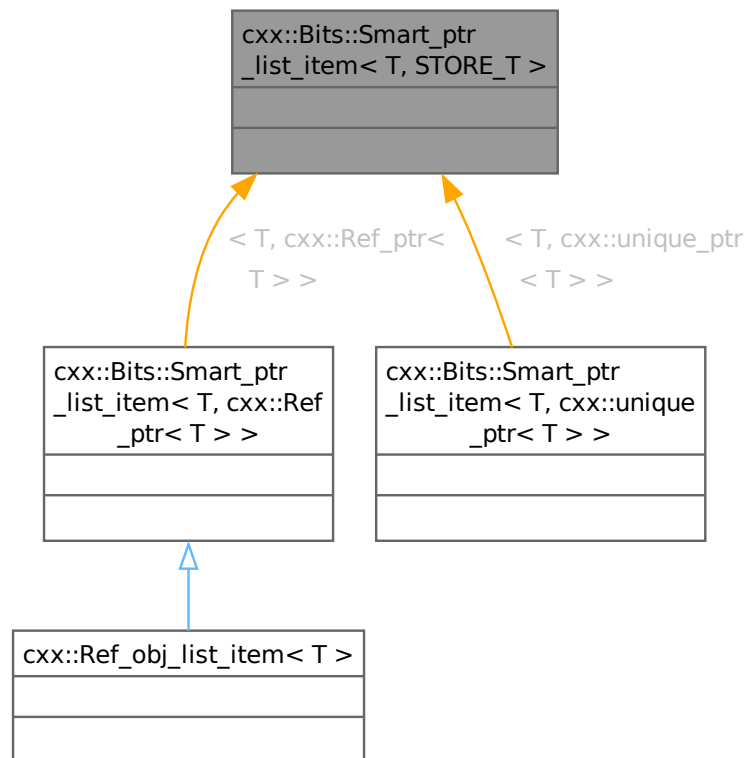
- [I4/cxx/bits/smart_ptr_list.h](#)

15.29 `cxx::Bits::Smart_ptr_list_item< T, STORE_T >` Class Template Reference

[List](#) item for an arbitrary item in a [Smart_ptr_list](#).

```
#include <smart_ptr_list.h>
```

Inheritance diagram for `cxx::Bits::Smart_ptr_list_item< T, STORE_T >`:



Collaboration diagram for `cxx::Bits::Smart_ptr_list_item< T, STORE_T >`:



15.29.1 Detailed Description

```

template<typename T, typename STORE_T>
class cxx::Bits::Smart_ptr_list_item< T, STORE_T >

```

[List](#) item for an arbitrary item in a [Smart_ptr_list](#).

Template Parameters

| | |
|----------------------|--|
| <i>T</i> | Type of object to be stored in the list. |
| <i>STORE↔ _T</i> | Storage type for pointer to next item. The class must implement a <code>get()</code> function that returns a pointer to the stored object and destroy the stored object when the item goes out of scope. |

Definition at line 27 of file [smart_ptr_list.h](#).

The documentation for this class was generated from the following file:

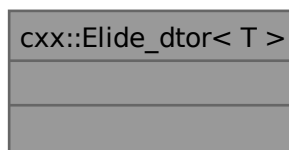
- [I4/cxx/bits/smart_ptr_list.h](#)

15.30 `cxx::Elide_dtor< T >` Class Template Reference

Wrapper class to remove destructor calls.

```
#include <elide_dtor>
```

Collaboration diagram for `cxx::Elide_dtor< T >`:



15.30.1 Detailed Description

```
template<typename T>
class cxx::Elide_dtor< T >
```

Wrapper class to remove destructor calls.

Use this class for global or static local objects that shall not be destructed. This can save some code size if the destructor of `T` is not trivial. It also prevents calls to `atexit()` at runtime.

Attention

Be careful if you use [Elide_dtor](#) in the global scope. Even if `T` is trivially constructible, the wrapper will force the compiler to emit a static initializer!

Definition at line 28 of file [elide_dtor](#).

The documentation for this class was generated from the following file:

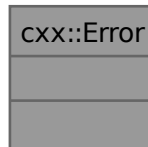
- [I4/cxx/elide_dtor](#)

15.31 cxx::Error Class Reference

[Error](#) value.

```
#include <result>
```

Collaboration diagram for cxx::Error:



15.31.1 Detailed Description

[Error](#) value.

Value class to make error value types explicit. Participates in the construction of [Result](#) objects.

Definition at line 19 of file [result](#).

The documentation for this class was generated from the following file:

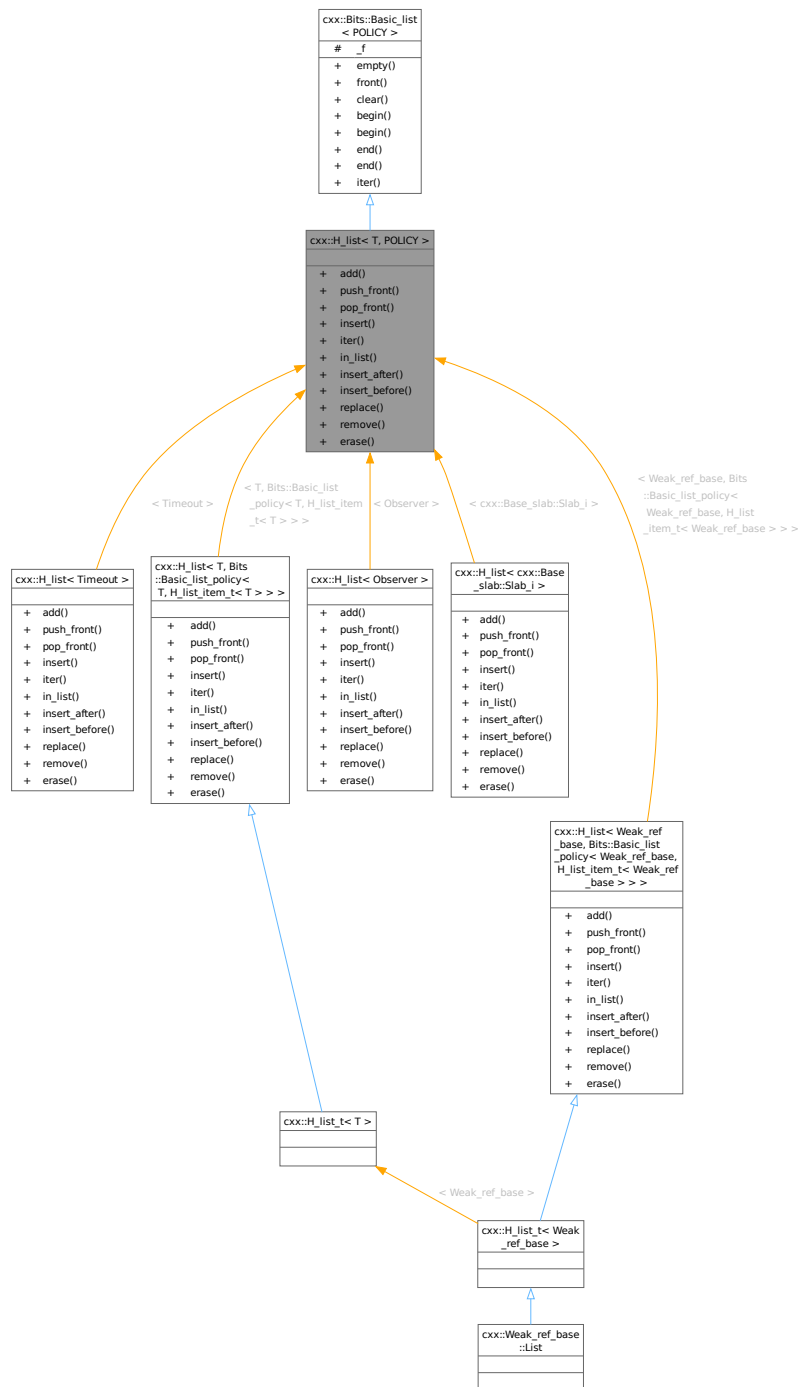
- I4/cxx/result

15.32 cxx::H_list< T, POLICY > Class Template Reference

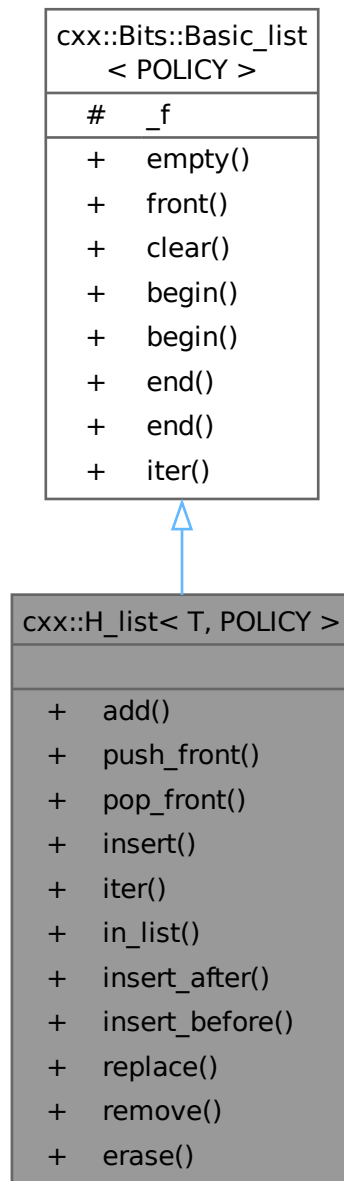
General double-linked list of unspecified [cxx::H_list_item](#) elements.

```
#include <hlist>
```


Inheritance diagram for cxx::H_list< T, POLICY >:



Collaboration diagram for `cxx::H_list< T, POLICY >`:



Public Member Functions

- void **add** (T *e)
Add element to the front of the list.
- void **push_front** (T *e)
Add element to the front of the list.
- T * **pop_front** ()
Remove and return the head element of the list.
- Iterator **insert** (T *e, Iterator const &pred)
Insert an element at the iterator position.

Public Member Functions inherited from `cxx::Bits::Basic_list< POLICY >`

- `bool empty () const`
Check if the list is empty.
- `Value_type front () const`
Return the first element in the list.
- `void clear ()`
Remove all elements from the list.
- `Iterator begin ()`
Return an iterator to the beginning of the list.
- `Const_iterator begin () const`
Return a const iterator to the beginning of the list.
- `Const_iterator end () const`
Return a const iterator to the end of the list.
- `Iterator end ()`
Return an iterator to the end of the list.

Static Public Member Functions

- `static Iterator iter (T *c)`
Return an iterator for an arbitrary list element.
- `static bool in_list (T const *e)`
Check if the given element is currently part of a list.
- `static Iterator insert_after (T *e, Iterator const &pred)`
Insert an element after the iterator position.
- `static void insert_before (T *e, Iterator const &succ)`
Insert an element before the iterator position.
- `static void replace (T *p, T *e)`
Replace an element in a list with a new element.
- `static void remove (T *e)`
Remove the given element from its list.
- `static Iterator erase (Iterator const &e)`
Remove the element at the given iterator position.

Static Public Member Functions inherited from `cxx::Bits::Basic_list< POLICY >`

- `static Const_iterator iter (Const_value_type c)`
Return a const iterator that begins at the given element.

Additional Inherited Members**Protected Attributes inherited from `cxx::Bits::Basic_list< POLICY >`**

- `POLICY::Head_type _f`
Pointer to front of the list.

15.32.1 Detailed Description

```
template<typename T, typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
class cxx::H_list< T, POLICY >
```

General double-linked list of unspecified [cxx::H_list_item](#) elements.

Most of the time, you want to use [H_list_t](#).

Definition at line 69 of file [hlist](#).

15.32.2 Member Function Documentation

15.32.2.1 erase()

```
template<typename T, typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
Iterator cxx::H_list< T, POLICY >::erase (
    Iterator const & e) [inline], [static]
```

Remove the element at the given iterator position.

Parameters

| | |
|----------|---|
| <i>e</i> | Iterator pointing to the element to be removed. Must not point to end() . |
|----------|---|

Returns

New iterator pointing to the element after the removed one.

Note

The hlist implementation guarantees that the original iterator is still valid after the element has been removed. In fact, the iterator returned is the same as the one supplied in the *e* parameter.

Definition at line 236 of file [hlist](#).

15.32.2.2 insert()

```
template<typename T, typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
Iterator cxx::H_list< T, POLICY >::insert (
    T * e,
    Iterator const & pred) [inline]
```

Insert an element at the iterator position.

Parameters

| | |
|-------------|--|
| <i>e</i> | New Element to be inserted |
| <i>pred</i> | Iterator pointing to the element after which the element will be inserted. If <code>end()</code> is given, the element will be inserted at the beginning of the queue. |

Returns

Iterator pointing to the newly inserted element.

Definition at line 133 of file [hlist](#).

15.32.2.3 `insert_after()`

```
template<typename T, typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
Iterator cxx::H_list< T, POLICY >::insert_after (
    T * e,
    Iterator const & pred) [inline], [static]
```

Insert an element after the iterator position.

Parameters

| | |
|-------------|---|
| <i>e</i> | New element to be inserted. |
| <i>pred</i> | Iterator pointing to the element after which the element will be inserted. Must not be <code>end()</code> . |

Returns

Iterator pointing to the newly inserted element.

Precondition

The list must not be empty.

Definition at line 160 of file [hlist](#).

15.32.2.4 `insert_before()`

```
template<typename T, typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
void cxx::H_list< T, POLICY >::insert_before (
    T * e,
    Iterator const & succ) [inline], [static]
```

Insert an element before the iterator position.

Parameters

| | |
|-------------|---|
| <i>e</i> | New element to be inserted. |
| <i>succ</i> | Iterator pointing to the element before which the element will be inserted. Must not be end() . |

Definition at line 180 of file [hlist](#).

15.32.2.5 `iter()`

```
template<typename T, typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
Iterator cxx::H\_list< T, POLICY >::iter (
    T * c)    [inline], [static]
```

Return an iterator for an arbitrary list element.

Parameters

| | |
|----------|--|
| <i>c</i> | List element to start the iteration. |
|----------|--|

Returns

A mutable forward iterator.

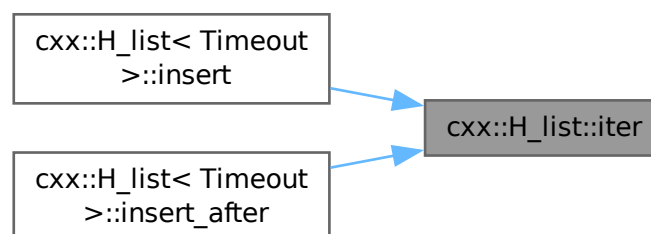
Precondition

The element must be in a list.

Definition at line 93 of file [hlist](#).

Referenced by [cxx::H_list< Timeout >::insert\(\)](#), and [cxx::H_list< Timeout >::insert_after\(\)](#).

Here is the caller graph for this function:



15.32.2.6 pop_front()

```
template<typename T, typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
T * cxx::H_list< T, POLICY >::pop_front () [inline]
```

Remove and return the head element of the list.

Precondition

The list must not be empty or the behaviour will be undefined.

Definition at line 116 of file [hlist](#).

15.32.2.7 remove()

```
template<typename T, typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
void cxx::H_list< T, POLICY >::remove (
    T * e) [inline], [static]
```

Remove the given element from its list.

Parameters

| | |
|----------|---|
| <i>e</i> | Element to be removed. Must be in a list. |
|----------|---|

Definition at line 220 of file [hlist](#).

Referenced by [cxx::H_list< Timeout >::pop_front\(\)](#).

Here is the caller graph for this function:



15.32.2.8 replace()

```
template<typename T, typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
void cxx::H_list< T, POLICY >::replace (
    T * p,
    T * e) [inline], [static]
```

Replace an element in a list with a new element.

Parameters

| | |
|----------|---|
| <i>p</i> | Element in list to be replaced. |
| <i>e</i> | Replacement element, must not yet be in a list. |

Precondition

p and *e* must not be NULL.

After the operation the *p* element is no longer in the list and may be reused.

Definition at line 204 of file [hlist](#).

The documentation for this class was generated from the following file:

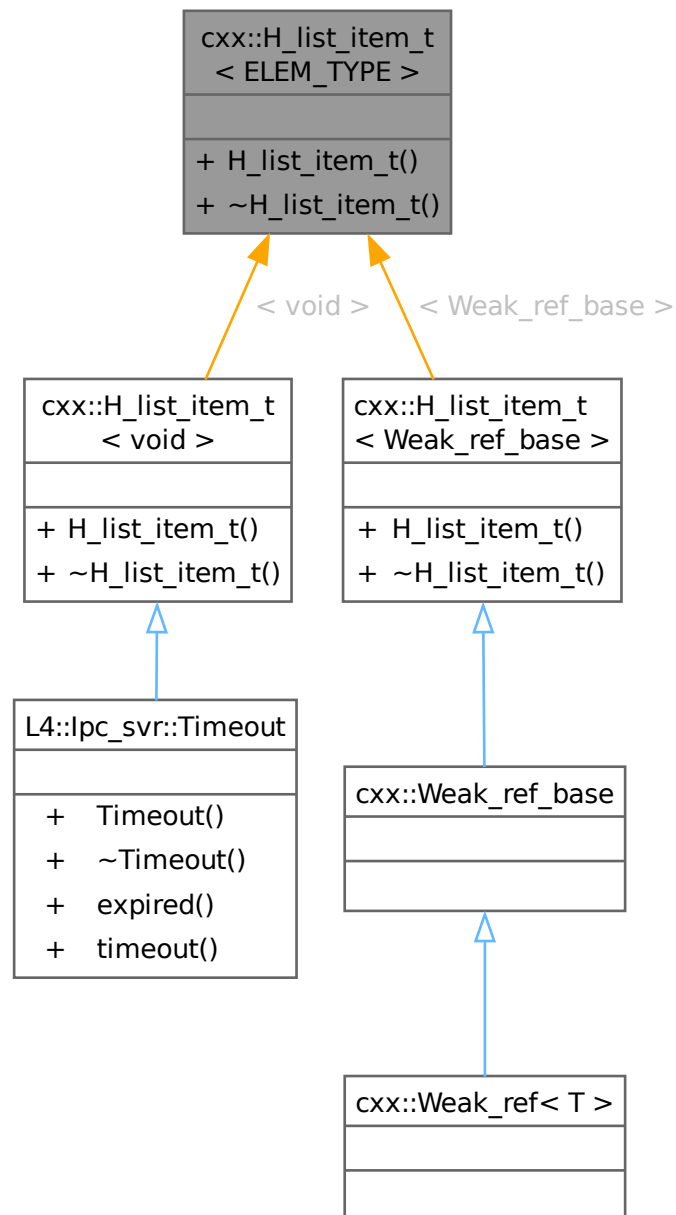
- I4/cxx/hlist

15.33 cxx::H_list_item_t< ELEM_TYPE > Class Template Reference

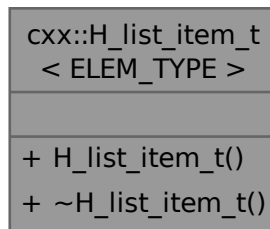
Basic element type for a double-linked [H_list](#).

```
#include <hlist>
```


Inheritance diagram for cxx::H_list_item_t< ELEM_TYPE >:



Collaboration diagram for `cxx::H_list_item_t< ELEM_TYPE >`:



Public Member Functions

- [`H_list_item_t\(\)`](#)
Constructor.
- [`~H_list_item_t\(\)`](#) noexcept
Destructor.

15.33.1 Detailed Description

```
template<typename ELEM_TYPE>
class cxx::H_list_item_t< ELEM_TYPE >
```

Basic element type for a double-linked [H_list](#).

Template Parameters

| | |
|------------------------|---------------------------------|
| <code>ELEM_TYPE</code> | Base class of the list element. |
|------------------------|---------------------------------|

Definition at line 22 of file [hlist](#).

15.33.2 Constructor & Destructor Documentation

15.33.2.1 `H_list_item_t()`

```
template<typename ELEM_TYPE>
cxx::H_list_item_t< ELEM_TYPE >::H_list_item_t () [inline]
```

Constructor.

Creates an element that is not in any list.

Definition at line 30 of file [hlist](#).

15.33.2.2 `~H_list_item_t()`

```
template<typename ELEM_TYPE>
cxx::H_list_item_t< ELEM_TYPE >::~~H_list_item_t () [inline], [noexcept]
```

Destructor.

Automatically removes the element from any list it still might be enchainned in.

Definition at line 37 of file [hlist](#).

The documentation for this class was generated from the following file:

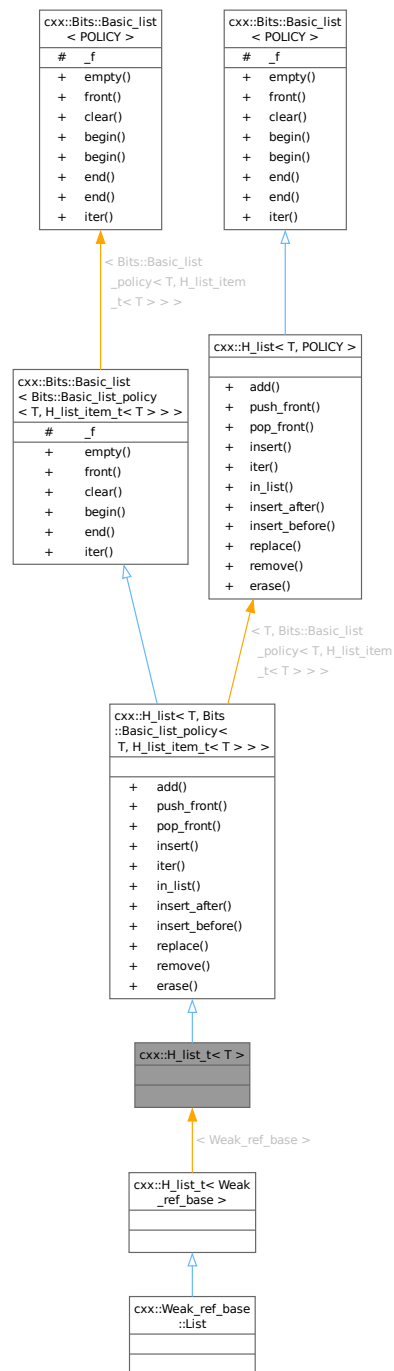
- `I4/cxx/hlist`

15.34 `cxx::H_list_t< T >` Struct Template Reference

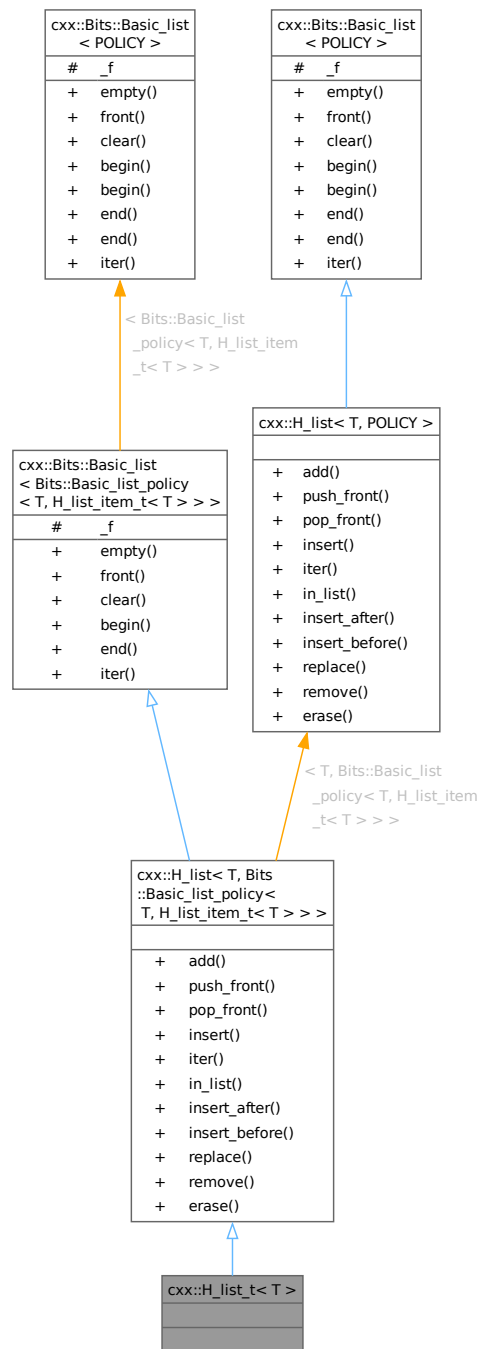
Double-linked list of typed [H_list_item_t](#) elements.

```
#include <hlist>
```

Inheritance diagram for `cxx::H_list_t< T >`:



Collaboration diagram for cxx::H_list_t< T >:



Additional Inherited Members

Public Member Functions inherited from

`cxx::H_list< T, Bits::Basic_list_policy< T, H_list_item_t< T > > >`

- `void add (T *e)`

- *Add element to the front of the list.*
- void **push_front** (T *e)
Add element to the front of the list.
- T * **pop_front** ()
Remove and return the head element of the list.
- Iterator **insert** (T *e, Iterator const &pred)
Insert an element at the iterator position.

Public Member Functions inherited from

cxx::Bits::Basic_list< **Bits::Basic_list_policy**< T, H_list_item_t< T > > >

- bool **empty** () const
Check if the list is empty.
- Value_type **front** () const
Return the first element in the list.
- void **clear** ()
Remove all elements from the list.
- Iterator **begin** ()
Return an iterator to the beginning of the list.
- Const_iterator **end** () const
Return a const iterator to the end of the list.

Static Public Member Functions inherited from

cxx::H_list< T, **Bits::Basic_list_policy**< T, H_list_item_t< T > > >

- static Iterator **iter** (T *c)
Return an iterator for an arbitrary list element.
- static bool **in_list** (T const *e)
Check if the given element is currently part of a list.
- static Iterator **insert_after** (T *e, Iterator const &pred)
Insert an element after the iterator position.
- static void **insert_before** (T *e, Iterator const &succ)
Insert an element before the iterator position.
- static void **replace** (T *p, T *e)
Replace an element in a list with a new element.
- static void **remove** (T *e)
Remove the given element from its list.
- static Iterator **erase** (Iterator const &e)
Remove the element at the given iterator position.

Static Public Member Functions inherited from

cxx::Bits::Basic_list< **Bits::Basic_list_policy**< T, H_list_item_t< T > > >

- static Const_iterator **iter** (Const_value_type c)
Return a const iterator that begins at the given element.

Protected Attributes inherited from**`cxx::Bits::Basic_list< Bits::Basic_list_policy< T, H_list_item_t< T > > >`**

- `Bits::Basic_list_policy< T, H_list_item_t< T > >::Head_type _f`

*Pointer to front of the list.***15.34.1 Detailed Description**

```
template<typename T>
struct cxx::H_list_t< T >
```

Double-linked list of typed `H_list_item_t` elements.

Note

`H_lists` are not self-cleaning. Elements that are still chained during destruction are not removed and will therefore be in an undefined state after the destruction.

Definition at line 248 of file [hlist](#).

The documentation for this struct was generated from the following file:

- `I4/cxx/hlist`

15.35 `cxx::List< D, Alloc >` Class Template Reference

Doubly linked list, with internal allocation.

```
#include <list>
```

Collaboration diagram for `cxx::List< D, Alloc >`:

| <code>cxx::List< D, Alloc ></code> |
|---|
| <ul style="list-style-type: none"> + <code>push_back()</code> + <code>push_front()</code> + <code>remove()</code> + <code>size()</code> + <code>operator[]()</code> + <code>operator[]()</code> + <code>items()</code> |

Data Structures

- class [Iter](#)
Iterator.

Public Member Functions

- void **push_back** (D const &d) noexcept
Add element at the end of the list.
- void **push_front** (D const &d) noexcept
Add element at the beginning of the list.
- void **remove** ([Iter](#) const &i) noexcept
Remove element pointed to by the iterator.
- unsigned long **size** () const noexcept
Get the length of the list.
- D const & **operator[]** (unsigned long idx) const noexcept
Random access.
- D & **operator[]** (unsigned long idx) noexcept
Random access.
- [Iter](#) **items** () noexcept
Get iterator for the list elements.

15.35.1 Detailed Description

```
template<typename D, template< typename A > class Alloc = New_allocator>
class cxx::List< D, Alloc >
```

Doubly linked list, with internal allocation.

Container for items of type D, implemented by a doubly linked list. Alloc defines the allocator policy.

Definition at line [326](#) of file [list](#).

15.35.2 Member Function Documentation

15.35.2.1 **operator[]**() [1/2]

```
template<typename D, template< typename A > class Alloc = New_allocator>
D const & cxx::List< D, Alloc >::operator[] (
    unsigned long idx) const [inline], [noexcept]
```

Random access.

Complexity is O(n).

Definition at line [396](#) of file [list](#).

15.35.2.2 operator[]() [2/2]

```
template<typename D, template< typename A > class Alloc = New_allocator>
D & cxx::List< D, Alloc >::operator[] (
    unsigned long idx) [inline], [noexcept]
```

Random access.

Complexity is O(n).

Definition at line 400 of file [list](#).

The documentation for this class was generated from the following file:

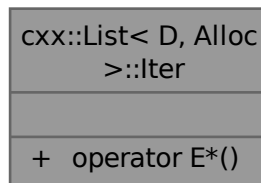
- l4/cxx/list

15.36 cxx::List< D, Alloc >::Iter Class Reference

Iterator.

```
#include <list>
```

Collaboration diagram for cxx::List< D, Alloc >::Iter:



Public Member Functions

- **operator E* ()** const noexcept
operator for testing validity (syntactically equal to pointers)

15.36.1 Detailed Description

```
template<typename D, template< typename A > class Alloc = New_allocator>
class cxx::List< D, Alloc >::Iter
```

Iterator.

Forward and backward iterable.

Definition at line 346 of file [list](#).

The documentation for this class was generated from the following file:

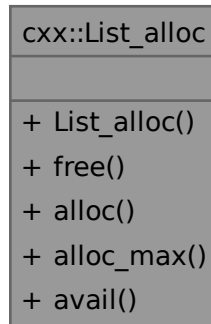
- l4/cxx/list

15.37 cxx::List_alloc Class Reference

Standard list-based allocator.

```
#include <list_alloc>
```

Collaboration diagram for cxx::List_alloc:



Public Member Functions

- [List_alloc](#) ()
Initializes an empty list allocator.
- void [free](#) (void *block, unsigned long size, bool initial_free=false)
Return a free memory block to the allocator.
- void * [alloc](#) (unsigned long size, unsigned long align, unsigned long lower=0, unsigned long upper=~0UL)
Allocate a memory block.
- void * [alloc_max](#) (unsigned long min, unsigned long *max, unsigned long align, unsigned granularity, unsigned long lower=0, unsigned long upper=~0UL)
Allocate a memory block of $min \leq size \leq max$.
- unsigned long [avail](#) () const
Get the amount of available memory.

15.37.1 Detailed Description

Standard list-based allocator.

Definition at line 22 of file [list_alloc](#).

15.37.2 Constructor & Destructor Documentation

15.37.2.1 List_alloc()

```
cxx::List_alloc::List_alloc () [inline]
```

Initializes an empty list allocator.

Note

To initialize the allocator with available memory use the [free\(\)](#) function.

Definition at line 47 of file [list_alloc](#).

15.37.3 Member Function Documentation

15.37.3.1 alloc()

```
void * cxx::List_alloc::alloc (
    unsigned long size,
    unsigned long align,
    unsigned long lower = 0,
    unsigned long upper = ~0UL) [inline]
```

Allocate a memory block.

Parameters

| | |
|--------------|---|
| <i>size</i> | Size of the memory block. |
| <i>align</i> | Alignment constraint. |
| <i>lower</i> | Lower bound of the physical region the memory block should be allocated from. |
| <i>upper</i> | Upper bound of the physical region the memory block should be allocated from, value is inclusive. |

Returns

Pointer to memory block

Precondition

```
0 < size <= ~0UL - 32.
```

Definition at line 390 of file [list_alloc](#).

15.37.3.2 alloc_max()

```
void * cxx::List_alloc::alloc_max (
    unsigned long min,
    unsigned long * max,
    unsigned long align,
    unsigned granularity,
    unsigned long lower = 0,
    unsigned long upper = ~0UL) [inline]
```

Allocate a memory block of $\text{min} \leq \text{size} \leq \text{max}$.

Parameters

| | | |
|----------------|--------------------|---|
| | <i>min</i> | Minimal size to allocate (in bytes). |
| <i>in, out</i> | <i>max</i> | Maximum size to allocate (in bytes). The actual allocated size is returned here. |
| | <i>align</i> | Alignment constraint. |
| | <i>granularity</i> | Granularity to use for the allocation (power of 2). |
| | <i>lower</i> | Lower bound of the physical region the memory block should be allocated from. |
| | <i>upper</i> | Upper bound of the physical region the memory block should be allocated from, value is inclusive. |

Returns

Pointer to memory block

Precondition

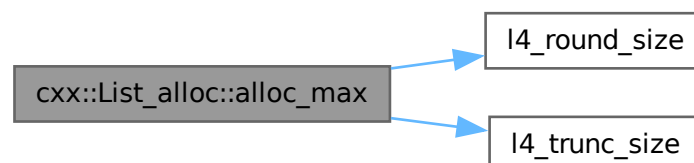
$0 < \text{min} \leq \sim 0\text{UL} - 32$.

$0 < \text{max}$.

Definition at line 270 of file [list_alloc](#).

References [l4_round_size\(\)](#), and [l4_trunc_size\(\)](#).

Here is the call graph for this function:



15.37.3.3 avail()

```
unsigned long cxx::List_alloc::avail () const [inline]
```

Get the amount of available memory.

Returns

Available memory in bytes

Definition at line 478 of file [list_alloc](#).

15.37.3.4 free()

```
void cxx::List_alloc::free (
    void * block,
    unsigned long size,
    bool initial_free = false) [inline]
```

Return a free memory block to the allocator.

Parameters

| | |
|---------------------|--|
| <i>block</i> | Pointer to memory block. |
| <i>size</i> | Size of memory block. |
| <i>initial_free</i> | Set to true for putting fresh memory to the allocator. This will enforce alignment on that memory. |

Precondition

block must not be NULL.

$2 * \text{sizeof}(\text{void} *) \leq \text{size} \leq \sim 0\text{UL} - 32$.

Definition at line 229 of file [list_alloc](#).

The documentation for this class was generated from the following file:

- [I4/cxx/list_alloc](#)

15.38 cxx::List_item Class Reference

Basic list item.

```
#include <list>
```

Collaboration diagram for cxx::List_item:

| cxx::List_item |
|--|
| <ul style="list-style-type: none"> + get_prev_item() + get_next_item() + insert_prev_item() + insert_next_item() + remove_me() + push_back() + push_front() + remove() |

Data Structures

- class [Iter](#)
Iterator for a list of ListItem-s.
- class [T_iter](#)
Iterator for derived classes from ListItem.

Public Member Functions

- List_item * **get_prev_item** () const noexcept
Get previous item.
- List_item * **get_next_item** () const noexcept
Get next item.
- void **insert_prev_item** (List_item *p) noexcept
Insert item p before this item.
- void **insert_next_item** (List_item *p) noexcept
Insert item p after this item.
- void **remove_me** () noexcept
Remove this item from the list.

Static Public Member Functions

- `template<typename C, typename N>`
`static C * push_back (C *head, N *p) noexcept`
Append item to a list.
- `template<typename C, typename N>`
`static C * push_front (C *head, N *p) noexcept`
Prepend item to a list.
- `template<typename C, typename N>`
`static C * remove (C *head, N *p) noexcept`
Remove item from a list.

15.38.1 Detailed Description

Basic list item.

Basic item that can be member of a doubly linked, cyclic list.

Definition at line [26](#) of file [list](#).

15.38.2 Member Function Documentation

15.38.2.1 [push_back\(\)](#)

```
template<typename C, typename N>
C * cxx::List_item::push_back (
    C * head,
    N * p) [inline], [static], [noexcept]
```

Append item to a list.

Convenience function for empty-head corner case.

Parameters

| | |
|-------------|-----------------------------------|
| <i>head</i> | Pointer to the current list head. |
| <i>p</i> | Pointer to new item. |

Returns

the pointer to the new head.

Definition at line [240](#) of file [list](#).

Referenced by [cxx::List< D, Alloc >::push_back\(\)](#).

Here is the caller graph for this function:



15.38.2.2 push_front()

```
template<typename C, typename N>
C * cxx::List_item::push_front (
    C * head,
    N * p) [inline], [static], [noexcept]
```

Prepend item to a list.

Convenience function for empty-head corner case.

Parameters

| | |
|-------------|-----------------------------------|
| <i>head</i> | pointer to the current list head. |
| <i>p</i> | pointer to new item. |

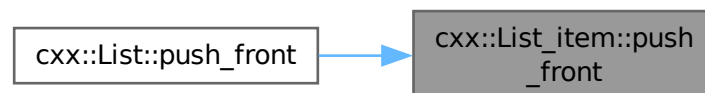
Returns

the pointer to the new head.

Definition at line 251 of file [list](#).

Referenced by [cxx::List< D, Alloc >::push_front\(\)](#).

Here is the caller graph for this function:



15.38.2.3 remove()

```
template<typename C, typename N>
C * cxx::List_item::remove (
    C * head,
    N * p) [inline], [static], [noexcept]
```

Remove item from a list.

Convenience function for remove-head corner case.

Parameters

| | |
|-------------|-----------------------------------|
| <i>head</i> | pointer to the current list head. |
| <i>p</i> | pointer to the item to remove. |

Returns

the pointer to the new head.

Definition at line 261 of file [list](#).

Referenced by [cxx::List< D, Alloc >::remove\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

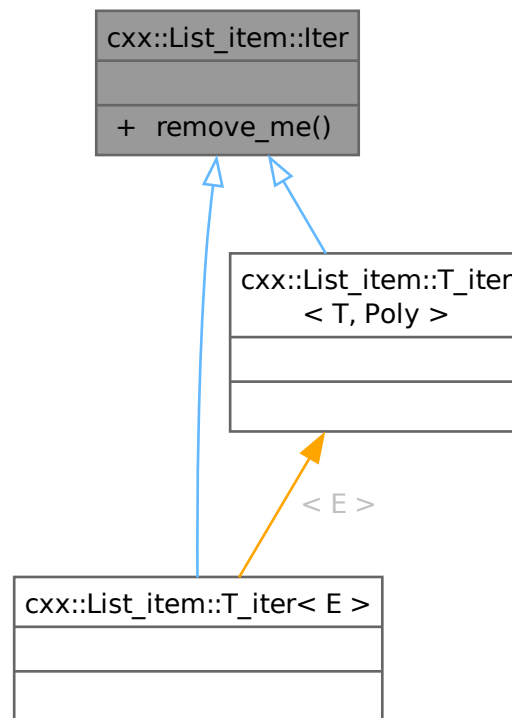
- [l4/cxx/list](#)

15.39 cxx::List_item::Iter Class Reference

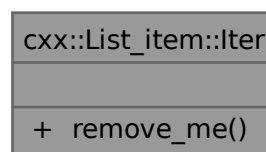
Iterator for a list of ListItem-s.

```
#include <list>
```

Inheritance diagram for `cxx::List_item::Iter`:



Collaboration diagram for `cxx::List_item::Iter`:



Public Member Functions

- `List_item * remove_me () noexcept`
Remove item pointed to by iterator, and return pointer to element.

15.39.1 Detailed Description

Iterator for a list of ListItem-s.

The Iterator iterates till it finds the first element again.

Definition at line 34 of file [list](#).

The documentation for this class was generated from the following file:

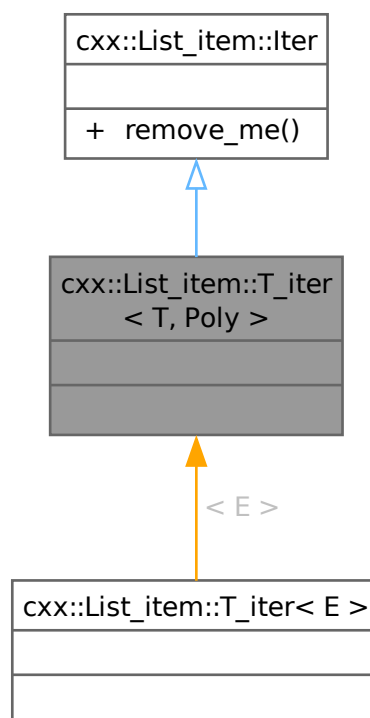
- l4/cxx/list

15.40 cxx::List_item::T_iter< T, Poly > Class Template Reference

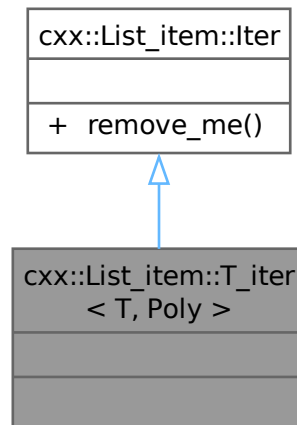
Iterator for derived classes from ListItem.

```
#include <list>
```

Inheritance diagram for cxx::List_item::T_iter< T, Poly >:



Collaboration diagram for `cxx::List_item::T_iter< T, Poly >`:



Additional Inherited Members

Public Member Functions inherited from `cxx::List_item::Iter`

- `List_item * remove_me () noexcept`
Remove item pointed to by iterator, and return pointer to element.

15.40.1 Detailed Description

```
template<typename T, bool Poly = false>
class cxx::List_item::T_iter< T, Poly >
```

Iterator for derived classes from `ListItem`.

Allows direct access to derived classes by `*` operator.

Example: `class Foo : public ListItem { public: typedef T_iter<Foo> Iter; ... };`

Definition at line 108 of file `list`.

The documentation for this class was generated from the following file:

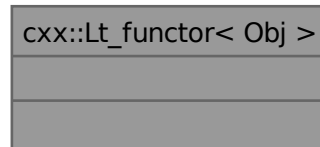
- `I4/cxx/list`

15.41 `cxx::Lt_functor< Obj >` Struct Template Reference

Generic comparator class that defaults to the less-than operator.

```
#include <std_ops>
```

Collaboration diagram for `cxx::Lt_functor< Obj >`:



15.41.1 Detailed Description

```
template<typename Obj>  
struct cxx::Lt_functor< Obj >
```

Generic comparator class that defaults to the less-than operator.

Definition at line 18 of file [std_ops](#).

The documentation for this struct was generated from the following file:

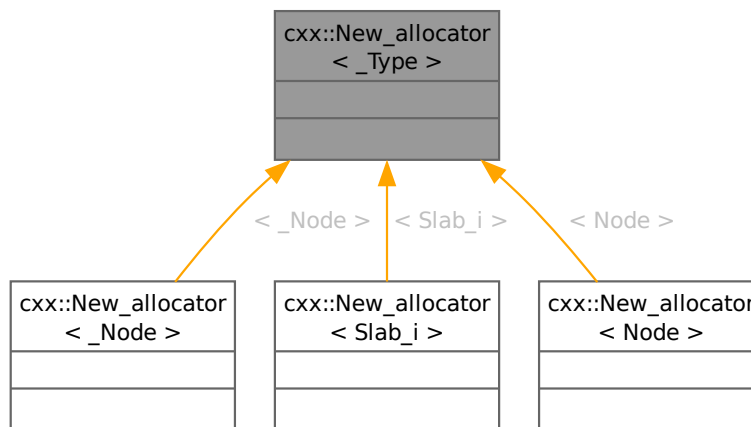
- `I4/cxx/std_ops`

15.42 `cxx::New_allocator< _Type >` Class Template Reference

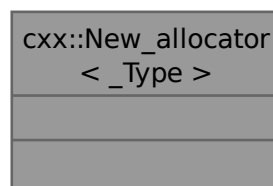
Standard allocator based on `operator new ()`.

```
#include <std_alloc>
```

Inheritance diagram for `cxx::New_allocator<_Type>`:



Collaboration diagram for `cxx::New_allocator<_Type>`:



15.42.1 Detailed Description

```

template<typename _Type>
class cxx::New_allocator<_Type>

```

Standard allocator based on `operator new ()`.

This allocator is the default allocator used for the *cxx Containers*, such as `cxx::Avl_set` and `cxx::Avl_map`, to allocate the internal data structures.

Definition at line 56 of file `std_alloc`.

The documentation for this class was generated from the following file:

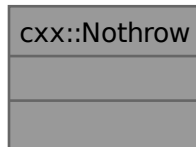
- `I4/cxx/std_alloc`

15.43 `cxx::Nothrow` Class Reference

Helper type to distinguish the `operator new` version that does not throw exceptions.

```
#include <std_alloc>
```

Collaboration diagram for `cxx::Nothrow`:



15.43.1 Detailed Description

Helper type to distinguish the `operator new` version that does not throw exceptions.

Definition at line 19 of file [std_alloc](#).

The documentation for this class was generated from the following file:

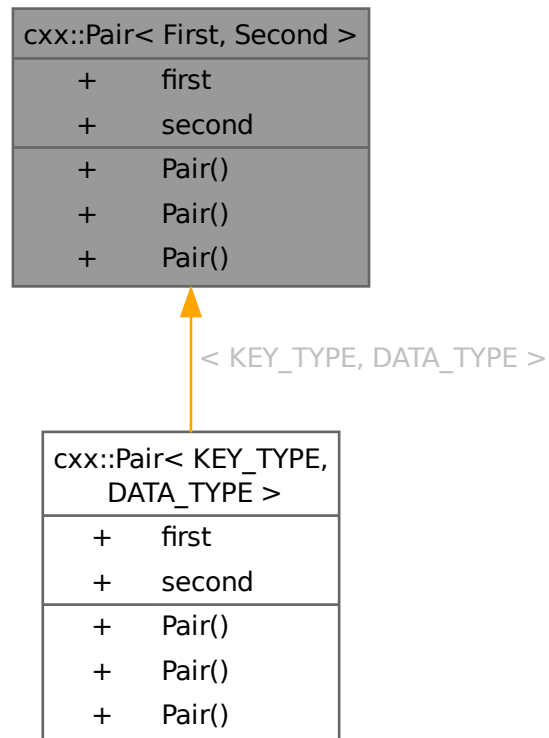
- `I4/cxx/std_alloc`

15.44 `cxx::Pair< First, Second >` Struct Template Reference

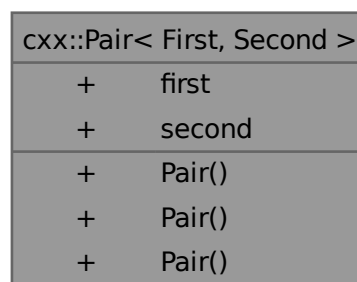
[Pair](#) of two values.

```
#include <pair>
```

Inheritance diagram for `cxx::Pair< First, Second >`:



Collaboration diagram for `cxx::Pair< First, Second >`:



Public Types

- typedef First **First_type**

Type of first value.

- `typedef Second Second_type`

Type of second value.

Public Member Functions

- `template<typename A1, typename A2>
Pair (A1 &&first, A2 &&second)`

Create a pair from the two values.

- `template<typename A1>
Pair (A1 &&first)`

Create a pair, default constructing the second value.

- `Pair ()=default`

Default construction.

Data Fields

- First **first**

First value.

- Second **second**

Second value.

15.44.1 Detailed Description

`template<typename First, typename Second>
struct cxx::Pair< First, Second >`

[Pair](#) of two values.

Standard container for a pair of values.

Parameters

| | |
|---------------|---------------------------|
| <i>First</i> | Type of the first value. |
| <i>Second</i> | Type of the second value. |

Definition at line 27 of file [pair](#).

15.44.2 Constructor & Destructor Documentation

15.44.2.1 Pair() [1/2]

```
template<typename First, typename Second>
template<typename A1, typename A2>
cxx::Pair< First, Second >::Pair (
    A1 && first,
    A2 && second) [inline]
```

Create a pair from the two values.

Parameters

| | |
|---------------|-------------------|
| <i>first</i> | The first value. |
| <i>second</i> | The second value. |

Definition at line 45 of file [pair](#).

15.44.2.2 Pair() [2/2]

```
template<typename First, typename Second>
template<typename A1>
cxx::Pair< First, Second >::Pair (
    A1 && first) [inline]
```

Create a pair, default constructing the second value.

Parameters

| | |
|--------------|------------------|
| <i>first</i> | The first value. |
|--------------|------------------|

Definition at line 53 of file [pair](#).

The documentation for this struct was generated from the following file:

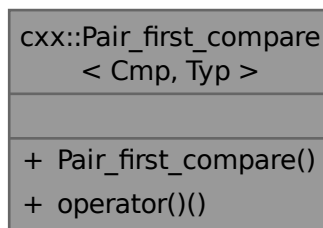
- [I4/cxx/pair](#)

15.45 cxx::Pair_first_compare< Cmp, Typ > Class Template Reference

Comparison functor for [Pair](#).

```
#include <pair>
```

Collaboration diagram for `cxx::Pair_first_compare< Cmp, Typ >`:



Public Member Functions

- [Pair_first_compare](#) (`Cmp const &cmp=Cmp()`)
Construction.
- `bool` [operator\(\)](#) (`Typ const &l, Typ const &r`) `const`
Do the comparison based on the first value.

15.45.1 Detailed Description

`template<typename Cmp, typename Typ>`
`class cxx::Pair_first_compare< Cmp, Typ >`

Comparison functor for [Pair](#).

Parameters

| | |
|------------|---|
| <i>Cmp</i> | Comparison functor for the first value of the pair. |
| <i>Typ</i> | The pair type. |

This functor can be used to compare [Pair](#) values with respect to the first value.

Definition at line 74 of file [pair](#).

15.45.2 Constructor & Destructor Documentation

15.45.2.1 Pair_first_compare()

```

template<typename Cmp, typename Typ>
cxx::Pair_first_compare< Cmp, Typ >::Pair_first_compare (
    Cmp const & cmp = Cmp()) [inline]
  
```

Construction.

Parameters

| | |
|------------|--|
| <i>cmp</i> | The comparison functor used for the first value. |
|------------|--|

Definition at line 84 of file [pair](#).

15.45.3 Member Function Documentation**15.45.3.1 operator>()**

```
template<typename Cmp, typename Typ>
bool cxx::Pair_first_compare< Cmp, Typ >::operator() (
    Typ const & l,
    Typ const & r) const [inline]
```

Do the comparison based on the first value.

Parameters

| | |
|----------|----------------------|
| <i>l</i> | The lefthand value. |
| <i>r</i> | The righthand value. |

Definition at line 91 of file [pair](#).

The documentation for this class was generated from the following file:

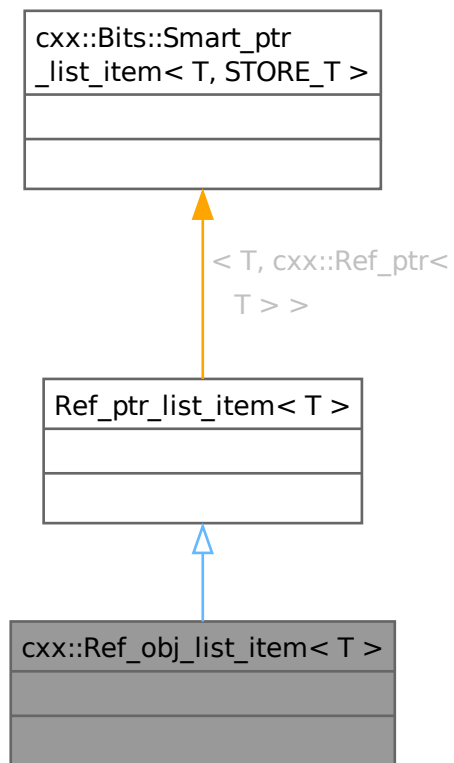
- [l4/cxx/pair](#)

15.46 cxx::Ref_obj_list_item< T > Struct Template Reference

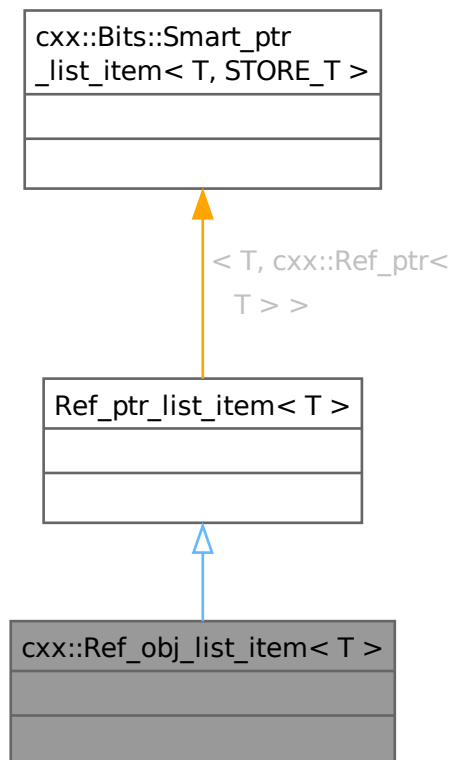
Item for list linked via [cxx::Ref_ptr](#) with default reference counting.

```
#include <ref_ptr_list>
```

Inheritance diagram for cxx::Ref_obj_list_item< T >:



Collaboration diagram for `cxx::Ref_obj_list_item< T >`:



15.46.1 Detailed Description

```
template<typename T>
struct cxx::Ref_obj_list_item< T >
```

Item for list linked via [cxx::Ref_ptr](#) with default reference counting.

Definition at line 26 of file [ref_ptr_list](#).

The documentation for this struct was generated from the following file:

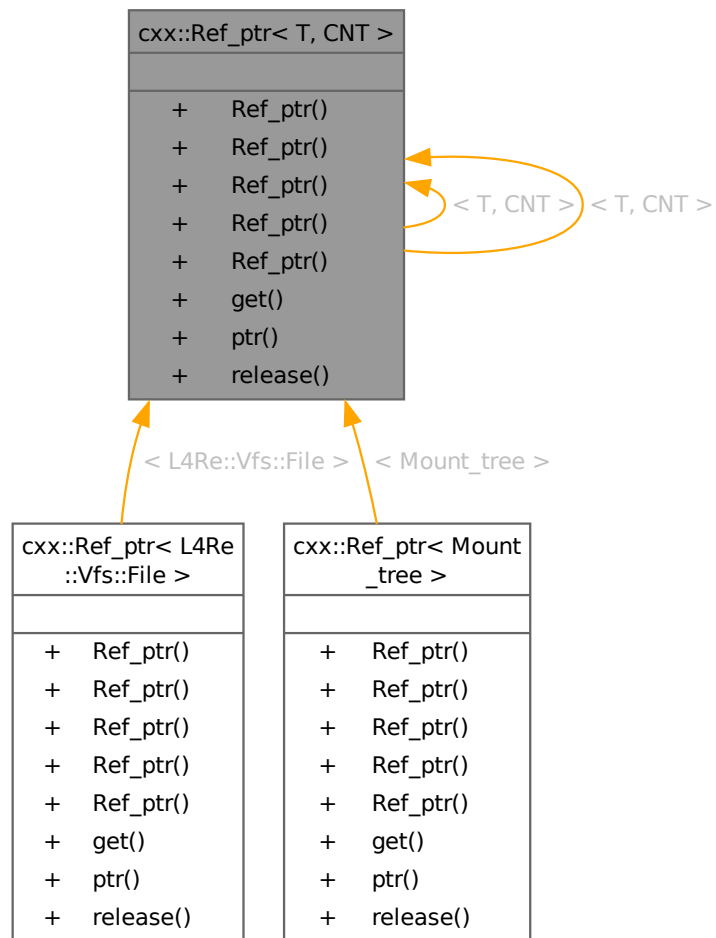
- [l4/cxx/ref_ptr_list](#)

15.47 `cxx::Ref_ptr< T, CNT >` Class Template Reference

A reference-counting pointer with automatic cleanup.

```
#include <ref_ptr>
```

Inheritance diagram for cxx::Ref_ptr< T, CNT >:



Collaboration diagram for `cxx::Ref_ptr< T, CNT >`:

| <code>cxx::Ref_ptr< T, CNT ></code> | |
|---|------------------------|
| | |
| + | <code>Ref_ptr()</code> |
| + | <code>Ref_ptr()</code> |
| + | <code>Ref_ptr()</code> |
| + | <code>Ref_ptr()</code> |
| + | <code>Ref_ptr()</code> |
| + | <code>get()</code> |
| + | <code>ptr()</code> |
| + | <code>release()</code> |

Public Member Functions

- **`Ref_ptr()`** noexcept
Default constructor creates a pointer with no managed object.
- **`Ref_ptr`** (`Wp const &o`) noexcept
Create a shared pointer from a weak pointer.
- **`Ref_ptr`** (`decltype(nullptr) n`) noexcept
allow creation from `nullptr`
- `template<typename X>`
`Ref_ptr` (`X *o`) noexcept
Create a shared pointer from a raw pointer.
- **`Ref_ptr`** (`T *o, bool d`) noexcept
Create a shared pointer from a raw pointer without creating a new reference.
- `T * get ()` const noexcept
Return a raw pointer to the object this shared pointer points to.
- `T * ptr ()` const noexcept
Return a raw pointer to the object this shared pointer points to.
- `T * release ()` noexcept
Release the shared pointer without removing the reference.

15.47.1 Detailed Description

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
class cxx::Ref_ptr< T, CNT >
```

A reference-counting pointer with automatic cleanup.

Template Parameters

| | |
|------------|--|
| <i>T</i> | Type of object the pointer points to. |
| <i>CNT</i> | Type of management class that manages the life time of the object. |

This pointer is similar to the standard C++-11 `shared_ptr` but it does the reference counting directly in the object being pointed to, so that no additional management structures need to be allocated from the heap.

Classes that use this pointer type must implement two functions:

```
int remove_ref()
```

is called when a reference is removed and must return 0 when there are no further references to the object.

```
void add_ref()
```

is called when another `ref_ptr` to the object is created.

`Ref_obj` provides a simple implementation of this interface from which classes may inherit.

Examples

[tmpfs/lib/src/fs.cc](#).

Definition at line 70 of file [ref_ptr](#).

15.47.2 Constructor & Destructor Documentation**15.47.2.1 `Ref_ptr()` [1/3]**

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
cxx::Ref_ptr< T, CNT >::Ref_ptr (
    Wp const & o) [inline], [noexcept]
```

Create a shared pointer from a weak pointer.

Increases references.

Definition at line 88 of file [ref_ptr](#).

15.47.2.2 `Ref_ptr()` [2/3]

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
template<typename X>
cxx::Ref_ptr< T, CNT >::Ref_ptr (
    X * o) [inline], [explicit], [noexcept]
```

Create a shared pointer from a raw pointer.

In contrast to C++11 `shared_ptr` it is safe to use this constructor multiple times and have the same reference counter.

Definition at line 101 of file [ref_ptr](#).

15.47.2.3 Ref_ptr() [3/3]

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
cxx::Ref_ptr< T, CNT >::Ref_ptr (
    T * o,
    bool d) [inline], [noexcept]
```

Create a shared pointer from a raw pointer without creating a new reference.

Parameters

| | |
|----------|---|
| <i>o</i> | Pointer to the object. |
| <i>d</i> | Dummy parameter to select this constructor at compile time. The value may be true or false. |

This is the counterpart to [release\(\)](#).

Definition at line 114 of file [ref_ptr](#).

15.47.3 Member Function Documentation

15.47.3.1 get()

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
T * cxx::Ref_ptr< T, CNT >::get () const [inline], [noexcept]
```

Return a raw pointer to the object this shared pointer points to.

This does not release the pointer or decrease the reference count.

Definition at line 121 of file [ref_ptr](#).

15.47.3.2 ptr()

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
T * cxx::Ref_ptr< T, CNT >::ptr () const [inline], [noexcept]
```

Return a raw pointer to the object this shared pointer points to.

This does not release the pointer or decrease the reference count.

Definition at line 127 of file [ref_ptr](#).

15.47.3.3 release()

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
T * cxx::Ref_ptr< T, CNT >::release () [inline], [noexcept]
```

Release the shared pointer without removing the reference.

Returns

A raw pointer to the managed object.

Definition at line 138 of file [ref_ptr](#).

The documentation for this class was generated from the following file:

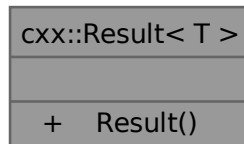
- I4/cxx/ref_ptr

15.48 `cxx::Result< T >` Class Template Reference

A result of a function call.

```
#include <result>
```

Collaboration diagram for `cxx::Result< T >`:



Public Member Functions

- [Result](#) ()=delete
Prevent default construction.

15.48.1 Detailed Description

```
template<typename T>
class cxx::Result< T >
```

A result of a function call.

Either has some resulting value or an error.

The error is always an integer and, if present, a negative value by convention. Any access to the value is fatal if the objects holds an error.

Definition at line [39](#) of file [result](#).

15.48.2 Constructor & Destructor Documentation

15.48.2.1 `Result()`

```
template<typename T>
cxx::Result< T >::Result () [delete]
```

Prevent default construction.

A [Result](#) object either holds a value or an error. There is no third "empty" state.

The documentation for this class was generated from the following file:

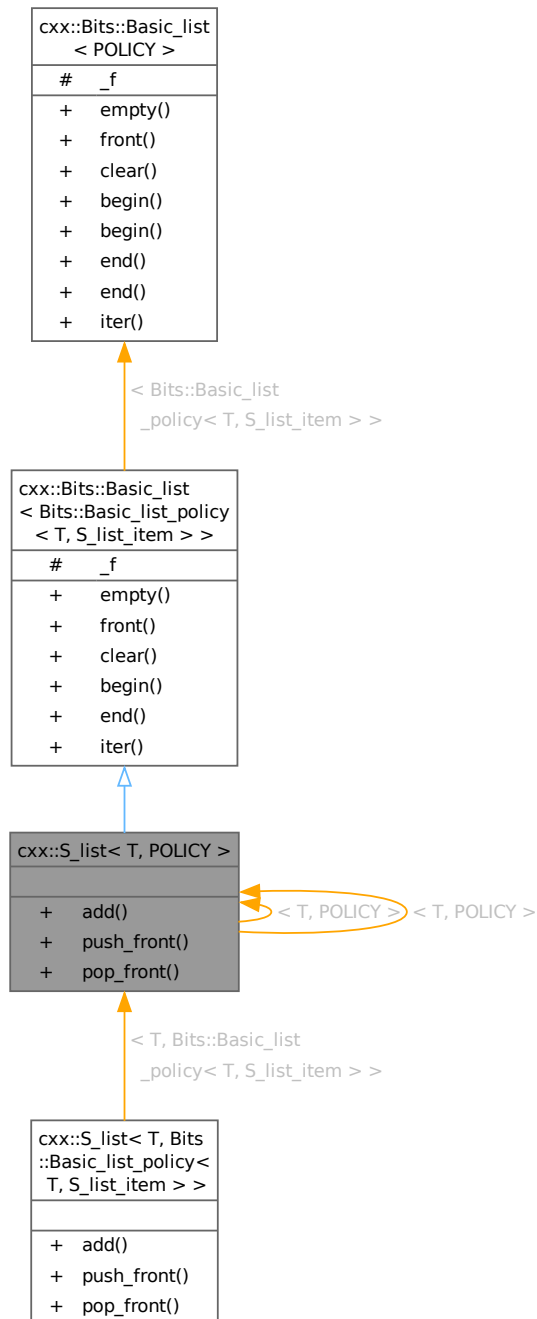
- `I4/cxx/result`

15.49 cxx::S_list< T, POLICY > Class Template Reference

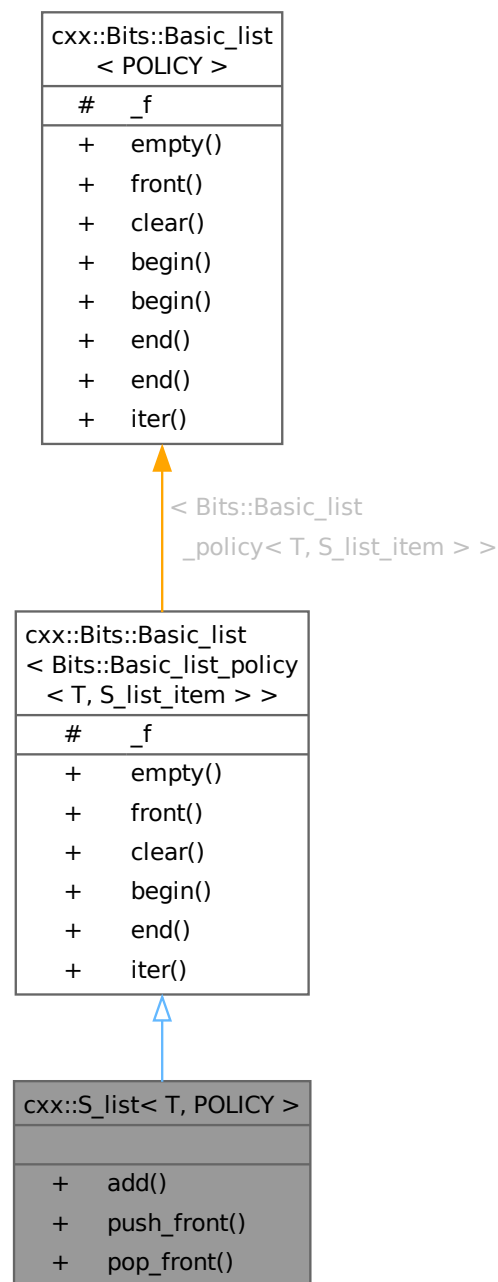
Simple single-linked list.

```
#include <slist>
```

Inheritance diagram for cxx::S_list< T, POLICY >:



Collaboration diagram for cxx::S_list< T, POLICY >:



Public Member Functions

- void **add** (T *e)
Add an element to the front of the list.
- void **push_front** (T *e)
Add an element to the front of the list.
- T * **pop_front** ()
Remove and return the head element of the list.

Public Member Functions inherited from**cxx::Bits::Basic_list< Bits::Basic_list_policy< T, S_list_item > >**

- bool **empty** () const
Check if the list is empty.
- Value_type **front** () const
Return the first element in the list.
- void **clear** ()
Remove all elements from the list.
- Iterator **begin** ()
Return an iterator to the beginning of the list.
- Const_iterator **end** () const
Return a const iterator to the end of the list.

Additional Inherited Members**Static Public Member Functions inherited from****cxx::Bits::Basic_list< Bits::Basic_list_policy< T, S_list_item > >**

- static Const_iterator **iter** (Const_value_type c)
Return a const iterator that begins at the given element.

Protected Attributes inherited from**cxx::Bits::Basic_list< Bits::Basic_list_policy< T, S_list_item > >**

- Bits::Basic_list_policy< T, S_list_item >::Head_type **_f**
Pointer to front of the list.

15.49.1 Detailed Description

```
template<typename T, typename POLICY = Bits::Basic_list_policy< T, S_list_item >>
class cxx::S_list< T, POLICY >
```

Simple single-linked list.

Template Parameters

| | |
|----------|--|
| <i>T</i> | Type of elements saved in the list. Must inherit from cxx::S_list_item |
|----------|--|

Definition at line 40 of file [slist](#).

15.49.2 Member Function Documentation

15.49.2.1 `pop_front()`

```
template<typename T, typename POLICY = Bits::Basic_list_policy< T, S_list_item >>
T * cxx::S_list< T, POLICY >::pop_front () [inline]
```

Remove and return the head element of the list.

Precondition

The list must not be empty or the behaviour will be undefined.

Definition at line 91 of file `slist`.

The documentation for this class was generated from the following file:

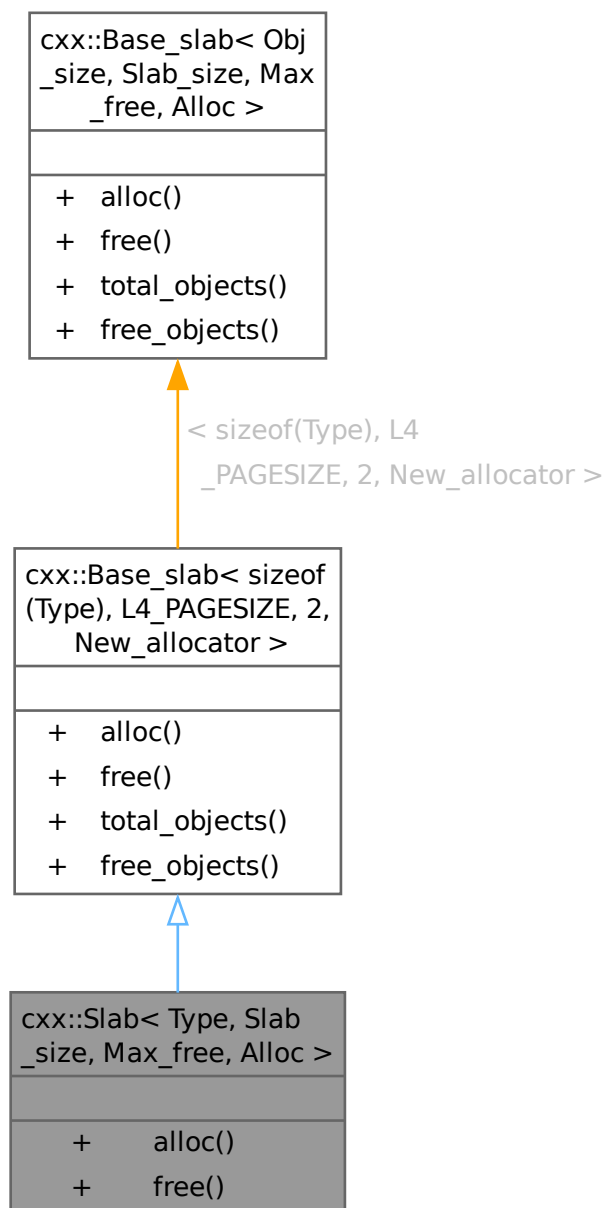
- `l4/cxx/slist`

15.50 `cxx::Slab< Type, Slab_size, Max_free, Alloc >` Class Template Reference

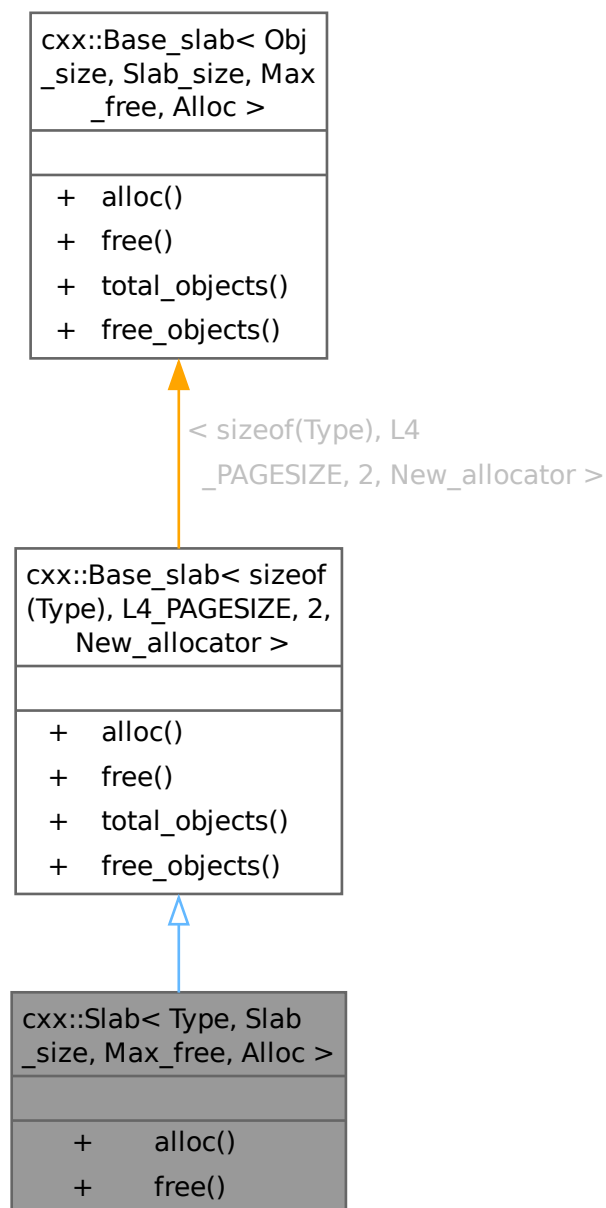
`Slab` allocator for object of type `Type`.

```
#include <slab_alloc>
```

Inheritance diagram for `cxx::Slab< Type, Slab_size, Max_free, Alloc >`:



Collaboration diagram for cxx::Slab< Type, Slab_size, Max_free, Alloc >:



Public Member Functions

- `Type * alloc () noexcept`
Allocate an object of type `Type`.
- `void free (Type *o) noexcept`
Free the object addressed by `o`.

Public Member Functions inherited from

[cxx::Base_slab< sizeof\(Type\), L4_PAGESIZE, 2, New_allocator >](#)

- void * [alloc](#) () noexcept
Allocate a new object.
- void [free](#) (void * _o) noexcept
Free the given object (_o).
- unsigned [total_objects](#) () const noexcept
Get the total number of objects managed by the slab allocator.
- unsigned [free_objects](#) () const noexcept
Get the number of objects which can be allocated before a new empty slab needs to be added to the slab allocator.

Additional Inherited Members

Public Types inherited from

[cxx::Base_slab< sizeof\(Type\), L4_PAGESIZE, 2, New_allocator >](#)

- typedef [New_allocator](#)< Slab_i > [Slab_alloc](#)
Type of the backend allocator.

15.50.1 Detailed Description

```
template<typename Type, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A > class
Alloc = New_allocator>
class cxx::Slab< Type, Slab_size, Max_free, Alloc >
```

[Slab](#) allocator for object of type `Type`.

Template Parameters

| | |
|------------------|------------------------------------|
| <i>Type</i> | The type of the objects to manage. |
| <i>Slab_size</i> | Size of a slab. |
| <i>Max_free</i> | The maximum number of free slabs. |
| <i>Alloc</i> | The allocator for the slabs. |

Definition at line [335](#) of file [slab_alloc](#).

15.50.2 Member Function Documentation

15.50.2.1 alloc()

```
template<typename Type, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
Type * cxx::Slab< Type, Slab\_size, Max\_free, Alloc >::alloc () [inline], [noexcept]
```

Allocate an object of type `Type`.

Returns

A pointer to the object just allocated, or 0 on failure.

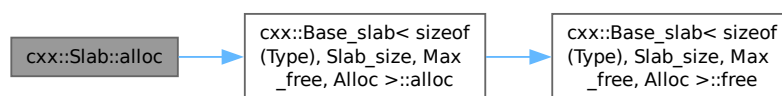
Note

The user is responsible for initializing the object.

Definition at line 355 of file `slab_alloc`.

References `cxx::Base_slab< sizeof(Type), Slab_size, Max_free, Alloc >::alloc()`.

Here is the call graph for this function:

**15.50.2.2 free()**

```

template<typename Type, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
void cxx::Slab< Type, Slab_size, Max_free, Alloc >::free (
    Type * o) [inline], [noexcept]
  
```

Free the object addressed by `o`.

Parameters

| | |
|----------------|------------------------------------|
| <code>o</code> | The pointer to the object to free. |
|----------------|------------------------------------|

Precondition

The object must have been allocated with this allocator.

Definition at line 366 of file `slab_alloc`.

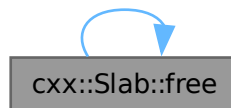
References `free()`.

Referenced by `free()`.

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

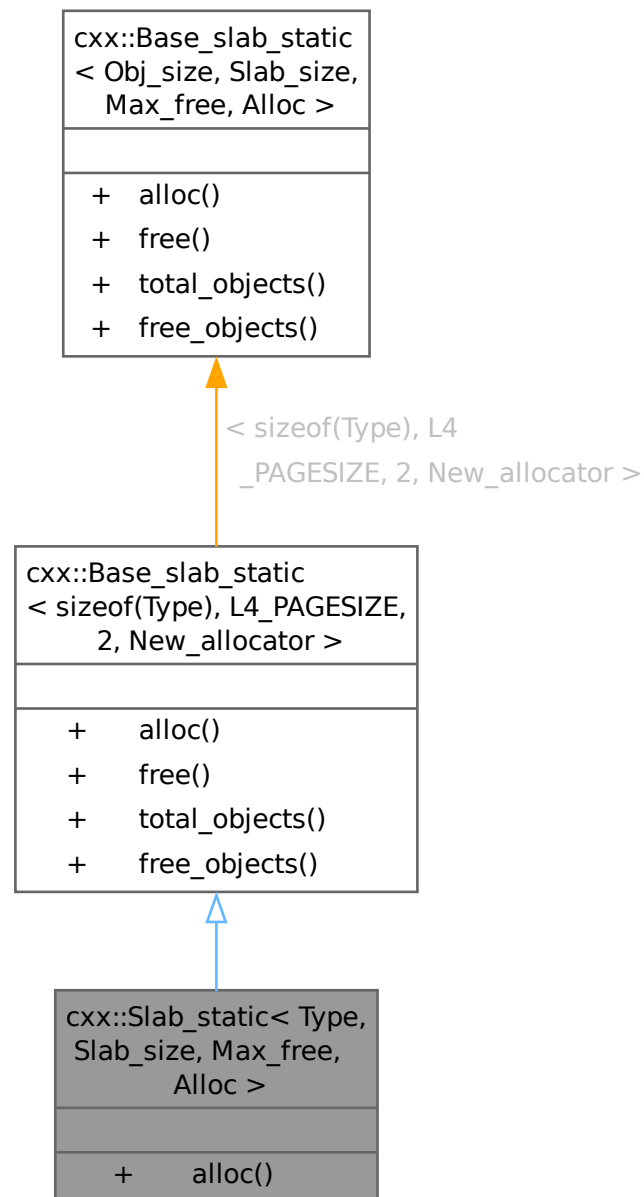
- `I4/cxx/slab_alloc`

15.51 `cxx::Slab_static< Type, Slab_size, Max_free, Alloc > Class` Template Reference

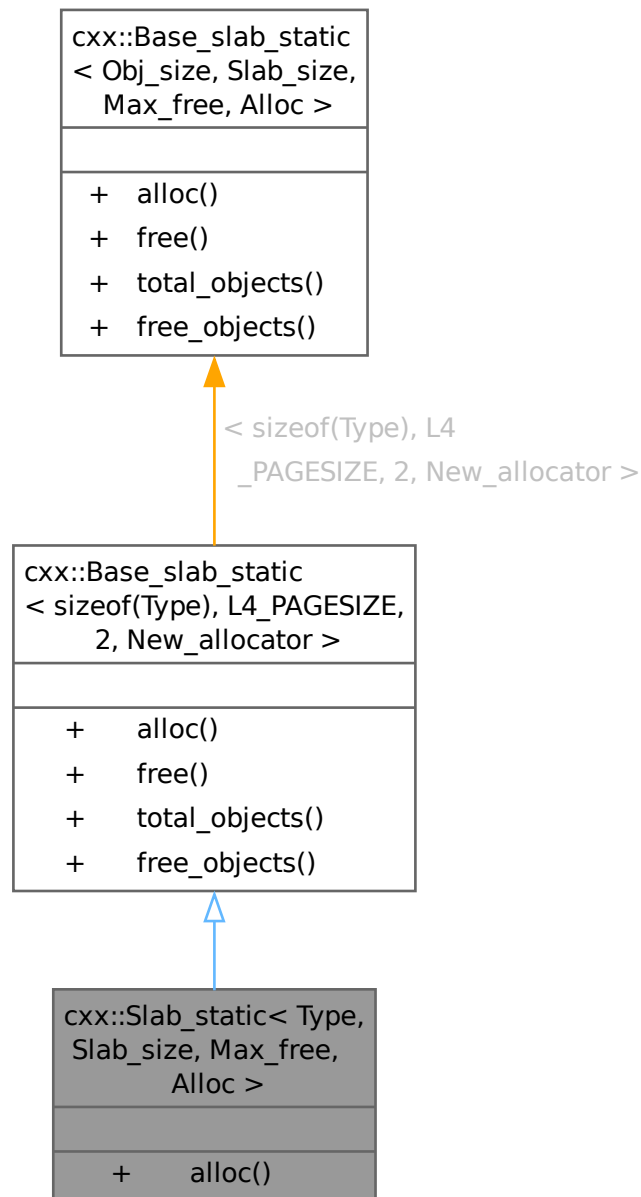
Merged slab allocator (allocators for objects of the same size are merged together).

```
#include <slab_alloc>
```

Inheritance diagram for cxx::Slab_static< Type, Slab_size, Max_free, Alloc >:



Collaboration diagram for `cxx::Slab_static< Type, Slab_size, Max_free, Alloc >`:



Public Member Functions

- `Type * alloc () noexcept`
Allocate an object of type *Type*.

Public Member Functions inherited from

`cxx::Base_slab_static< sizeof(Type), L4_PAGESIZE, 2, New_allocator >`

- `void * alloc () noexcept`

Allocate an object.

- void `free` (void *p) noexcept

Free the given object (p).

- unsigned `total_objects` () const noexcept

Get the total number of objects managed by the slab allocator.

- unsigned `free_objects` () const noexcept

Get the number of free objects in the slab allocator.

Additional Inherited Members

Public Types inherited from

`cxx::Base_slab_static< sizeof(Type), L4_PAGESIZE, 2, New_allocator >`

15.51.1 Detailed Description

```
template<typename Type, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A > class
Alloc = New_allocator>
```

```
class cxx::Slab_static< Type, Slab_size, Max_free, Alloc >
```

Merged slab allocator (allocators for objects of the same size are merged together).

Template Parameters

| | |
|------------------|------------------------------------|
| <i>Type</i> | The type of the objects to manage. |
| <i>Slab_size</i> | The size of a slab. |
| <i>Max_free</i> | The maximum number of free slabs. |
| <i>Alloc</i> | The allocator for the slabs. |

This slab allocator class is useful for merging slab allocators with the same parameters (equal `sizeof(Type)`, `Slab_size`, `Max_free`, and `Alloc` parameters) together and share the overhead for the slab caches among all equal-sized objects.

Definition at line 465 of file `slab_alloc`.

15.51.2 Member Function Documentation

15.51.2.1 `alloc()`

```
template<typename Type, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator>
Type * cxx::Slab_static< Type, Slab_size, Max_free, Alloc >::alloc () [inline], [noexcept]
```

Allocate an object of type `Type`.

Returns

A pointer to the just allocated object, or 0 on failure.

Note

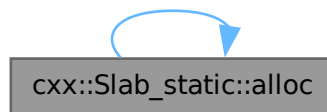
The object is not zeroed out by the slab allocator.

Definition at line 478 of file [slab_alloc](#).

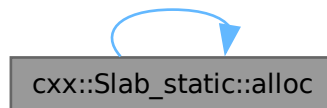
References [alloc\(\)](#).

Referenced by [alloc\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

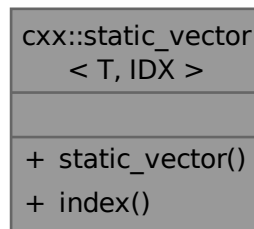
- `l4/cxx/slab_alloc`

15.52 `cxx::static_vector< T, IDX >` Class Template Reference

Simple encapsulation for a dynamically allocated array.

```
#include <static_vector>
```


Collaboration diagram for `cxx::static_vector< T, IDX >`:



Public Member Functions

- `template<typename X, typename = enable_if_t<is_convertible_v<X, T>>>`
static_vector (`static_vector< X, IDX > const &o`)
Conversion from compatible arrays.
- `index_type` **index** (`value_type const *o`) `const`
Get the index of the given element of the array.

15.52.1 Detailed Description

```
template<typename T, typename IDX = unsigned>
class cxx::static_vector< T, IDX >
```

Simple encapsulation for a dynamically allocated array.

The main purpose of this class is to support C++11 range for for simple dynamically allocated array with static size.

Definition at line 16 of file [static_vector](#).

The documentation for this class was generated from the following file:

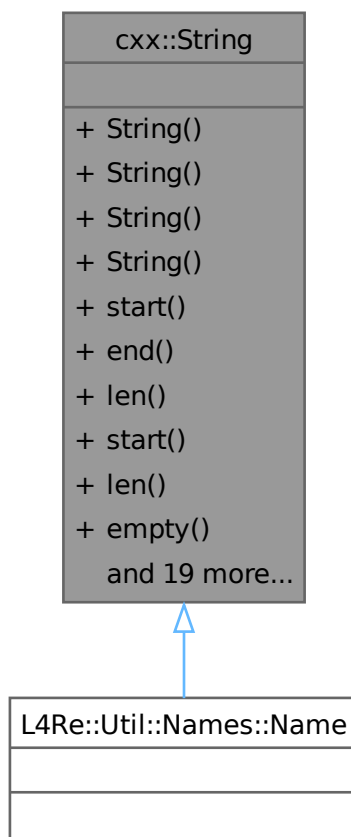
- `l4/cxx/static_vector`

15.53 cxx::String Class Reference

Allocation free string class with explicit length field.

```
#include <string>
```

Inheritance diagram for `cxx::String`:



Collaboration diagram for cxx::String:

| cxx::String |
|--|
| <ul style="list-style-type: none"> + String() + String() + String() + String() + start() + end() + len() + start() + len() + empty() and 19 more... |

Public Types

- typedef char const * **Index**
Character index type.

Public Member Functions

- **String** (char const *s) noexcept
Initialize from a zero-terminated string.
- **String** (char const *s, unsigned long len) noexcept
Initialize from a pointer to first character and a length.
- **String** (char const *s, char const *e) noexcept
Initialize with start and end pointer.
- **String** ()
Zero-initialize. Create an invalid string.
- **Index start** () const
Pointer to first character.
- **Index end** () const
Pointer to first byte behind the string.
- int **len** () const
Length.
- void **start** (char const *s)
Set start.
- void **len** (unsigned long len)

- Set length.*
- **bool empty ()** const
 - Check if the string has length zero.*
- **String head (Index end)** const
 - Return prefix up to index.*
- **String head** (unsigned long end) const
 - Prefix of length end.*
- **String substr** (unsigned long idx, unsigned long len=`~0UL`) const
 - Substring of length len starting at idx.*
- **String substr** (char const *start, unsigned long len=0) const
 - Substring of length len starting at start.*
- **template<typename F>**
char const * find_match (F &&match) const
 - Find matching character. match should be a function such as isspace.*
- **char const * find** (char const *c) const
 - Find character. Return end() if not found.*
- **char const * find** (int c) const
 - Find character. Return end() if not found.*
- **char const * rfind** (char const *c) const
 - Find right-most character. Return end() if not found.*
- **Index starts_with** (cxx::String const &c) const
 - Check if c is a prefix of string.*
- **char const * find** (int c, char const *s) const
 - Find character c starting at position s. Return end() if not found.*
- **char const * find** (char const *c, char const *s) const
 - Find character set at position.*
- **char const & operator[]** (unsigned long idx) const
 - Get character at idx.*
- **char const & operator[]** (int idx) const
 - Get character at idx.*
- **char const & operator[]** (Index idx) const
 - Get character at idx.*
- **bool eof** (char const *s) const
 - Check if pointer s points behind string.*
- **template<typename INT>**
int from_dec (INT *v) const
 - Convert decimal string to integer.*
- **template<typename INT>**
int from_hex (INT *v) const
 - Convert hex string to integer.*
- **bool operator==** (String const &o) const
 - Equality.*
- **bool operator!=** (String const &o) const
 - Inequality.*

15.53.1 Detailed Description

Allocation free string class with explicit length field.

This class is used to group characters of a string which belong to one syntactical token types number, identifier, string, whitespace or another single character.

Stings in this class can contain null bytes and may denote parts of other strings.

Examples

[tmpfs/lib/src/fs.cc](#).

Definition at line 30 of file [string](#).

15.53.2 Constructor & Destructor Documentation

15.53.2.1 String()

```
cxx::String::String (  
    char const * s,  
    char const * e) [inline], [noexcept]
```

Initialize with start and end pointer.

Parameters

| | |
|----------|---|
| <i>s</i> | first character of the string |
| <i>e</i> | pointer to first byte behind the string |

Definition at line 48 of file [string](#).

15.53.3 Member Function Documentation

15.53.3.1 find()

```
char const * cxx::String::find (  
    char const * c,  
    char const * s) const [inline]
```

Find character set at position.

Parameters

| | |
|----------|--|
| <i>c</i> | zero-terminated string of characters to search for |
| <i>s</i> | start position of search in string |

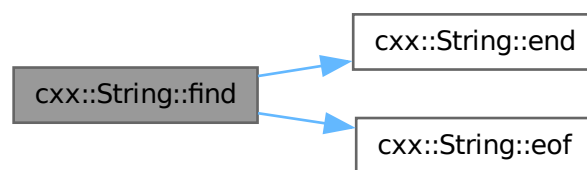
Return values

| | |
|-----------------------|---|
| end() | if no char in <code>c</code> is contained in string at or behind <code>s</code> . |
| <i>position</i> | in string of some character in <code>c</code> . |

Definition at line 191 of file [string](#).

References [end\(\)](#), and [eof\(\)](#).

Here is the call graph for this function:

**15.53.3.2 from_dec()**

```

template<typename INT>
int cxx::String::from_dec (
    INT * v) const [inline]
  
```

Convert decimal string to integer.

Template Parameters

| | |
|------------|---------------------|
| <i>INT</i> | result integer type |
|------------|---------------------|

Parameters

| | | |
|------------|----------|-------------------|
| <i>out</i> | <i>v</i> | conversion result |
|------------|----------|-------------------|

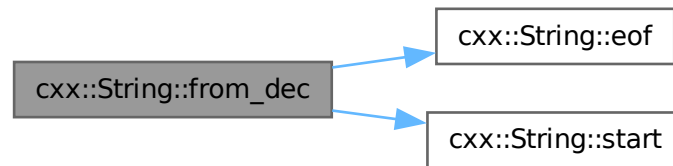
Returns

position of first character not converted.

Definition at line 228 of file [string](#).

References [eof\(\)](#), and [start\(\)](#).

Here is the call graph for this function:



15.53.3.3 from_hex()

```
template<typename INT>  
int cxx::String::from_hex (  
    INT * v) const [inline]
```

Convert hex string to integer.

Template Parameters

| | |
|------------|---------------------|
| <i>INT</i> | result integer type |
|------------|---------------------|

Parameters

| | | |
|-----|----------|-------------------|
| out | <i>v</i> | conversion result |
|-----|----------|-------------------|

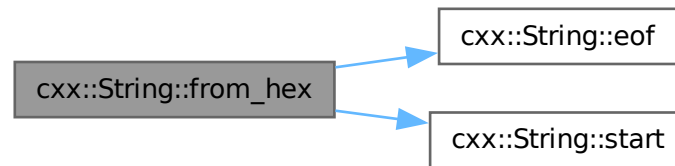
Return values

| | |
|-----------------|---|
| <i>-1</i> | if the maximal amount of digits fitting into <code>INT</code> have been read, |
| <i>position</i> | of first character not converted otherwise. |

Definition at line 257 of file [string](#).

References [eof\(\)](#), and [start\(\)](#).

Here is the call graph for this function:



15.53.3.4 starts_with()

```
Index cxx::String::starts_with (  
    cxx::String const & c) const [inline]
```

Check if `c` is a prefix of string.

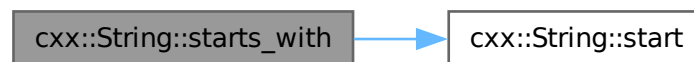
Returns

0 if `c` is not a prefix, if it is a prefix, return first position not in `c` (which might be `end()`).

Definition at line 155 of file `string`.

References `start()`.

Here is the call graph for this function:



The documentation for this class was generated from the following file:

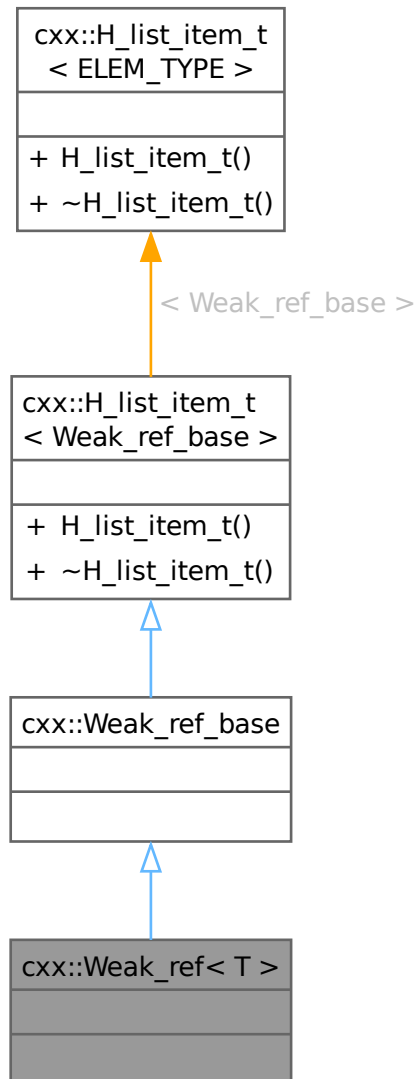
- `I4/cxx/string`

15.54 cxx::Weak_ref< T > Class Template Reference

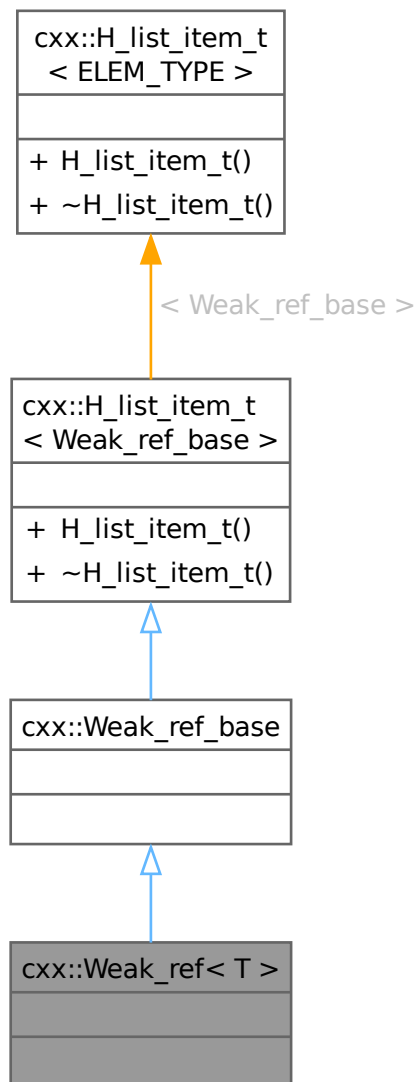
Typed weak reference to an object of type T.

```
#include <weak_ref>
```

Inheritance diagram for cxx::Weak_ref< T >:



Collaboration diagram for `cxx::Weak_ref< T >`:



Additional Inherited Members

Public Member Functions inherited from `cxx::H_list_item_t< Weak_ref_base >`

- `H_list_item_t()`
Constructor.
- `~H_list_item_t()` noexcept
Destructor.

15.54.1 Detailed Description

```
template<typename T>
class cxx::Weak_ref< T >
```

Typed weak reference to an object of type `T`.

Template Parameters

| | |
|----------------|------------------------------------|
| <code>T</code> | The type of the referenced object. |
|----------------|------------------------------------|

A weak reference is a reference that is invalidated when the referenced object is about to be deleted. All weak references to an object are kept in a linked list (see [Weak_ref_base::List](#)) and all the weak references are iterated and reset by the [Weak_ref_base::List](#) destructor or `Weak_ref_base::List::reset()`.

The type `T` must provide two methods that handle the housekeeping of weak references: `remove_weak_ref(Weak_ref_base *)` and `add_weak_ref(Weak_ref_base *)`. These functions must handle the insertion and removal of the weak reference into the respective [Weak_ref_base::List](#) object. For convenience one can use the `cxx::Weak_ref_obj` as a base class that handles weak references for you.

For example:

```
class C : public cxx::Weak_ref_obj {};

int main()
{
    cxx::Weak_ref<C> r; // r is nullptr
    {
        C c;
        r = &c; // now r points to c
    } // c is destructed, which implies resetting all weak references to c
    // now r is nullptr
    return 0;
}
```

Note

Weak references have no effect on the lifetime of the referenced object. Hence, a referenced object is *not* deleted when all weak references for it are gone. If automatic deletion is needed, see [cxx::Ref_ptr](#).

Definition at line 95 of file [weak_ref](#).

The documentation for this class was generated from the following file:

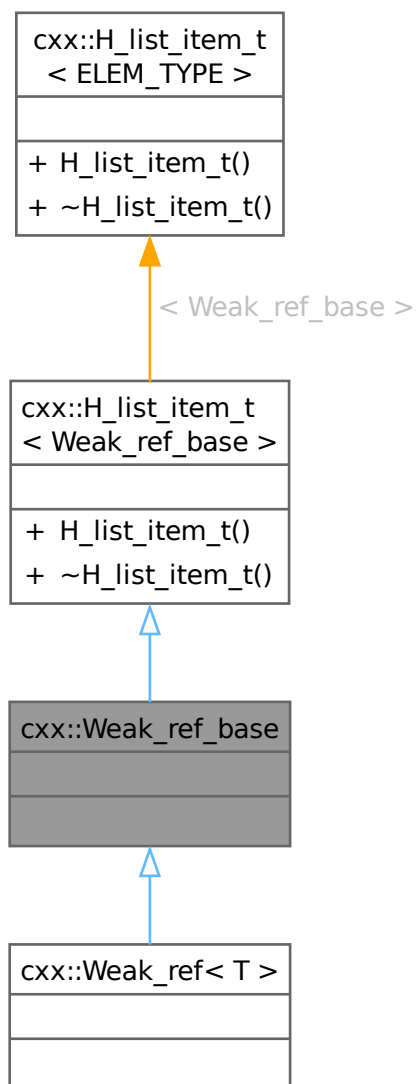
- `I4/cxx/weak_ref`

15.55 cxx::Weak_ref_base Class Reference

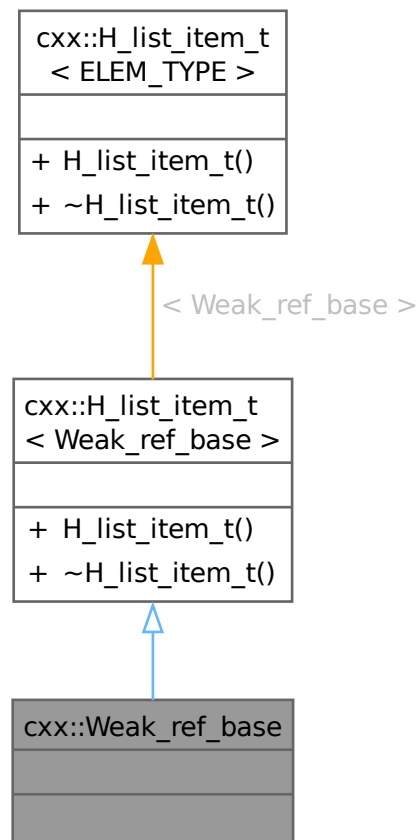
Generic (base) weak reference to some object.

```
#include <weak_ref>
```

Inheritance diagram for `cxx::Weak_ref_base`:



Collaboration diagram for cxx::Weak_ref_base:



Data Structures

- struct [List](#)

The list type for keeping all weak references to an object.

Additional Inherited Members

Public Member Functions inherited from [cxx::H_list_item_t< Weak_ref_base >](#)

- [H_list_item_t](#) ()
Constructor.
- [~H_list_item_t](#) () noexcept
Destructor.

15.55.1 Detailed Description

Generic (base) weak reference to some object.

A weak reference is a reference that gets reset to NULL when the object shall be deleted. All weak references to the same object are kept in a linked list of weak references.

For typed weak references see [cxx::Weak_ref](#).

Definition at line 24 of file [weak_ref](#).

The documentation for this class was generated from the following file:

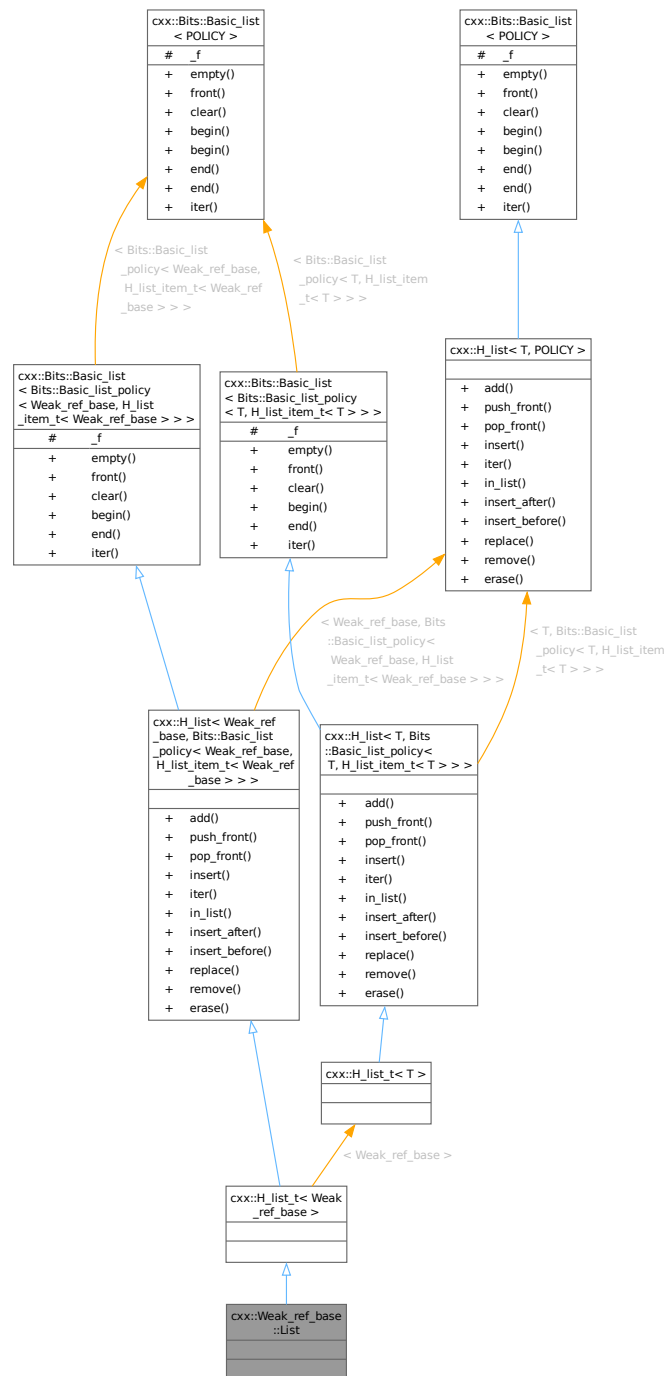
- I4/cxx/weak_ref

15.56 cxx::Weak_ref_base::List Struct Reference

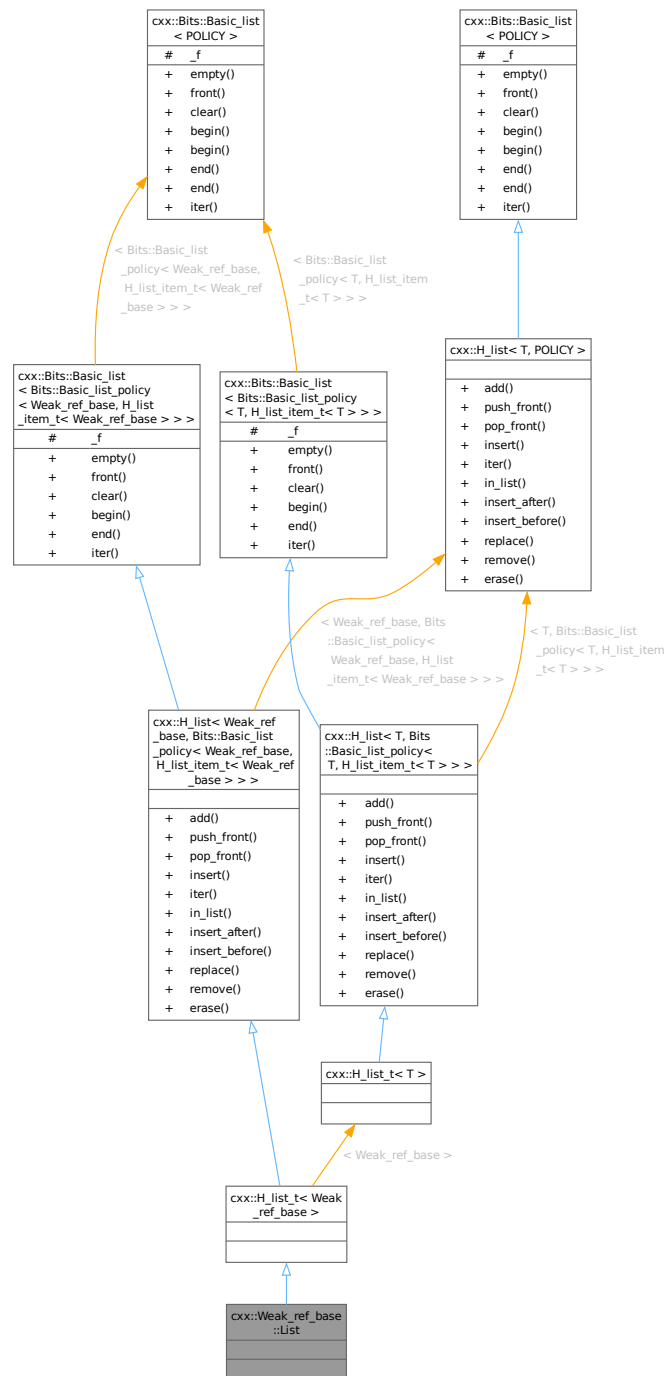
The list type for keeping all weak references to an object.

```
#include <weak_ref>
```

Inheritance diagram for cxx::Weak_ref_base::List:



Collaboration diagram for `cxx::Weak_ref_base::List`:



Additional Inherited Members

Public Member Functions inherited from

`cxx::H_list< Weak_ref_base, Bits::Basic_list_policy< Weak_ref_base, H_list_item_t< Weak_ref_base > >`

- `void add (Weak_ref_base *e)`

- *Add element to the front of the list.*
- void **push_front** (`Weak_ref_base *e`)
Add element to the front of the list.
- `Weak_ref_base * pop_front` ()
Remove and return the head element of the list.
- Iterator **insert** (`Weak_ref_base *e`, Iterator const &pred)
Insert an element at the iterator position.

Public Member Functions inherited from

`cxx::Bits::Basic_list< Bits::Basic_list_policy< Weak_ref_base, H_list_item_t< Weak_ref_base > > >`

- bool **empty** () const
Check if the list is empty.
- Value_type **front** () const
Return the first element in the list.
- void **clear** ()
Remove all elements from the list.
- Iterator **begin** ()
Return an iterator to the beginning of the list.
- Const_iterator **end** () const
Return a const iterator to the end of the list.

Static Public Member Functions inherited from

`cxx::H_list< Weak_ref_base, Bits::Basic_list_policy< Weak_ref_base, H_list_item_t< Weak_ref_base > > >`

- static Iterator **iter** (`Weak_ref_base *c`)
Return an iterator for an arbitrary list element.
- static bool **in_list** (`Weak_ref_base const *e`)
Check if the given element is currently part of a list.
- static Iterator **insert_after** (`Weak_ref_base *e`, Iterator const &pred)
Insert an element after the iterator position.
- static void **insert_before** (`Weak_ref_base *e`, Iterator const &succ)
Insert an element before the iterator position.
- static void **replace** (`Weak_ref_base *p`, `Weak_ref_base *e`)
Replace an element in a list with a new element.
- static void **remove** (`Weak_ref_base *e`)
Remove the given element from its list.
- static Iterator **erase** (Iterator const &e)
Remove the element at the given iterator position.

Static Public Member Functions inherited from

`cxx::Bits::Basic_list< Bits::Basic_list_policy< Weak_ref_base, H_list_item_t< Weak_ref_base > > >`

- static Const_iterator **iter** (Const_value_type c)
Return a const iterator that begins at the given element.

Protected Attributes inherited from

[cxx::Bits::Basic_list](#)< [Bits::Basic_list_policy](#)< [Weak_ref_base](#), [H_list_item_t](#)< [Weak_ref_base](#) > > >

- [Bits::Basic_list_policy](#)< [Weak_ref_base](#), [H_list_item_t](#)< [Weak_ref_base](#) > >::Head_type_f
Pointer to front of the list.

15.56.1 Detailed Description

The list type for keeping all weak references to an object.

On destruction of a list, all weak references to the respective object are set to `nullptr`.

Definition at line 38 of file [weak_ref](#).

The documentation for this struct was generated from the following file:

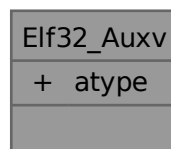
- `l4/cxx/weak_ref`

15.57 Elf32_Auxv Struct Reference

Auxiliary vector (32-bit).

```
#include <elf.h>
```

Collaboration diagram for Elf32_Auxv:

**Data Fields**

- [Elf32_Word atype](#)

15.57.1 Detailed Description

Auxiliary vector (32-bit).

Definition at line 963 of file [elf.h](#).

15.57.2 Field Documentation

15.57.2.1 atype

`Elf32_Word Elf32_Auxv::atype`

See also

[Elf_ATs](#)

Definition at line 965 of file [elf.h](#).

The documentation for this struct was generated from the following file:

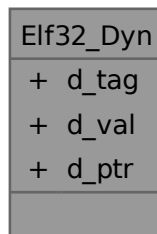
- [l4/util/elf.h](#)

15.58 Elf32_Dyn Struct Reference

ELF32 dynamic entry.

```
#include <elf.h>
```

Collaboration diagram for Elf32_Dyn:



Data Fields

- [Elf32_Sword d_tag](#)

15.58.1 Detailed Description

ELF32 dynamic entry.

Definition at line 513 of file [elf.h](#).

15.58.2 Field Documentation

15.58.2.1 d_tag

[Elf32_Sword](#) Elf32_Dyn::d_tag

See also

[Elf_DTs](#)

Definition at line 515 of file [elf.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/elf.h](#)

15.59 Elf32_Ehdr Struct Reference

ELF32 header.

```
#include <elf.h>
```

Collaboration diagram for Elf32_Ehdr:

| Elf32_Ehdr |
|---------------|
| + e_ident |
| + e_type |
| + e_machine |
| + e_version |
| + e_entry |
| + e_phoff |
| + e_shoff |
| + e_flags |
| + e_ehsize |
| + e_phentsize |
| + e_phnum |
| + e_shentsize |
| + e_shnum |
| + e_shstrndx |
| |

Data Fields

- unsigned char **e_ident** [[EI_NIDENT](#)]
see [Elf_EI](#)s
- [Elf32_Half](#) **e_type**
type of ELF file
- [Elf32_Half](#) **e_machine**
required architecture
- [Elf32_Word](#) **e_version**
file version
- [Elf32_Addr](#) **e_entry**
initial program counter
- [Elf32_Off](#) **e_phoff**
offset of program header table
- [Elf32_Off](#) **e_shoff**
offset of file header table
- [Elf32_Word](#) **e_flags**
processor-specific flags
- [Elf32_Half](#) **e_ehsize**
size of ELF header
- [Elf32_Half](#) **e_phentsize**
size of program header entry
- [Elf32_Half](#) **e_phnum**
number of entries in program header table
- [Elf32_Half](#) **e_shentsize**
size of section header entry
- [Elf32_Half](#) **e_shnum**
number of entries in section header table
- [Elf32_Half](#) **e_shstrndx**
section header table index of strtab

15.59.1 Detailed Description

ELF32 header.

Definition at line 125 of file [elf.h](#).

15.59.2 Field Documentation

15.59.2.1 e_flags

[Elf32_Word](#) [Elf32_Ehdr::e_flags](#)

processor-specific flags

See also

[Elf_EF_ARM_s](#)

Definition at line 134 of file [elf.h](#).

15.59.2.2 e_machine

`Elf32_Half Elf32_Ehdr::e_machine`

required architecture

See also

[Elf_EMs](#)

Definition at line 129 of file [elf.h](#).

15.59.2.3 e_type

`Elf32_Half Elf32_Ehdr::e_type`

type of ELF file

See also

[Elf_ETs](#)

Definition at line 128 of file [elf.h](#).

15.59.2.4 e_version

`Elf32_Word Elf32_Ehdr::e_version`

file version

See also

[Elf_EVs](#)

Definition at line 130 of file [elf.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/elf.h](#)

15.60 Elf32_Phdr Struct Reference

ELF32 program header.

```
#include <elf.h>
```

Collaboration diagram for Elf32_Phdr:

| Elf32_Phdr |
|------------|
| + p_type |
| + p_offset |
| + p_vaddr |
| + p_paddr |
| + p_filesz |
| + p_memsz |
| + p_flags |
| + p_align |
| |

Data Fields

- [Elf32_Word p_type](#)
type of program section
- [Elf32_Off p_offset](#)
file offset of program section
- [Elf32_Addr p_vaddr](#)
memory address of prog section
- [Elf32_Addr p_paddr](#)
physical address (ignored)
- [Elf32_Word p_filesz](#)
file size of program section
- [Elf32_Word p_memsz](#)
memory size of program section
- [Elf32_Word p_flags](#)
flags
- [Elf32_Word p_align](#)
alignment of section

15.60.1 Detailed Description

ELF32 program header.

Definition at line 425 of file [elf.h](#).

15.60.2 Field Documentation

15.60.2.1 p_flags

`Elf32_Word` `Elf32_Phdr::p_flags`

flags

See also

`Elf_PFs`

Definition at line 433 of file `elf.h`.

15.60.2.2 p_type

`Elf32_Word` `Elf32_Phdr::p_type`

type of program section

See also

`Elf_PTs`

Definition at line 427 of file `elf.h`.

The documentation for this struct was generated from the following file:

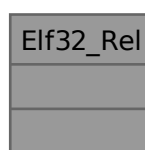
- `l4/util/elf.h`

15.61 Elf32_Rel Struct Reference

ELF32 relocation entry w/o addend.

```
#include <elf.h>
```

Collaboration diagram for `Elf32_Rel`:



15.61.1 Detailed Description

ELF32 relocation entry w/o addend.

Definition at line 631 of file [elf.h](#).

The documentation for this struct was generated from the following file:

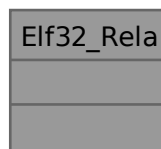
- [l4/util/elf.h](#)

15.62 Elf32_Rela Struct Reference

ELF32 relocation entry w/ addend.

```
#include <elf.h>
```

Collaboration diagram for Elf32_Rela:



15.62.1 Detailed Description

ELF32 relocation entry w/ addend.

Definition at line 638 of file [elf.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/elf.h](#)

15.63 Elf32_Shdr Struct Reference

ELF32 section header.

```
#include <elf.h>
```

Collaboration diagram for Elf32_Shdr:

| Elf32_Shdr |
|----------------|
| + sh_name |
| + sh_type |
| + sh_flags |
| + sh_addr |
| + sh_offset |
| + sh_size |
| + sh_link |
| + sh_info |
| + sh_addralign |
| + sh_entsize |

Data Fields

- [Elf32_Word](#) **sh_name**
name of sect (idx into strtab)
- [Elf32_Word](#) **sh_type**
section's type
- [Elf32_Word](#) **sh_flags**
section's flags
- [Elf32_Addr](#) **sh_addr**
memory address of section
- [Elf32_Off](#) **sh_offset**
file offset of section
- [Elf32_Word](#) **sh_size**
file size of section
- [Elf32_Word](#) **sh_link**
idx to associated header section
- [Elf32_Word](#) **sh_info**
extra info of header section
- [Elf32_Word](#) **sh_addralign**
address alignment constraints
- [Elf32_Word](#) **sh_entsize**
size of entry if sect is table

15.63.1 Detailed Description

ELF32 section header.

Definition at line 347 of file [elf.h](#).

15.63.2 Field Documentation

15.63.2.1 sh_flags

[Elf32_Word](#) Elf32_Shdr::sh_flags

section's flags

See also

[Elf_SHFs](#)

Definition at line 351 of file [elf.h](#).

15.63.2.2 sh_type

[Elf32_Word](#) Elf32_Shdr::sh_type

section's type

See also

[Elf_SHTs](#)

Definition at line 350 of file [elf.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/elf.h](#)

15.64 Elf32_Sym Struct Reference

ELF32 symbol table entry.

```
#include <elf.h>
```

Collaboration diagram for Elf32_Sym:

| Elf32_Sym |
|------------|
| + st_name |
| + st_value |
| + st_size |
| + st_info |
| + st_other |
| + st_shndx |
| |

Data Fields

- [Elf32_Word](#) **st_name**
name of symbol (idx symstrtab)
- [Elf32_Addr](#) **st_value**
value of associated symbol
- [Elf32_Word](#) **st_size**
size of associated symbol
- unsigned char **st_info**
type and binding info
- unsigned char **st_other**
undefined
- [Elf32_Half](#) **st_shndx**
associated section header

15.64.1 Detailed Description

ELF32 symbol table entry.

Definition at line 871 of file [elf.h](#).

The documentation for this struct was generated from the following file:

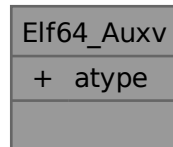
- [l4/util/elf.h](#)

15.65 Elf64_Auxv Struct Reference

Auxiliary vector (64-bit).

```
#include <elf.h>
```

Collaboration diagram for Elf64_Auxv:



Data Fields

- [Elf64_Word atype](#)

15.65.1 Detailed Description

Auxiliary vector (64-bit).

Definition at line [970](#) of file [elf.h](#).

15.65.2 Field Documentation

15.65.2.1 atype

```
Elf64_Word Elf64_Auxv::atype
```

See also

[Elf_ATs](#)

Definition at line [972](#) of file [elf.h](#).

The documentation for this struct was generated from the following file:

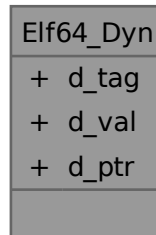
- [l4/util/elf.h](#)

15.66 Elf64_Dyn Struct Reference

ELF64 dynamic entry.

```
#include <elf.h>
```

Collaboration diagram for Elf64_Dyn:



Data Fields

- [Elf64_Sxword d_tag](#)

15.66.1 Detailed Description

ELF64 dynamic entry.

Definition at line [524](#) of file [elf.h](#).

15.66.2 Field Documentation

15.66.2.1 d_tag

```
Elf64_Sxword Elf64_Dyn::d_tag
```

See also

[Elf_DTs](#)

Definition at line [526](#) of file [elf.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/elf.h](#)

15.67 Elf64_Ehdr Struct Reference

ELF64 header.

```
#include <elf.h>
```

Collaboration diagram for Elf64_Ehdr:

| Elf64_Ehdr |
|---------------|
| + e_ident |
| + e_type |
| + e_machine |
| + e_version |
| + e_entry |
| + e_phoff |
| + e_shoff |
| + e_flags |
| + e_ehsize |
| + e_phentsize |
| + e_phnum |
| + e_shentsize |
| + e_shnum |
| + e_shstrndx |
| |

Data Fields

- unsigned char **e_ident** [[EI_NIDENT](#)]
see [Elf_EI](#)s
- [Elf64_Half](#) **e_type**
type of ELF file
- [Elf64_Half](#) **e_machine**
required architecture
- [Elf64_Word](#) **e_version**
file version
- [Elf64_Addr](#) **e_entry**
initial program counter
- [Elf64_Off](#) **e_phoff**
offset of program header table
- [Elf64_Off](#) **e_shoff**

- offset of file header table*
- [Elf64_Word e_flags](#)
processor-specific flags
- [Elf64_Half e_ehsize](#)
size of ELF header
- [Elf64_Half e_phentsize](#)
size of program header entry
- [Elf64_Half e_phnum](#)
number of entries in program header table
- [Elf64_Half e_shentsize](#)
size of section header entry
- [Elf64_Half e_shnum](#)
number of entries in section header table
- [Elf64_Half e_shstrndx](#)
section header table index of strtab

15.67.1 Detailed Description

ELF64 header.

Definition at line 146 of file [elf.h](#).

15.67.2 Field Documentation

15.67.2.1 e_flags

```
Elf64_Word Elf64_Ehdr::e_flags
```

processor-specific flags

See also

[Elf_EF_ARM_s](#)

Definition at line 155 of file [elf.h](#).

15.67.2.2 e_machine

```
Elf64_Half Elf64_Ehdr::e_machine
```

required architecture

See also

[Elf_EMs](#)

Definition at line 150 of file [elf.h](#).

15.67.2.3 e_type

`Elf64_Half Elf64_Ehdr::e_type`

type of ELF file

See also

[Elf_ETs](#)

Definition at line 149 of file [elf.h](#).

15.67.2.4 e_version

`Elf64_Word Elf64_Ehdr::e_version`

file version

See also

[Elf_EVs](#)

Definition at line 151 of file [elf.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/elf.h](#)

15.68 Elf64_Phdr Struct Reference

ELF64 program header.

```
#include <elf.h>
```

Collaboration diagram for Elf64_Phdr:

| Elf64_Phdr |
|------------|
| + p_type |
| + p_flags |
| + p_offset |
| + p_vaddr |
| + p_paddr |
| + p_filesz |
| + p_memsz |
| + p_align |
| |

Data Fields

- [Elf64_Word p_type](#)
type of program section
- [Elf64_Word p_flags](#)
flags
- [Elf64_Off p_offset](#)
file offset of program section
- [Elf64_Addr p_vaddr](#)
memory address of prog section
- [Elf64_Addr p_paddr](#)
physical address (ignored)
- [Elf64_Xword p_filesz](#)
file size of program section
- [Elf64_Xword p_memsz](#)
memory size of program section
- [Elf64_Xword p_align](#)
alignment of section

15.68.1 Detailed Description

ELF64 program header.

Definition at line [438](#) of file [elf.h](#).

15.68.2 Field Documentation

15.68.2.1 p_flags

[Elf64_Word](#) [Elf64_Phdr::p_flags](#)

flags

See also

[Elf_PFs](#)

Definition at line [441](#) of file [elf.h](#).

15.68.2.2 p_type

[Elf64_Word](#) [Elf64_Phdr::p_type](#)

type of program section

See also

[Elf_PT](#)s

Definition at line [440](#) of file [elf.h](#).

The documentation for this struct was generated from the following file:

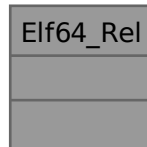
- [l4/util/elf.h](#)

15.69 Elf64_Rel Struct Reference

ELF64 relocation entry w/o addend.

```
#include <elf.h>
```

Collaboration diagram for Elf64_Rel:



15.69.1 Detailed Description

ELF64 relocation entry w/o addend.

Definition at line 646 of file [elf.h](#).

The documentation for this struct was generated from the following file:

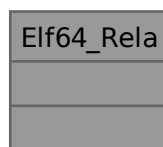
- [l4/util/elf.h](#)

15.70 Elf64_Rela Struct Reference

ELF64 relocation entry w/ addend.

```
#include <elf.h>
```

Collaboration diagram for Elf64_Rela:



15.70.1 Detailed Description

ELF64 relocation entry w/ addend.

Definition at line 653 of file [elf.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/elf.h](#)

15.71 Elf64_Shdr Struct Reference

ELF64 section header.

```
#include <elf.h>
```

Collaboration diagram for Elf64_Shdr:

| Elf64_Shdr |
|----------------|
| + sh_name |
| + sh_type |
| + sh_flags |
| + sh_addr |
| + sh_offset |
| + sh_size |
| + sh_link |
| + sh_info |
| + sh_addralign |
| + sh_entsize |

Data Fields

- [Elf64_Word](#) **sh_name**
name of sect (idx into strtab)
- [Elf64_Word](#) **sh_type**
section's type
- [Elf64_Xword](#) **sh_flags**
section's flags
- [Elf64_Addr](#) **sh_addr**

- memory address of section*
 - [Elf64_Off](#) **sh_offset**
 - file offset of section*
 - [Elf64_Xword](#) **sh_size**
 - file size of section*
 - [Elf64_Word](#) **sh_link**
 - idx to associated header section*
 - [Elf64_Word](#) **sh_info**
 - extra info of header section*
 - [Elf64_Xword](#) **sh_addralign**
 - address alignment constraints*
 - [Elf64_Xword](#) **sh_entsize**
 - size of entry if sect is table*

15.71.1 Detailed Description

ELF64 section header.

Definition at line 362 of file [elf.h](#).

15.71.2 Field Documentation

15.71.2.1 sh_flags

[Elf64_Xword](#) [Elf64_Shdr::sh_flags](#)

section's flags

See also

[Elf_SHFs](#)

Definition at line 366 of file [elf.h](#).

15.71.2.2 sh_type

[Elf64_Word](#) [Elf64_Shdr::sh_type](#)

section's type

See also

[Elf_SHTs](#)

Definition at line 365 of file [elf.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/elf.h](#)

15.72 Elf64_Sym Struct Reference

ELF64 symbol table entry.

```
#include <elf.h>
```

Collaboration diagram for Elf64_Sym:

| Elf64_Sym |
|------------|
| + st_name |
| + st_info |
| + st_other |
| + st_shndx |
| + st_value |
| + st_size |
| |

Data Fields

- [Elf64_Word](#) **st_name**
name of symbol (idx symstrtab)
- unsigned char **st_info**
type and binding info
- unsigned char **st_other**
undefined
- [Elf64_Half](#) **st_shndx**
associated section header
- [Elf64_Addr](#) **st_value**
value of associated symbol
- [Elf64_Xword](#) **st_size**
size of associated symbol

15.72.1 Detailed Description

ELF64 symbol table entry.

Definition at line 882 of file [elf.h](#).

The documentation for this struct was generated from the following file:

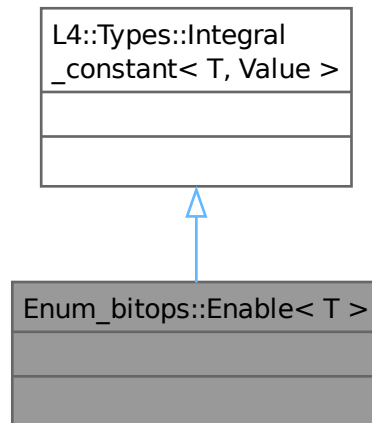
- [l4/util/elf.h](#)

15.73 Enum_bitops::Enable< T > Struct Template Reference

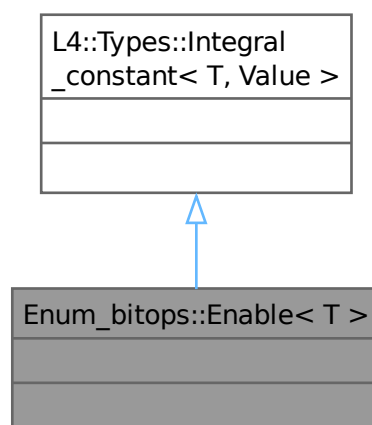
Check whether the given enum type opts in for the bitwise operators.

```
#include <types>
```

Inheritance diagram for Enum_bitops::Enable< T >:



Collaboration diagram for Enum_bitops::Enable< T >:



15.73.1 Detailed Description

```
template<typename T>  
struct Enum_bitops::Enable< T >
```

Check whether the given enum type opts in for the bitwise operators.

Definition at line 482 of file [types](#).

The documentation for this struct was generated from the following file:

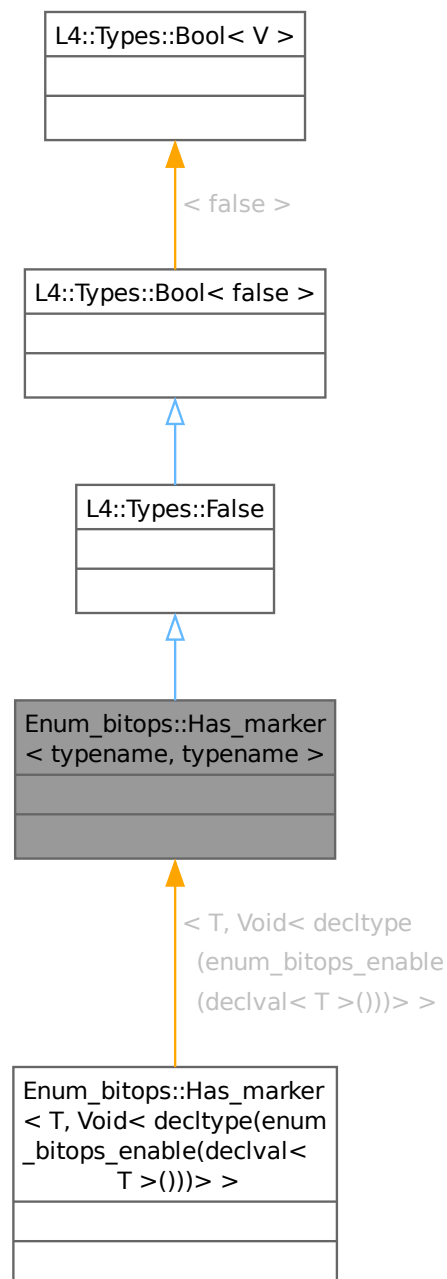
- [l4/sys/cxx/types](#)

15.74 Enum_bitops::Has_marker< typename, typename > Struct Template Reference

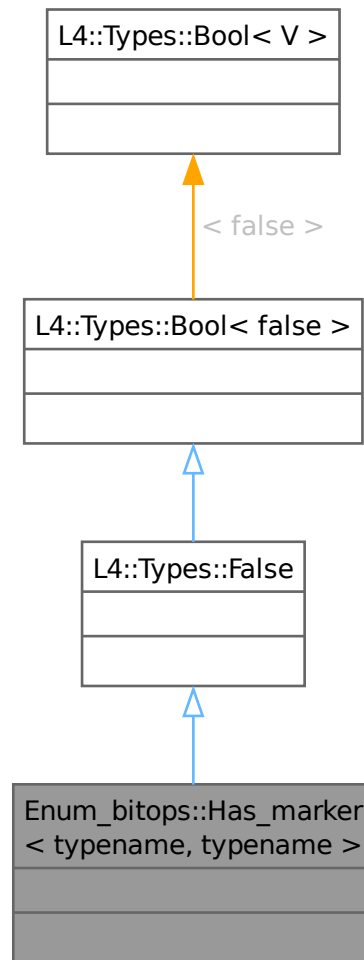
Marker for the opt-in ADL function.

```
#include <types>
```


Inheritance diagram for Enum_bitops::Has_marker< typename, typename >:



Collaboration diagram for Enum_bitops::Has_marker< typename, typename >:



Additional Inherited Members

Public Types inherited from [L4::Types::Bool< false >](#)

- using **type**
The meta type itself.

15.74.1 Detailed Description

```
template<typename, typename = void>
struct Enum_bitops::Has_marker< typename, typename >
```

Marker for the opt-in ADL function.

Definition at line 473 of file [types](#).

The documentation for this struct was generated from the following file:

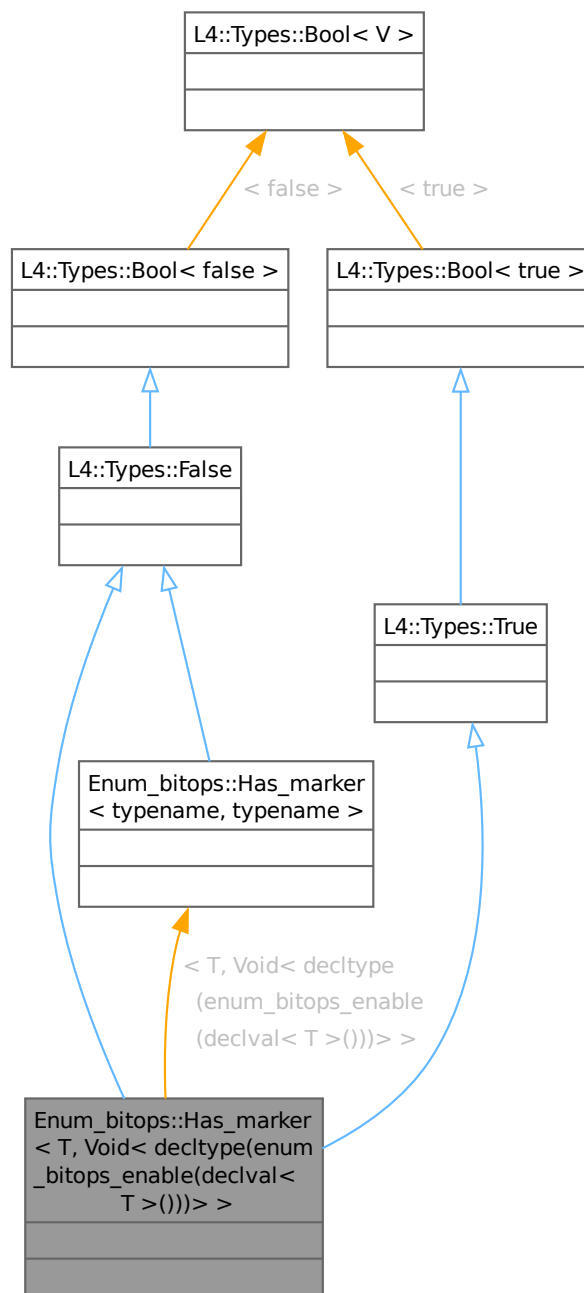
- [l4/sys/cxx/types](#)

15.75 Enum_bitops::Has_marker< T, Void< decltype(enum_bitops_enable(declval< T >()))> > Struct Template Reference

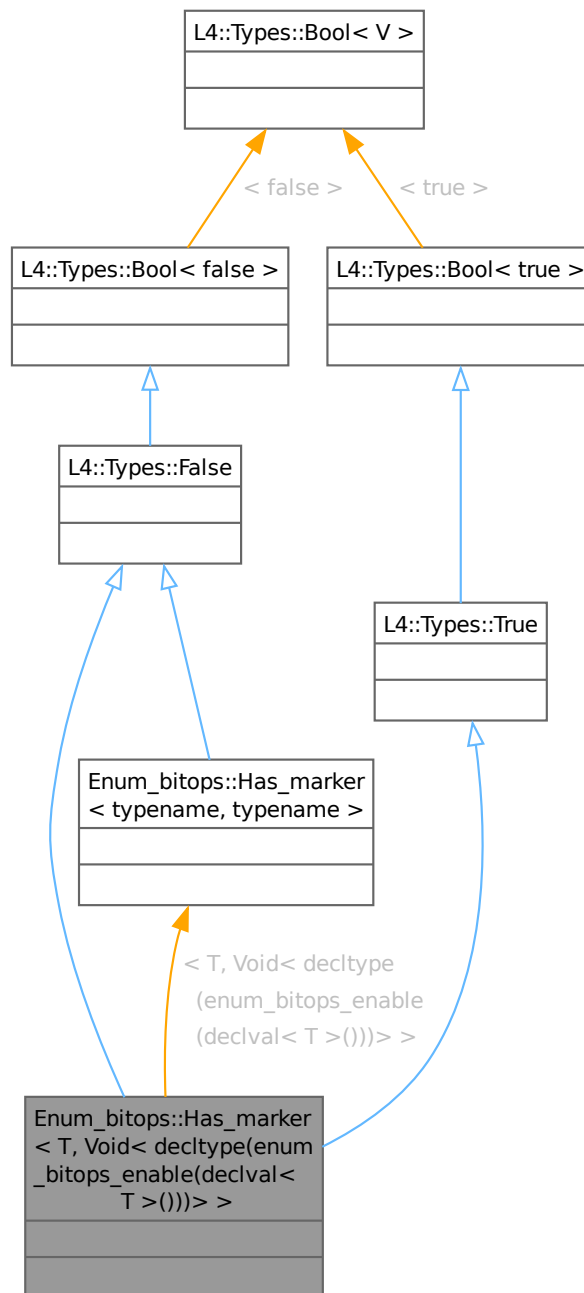
Marker for the opt-in ADL function.

```
#include <types>
```

Inheritance diagram for Enum_bitops::Has_marker< T, Void< decltype(enum_bitops_enable(declval< T >()))> >:



Collaboration diagram for Enum_bitops::Has_marker< T, Void< decltype(enum_bitops_enable(declval< T >()))>> >:



Additional Inherited Members

Public Types inherited from `L4::Types::Bool< false >`

- using **type**
The meta type itself.

Public Types inherited from [L4::Types::Bool< true >](#)

- using **type**

The meta type itself.

15.75.1 Detailed Description

```
template<typename T>
struct Enum_bitops::Has_marker< T, Void< decltype(enum_bitops_enable(declval< T >()))> >
```

Marker for the opt-in ADL function.

Definition at line 477 of file [types](#).

The documentation for this struct was generated from the following file:

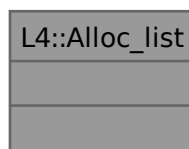
- [l4/sys/cxx/types](#)

15.76 L4::Alloc_list Class Reference

A simple list-based allocator.

```
#include <alloc.h>
```

Collaboration diagram for L4::Alloc_list:



15.76.1 Detailed Description

A simple list-based allocator.

Definition at line 20 of file [alloc.h](#).

The documentation for this class was generated from the following file:

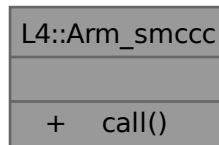
- [l4/cxx/alloc.h](#)

15.77 L4::Arm_smccc Class Reference

Wrapper for function calls that follow the ARM SMC/HVC calling convention.

```
#include <arm_smccc>
```

Collaboration diagram for L4::Arm_smccc:



Public Member Functions

- [l4_msgtag_t](#) [call](#) ([l4_umword_t](#) func, [l4_umword_t](#) in0, [l4_umword_t](#) in1, [l4_umword_t](#) in2, [l4_umword_t](#) in3, [l4_umword_t](#) in4, [l4_umword_t](#) in5, [l4_umword_t](#) *out0, [l4_umword_t](#) *out1, [l4_umword_t](#) *out2, [l4_umword_t](#) *out3, [l4_umword_t](#) client_id)

ARM SMC/HVC function call.

15.77.1 Detailed Description

Wrapper for function calls that follow the ARM SMC/HVC calling convention.

See [l4_arm_smccc_call\(\)](#) for the corresponding C interface.

Definition at line 23 of file [arm_smccc](#).

15.77.2 Member Function Documentation

15.77.2.1 call()

```

l4_msgtag_t L4::Arm_smccc::call (
    l4_umword_t func,
    l4_umword_t in0,
    l4_umword_t in1,
    l4_umword_t in2,
    l4_umword_t in3,
    l4_umword_t in4,
    l4_umword_t in5,
    l4_umword_t * out0,
    l4_umword_t * out1,
    l4_umword_t * out2,

```

```

14_umword_t * out3,
14_umword_t client_id)

```

ARM SMC/HVC function call.

The input parameters consist of a function identifier, 6 arguments and a client id. Results are returned in 4 output parameters.

Parameters

| | | |
|-----|------------------|--|
| | <i>func</i> | Function identifier. <ul style="list-style-type: none"> • Bit 31 has to be set: This marks the call as <i>Fast Call</i>. <i>Yielding Calls</i> (bit 31 unset) are rejected by the kernel. • Bit 30 defines the calling convention: • Bit 30 == 1: 64-bit calling convention. • Bit 30 == 0: 32-bit calling convention. • Bits 24..29 determine the service call ID. The permitted IDs are set in the kernel configuration. By default only service IDs $\geq 0x30000000$ (<i>Trusted Application Calls</i> and <i>Trusted OS Calls</i>) are allowed. |
| in | <i>in0</i> | First input parameter. |
| in | <i>in1</i> | Second input parameter. |
| in | <i>in2</i> | Third input parameter. |
| in | <i>in3</i> | Fourth input parameter. |
| in | <i>in4</i> | Fifth input parameter. |
| in | <i>in5</i> | Sixth input parameter. |
| out | <i>out0</i> | First output parameter. |
| out | <i>out1</i> | Second output parameter. |
| out | <i>out2</i> | Third output parameter. |
| out | <i>out3</i> | Fourth output parameter. |
| in | <i>client_id</i> | Client ID. According to the specification, this value might be ignored by certain functions. |

Return values

| | |
|------------|---|
| -L4_ENOSYS | Either bit 31 of the function call not set or service ID outside the range permitted by kernel configuration. |
| -L4_EINVAL | Invalid number of parameters. |
| < 0 | Other L4 error. |
| 0 | Success. |

The documentation for this class was generated from the following file:

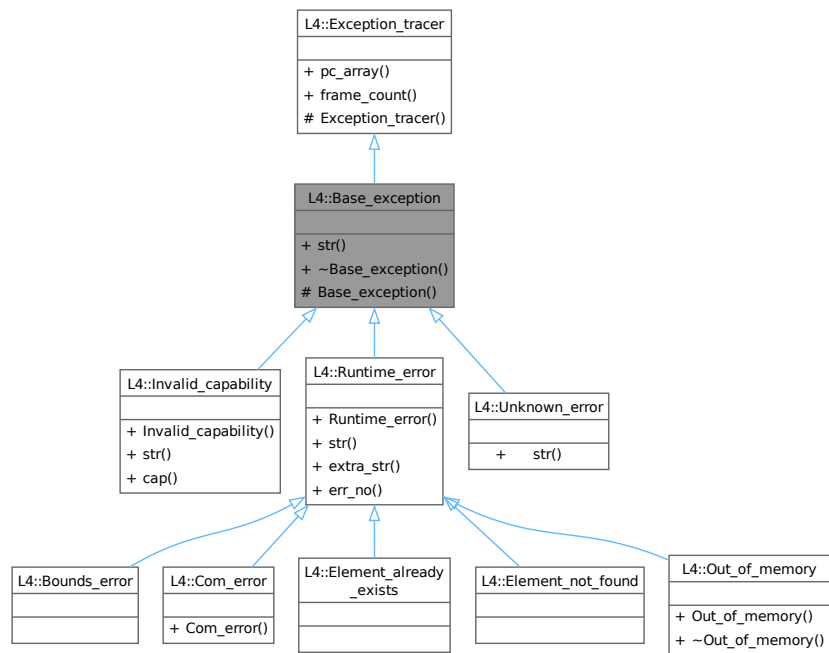
- [l4/sys/arm_smccc](#)

15.78 L4::Base_exception Class Reference

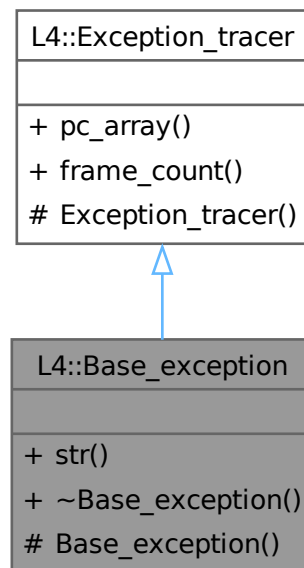
Base class for all exceptions, thrown by the [L4Re](#) framework.

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Base_exception:



Collaboration diagram for L4::Base_exception:



Public Member Functions

- virtual char const * **str** () const noexcept=0
Return a human readable string for the exception.
- virtual ~**Base_exception** () noexcept
Destruction.

Public Member Functions inherited from [L4::Exception_tracer](#)

- void const *const * **pc_array** () const noexcept
Get the array containing the call trace.
- int **frame_count** () const noexcept
Get the number of entries that are valid in the call trace.

Protected Member Functions

- **Base_exception** () noexcept
Create a base exception.

Protected Member Functions inherited from [L4::Exception_tracer](#)

- **Exception_tracer** () noexcept
Create a back trace.

15.78.1 Detailed Description

Base class for all exceptions, thrown by the [L4Re](#) framework.

This is the abstract base of all exceptions thrown within the [L4Re](#) framework. It is basically also a good idea to use it as base of all user defined exceptions.

Definition at line 105 of file [exceptions](#).

The documentation for this class was generated from the following file:

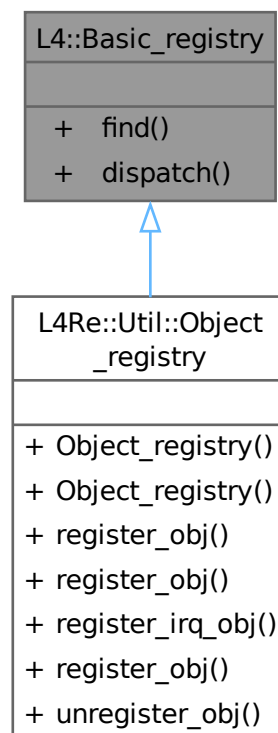
- [l4/cxx/exceptions](#)

15.79 L4::Basic_registry Class Reference

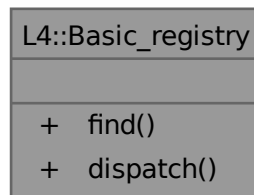
This registry returns the corresponding server object based on the label of an [lpc_gate](#).

```
#include <ipc_epiface>
```

Inheritance diagram for L4::Basic_registry:



Collaboration diagram for L4::Basic_registry:



Static Public Member Functions

- static Value * [find](#) ([l4_umword_t](#) label)
Get the server object for an [lpc_gate](#) label.
- static [l4_msgtag_t](#) [dispatch](#) ([l4_msgtag_t](#) tag, [l4_umword_t](#) label, [l4_utcb_t](#) *utcb)
The dispatch function called by the server loop.

15.79.1 Detailed Description

This registry returns the corresponding server object based on the label of an [lpc_gate](#).

Definition at line 684 of file [ipc_epiface](#).

15.79.2 Member Function Documentation

15.79.2.1 dispatch()

```

l4_msgtag_t L4::Basic_registry::dispatch (
    l4_msgtag_t tag,
    l4_umword_t label,
    l4_utcb_t * utcb) [inline], [static]
  
```

The dispatch function called by the server loop.

This function forwards the message to the server object identified by the given *label*.

Parameters

| | |
|--------------|--|
| <i>tag</i> | The message tag used for the invocation. |
| <i>label</i> | The label used to find the object including the rights bits of the invoked capability. |
| <i>utcb</i> | The UTCB used for the invocation. |

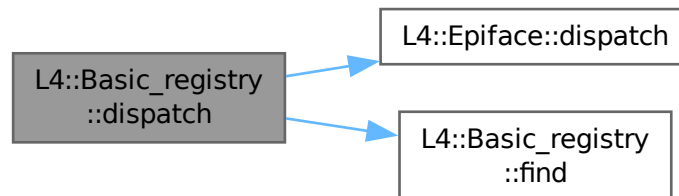
Returns

The return code from the object's dispatch function or -L4_ENOENT if the object does not exist.

Definition at line 709 of file [ipc_epiface](#).

References [L4::Epiface::dispatch\(\)](#), and [find\(\)](#).

Here is the call graph for this function:

**15.79.2.2 find()**

```
Value * L4::Basic_registry::find (
    l4_umword_t label) [inline], [static]
```

Get the server object for an [lpc_gate](#) label.

Parameters

| | |
|--------------|---|
| <i>label</i> | The label usually stored in an lpc_gate . |
|--------------|---|

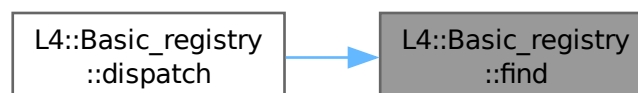
Returns

A pointer to the [Epiface](#) identified by the given label.

Definition at line 693 of file [ipc_epiface](#).

Referenced by [dispatch\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

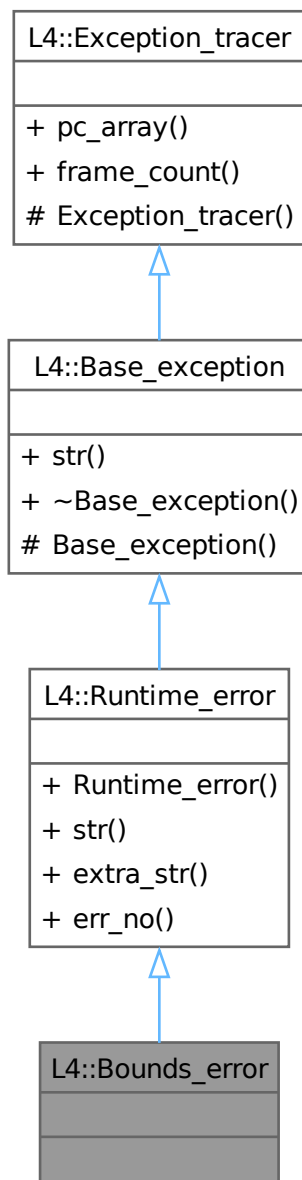
- `l4/sys/cxx/ipc_epiface`

15.80 L4::Bounds_error Class Reference

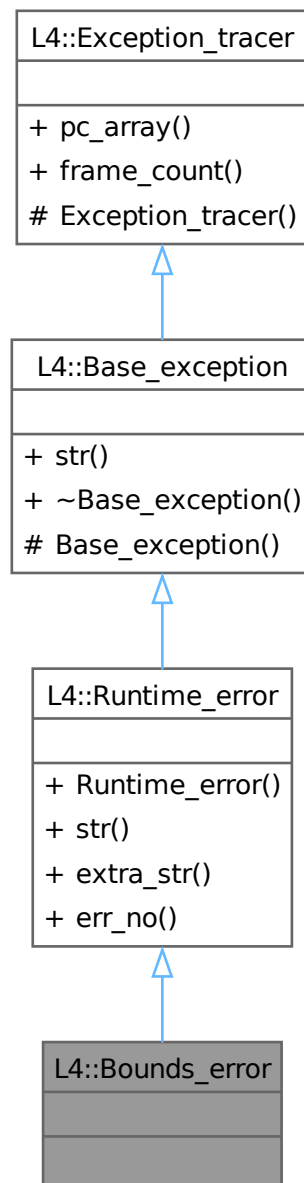
Access out of bounds.

```
#include <exceptions>
```

Inheritance diagram for L4::Bounds_error:



Collaboration diagram for L4::Bounds_error:



Additional Inherited Members

Public Member Functions inherited from [L4::Runtime_error](#)

- [Runtime_error](#) (long [err_no](#), char const *extra=0) noexcept
Create a new [Runtime_error](#).
- char const * **str** () const noexcept override
Return a human readable string for the exception.

- char const * [extra_str](#) () const
Get the description text for this runtime error.
- long [err_no](#) () const noexcept
Get the error value for this runtime error.

Public Member Functions inherited from [L4::Base_exception](#)

- virtual ~**Base_exception** () noexcept
Destruction.

Public Member Functions inherited from [L4::Exception_tracer](#)

- void const *const * **pc_array** () const noexcept
Get the array containing the call trace.
- int **frame_count** () const noexcept
Get the number of entries that are valid in the call trace.

Protected Member Functions inherited from [L4::Base_exception](#)

- **Base_exception** () noexcept
Create a base exception.

Protected Member Functions inherited from [L4::Exception_tracer](#)

- **Exception_tracer** () noexcept
Create a back trace.

15.80.1 Detailed Description

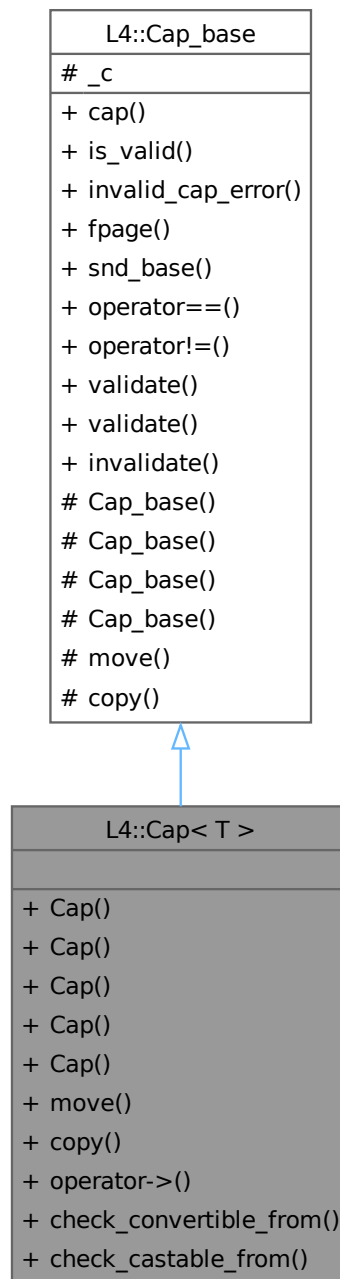
Access out of bounds.

Definition at line 278 of file [exceptions](#).

The documentation for this class was generated from the following file:

- [l4/cxx/exceptions](#)

Collaboration diagram for L4::Cap< T >:



Public Member Functions

- `template<typename O>`
`Cap (Cap< O > const &o) noexcept`
Create a copy from o, supporting implicit type casting.
- `Cap (Cap_type cap) noexcept`
Constructor to create an invalid capability selector.

- [Cap](#) ([l4_default_caps_t cap](#)) noexcept
Initialize capability with one of the default capability selectors.
- [Cap](#) ([l4_cap_idx_t idx=L4_INVALID_CAP](#)) noexcept
Initialize capability, defaults to the invalid capability selector.
- [Cap](#) ([No_init_type](#)) noexcept
Create an uninitialized cap selector.
- [Cap](#) [move](#) ([Cap](#) const &src) const
Move a capability to this cap slot.
- [Cap](#) [copy](#) ([Cap](#) const &src) const
Copy a capability to this cap slot.
- [T](#) * [operator->](#) () const noexcept
Member access of a [T](#).

Public Member Functions inherited from [L4::Cap_base](#)

- [l4_cap_idx_t cap](#) () const noexcept
Return capability selector.
- bool [is_valid](#) () const noexcept
Test whether the capability is a valid capability index (i.e., not [L4_INVALID_CAP](#)).
- int [invalid_cap_error](#) () const noexcept
Return the transported error code in an invalid capability index.
- [l4_fpage_t fpage](#) (unsigned rights=[L4_CAP_FPAGE_RWS](#)) const noexcept
Return flexpage for the capability.
- [l4_umword_t snd_base](#) (unsigned grant=[L4_MAP_ITEM_MAP](#), [l4_cap_idx_t](#) base=[L4_INVALID_CAP](#)) const noexcept
Return send base.
- bool [operator==](#) ([Cap_base](#) const &o) const noexcept
Test if two capabilities are equal.
- bool [operator!=](#) ([Cap_base](#) const &o) const noexcept
Test if two capabilities are not equal.
- [l4_msgtag_t](#) [validate](#) ([l4_utcb_t](#) *u=[l4_utcb\(\)](#)) const noexcept
Check whether a capability is present (refers to an object).
- [l4_msgtag_t](#) [validate](#) ([Cap](#)< [Task](#) > task, [l4_utcb_t](#) *u=[l4_utcb\(\)](#)) const noexcept
Check whether a capability is present (refers to an object).
- void [invalidate](#) () noexcept
Set this capability to invalid ([L4_INVALID_CAP](#)).

Static Public Member Functions

- template<typename From>
static void [check_convertible_from](#) () noexcept
Perform the type conversion that needs to compile in order for a capability of type From to be convertible to one of type T.
- template<typename From>
static void [check_castable_from](#) () noexcept
Perform the type conversion that needs to compile in order for a capability of type From to be castable (via the correct cap_cast) to one of type T.

Friends

- class [L4::Kobject](#)

Additional Inherited Members

Public Types inherited from L4::Cap_base

- enum [No_init_type](#) { [No_init](#) }
Special value for uninitialized capability objects.
- enum [Cap_type](#) { [Invalid](#) = L4_INVALID_CAP }
Invalid capability type.

Protected Member Functions inherited from L4::Cap_base

- [Cap_base](#) ([l4_cap_idx_t](#) c) noexcept
Generate a capability from its C representation.
- [Cap_base](#) ([Cap_type](#) cap) noexcept
Constructor to create an invalid capability.
- [Cap_base](#) ([l4_default_caps_t](#) cap) noexcept
Initialize capability with one of the default capabilities.
- [Cap_base](#) () noexcept
Create an uninitialized instance.
- void [move](#) ([Cap_base](#) const &src) const
Replace this capability with the contents of `src`.
- void [copy](#) ([Cap_base](#) const &src) const
Copy a capability.

Protected Attributes inherited from L4::Cap_base

- [l4_cap_idx_t_c](#)
The C representation of a capability selector.

15.81.1 Detailed Description

```
template<typename T>
class L4::Cap< T >
```

C++ interface for capabilities.

Template Parameters

| | |
|----------|--|
| <i>T</i> | Type of the object the capability points to. |
|----------|--|

The C++ version of a capability is comparable to a pointer, in fact it is a kind of smart pointer for our kernel objects and the objects derived from the kernel objects ([L4::Kobject](#)).

Add

```
#include <l4/sys/capability>
```

to your code to use the capability interface.

Examples

[examples/clntsrv/src/client.cc](#), [examples/libs/l4re/c++/mem_alloc/ma+rm.cc](#), [examples/libs/l4re/c++/shared_ds/ds_clnt.cc](#), [examples/libs/l4re/c++/shared_ds/ds_srv.cc](#), [examples/libs/l4re/streammap/client.cc](#), and [examples/sys/migrate/thread_migrate](#)

Definition at line 223 of file [capability.h](#).

15.81.2 Constructor & Destructor Documentation

15.81.2.1 Cap() [1/4]

```
template<typename T>
template<typename O>
L4::Cap< T >::Cap (
    Cap< O > const & o) [inline], [noexcept]
```

Create a copy from *o*, supporting implicit type casting.

Parameters

| | |
|----------|--|
| <i>o</i> | The source selector that shall be copied (and casted). |
|----------|--|

Definition at line 275 of file [capability.h](#).

15.81.2.2 Cap() [2/4]

```
template<typename T>
L4::Cap< T >::Cap (
    Cap_type cap) [inline], [noexcept]
```

Constructor to create an invalid capability selector.

Parameters

| | |
|------------|----------------------|
| <i>cap</i> | Capability selector. |
|------------|----------------------|

Definition at line 282 of file [capability.h](#).

15.81.2.3 Cap() [3/4]

```
template<typename T>
L4::Cap< T >::Cap (
    l4_default_caps_t cap) [inline], [noexcept]
```

Initialize capability with one of the default capability selectors.

Parameters

| | |
|------------|----------------------|
| <i>cap</i> | Capability selector. |
|------------|----------------------|

Definition at line 288 of file [capability.h](#).

15.81.2.4 Cap() [4/4]

```
template<typename T>
L4::Cap< T >::Cap (
    l4_cap_idx_t idx = L4_INVALID_CAP) [inline], [explicit], [noexcept]
```

Initialize capability, defaults to the invalid capability selector.

Parameters

| | |
|------------|----------------------|
| <i>idx</i> | Capability selector. |
|------------|----------------------|

Definition at line 294 of file [capability.h](#).

15.81.3 Member Function Documentation**15.81.3.1 check_castable_from()**

```
template<typename T>
template<typename From>
void L4::Cap< T >::check_castable_from () [inline], [static], [noexcept]
```

Perform the type conversion that needs to compile in order for a capability of type From to be castable (via the correct `cap_cast`) to one of type T.

Template Parameters

| | |
|-------------|----------------------|
| <i>From</i> | Type to convert from |
|-------------|----------------------|

Definition at line 264 of file [capability.h](#).

15.81.3.2 check_convertible_from()

```
template<typename T>
template<typename From>
void L4::Cap< T >::check_convertible_from () [inline], [static], [noexcept]
```

Perform the type conversion that needs to compile in order for a capability of type From to be convertible to one of type T.

Template Parameters

| | |
|-------------|----------------------|
| <i>From</i> | Type to convert from |
|-------------|----------------------|

Definition at line 251 of file [capability.h](#).

15.81.3.3 copy()

```
template<typename T>
Cap L4::Cap< T >::copy (
    Cap< T > const & src) const [inline]
```

Copy a capability to this cap slot.

Parameters

| | |
|------------|-----------------------------|
| <i>src</i> | the source capability slot. |
|------------|-----------------------------|

Definition at line 317 of file [capability.h](#).

15.81.3.4 move()

```
template<typename T>
Cap L4::Cap< T >::move (
    Cap< T > const & src) const [inline]
```

Move a capability to this cap slot.

Parameters

| | |
|------------|-----------------------------|
| <i>src</i> | the source capability slot. |
|------------|-----------------------------|

After this operation the source slot is no longer valid.

Definition at line 307 of file [capability.h](#).

The documentation for this class was generated from the following file:

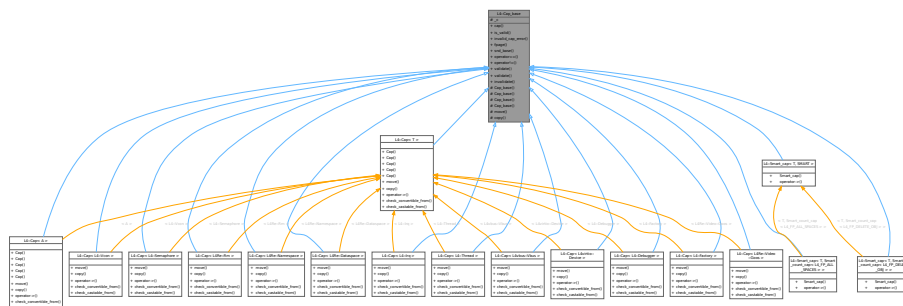
- l4/sys/cxx/capability.h

15.82 L4::Cap_base Class Reference

Base class for all kinds of capabilities.

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Cap_base:



Collaboration diagram for L4::Cap_base:

| L4::Cap_base |
|-----------------------|
| # _c |
| + cap() |
| + is_valid() |
| + invalid_cap_error() |
| + fpage() |
| + snd_base() |
| + operator==() |
| + operator!=() |
| + validate() |
| + validate() |
| + invalidate() |
| # Cap_base() |
| # Cap_base() |
| # Cap_base() |
| # Cap_base() |
| # move() |
| # copy() |

Public Types

- enum [No_init_type](#) { [No_init](#) }
Special value for uninitialized capability objects.
- enum [Cap_type](#) { [Invalid](#) = L4_INVALID_CAP }
Invalid capability type.

Public Member Functions

- [l4_cap_idx_t](#) [cap](#) () const noexcept
Return capability selector.
- bool [is_valid](#) () const noexcept
Test whether the capability is a valid capability index (i.e., not L4_INVALID_CAP).
- int [invalid_cap_error](#) () const noexcept
Return the transported error code in an invalid capability index.
- [l4_fpage_t](#) [fpage](#) (unsigned rights=[L4_CAP_FPAGE_RWS](#)) const noexcept
Return flexpage for the capability.

- [l4_umword_t snd_base](#) (unsigned grant=[L4_MAP_ITEM_MAP](#), [l4_cap_idx_t](#) base=[L4_INVALID_CAP](#)) const noexcept
Return send base.
- bool **operator==** ([Cap_base](#) const &o) const noexcept
Test if two capabilities are equal.
- bool **operator!=** ([Cap_base](#) const &o) const noexcept
Test if two capabilities are not equal.
- [l4_msgtag_t validate](#) ([l4_utcb_t](#) *u=[l4_utcb](#)()) const noexcept
Check whether a capability is present (refers to an object).
- [l4_msgtag_t validate](#) ([Cap](#)< [Task](#) > task, [l4_utcb_t](#) *u=[l4_utcb](#)()) const noexcept
Check whether a capability is present (refers to an object).
- void **invalidate** () noexcept
Set this capability to invalid ([L4_INVALID_CAP](#)).

Protected Member Functions

- [Cap_base](#) ([l4_cap_idx_t](#) c) noexcept
Generate a capability from its C representation.
- **Cap_base** ([Cap_type](#) cap) noexcept
Constructor to create an invalid capability.
- [Cap_base](#) ([l4_default_caps_t](#) cap) noexcept
Initialize capability with one of the default capabilities.
- **Cap_base** () noexcept
Create an uninitialized instance.
- void **move** ([Cap_base](#) const &src) const
*Replace this capability with the contents of *src*.*
- void **copy** ([Cap_base](#) const &src) const
Copy a capability.

Protected Attributes

- [l4_cap_idx_t _c](#)
The C representation of a capability selector.

15.82.1 Detailed Description

Base class for all kinds of capabilities.

Attention

This class is not for direct use, use [L4::Cap](#) instead.

This class contains all the things that are independent of the type of the object referred by the capability.

See also

[L4::Cap](#) for typed capabilities.

Definition at line 25 of file [capability.h](#).

15.82.2 Member Enumeration Documentation

15.82.2.1 Cap_type

enum [L4::Cap_base::Cap_type](#)

Invalid capability type.

Enumerator

| | |
|---------|------------------------------|
| Invalid | Invalid capability selector. |
|---------|------------------------------|

Definition at line [40](#) of file [capability.h](#).

15.82.2.2 No_init_type

enum [L4::Cap_base::No_init_type](#)

Special value for uninitialized capability objects.

Enumerator

| | |
|---------|---|
| No_init | Special value for constructing uninitialized Cap objects. |
|---------|---|

Definition at line [29](#) of file [capability.h](#).

15.82.3 Constructor & Destructor Documentation

15.82.3.1 Cap_base() [1/2]

```
L4::Cap_base::Cap_base (
    l4\_cap\_idx\_t c) [inline], [explicit], [protected], [noexcept]
```

Generate a capability from its C representation.

Parameters

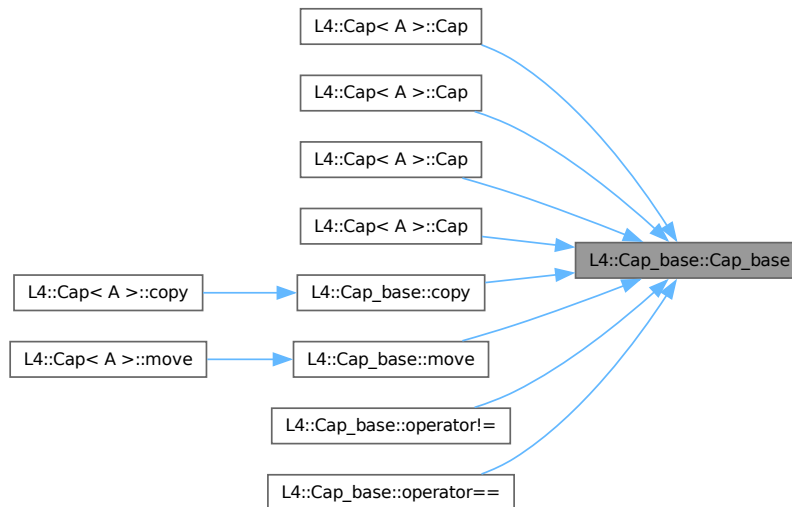
| | |
|----------|------------------|
| <i>c</i> | The C capability |
|----------|------------------|

Definition at line [149](#) of file [capability.h](#).

References [_c](#).

Referenced by [L4::Cap< A >::Cap\(\)](#), [L4::Cap< A >::Cap\(\)](#), [L4::Cap< A >::Cap\(\)](#), [L4::Cap< A >::Cap\(\)](#), [copy\(\)](#), [move\(\)](#), [operator!=\(\)](#), and [operator==\(\)](#).

Here is the caller graph for this function:



15.82.3.2 Cap_base() [2/2]

```
L4::Cap_base::Cap_base (
    l4\_default\_caps\_t cap) [inline], [explicit], [protected], [noexcept]
```

Initialize capability with one of the default capabilities.

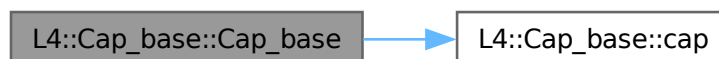
Parameters

| | |
|------------|-------------|
| <i>cap</i> | Capability. |
|------------|-------------|

Definition at line 160 of file [capability.h](#).

References [_c](#), and [cap\(\)](#).

Here is the call graph for this function:



15.82.4 Member Function Documentation

15.82.4.1 cap()

```
l4_cap_idx_t L4::Cap_base::cap () const [inline], [noexcept]
```

Return capability selector.

Returns

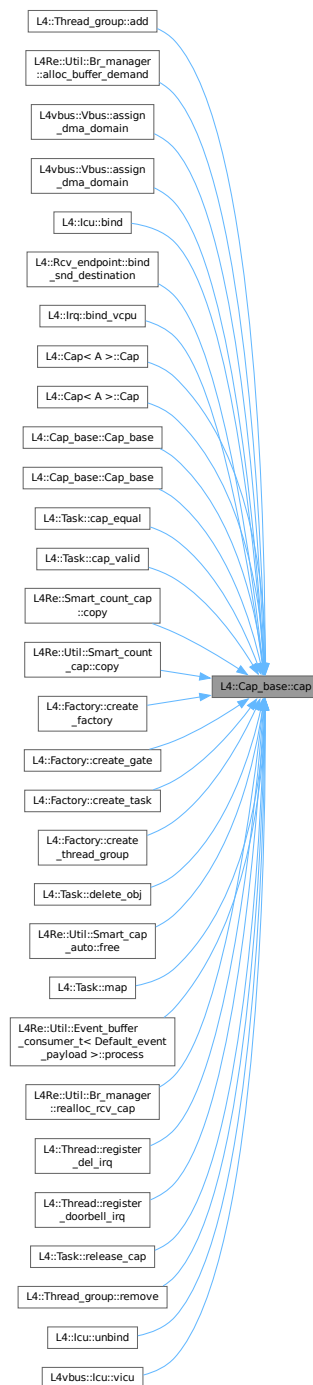
Capability selector.

Definition at line 49 of file [capability.h](#).

References [_c](#).

Referenced by [L4::Thread_group::add\(\)](#), [L4Re::Util::Br_manager::alloc_buffer_demand\(\)](#), [L4vbus::Vbus::assign_dma_domain\(\)](#), [L4vbus::Vbus::assign_dma_domain\(\)](#), [L4::lcu::bind\(\)](#), [L4::Rcv_endpoint::bind_snd_destination\(\)](#), [L4::Irq::bind_vcpu\(\)](#), [L4::Cap< A >::Cap\(\)](#), [L4::Cap< A >::Cap\(\)](#), [Cap_base\(\)](#), [Cap_base\(\)](#), [L4::Task::cap_equal\(\)](#), [L4::Task::cap_valid\(\)](#), [L4Re::Smart_count_cap< Unmap_flags >::copy\(\)](#), [L4Re::Util::Smart_count_cap< Unmap_flags >::copy\(\)](#), [L4::Factory::create_factory\(\)](#), [L4::Factory::create_gate\(\)](#), [L4::Factory::create_task\(\)](#), [L4::Factory::create_thread_group\(\)](#), [L4::Task::delete_obj\(\)](#), [L4Re::Util::Smart_cap_auto< Unmap_flags >::free\(\)](#), [L4::Task::map\(\)](#), [L4Re::Util::Event_buffer_consumer_t< T >::map\(\)](#), [L4Re::Util::Br_manager::realloc_rcv_cap\(\)](#), [L4::Thread::register_del_irq\(\)](#), [L4::Thread::register_doorbell_irq\(\)](#), [L4::Task::release_cap\(\)](#), [L4::Thread_group::remove\(\)](#), [L4::lcu::unbind\(\)](#), and [L4vbus::lcu::vicu\(\)](#).

Here is the caller graph for this function:



15.82.4.2 copy()

```
void L4::Cap_base::copy (
    Cap_base const & src) const [inline], [protected]
```

Copy a capability.

Parameters

| | |
|------------|------------------------|
| <i>src</i> | the source capability. |
|------------|------------------------|

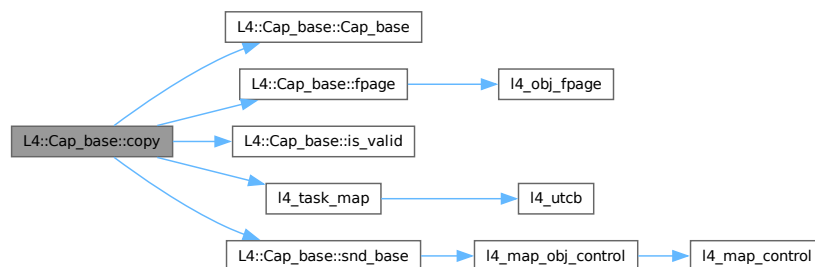
After this operation this capability refers to the same object as `src`.

Definition at line 192 of file `capability.h`.

References `Cap_base()`, `fpage()`, `is_valid()`, `L4_BASE_TASK_CAP`, `L4_CAP_FPAGE_RWSD`, `L4_FPAGE_C_OBJ_RIGHTS`, `l4_task_map()`, and `snd_base()`.

Referenced by `L4::Cap< A >::copy()`.

Here is the call graph for this function:



Here is the caller graph for this function:

**15.82.4.3 fpage()**

```
l4_fpage_t L4::Cap_base::fpage (
    unsigned rights = L4_CAP_FPAGE_RWS) const [inline], [noexcept]
```

Return flexpage for the capability.

Parameters

| | |
|---------------|---------------------------|
| <i>rights</i> | Rights, defaults to 'rws' |
|---------------|---------------------------|

Returns

flexpage

Definition at line 74 of file [capability.h](#).

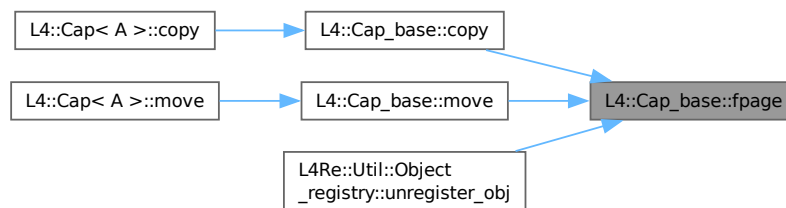
References [_c](#), [L4_CAP_FPAGE_RWS](#), and [l4_obj_fpage\(\)](#).

Referenced by [copy\(\)](#), [move\(\)](#), and [L4Re::Util::Object_registry::unregister_obj\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**15.82.4.4 is_valid()**

```
bool L4::Cap_base::is_valid () const [inline], [noexcept]
```

Test whether the capability is a valid capability index (i.e., not `L4_INVALID_CAP`).

Returns

True if capability is not invalid, false if invalid

Examples

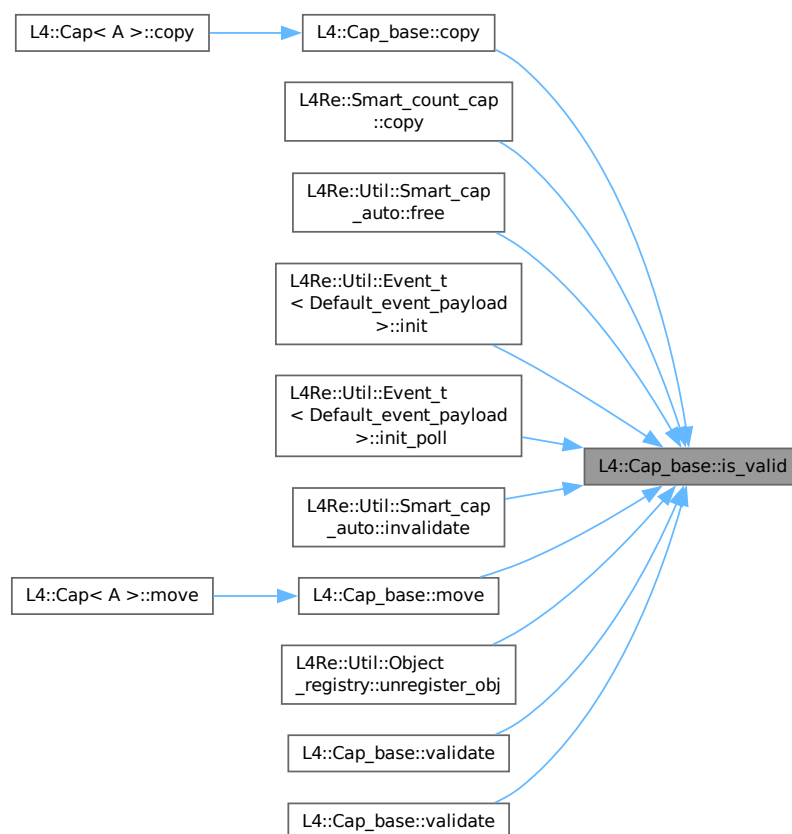
[examples/libs/l4re/c++/mem_alloc/ma+rm.cc](#), and [examples/libs/l4re/c++/shared_ds/ds_clnt.cc](#).

Definition at line 57 of file [capability.h](#).

References [_c](#).

Referenced by [copy\(\)](#), [L4Re::Smart_count_cap< Unmap_flags >::copy\(\)](#), [L4Re::Util::Smart_cap_auto< Unmap_flags >::free\(\)](#), [L4Re::Util::Event_t< Default_event_payload >::init\(\)](#), [L4Re::Util::Event_t< Default_event_payload >::init_poll\(\)](#), [L4Re::Util::Smart_cap_auto< Unmap_flags >::invalidate\(\)](#), [move\(\)](#), [L4Re::Util::Object_registry::unregister_obj\(\)](#), [validate\(\)](#), and [validate\(\)](#).

Here is the caller graph for this function:



15.82.4.5 move()

```
void L4::Cap_base::move (
    Cap\_base const & src) const [inline], [protected]
```

Replace this capability with the contents of `src`.

Parameters

| | |
|------------------|------------------------|
| <code>src</code> | the source capability. |
|------------------|------------------------|

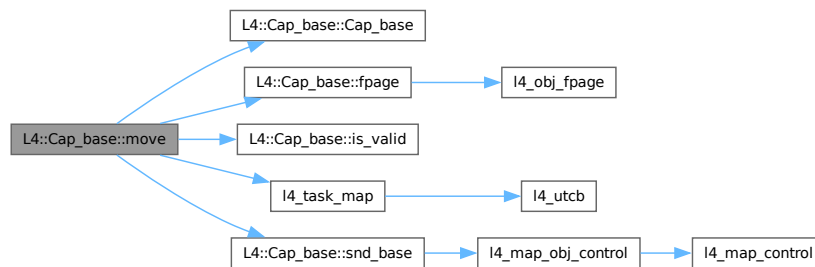
After the operation this capability refers to the object formerly referred to by the source capability `src`, and the source capability no longer refers to an object.

Definition at line 176 of file [capability.h](#).

References [Cap_base\(\)](#), [fpage\(\)](#), [is_valid\(\)](#), [L4_BASE_TASK_CAP](#), [L4_CAP_FPAGE_RWSD](#), [L4_FPAGE_C_OBJ_RIGHTS](#), [L4_MAP_ITEM_GRANT](#), [l4_task_map\(\)](#), and [snd_base\(\)](#).

Referenced by [L4::Cap< A >::move\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.82.4.6 `snd_base()`

```

l4_umword_t L4::Cap_base::snd_base (
    unsigned grant = L4_MAP_ITEM_MAP,
    l4_cap_idx_t base = L4_INVALID_CAP) const [inline], [noexcept]
  
```

Return send base.

Parameters

| | |
|--------------|---|
| <i>grant</i> | Indicates if object shall be granted. Allowed values: L4_MAP_ITEM_MAP , L4_MAP_ITEM_GRANT . |
| <i>base</i> | Base capability (first in a bundle of aligned capabilities) |

Returns

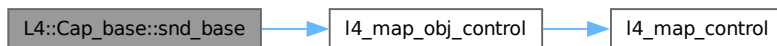
Map object.

Definition at line 86 of file [capability.h](#).

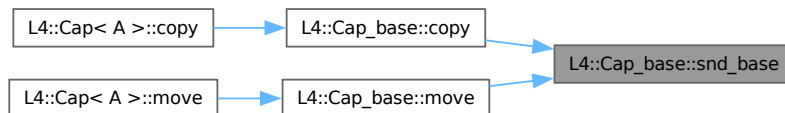
References [_c](#), [L4_INVALID_CAP](#), [L4_MAP_ITEM_MAP](#), and [l4_map_obj_control\(\)](#).

Referenced by [copy\(\)](#), and [move\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**15.82.4.7 validate() [1/2]**

```

l4_msgtag_t L4::Cap_base::validate (
    Cap< Task > task,
    l4_utcb_t * u = l4_utcb()) const [inline], [noexcept]
  
```

Check whether a capability is present (refers to an object).

Parameters

| | |
|-------------|--|
| <i>task</i> | Task to check the capability in. |
| <i>u</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |

Return values

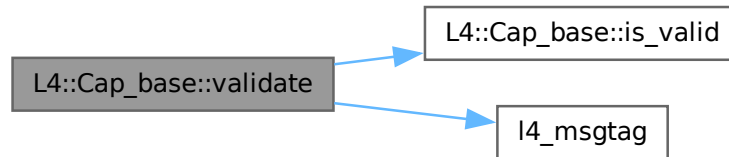
| | |
|----------------------------------|---|
| <i>l4_msgtag_t::label()</i> > 0 | Capability is present (refers to an object). |
| <i>l4_msgtag_t::label()</i> == 0 | No capability present (void object or invalid capability slot). |

A capability is considered present when it refers to an existing kernel object.

Definition at line 74 of file [capability](#).

References [_c](#), [is_valid\(\)](#), and [l4_msgtag\(\)](#).

Here is the call graph for this function:



15.82.4.8 validate() [2/2]

```
l4_msgtag_t L4::Cap_base::validate (
    l4_utcb_t * u = l4_utcb()) const [inline], [noexcept]
```

Check whether a capability is present (refers to an object).

Parameters

| | |
|----------|--|
| <i>u</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |
|----------|--|

Return values

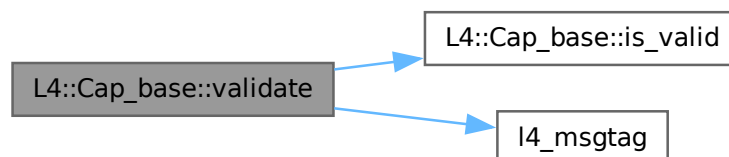
| | |
|----------------------------------|---|
| <i>l4_msgtag_t::label()</i> > 0 | Capability is present (refers to an object). |
| <i>l4_msgtag_t::label()</i> == 0 | No capability present (void object or invalid capability slot). |

A capability is considered present when it refers to an existing kernel object.

Definition at line 81 of file [capability](#).

References [_c](#), [is_valid\(\)](#), [L4_BASE_TASK_CAP](#), and [l4_msgtag\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following files:

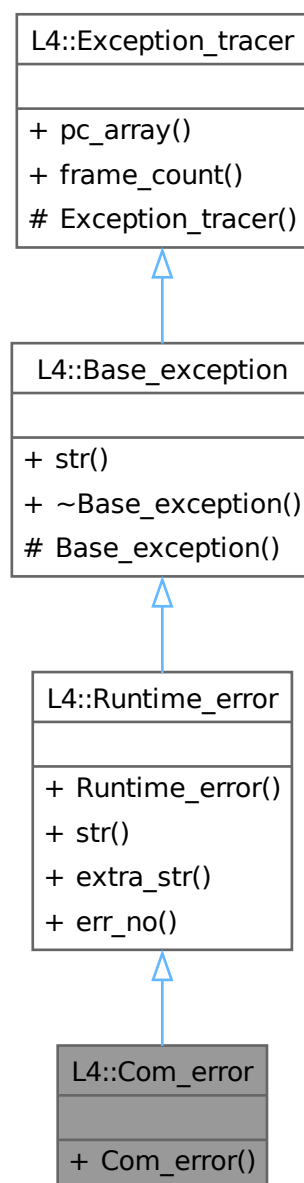
- l4/sys/cxx/capability.h
- l4/sys/[capability](#)

15.83 L4::Com_error Class Reference

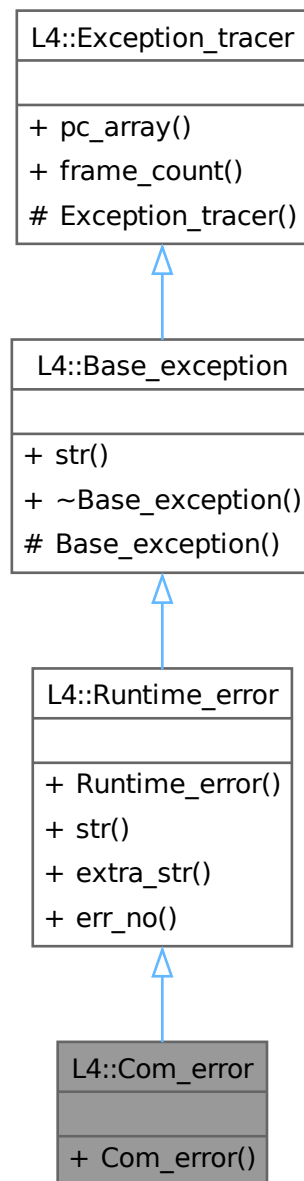
Error conditions during IPC.

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Com_error:



Collaboration diagram for L4::Com_error:



Public Member Functions

- [Com_error](#) (long err) noexcept
Create a [Com_error](#) for the given [L4](#) IPC error code.

Public Member Functions inherited from [L4::Runtime_error](#)

- [Runtime_error](#) (long [err_no](#), char const *extra=0) noexcept

Create a new [Runtime_error](#).

- char const * **str** () const noexcept override
Return a human readable string for the exception.
- char const * **extra_str** () const
Get the description text for this runtime error.
- long **err_no** () const noexcept
Get the error value for this runtime error.

Public Member Functions inherited from [L4::Base_exception](#)

- virtual ~**Base_exception** () noexcept
Destruction.

Public Member Functions inherited from [L4::Exception_tracer](#)

- void const *const * **pc_array** () const noexcept
Get the array containing the call trace.
- int **frame_count** () const noexcept
Get the number of entries that are valid in the call trace.

Additional Inherited Members

Protected Member Functions inherited from [L4::Base_exception](#)

- **Base_exception** () noexcept
Create a base exception.

Protected Member Functions inherited from [L4::Exception_tracer](#)

- **Exception_tracer** () noexcept
Create a back trace.

15.83.1 Detailed Description

Error conditions during IPC.

This exception encapsulates all IPC error conditions of [L4](#) IPC.

Definition at line [263](#) of file [exceptions](#).

15.83.2 Constructor & Destructor Documentation

15.83.2.1 Com_error()

```
L4::Com_error::Com_error (  
    long err) [inline], [explicit], [noexcept]
```

Create a [Com_error](#) for the given [L4](#) IPC error code.

Parameters

| | |
|------------|---|
| <i>err</i> | The L4 IPC error code (l4_ipc... return value). |
|------------|---|

Definition at line [270](#) of file [exceptions](#).

The documentation for this class was generated from the following file:

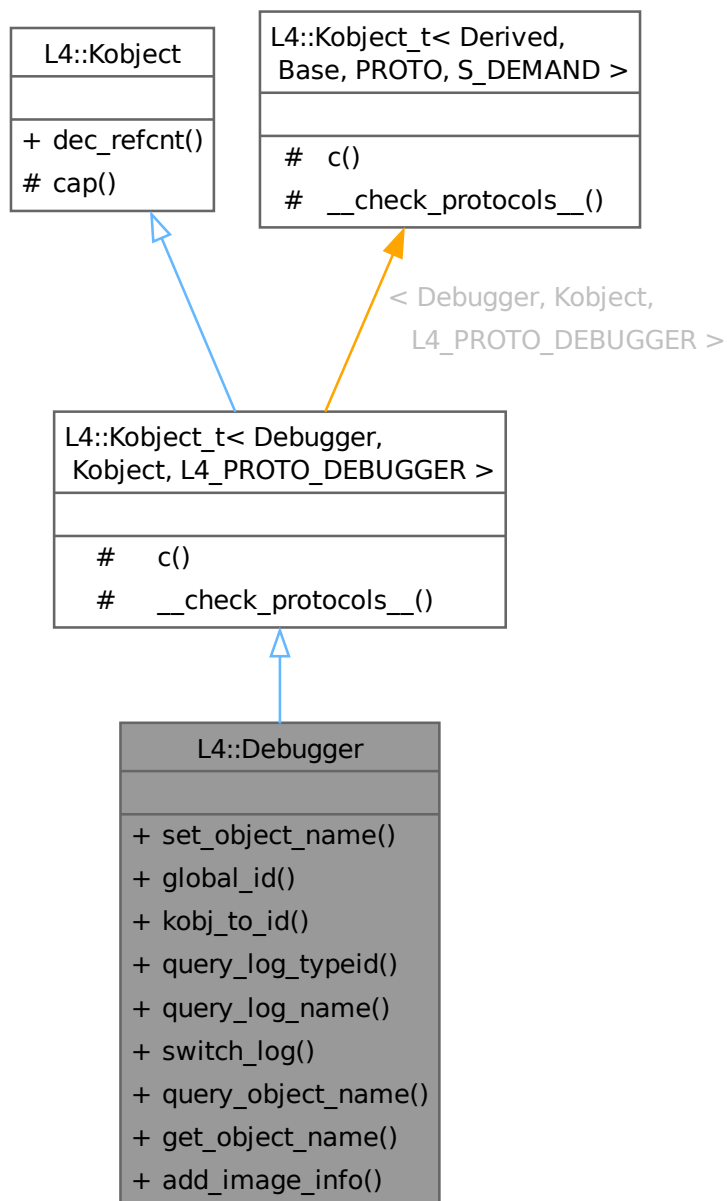
- [l4/cxx/exceptions](#)

15.84 L4::Debugger Class Reference

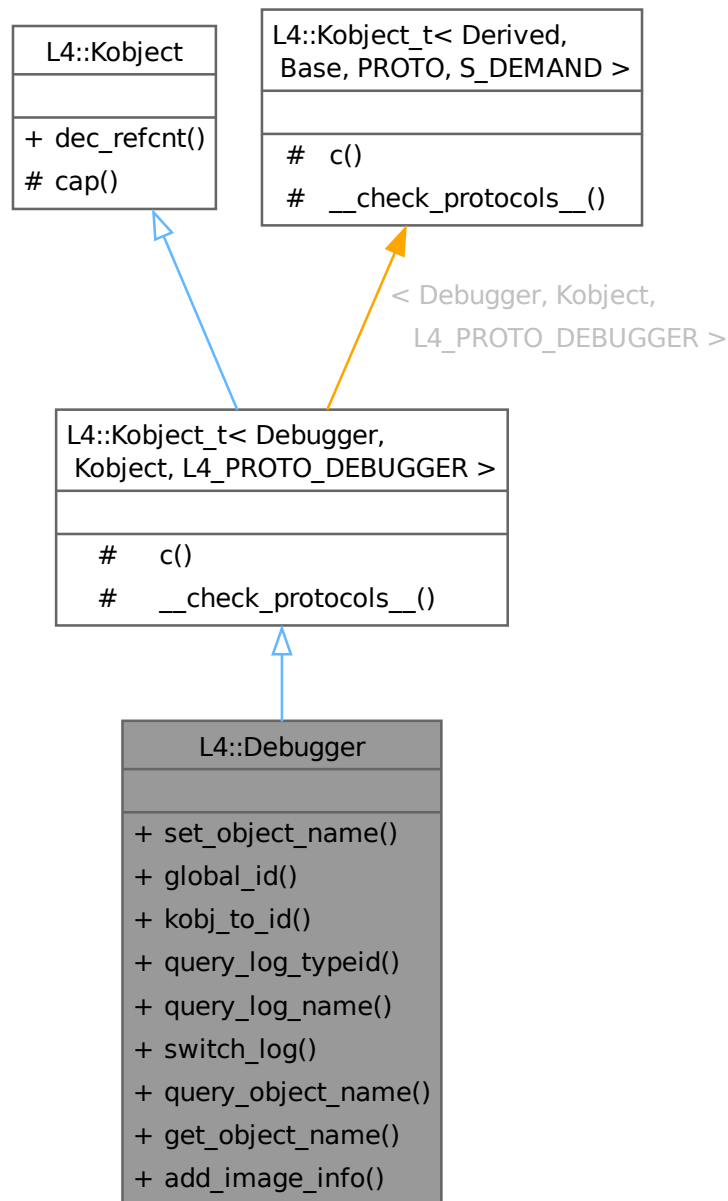
C++ kernel debugger API.

```
#include <debugger>
```

Inheritance diagram for L4::Debugger:



Collaboration diagram for L4::Debugger:



Public Member Functions

- [l4_msgtag_t set_object_name](#) (const char *name, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) noexcept
Set the name of a kernel object.
- unsigned long [global_id](#) ([l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) noexcept
Get the globally unique ID of the object behind a capability.
- unsigned long [kobj_to_id](#) ([l4_addr_t](#) kobjp, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) noexcept
Get the globally unique ID of the object behind the kobject pointer.

- [l4_ret_t query_log_typeid](#) (const char *name, unsigned idx, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) noexcept
Query the log-id for a log type.
- [l4_ret_t query_log_name](#) (unsigned idx, char *name, unsigned namelen, char *shortname, unsigned short-namelen, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) noexcept
Query the name of a log type given the ID.
- [l4_msgtag_t switch_log](#) (const char *name, unsigned on_off, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) noexcept
Set or unset log.
- [l4_msgtag_t query_object_name](#) (unsigned id, char *name, unsigned size, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) noexcept
Get name of object with Id id.
- [l4_msgtag_t get_object_name](#) (char *name, unsigned size, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) noexcept
Get name of object.
- [l4_msgtag_t add_image_info](#) ([l4_addr_t](#) base, const char *name, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) noexcept
Add loaded image information for a task.

Public Member Functions inherited from [L4::Kobject](#)

- [l4_msgtag_t dec_refcnt](#) ([l4_mword_t](#) diff, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#))
Decrement the in kernel reference counter for the object.

Additional Inherited Members

Protected Types inherited from

[L4::Kobject_t](#) < [Debugger](#), [Kobject](#), [L4_PROTO_DEBUGGER](#) >

- typedef [Debugger](#) **Class**
The target interface type (inheriting from [Kobject_t](#)).
- typedef Typeid::Iface< [PROTO](#), [Debugger](#) > **__Iface**
The interface description for the derived class.
- typedef Typeid::Merge_list< Typeid::Iface_list< [__Iface](#) >, typename [Kobject](#)::__Iface_list > **__Iface_list**
The list of all RPC interfaces provided directly or through inheritance.

Protected Member Functions inherited from

[L4::Kobject_t](#) < [Debugger](#), [Kobject](#), [L4_PROTO_DEBUGGER](#) >

- [L4::Cap](#) < [Class](#) > **c** () const noexcept
Get the capability to ourselves.

Protected Member Functions inherited from [L4::Kobject](#)

- [l4_cap_idx_t cap](#) () const noexcept
Return capability selector.

Static Protected Member Functions inherited from

[L4::Kobject_t](#) < [Debugger](#), [Kobject](#), [L4_PROTO_DEBUGGER](#) >

- static void **__check_protocols** () noexcept
Helper to check for protocol conflicts.

15.84.1 Detailed Description

C++ kernel debugger API.

Attention

This API is subject to change! Do not rely on it in production code.

This API is to be used for debugging exclusively.

This is the API for accessing kernel-debugger functionality from user-level programs. Specifically, it provides functionality to enrich the kernel debugger with insights into the program. The purpose is to facilitate debugging with the kernel debugger. For instance, a developer might choose to name the threads of her program so that she can find them in the kernel debugger thread list.

This API interacts with a kernel object that interfaces with the kernel debugger, the jdb-kernel object. The jdb-kernel object is fix and only available when the kernel debugger is built into the microkernel. The developer needs to pass the capability through to her program.

Include File

```
#include <l4/sys/debugger>
```

Definition at line 42 of file [debugger](#).

15.84.2 Member Function Documentation

15.84.2.1 add_image_info()

```
l4_msgtag_t L4::Debugger::add_image_info (
    l4_addr_t base,
    const char * name,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
```

Add loaded image information for a task.

Parameters

| | |
|-------------|------------------------------------|
| <i>base</i> | Image load base address |
| <i>name</i> | Image name |
| <i>utcb</i> | The UTCB to use for the operation. |

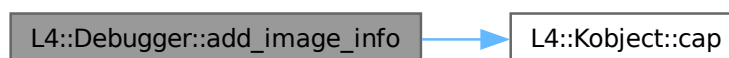
Returns

System call return tag.

Definition at line 175 of file [debugger](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



15.84.2.2 `get_object_name()`

```
l4_msgtag_t L4::Debugger::get_object_name (
    char * name,
    unsigned size,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
```

Get name of object.

Parameters

| | | |
|-----|-------------|--|
| out | <i>name</i> | Buffer to copy the name into. The buffer must be allocated by the caller. |
| | <i>size</i> | Length of the <i>name</i> buffer. |
| | <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |

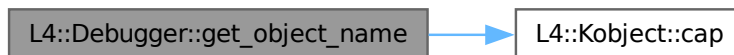
Returns

Syscall return tag

Definition at line 162 of file [debugger](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



15.84.2.3 `global_id()`

```
unsigned long L4::Debugger::global_id (
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
```

Get the globally unique ID of the object behind a capability.

Parameters

| | |
|-------------|--|
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb . |
|-------------|--|

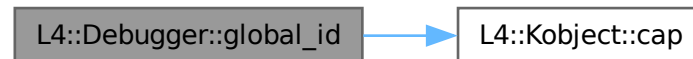
Return values

| | |
|------------|----------------------------|
| $\sim 0UL$ | The capability is invalid. |
| ≥ 0 | The global debugger id. |

Definition at line 71 of file [debugger](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



15.84.2.4 kobj_to_id()

```

unsigned long L4::Debugger::kobj_to_id (
    l4_addr_t kobjp,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
  
```

Get the globally unique ID of the object behind the kobject pointer.

Parameters

| | |
|--------------|--|
| <i>kobjp</i> | Kobject pointer |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |

Return values

| | |
|------------|--|
| $\sim 0UL$ | The capability or the Kobject pointer are invalid. |
| ≥ 0 | The globally unique id. |

Definition at line 83 of file [debugger](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



15.84.2.5 query_log_name()

```
l4_ret_t L4::Debugger::query_log_name (
    unsigned idx,
    char * name,
    unsigned namelen,
    char * shortname,
    unsigned shortnamelen,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
```

Query the name of a log type given the ID.

Parameters

| | | |
|-----|---------------------|--|
| | <i>idx</i> | ID to query. |
| out | <i>name</i> | Buffer to copy name to. The buffer must be allocated by the caller. |
| | <i>namelen</i> | Buffer length of name. |
| out | <i>shortname</i> | Buffer to copy shortname to. The buffer must be allocated by the caller. |
| | <i>shortnamelen</i> | Buffer length of shortname. |
| | <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |

Return values

| | |
|----|---------|
| 0 | Success |
| <0 | Error |

Definition at line 116 of file [debugger](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



15.84.2.6 query_log_typeid()

```
l4_ret_t L4::Debugger::query_log_typeid (
    const char * name,
```

```

    unsigned idx,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]

```

Query the log-id for a log type.

Parameters

| | |
|-------------|--|
| <i>name</i> | Name to query for. |
| <i>idx</i> | Idx to start searching, start with 0 |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |

Return values

| | |
|----------|-------|
| ≥ 0 | Id |
| < 0 | Error |

Definition at line 97 of file [debugger](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



15.84.2.7 query_object_name()

```

l4_msgtag_t L4::Debugger::query_object_name (
    unsigned id,
    char * name,
    unsigned size,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]

```

Get name of object with Id *id*.

Parameters

| | | |
|-----|-------------|---|
| | <i>id</i> | Id of the object whose name is asked. |
| out | <i>name</i> | Buffer to copy the name into. The buffer must be allocated by the caller. |
| | <i>size</i> | Length of the <i>name</i> buffer. |

| | | |
|--|-------------|--|
| | <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |
|--|-------------|--|

Returns

Syscall return tag

Definition at line 148 of file [debugger](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:

**15.84.2.8 set_object_name()**

```

l4_msgtag_t L4::Debugger::set_object_name (
    const char * name,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
  
```

Set the name of a kernel object.

Parameters

| | |
|-------------|--|
| <i>name</i> | Name |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb . |

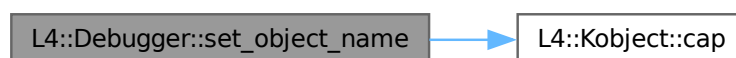
Returns

System call return tag.

Definition at line 59 of file [debugger](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



15.84.2.9 switch_log()

```
l4_msgtag_t L4::Debugger::switch_log (
    const char * name,
    unsigned on_off,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
```

Set or unset log.

Parameters

| | |
|---------------|--|
| <i>name</i> | Name of the log type. |
| <i>on_off</i> | 1: turn log on, 0: turn log off |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |

Returns

Syscall return tag

Definition at line [133](#) of file [debugger](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

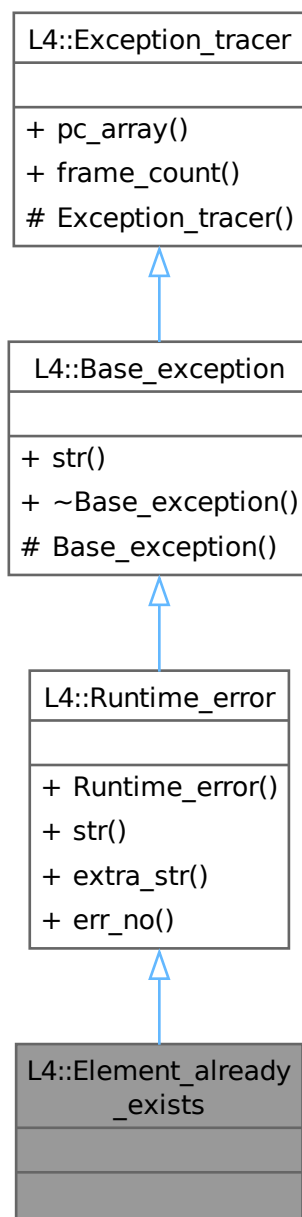
- [l4/sys/debugger](#)

15.85 L4::Element_already_exists Class Reference

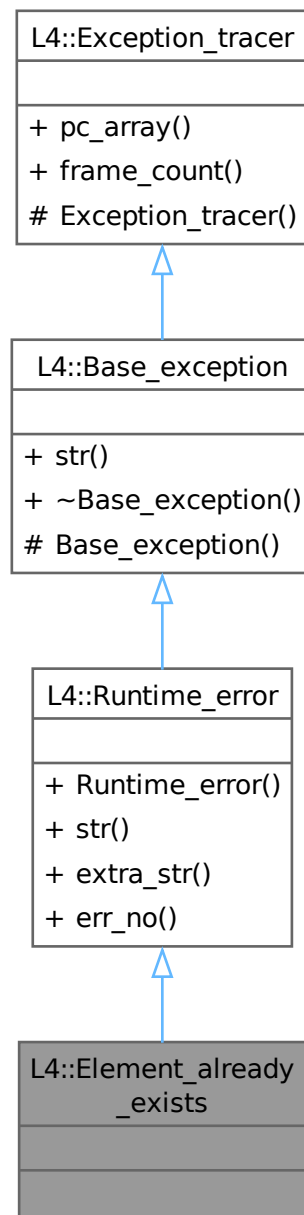
[Exception](#) for duplicate element insertions.

```
#include <l4/cxx/exceptions>
```


Inheritance diagram for L4::Element_already_exists:



Collaboration diagram for L4::Element_already_exists:



Additional Inherited Members

Public Member Functions inherited from [L4::Runtime_error](#)

- [Runtime_error](#) (long `err_no`, char const *extra=0) noexcept
Create a new [Runtime_error](#).
- char const * **str** () const noexcept override

Return a human readable string for the exception.

- `char const * extra_str () const`

Get the description text for this runtime error.

- `long err_no () const noexcept`

Get the error value for this runtime error.

Public Member Functions inherited from [L4::Base_exception](#)

- `virtual ~Base_exception () noexcept`

Destruction.

Public Member Functions inherited from [L4::Exception_tracer](#)

- `void const *const * pc_array () const noexcept`

Get the array containing the call trace.

- `int frame_count () const noexcept`

Get the number of entries that are valid in the call trace.

Protected Member Functions inherited from [L4::Base_exception](#)

- `Base_exception () noexcept`

Create a base exception.

Protected Member Functions inherited from [L4::Exception_tracer](#)

- `Exception_tracer () noexcept`

Create a back trace.

15.85.1 Detailed Description

[Exception](#) for duplicate element insertions.

Definition at line 192 of file [exceptions](#).

The documentation for this class was generated from the following file:

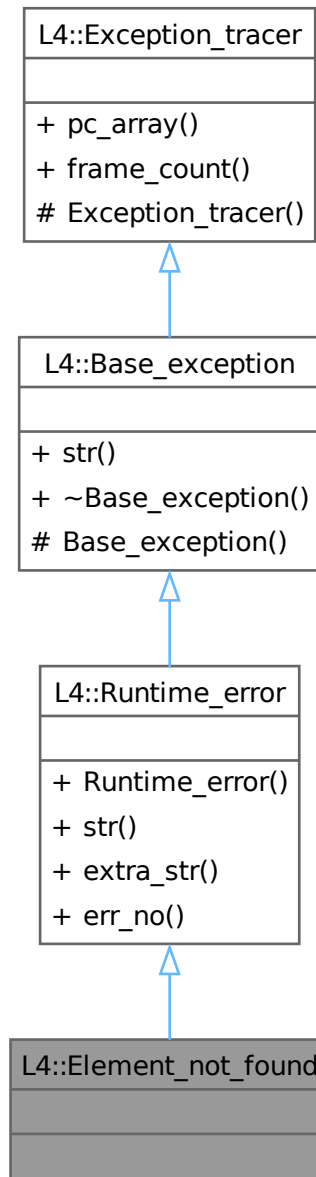
- [I4/cxx/exceptions](#)

15.86 L4::Element_not_found Class Reference

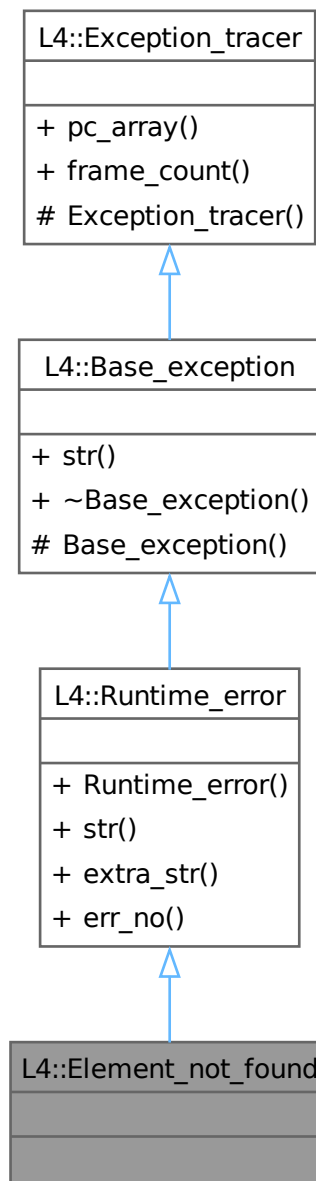
[Exception](#) for a failed lookup (element not found).

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Element_not_found:



Collaboration diagram for L4::Element_not_found:



Additional Inherited Members

Public Member Functions inherited from [L4::Runtime_error](#)

- [Runtime_error](#) (long `err_no`, char const *extra=0) noexcept
Create a new [Runtime_error](#).
- char const * **str** () const noexcept override
Return a human readable string for the exception.

- char const * [extra_str](#) () const
Get the description text for this runtime error.
- long [err_no](#) () const noexcept
Get the error value for this runtime error.

Public Member Functions inherited from [L4::Base_exception](#)

- virtual ~**Base_exception** () noexcept
Destruction.

Public Member Functions inherited from [L4::Exception_tracer](#)

- void const *const * **pc_array** () const noexcept
Get the array containing the call trace.
- int **frame_count** () const noexcept
Get the number of entries that are valid in the call trace.

Protected Member Functions inherited from [L4::Base_exception](#)

- **Base_exception** () noexcept
Create a base exception.

Protected Member Functions inherited from [L4::Exception_tracer](#)

- **Exception_tracer** () noexcept
Create a back trace.

15.86.1 Detailed Description

[Exception](#) for a failed lookup (element not found).

Definition at line 220 of file [exceptions](#).

The documentation for this class was generated from the following file:

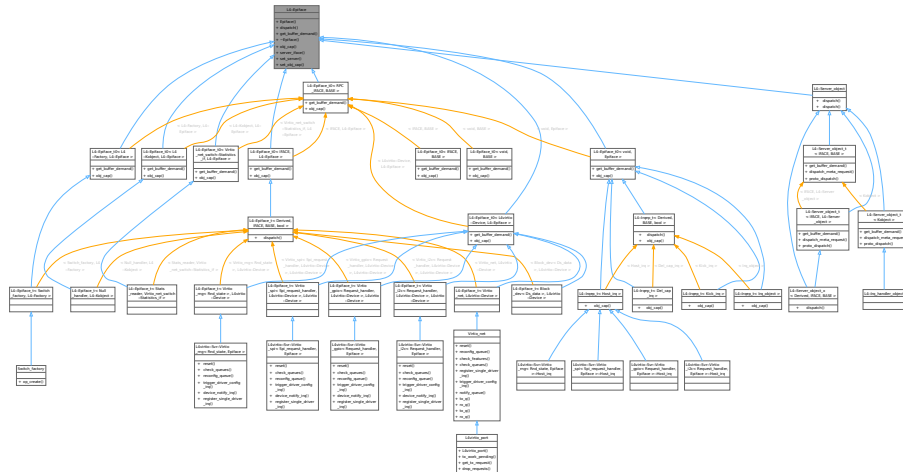
- [l4/cxx/exceptions](#)

15.87 L4::Epiface Struct Reference

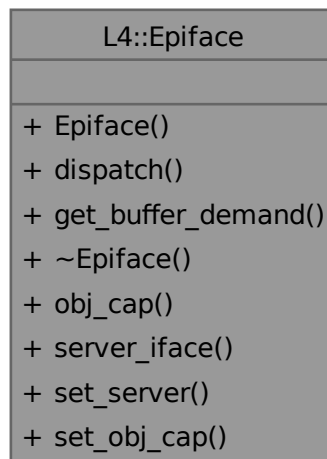
Base class for interface implementations.

```
#include <ipc_epiface>
```

Inheritance diagram for L4::Epiface:



Collaboration diagram for L4::Epiface:



Public Types

- using **Server_iface** = [lpc_svr::Server_iface](#)
Type for abstract server interface.
- using **Demand** = [lpc_svr::Server_iface::Demand](#)
Type for server-side receive buffer demand.

Public Member Functions

- **Epiface ()**
Make a server object.
- virtual [l4_msgtag_t dispatch](#) ([l4_msgtag_t](#) tag, unsigned rights, [l4_utcb_t](#) *utcb)=0
The abstract handler for client requests to the object.
- virtual [Demand get_buffer_demand](#) () const =0
Get the server-side receive buffer demand for this object.
- virtual **~Epiface ()**=0
Destroy the object.
- Stored_cap [obj_cap](#) () const
Get the capability to the kernel object belonging to this object.
- [Server_iface](#) * [server_iface](#) () const
Get pointer to server interface at which the object is currently registered.
- int [set_server](#) ([Server_iface](#) *srv, [Cap](#)< void > cap, bool managed=false)
Set server registration info for the object.
- void **set_obj_cap** ([Cap](#)< void > const &cap)
Deprecated server registration function.

15.87.1 Detailed Description

Base class for interface implementations.

An [Epiface](#) is the base interface of objects registered in the server loop. Incoming IPC gets dispatched to the appropriate [Epiface](#) object where the call is then handled appropriately.

Note

[Server](#) loops are allowed to internally keep raw pointers to [Epiface](#) objects for dispatching calls. Instances must therefore never be copied or moved.

Definition at line 257 of file [ipc_epiface](#).

15.87.2 Member Function Documentation

15.87.2.1 dispatch()

```
virtual l4\_msgtag\_t L4::Epiface::dispatch (
    l4\_msgtag\_t tag,
    unsigned rights,
    l4\_utcb\_t * utcb) [pure virtual]
```

The abstract handler for client requests to the object.

Parameters

| | |
|---------------|--|
| <i>tag</i> | The message tag for this invocation. |
| <i>rights</i> | The rights bits in the invoked capability. |

| | |
|-------------|-----------------------------------|
| <i>utcb</i> | The UTCB used for the invocation. |
|-------------|-----------------------------------|

Return values

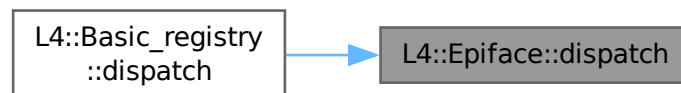
| | |
|---------------------------|-----------------------------------|
| <code>-L4_ENOREPLY</code> | No reply message is sent. |
| <code><0</code> | Error, reply with error code. |
| <code>>=0</code> | Success, reply with return value. |

This function must be implemented by application specific server objects.

Implemented in [L4::Epiface_t< Derived, IFACE, BASE, bool >](#), [L4::Epiface_t< Block_dev< Ds_data >, L4virtio::Device >](#), [L4::Epiface_t< Null_handler, L4::Kobject >](#), [L4::Epiface_t< Stats_reader, Virtio_net_switch::Statistics_if >](#), [L4::Epiface_t< Switch_factory, L4::Factory >](#), [L4::Epiface_t< Virtio_gpio< Request_handler, L4virtio::Device >, L4virtio::Device >](#), [L4::Epiface_t< Virtio_i2c< Request_handler, L4virtio::Device >, L4virtio::Device >](#), [L4::Epiface_t< Virtio_net, L4virtio::Device >](#), [L4::Epiface_t< Virtio_rng< Rnd_state >, L4virtio::Device >](#), [L4::Epiface_t< Virtio_spi< Spi_request_handler, L4virtio::Device >, L4virtio::Device >](#), [L4::lrqep_t< Derived, BASE, bool >](#), [L4::lrqep_t< Del_cap_irq >](#), [L4::lrqep_t< Host_irq >](#), [L4::lrqep_t< Irq_object >](#), [L4::lrqep_t< Kick_irq >](#), and [L4::Server_object](#).

Referenced by [L4::Basic_registry::dispatch\(\)](#).

Here is the caller graph for this function:



15.87.2.2 get_buffer_demand()

```
virtual Demand L4::Epiface::get_buffer_demand () const [pure virtual]
```

Get the server-side receive buffer demand for this object.

Note

This function is usually not implemented directly, but by using [Server_object_t](#) template with an IPC interface definition.

Returns

The needed server-side receive buffers for this object

Implemented in [L4::Epiface_t0< RPC_IFACE, BASE >](#), [L4::Epiface_t0< IFACE, L4::Epiface >](#), [L4::Epiface_t0< L4::Factory, L4::Epiface >](#), [L4::Epiface_t0< L4::Kobject, L4::Epiface >](#), [L4::Epiface_t0< L4virtio::Device, L4::Epiface >](#), [L4::Epiface_t0< Virtio_net_switch::Statistics_if, L4::Epiface >](#), [L4::Epiface_t0< void, Epiface >](#), [L4::Server_object_t< IFACE, BASE >](#), [L4::Server_object_t< IFACE, L4::Server_object >](#), and [L4::Server_object_t< Kobject >](#).

15.87.2.3 obj_cap()

```
Stored_cap L4::Epiface::obj_cap () const [inline]
```

Get the capability to the kernel object belonging to this object.

Returns

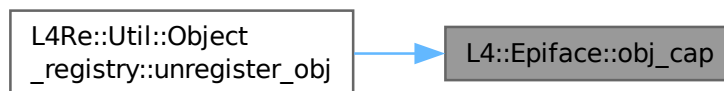
Capability for the kernel object behind the server.

This is usually either an [lpc_gate](#) or an [lrc](#).

Definition at line 318 of file [ipc_epiface](#).

Referenced by [L4Re::Util::Object_registry::unregister_obj\(\)](#).

Here is the caller graph for this function:



15.87.2.4 server_iface()

```
Server_iface * L4::Epiface::server_iface () const [inline]
```

Get pointer to server interface at which the object is currently registered.

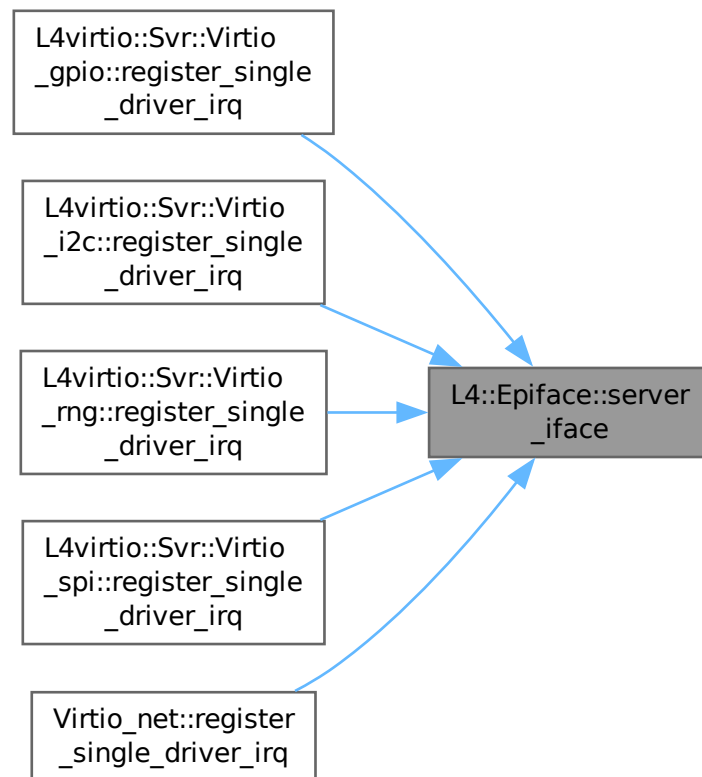
Returns

Pointer to the server at which the object is currently registered, NULL if the object is not registered at any server.

Definition at line 325 of file [ipc_epiface](#).

Referenced by [L4virtio::Svr::Virtio_gpio< Request_handler, Epiface >::register_single_driver_irq\(\)](#), [L4virtio::Svr::Virtio_i2c< Request_handler, Epiface >::register_single_driver_irq\(\)](#), [L4virtio::Svr::Virtio_rng< Rnd_state, Epiface >::register_single_driver_irq\(\)](#), [L4virtio::Svr::Virtio_spi< Spi_request_handler, Epiface >::register_single_driver_irq\(\)](#), and [Virtio_net::register_single_driver_irq\(\)](#).

Here is the caller graph for this function:



15.87.2.5 set_server()

```
int L4::Epiface::set_server (
    Server_iface * srv,
    Cap< void > cap,
    bool managed = false) [inline]
```

Set server registration info for the object.

Parameters

| | |
|----------------|---|
| <i>srv</i> | The server to register at |
| <i>cap</i> | The capability that connects the object. |
| <i>managed</i> | Mark the capability as managed or unmanaged. Typical server implementations use this flag to remember whether the capability was internally allocated or not. |

Returns

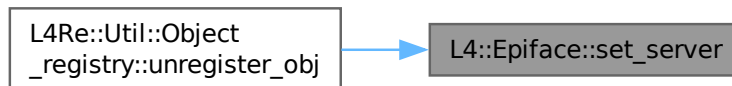
0 on success, -L4_EINVAL if the srv and cap are not consistent.

Definition at line 336 of file [ipc_epiface](#).

References [L4_EINVAL](#).

Referenced by [L4Re::Util::Object_registry::unregister_obj\(\)](#).

Here is the caller graph for this function:



The documentation for this struct was generated from the following file:

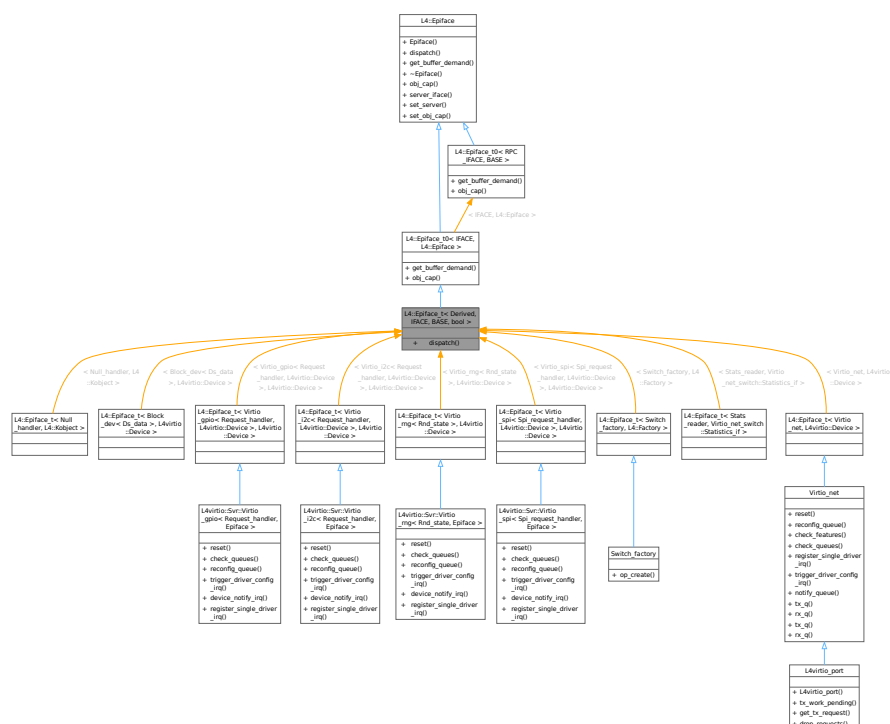
- [l4/sys/cxx/ipc_epiface](#)

15.88 L4::Epiface_t< Derived, IFACE, BASE, bool > Struct Template Reference

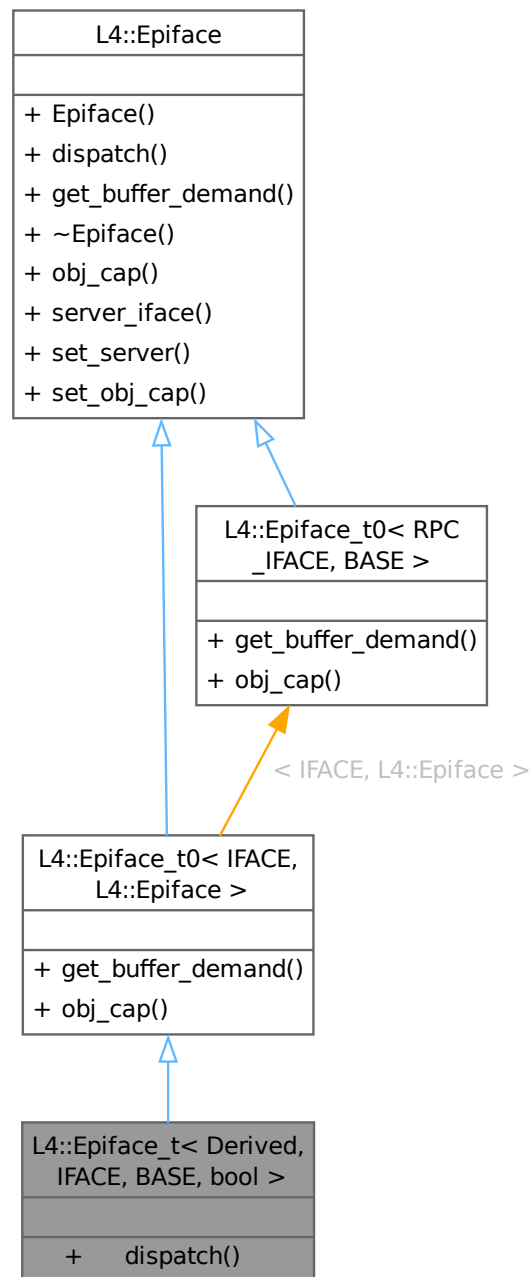
[Epiface](#) implementation for Kobject-based interface implementations.

```
#include <ipc_epiface>
```

Inheritance diagram for L4::Epiface_t< Derived, IFACE, BASE, bool >:



Collaboration diagram for L4::Epiface_t< Derived, IFACE, BASE, bool >:



Public Member Functions

- `l4_msgtag_t dispatch (l4_msgtag_t tag, unsigned rights, l4_utcb_t *utcb) final`
The abstract handler for client requests to the object.

Public Member Functions inherited from [L4::Epiface_t0< IFACE, L4::Epiface >](#)

- `Type_info::Demand get_buffer_demand () const`

Get the server-side buffer demand based in IFACE.

- `Cap< IFACE > obj_cap () const`

Get the (typed) capability to this object.

Public Member Functions inherited from `L4::Epiface`

- `Epiface ()`

Make a server object.

- `virtual ~Epiface ()=0`

Destroy the object.

- `Stored_cap obj_cap () const`

Get the capability to the kernel object belonging to this object.

- `Server_iface * server_iface () const`

Get pointer to server interface at which the object is currently registered.

- `int set_server (Server_iface *srv, Cap< void > cap, bool managed=false)`

Set server registration info for the object.

- `void set_obj_cap (Cap< void > const &cap)`

Deprecated server registration function.

Additional Inherited Members

Public Types inherited from `L4::Epiface_t< IFACE, L4::Epiface >`

- using `Interface`

Data type of the IPC interface definition.

Public Types inherited from `L4::Epiface`

- using `Server_iface = ipc_svr::Server_iface`

Type for abstract server interface.

- using `Demand = ipc_svr::Server_iface::Demand`

Type for server-side receive buffer demand.

15.88.1 Detailed Description

```
template<typename Derived, typename IFACE, typename BASE = L4::Epiface, bool = cxx::is_↵
polymorphic<BASE>::value>
struct L4::Epiface_t< Derived, IFACE, BASE, bool >
```

`Epiface` implementation for Kobject-based interface implementations.

Template Parameters

| | |
|----------------|--|
| <i>Derived</i> | Class providing the interface implementations. |
| <i>BASE</i> | <code>Epiface</code> base class. |

Examples

`examples/clntsrv/src/server.cc.`

Definition at line 658 of file `ipc_epiface`.

15.88.2 Member Function Documentation

15.88.2.1 dispatch()

```
template<typename Derived, typename IFACE, typename BASE = L4::Epiface, bool = cxx::is_↵
polymorphic<BASE>::value>
l4_msgtag_t L4::Epiface_t< Derived, IFACE, BASE, bool >::dispatch (
    l4_msgtag_t tag,
    unsigned rights,
    l4_utcb_t * utcb) [inline], [final], [virtual]
```

The abstract handler for client requests to the object.

Parameters

| | |
|---------------|--|
| <i>tag</i> | The message tag for this invocation. |
| <i>rights</i> | The rights bits in the invoked capability. |
| <i>utcb</i> | The UTCB used for the invocation. |

Return values

| | |
|---------------------------|-----------------------------------|
| <code>-L4_ENOREPLY</code> | No reply message is sent. |
| <code>< 0</code> | Error, reply with error code. |
| <code>>= 0</code> | Success, reply with return value. |

This function must be implemented by application specific server objects.

Implements [L4::Epiface](#).

Definition at line 661 of file [ipc_epiface](#).

The documentation for this struct was generated from the following file:

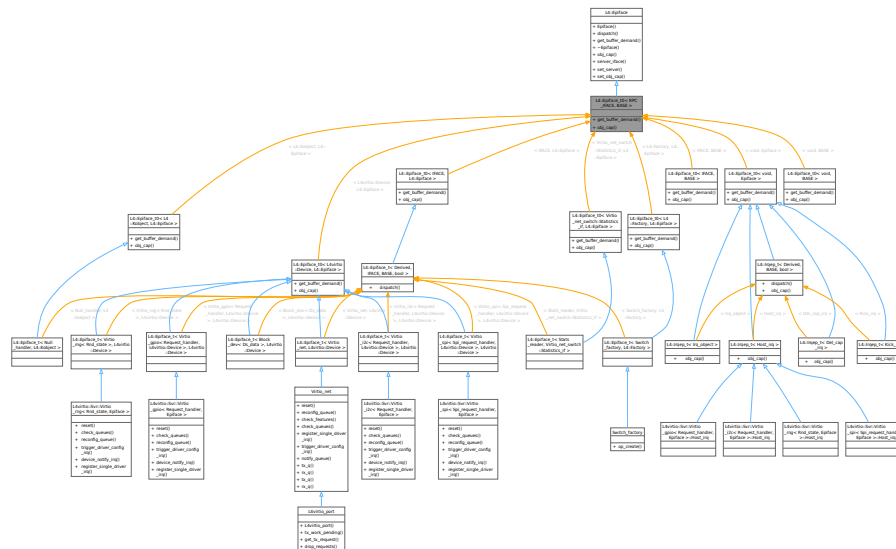
- `l4/sys/cxx/ipc_epiface`

15.89 L4::Epiface_t0< RPC_IFACE, BASE > Struct Template Reference

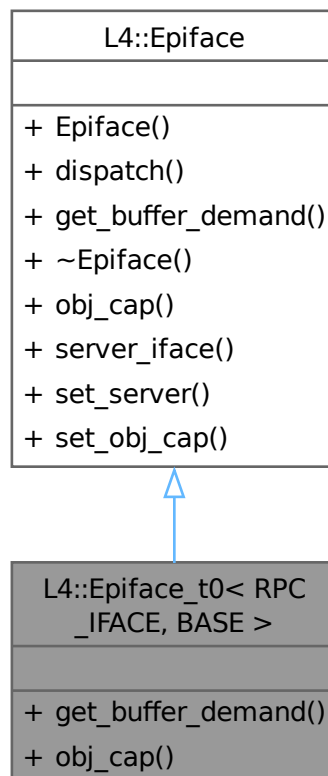
[Epiface](#) mixin for generic Kobject-based interfaces.

```
#include <ipc_epiface>
```

Inheritance diagram for L4::Epiface_t0< RPC_IFACE, BASE >:



Collaboration diagram for L4::Epiface_t0< RPC_IFACE, BASE >:



Public Types

- using **Interface** = RPC_IFACE
Data type of the IPC interface definition.

Public Types inherited from L4::Epiface

- using **Server_iface** = lpc_svr::Server_iface
Type for abstract server interface.
- using **Demand** = lpc_svr::Server_iface::Demand
Type for server-side receive buffer demand.

Public Member Functions

- [Type_info::Demand](#) **get_buffer_demand** () const
Get the server-side buffer demand based in IFACE.
- [Cap](#)< RPC_IFACE > **obj_cap** () const
Get the (typed) capability to this object.

Public Member Functions inherited from L4::Epiface

- **Epiface** ()
Make a server object.
- virtual [l4_msgtag_t](#) **dispatch** ([l4_msgtag_t](#) tag, unsigned rights, [l4_utcb_t](#) *utcb)=0
The abstract handler for client requests to the object.
- virtual ~**Epiface** ()=0
Destroy the object.
- Stored_cap [obj_cap](#) () const
Get the capability to the kernel object belonging to this object.
- [Server_iface](#) * **server_iface** () const
Get pointer to server interface at which the object is currently registered.
- int **set_server** ([Server_iface](#) *srv, [Cap](#)< void > cap, bool managed=false)
Set server registration info for the object.
- void **set_obj_cap** ([Cap](#)< void > const &cap)
Deprecated server registration function.

15.89.1 Detailed Description

```
template<typename RPC_IFACE, typename BASE = Epiface>
struct L4::Epiface_t0< RPC_IFACE, BASE >
```

[Epiface](#) mixin for generic Kobject-based interfaces.

Template Parameters

| | |
|---------------------------|--|
| RPC_IFACE | Data type of the IPC interface definition. |
|---------------------------|--|

| | |
|-------------|-------------------------------------|
| <i>BASE</i> | Base Epiface class. |
|-------------|-------------------------------------|

Definition at line [368](#) of file [ipc_epiface](#).

15.89.2 Member Function Documentation

15.89.2.1 `obj_cap()`

```
template<typename RPC_IFACE, typename BASE = Epiface>
Cap< RPC_IFACE > L4::Epiface_t0< RPC_IFACE, BASE >::obj_cap () const [inline]
```

Get the (typed) capability to this object.

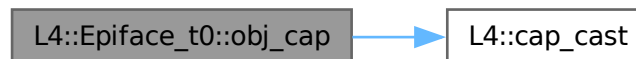
Returns

Capability for the kernel object behind the server.

Definition at line [381](#) of file [ipc_epiface](#).

References [L4::cap_cast\(\)](#).

Here is the call graph for this function:



The documentation for this struct was generated from the following file:

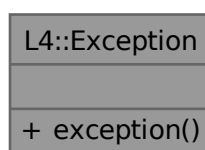
- `l4/sys/cxx/ipc_epiface`

15.90 L4::Exception Class Reference

[Exception](#) interface.

```
#include <exception>
```

Collaboration diagram for `L4::Exception`:



Public Member Functions

- [l4_msgtag_t exception](#) ([L4::lpc::In_out](#)< [l4_exc_regs_t](#) * > *regs*, [L4::lpc::Rcv_fpage](#) *rwin*, [L4::lpc::Opt](#)< [L4::lpc::Snd_fpage](#) & > *fp*)
Exception call.

15.90.1 Detailed Description

[Exception](#) interface.

This class defines the interface for handling exception IPC. When an exception occurs during program execution, for example due to a division by zero, the kernel will synthesise an exception IPC and send it to the thread's exception handler, who can then handle it.

The exception handler is set with the [L4::Thread::control](#) interface.

Definition at line 31 of file [exception](#).

15.90.2 Member Function Documentation

15.90.2.1 exception()

```
l4\_msgtag\_t L4::Exception::exception (
    L4::Ipc::In\_out< l4\_exc\_regs\_t * > regs,
    L4::Ipc::Rcv\_fpage rwin,
    L4::Ipc::Opt< L4::Ipc::Snd\_fpage & > fp)
```

[Exception](#) call.

Parameters

| | | |
|-----|-------------|---|
| | <i>regs</i> | Register state of the faulting thread. |
| | <i>rwin</i> | Receive window in the address space. |
| out | <i>fp</i> | Optional flexpage to resolve the exception. |

Returns

Message tag containing error code.

The documentation for this class was generated from the following file:

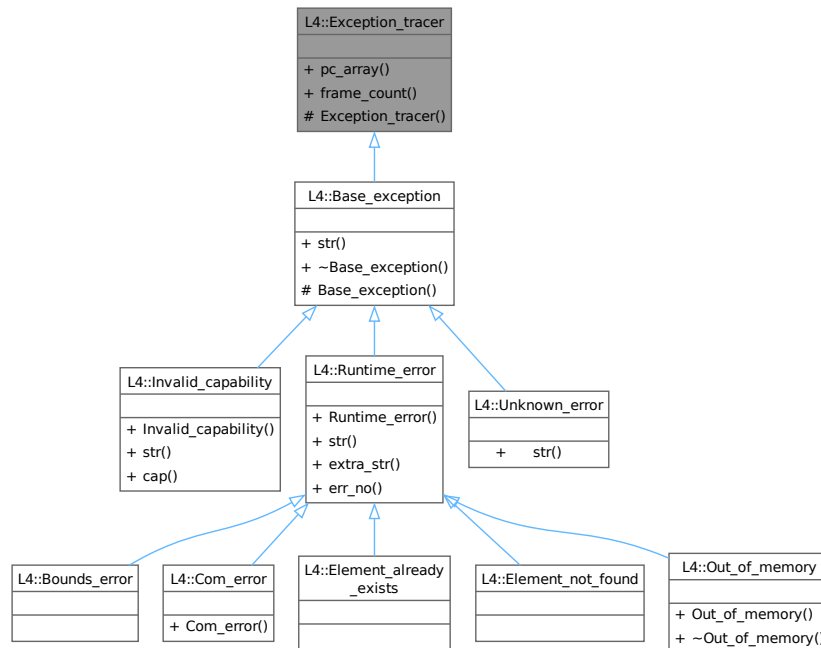
- [l4/sys/exception](#)

15.91 L4::Exception_tracer Class Reference

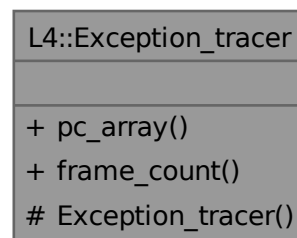
Back-trace support for exceptions.

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Exception_tracer:



Collaboration diagram for L4::Exception_tracer:



Public Member Functions

- void const *const * **pc_array** () const noexcept
Get the array containing the call trace.
- int **frame_count** () const noexcept
Get the number of entries that are valid in the call trace.

Protected Member Functions

- **Exception_tracer** () noexcept
Create a back trace.

15.91.1 Detailed Description

Back-trace support for exceptions.

This class holds an array of at most [L4_CXX_EXCEPTION_BACKTRACE](#) instruction pointers containing the call trace at the instant when an exception was thrown.

Definition at line 51 of file [exceptions](#).

The documentation for this class was generated from the following file:

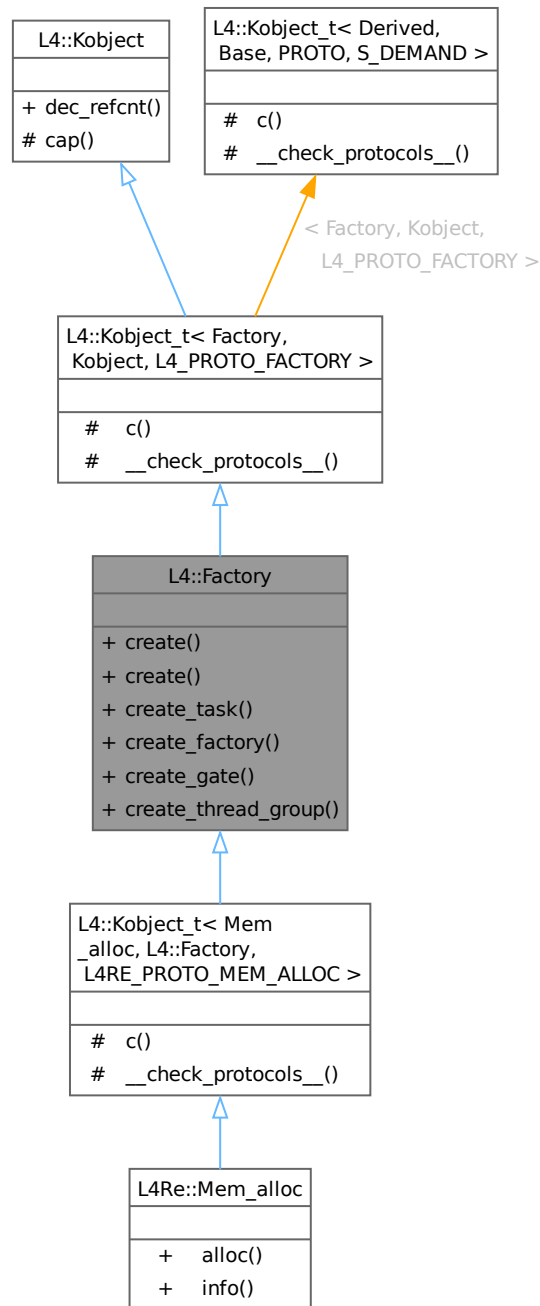
- [l4/cxx/exceptions](#)

15.92 L4::Factory Class Reference

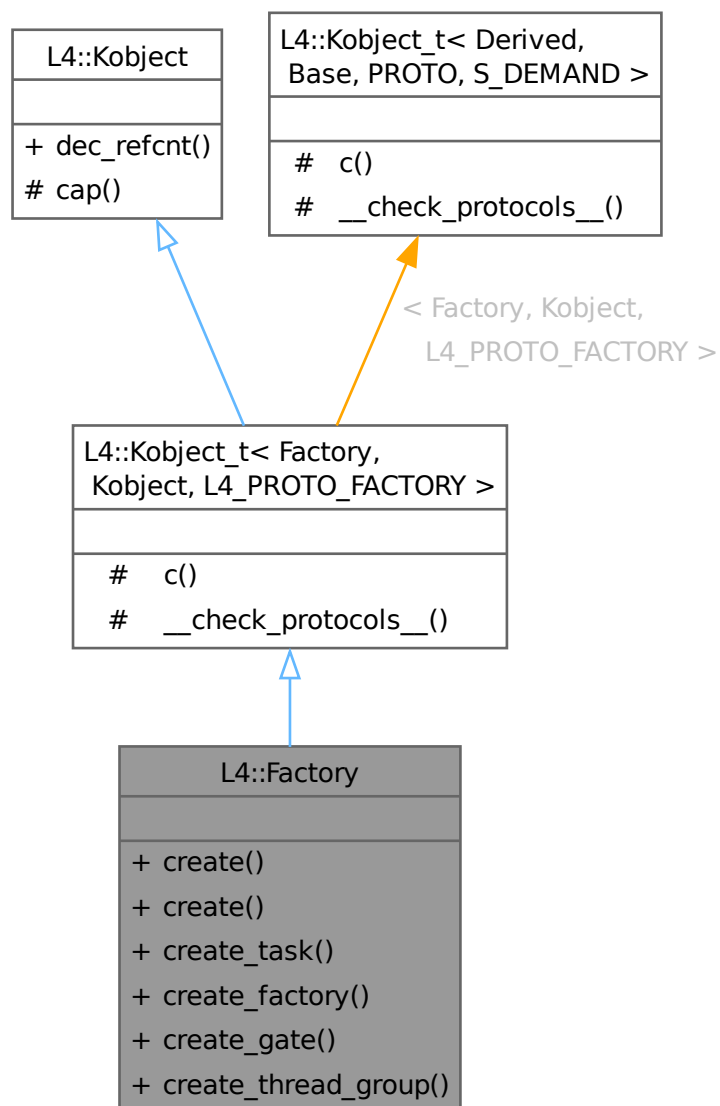
C++ Factory interface, see [Factory](#) for the C interface.

```
#include <factory>
```

Inheritance diagram for L4::Factory:



Collaboration diagram for L4::Factory:



Data Structures

- struct [Nil](#)
Special type to add a void argument into the factory create stream.
- struct [Lstr](#)
Special type to add a pascal string into the factory create stream.
- class [S](#)
Stream class for the [create\(\)](#) argument stream.

Public Member Functions

- **S create** (**Cap**< void > target, long obj, **l4_utcb_t** *utcb=**l4_utcb**()) noexcept
Generic create call to the factory.
- template<typename OBJ>
S create (**Cap**< OBJ > target, **l4_utcb_t** *utcb=**l4_utcb**()) noexcept
Create call for typed capabilities.
- **l4_msgtag_t create_task** (**Cap**< **Task** > const &target_cap, **l4_fpage_t** *utcb_area, **l4_utcb_t** *utcb=**l4_utcb**()) noexcept
Create a new task.
- **l4_msgtag_t create_factory** (**Cap**< **Factory** > const &target_cap, unsigned long limit, **l4_utcb_t** *utcb=**l4_utcb**()) noexcept
Create a new factory.
- **l4_msgtag_t create_gate** (**Cap**< void > const &target_cap, **Cap**< **Snd_destination** > const &snd_dst_cap, **l4_umword_t** label, **l4_utcb_t** *utcb=**l4_utcb**()) noexcept
Create a new IPC gate, optionally bound to a send destination (a thread or thread group).
- **l4_msgtag_t create_thread_group** (**Cap**< **Thread_group** > const &target_cap, unsigned policy, **l4_utcb_t** *utcb=**l4_utcb**()) noexcept
Create a new thread group.

Public Member Functions inherited from **L4::Kobject**

- **l4_msgtag_t dec_refcnt** (**l4_mword_t** diff, **l4_utcb_t** *utcb=**l4_utcb**())
Decrement the in kernel reference counter for the object.

Additional Inherited Members

Protected Types inherited from **L4::Kobject_t**< **Factory**, **Kobject**, **L4_PROTO_FACTORY** >

- typedef **Factory Class**
*The target interface type (inheriting from **Kobject_t**).*
- typedef Typeid::Iface< **PROTO**, **Factory** > **__Iface**
The interface description for the derived class.
- typedef Typeid::Merge_list< Typeid::Iface_list< **__Iface** >, typename **Kobject**::__Iface_list > **__Iface_list**
The list of all RPC interfaces provided directly or through inheritance.

Protected Member Functions inherited from **L4::Kobject_t**< **Factory**, **Kobject**, **L4_PROTO_FACTORY** >

- **L4::Cap**< **Class** > **c** () const noexcept
Get the capability to ourselves.

Protected Member Functions inherited from **L4::Kobject**

- **l4_cap_idx_t cap** () const noexcept
Return capability selector.

Static Protected Member Functions inherited from [L4::Kobject_t< Factory, Kobject, L4_PROTO_FACTORY >](#)

- static void `__check_protocols__()` noexcept
Helper to check for protocol conflicts.

15.92.1 Detailed Description

C++ Factory interface, see [Factory](#) for the C interface.

Factories provide an interface to create objects which are accessed via capabilities.

For additional information about which objects can be created via this interface, see server-specific information in [Kernel Factory](#) and [L4Re Servers](#).

Include File

```
#include <l4/sys/factory>
```

For the C interface refer to [Factory](#).

Definition at line 38 of file [factory](#).

15.92.2 Member Function Documentation

15.92.2.1 `create()` [1/2]

```
template<typename OBJ>
S L4::Factory::create (
    Cap< OBJ > target,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
```

Create call for typed capabilities.

Template Parameters

| | |
|------------|--|
| <i>OBJ</i> | Capability type of the object to be created. |
|------------|--|

Parameters

| | | |
|-----|---------------|--|
| out | <i>target</i> | Capability of type OBJ. |
| | <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb . |

Returns

A create stream that allows additional arguments to be passed to the `create()` call via the left-shift (`<<`) operator (see [S::operator <<](#)).

This method does not directly invoke the factory. The factory is invoked when the create stream returned by this method is converted to an [l4_msgtag_t](#) (see [S::operator l4_msgtag_t\(\)](#)), or otherwise when the stream goes out of scope (not recommended; see [S::~S\(\)](#)).

Precondition

The invoked [Factory](#) capability must have the permission [L4_CAP_FPAGE_S](#), otherwise the later factory IPC will fail with [L4_EPERM](#) (see [S::operator l4_msgtag_t\(\)](#)).

Note

The create stream uses the UTCB to store parameters for the service call. During the lifetime of a create stream or, until it is converted to an [l4_msgtag_t](#), other UTCB-using operations must not be used.

Usage:

```
L4::Cap<L4Re::Dataspace> ds = L4Re::Util::cap_alloc.alloc<L4Re::Dataspace>();
factory->create(ds) << l4_mword_t(size_in_bytes);
```

Definition at line 329 of file [factory](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:

**15.92.2.2 create() [2/2]**

```
S L4::Factory::create (
    Cap< void > target,
    long obj,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
```

Generic create call to the factory.

Parameters

| | | |
|-----|---------------|---|
| out | <i>target</i> | Capability selector for the new object. The caller must allocate the capability slot. The kernel stores the new object's capability into this slot. |
|-----|---------------|---|

| | |
|-------------|--|
| <i>obj</i> | The protocol ID that specifies which kind of object shall be created. |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb . |

Returns

A create stream that allows additional arguments to be passed to the `create()` call via the left-shift (`<<`) operator (see [S::operator <<](#)).

This method does not directly invoke the factory. The factory is invoked when the create stream returned by this method is converted to an [l4_msgtag_t](#) (see [S::operator l4_msgtag_t\(\)](#)), or otherwise when the stream goes out of scope (not recommended; see [S::~S\(\)](#)).

Precondition

The invoked [Factory](#) capability must have the permission [L4_CAP_FPAGE_S](#), otherwise the later factory IPC will fail with [L4_EPERM](#) (see [S::operator l4_msgtag_t\(\)](#)).

Note

The create stream uses the UTCB to store parameters for the service call. During the lifetime of a create stream or, until it is converted to an [l4_msgtag_t](#), other UTCB-using operations must not be used.

See also

[create\(Cap<OBJ>, l4_utcb_t *\)](#)

Definition at line 292 of file [factory](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



15.92.2.3 create_factory()

```
l4_msgtag_t L4::Factory::create_factory (
    Cap< Factory > const & target_cap,
    unsigned long limit,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
```

Create a new factory.

Parameters

| | | |
|-----|-------------------|--|
| out | <i>target_cap</i> | The kernel stores the new factory's capability into this slot. |
| | <i>limit</i> | Limit for the new factory in bytes. |
| | <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb . |

Returns

Syscall return tag

Return values

| | |
|------------------|---|
| <i>L4_EOK</i> | No error occurred. |
| <i>-L4_EPERM</i> | Insufficient permissions; see precondition. |
| <i><0</i> | Error code. |

Precondition

The invoked [Factory](#) capability must have the permission [L4_CAP_FPAGE_S](#).

Note

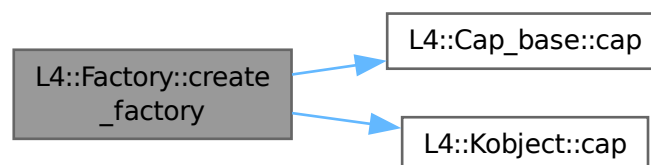
In addition to memory needed for internal data structures, the *limit* (quota) of the new factory is counted towards the quota of the creating factory. The *limit* must be within $1 \leq \text{limit} \leq 2^{(8 * \text{sizeof}(\text{l4_umword_t}) - 1) - 2}$ otherwise the behavior is undefined.

This method is only guaranteed to work with the [Kernel Factory](#). For other services, use the generic [create\(\)](#) method and consult the service documentation for information on the arguments that need to be passed to the create stream.

Definition at line [404](#) of file [factory](#).

References [L4::Cap_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



15.92.2.4 create_gate()

```
l4_msgtag_t L4::Factory::create_gate (
    Cap< void > const & target_cap,
    Cap< Snd_destination > const & snd_dst_cap,
    l4_umword_t label,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
```

Create a new IPC gate, optionally bound to a send destination (a thread or thread group).

Parameters

| | | |
|-----|--------------------|---|
| out | <i>target_cap</i> | The kernel stores the new IPC gate's capability into this slot. |
| | <i>snd_dst_cap</i> | Optional capability selector of a thread or thread group to bind the gate to. Use L4_INVALID_CAP to create an unbound IPC gate. |
| | <i>label</i> | Optional label of the gate (precisely used if <i>snd_dst_cap</i> is valid). If <i>snd_dst_cap</i> is valid, <i>label</i> must be present. |
| | <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb . |

Returns

Syscall return tag containing one of the following return codes.

Return values

| | |
|-------------------|---|
| <i>L4_EOK</i> | No error occurred. |
| <i>-L4_ENOMEM</i> | Out-of-memory during allocation of the lpc_gate object. |
| <i>-L4_EINVAL</i> | <i>snd_dst_cap</i> is void or points to something that is not a thread or thread group. |
| <i>-L4_EPERM</i> | Insufficient permissions; see precondition. |

Precondition

The invoked [Factory](#) capability must have the permission [L4_CAP_FPAGE_S](#). Also *snd_dst_cap* (if *valid*) must have the permission [L4_CAP_FPAGE_S](#).

An unbound IPC gate can be bound to a thread or thread group using [L4::lpc_gate::bind_thread\(\)](#) or [bind_snd_↔destination\(\)](#).

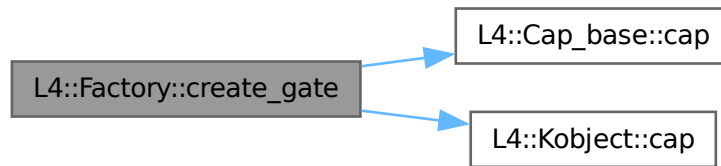
See also

[L4::lpc_gate](#)

Definition at line 440 of file [factory](#).

References [L4::Cap_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



15.92.2.5 create_task()

```

l4_msgtag_t L4::Factory::create_task (
    Cap< Task > const & target_cap,
    l4_fpage_t * utcb_area,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
  
```

Create a new task.

Parameters

| | | |
|---------|-------------------|--|
| out | <i>target_cap</i> | The kernel stores the new task's capability into this slot. |
| in, out | <i>utcb_area</i> | Flexpage that describes an area in the address space of the new task, where the kernel should map the kernel-allocated kernel-user memory to. The kernel uses the kernel-user memory to store UTCBs and vCPU state-save-areas of the new task. |

On systems without MMU, the flexpage is adjusted to reflect the actually allocated physical address.

Parameters

| | |
|-------------|--|
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb . |
|-------------|--|

Returns

Syscall return tag

Return values

| | |
|------------------|---|
| <i>L4_EOK</i> | No error occurred. |
| <i>-L4_EPERM</i> | Insufficient permissions; see precondition. |
| <i><0</i> | Error code. |

Precondition

The invoked [Factory](#) capability must have the permission [L4_CAP_FPAGE_S](#).

Note

The size of the UTCB area specifies indirectly the number of UTCBs available for this task. Refer to [L4::Task::add_ku_mem](#) for adding more of this type of memory.

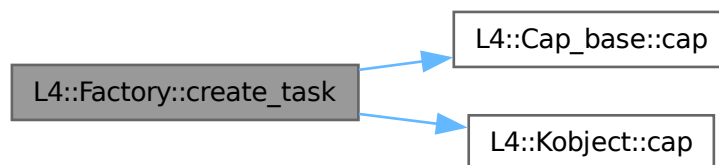
See also

[L4::Task](#)

Definition at line 370 of file [factory](#).

References [L4::Cap_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:

**15.92.2.6 create_thread_group()**

```

l4_msgtag_t L4::Factory::create_thread_group (
    Cap< Thread_group > const & target_cap,
    unsigned policy,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
  
```

Create a new thread group.

An IPC endpoint can be bound to a thread group. When a message arrives at the IPC endpoint, a specific thread of the thread group is selected to actually receive the message. A thread group is a send destination for an IPC endpoint.

Parameters

| | | |
|-----|-------------------|--|
| out | <i>target_cap</i> | The kernel stores the new thread group's capability into this slot. |
| | <i>policy</i> | Policy parameter for the thread group. See <code>L4_thread_group_policy</code> for a list of supported values. |

| | | |
|--|-------------|--|
| | <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb . |
|--|-------------|--|

Returns

Syscall return tag containing one of the following return codes.

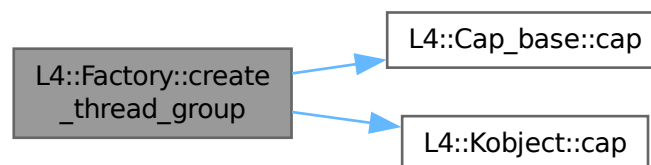
Return values

| | |
|-------------------|--|
| <i>L4_EOK</i> | No error occurred. |
| <i>-L4_ENOMEM</i> | Out-of-memory during allocation of the Thread_group object. |
| <i>-L4_EINVAL</i> | Invalid policy parameter. |
| <i>-L4_EPERM</i> | The factory instance requires L4_CAP_FPAGE_S rights on the invoked capability and L4_CAP_FPAGE_S is not present. |

Definition at line [475](#) of file [factory](#).

References [L4::Cap_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

- [l4/sys/factory](#)

15.93 L4::Factory::Lstr Struct Reference

Special type to add a pascal string into the factory create stream.

```
#include <factory>
```


Collaboration diagram for L4::Factory::Lstr:

| L4::Factory::Lstr | |
|-------------------|--------|
| + | s |
| + | len |
| + | Lstr() |

Public Member Functions

- [Lstr](#) (char const *[s](#), unsigned [len](#)) noexcept

Data Fields

- char const * **s**
The character buffer.
- unsigned **len**
The number of characters in the buffer.

15.93.1 Detailed Description

Special type to add a pascal string into the factory create stream.

This encapsulates a string that has an explicit length.

Definition at line [54](#) of file [factory](#).

15.93.2 Constructor & Destructor Documentation

15.93.2.1 Lstr()

```
L4::Factory::Lstr::Lstr (  
    char const * s,  
    unsigned len) [inline], [noexcept]
```

Parameters

| | |
|------------|---|
| <i>s</i> | Pointer to the c-style string. |
| <i>len</i> | Length in number of characters of the string <i>s</i> . |

Definition at line [70](#) of file [factory](#).

References [len](#), and [s](#).

The documentation for this struct was generated from the following file:

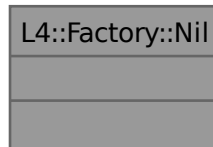
- [l4/sys/factory](#)

15.94 L4::Factory::Nil Struct Reference

Special type to add a void argument into the factory create stream.

```
#include <factory>
```

Collaboration diagram for L4::Factory::Nil:



15.94.1 Detailed Description

Special type to add a void argument into the factory create stream.

Definition at line 47 of file [factory](#).

The documentation for this struct was generated from the following file:

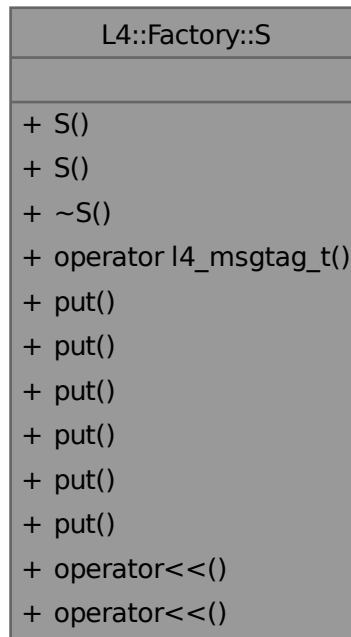
- [l4/sys/factory](#)

15.95 L4::Factory::S Class Reference

Stream class for the [create\(\)](#) argument stream.

```
#include <factory>
```

Collaboration diagram for L4::Factory::S:



Public Member Functions

- **S** (S &&o) noexcept
Move constructor.
- **S** (l4_cap_idx_t f, long obj, L4::Cap< void > target, l4_utcb_t *utcb) noexcept
Create a stream for a specific [create\(\)](#) call.
- **~S** () noexcept
Commit the [create\(\)](#) operation if not already done explicitly via [operator l4_msgtag_t\(\)](#).
- **operator l4_msgtag_t** () noexcept
Explicitly commits the operation and returns the result.
- void **put** (l4_mword_t i) noexcept
Put a single [l4_mword_t](#) as next argument.
- void **put** (l4_umword_t i) noexcept
Put a single [l4_umword_t](#) as next argument.
- void **put** (char const *s) &noexcept
Add a zero-terminated string as next argument.
- void **put** (Lstr const &s) &noexcept
Add a pascal string as next argument.
- void **put** (Nil) &noexcept
Add an empty argument.
- void **put** (l4_fpage_t d) &noexcept
Add a flexpage as next argument.
- template<typename T>
S & **operator<<** (T const &d) &noexcept

- Add next argument.*
- `template<typename T>`
`S && operator<< (T const &d) &&noexcept`
Add next argument.

15.95.1 Detailed Description

Stream class for the `create()` argument stream.

This stream allows a variable number of arguments to be added to a `create()` call.

Definition at line 79 of file `factory`.

15.95.2 Constructor & Destructor Documentation

15.95.2.1 S() [1/2]

```
L4::Factory::S::S (
    S && o) [inline], [noexcept]
```

Move constructor.

Parameters

| | |
|----------|-------------------------------------|
| <i>o</i> | Instance of <code>S</code> to move. |
|----------|-------------------------------------|

Definition at line 98 of file `factory`.

15.95.2.2 S() [2/2]

```
L4::Factory::S::S (
    l4_cap_idx_t f,
    long obj,
    L4::Cap< void > target,
    l4_utcb_t * utcb) [inline], [noexcept]
```

Create a stream for a specific `create()` call.

Parameters

| | | |
|-----|---------------|---|
| | <i>f</i> | The capability for the factory object (<code>L4::Factory</code>). |
| | <i>obj</i> | The protocol ID to describe the type of the object that shall be created. |
| out | <i>target</i> | The capability selector for the new object. The caller must allocate the capability slot. The kernel stores the new object's capability into this slot. |
| | <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See <code>l4_utcb</code> . |

Precondition

The capability `f` must have the permission `L4_CAP_FPAGE_S`, otherwise the later factory IPC will fail with `L4_EPERM`.

Definition at line 125 of file `factory`.

15.95.2.3 ~S()

```
L4::Factory::S::~~S () [inline], [noexcept]
```

Commit the [create\(\)](#) operation if not already done explicitly via [operator l4_msgtag_t\(\)](#).

Warning

If the commit is deferred until destruction, potential errors are silently ignored. It is therefore recommended to commit explicitly via [operator l4_msgtag_t\(\)](#) and check the return value.

Definition at line 139 of file [factory](#).

15.95.3 Member Function Documentation

15.95.3.1 operator l4_msgtag_t()

```
L4::Factory::S::operator l4_msgtag_t () [inline], [noexcept]
```

Explicitly commits the operation and returns the result.

Returns

The result of the [create\(\)](#) operation.

Return values

| | |
|------------------|---|
| <i>L4_EOK</i> | No error occurred. |
| <i>-L4_EPERM</i> | Insufficient permissions; see precondition. |
| <i><0</i> | Error code. |

Precondition

The invoked [Factory](#) capability must have the permission [L4_CAP_FPAGE_S](#).

Definition at line 157 of file [factory](#).

15.95.3.2 operator<<() [1/2]

```
template<typename T>
S && L4::Factory::S::operator<< (
    T const & d) && [inline], [noexcept]
```

Add next argument.

Template Parameters

| | |
|----------|--|
| <i>T</i> | The argument type. Compilation succeeds only if it is a possible argument type for <code>S::put()</code> . |
|----------|--|

Parameters

| | |
|----------|------------------------------------|
| <i>d</i> | The value to add as next argument. |
|----------|------------------------------------|

Definition at line 252 of file [factory](#).

References [put\(\)](#).

Here is the call graph for this function:

**15.95.3.3 operator<<() [2/2]**

```
template<typename T>
S & L4::Factory::S::operator<< (
    T const & d) & [inline], [noexcept]
```

Add next argument.

Template Parameters

| | |
|----------|--|
| <i>T</i> | The argument type. Compilation succeeds only if it is a possible argument type for <code>S::put()</code> . |
|----------|--|

Parameters

| | |
|----------|------------------------------------|
| <i>d</i> | The value to add as next argument. |
|----------|------------------------------------|

Definition at line 237 of file [factory](#).

References [put\(\)](#).

Here is the call graph for this function:



15.95.3.4 [put\(\)](#) [1/5]

```
void L4::Factory::S::put (
    char const * s) & [inline], [noexcept]
```

Add a zero-terminated string as next argument.

Parameters

| | |
|----------|-------------------------------------|
| <i>s</i> | The string to add as next argument. |
|----------|-------------------------------------|

The string will be added with the zero-terminator.

Definition at line 191 of file [factory](#).

15.95.3.5 [put\(\)](#) [2/5]

```
void L4::Factory::S::put (
    l4_fpage_t d) & [inline], [noexcept]
```

Add a flexpage as next argument.

Parameters

| | |
|----------|---|
| <i>d</i> | The flexpage to add (there will be no map operation). |
|----------|---|

Definition at line 223 of file [factory](#).

15.95.3.6 put() [3/5]

```
void L4::Factory::S::put (
    l4_mword_t i) [inline], [noexcept]
```

Put a single [l4_mword_t](#) as next argument.

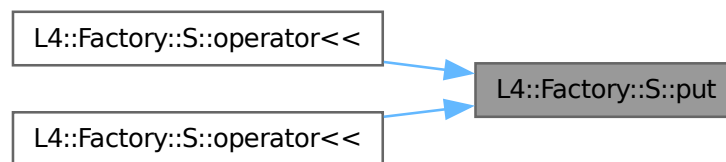
Parameters

| | |
|----------|------------------------------------|
| <i>i</i> | The value to add as next argument. |
|----------|------------------------------------|

Definition at line [169](#) of file [factory](#).

Referenced by [operator<<\(\)](#), and [operator<<\(\)](#).

Here is the caller graph for this function:

**15.95.3.7 put()** [4/5]

```
void L4::Factory::S::put (
    l4_umword_t i) [inline], [noexcept]
```

Put a single [l4_umword_t](#) as next argument.

Parameters

| | |
|----------|------------------------------------|
| <i>i</i> | The value to add as next argument. |
|----------|------------------------------------|

Definition at line [179](#) of file [factory](#).

15.95.3.8 put() [5/5]

```
void L4::Factory::S::put (  
    Lstr const & s) & [inline], [noexcept]
```

Add a pascal string as next argument.

Parameters

| | |
|---|-------------------------------------|
| s | The string to add as next argument. |
|---|-------------------------------------|

The string will be added with the exact length given. It is the responsibility of the caller to make sure that the string is zero- terminated when that is required by the server.

Definition at line [205](#) of file [factory](#).

The documentation for this class was generated from the following file:

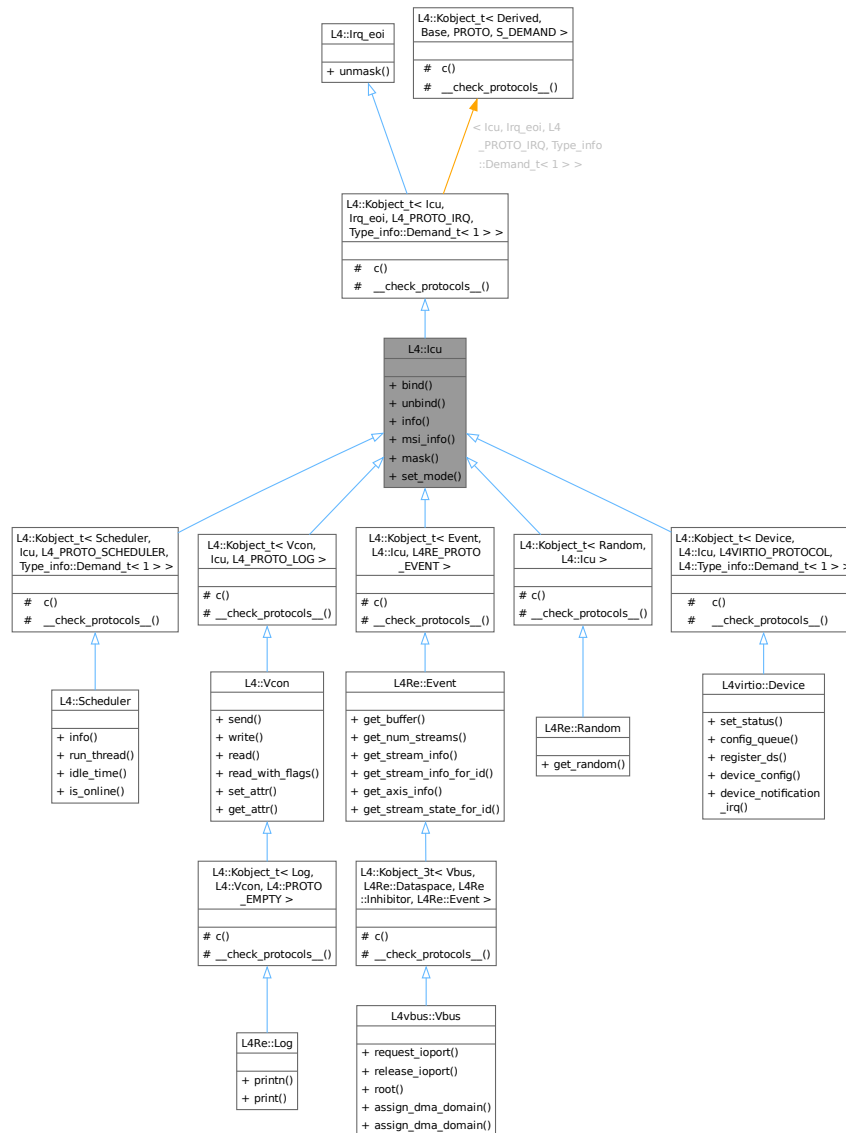
- [l4/sys/factory](#)

15.96 L4::lcu Class Reference

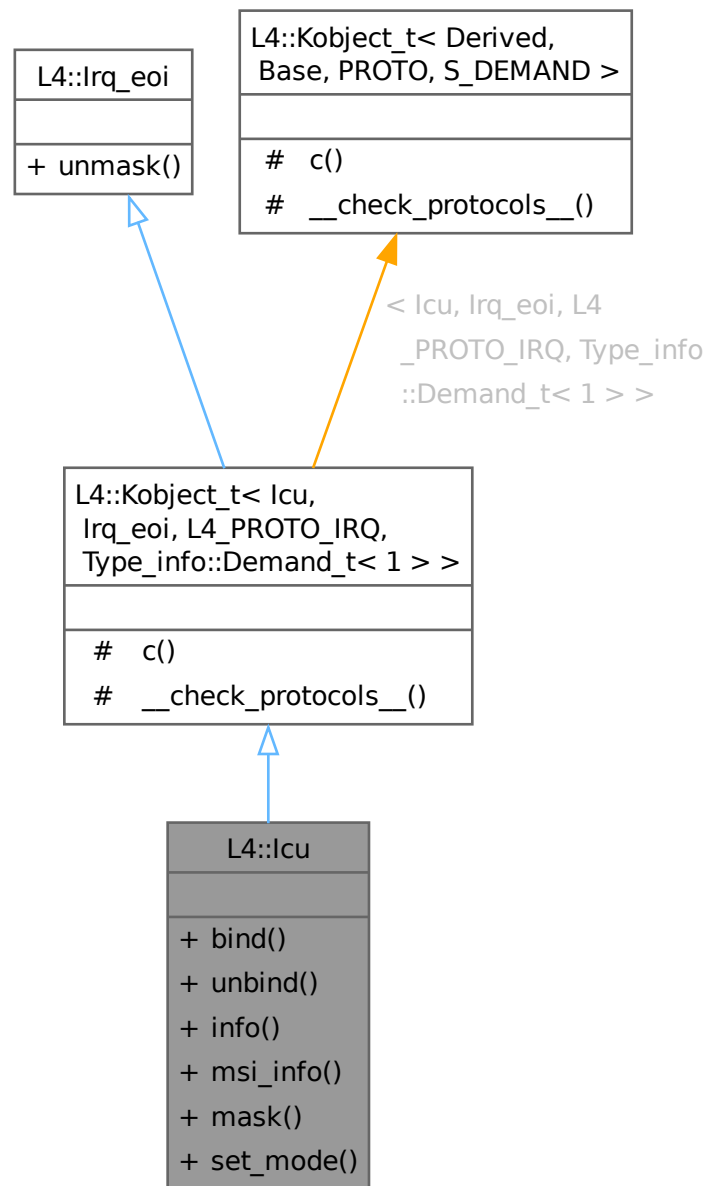
C++ [lcu](#) interface, see [Interrupt controller](#) for the C interface.

```
#include <irq>
```

Inheritance diagram for L4::Icu:



Collaboration diagram for L4::Icu:



Data Structures

- class [Info](#)

This class encapsulates information about an ICU.

Public Member Functions

- [I4_msgtag_t bind](#) (unsigned irqnum, [L4::Cap< Triggerable >](#) irq, [I4_utcb_t](#) *utcb=[I4_utcb\(\)](#)) noexcept

Bind an interrupt line of an interrupt controller to an interrupt object.

- [l4_msgtag_t unbind](#) (unsigned irqnum, [L4::Cap< Triggerable >](#) irq, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) noexcept

Remove binding of an interrupt line from the interrupt controller object.

- [l4_msgtag_t info](#) ([l4_icu_info_t](#) *info, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) noexcept

Get information about the ICU features.

- [l4_msgtag_t msi_info](#) ([l4_umword_t](#) irqnum, [l4_uint64_t](#) source, [l4_icu_msi_info_t](#) *msi_info)

Get MSI info about IRQ.

- [l4_msgtag_t mask](#) (unsigned irqnum, [l4_umword_t](#) *label=0, [l4_timeout_t](#) to=[L4_IPC_NEVER](#), [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) noexcept

Mask an IRQ line.

- [l4_msgtag_t set_mode](#) (unsigned irqnum, [l4_umword_t](#) mode, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) noexcept

Set interrupt mode.

Public Member Functions inherited from [L4::Irq_eoi](#)

- [l4_msgtag_t unmask](#) (unsigned irqnum, [l4_umword_t](#) *label=0, [l4_timeout_t](#) to=[L4_IPC_NEVER](#), [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) noexcept

Unmask the given interrupt line.

Additional Inherited Members

Protected Types inherited from

[L4::Kobject_t< lcu, Irq_eoi, L4_PROTO_IRQ, Type_info::Demand_t< 1 > >](#)

- typedef [lcu](#) **Class**

The target interface type (inheriting from [Kobject_t](#)).

- typedef [Typeid::Iface< PROTO, lcu >](#) **__Iface**

The interface description for the derived class.

- typedef [Typeid::Merge_list< Typeid::Iface_list< __Iface >, typename Irq_eoi::__Iface_list >](#) **__Iface_list**

The list of all RPC interfaces provided directly or through inheritance.

Protected Member Functions inherited from

[L4::Kobject_t< lcu, Irq_eoi, L4_PROTO_IRQ, Type_info::Demand_t< 1 > >](#)

- [L4::Cap< Class > c](#) () const noexcept

Get the capability to ourselves.

Static Protected Member Functions inherited from

[L4::Kobject_t< lcu, Irq_eoi, L4_PROTO_IRQ, Type_info::Demand_t< 1 > >](#)

- static void [__check_protocols__](#) () noexcept

Helper to check for protocol conflicts.

15.96.1 Detailed Description

C++ [Icu](#) interface, see [Interrupt controller](#) for the C interface.

Note

"ICU" is short for "interrupt control unit".

This class defines the interface for interrupt controllers. It defines functions for binding [L4::Irq](#) objects to interrupt lines and other interrupt sources, as well as functions for masking and unmasking of interrupts.

To setup an interrupt line the following steps are required:

1. [set_mode\(\)](#) (optional if interrupt has a default mode)
2. [L4::Rcv_endpoint::bind_thread\(\)](#) or [L4::Rcv_endpoint::bind_snd_destination\(\)](#) to attach the [L4::Irq](#) object to a send destination.
3. [bind\(\)](#)
4. [unmask\(\)](#) to receive the first interrupt

For certain interrupt sources only some of these steps are necessary and supported, see [L4::Scheduler](#) and [L4::Vcon](#).

At most one [L4::Irq](#) object can be bound to a certain interrupt source and a certain [L4::Irq](#) object can be bound to at most one interrupt source.

Include File

```
#include <l4/sys/icu>
```

Definition at line 250 of file [irq](#).

15.96.2 Member Function Documentation

15.96.2.1 bind()

```
l4_msgtag_t L4::Icu::bind (
    unsigned irqnum,
    L4::Cap< Triggerable > irq,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
```

Bind an interrupt line of an interrupt controller to an interrupt object.

Parameters

| | |
|---------------|--|
| <i>irqnum</i> | IRQ line at the ICU. |
| <i>irq</i> | IRQ object for the given IRQ line to bind to this ICU. |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |

Returns

Syscall return tag. The caller should check the return value using [l4_error\(\)](#) to check for errors and to identify the correct method for unmasking the interrupt. Return values < 0 indicate an error. A return value of 0 means a direct unmask via the IRQ object using [L4::irq::unmask](#). A return value of 1 means that the interrupt has to be unmasked via the ICU using [L4::icu::unmask](#).

Return values

| | |
|----------------------------|---|
| -L4_EINVAL | <code>irq</code> is bound to an interrupt source. |
| -L4_EPERM | Insufficient permissions; see precondition. |

Precondition

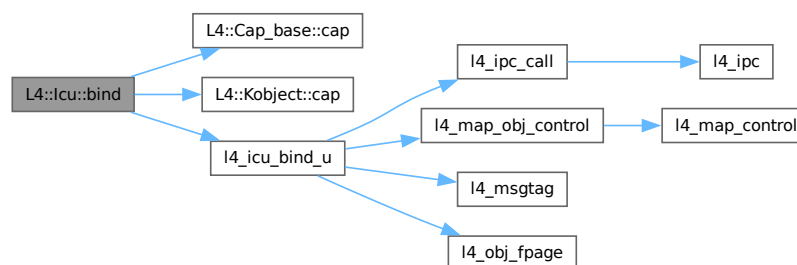
The capability `irq` must have the permission [L4_CAP_FPAGE_W](#).

In case the `irq` is already bound to an interrupt source, it is unbound first. In case the `irq` is bound and the interrupt source is bound to a different [L4::irq](#) object, only the unbinding happens. An [L4::irq](#) object that is bound to an interrupt source will get unbound if the [L4::irq](#) object is deleted.

Definition at line 310 of file [irq](#).

References [L4::Cap_base::cap\(\)](#), [L4::Kobject::cap\(\)](#), and [l4_icu_bind_u\(\)](#).

Here is the call graph for this function:

**15.96.2.2 info()**

```

l4_msgtag_t L4::Icu::info (
    l4_icu_info_t * info,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]

```

Get information about the ICU features.

Parameters

| | | |
|------------------|-------------------|---|
| <code>out</code> | <code>info</code> | Info structure to be filled with information. |
|------------------|-------------------|---|

| | | |
|--|-------------|--|
| | <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |
|--|-------------|--|

Returns

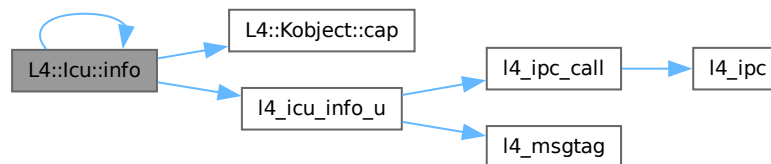
Syscall return tag

Definition at line 345 of file [irq](#).

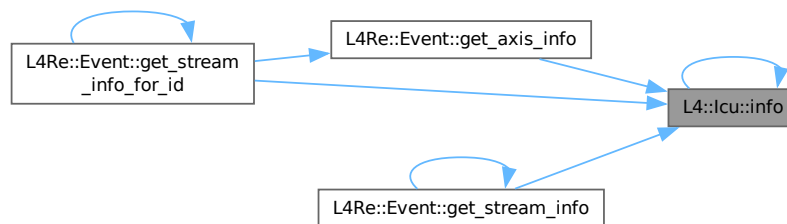
References [L4::Kobject::cap\(\)](#), [info\(\)](#), and [l4_icu_info_u\(\)](#).

Referenced by [L4Re::Event::get_axis_info\(\)](#), [L4Re::Event::get_stream_info\(\)](#), [L4Re::Event::get_stream_info_for_id\(\)](#), and [info\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.96.2.3 mask()

```

l4_msgtag_t L4::Icu::mask (
    unsigned irqnum,
    l4_umword_t * label = 0,
    l4_timeout_t to = L4_IPC_NEVER,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
  
```

Mask an IRQ line.

Parameters

| | |
|---------------|--|
| <i>irqnum</i> | IRQ line at the ICU. |
| <i>label</i> | If NULL, this function is a send-only message to the ICU. If not NULL, this function will enter an open wait after sending the mask message and the received label is returned here. |
| <i>to</i> | The timeout-pair (send and receive) that shall be used for this operation. The receive timeout is used with a non-NULL <i>label</i> only. |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |

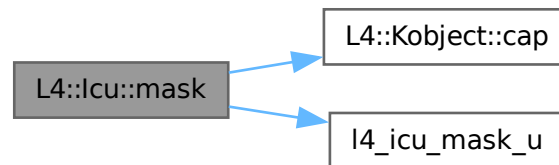
Returns

Syscall return tag. If *label* is NULL, this function performs an IPC send-only operation and there is no return value except [L4_MSGTAG_ERROR](#) indicating success or failure of the send operation. In this case use [l4_ipc_error\(\)](#) to check for errors and **do not** use [l4_error\(\)](#).

Definition at line 393 of file [irq](#).

References [L4::Kobject::cap\(\)](#), [l4_icu_mask_u\(\)](#), and [L4_IPC_NEVER](#).

Here is the call graph for this function:

**15.96.2.4 msi_info()**

```

l4_msgtag_t L4::Icu::msi_info (
    l4_umword_t irqnum,
    l4_uint64_t source,
    l4_icu_msi_info_t * msi_info)
  
```

Get MSI info about IRQ.

Parameters

| | | |
|-----|-----------------|---|
| | <i>irqnum</i> | IRQ line at the ICU. |
| | <i>source</i> | Platform dependent requester ID for MSIs. On IA32 we use a 20bit source filter value as described in the Intel IRQ remapping specification. |
| out | <i>msi_info</i> | A l4_icu_msi_info_t structure receiving the address and the data value to trigger this MSI. |

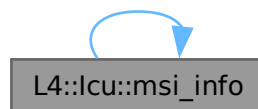
Returns

Syscall return tag

References [msi_info\(\)](#).

Referenced by [msi_info\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**15.96.2.5 set_mode()**

```

l4_msgtag_t L4::Icu::set_mode (
    unsigned irqnum,
    l4_umword_t mode,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
  
```

Set interrupt mode.

Parameters

| | |
|---------------|--|
| <i>irqnum</i> | IRQ line at the ICU. |
| <i>mode</i> | Mode, see L4_irq_mode . |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |

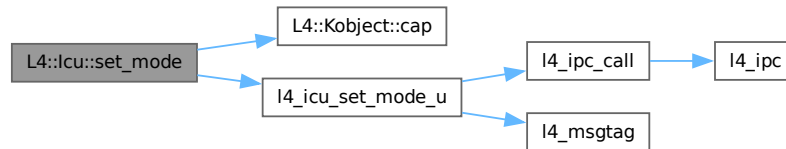
Returns

Syscall return tag

Definition at line 421 of file [irq](#).

References [L4::Kobject::cap\(\)](#), and [l4_icu_set_mode_u\(\)](#).

Here is the call graph for this function:

**15.96.2.6 unbind()**

```

l4_msgtag_t L4::Icu::unbind (
    unsigned irqnum,
    L4::Cap< Triggerable > irq,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
  
```

Remove binding of an interrupt line from the interrupt controller object.

Parameters

| | |
|---------------|--|
| <i>irqnum</i> | IRQ line at the ICU. |
| <i>irq</i> | IRQ object to remove from the ICU. |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |

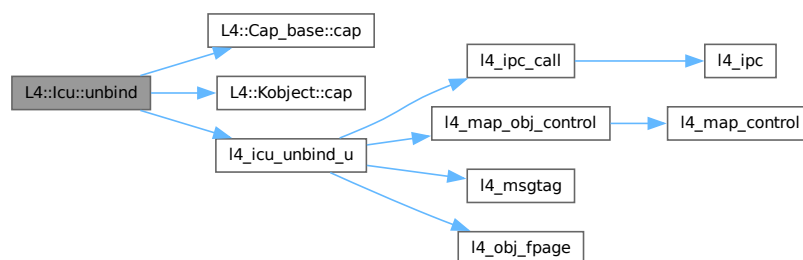
Returns

Syscall return tag

Definition at line 328 of file [irq](#).

References [L4::Cap_base::cap\(\)](#), [L4::Kobject::cap\(\)](#), and [l4_icu_unbind_u\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

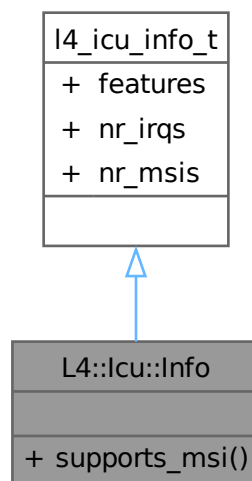
- [l4/sys/irq](#)

15.97 L4::lcu::Info Class Reference

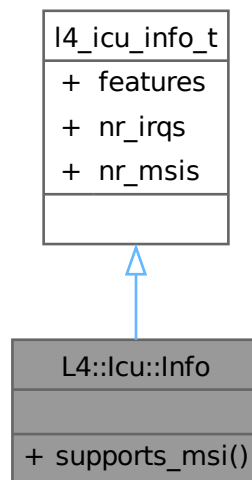
This class encapsulates information about an ICU.

```
#include <irq>
```

Inheritance diagram for L4::lcu::Info:



Collaboration diagram for L4::Icu::Info:



Public Member Functions

- bool **supports_msi** () const noexcept
True, if the ICU has support for MSIs.

Additional Inherited Members

Data Fields inherited from [l4_icu_info_t](#)

- unsigned [features](#)
Feature flags.
- unsigned **nr_irqs**
The number of IRQ lines supported by the ICU,.
- unsigned **nr_msis**
The number of MSI vectors supported by the ICU,.

15.97.1 Detailed Description

This class encapsulates information about an ICU.

Definition at line [277](#) of file [irq](#).

The documentation for this class was generated from the following file:

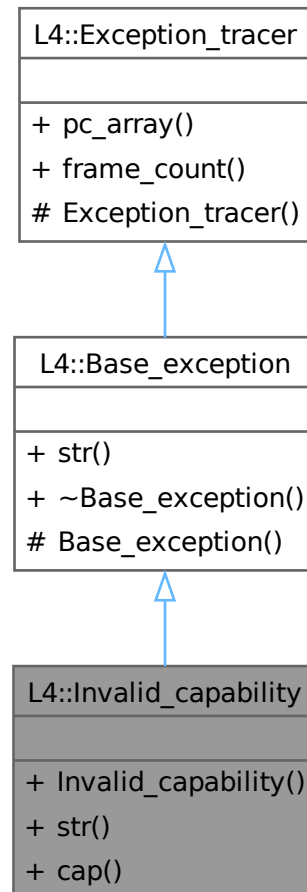
- [l4/sys/irq](#)

15.98 L4::Invalid_capability Class Reference

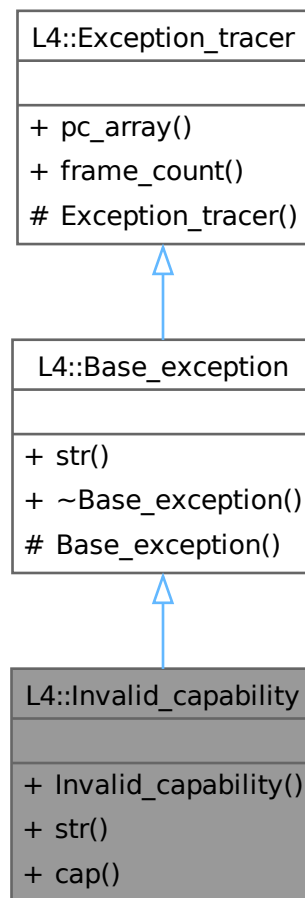
Indicates that an invalid object was invoked.

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Invalid_capability:



Collaboration diagram for L4::Invalid_capability:



Public Member Functions

- [Invalid_capability](#) ([Cap](#)< void > const &o) noexcept
Create an *Invalid_object* exception for the Object o.
- char const * **str** () const noexcept override
Return a human readable string for the exception.
- [Cap](#)< void > const & [cap](#) () const noexcept
Get the object that caused the error.

Public Member Functions inherited from [L4::Base_exception](#)

- virtual **~Base_exception** () noexcept
Destruction.

Public Member Functions inherited from [L4::Exception_tracer](#)

- void const *const * **pc_array** () const noexcept
Get the array containing the call trace.
- int **frame_count** () const noexcept
Get the number of entries that are valid in the call trace.

Additional Inherited Members

Protected Member Functions inherited from [L4::Base_exception](#)

- **Base_exception** () noexcept
Create a base exception.

Protected Member Functions inherited from [L4::Exception_tracer](#)

- **Exception_tracer** () noexcept
Create a back trace.

15.98.1 Detailed Description

Indicates that an invalid object was invoked.

An Object is invalid if it has L4_INVALID_ID as server [L4](#) UID, or if the server does not know the object ID.

Definition at line [234](#) of file [exceptions](#).

15.98.2 Constructor & Destructor Documentation

15.98.2.1 Invalid_capability()

```
L4::Invalid_capability::Invalid_capability (
    Cap< void > const & o) [inline], [explicit], [noexcept]
```

Create an Invalid_object exception for the Object o.

Parameters

| | |
|----------|---|
| o | The object that caused the server side error. |
|----------|---|

Definition at line [244](#) of file [exceptions](#).

15.98.3 Member Function Documentation

15.98.3.1 cap()

```
Cap< void > const & L4::Invalid_capability::cap () const [inline], [noexcept]
```

Get the object that caused the error.

Returns

The object that caused the error on invocation.

Definition at line 253 of file [exceptions](#).

The documentation for this class was generated from the following file:

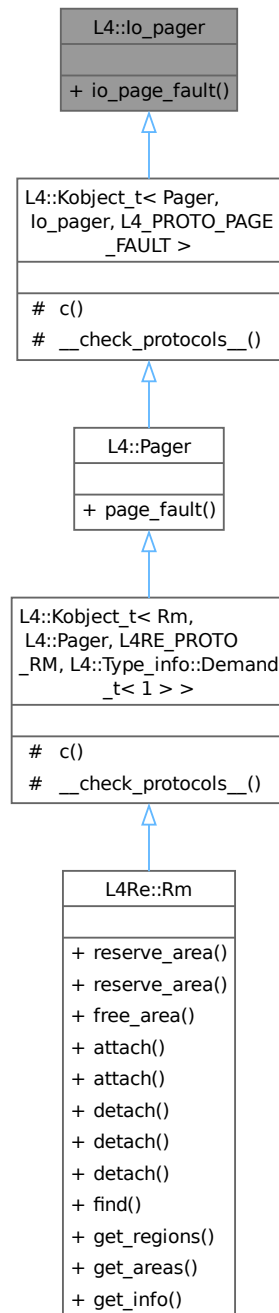
- [l4/cxx/exceptions](#)

15.99 L4::lo_pager Class Reference

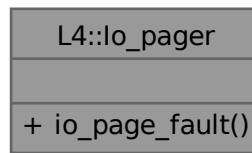
[lo_pager](#) interface.

```
#include <pager>
```


Inheritance diagram for L4::lo_pager:



Collaboration diagram for L4::io_pager:



Public Member Functions

- [l4_msgtag_t](#) [io_page_fault](#) ([l4_fpage_t](#) io_pfa, [l4_umword_t](#) pc, [L4::lpc::Rcv_fpage](#) rwin, [L4::lpc::Opt<L4::lpc::Snd_fpage & >](#) fp)
IO page fault protocol message.

15.99.1 Detailed Description

[io_pager](#) interface.

Note

This interface is IA32 specific.

This class defines the interface for handling IO page faults. IO page faults happen when a thread tries to access an IO port that it does not currently have access to.

Depending on the microkernel's implementation, IO page faults can be handled in two ways.

If the microkernel does not support IO page faults, this IO pagefault interface is not used. Instead, the microkernel sends an exception IPC to the thread's exception handler ([L4::Exception](#)), indicating a `GP` (exception number 13). The exception handler must consult the faulting instruction to determine the cause of the exception. This is the default in Fiasco.OC.

In contrast, if the microkernel supports IO page faults, the microkernel will generate an IO page fault message and send it to the thread's page fault handler (pager). The page fault handler can implement this interface to handle the IO page faults.

Note

A program may use this mechanism to implement a lazy IO port access scheme.

The page fault and exception handlers are set with the [L4::Thread::control](#) interface.

Definition at line 50 of file [pager](#).

15.99.2 Member Function Documentation

15.99.2.1 io_page_fault()

```
l4_msgtag_t L4::Io_pager::io_page_fault (
    l4_fpage_t io_pfa,
    l4_umword_t pc,
    L4::Ipc::Rcv_fpage rwin,
    L4::Ipc::Opt< L4::Ipc::Snd_fpage & > fp)
```

IO page fault protocol message.

Parameters

| | | |
|-----|---------------|---|
| | <i>io_pfa</i> | Flexpage describing the faulting IO-port. |
| | <i>pc</i> | Faulting program counter. |
| | <i>rwin</i> | The receive window for a flexpage mapping. |
| out | <i>fp</i> | Optional: flexpage descriptor to send to the task raising the page fault. |

Returns

System call message tag; use [l4_error\(\)](#) to check for errors.

IO-port fault messages are usually generated by the kernel and an IO-page-fault handler needs to be in place to handle such faults and generate a reply, potentially filling in *fp*.

The documentation for this class was generated from the following file:

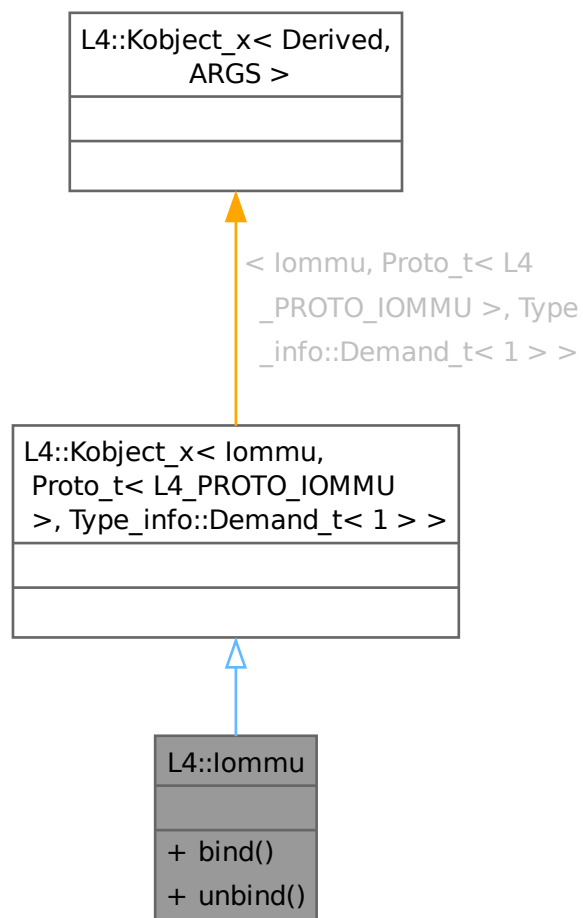
- [l4/sys/pager](#)

15.100 L4::lommu Class Reference

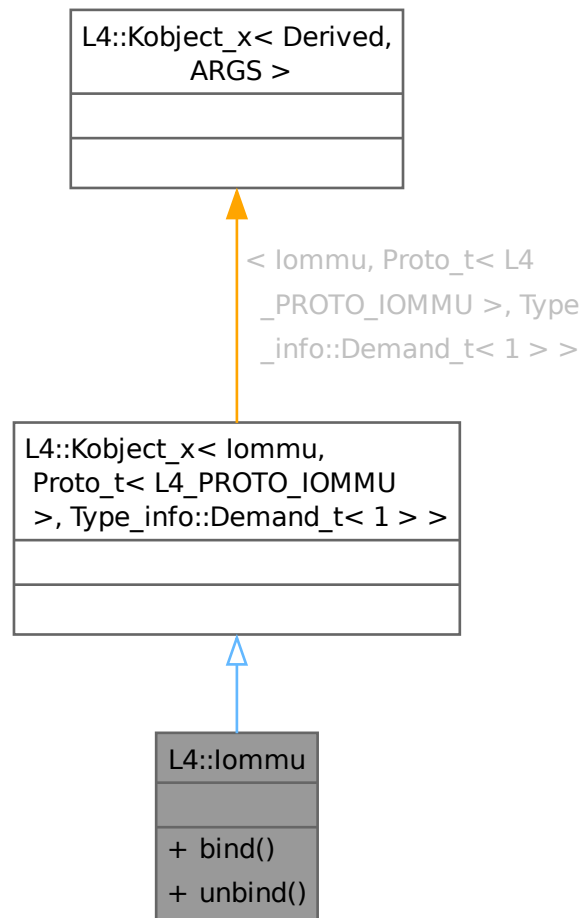
Interface for IO-MMUs used for DMA remapping.

```
#include <iommu>
```

Inheritance diagram for L4::lomu:



Collaboration diagram for L4::lomu:



Public Member Functions

- `l4_msgtag_t bind (l4_uint64_t src_id, lpc::Cap< Task > dma_space)`
Associate *dma_space* with the set of device(s) specified by *src_id*.
- `l4_msgtag_t unbind (l4_uint64_t src_id, lpc::Cap< Task > dma_space)`
Remove the association of the given DMA address space from the device(s) specified by *src_id*.

15.100.1 Detailed Description

Interface for IO-MMUs used for DMA remapping.

Note

This interface is only present in the kernel if the kernel detected an IOMMU during boot.

This interface allows to associate a DMA address space with a platform dependent set of devices. The kernel automatically keeps the memory spaces of associated DMA spaces in sync with the respective page table structures in the IOMMU.

Definition at line 21 of file [lomu](#).

15.100.2 Member Function Documentation

15.100.2.1 bind()

```
l4_msgtag_t L4::Iommu::bind (
    l4_uint64_t src_id,
    Ipc::Cap< Task > dma_space)
```

Associate `dma_space` with the set of device(s) specified by `src_id`.

Updates the respective page table structures in the IOMMU and keeps them in sync when memory is mapped to the `dma_space` or revoked from it.

Parameters

| | |
|------------------|--|
| <i>src_id</i> | Platform dependent source ID specifying the set of devices that shall use <code>dma_space</code> for DMA remapping. |
| <i>dma_space</i> | The DMA space (L4::Task created with <code>L4_PROTO_DMA_SPACE</code>) providing the mappings that shall be used for the device(s). |

References [bind\(\)](#).

Referenced by [bind\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.100.2.2 `unbind()`

```
l4_msgtag_t L4::Iommu::unbind (
    l4_uint64_t src_id,
    Ipc::Cap< Task > dma_space)
```

Remove the association of the given DMA address space from the device(s) specified by `src_id`.

Clear the respective page table structures in the IOMMU.

Parameters

| | |
|------------------------|---|
| <code>src_id</code> | Platform dependent source ID specifying the set of devices that shall no longer use <code>dma_space</code> for DMA remapping. |
| <code>dma_space</code> | The DMA space formerly associated with <code>bind()</code> . |

References `unbind()`.

Referenced by `unbind()`.

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

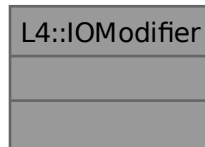
- `l4/sys/iommu`

15.101 L4::IOModifier Class Reference

Modifier class for the IO stream.

```
#include <basic_ostream>
```

Collaboration diagram for L4::IOModifier:



15.101.1 Detailed Description

Modifier class for the IO stream.

An IO Modifier can be used to change properties of an IO stream for example the number format.

Definition at line 22 of file [basic_ostream](#).

The documentation for this class was generated from the following file:

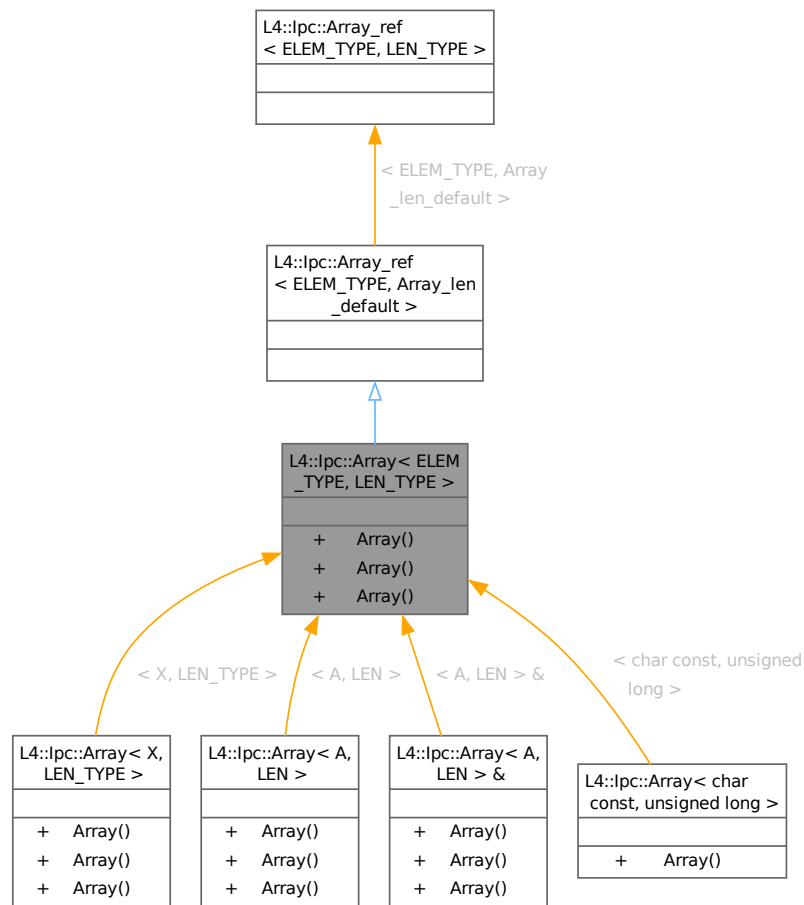
- [l4/cxx/basic_ostream](#)

15.102 L4::lpc::Array< ELEM_TYPE, LEN_TYPE > Struct Template Reference

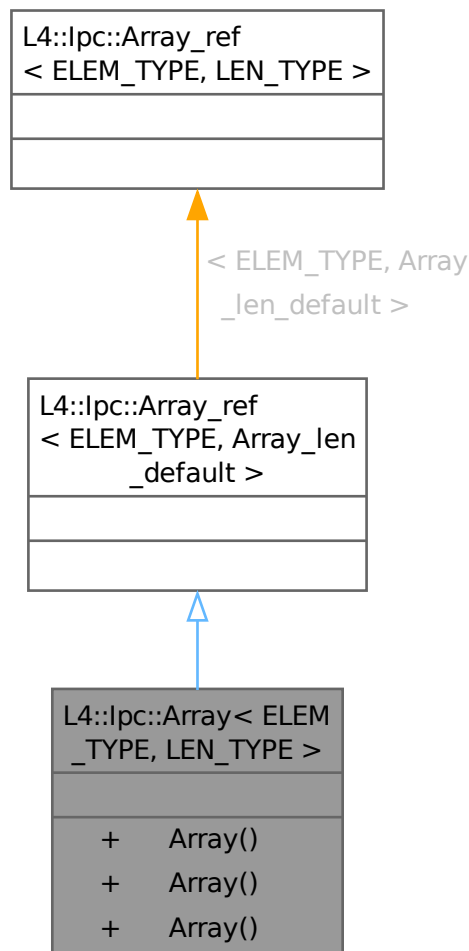
[Array](#) data type for dynamically sized arrays in RPCs.

```
#include <ipc_array>
```


Inheritance diagram for L4::lpc::Array< ELEM_TYPE, LEN_TYPE >:



Collaboration diagram for L4::lpc::Array< ELEM_TYPE, LEN_TYPE >:



Public Member Functions

- **Array** ()
Make array.
- **Array** (LEN_TYPE length, ELEM_TYPE *data)
Make array from length and data pointer.
- **Array** (typename Non_const< ELEM_TYPE >::type const &other)
Make a const array from a non-const array.

15.102.1 Detailed Description

```
template<typename ELEM_TYPE, typename LEN_TYPE = Array_len_default>
struct L4::lpc::Array< ELEM_TYPE, LEN_TYPE >
```

[Array](#) data type for dynamically sized arrays in RPCs.

Template Parameters

| | |
|------------------|--|
| <i>ELEM_TYPE</i> | The data type of an array element, should be 'const' when used as input. |
| <i>LEN_TYPE</i> | Data type used to store the number of elements in the array. |

An [Array](#) generally encapsulates a data pointer and a length (number of elements). [Array](#) does *not* provide any storage for the data itself. The storage is either provided by a client-side caller or in the case of [Array_ref](#) is the message itself.

Arrays can be used as input or as output arguments, when used as input *ELEM_TYPE* should be qualified *const*, when used as output a reference to an array must be used and the *ELEM_TYPE* must *not* be qualified *const*. It is the caller's responsibility to provide an array buffer of sufficient length. If a message from the server is too large it will be silently truncated.

If backward compatibility with `lpc::Stream` is required, then *LEN_TYPE* must be `unsigned long`.

Definition at line 81 of file [ipc_array](#).

The documentation for this struct was generated from the following file:

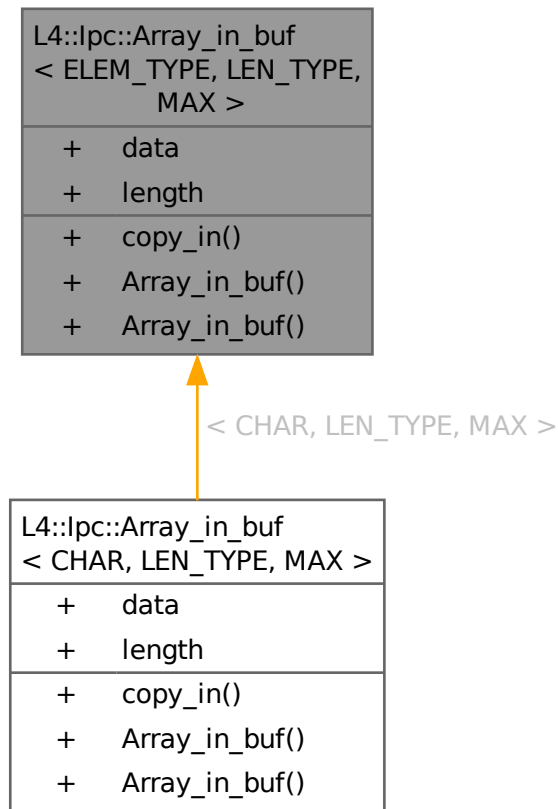
- `l4/sys/cxx/ipc_array`

15.103 L4::lpc::Array_in_buf< ELEM_TYPE, LEN_TYPE, MAX > Struct Template Reference

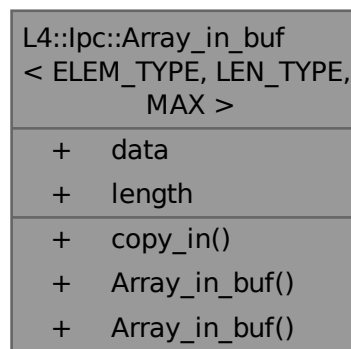
Server-side copy in buffer for [Array](#).

```
#include <ipc_array>
```

Inheritance diagram for L4::lpc::Array_in_buf< ELEM_TYPE, LEN_TYPE, MAX >:



Collaboration diagram for L4::lpc::Array_in_buf< ELEM_TYPE, LEN_TYPE, MAX >:



Public Member Functions

- void **copy_in** (const_array a)
copy in data from a source array
- **Array_in_buf** (const_array a)
Make [Array_in_buf](#) from a const array.
- **Array_in_buf** (array a)
Make [Array_in_buf](#) from a non-const array.

Data Fields

- ELEM_TYPE **data** [MAX]
The data elements.
- LEN_TYPE **length**
The length of the array.

15.103.1 Detailed Description

```
template<typename ELEM_TYPE, typename LEN_TYPE = Array_len_default, LEN_TYPE MAX = (L4_↔
UTCB_GENERIC_DATA_SIZE * sizeof(I4_umword_t)) / sizeof(ELEM_TYPE)>
struct L4::lpc::Array_in_buf< ELEM_TYPE, LEN_TYPE, MAX >
```

Server-side copy in buffer for [Array](#).

Template Parameters

| | |
|------------------|---|
| <i>ELEM_TYPE</i> | Data type of an array element. |
| <i>LEN_TYPE</i> | Data type for the number of elements in the array. |
| <i>MAX</i> | The maximum number of elements in the buffer. If the actual message is longer than the buffer, it will be silently truncated. |

This type is assignment compatible to [Array_ref<ELEM_TYPE, LEN_TYPE>](#) and provides a transparent server-side copy-in mechanism for array parameters. The [Array_in_buf](#) provides the storage for the array data and receives a copy of the data passed to the server-function.

Definition at line 126 of file [ipc_array](#).

The documentation for this struct was generated from the following file:

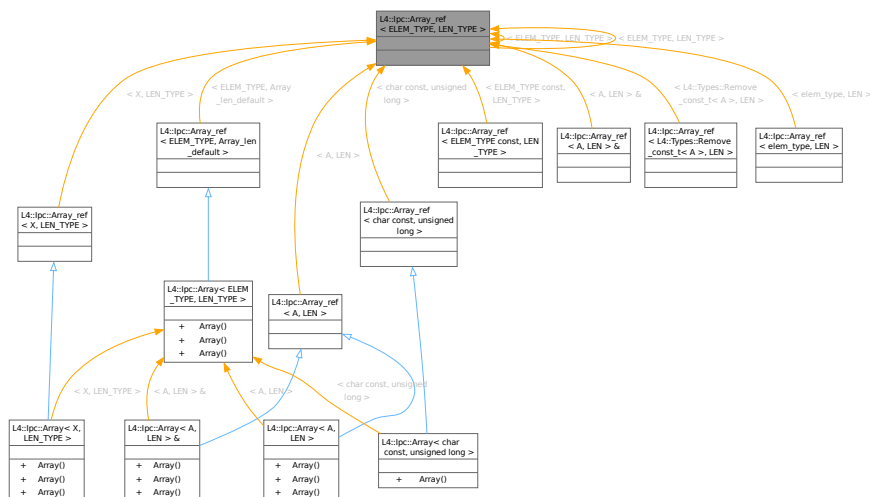
- I4/sys/cxx/ipc_array

15.104 L4::lpc::Array_ref< ELEM_TYPE, LEN_TYPE > Struct Template Reference

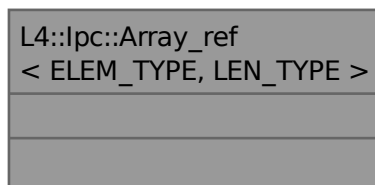
[Array](#) reference data type for arrays located in the message.

```
#include <ipc_array>
```

Inheritance diagram for L4::lpc::Array_ref< ELEM_TYPE, LEN_TYPE >:



Collaboration diagram for L4::lpc::Array_ref< ELEM_TYPE, LEN_TYPE >:



15.104.1 Detailed Description

```
template<typename ELEM_TYPE, typename LEN_TYPE = Array_len_default>
struct L4::lpc::Array_ref< ELEM_TYPE, LEN_TYPE >
```

[Array](#) reference data type for arrays located in the message.

Note

Use [Array](#) for normal RPC interfaces, [Array_ref](#) is usually used as server-side argument, see [Array](#).

Template Parameters

| | |
|------------------|--|
| <i>ELEM_TYPE</i> | The data type of an array element, should be 'const' when used as input. |
| <i>LEN_TYPE</i> | Data type used to store the number of elements in the array. |

Definition at line 28 of file [ipc_array](#).

The documentation for this struct was generated from the following file:

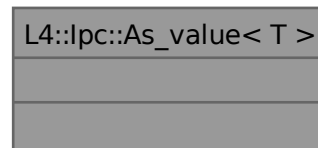
- l4/sys/cxx/ipc_array

15.105 L4::lpc::As_value< T > Struct Template Reference

Pass the argument as plain data value.

```
#include <ipc_types>
```

Collaboration diagram for L4::lpc::As_value< T >:



15.105.1 Detailed Description

```
template<typename T>
struct L4::lpc::As_value< T >
```

Pass the argument as plain data value.

Template Parameters

| | |
|----------|------------------------------------|
| <i>T</i> | The type of the original argument. |
|----------|------------------------------------|

[As_value<T>](#) is used when *T* would be otherwise interpreted specially, for example as flexpage. When using [As_value<>](#) then the argument is transmitted as plain data element.

Definition at line 116 of file [ipc_types](#).

The documentation for this struct was generated from the following file:

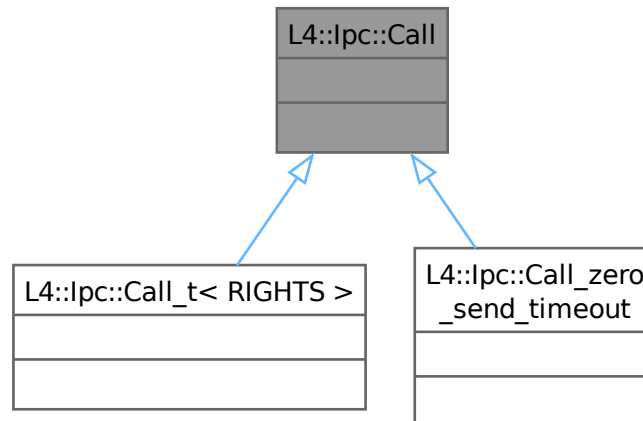
- l4/sys/cxx/ipc_types

15.106 L4::ipc::Call Struct Reference

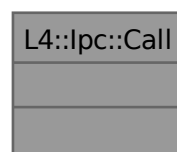
RPC attribute for a standard RPC call.

```
#include <ipc_iface>
```

Inheritance diagram for L4::ipc::Call:



Collaboration diagram for L4::ipc::Call:



15.106.1 Detailed Description

RPC attribute for a standard RPC call.

This is the default for the *FLAGS* parameter for L4::ipc::Msg::Rpc_call L4::ipc::Msg::Rpc_inline_call templates and declares the RPC to have default call semantics and timeouts.

Examples:

```
L4_RPC(l4_ret_t, send, (unsigned value), L4::Ipc::Call);
```


which is equivalent to:

```
L4_RPC(l4_ret_t, send, (unsigned value));
```

Definition at line 239 of file [ipc_iface](#).

The documentation for this struct was generated from the following file:

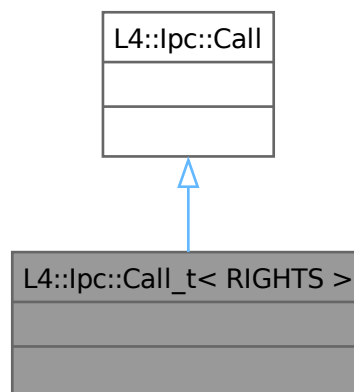
- [l4/sys/cxx/ipc_iface](#)

15.107 L4::lpc::Call_t< RIGHTS > Struct Template Reference

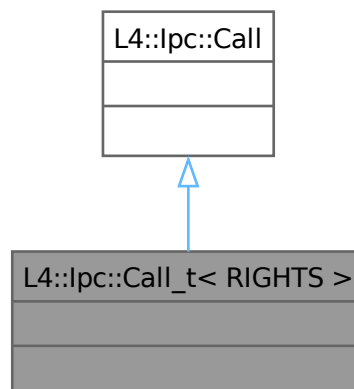
RPC attribute for an RPC call with required rights.

```
#include <ipc_iface>
```

Inheritance diagram for L4::lpc::Call_t< RIGHTS >:



Collaboration diagram for L4::lpc::Call_t< RIGHTS >:



15.107.1 Detailed Description

```
template<unsigned RIGHTS>
struct L4::lpc::Call_t< RIGHTS >
```

RPC attribute for an RPC call with required rights.

Template Parameters

| | |
|---------------|---|
| <i>RIGHTS</i> | The capability rights required for this call. L4_CAP_FPAGE_W and L4_CAP_FPAGE_S are checked within the server (and -L4_EPERM shall be returned if the caller has insufficient rights). L4_CAP_FPAGE_R is always on but might be specified for documentation purposes. Other rights cannot be used in this context, because they cannot be checked at the server side. |
|---------------|---|

Examples:

```
L4_RPC(l4_ret_t, func, (unsigned value), L4::lpc::Call_t<L4_CAP_FPAGE_RW>);
```

Definition at line 270 of file [ipc_iface](#).

The documentation for this struct was generated from the following file:

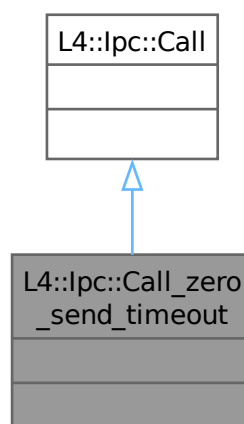
- [l4/sys/cxx/ipc_iface](#)

15.108 L4::lpc::Call_zero_send_timeout Struct Reference

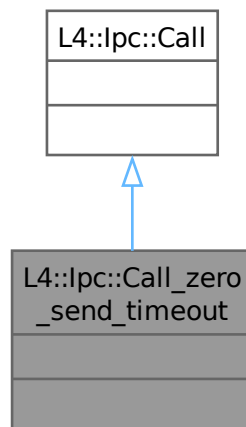
RPC attribute for an RPC call, with zero send timeout.

```
#include <ipc_iface>
```

Inheritance diagram for L4::lpc::Call_zero_send_timeout:



Collaboration diagram for L4::lpc::Call_zero_send_timeout:



15.108.1 Detailed Description

RPC attribute for an RPC call, with zero send timeout.

Definition at line 249 of file [ipc_iface](#).

The documentation for this struct was generated from the following file:

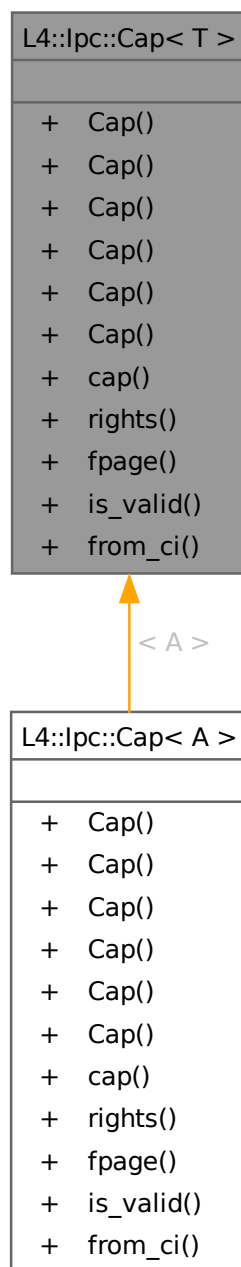
- [l4/sys/cxx/ipc_iface](#)

15.109 L4::lpc::Cap< T > Class Template Reference

Capability type for RPC interfaces (see [L4::Cap<T>](#)).

```
#include <ipc_types>
```

Inheritance diagram for L4::lpc::Cap< T >:



Collaboration diagram for L4::lpc::Cap< T >:

| L4::lpc::Cap< T > |
|---|
| <ul style="list-style-type: none"> + Cap() + Cap() + Cap() + Cap() + Cap() + Cap() + cap() + rights() + fpage() + is_valid() + from_ci() |

Public Types

- enum { [Rights_mask](#) = 0xff , [Map_type_mask](#) = 0x100 , [Cap_mask](#) = L4_CAP_MASK }
- enum [Map_type](#) { [Map](#) = 0 , [Grant](#) = 0x100 }

Kind of mapping.

Public Member Functions

- template<typename O>
Cap (Cap< O > const &o) noexcept
Make copy with conversion.
- **Cap** (L4::Cap< T > [cap](#)) noexcept
Make a [Cap](#) from [L4::Cap<T>](#), with minimal rights.
- template<typename O>
Cap (L4::Cap< O > [cap](#)) noexcept
Make IPC [Cap](#) from [L4::Cap](#) with conversion (and minimal rights).
- **Cap** () noexcept
Make an invalid cap.
- **Cap** (L4::Cap< T > [cap](#), unsigned char [rights](#)) noexcept
Make a [Cap](#) from [L4::Cap<T>](#) with the given rights.
- **Cap** (L4::Cap< T > [cap](#), unsigned char [rights](#), [Map_type](#) map_type) noexcept
Make a [Cap](#) from [L4::Cap<T>](#) with the given rights and map type.
- L4::Cap< T > **cap** () const noexcept
Return the [L4::Cap<T>](#) of this [Cap](#).

- unsigned **rights** () const noexcept
Return the rights bits stored in this IPC cap.
- [L4::lpc::Snd_fpage fpage](#) () const noexcept
Return the send flexpage for this [Cap](#) (see [l4_fpage_t](#)).
- bool **is_valid** () const noexcept
Return true if this [Cap](#) is valid.

Static Public Member Functions

- static Cap [from_ci](#) ([l4_cap_idx_t](#) c) noexcept
Create an IPC capability from a C capability index plus rights.

15.109.1 Detailed Description

```
template<typename T>
class L4::lpc::Cap< T >
```

Capability type for RPC interfaces (see [L4 : :Cap<T>](#)).

Template Parameters

| | |
|----------|---|
| <i>T</i> | type of the interface referenced by the capability. |
|----------|---|

In contrast to [L4 : :Cap<T>](#) this type additionally stores a rights mask that shall be used when the capability is transferred to the receiver. This allows to apply restrictions to the transferred capability in the form of a subset of the rights possessed by the sender.

See also

[L4::lpc::make_cap\(\)](#)

Definition at line 698 of file [ipc_types](#).

15.109.2 Member Enumeration Documentation

15.109.2.1 anonymous enum

```
template<typename T>
anonymous enum
```

Enumerator

| | |
|---------------|---|
| Rights_mask | Mask for rights bits stored internally. L4_FPAGE_RIGHTS_MASK L4_FPAGE_C_NO_REF_CNT L4_FPAGE_C_OBJ_RIGHTS). |
| Map_type_mask | Mask for map vs. grant operation. |

| | |
|----------|---|
| Cap_mask | Mask for significant capability bits. (incl. the invalid bit to support invalid caps) |
|----------|---|

Definition at line 704 of file [ipc_types](#).

15.109.2.2 Map_type

```
template<typename T>
enum L4::Ipc::Cap::Map_type
```

Kind of mapping.

Enumerator

| | |
|-------|---|
| Map | Map capability in RPC operation. |
| Grant | Grant capability in RPC operation. See also L4_MAP_ITEM_GRANT |

Definition at line 726 of file [ipc_types](#).

15.109.3 Constructor & Destructor Documentation

15.109.3.1 Cap() [1/2]

```
template<typename T>
L4::Ipc::Cap< T >::Cap (
    L4::Cap< T > cap,
    unsigned char rights) [inline], [noexcept]
```

Make a [Cap](#) from [L4::Cap<T>](#) with the given rights.

Parameters

| | |
|---------------|---|
| <i>cap</i> | Capability to be sent. |
| <i>rights</i> | Rights to be sent. Consists of L4_fpage_rights and Attributes and additional permissions for object send items. |

Definition at line 764 of file [ipc_types](#).

15.109.3.2 Cap() [2/2]

```
template<typename T>
L4::Ipc::Cap< T >::Cap (
    L4::Cap< T > cap,
    unsigned char rights,
    Map_type map_type) [inline], [noexcept]
```

Make a [Cap](#) from [L4::Cap<T>](#) with the given rights and map type.

Parameters

| | |
|-----------------|--|
| <i>cap</i> | Capability to be sent. |
| <i>rights</i> | Rights to be sent. Consists of L4_fpage_rights and Attributes and additional permissions for object send items . |
| <i>map_type</i> | Indicate if capability is mapped or granted. |

Definition at line [775](#) of file [ipc_types](#).

15.109.4 Member Function Documentation

15.109.4.1 from_ci()

```
template<typename T>
Cap L4::Ipc::Cap< T >::from_ci (
    l4_cap_idx_t c) [inline], [static], [noexcept]
```

Create an IPC capability from a C capability index plus rights.

Parameters

| | |
|----------|--|
| <i>c</i> | C capability index with the lowest 8 bits used as rights for the map operation (see L4_fpage_rights). |
|----------|--|

Definition at line [785](#) of file [ipc_types](#).

The documentation for this class was generated from the following file:

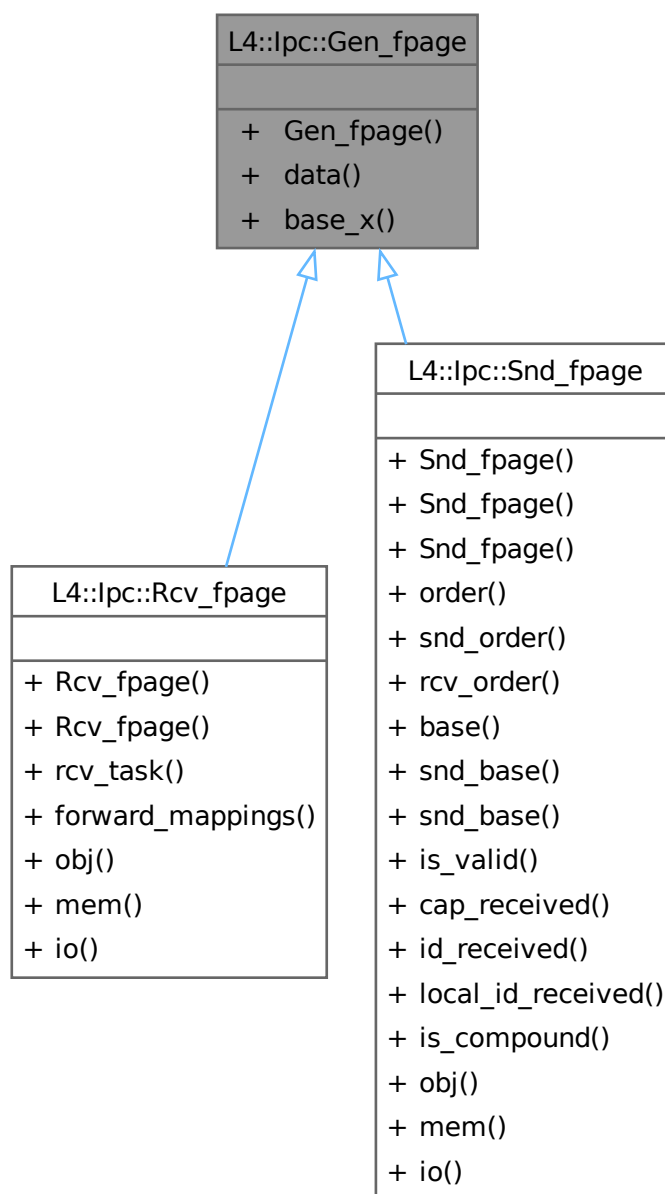
- [l4/sys/cxx/ipc_types](#)

15.110 L4::Ipc::Gen_fpage Class Reference

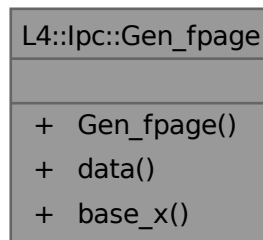
Generic RPC base for typed message items.

```
#include <ipc_types>
```


Inheritance diagram for L4::lpc::Gen_fpage:



Collaboration diagram for L4::Ipc::Gen_fpage:



Public Types

- enum [Type](#) { [Special](#) = L4_FPAGE_SPECIAL << 4 , [Memory](#) = L4_FPAGE_MEMORY << 4 , [Io](#) = L4_FPAGE_IO << 4 , [Obj](#) = L4_FPAGE_OBJ << 4 }
- Type of mapping object, see [L4_fpage_type](#).*

Public Member Functions

- **Gen_fpage** ([l4_umword_t](#) base, [l4_umword_t](#) data) noexcept
Construct from raw values.
- [l4_umword_t](#) **data** () const noexcept
Return the raw flexpage descriptor.
- [l4_umword_t](#) **base_x** () const noexcept
Return the raw base descriptor.

15.110.1 Detailed Description

Generic RPC base for typed message items.

Definition at line 286 of file [ipc_types](#).

15.110.2 Member Enumeration Documentation

15.110.2.1 Type

enum [L4::Ipc::Gen_fpage::Type](#)

[Type](#) of mapping object, see [L4_fpage_type](#).

Enumerator

| | |
|---------|--|
| Special | Special flexpage, either l4_fpage_invalid() or l4_fpage_all() ; only supported by selected interfaces. |
| Memory | Flexpage for memory spaces. |
| Io | Flexpage for I/O port spaces. |
| Obj | Flexpage for object spaces. |

Definition at line 290 of file [ipc_types](#).

The documentation for this class was generated from the following file:

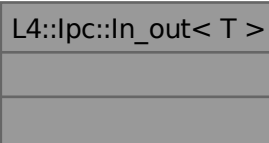
- [l4/sys/cxx/ipc_types](#)

15.111 L4::lpc::ln_out< T > Struct Template Reference

Mark an argument as in-out argument.

```
#include <ipc_types>
```

Collaboration diagram for L4::lpc::ln_out< T >:

**15.111.1 Detailed Description**

```
template<typename T>
struct L4::lpc::ln_out< T >
```

Mark an argument as in-out argument.

Template Parameters

| | |
|----------|---|
| <i>T</i> | The original argument type, usually a pointer or a reference. |
|----------|---|

[ln_out<>](#) is used when an otherwise output-only value shall also be used as input value.

Definition at line 41 of file [ipc_types](#).

The documentation for this struct was generated from the following file:

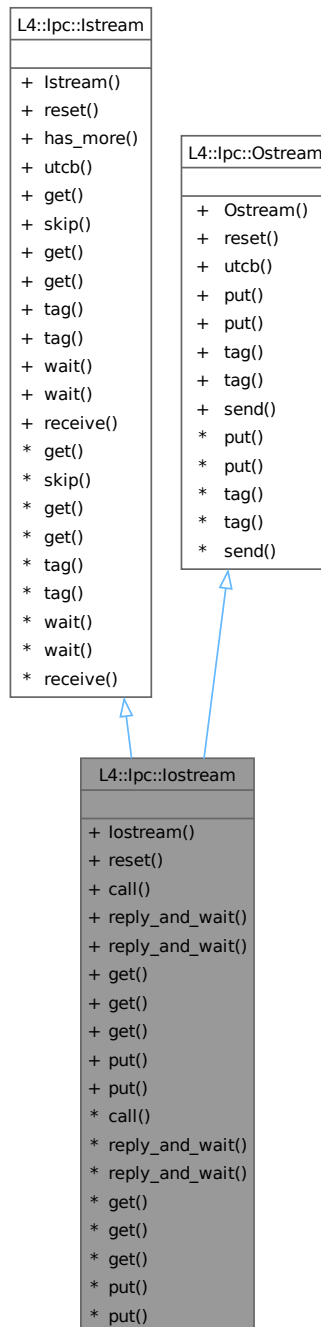
- [l4/sys/cxx/ipc_types](#)

15.112 L4::lpc::lostream Class Reference

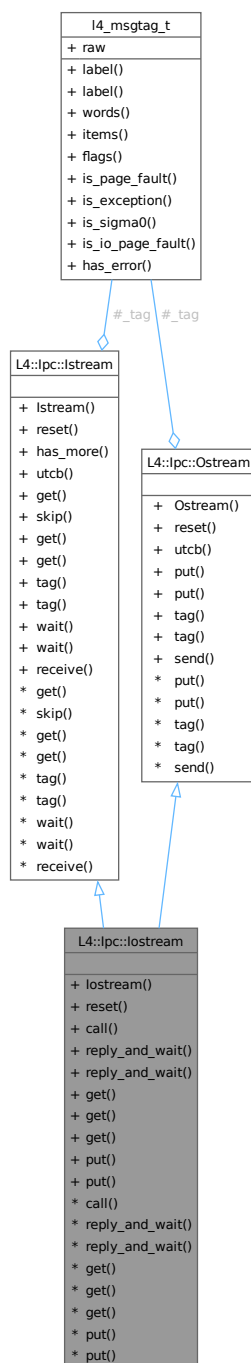
Input/Output stream for IPC [un]marshalling.

```
#include <ipc_stream>
```

Inheritance diagram for L4::lpc::lostream:



Collaboration diagram for L4::lpc::lostream:



Public Member Functions

- `lostream (l4_utcb_t *utcb)`
Create an IPC IO stream with a single message buffer.
- `void reset ()`
Reset the stream to its initial state.

IPC operations.

- `l4_msgtag_t call (l4_cap_idx_t dst, l4_timeout_t timeout, long proto=0)`
Do an IPC call using the message in the output stream and receive the reply in the input stream.
- `l4_msgtag_t reply_and_wait (l4_umword_t *src_dst, long proto=0)`
Do an IPC reply and wait.
- `l4_msgtag_t reply_and_wait (l4_umword_t *src_dst, l4_timeout_t timeout, long proto=0)`
Do an IPC reply and wait.

Get/Put functions.

These functions are basically used to implement the insertion operators (<<) and should not be called directly.

- `template<typename T>`
`unsigned long get (T *buf, unsigned long elems)`
Copy out an array of type T with size elements.
- `template<typename T>`
`unsigned long get (Msg_ptr< T > const &buf, unsigned long elems=1)`
Read one size elements of type T from the stream and return a pointer.
- `template<typename T>`
`bool get (T &v)`
Extract a single element of type T from the stream.
- `template<typename T>`
`bool put (T *buf, unsigned long size)`
Put an array with size elements of type T into the stream.
- `template<typename T>`
`bool put (T const &v)`
Insert an element of type T into the stream.

Public Member Functions inherited from `L4::lpc::lstream`

- `lstream (l4_utcb_t *utcb)`
Create an input stream for the given message buffer.
- `void reset ()`
Reset the stream to empty, and ready for `receive()/wait()`.
- `template<typename T>`
`bool has_more (unsigned long count=1)`
Check whether a value of type T can be obtained from the stream.
- `l4_utcb_t * utcb () const`
Return utcb pointer.
- `template<typename T>`
`unsigned long get (T *buf, unsigned long elems)`
Copy out an array of type T with size elements.
- `template<typename T>`
`void skip (unsigned long elems)`
Skip size elements of type T in the stream.
- `template<typename T>`
`unsigned long get (Msg_ptr< T > const &buf, unsigned long elems=1)`
Read one size elements of type T from the stream and return a pointer.
- `template<typename T>`
`bool get (T &v)`
Extract a single element of type T from the stream.
- `l4_msgtag_t tag () const`

- Get the message tag of a received IPC.
 • [l4_msgtag_t & tag \(\)](#)
 Get the message tag of a received IPC.
- [l4_msgtag_t wait \(l4_umword_t *src\)](#)
 Wait for an incoming message from any sender.
- [l4_msgtag_t wait \(l4_umword_t *src, l4_timeout_t timeout\)](#)
 Wait for an incoming message from any sender.
- [l4_msgtag_t receive \(l4_cap_idx_t src\)](#)
 Wait for a message from the specified sender.

Public Member Functions inherited from L4::lpc::Ostream

- **Ostream** ([l4_utcb_t *utcb](#))
 Create an IPC output stream using the given message buffer [utcb](#).
- void **reset** ()
 Reset the stream to empty, same state as a newly created stream.
- [l4_utcb_t * utcb](#) () const
 Return utcb pointer.
- template<typename T>
 bool **put** (T *buf, unsigned long size)
 Put an array with *size* elements of type *T* into the stream.
- template<typename T>
 bool **put** (T const &v)
 Insert an element of type *T* into the stream.
- [l4_msgtag_t tag](#) () const
 Extract the [L4](#) message tag from the stream.
- [l4_msgtag_t & tag](#) ()
 Extract a reference to the [L4](#) message tag from the stream.
- [l4_msgtag_t send](#) ([l4_cap_idx_t](#) dst, long proto=0, unsigned flags=0)
 Send the message via IPC to the given receiver.

15.112.1 Detailed Description

Input/Output stream for IPC [un]marshalling.

The [lpc::lostream](#) is part of the AW Env IPC framework as well as [lpc::lstream](#) and [lpc::Ostream](#). In particular an [lpc::lostream](#) is a combination of an [lpc::lstream](#) and an [lpc::Ostream](#). It can use either a single message buffer for receiving and sending messages or a pair of a receive and a send buffer. The stream also supports combined IPC operations such as [call\(\)](#) and [reply_and_wait\(\)](#), which can be used to implement RPC functionality.

Examples

[examples/libs/l4re/c++/shared_ds/ds_srv.cc](#), [examples/libs/l4re/streammap/client.cc](#), and [examples/libs/l4re/streammap/server.cc](#).

Definition at line 789 of file [ipc_stream](#).

15.112.2 Constructor & Destructor Documentation

15.112.2.1 `lostream()`

```
L4::Ipc::Iostream::Iostream (
    l4_utcb_t * utcb) [inline], [explicit]
```

Create an IPC IO stream with a single message buffer.

Parameters

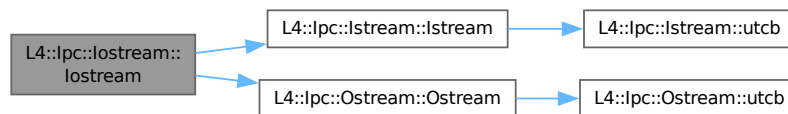
| | |
|-------------|---|
| <i>utcb</i> | The message buffer used as backing store. |
|-------------|---|

The created IO stream uses the same message buffer for sending and receiving IPC messages.

Definition at line 801 of file [ipc_stream](#).

References [L4::Ipc::Istream::Istream\(\)](#), and [L4::Ipc::Ostream::Ostream\(\)](#).

Here is the call graph for this function:



15.112.3 Member Function Documentation

15.112.3.1 `call()`

```
l4_msgtag_t L4::Ipc::Iostream::call (
    l4_cap_idx_t dst,
    l4_timeout_t timeout,
    long proto = 0) [inline]
```

Do an IPC call using the message in the output stream and receive the reply in the input stream.

Parameters

| | |
|----------------|---|
| <i>dst</i> | The destination to call. |
| <i>timeout</i> | The IPC timeout for the call. |
| <i>proto</i> | The protocol value to use in the message tag. |

Returns

The result tag of the IPC operation.

This is a combined IPC operation consisting of a send and a receive to/from the given destination `dst`.

A call is usually used by clients for RPCs to a server.

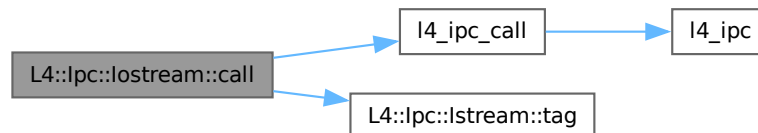
Examples

[examples/libs/l4re/streammap/client.cc](#).

Definition at line 966 of file [ipc_stream](#).

References [l4_ipc_call\(\)](#), and [L4::lpc::lstream::tag\(\)](#).

Here is the call graph for this function:

**15.112.3.2 get() [1/3]**

```

template<typename T>
unsigned long L4::lpc::lstream::get (
    Msg_ptr< T > const & buf,
    unsigned long elems = 1) [inline]
  
```

Read one size elements of type T from the stream and return a pointer.

Parameters

| | |
|--------------|---|
| <i>buf</i> | A Msg_ptr that is actually set to point to the element in the stream. |
| <i>elems</i> | Number of elements to extract (default is 1). |

Returns

The number of elements extracted.

In contrast to a normal `get`, this version does actually not copy the data but returns a pointer to the data.

See [operator>>\(\)](#)

Definition at line 439 of file [ipc_stream](#).

15.112.3.3 `get()` [2/3]

```
template<typename T>
bool L4::Ipc::Istream::get (
    T & v) [inline]
```

Extract a single element of type T from the stream.

Parameters

| | | |
|------------|----------|--------------|
| <i>out</i> | <i>v</i> | The element. |
|------------|----------|--------------|

Return values

| | |
|--------------|--|
| <i>true</i> | An element was successfully extracted. |
| <i>false</i> | An element could not be extracted. |

See [operator>>\(\)](#)

Definition at line 464 of file [ipc_stream](#).

15.112.3.4 `get()` [3/3]

```
template<typename T>
unsigned long L4::Ipc::Istream::get (
    T * buf,
    unsigned long elems) [inline]
```

Copy out an array of type T with `size` elements.

Parameters

| | |
|--------------|--|
| <i>buf</i> | Pointer to a buffer for size elements of type T. |
| <i>elems</i> | Number of elements of type T to copy out. |

Returns

The number of elements copied out.

See [operator>>\(\)](#)

Definition at line 394 of file [ipc_stream](#).

15.112.3.5 put() [1/2]

```
template<typename T>
bool L4::Ipc::Ostream::put (
    T * buf,
    unsigned long size) [inline]
```

Put an array with *size* elements of type *T* into the stream.

Parameters

| | |
|-------------|---|
| <i>buf</i> | A pointer to the array to insert into the buffer. |
| <i>size</i> | The number of elements in the array. |

Definition at line 660 of file [ipc_stream](#).

15.112.3.6 put() [2/2]

```
template<typename T>
bool L4::Ipc::Ostream::put (
    T const & v) [inline]
```

Insert an element of type *T* into the stream.

Parameters

| | |
|----------|------------------------|
| <i>v</i> | The element to insert. |
|----------|------------------------|

Definition at line 678 of file [ipc_stream](#).

15.112.3.7 reply_and_wait() [1/2]

```
l4_msgtag_t L4::Ipc::Iostream::reply_and_wait (
    l4_umword_t * src_dst,
    l4_timeout_t timeout,
    long proto = 0) [inline]
```

Do an IPC reply and wait.

Parameters

| | | |
|----------------|----------------|---|
| <i>in, out</i> | <i>src_dst</i> | Input: the destination for the send operation. Output: the source of the received message. |
| | <i>timeout</i> | Timeout used for IPC. |
| | <i>proto</i> | Protocol to use. |

Returns

The result tag of the IPC operation.

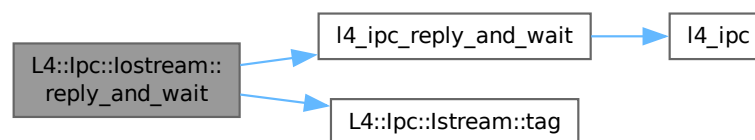
This is a combined IPC operation consisting of a send operation and an open wait for any message.

A reply and wait is usually used by servers that reply to a client and wait for the next request by any other client.

Definition at line 981 of file [ipc_stream](#).

References [l4_ipc_reply_and_wait\(\)](#), and [L4::Ipc::Istream::tag\(\)](#).

Here is the call graph for this function:

**15.112.3.8 reply_and_wait() [2/2]**

```

l4_msgtag_t L4::Ipc::Istream::reply_and_wait (
    l4_umword_t * src_dst,
    long proto = 0) [inline]
  
```

Do an IPC reply and wait.

Parameters

| | | |
|----------------------|----------------------|---|
| <code>in, out</code> | <code>src_dst</code> | Input: the destination for the send operation. Output: the source of the received message. |
| | <code>proto</code> | Protocol to use. |

Returns

The result tag of the IPC operation.

This is a combined IPC operation consisting of a send operation and an open wait for any message.

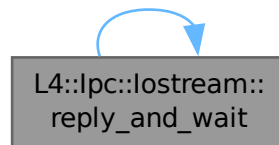
A reply and wait is usually used by servers that reply to a client and wait for the next request by any other client.

Definition at line 874 of file [ipc_stream](#).

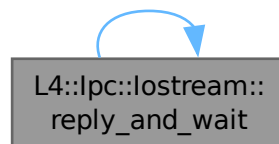
References [L4_IPC_SEND_TIMEOUT_0](#), and [reply_and_wait\(\)](#).

Referenced by [reply_and_wait\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.112.3.9 reset()

```
void L4::Ipc::Iostream::reset () [inline]
```

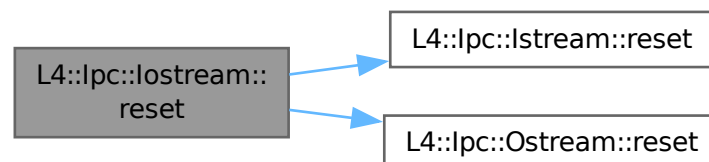
Reset the stream to its initial state.

Input as well as the output stream are reset.

Definition at line 815 of file [ipc_stream](#).

References [L4::lpc::Istream::reset\(\)](#), and [L4::lpc::Ostream::reset\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

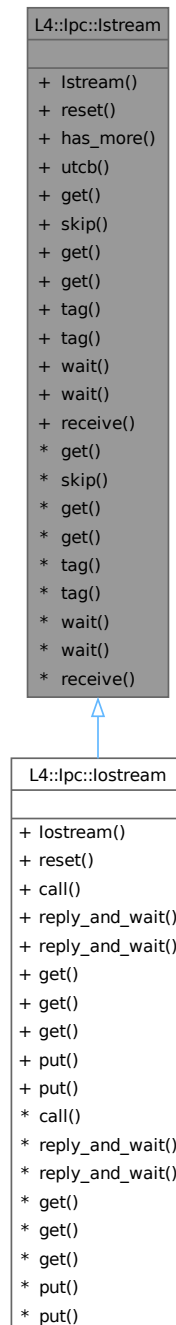
- [l4/cxx/ipc_stream](#)

15.113 L4::lpc::Istream Class Reference

Input stream for IPC unmarshalling.

```
#include <ipc_stream>
```

Inheritance diagram for L4::lpc::Istream:



Collaboration diagram for L4::lpc::Istream:



Public Member Functions

- `Istream (l4_utcb_t *utcb)`
Create an input stream for the given message buffer.
- `void reset ()`
Reset the stream to empty, and ready for `receive()/wait()`.

- `template<typename T>`
`bool has_more (unsigned long count=1)`
Check whether a value of type T can be obtained from the stream.
- `l4_utcb_t * utcb () const`
Return utcb pointer.

Get/Put Functions.

- `template<typename T>`
`unsigned long get (T *buf, unsigned long elems)`
Copy out an array of type T with size elements.
- `template<typename T>`
`void skip (unsigned long elems)`
Skip size elements of type T in the stream.
- `template<typename T>`
`unsigned long get (Msg_ptr< T > const &buf, unsigned long elems=1)`
Read one size elements of type T from the stream and return a pointer.
- `template<typename T>`
`bool get (T &v)`
Extract a single element of type T from the stream.
- `l4_msgtag_t tag () const`
Get the message tag of a received IPC.
- `l4_msgtag_t & tag ()`
Get the message tag of a received IPC.

IPC operations.

- `l4_msgtag_t wait (l4_umword_t *src)`
Wait for an incoming message from any sender.
- `l4_msgtag_t wait (l4_umword_t *src, l4_timeout_t timeout)`
Wait for an incoming message from any sender.
- `l4_msgtag_t receive (l4_cap_idx_t src)`
Wait for a message from the specified sender.

15.113.1 Detailed Description

Input stream for IPC unmarshalling.

`lpc::lstream` is part of the dynamic IPC marshalling infrastructure, as well as `lpc::Ostream` and `lpc::lostream`.

`lpc::lstream` is an input stream supporting extraction of values from an IPC message buffer. A received IPC message can be unmarshalled using the usual extraction operator (>>).

There exist some special wrapper classes to extract arrays (see `lpc_buf_cp_in` and `lpc_buf_in`) and indirect strings (see `Msg_in_buffer` and `Msg_io_buffer`).

Definition at line 334 of file `ipc_stream`.

15.113.2 Constructor & Destructor Documentation

15.113.2.1 Istream()

```
L4::Ipc::Istream::Istream (
    l4_utcb_t * utcb) [inline]
```

Create an input stream for the given message buffer.

The given message buffer is used for IPC operations [wait\(\)](#)/[receive\(\)](#) and received data can be extracted using the `>>` operator afterwards. In the case of indirect message parts a buffer of type `Msg_in_buffer` must be inserted into the stream before the IPC operation and contains received data afterwards.

Parameters

| | |
|-------------|---|
| <i>utcb</i> | The message buffer to receive IPC messages. |
|-------------|---|

Definition at line [348](#) of file [ipc_stream](#).

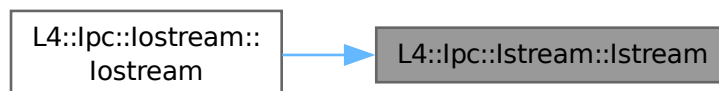
References [utcb\(\)](#).

Referenced by [L4::lpc::lostream::lostream\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.113.3 Member Function Documentation

15.113.3.1 `get()` [1/3]

```
template<typename T>
unsigned long L4::Ipc::Istream::get (
    Msg_ptr< T > const & buf,
    unsigned long elems = 1) [inline]
```

Read one size elements of type T from the stream and return a pointer.

Parameters

| | |
|--------------|---|
| <i>buf</i> | A Msg_ptr that is actually set to point to the element in the stream. |
| <i>elems</i> | Number of elements to extract (default is 1). |

Returns

The number of elements extracted.

In contrast to a normal get, this version does actually not copy the data but returns a pointer to the data.

See [operator>>\(\)](#)

Definition at line 439 of file [ipc_stream](#).

References [has_more\(\)](#), and [L4_UNLIKELY](#).

Here is the call graph for this function:



15.113.3.2 `get()` [2/3]

```
template<typename T>
bool L4::Ipc::Istream::get (
    T & v) [inline]
```

Extract a single element of type T from the stream.

Parameters

| | | |
|-----|---|--------------|
| out | v | The element. |
|-----|---|--------------|

Return values

| | |
|--------------|--|
| <i>true</i> | An element was successfully extracted. |
| <i>false</i> | An element could not be extracted. |

See [operator>>\(\)](#)

Definition at line 464 of file [ipc_stream](#).

References [has_more\(\)](#), and [L4_UNLIKELY](#).

Here is the call graph for this function:

**15.113.3.3 get() [3/3]**

```

template<typename T>
unsigned long L4::Ipc::Istream::get (
    T * buf,
    unsigned long elems) [inline]
  
```

Copy out an array of type T with *size* elements.

Parameters

| | |
|--------------|---|
| <i>buf</i> | Pointer to a buffer for <i>size</i> elements of type T. |
| <i>elems</i> | Number of elements of type T to copy out. |

Returns

The number of elements copied out.

See [operator>>\(\)](#)

Definition at line 394 of file [ipc_stream](#).

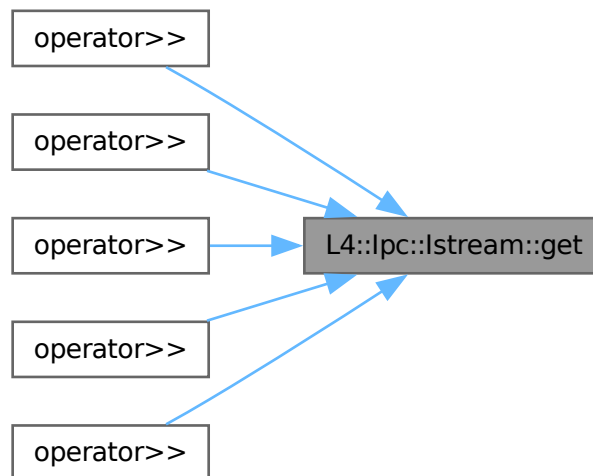
References [has_more\(\)](#), and [L4_UNLIKELY](#).

Referenced by [operator>>\(\)](#), [operator>>\(\)](#), [operator>>\(\)](#), [operator>>\(\)](#), and [operator>>\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.113.3.4 receive()

```

l4_msgtag_t L4::Ipc::Istream::receive (
    l4_cap_idx_t src) [inline]
  
```

Wait for a message from the specified sender.

Parameters

| | |
|------------|--------------------------------|
| <i>src</i> | The sender id to receive from. |
|------------|--------------------------------|

Returns

The IPC result tag ([l4_msgtag_t](#)).

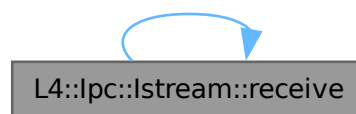
This is commonly known as 'closed wait'.

Definition at line [572](#) of file [ipc_stream](#).

References [L4_IPC_NEVER](#), and [receive\(\)](#).

Referenced by [receive\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.113.3.5 reset()

```
void L4::Ipc::Istream::reset () [inline]
```

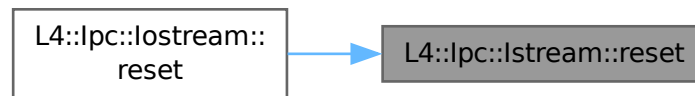
Reset the stream to empty, and ready for [receive\(\)/wait\(\)](#).

The stream is reset to the same state as on its creation.

Definition at line [358](#) of file [ipc_stream](#).

Referenced by [L4::lpc::lostream::reset\(\)](#).

Here is the caller graph for this function:



15.113.3.6 skip()

```
template<typename T>
void L4::Ipc::Istream::skip (
    unsigned long elems) [inline]
```

Skip size elements of type T in the stream.

Parameters

| | |
|--------------|-----------------------------|
| <i>elems</i> | Number of elements to skip. |
|--------------|-----------------------------|

Definition at line 414 of file [ipc_stream](#).

References [has_more\(\)](#), and [L4_UNLIKELY](#).

Here is the call graph for this function:



15.113.3.7 tag() [1/2]

```
l4\_msgtag\_t & L4::Ipc::Istream::tag () [inline]
```

Get the message tag of a received IPC.

Returns

A reference to the [L4](#) message tag for the received IPC.

This is in particular useful for handling page faults or exceptions.

See [operator>>\(\)](#)

Definition at line 517 of file [ipc_stream](#).

15.113.3.8 tag() [2/2]

```
l4_msgtag_t L4::Ipc::Istream::tag () const [inline]
```

Get the message tag of a received IPC.

Returns

The [L4](#) message tag for the received IPC.

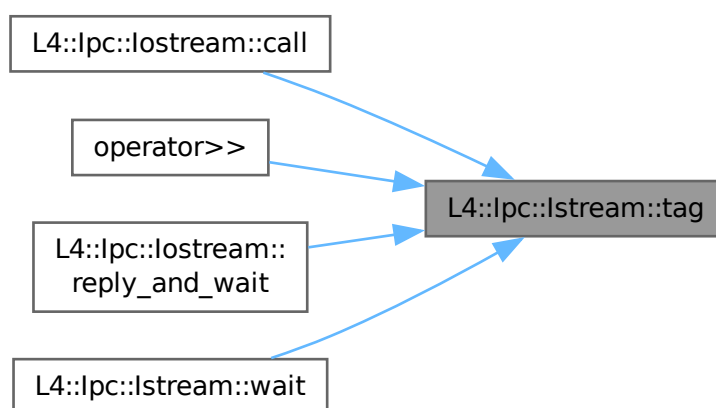
This is in particular useful for handling page faults or exceptions.

See [operator>>\(\)](#)

Definition at line [505](#) of file [ipc_stream](#).

Referenced by [L4::lpc::lostream::call\(\)](#), [operator>>\(\)](#), [L4::lpc::lostream::reply_and_wait\(\)](#), and [wait\(\)](#).

Here is the caller graph for this function:

**15.113.3.9 wait()** [1/2]

```
l4_msgtag_t L4::Ipc::Istream::wait (
    l4_umword_t * src) [inline]
```

Wait for an incoming message from any sender.

Parameters

| | | |
|-----|-----|---|
| out | src | Contains the sender after a successful IPC operation. |
|-----|-----|---|

Returns

Syscall return tag.

This wait is actually known as 'open wait'.

Definition at line 548 of file [ipc_stream](#).

References [L4_IPC_NEVER](#), and [wait\(\)](#).

Referenced by [wait\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**15.113.3.10 wait() [2/2]**

```
l4_msgtag_t L4::Ipc::Istream::wait (  
    l4_unword_t * src,  
    l4_timeout_t timeout) [inline]
```

Wait for an incoming message from any sender.

Parameters

| | | |
|-----|---------|---|
| out | src | Contains the sender after a successful IPC operation. |
| | timeout | Timeout used for IPC. |

Returns

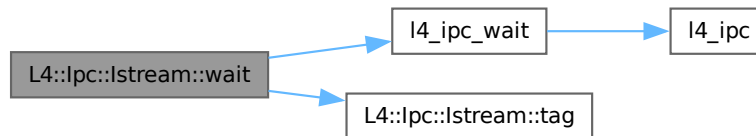
The IPC result tag ([l4_msgtag_t](#)).

This wait is actually known as 'open wait'.

Definition at line 1013 of file [ipc_stream](#).

References [l4_ipc_wait\(\)](#), and [tag\(\)](#).

Here is the call graph for this function:



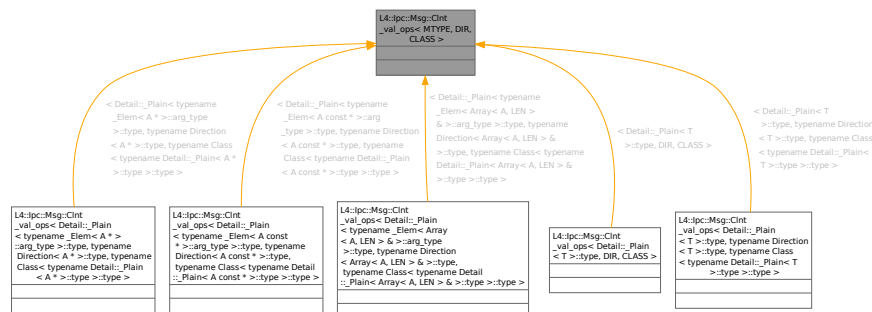
The documentation for this class was generated from the following file:

- [l4/cxx/ipc_stream](#)

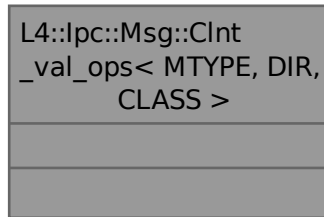
15.114 L4::lpc::Msg::Cnt_val_ops< MTYPE, DIR, CLASS > Struct Template Reference

Defines client-side handling of 'MTYPE' as RPC argument.

Inheritance diagram for L4::lpc::Msg::Cnt_val_ops< MTYPE, DIR, CLASS >:



Collaboration diagram for L4::lpc::Msg::Cnt_val_ops< MTYPE, DIR, CLASS >:



15.114.1 Detailed Description

```
template<typename MTYPE, typename DIR, typename CLASS>
struct L4::lpc::Msg::Cnt_val_ops< MTYPE, DIR, CLASS >
```

Defines client-side handling of 'MTYPE' as RPC argument.

Template Parameters

| | |
|--------------|--|
| <i>MTYPE</i> | Elem<T>::arg_type (where T is the type used in the RPC definition) |
| <i>DIR</i> | Dir_in (client -> server), or Dir_out (server -> client) |
| <i>CLASS</i> | Cls_data , Cls_item , or Cls_buffer |

Definition at line 210 of file [ipc_basics](#).

The documentation for this struct was generated from the following file:

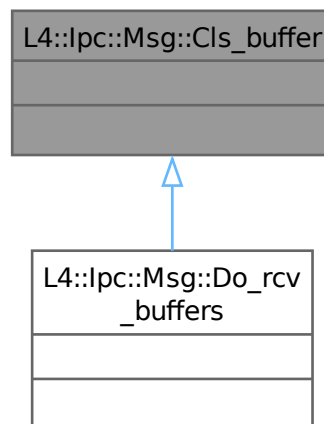
- `l4/sys/cxx/ipc_basics`

15.115 L4::lpc::Msg::Cls_buffer Struct Reference

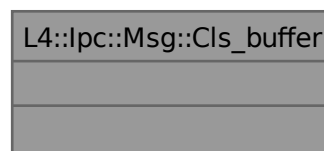
Marker type for receive buffer values.

```
#include <ipc_basics>
```

Inheritance diagram for L4::lpc::Msg::Cls_buffer:



Collaboration diagram for L4::lpc::Msg::Cls_buffer:



15.115.1 Detailed Description

Marker type for receive buffer values.

Definition at line 154 of file [ipc_basics](#).

The documentation for this struct was generated from the following file:

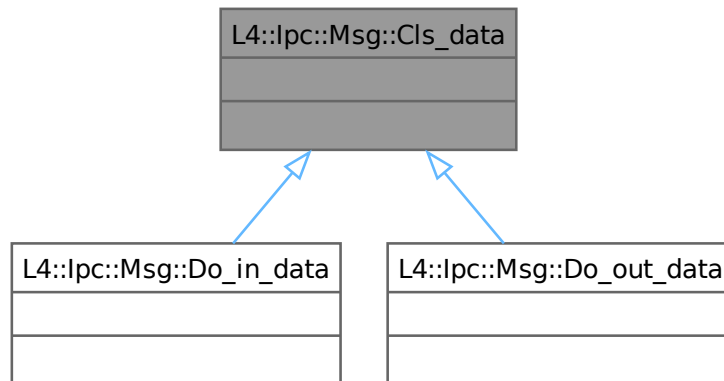
- `l4/sys/cxx/ipc_basics`

15.116 L4::lpc::Msg::Cls_data Struct Reference

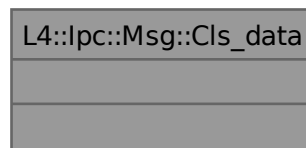
Marker type for data values.

```
#include <ipc_basics>
```

Inheritance diagram for L4::lpc::Msg::Cls_data:



Collaboration diagram for L4::lpc::Msg::Cls_data:



15.116.1 Detailed Description

Marker type for data values.

Definition at line 150 of file [ipc_basics](#).

The documentation for this struct was generated from the following file:

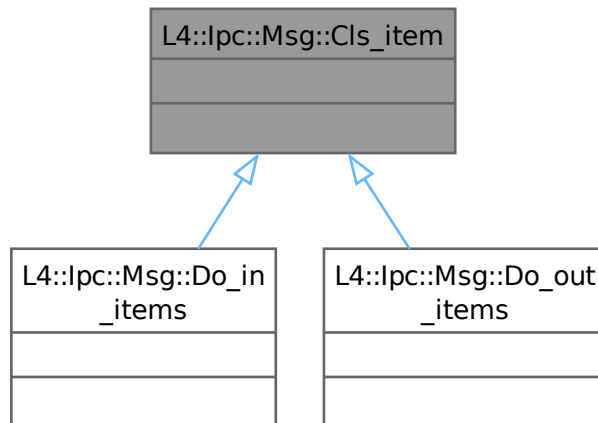
- l4/sys/cxx/ipc_basics

15.117 L4::lpc::Msg::Cls_item Struct Reference

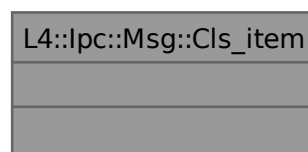
Marker type for item values.

```
#include <ipc_basics>
```

Inheritance diagram for L4::lpc::Msg::Cls_item:



Collaboration diagram for L4::lpc::Msg::Cls_item:



15.117.1 Detailed Description

Marker type for item values.

Definition at line 152 of file [ipc_basics](#).

The documentation for this struct was generated from the following file:

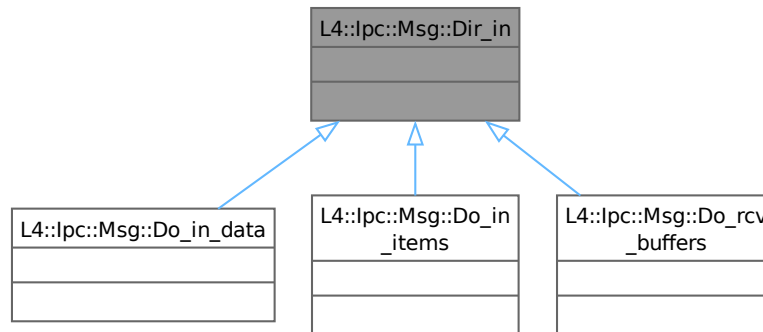
- `l4/sys/cxx/ipc_basics`

15.118 L4::lpc::Msg::Dir_in Struct Reference

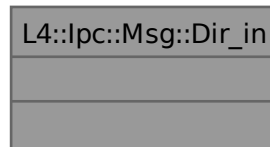
Marker type for input values.

```
#include <ipc_basics>
```

Inheritance diagram for L4::lpc::Msg::Dir_in:



Collaboration diagram for L4::lpc::Msg::Dir_in:



15.118.1 Detailed Description

Marker type for input values.

Definition at line [145](#) of file [ipc_basics](#).

The documentation for this struct was generated from the following file:

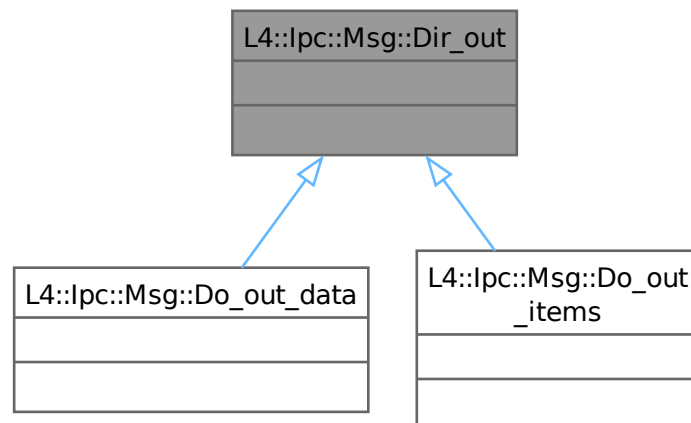
- `l4/sys/cxx/ipc_basics`

15.119 L4::lpc::Msg::Dir_out Struct Reference

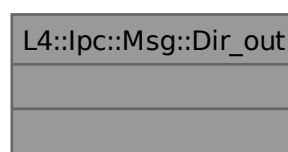
Marker type for output values.

```
#include <ipc_basics>
```

Inheritance diagram for L4::lpc::Msg::Dir_out:



Collaboration diagram for L4::lpc::Msg::Dir_out:



15.119.1 Detailed Description

Marker type for output values.

Definition at line 147 of file [ipc_basics](#).

The documentation for this struct was generated from the following file:

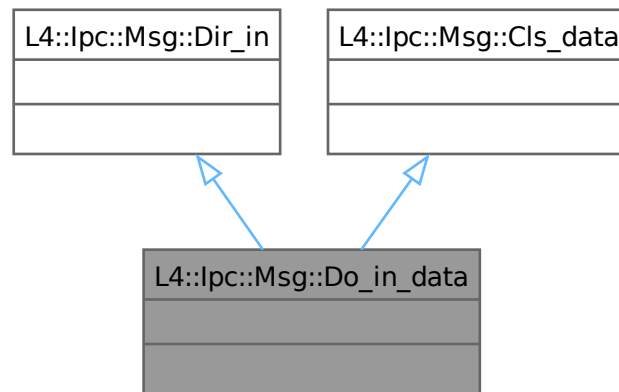
- `l4/sys/cxx/ipc_basics`

15.120 L4::lpc::Msg::Do_in_data Struct Reference

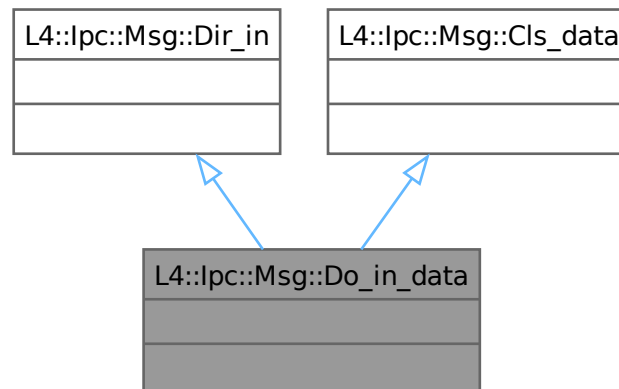
Marker for Input data.

```
#include <ipc_basics>
```

Inheritance diagram for L4::lpc::Msg::Do_in_data:



Collaboration diagram for L4::lpc::Msg::Do_in_data:



15.120.1 Detailed Description

Marker for Input data.

Definition at line 158 of file [ipc_basics](#).

The documentation for this struct was generated from the following file:

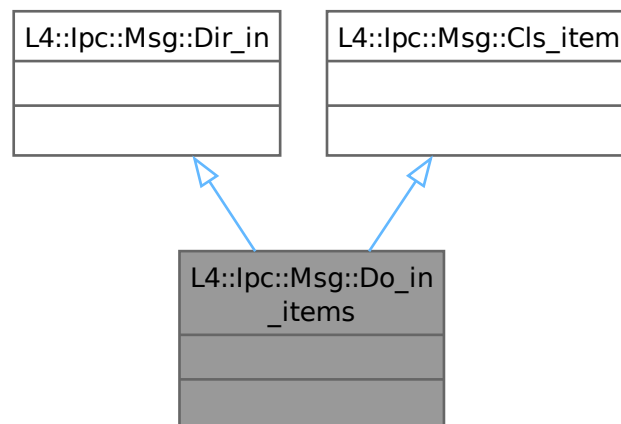
- `l4/sys/cxx/ipc_basics`

15.121 L4::lpc::Msg::Do_in_items Struct Reference

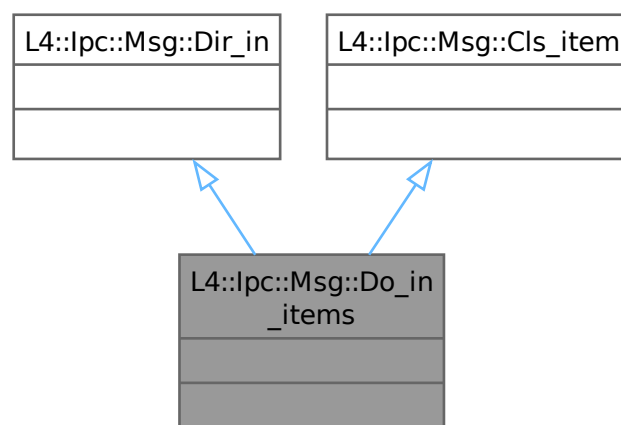
Marker for Input items.

```
#include <ipc_basics>
```

Inheritance diagram for L4::lpc::Msg::Do_in_items:



Collaboration diagram for L4::lpc::Msg::Do_in_items:



15.121.1 Detailed Description

Marker for Input items.

Definition at line 162 of file [ipc_basics](#).

The documentation for this struct was generated from the following file:

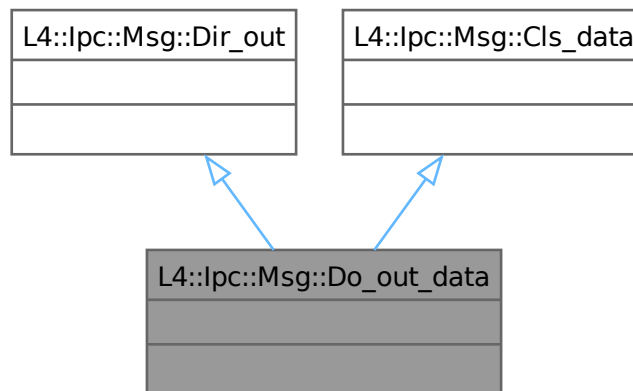
- l4/sys/cxx/ipc_basics

15.122 L4::lpc::Msg::Do_out_data Struct Reference

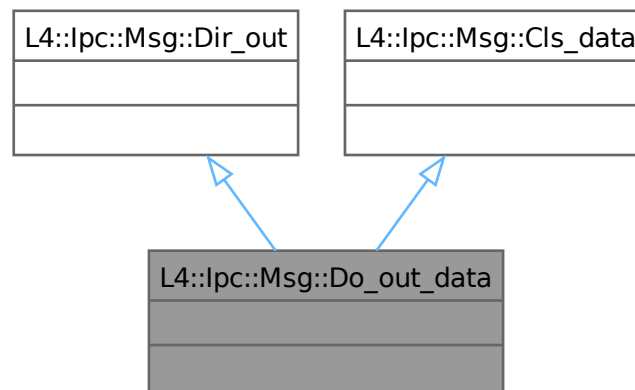
Marker for Output data.

```
#include <ipc_basics>
```

Inheritance diagram for L4::lpc::Msg::Do_out_data:



Collaboration diagram for L4::lpc::Msg::Do_out_data:



15.122.1 Detailed Description

Marker for Output data.

Definition at line 160 of file [ipc_basics](#).

The documentation for this struct was generated from the following file:

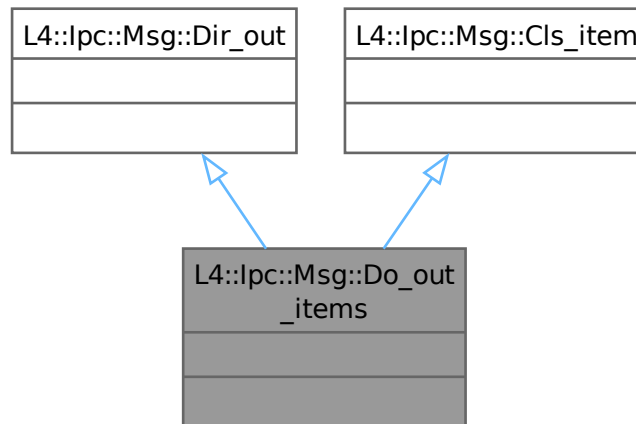
- `I4/sys/cxx/ipc_basics`

15.123 L4::lpc::Msg::Do_out_items Struct Reference

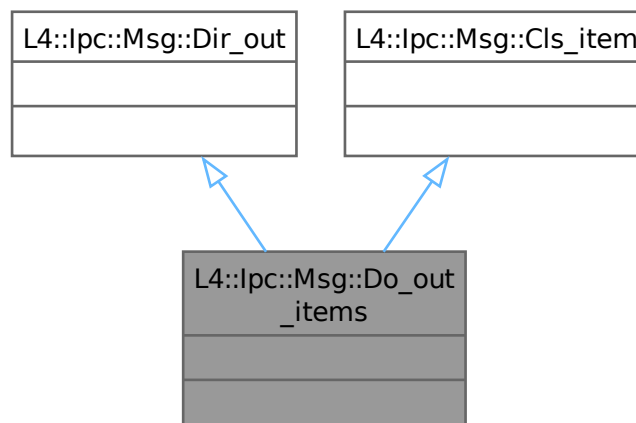
Marker for Output items.

```
#include <ipc_basics>
```

Inheritance diagram for L4::lpc::Msg::Do_out_items:



Collaboration diagram for L4::lpc::Msg::Do_out_items:



15.123.1 Detailed Description

Marker for Output items.

Definition at line 164 of file [ipc_basics](#).

The documentation for this struct was generated from the following file:

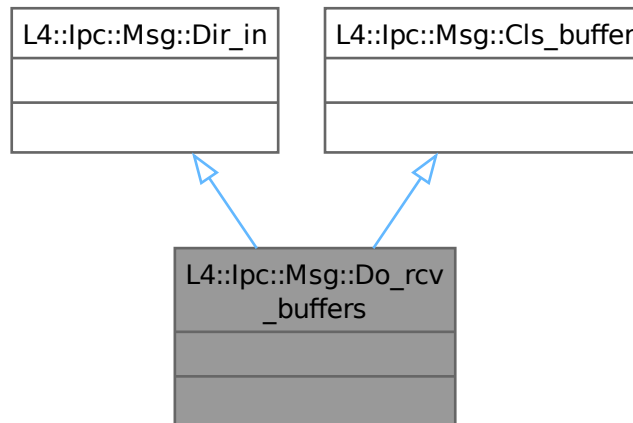
- `l4/sys/cxx/ipc_basics`

15.124 L4::lpc::Msg::Do_rcv_buffers Struct Reference

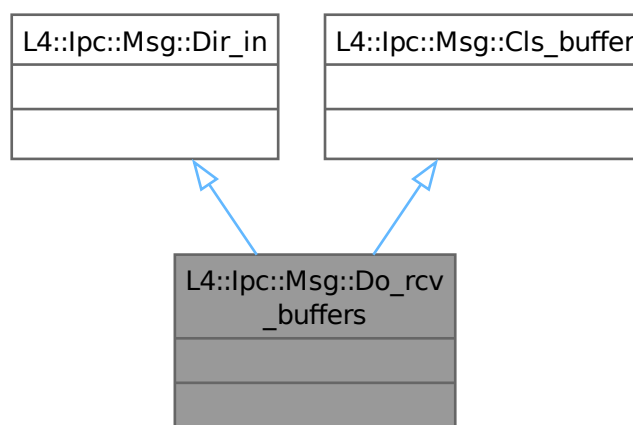
Marker for receive buffers.

```
#include <ipc_basics>
```

Inheritance diagram for L4::lpc::Msg::Do_rcv_buffers:



Collaboration diagram for L4::lpc::Msg::Do_rcv_buffers:



15.124.1 Detailed Description

Marker for receive buffers.

Definition at line 166 of file [ipc_basics](#).

The documentation for this struct was generated from the following file:

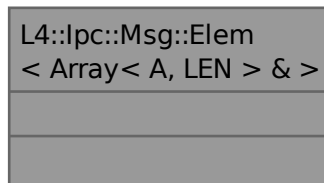
- `l4/sys/cxx/ipc_basics`

15.125 L4::ipc::Msg::Elem< Array< A, LEN > & > Struct Template Reference

[Array](#) as output argument.

```
#include <ipc_array>
```

Collaboration diagram for L4::ipc::Msg::Elem< Array< A, LEN > & >:



Public Types

- using **arg_type** = [Array](#)<A, LEN> & [Array](#)<> & at the interface.
- using **svr_type** = [Array_ref](#)<A, LEN> [Array_ref](#)<> as server storage type.
- using **svr_arg_type** = [svr_type](#) & [Array_ref](#)<> & at the server side.

15.125.1 Detailed Description

```
template<typename A, typename LEN>
struct L4::ipc::Msg::Elem< Array< A, LEN > & >
```

[Array](#) as output argument.

Definition at line 170 of file [ipc_array](#).

The documentation for this struct was generated from the following file:

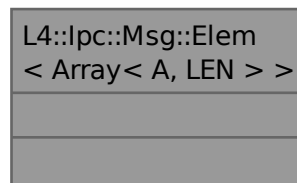
- `l4/sys/cxx/ipc_array`

15.126 L4::lpc::Msg::Elem< Array< A, LEN > > Struct Template Reference

[Array](#) as input arguments.

```
#include <ipc_array>
```

Collaboration diagram for L4::lpc::Msg::Elem< Array< A, LEN > >:



Public Types

- using **arg_type** = [Array](#)<A, LEN>
[Array](#)<> as argument at the interface.
- using **svr_type** = [Array_ref](#)<A, LEN>
[Array_ref](#)<> at the server side.

15.126.1 Detailed Description

```
template<typename A, typename LEN>
struct L4::lpc::Msg::Elem< Array< A, LEN > >
```

[Array](#) as input arguments.

Definition at line 158 of file [ipc_array](#).

The documentation for this struct was generated from the following file:

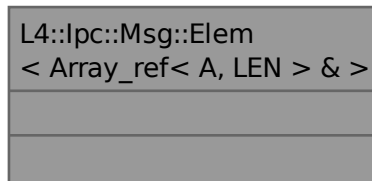
- l4/sys/cxx/ipc_array

15.127 L4::lpc::Msg::Elem< Array_ref< A, LEN > & > Struct Template Reference

[Array_ref](#) as output argument.

```
#include <ipc_array>
```

Collaboration diagram for L4::lpc::Msg::Elem< Array_ref< A, LEN > & >:



Public Types

- using **arg_type** = [Array_ref](#)<A, LEN> &
[Array_ref](#)<> at the interface.
- using **svr_type** = [Array_ref](#)<L4::Types::Remove_const_t<A>, LEN>
[Array_ref](#)<> as server storage.
- using **svr_arg_type** = [svr_type](#) &
[Array_ref](#)<> & as server argument.

15.127.1 Detailed Description

```
template<typename A, typename LEN>
struct L4::lpc::Msg::Elem< Array_ref< A, LEN > & >
```

[Array_ref](#) as output argument.

Definition at line 183 of file [ipc_array](#).

The documentation for this struct was generated from the following file:

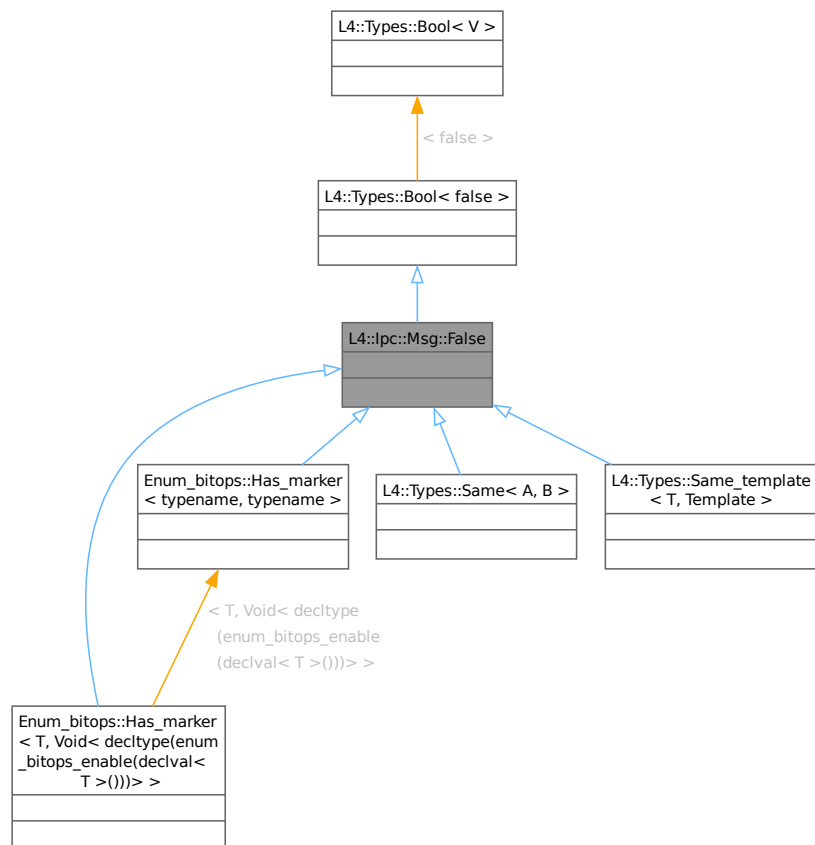
- l4/sys/cxx/ipc_array

15.128 L4::lpc::Msg::False Struct Reference

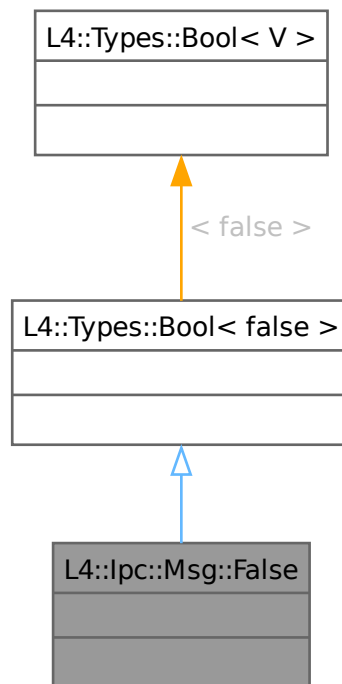
False meta value.

```
#include <types>
```

Inheritance diagram for L4::lpc::Msg::False:



Collaboration diagram for L4::Ipc::Msg::False:



Additional Inherited Members

Public Types inherited from [L4::Types::Bool< false >](#)

- using **type**
The meta type itself.

15.128.1 Detailed Description

[False](#) meta value.

Definition at line [320](#) of file [types](#).

The documentation for this struct was generated from the following file:

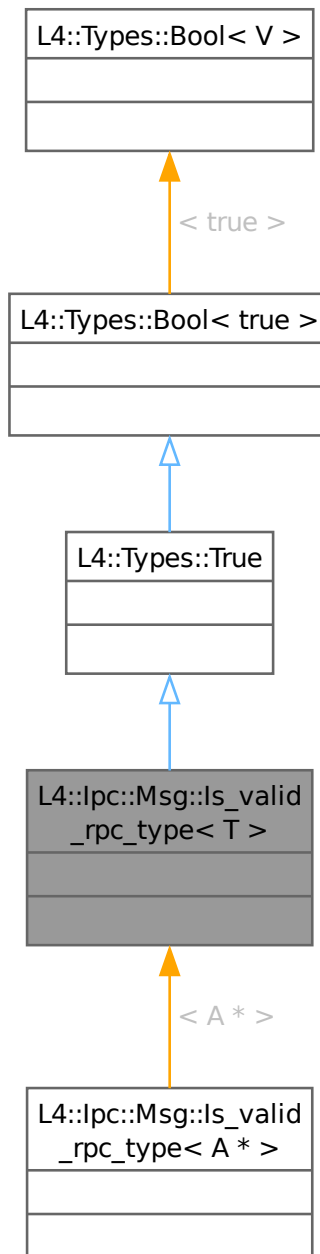
- [l4/sys/cxx/types](#)

15.129 L4::lpc::Msg::ls_valid_rpc_type< T > Struct Template Reference

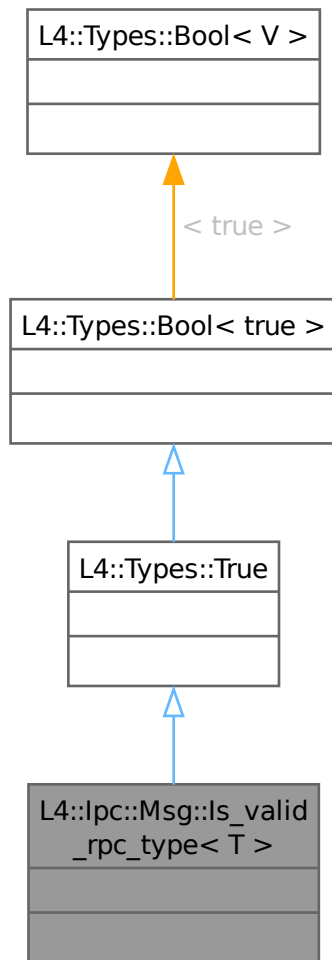
Type trait defining a valid RPC parameter type.

```
#include <ipc_basics>
```

Inheritance diagram for L4::lpc::Msg::ls_valid_rpc_type< T >:



Collaboration diagram for `L4::ipc::Msg::ls_valid_rpc_type< T >`:



Additional Inherited Members

Public Types inherited from `L4::Types::Bool< true >`

- using `type`
The meta type itself.

15.129.1 Detailed Description

```
template<typename T>
struct L4::ipc::Msg::ls_valid_rpc_type< T >
```

Type trait defining a valid RPC parameter type.

Definition at line 361 of file `ipc_basics`.

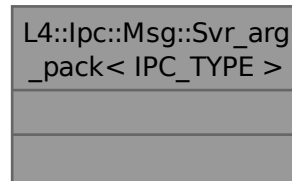
The documentation for this struct was generated from the following file:

- `I4/sys/cxx/ipc_basics`

15.130 L4::lpc::Msg::Svr_arg_pack< IPC_TYPE > Struct Template Reference

Server-side RPC arguments data structure used to provide arguments to the server-side implementation of an RPC function.

Collaboration diagram for L4::lpc::Msg::Svr_arg_pack< IPC_TYPE >:



15.130.1 Detailed Description

```
template<typename IPC_TYPE>
struct L4::lpc::Msg::Svr_arg_pack< IPC_TYPE >
```

Server-side RPC arguments data structure used to provide arguments to the server-side implementation of an RPC function.

Definition at line 189 of file [ipc_server](#).

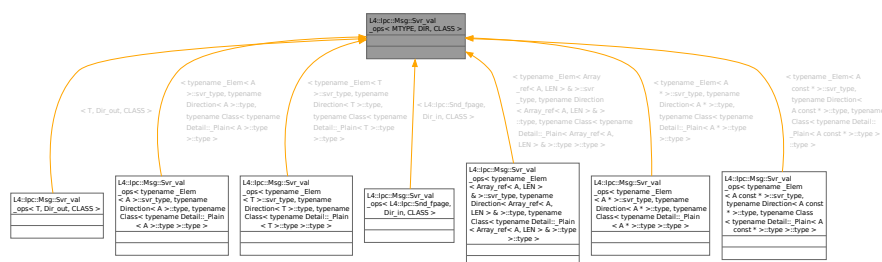
The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_server

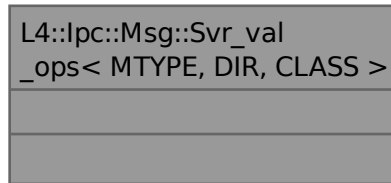
15.131 L4::lpc::Msg::Svr_val_ops< MTYPE, DIR, CLASS > Struct Template Reference

Defines server-side handling for MTYPE server arguments.

Inheritance diagram for L4::lpc::Msg::Svr_val_ops< MTYPE, DIR, CLASS >:



Collaboration diagram for L4::lpc::Msg::Svr_val_ops< MTYPE, DIR, CLASS >:



15.131.1 Detailed Description

```
template<typename MTYPE, typename DIR, typename CLASS>
struct L4::lpc::Msg::Svr_val_ops< MTYPE, DIR, CLASS >
```

Defines server-side handling for `MTYPE` server arguments.

Template Parameters

| | |
|--------------|--|
| <i>MTYPE</i> | Elem<T>::svr_type (where T is the type used in the RPC definition) |
| <i>DIR</i> | Dir_in (client -> server), or Dir_out (server -> client) |
| <i>CLASS</i> | Cls_data , Cls_item , or Cls_buffer |

Definition at line [281](#) of file [ipc_basics](#).

The documentation for this struct was generated from the following file:

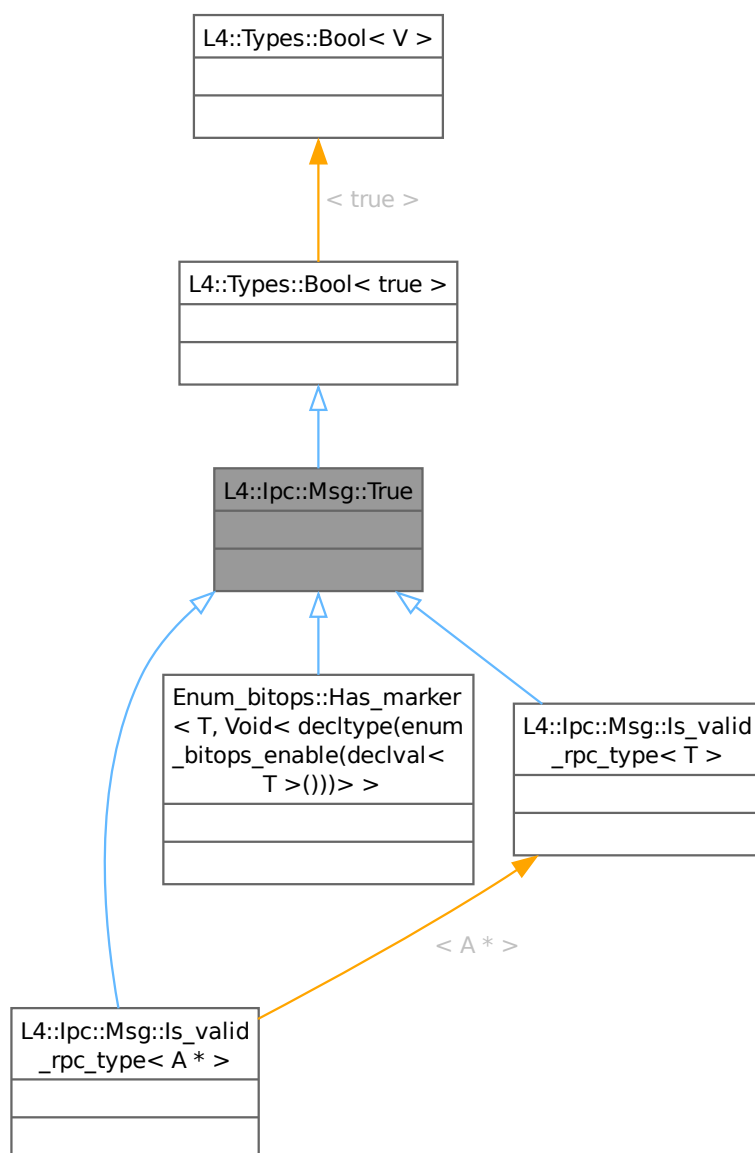
- `l4/sys/cxx/ipc_basics`

15.132 L4::lpc::Msg::True Struct Reference

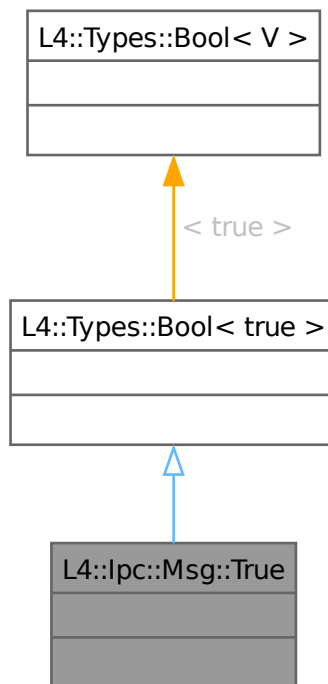
[True](#) meta value.

```
#include <types>
```

Inheritance diagram for L4::lpc::Msg::True:



Collaboration diagram for L4::lpc::Msg::True:



Additional Inherited Members

Public Types inherited from [L4::Types::Bool< true >](#)

- using **type**
The meta type itself.

15.132.1 Detailed Description

[True](#) meta value.

Definition at line [324](#) of file [types](#).

The documentation for this struct was generated from the following file:

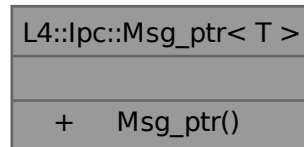
- [l4/sys/cxx/types](#)

15.133 L4::lpc::Msg_ptr< T > Class Template Reference

Pointer to an element of type T in an [lpc::lstream](#).

```
#include <ipc_stream>
```

Collaboration diagram for L4::lpc::Msg_ptr< T >:



Public Member Functions

- [Msg_ptr](#) (T *&p)

Create a [Msg_ptr](#) object that set pointer *p* to point into the message buffer.

15.133.1 Detailed Description

```
template<typename T>
class L4::lpc::Msg_ptr< T >
```

Pointer to an element of type T in an [lpc::lstream](#).

This wrapper can be used to extract an element of type T from an [lpc::lstream](#), whereas the data is not copied out, but a pointer into the message buffer itself is returned. With is mechanism it is possible to avoid an extra copy of large data structures from a received IPC message, instead the returned pointer gives direct access to the data in the message.

See [msg_ptr\(\)](#).

Definition at line 229 of file [ipc_stream](#).

15.133.2 Constructor & Destructor Documentation

15.133.2.1 Msg_ptr()

```
template<typename T>
L4::Ipc::Msg_ptr< T >::Msg_ptr (
    T *& p) [inline], [explicit]
```

Create a [Msg_ptr](#) object that set pointer p to point into the message buffer.

Parameters

| | |
|----------|--|
| <i>p</i> | The pointer that is adjusted to point into the message buffer. |
|----------|--|

Definition at line 240 of file [ipc_stream](#).

The documentation for this class was generated from the following file:

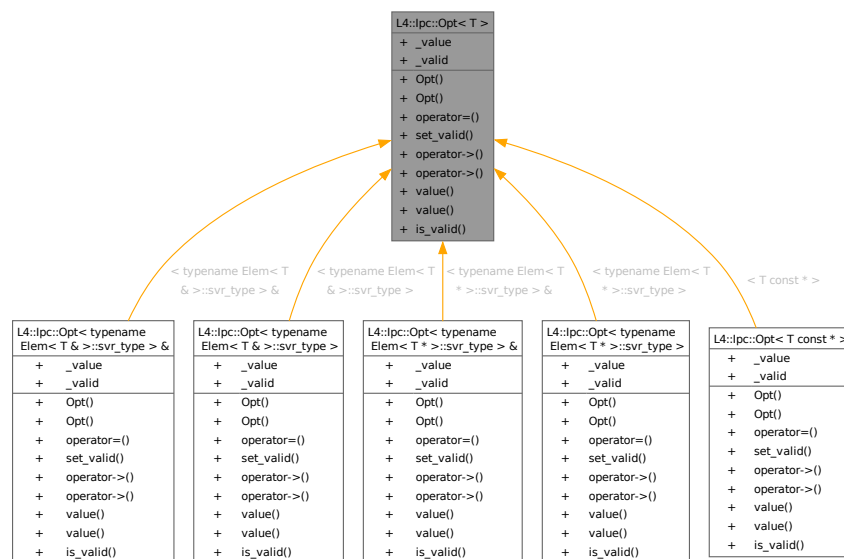
- [l4/cxx/ipc_stream](#)

15.134 L4::Ipc::Opt< T > Struct Template Reference

Attribute for defining an optional RPC argument.

```
#include <ipc_types>
```

Inheritance diagram for L4::Ipc::Opt< T >:



Collaboration diagram for L4::lpc::Opt< T >:

| L4::lpc::Opt< T > |
|-------------------|
| + _value |
| + _valid |
| + Opt() |
| + Opt() |
| + operator=() |
| + set_valid() |
| + operator->() |
| + operator->() |
| + value() |
| + value() |
| + is_valid() |

Public Member Functions

- **Opt** () noexcept
Make an absent optional argument.
- **Opt** (T **value**) noexcept
Make a present optional argument with the given value.
- **Opt** & **operator=** (T **value**) noexcept
Assign a value to the optional argument (makes the argument present).
- void **set_valid** (bool valid=true) noexcept
Set the argument to present or absent.
- T * **operator->** () noexcept
Get the pointer to the value.
- T const * **operator->** () const noexcept
Get the const pointer to the value.
- T **value** () const noexcept
Get the value.
- T & **value** () noexcept
Get the value.
- bool **is_valid** () const noexcept
Get true if present, false if not.

Data Fields

- T **_value**
The value.
- bool **_valid**
True if the optional argument is present, false else.

15.134.1 Detailed Description

```
template<typename T>  
struct L4::lpc::Opt< T >
```

Attribute for defining an optional RPC argument.

Definition at line 136 of file [ipc_types](#).

The documentation for this struct was generated from the following file:

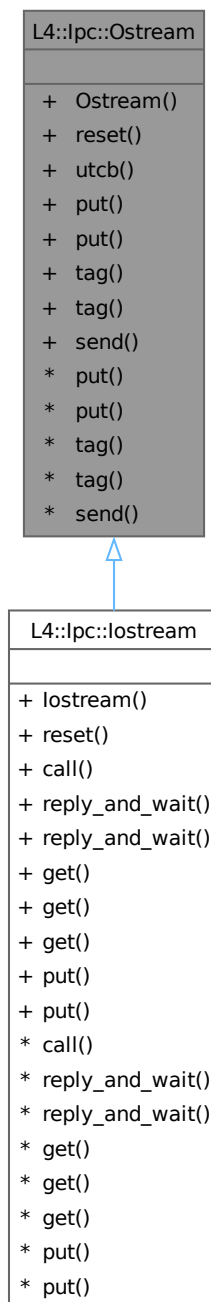
- [l4/sys/cxx/ipc_types](#)

15.135 L4::lpc::Ostream Class Reference

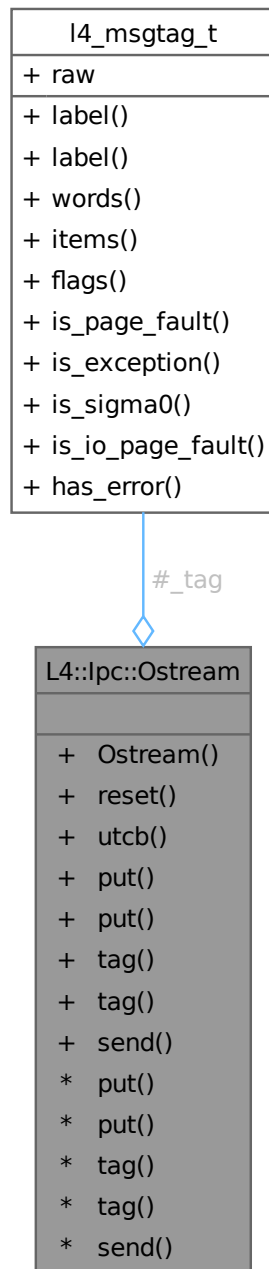
Output stream for IPC marshalling.

```
#include <ipc_stream>
```

Inheritance diagram for L4::lpc::Ostream:



Collaboration diagram for L4::lpc::Ostream:



Public Member Functions

- **Ostream** ([l4_utcb_t](#) *[utcb](#))
Create an IPC output stream using the given message buffer [utcb](#).
- void **reset** ()
Reset the stream to empty, same state as a newly created stream.
- [l4_utcb_t](#) * **utcb** () const

Return utcb pointer.

Get/Put functions.

These functions are basically used to implement the insertion operators (<<) and should not be called directly.

- `template<typename T>`
`bool put (T *buf, unsigned long size)`
Put an array with `size` elements of type `T` into the stream.
- `template<typename T>`
`bool put (T const &v)`
Insert an element of type `T` into the stream.
- `l4_msgtag_t tag () const`
Extract the `L4` message tag from the stream.
- `l4_msgtag_t & tag ()`
Extract a reference to the `L4` message tag from the stream.

IPC operations.

- `l4_msgtag_t send (l4_cap_idx_t dst, long proto=0, unsigned flags=0)`
Send the message via IPC to the given receiver.

15.135.1 Detailed Description

Output stream for IPC marshalling.

`lpc::Ostream` is part of the dynamic IPC marshalling infrastructure, as well as `lpc::Istream` and `lpc::lostream`.

`lpc::Ostream` is an output stream supporting insertion of values into an IPC message buffer. A IPC message can be marshalled using the usual insertion operator <<, see [IPC stream operators](#) .

There exist some special wrapper classes to insert arrays (see `lpc::Buf_cp_out`) and indirect strings (see `Msg_↔out_buffer` and `Msg_io_buffer`).

Definition at line 623 of file [ipc_stream](#).

15.135.2 Member Function Documentation

15.135.2.1 put() [1/2]

```
template<typename T>
bool L4::Ipc::Ostream::put (
    T * buf,
    unsigned long size) [inline]
```

Put an array with `size` elements of type `T` into the stream.

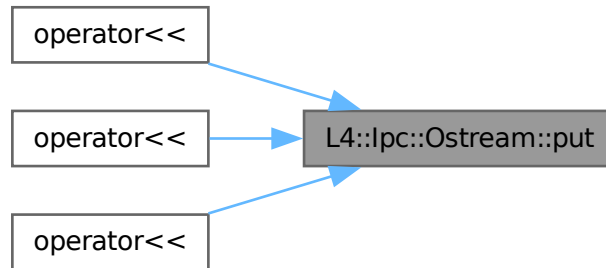
Parameters

| | |
|-------------|---|
| <i>buf</i> | A pointer to the array to insert into the buffer. |
| <i>size</i> | The number of elements in the array. |

Definition at line 660 of file [ipc_stream](#).

Referenced by [operator<<\(\)](#), [operator<<\(\)](#), and [operator<<\(\)](#).

Here is the caller graph for this function:



15.135.2.2 put() [2/2]

```
template<typename T>
bool L4::Ipc::Ostream::put (
    T const & v) [inline]
```

Insert an element of type T into the stream.

Parameters

| | |
|----------|------------------------|
| <i>v</i> | The element to insert. |
|----------|------------------------|

Definition at line 678 of file [ipc_stream](#).

15.135.2.3 send()

```
l4_msgtag_t L4::Ipc::Ostream::send (
    l4_cap_idx_t dst,
    long proto = 0,
    unsigned flags = 0) [inline]
```

Send the message via IPC to the given receiver.

Parameters

| | |
|--------------|----------------------------------|
| <i>dst</i> | The destination for the message. |
| <i>proto</i> | Protocol to use. |

| | |
|--------------|---------------|
| <i>flags</i> | Flags to use. |
|--------------|---------------|

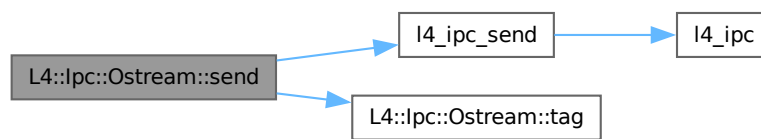
Returns

The syscall return tag.

Definition at line 959 of file [ipc_stream](#).

References [L4_IPC_NEVER](#), [l4_ipc_send\(\)](#), [L4_MSGTAG_FLAGS](#), and [tag\(\)](#).

Here is the call graph for this function:



15.135.2.4 tag() [1/2]

```
l4\_msgtag\_t & L4::Ipcc::Ostream::tag () [inline]
```

Extract a reference to the [L4](#) message tag from the stream.

Returns

A reference to the [L4](#) message tag.

Definition at line 713 of file [ipc_stream](#).

15.135.2.5 tag() [2/2]

```
l4\_msgtag\_t L4::Ipcc::Ostream::tag () const [inline]
```

Extract the [L4](#) message tag from the stream.

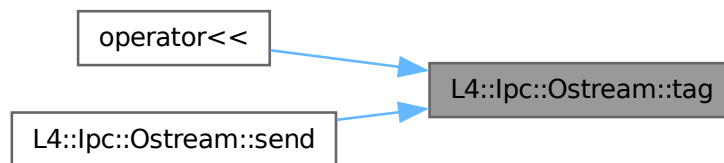
Returns

The extracted [L4](#) message tag.

Definition at line [706](#) of file [ipc_stream](#).

Referenced by [operator<<\(\)](#), and [send\(\)](#).

Here is the caller graph for this function:



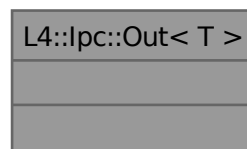
The documentation for this class was generated from the following file:

- [l4/cxx/ipc_stream](#)

15.136 L4::ipc::Out< T > Struct Template Reference

Mark an argument as a output value in an RPC signature.

Collaboration diagram for `L4::ipc::Out< T >`:



15.136.1 Detailed Description

```
template<typename T>
struct L4::lpc::Out< T >
```

Mark an argument as a output value in an RPC signature.

Template Parameters

| | |
|----------|------------------------------------|
| <i>T</i> | The original type of the argument. |
|----------|------------------------------------|

Note

The use of [Out<>](#) is usually not needed, because typical output data types in C++ (pointers to non-const objects or non-const references are interpreted as output values anyway. However, there are some data types, such as returned capabilities that can be marked as such by using [Out<>](#).

Definition at line 31 of file [ipc_types](#).

The documentation for this struct was generated from the following file:

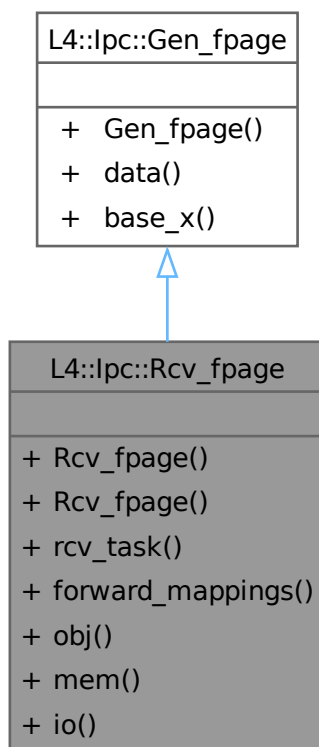
- [l4/sys/cxx/ipc_types](#)

15.137 L4::lpc::Rcv_fpage Class Reference

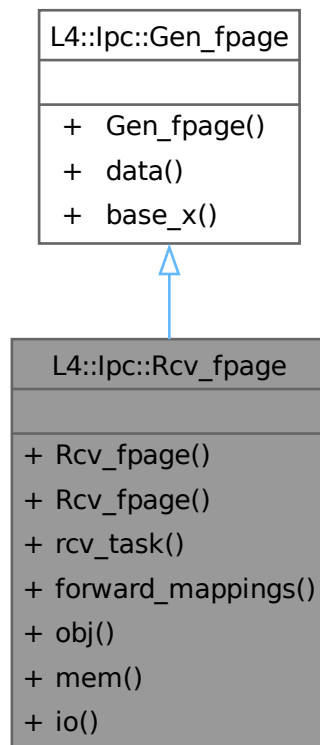
Non-small receive item.

```
#include <ipc_types>
```

Inheritance diagram for L4::lpc::Rcv_fpage:



Collaboration diagram for L4::lpc::Rcv_fpage:



Public Member Functions

- **Rcv_fpage** () noexcept
Construct a void receive item.
- **Rcv_fpage** (l4_fpage_t const &fp, l4_addr_t snd_base=0, l4_cap_idx_t rcv_task=L4_INVALID_CAP) noexcept
Construct a non-small receive item.
- **l4_cap_idx_t rcv_task** () const
Get the capability index of the destination task for received capabilities.
- bool **forward_mappings** () const noexcept
Check if rcv_task() shall be used as destination for received capabilities.

Public Member Functions inherited from L4::lpc::Gen_fpage

- **Gen_fpage** (l4_umword_t base, l4_umword_t data) noexcept
Construct from raw values.
- **l4_umword_t data** () const noexcept
Return the raw flexpage descriptor.
- **l4_umword_t base_x** () const noexcept
Return the raw base descriptor.

Static Public Member Functions

- static [Rcv_fpage obj](#) ([l4_cap_idx_t](#) base, int order, [l4_addr_t](#) snd_base=0, [L4::Cap](#)< void > [rcv_task](#)=[L4::Cap](#)< void >::Invalid) noexcept
Construct a non-small receive item for the object space.
- static [Rcv_fpage mem](#) ([l4_addr_t](#) base, int order, [l4_addr_t](#) snd_base=0, [L4::Cap](#)< void > [rcv_task](#)=[L4::Cap](#)< void >::Invalid) noexcept
Construct a receive item for the memory space.
- static [Rcv_fpage io](#) (unsigned long base, int order, [l4_addr_t](#) snd_base=0, [L4::Cap](#)< void > [rcv_task](#)=[L4::Cap](#)< void >::Invalid) noexcept
Construct a receive item for the I/O port space.

Additional Inherited Members

Public Types inherited from [L4::lpc::Gen_fpage](#)

- enum [Type](#) { [Special](#) = [L4_FPAGE_SPECIAL](#) << 4 , [Memory](#) = [L4_FPAGE_MEMORY](#) << 4 , [Io](#) = [L4_FPAGE_IO](#) << 4 , [Obj](#) = [L4_FPAGE_OBJ](#) << 4 }
- Type of mapping object, see [L4_fpage_type](#).*

15.137.1 Detailed Description

Non-small receive item.

This class represents a non-small receive item. A receive item is a message item in the buffer registers of the UTCB of the receiver (see [l4_utcb_br\(\)](#)).

Definition at line 544 of file [ipc_types](#).

15.137.2 Constructor & Destructor Documentation

15.137.2.1 [Rcv_fpage\(\)](#)

```
L4::Ipc::Rcv_fpage::Rcv_fpage (
    l4\_fpage\_t const & fp,
    l4\_addr\_t snd_base = 0,
    l4\_cap\_idx\_t rcv_task = L4\_INVALID\_CAP) [inline], [noexcept]
```

Construct a non-small receive item.

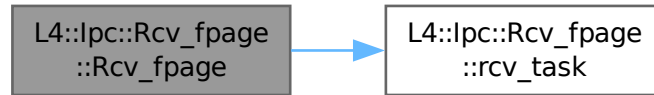
Parameters

| | |
|-----------------|--|
| <i>fp</i> | Flexpage defining where and which kind of capabilities may be received. |
| <i>snd_base</i> | Reserved; should be zero. |
| <i>rcv_task</i> | Optional destination task for received capabilities. If invalid, capabilities are received in the invoking task. |

Definition at line 561 of file [ipc_types](#).

References [L4_INVALID_CAP](#), [L4_ITEM_MAP](#), [L4_RCV_ITEM_FORWARD_MAPPINGS](#), and [rcv_task\(\)](#).

Here is the call graph for this function:



15.137.3 Member Function Documentation

15.137.3.1 io()

```

Rcv_fpage L4::lpc::Rcv_fpage::io (
    unsigned long base,
    int order,
    l4_addr_t snd_base = 0,
    L4::Cap< void > rcv_task = L4::Cap<void>::Invalid) [inline], [static], [noexcept]
  
```

Construct a receive item for the I/O port space.

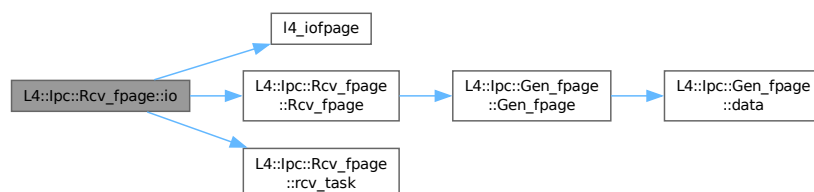
Parameters

| | |
|-----------------|--|
| <i>base</i> | Start of flexpage (see l4_iofpage()). |
| <i>order</i> | Log ₂ size of flexpage (see l4_iofpage()). |
| <i>snd_base</i> | Reserved; should be zero. |
| <i>rcv_task</i> | Optional destination task for received capabilities. If invalid, capabilities are received in the invoking task. |

Definition at line 609 of file [ipc_types](#).

References [L4::Cap_base::Invalid](#), [l4_iofpage\(\)](#), [Rcv_fpage\(\)](#), and [rcv_task\(\)](#).

Here is the call graph for this function:



15.137.3.2 mem()

```
Rcv_fpage L4::Ipc::Rcv_fpage::mem (
    l4_addr_t base,
    int order,
    l4_addr_t snd_base = 0,
    L4::Cap< void > rcv_task = L4::Cap<void>::Invalid) [inline], [static], [noexcept]
```

Construct a receive item for the memory space.

Parameters

| | |
|-----------------|--|
| <i>base</i> | Start of flexpage (see l4_fpage()). |
| <i>order</i> | Log ₂ size of flexpage (see l4_fpage()). |
| <i>snd_base</i> | Reserved; should be zero. |
| <i>rcv_task</i> | Optional destination task for received capabilities. If invalid, capabilities are received in the invoking task. |

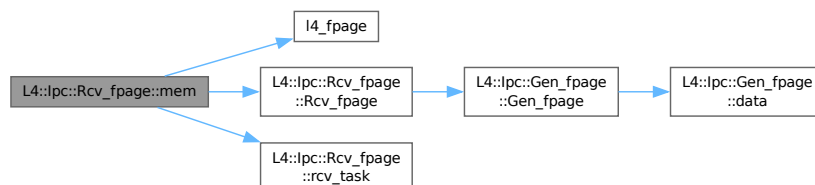
Examples

[examples/libs/l4re/streammap/client.cc](#).

Definition at line 594 of file [ipc_types](#).

References [L4::Cap_base::Invalid](#), [l4_fpage\(\)](#), [Rcv_fpage\(\)](#), and [rcv_task\(\)](#).

Here is the call graph for this function:



15.137.3.3 obj()

```
Rcv_fpage L4::Ipc::Rcv_fpage::obj (
    l4_cap_idx_t base,
    int order,
    l4_addr_t snd_base = 0,
    L4::Cap< void > rcv_task = L4::Cap<void>::Invalid) [inline], [static], [noexcept]
```

Construct a non-small receive item for the object space.

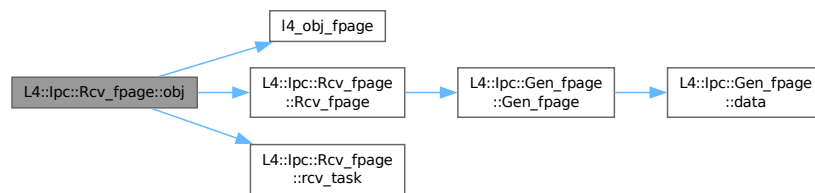
Parameters

| | |
|-----------------|--|
| <i>base</i> | Start of flexpage (see l4_obj_fpage()). |
| <i>order</i> | \log_2 size of flexpage (see l4_obj_fpage()). |
| <i>snd_base</i> | Reserved; should be zero. |
| <i>rcv_task</i> | Optional destination task for received capabilities. If invalid, capabilities are received in the invoking task. |

Definition at line 578 of file [ipc_types](#).

References [L4::Cap_base::Invalid](#), [l4_obj_fpage\(\)](#), [Rcv_fpage\(\)](#), and [rcv_task\(\)](#).

Here is the call graph for this function:



15.137.3.4 rcv_task()

```
l4_cap_idx_t L4::lpc::Rcv_fpage::rcv_task () const [inline]
```

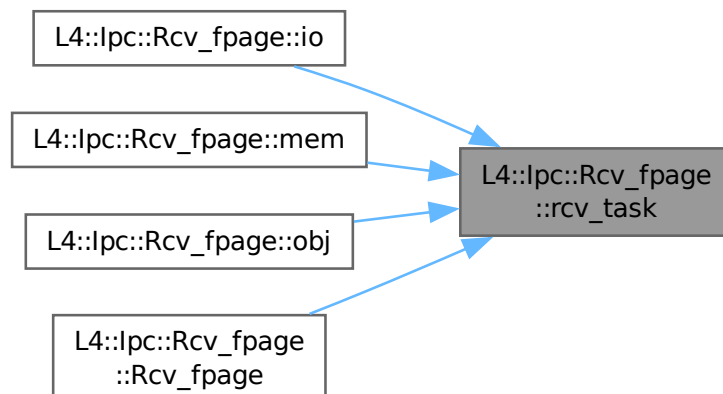
Get the capability index of the destination task for received capabilities.

Only relevant if [forward_mappings\(\)](#) is true.

Definition at line 620 of file [ipc_types](#).

Referenced by [io\(\)](#), [mem\(\)](#), [obj\(\)](#), and [Rcv_fpage\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

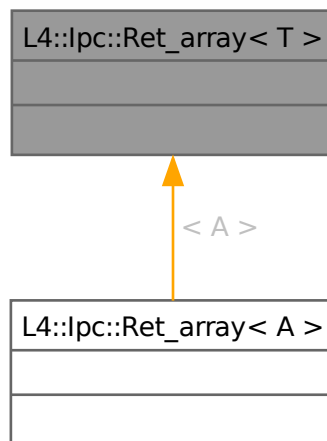
- [l4/sys/cxx/ipc_types](#)

15.138 L4::ipc::Ret_array< T > Struct Template Reference

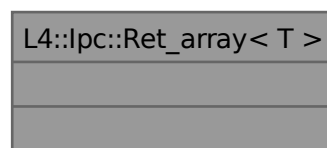
Dynamically sized output array of type T.

```
#include <ipc_ret_array>
```

Inheritance diagram for L4::ipc::Ret_array< T >:



Collaboration diagram for L4::ipc::Ret_array< T >:



15.138.1 Detailed Description

```
template<typename T>
struct L4::lpc::Ret_array< T >
```

Dynamically sized output array of type T.

Template Parameters

| | |
|----------|--------------------------------------|
| <i>T</i> | The data-type of each array element. |
|----------|--------------------------------------|

[Ret_array<>](#) is a special dynamically sized output array where the number of transmitted elements is passed in the return value of the call (if positive)

Definition at line 23 of file [ipc_ret_array](#).

The documentation for this struct was generated from the following file:

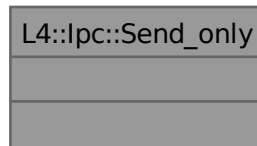
- [l4/sys/cxx/ipc_ret_array](#)

15.139 L4::lpc::Send_only Struct Reference

RPC attribute for a send-only RPC.

```
#include <ipc_iface>
```

Collaboration diagram for L4::lpc::Send_only:



15.139.1 Detailed Description

RPC attribute for a send-only RPC.

This class can be used as FLAGS parameter to [L4::lpc::Msg::Rpc_call](#) and [L4::lpc::Msg::Rpc_inline_call](#) templates and declares the RPC to use send-only semantics and timeouts.

Examples:

```
L4\_RPC(l4\_ret\_t, send, (unsigned value), L4::lpc::Send_only);
```

Definition at line 287 of file [ipc_iface](#).

The documentation for this struct was generated from the following file:

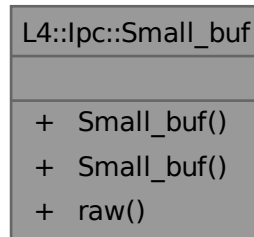
- [l4/sys/cxx/ipc_iface](#)

15.140 L4::lpc::Small_buf Class Reference

A receive item for receiving a single object capability.

```
#include <ipc_types>
```

Collaboration diagram for L4::lpc::Small_buf:



Public Member Functions

- [Small_buf](#) ([L4::Cap](#)< void > cap, unsigned long flags=0) noexcept
Create a receive item from a C++ cap.
- [Small_buf](#) ([l4_cap_idx_t](#) cap, unsigned long flags=0) noexcept
Create a receive item from a C cap.
- [l4_umword_t raw](#) () const noexcept
Return the raw data.

15.140.1 Detailed Description

A receive item for receiving a single object capability.

This class is the main abstraction for receiving object capabilities via [lpc::lstream](#). To receive an object capability, an instance of [Small_buf](#) that refers to an empty capability slot must be inserted into the [lpc::lstream](#) before the receive operation.

Definition at line 257 of file [ipc_types](#).

15.140.2 Constructor & Destructor Documentation

15.140.2.1 Small_buf() [1/2]

```
L4::Ipc::Small_buf::Small_buf (  
    L4::Cap< void > cap,  
    unsigned long flags = 0) [inline], [explicit], [noexcept]
```

Create a receive item from a C++ cap.

Parameters

| | |
|--------------|---|
| <i>cap</i> | Capability slot where to save the capability. |
| <i>flags</i> | Receive buffer flags, see l4_msg_item_consts_t . L4_RCV_ITEM_SINGLE_CAP will always be set. |

Definition at line 267 of file [ipc_types](#).

References [L4_RCV_ITEM_SINGLE_CAP](#).

15.140.2.2 Small_buf() [2/2]

```
L4::Ipc::Small_buf::Small_buf (  
    l4_cap_idx_t cap,  
    unsigned long flags = 0) [inline], [explicit], [noexcept]
```

Create a receive item from a C cap.

Parameters

| | |
|--------------|---|
| <i>cap</i> | Capability slot where to save the capability. |
| <i>flags</i> | Receive buffer flags, see l4_msg_item_consts_t . L4_RCV_ITEM_SINGLE_CAP will always be set. |

Definition at line 274 of file [ipc_types](#).

References [L4_RCV_ITEM_SINGLE_CAP](#).

The documentation for this class was generated from the following file:

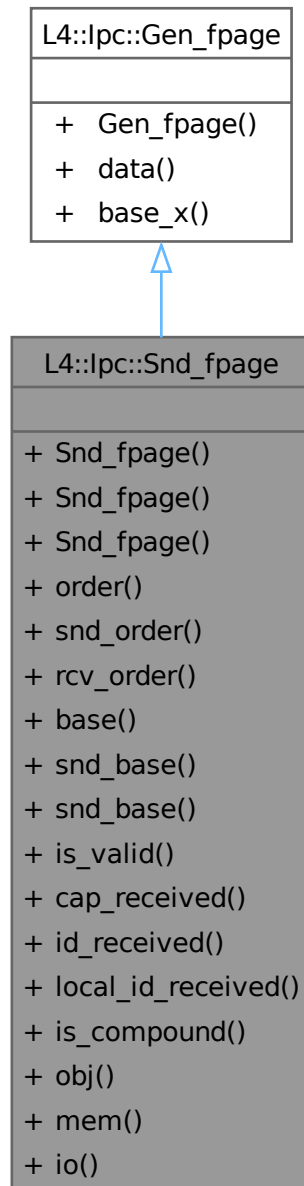
- [l4/sys/cxx/ipc_types](#)

15.141 L4::lpc::Snd_fpage Class Reference

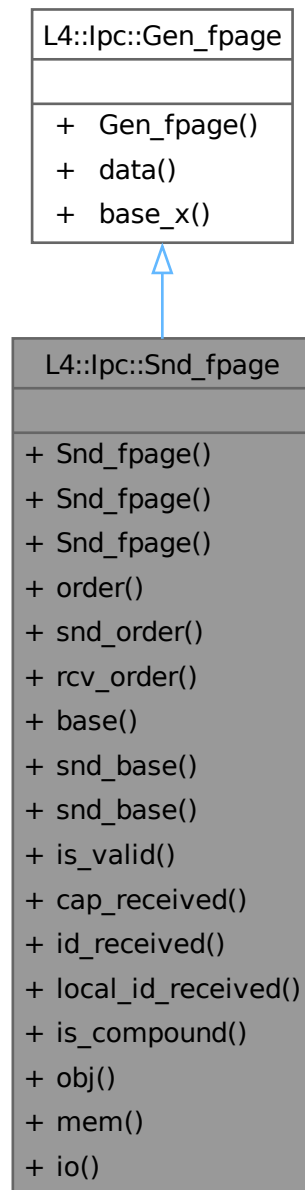
Send item or return item.

```
#include <ipc_types>
```

Inheritance diagram for L4::lpc::Snd_fpage:



Collaboration diagram for L4::lpc::Snd_fpage:



Public Types

- enum [Map_type](#) { [Map](#) = L4_MAP_ITEM_MAP , [Grant](#) = L4_MAP_ITEM_GRANT }
(Defined for send items only.) Kind of mapping.
- enum [Cacheopt](#) { [None](#) = 0 , [Cached](#) = L4_FPAGE_CACHEABLE << 4 , [Buffered](#) = L4_FPAGE_BUFFERABLE << 4 , [Uncached](#) = L4_FPAGE_UNCACHEABLE << 4 }
(Defined for memory send items only.) Caching options, see [l4_fpage_cacheability_opt_t](#).
- enum [Continue](#) { [Single](#) = 0 , [Last](#) = 0 , [More](#) = L4_ITEM_CONT , [Compound](#) = L4_ITEM_CONT }
Specify if the following item is associated with the same receive item as this one, see [L4_ITEM_CONT](#).

Public Types inherited from [L4::lpc::Gen_fpage](#)

- enum [Type](#) { [Special](#) = L4_FPAGE_SPECIAL << 4 , [Memory](#) = L4_FPAGE_MEMORY << 4 , [Io](#) = L4_FPAGE_IO << 4 , [Obj](#) = L4_FPAGE_OBJ << 4 }

Type of mapping object, see [L4_fpage_type](#).

Public Member Functions

- [Snd_fpage](#) ([l4_umword_t](#) base=0, [l4_umword_t](#) data=0) noexcept
Construct from raw values.
- [Snd_fpage](#) ([l4_fpage_t](#) const &fp, [l4_addr_t](#) snd_base=0, [Map_type](#) map_type=[Map](#), [Cacheopt](#) cache=[None](#), [Continue](#) cont=[Last](#)) noexcept
Construct a send item for the memory space.
- [Snd_fpage](#) ([L4::Cap](#)< void > cap, unsigned rights, [Map_type](#) map_type=[Map](#)) noexcept
Construct a send item for the object space.
- unsigned [order](#) () const noexcept
(Defined only if send item or if [local_id_received\(\)](#) is true.) Get log₂ size.
- unsigned [snd_order](#) () const noexcept
(Defined only if send item or if [local_id_received\(\)](#) is true.) Get log₂ size.
- unsigned [rcv_order](#) () const noexcept
(Defined for return items only.) Get log₂ size.
- [l4_addr_t](#) [base](#) () const noexcept
(Defined only if send item or if [local_id_received\(\)](#) is true.) Get the start of the item (i.e., the start of its flexpage).
- [l4_addr_t](#) [snd_base](#) () const noexcept
Get the position in receive window for the case that this item has a different size than the corresponding receive item.
- void [snd_base](#) ([l4_addr_t](#) b) noexcept
Set the position in receive window for the case that this item has a different size than the corresponding receive item.
- bool [is_valid](#) () const noexcept
Check if the capability is valid.
- bool [cap_received](#) () const noexcept
(Defined for return items only.) Check if at least one object capability has been mapped for this item.
- bool [id_received](#) () const noexcept
(Defined for return items only.) Check if an IPC gate label has been received instead of a mapping.
- bool [local_id_received](#) () const noexcept
(Defined for return items only.) Check if a raw object flexpage has been received instead of a mapping.
- bool [is_compound](#) () const noexcept
Check if the item has the compound bit set, see [Continue](#).

Public Member Functions inherited from [L4::lpc::Gen_fpage](#)

- [Gen_fpage](#) ([l4_umword_t](#) base, [l4_umword_t](#) data) noexcept
Construct from raw values.
- [l4_umword_t](#) [data](#) () const noexcept
Return the raw flexpage descriptor.
- [l4_umword_t](#) [base_x](#) () const noexcept
Return the raw base descriptor.

Static Public Member Functions

- static [Snd_fpage obj](#) ([l4_cap_idx_t](#) base, int [order](#), unsigned char rights, [l4_addr_t](#) snd_base=0, [Map_type](#) map_type=[Map](#), [Continue](#) cont=[Last](#)) noexcept
Construct a send item for the object space.
- static [Snd_fpage mem](#) ([l4_addr_t](#) base, int [order](#), unsigned char rights, [l4_addr_t](#) snd_base=0, [Map_type](#) map_type=[Map](#), [Cacheopt](#) cache=[None](#), [Continue](#) cont=[Last](#)) noexcept
Construct a send item for the memory space.
- static [Snd_fpage io](#) (unsigned long base, int [order](#), unsigned char rights, [l4_addr_t](#) snd_base=0, [Map_type](#) map_type=[Map](#), [Continue](#) cont=[Last](#)) noexcept
Construct a send item for the I/O port space.

15.141.1 Detailed Description

Send item or return item.

This class represents a typed message item in the message registers of the UTCB. If it is provided by the sender, then it is a *send item*. If it is provided by the kernel during IPC, it is a *return item*.

Note that some members are dedicated for send items only or return items only.

Definition at line 323 of file [ipc_types](#).

15.141.2 Member Enumeration Documentation

15.141.2.1 Cacheopt

enum [L4::Ipc::Snd_fpage::Cacheopt](#)

(Defined for memory send items only.) Caching options, see [l4_fpage_cacheability_opt_t](#).

Enumerator

| | |
|----------|--|
| None | Copy options from sender. |
| Cached | Cacheability option to enable caches for the mapping. |
| Buffered | Cacheability option to enable buffered writes for the mapping. |
| Uncached | Cacheability option to disable caching for the mapping. |

Definition at line 336 of file [ipc_types](#).

15.141.2.2 Continue

```
enum L4::Ipc::Snd_fpage::Continue
```

Specify if the following item is associated with the same receive item as this one, see [L4_ITEM_CONT](#).

Enumerator

| | |
|----------|---|
| Single | Inverse of Compound . |
| Last | Inverse of More . |
| More | Alias for Compound . |
| Compound | Denote that the following item shall be put into the same receive item as this one. |

Definition at line 346 of file [ipc_types](#).

15.141.2.3 Map_type

```
enum L4::Ipc::Snd_fpage::Map_type
```

(Defined for send items only.) Kind of mapping.

Enumerator

| | |
|-------|---|
| Map | Flag as usual <i>map</i> operation. |
| Grant | <p>Flag as <i>grant</i> instead of <i>map</i> operation. This means, the sender delegates access to the receiver and the kernel removes the rights from the sender (basically a move operation). The mapping in the receiver gets the new parent of any child mappings of the mapping of the sender. Rights revocation via send item/flexpage is <i>not</i> guaranteed to be applied to descendant mappings in case of grant. See Spaces and Mappings for more details on map/grant.</p> <p>Note</p> <p>The grant operation is not performed if the resulting rights of the receiver mapping would not contain the L4_CAP_FPAGE_R bit (for object capabilities) or none of the L4_FPAGE_RWX bits (memory and IO ports). In that case, the mapping is not created in the receiver space and not removed from the sender space.</p> <p>If the removal of the whole mapping from the sender is not possible because the size of the mapped frame at the sender exceeds the size defined by the send or receive flexpage, the grant operation is turned into a regular map operation and the mapping is <i>not</i> removed from the sender. This would happen if, for example, a smaller part of an L4 superpage mapping shall be granted.</p> |

Definition at line 328 of file [ipc_types](#).

15.141.3 Constructor & Destructor Documentation

15.141.3.1 Snd_fpage() [1/2]

```
L4::Ipc::Snd_fpage::Snd_fpage (
    l4_fpage_t const & fp,
```

```

l4_addr_t snd_base = 0,
Map_type map_type = Map,
Cacheopt cache = None,
Continue cont = Last) [inline], [noexcept]

```

Construct a send item for the memory space.

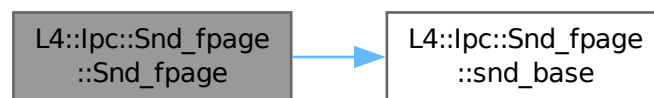
Parameters

| | |
|-----------------|--|
| <i>fp</i> | Memory flexpage defining which range and kind of capabilities shall be sent (see l4_fpage()). |
| <i>snd_base</i> | Position in receive window in case it has a different size than <i>fp</i> . |
| <i>map_type</i> | See Map_type . |
| <i>cache</i> | See Cacheopt . |
| <i>cont</i> | See Continue . |

Definition at line 370 of file [ipc_types](#).

References [L4_ITEM_MAP](#), [Last](#), [Map](#), [None](#), and [snd_base\(\)](#).

Here is the call graph for this function:



15.141.3.2 Snd_fpage() [2/2]

```

L4::Ipc::Snd_fpage::Snd_fpage (
    L4::Cap< void > cap,
    unsigned rights,
    Map_type map_type = Map) [inline], [noexcept]

```

Construct a send item for the object space.

Parameters

| | |
|-----------------|--|
| <i>cap</i> | Capability to be sent. |
| <i>rights</i> | Permissions to be transferred. See L4_cap_fpage_rights and Attributes and additional permissions for object send items . |
| <i>map_type</i> | See Map_type . |

Definition at line 386 of file [ipc_types](#).

References [L4_ITEM_MAP](#), and [Map](#).

15.141.4 Member Function Documentation

15.141.4.1 cap_received()

```
bool L4::Ipc::Snd_fpage::cap_received () const [inline], [noexcept]
```

(*Defined for return items only.*) Check if at least one object capability has been mapped for this item.

The capabilities themselves can then be retrieved from the cap slots that have been provided in the receive operation.

Note

If this function returns `true` and the receive window covers more than one capability slot, then it is not possible to determine which slots actually got capabilities mapped from the sender.

If the received capabilities do not have type object (see [L4_FPAGE_OBJ](#)), then this function returns `false`.

Definition at line 496 of file [ipc_types](#).

15.141.4.2 id_received()

```
bool L4::Ipc::Snd_fpage::id_received () const [inline], [noexcept]
```

(*Defined for return items only.*) Check if an IPC gate label has been received instead of a mapping.

If the [L4_RCV_ITEM_LOCAL_ID](#) flag has been set by the receiver, the conditions for [local_id_received\(\)](#) do not apply, the sender sent an IPC gate capability, and the receiving thread is in the same task as the thread that is attached to the IPC gate, then no mapping is done for this item and only the bitwise OR (|) of the label of the IPC gate and the special and write permission ([L4_CAP_FPAGE_S](#) and [L4_CAP_FPAGE_W](#)) that would have been mapped is received.

The bitwise OR of the label and the permissions can be retrieved with [Gen_fpage::data\(\)](#).

Definition at line 512 of file [ipc_types](#).

15.141.4.3 io()

```
Snd_fpage L4::Ipc::Snd_fpage::io (
    unsigned long base,
    int order,
    unsigned char rights,
    l4_addr_t snd_base = 0,
    Map_type map_type = Map,
    Continue cont = Last) [inline], [static], [noexcept]
```

Construct a send item for the I/O port space.

Parameters

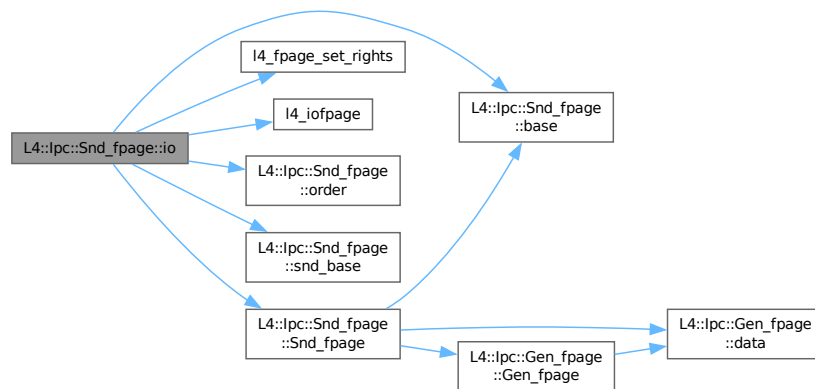
| | |
|-------------|--|
| <i>base</i> | Start of flexpage (see l4_iofpage()). |
|-------------|--|

| | |
|-----------------|--|
| <i>order</i> | Log ₂ size of flexpage (see l4_iofpage()). |
| <i>rights</i> | Permissions of flexpage (see L4_fpage_rights). |
| <i>snd_base</i> | Position in receive window in case it has a different size than 1 << order . |
| <i>map_type</i> | See Map_type . |
| <i>cont</i> | See Continue . |

Definition at line 445 of file [ipc_types](#).

References [base\(\)](#), [l4_fpage_set_rights\(\)](#), [l4_iofpage\(\)](#), [Last](#), [Map](#), [None](#), [order\(\)](#), [snd_base\(\)](#), and [Snd_fpage\(\)](#).

Here is the call graph for this function:



15.141.4.4 is_compound()

```
bool L4::Ipc::Snd_fpage::is_compound () const [inline], [noexcept]
```

Check if the item has the compound bit set, see [Continue](#).

A set compound bit means the next message item of the same type will be mapped to the same receive buffer as this message item.

Definition at line 535 of file [ipc_types](#).

15.141.4.5 local_id_received()

```
bool L4::Ipc::Snd_fpage::local_id_received () const [inline], [noexcept]
```

(Defined for return items only.) Check if a raw object flexpage has been received instead of a mapping.

If the [L4_RCV_ITEM_LOCAL_ID](#) flag has been set by the receiver, and sender and receiver are in the same task, then no mapping is done for this item and only the raw flexpage ([l4_fpage_t](#)) is received.

This function checks if this is the case and if it is an object flexpage.

The flexpage can be retrieved with [Gen_fpage::data\(\)](#).

Note

If a raw flexpage was received, but it does not have type object (see [L4_FPAGE_OBJ](#)), then this function returns `false`.

Definition at line 528 of file [ipc_types](#).

15.141.4.6 mem()

```
Snd_fpage L4::Ipc::Snd_fpage::mem (
    l4_addr_t base,
    int order,
    unsigned char rights,
    l4_addr_t snd_base = 0,
    Map_type map_type = Map,
    Cacheopt cache = None,
    Continue cont = Last) [inline], [static], [noexcept]
```

Construct a send item for the memory space.

Parameters

| | |
|-----------------|---|
| <i>base</i> | Start of flexpage (see l4_fpage()). |
| <i>order</i> | Log ₂ size of flexpage (see l4_fpage()). |
| <i>rights</i> | Permissions of flexpage (see l4_fpage()). |
| <i>snd_base</i> | Position in receive window in case it has a different size than $1 \ll order$. |
| <i>map_type</i> | See Map_type . |
| <i>cache</i> | See Cacheopt . |
| <i>cont</i> | See Continue . |

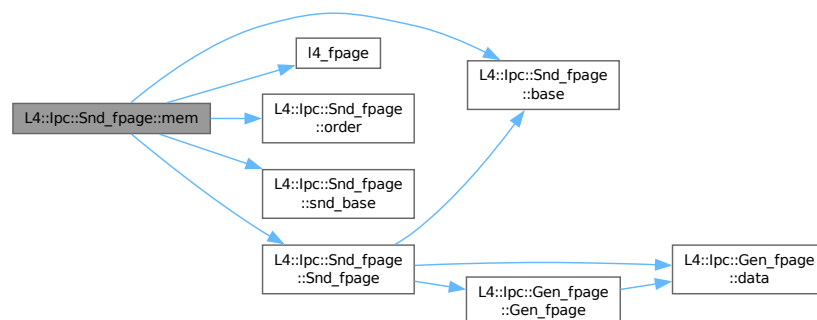
Examples

[examples/libs/l4re/streammap/server.cc](#).

Definition at line [424](#) of file [ipc_types](#).

References [base\(\)](#), [l4_fpage\(\)](#), [Last](#), [Map](#), [None](#), [order\(\)](#), [snd_base\(\)](#), and [Snd_fpage\(\)](#).

Here is the call graph for this function:



15.141.4.7 obj()

```

Snd_fpage L4::Ipc::Snd_fpage::obj (
    l4_cap_idx_t base,
    int order,
    unsigned char rights,
    l4_addr_t snd_base = 0,
    Map_type map_type = Map,
    Continue cont = Last) [inline], [static], [noexcept]

```

Construct a send item for the object space.

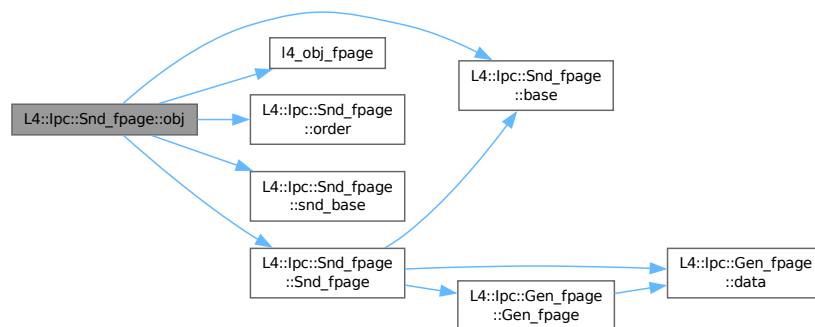
Parameters

| | |
|-----------------|---|
| <i>base</i> | Start of flexpage (see l4_obj_fpage()). |
| <i>order</i> | \log_2 size of flexpage (see l4_obj_fpage()). |
| <i>rights</i> | Permissions of flexpage (see l4_obj_fpage()). |
| <i>snd_base</i> | Position in receive window in case it has a different size than $1 \ll order$. |
| <i>map_type</i> | See Map_type . |
| <i>cont</i> | See Continue . |

Definition at line 402 of file [ipc_types](#).

References [base\(\)](#), [l4_obj_fpage\(\)](#), [Last](#), [Map](#), [None](#), [order\(\)](#), [snd_base\(\)](#), and [Snd_fpage\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

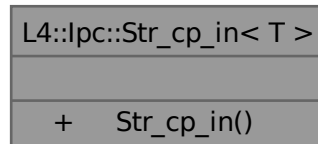
- [l4/sys/cxx/ipc_types](#)

15.142 L4::lpc::Str_cp_in< T > Class Template Reference

Abstraction for extracting a zero-terminated string from an [lpc::lstream](#).

```
#include <ipc_stream>
```

Collaboration diagram for L4::lpc::Str_cp_in< T >:



Public Member Functions

- [Str_cp_in](#) (T *v, unsigned long &size)
Create a buffer for extracting an array from an [lpc::lstream](#).

15.142.1 Detailed Description

```
template<typename T>
class L4::lpc::Str_cp_in< T >
```

Abstraction for extracting a zero-terminated string from an [lpc::lstream](#).

An instance of [Str_cp_in](#) can be used to extract a zero-terminated string an [lpc::lstream](#). The data from the received message is thereby copied to the given buffer and size is set to the number of characters found in the stream. The string is zero terminated in any circumstances. When the given buffer is smaller than the received string the last byte in the buffer will be the zero terminator. In the case the received string is shorter than the given buffer the zero termination will be placed behind the received data. This provides a zero-terminated result even in cases where the sender did not provide proper termination or in cases of too small receiver buffers.

See also

[str_cp_in\(\)](#).

Definition at line 178 of file [ipc_stream](#).

15.142.2 Constructor & Destructor Documentation

15.142.2.1 Str_cp_in()

```
template<typename T>
L4::Ipc::Str_cp_in< T >::Str_cp_in (
    T * v,
    unsigned long & size) [inline]
```

Create a buffer for extracting an array from an [lpc::Istream](#).

Parameters

| | | |
|----------------|-------------|--|
| | <i>v</i> | The buffer for string. |
| <i>in, out</i> | <i>size</i> | Input: The number of bytes available in <i>v</i> Output: The number of bytes received (including the terminator). |

Definition at line 189 of file [ipc_stream](#).

The documentation for this class was generated from the following file:

- [l4/cxx/ipc_stream](#)

15.143 L4::lpc::Varg Class Reference

Variably sized RPC argument.

```
#include <ipc_varg>
```

Collaboration diagram for L4::lpc::Varg:

| L4::lpc::Varg |
|---|
| <ul style="list-style-type: none"> + type() + length() + tag() + tag() + data() + data() + Varg() + Varg() + value() + is_of() and 8 more... |

Public Types

- using **Tag** = [l4_umword_t](#)
The data type for the tag.

Public Member Functions

- [L4_varg_type](#) [type](#) () const
- unsigned [length](#) () const
Get the size of the RPC argument.
- [Tag](#) [tag](#) () const
- void **tag** ([Tag](#) tag)
Set [Varg](#) tag (usually from message).
- void **data** (char const *d)
Set [Varg](#) to indirect data value (usually in UTCB).
- char const * [data](#) () const
- **Varg** ()=default
Make uninitialized [Varg](#).
- **Varg** ([L4_varg_type](#) t, void const *v, int len)
Make an indirect varg.
- template<typename V>
 [Va_type](#)< V >::Ret_value [value](#) () const
- template<typename T>
 bool [is_of](#) () const
- bool [is_nil](#) () const
- bool [is_of_int](#) () const
- template<typename T>
 bool [get_value](#) (typename [Va_type](#)< T >::Value *v) const
Get the value of the [Varg](#) as type T.
- template<typename T>
 void **set_value** (void const *d)
Set to indirect value of type T.
- template<typename T>
 void **set_direct_value** (T val, [L4::Types::Enable_if_t](#)< sizeof(T)<=sizeof(char const *), bool >=true)
Set to directly stored value of type T.
- template<typename T>
 Varg (T const *[data](#))
Make [Varg](#) from indirect value (pointer).
- **Varg** (char const *[data](#))
Make [Varg](#) from null-terminated string.
- template<typename T>
 Varg (T [data](#), [L4::Types::Enable_if_t](#)< sizeof(T)<=sizeof(char const *), bool >=true)
Make [Varg](#) from direct value.

15.143.1 Detailed Description

Variably sized RPC argument.

Definition at line 96 of file [ipc_varg](#).

15.143.2 Member Function Documentation

15.143.2.1 data()

```
char const * L4::Ipc::Varg::data () const [inline]
```

Returns

pointer to the data, also safe for direct data

Definition at line 123 of file [ipc_varg](#).

15.143.2.2 get_value()

```
template<typename T>
bool L4::Ipc::Varg::get_value (
    typename Va_type< T >::Value * v) const [inline]
```

Get the value of the [Varg](#) as type T.

Template Parameters

| | |
|----------|---|
| <i>T</i> | The expected type of the Varg . |
|----------|---|

Parameters

| | |
|----------|----------------------------|
| <i>v</i> | Pointer to store the value |
|----------|----------------------------|

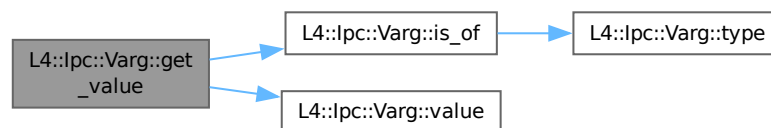
Returns

true when the [Varg](#) is of type T, false if not

Definition at line 185 of file [ipc_varg](#).

References [is_of\(\)](#), and [value\(\)](#).

Here is the call graph for this function:



15.143.2.3 is_nil()

```
bool L4::Ipc::Varg::is_nil () const [inline]
```

Returns

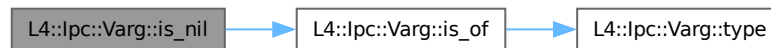
true if the [Varg](#) is of nil type.

Definition at line 172 of file [ipc_varg](#).

References [is_of\(\)](#).

Referenced by [L4::Ipc::Varg_list_ref::iterator::equals\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.143.2.4 is_of()

```
template<typename T>  
bool L4::Ipc::Varg::is_of () const [inline]
```

Returns

true if the [Varg](#) is of type T

Definition at line 169 of file [ipc_varg](#).

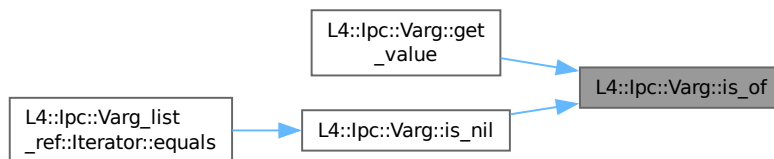
References [type\(\)](#).

Referenced by [get_value\(\)](#), and [is_nil\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**15.143.2.5 is_of_int()**

```
bool L4::Ipc::Varg::is_of_int () const [inline]
```

Returns

true if the [Varg](#) is an integer type (signed or unsigned).

Definition at line 175 of file [ipc_varg](#).

References [type\(\)](#).

Here is the call graph for this function:



15.143.2.6 length()

```
unsigned L4::Ipc::Varg::length () const [inline]
```

Get the size of the RPC argument.

Returns

The size of the RPC argument

Definition at line 114 of file [ipc_varg](#).

15.143.2.7 tag()

```
Tag L4::Ipc::Varg::tag () const [inline]
```

Returns

the tag value (the Direct_data bit masked)

Definition at line 116 of file [ipc_varg](#).

Referenced by [tag\(\)](#).

Here is the caller graph for this function:



15.143.2.8 type()

```
L4_varg_type L4::Ipc::Varg::type () const [inline]
```

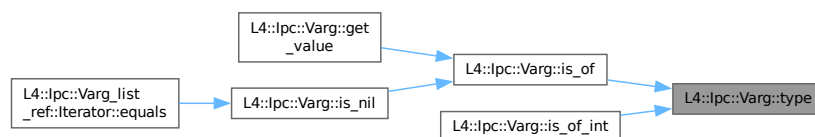
Returns

the type field of the tag

Definition at line 109 of file [ipc_varg](#).

Referenced by [is_of\(\)](#), and [is_of_int\(\)](#).

Here is the caller graph for this function:



15.143.2.9 value()

```
template<typename V>
Va_type< V >::Ret_value L4::Ipc::Varg::value () const [inline]
```

Template Parameters

| | |
|----------|---|
| V | The data type of the value to retrieve. |
|----------|---|

Precondition

The [Varg](#) must be of type *V* (otherwise the result is unpredictable).

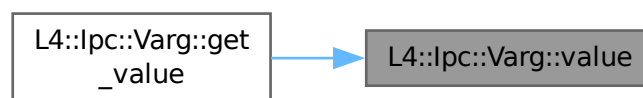
Returns

The value of the [Varg](#) as type *V*.

Definition at line [155](#) of file [ipc_varg](#).

Referenced by [get_value\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

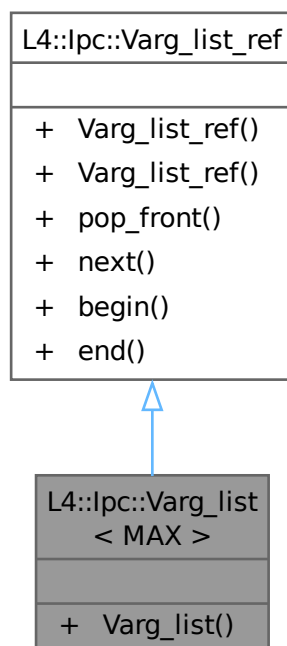
- `l4/sys/cxx/ipc_varg`

15.144 L4::lpc::Varg_list< MAX > Class Template Reference

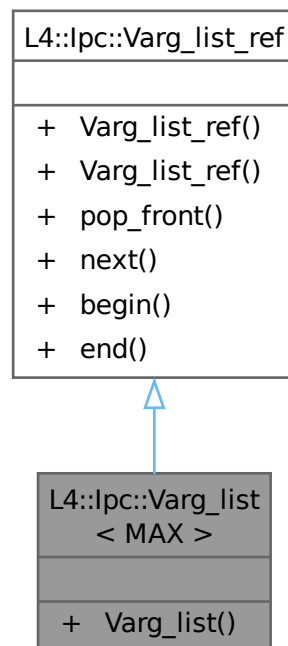
Self-contained list of variable-sized RPC parameters.

```
#include <ipc_varg>
```

Inheritance diagram for L4::lpc::Varg_list< MAX >:



Collaboration diagram for L4::lpc::Varg_list< MAX >:



Public Member Functions

- **Varg_list** ([Varg_list_ref](#) const &r)
Create a parameter list as a copy from a referencing list.

Public Member Functions inherited from [L4::lpc::Varg_list_ref](#)

- **Varg_list_ref** ()=default
Create an empty parameter list.
- [Varg_list_ref](#) (void const *start, void const *end)
Create a parameter list over a given memory region.
- [Varg](#) **pop_front** ()
Get the next parameter in the list.
- [Varg](#) **next** ()
Get the next parameter in the list.
- [Iterator](#) **begin** () const
Returns an iterator to the first [Varg](#).
- [Iterator](#) **end** () const
Returns the end of the list.

15.144.1 Detailed Description

```
template<unsigned MAX>
class L4::lpc::Varg_list< MAX >
```

Self-contained list of variable-sized RPC parameters.

Works like [Varg_list_ref](#) but contains a full copy of the data. Use this as a parameter in server functions, if the handler function needs to use the UTCB (e.g. while sending further IPC).

Definition at line [411](#) of file [ipc_varg](#).

The documentation for this class was generated from the following file:

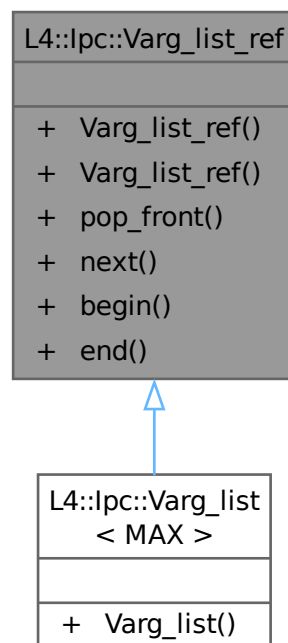
- [l4/sys/cxx/ipc_varg](#)

15.145 L4::lpc::Varg_list_ref Class Reference

List of variable-sized RPC parameters as received by the server.

```
#include <ipc_varg>
```

Inheritance diagram for L4::lpc::Varg_list_ref:



Collaboration diagram for L4::lpc::Varg_list_ref:

| L4::lpc::Varg_list_ref |
|---|
| <ul style="list-style-type: none"> + Varg_list_ref() + Varg_list_ref() + pop_front() + next() + begin() + end() |

Data Structures

- class [Iterator](#)
Iterator for Valists.

Public Member Functions

- **Varg_list_ref** ()=default
Create an empty parameter list.
- **Varg_list_ref** (void const *start, void const *end)
Create a parameter list over a given memory region.
- **Varg pop_front** ()
Get the next parameter in the list.
- **Varg next** ()
Get the next parameter in the list.
- **Iterator begin** () const
Returns an iterator to the first [Varg](#).
- **Iterator end** () const
Returns the end of the list.

15.145.1 Detailed Description

List of variable-sized RPC parameters as received by the server.

The list can be traversed exactly once using [next\(\)](#).

This is a reference list, where the returned [Varg](#) point to data in the underlying storage, conventionally the UTCB. This type should only be used in server functions when the implementation can ensure that all content is read before the UTCB is reused (e.g. for IPC), otherwise use [Varg_list](#).

Definition at line 253 of file [ipc_varg](#).

15.145.2 Constructor & Destructor Documentation

15.145.2.1 Varg_list_ref()

```
L4::Ipc::Varg_list_ref::Varg_list_ref (
    void const * start,
    void const * end) [inline]
```

Create a parameter list over a given memory region.

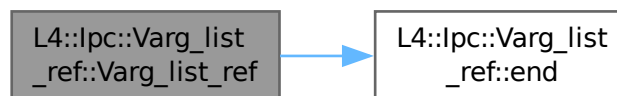
Parameters

| | |
|--------------|---|
| <i>start</i> | Pointer to start of the parameter list. |
| <i>end</i> | Pointer to end of the list (inclusive). |

Definition at line [332](#) of file [ipc_varg](#).

References [end\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

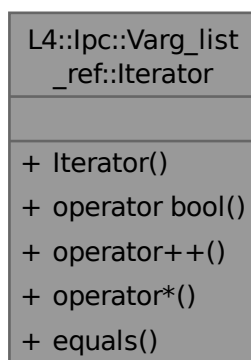
- [l4/sys/cxx/ipc_varg](#)

15.146 L4::Ipc::Varg_list_ref::Iterator Class Reference

[Iterator](#) for Valists.

```
#include <ipc_varg>
```

Collaboration diagram for L4::lpc::Varg_list_ref::Iterator:



Public Member Functions

- **Iterator** (Iter_state const &s)
Create a new iterator.
- **operator bool** () const
validity check for the iterator
- **Iterator** & **operator++** ()
increment iterator to the next arg
- **Varg** **operator*** () const
dereference the iterator, get [Varg](#)
- bool **equals** (**Iterator** const &o) const
check for equality

15.146.1 Detailed Description

[Iterator](#) for Valists.

Definition at line 338 of file [ipc_varg](#).

The documentation for this class was generated from the following file:

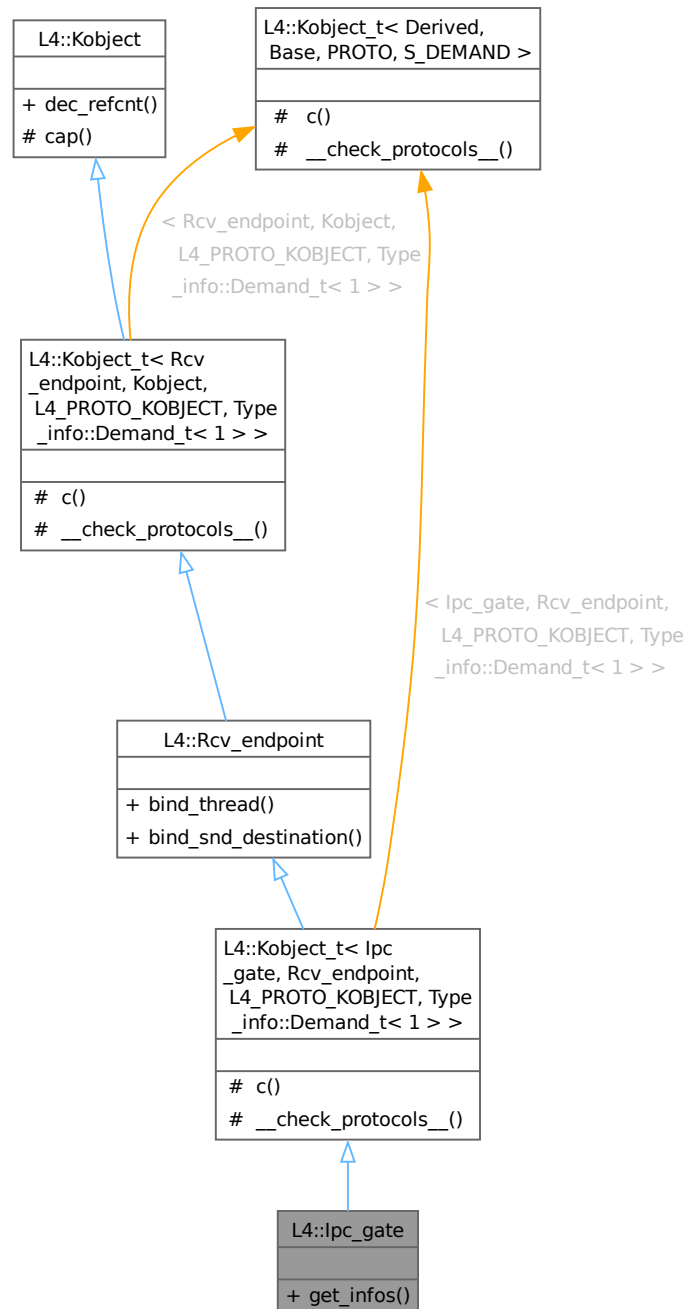
- [l4/sys/cxx/ipc_varg](#)

15.147 L4::lpc_gate Class Reference

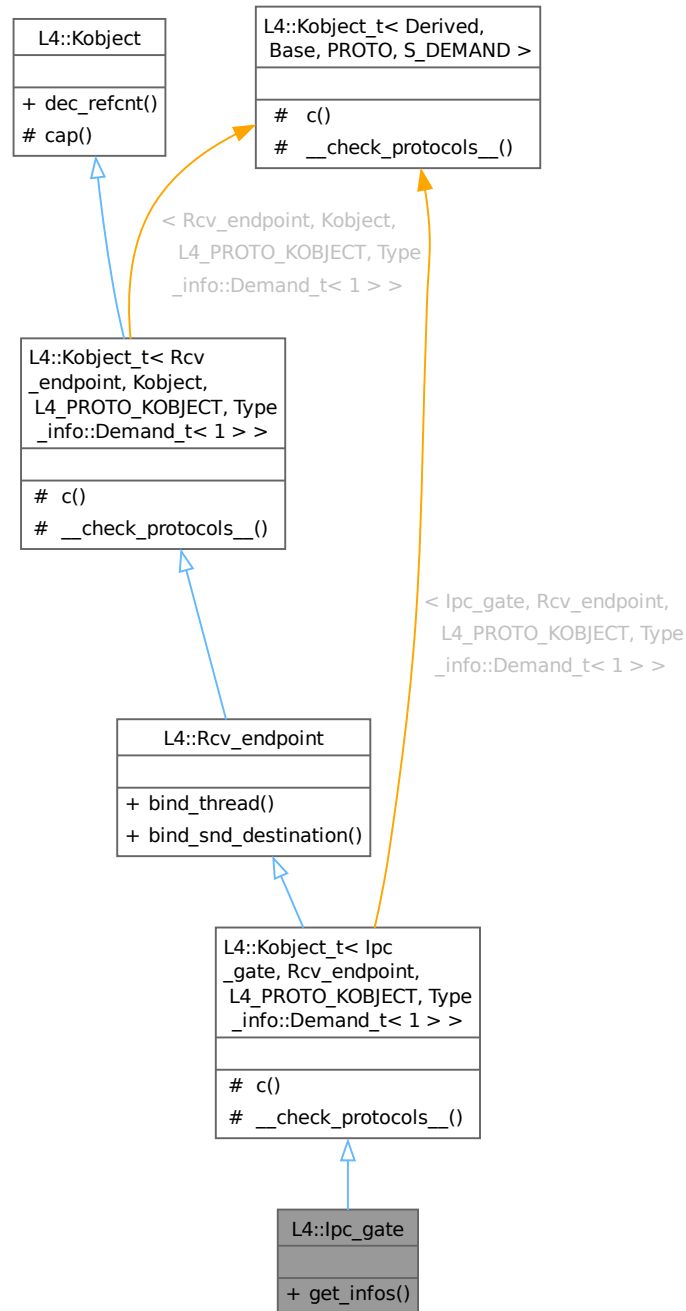
The C++ IPC gate interface, see [IPC-Gate API](#) for the C interface.

```
#include <ipc_gate>
```

Inheritance diagram for L4::lpc_gate:



Collaboration diagram for L4::lpc_gate:



Public Member Functions

- `l4_msgtag_t get_infos (l4_umword_t *label)`
Get information about the IPC-gate.

Public Member Functions inherited from L4::Rcv_endpoint

- `l4_msgtag_t bind_thread (lpc::Cap< Thread > t, l4_umword_t label)`

Bind the IPC receive endpoint to a thread.

- `l4_msgtag_t bind_snd_destination (Cap< Snd_destination > snd_dst, l4_umword_t label)`

Bind a send destination (a thread or thread group) to an IPC receive endpoint.

Public Member Functions inherited from `L4::Kobject`

- `l4_msgtag_t dec_refcnt (l4_mword_t diff, l4_utcb_t *utcb=l4_utcb())`

Decrement the in kernel reference counter for the object.

Additional Inherited Members

Protected Types inherited from

`L4::Kobject_t< lpc_gate, Rcv_endpoint, L4_PROTO_KOBJECT, Type_info::Demand_t< 1 > >`

- typedef `lpc_gate` **Class**

The target interface type (inheriting from `Kobject_t`).

- typedef `Typeid::Iface< PROTO, lpc_gate > __Iface`

The interface description for the derived class.

- typedef `Typeid::Merge_list< Typeid::Iface_list< __Iface >, typename Rcv_endpoint::__Iface_list > __Iface_list`

The list of all RPC interfaces provided directly or through inheritance.

Protected Types inherited from

`L4::Kobject_t< Rcv_endpoint, Kobject, L4_PROTO_KOBJECT, Type_info::Demand_t< 1 > >`

- typedef `Rcv_endpoint` **Class**

The target interface type (inheriting from `Kobject_t`).

- typedef `Typeid::Iface< PROTO, Rcv_endpoint > __Iface`

The interface description for the derived class.

- typedef `Typeid::Merge_list< Typeid::Iface_list< __Iface >, typename Kobject::__Iface_list > __Iface_list`

The list of all RPC interfaces provided directly or through inheritance.

Protected Member Functions inherited from

`L4::Kobject_t< lpc_gate, Rcv_endpoint, L4_PROTO_KOBJECT, Type_info::Demand_t< 1 > >`

- `L4::Cap< Class > c () const noexcept`

Get the capability to ourselves.

Protected Member Functions inherited from

`L4::Kobject_t< Rcv_endpoint, Kobject, L4_PROTO_KOBJECT, Type_info::Demand_t< 1 > >`

- `L4::Cap< Class > c () const noexcept`

Get the capability to ourselves.

Protected Member Functions inherited from L4::Kobject

- [l4_cap_idx_t cap](#) () const noexcept
Return capability selector.

Static Protected Member Functions inherited from

[L4::Kobject_t< lpc_gate, Rcv_endpoint, L4_PROTO_KOBJECT, Type_info::Demand_t< 1 > >](#)

- static void [__check_protocols__](#) () noexcept
Helper to check for protocol conflicts.

Static Protected Member Functions inherited from

[L4::Kobject_t< Rcv_endpoint, Kobject, L4_PROTO_KOBJECT, Type_info::Demand_t< 1 > >](#)

- static void [__check_protocols__](#) () noexcept
Helper to check for protocol conflicts.

15.147.1 Detailed Description

The C++ IPC gate interface, see [IPC-Gate API](#) for the C interface.

IPC gates are used to create secure communication channels between protection domains. An IPC gate can be created using the [L4::Factory](#) interface.

Depending on the permissions of the capability used, an IPC gate forwards IPC to the [L4::Thread](#) or [L4::Thread_group](#) the IPC gate is *bound* to (cf. [bind_thread\(\)](#) and [bind_snd_destination\(\)](#)). If the capability has the [L4_FPAGE_C_IPCGATE_SVR](#) permission, only IPC using a protocol different from the [L4_PROTO_KOBJECT](#) protocol is forwarded. Without the [L4_FPAGE_C_IPCGATE_SVR](#) permission, all IPC is forwarded. The latter is the usual case for a client in a client/server scenario. When not bound to a thread or thread group yet, the forwarded IPC blocks until the IPC gate is bound to a thread or thread group, or the IPC times out.

Forwarded IPC is always forwarded to the userland of the thread the IPC gate is bound to, either directly or indirectly using a thread group. That means, the [L4::Thread](#) interface of that thread is not accessible via an IPC gate. The [L4::lpc_gate](#) interface of an IPC gate is only accessible if the capability used has the [L4_FPAGE_C_IPCGATE_SVR](#) permission (cf. previous paragraph). Conversely that means, if the capability used lacks the [L4_FPAGE_C_IPCGATE_SVR](#) permission, [L4::lpc_gate](#) interface calls are forwarded to the thread or thread group the IPC gate is bound to instead of being processed by the IPC gate itself. In a client/server scenario, a client should only get IPC gate capabilities without [L4_FPAGE_C_IPCGATE_SVR](#) permission so the client cannot tamper with the IPC gate.

When binding an IPC gate to a thread or thread group, a user-defined, kernel protected, machine-word sized payload called the IPC gate's *label* is assigned to the IPC gate (note that the two least significant bits of the label must be zero; cf. [bind_thread\(\)](#) and [bind_snd_destination\(\)](#)). When a send-only IPC or call IPC is forwarded via an IPC gate, the label provided by the sender is ignored and replaced by the IPC gate's label where the two least significant bits are set to the [L4_CAP_FPAGE_S](#) and [L4_CAP_FPAGE_W](#) permissions of the capability used. The replaced label is only visible to the thread the IPC gate is bound to (or to the selected thread of the thread group the IPC gate is bound to) upon receive. However, the configured label of an IPC gate can also be queried via [get_infos\(\)](#) if the capability used has the [L4_FPAGE_C_IPCGATE_SVR](#) permission.

When deleting an IPC gate or when unbinding it from a thread or thread group, the label of IPC already in flight won't be changed. To ensure that no IPC from this IPC gate is received by a thread with an unexpected label, [L4::Thread::modify_senders\(\)](#) shall be used to change the labels of every pending IPC to that gate. This is also required if the label of an already bound IPC gate is changed. It is not necessary after binding the IPC gate to a thread or thread group for the first time.

When binding a currently bound IPC gate to a new thread or thread group, the same label should be used that was used with the old thread. Otherwise the old and the new thread need to synchronize to avoid IPC messages with unexpected labels.

Include File

```
#include <l4/sys/ipc_gate>
```

For the C interface refer to the C [IPC-Gate API](#).

See also

[Object Invocation](#)

Definition at line 85 of file [ipc_gate](#).

15.147.2 Member Function Documentation**15.147.2.1 get_infos()**

```
l4_msgtag_t L4::Ipc_gate::get_infos (
    l4_umword_t * label)
```

Get information about the IPC-gate.

Parameters

| | | |
|-----|--------------|---|
| out | <i>label</i> | The label of the IPC gate is returned here. |
|-----|--------------|---|

Returns

System call return tag.

Precondition

If the IPC gate capability used to invoke this operation does not possess the [L4_FPAGE_C_IPCGATE_SVR](#) right, the kernel will not perform the operation. Instead, the underlying IPC message will be forwarded to the thread or thread group the IPC gate is bound to, blocking the caller if the IPC gate is not bound yet.

References [get_infos\(\)](#).

Referenced by [get_infos\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

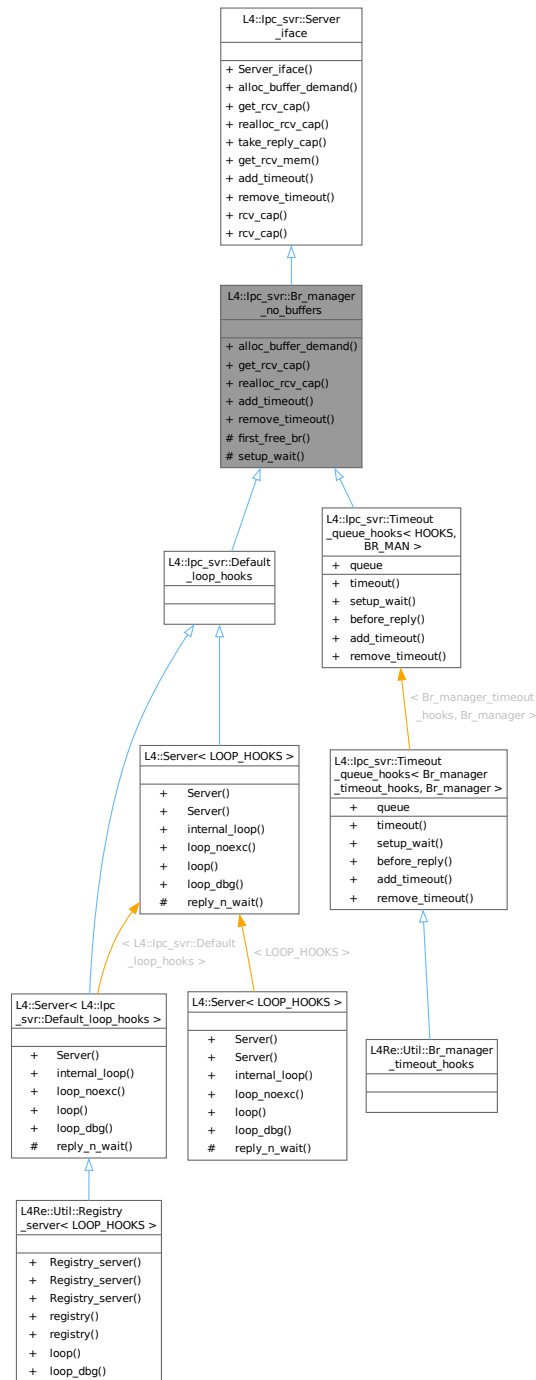
- [l4/sys/ipc_gate](#)

15.148 L4::lpc_svr::Br_manager_no_buffers Class Reference

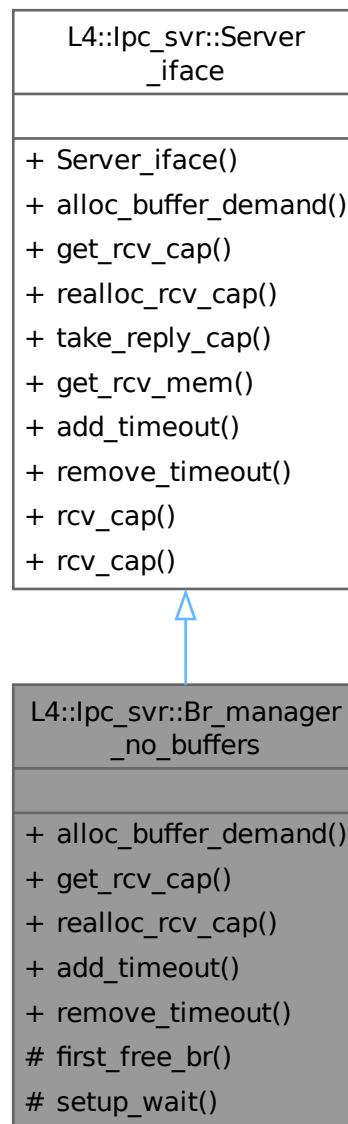
Empty implementation of [Server_iface](#).

```
#include <ipc_server_loop>
```

Inheritance diagram for L4::lpc_svr::Br_manager_no_buffers:



Collaboration diagram for L4::lpc_svr::Br_manager_no_buffers:



Public Member Functions

- int `alloc_buffer_demand` (`Demand` const &demand) override
Tells the server to allocate buffers for the given demand.
- L4::Cap< void > `get_rcv_cap` (int) const override
Returns L4::Cap< void> ::Invalid, we have no buffer management.
- int `realloc_rcv_cap` (int) override
Returns -L4_ENOMEM, we have no buffer management.
- int `add_timeout` (`Timeout` *, `l4_kernel_clock_t`) override
Returns -L4_ENOSYS, we have no timeout queue.
- int `remove_timeout` (`Timeout` *) override
Returns -L4_ENOSYS, we have no timeout queue.

Public Member Functions inherited from [L4::lpc_svr::Server_iface](#)

- **Server_iface** ()
Make a server interface.
- virtual [cxx::Result](#)< [L4::Reply_cap](#) > [take_reply_cap](#) () noexcept
Take the currently used reply capability.
- virtual [cxx::Result](#)< [Mem_window](#) > [get_rcv_mem](#) () noexcept
Take the current memory receive window.
- template<typename T>
[L4::Cap](#)< T > [rcv_cap](#) (int index) const
Get given receive buffer as typed capability.
- [L4::Cap](#)< void > [rcv_cap](#) (int index) const
Get receive cap with the given index as generic (void) type.

Protected Member Functions

- unsigned **first_free_br** () const
Returns 1 as first free buffer.
- void **setup_wait** ([l4_utcb_t](#) *utcb, [L4::lpc_svr::Reply_mode](#))
Setup wait function for the server loop ([Server<>](#)).

Additional Inherited Members

Public Types inherited from [L4::lpc_svr::Server_iface](#)

- using **Demand** = [L4::Type_info::Demand](#)
Data type expressing server-side demand for receive buffers.

15.148.1 Detailed Description

Empty implementation of [Server_iface](#).

This implementation of [Server_iface](#) provides no buffer or timeout management at all it just returns errors for all calls that express other than empty demands. However, this may be useful for very simple servers that serve simple server objects only.

Definition at line 233 of file [ipc_server_loop](#).

15.148.2 Member Function Documentation

15.148.2.1 alloc_buffer_demand()

```
int L4::Ipc_svr::Br_manager_no_buffers::alloc_buffer_demand (
    Demand const & demand) [inline], [override], [virtual]
```

Tells the server to allocate buffers for the given demand.

Parameters

| | |
|---------------|--|
| <i>demand</i> | The total server-side demand of receive buffers needed for a given interface, see Demand . |
|---------------|--|

This function is not called by user applications directly. Usually the server implementation or the registry implementation calls this function whenever a new object is registered at the server.

Returns

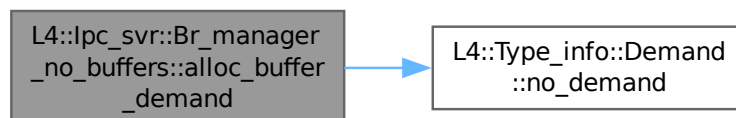
success (0) if demand is empty, -L4_ENOMEM else.

Implements [L4::lpc_svr::Server_iface](#).

Definition at line 240 of file [ipc_server_loop](#).

References [L4_ENOMEM](#), [L4_EOK](#), and [L4::Type_info::Demand::no_demand\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

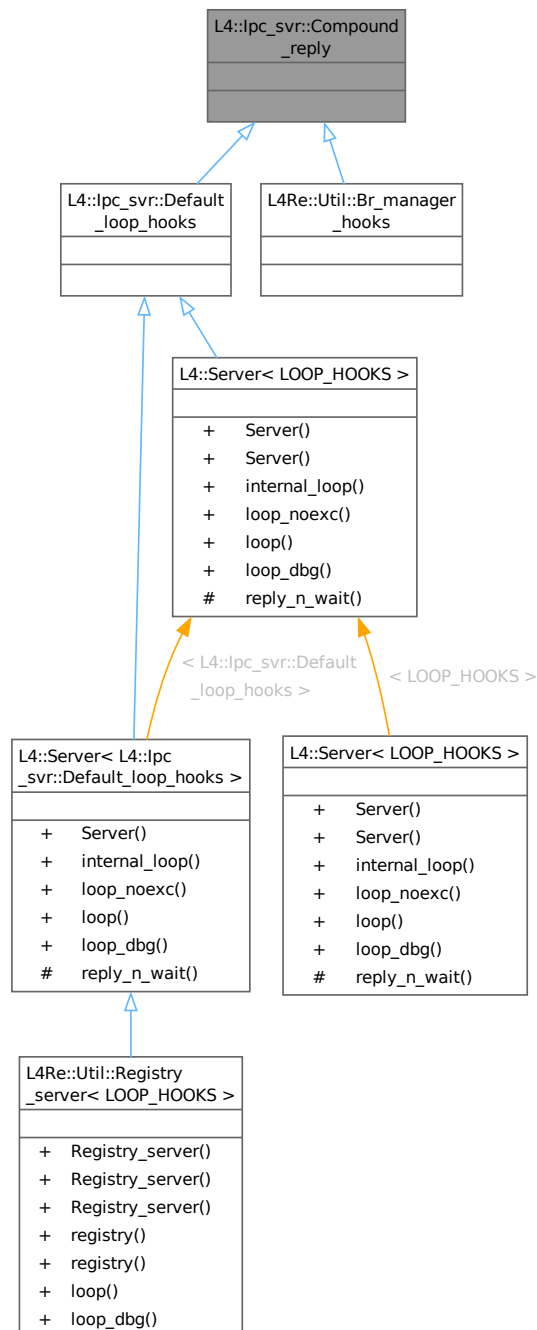
- `l4/sys/cxx/ipc_server_loop`

15.149 L4::lpc_svr::Compound_reply Struct Reference

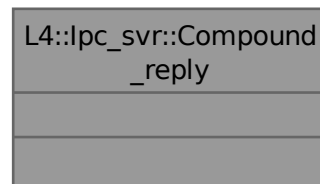
Mix in for LOOP_HOOKS to always use compound reply and wait.

```
#include <ipc_server_loop>
```

Inheritance diagram for L4::lpc_svr::Compound_reply:



Collaboration diagram for L4::lpc_svr::Compound_reply:



15.149.1 Detailed Description

Mix in for LOOP_HOOKS to always use compound reply and wait.

Definition at line 73 of file [ipc_server_loop](#).

The documentation for this struct was generated from the following file:

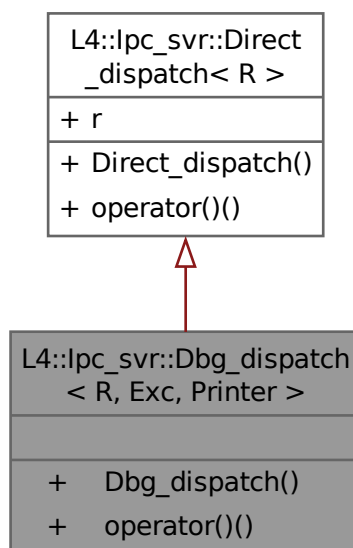
- l4/sys/cxx/ipc_server_loop

15.150 L4::lpc_svr::Dbg_dispatch< R, Exc, Printer > Struct Template Reference

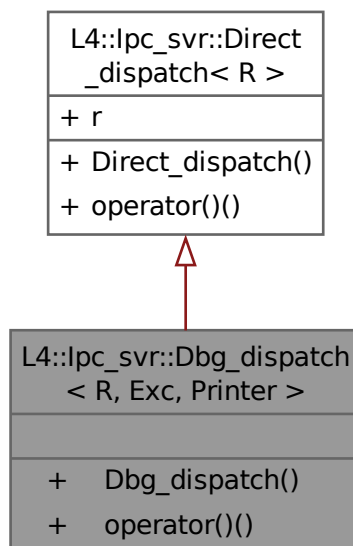
Dispatch helper that, in addition to what [Exc_dispatch](#) does, prints exception messages.

```
#include <ipc_server_loop>
```

Inheritance diagram for L4::lpc_svr::Dbg_dispatch< R, Exc, Printer >:



Collaboration diagram for L4::lpc_svr::Dbg_dispatch< R, Exc, Printer >:



Public Member Functions

- **Dbg_dispatch** (R [r](#), Printer p)

Make an exception handling dispatcher.

- [l4_msgtag_t](#) **operator()** ([l4_msgtag_t](#) tag, [l4_umword_t](#) obj, [l4_utcb_t](#) *utcb)

Dispatch the call via [Direct_dispatch<R>\(\)](#), handle exceptions, and print the exception message.

15.150.1 Detailed Description

```
template<typename R, typename Exc, typename Printer>
struct L4::lpc_svr::Dbg_dispatch< R, Exc, Printer >
```

Dispatch helper that, in addition to what [Exc_dispatch](#) does, prints exception messages.

Template Parameters

| | |
|----------------|---|
| <i>R</i> | Data type of the registry used for dispatching to objects. |
| <i>Exc</i> | Data type of the exceptions that shall be caught. This data type must provide a member <code>err_no()</code> that returns the negative integer (int) error code for the exception. In addition, methods <code>str()</code> and <code>extra_str()</code> are required that return a c-string describing the error. |
| <i>Printer</i> | A type that provides a <code>printf()</code> member that is used (with the usual format string syntax) to print error messages. |

This dispatcher wraps `Direct_dispatch<R>` with a try-catch (Exc).

Definition at line 184 of file [ipc_server_loop](#).

The documentation for this struct was generated from the following file:

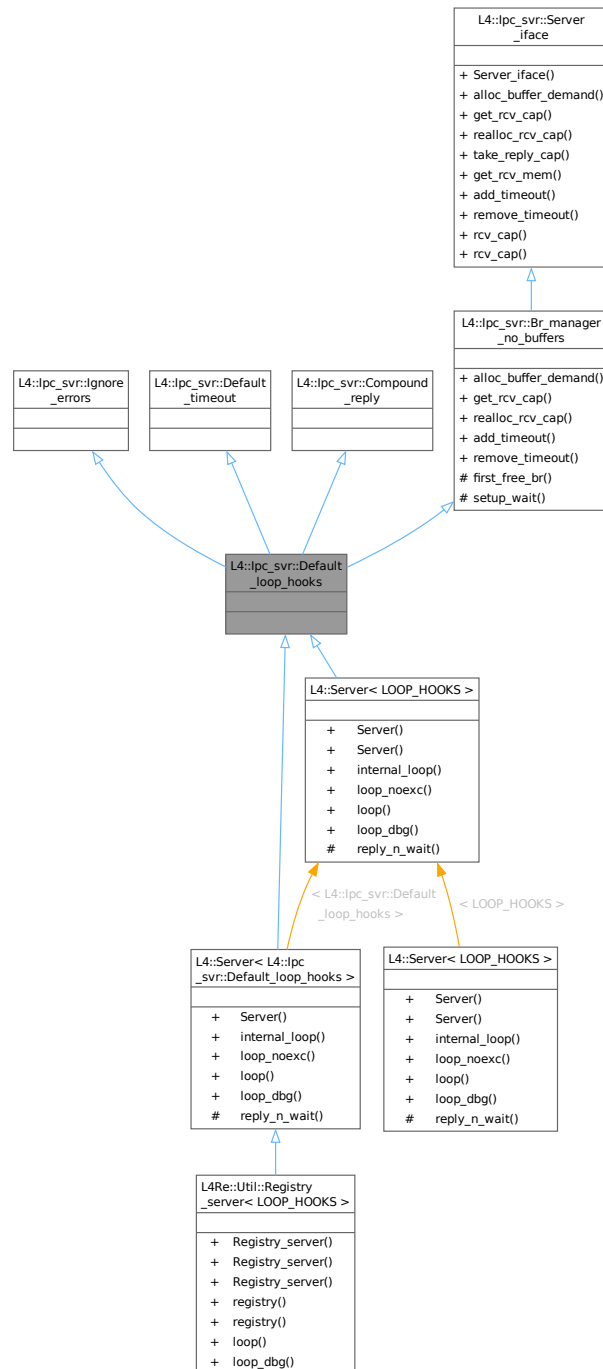
- `l4/sys/cxx/ipc_server_loop`

15.151 L4::lpc_svr::Default_loop_hooks Struct Reference

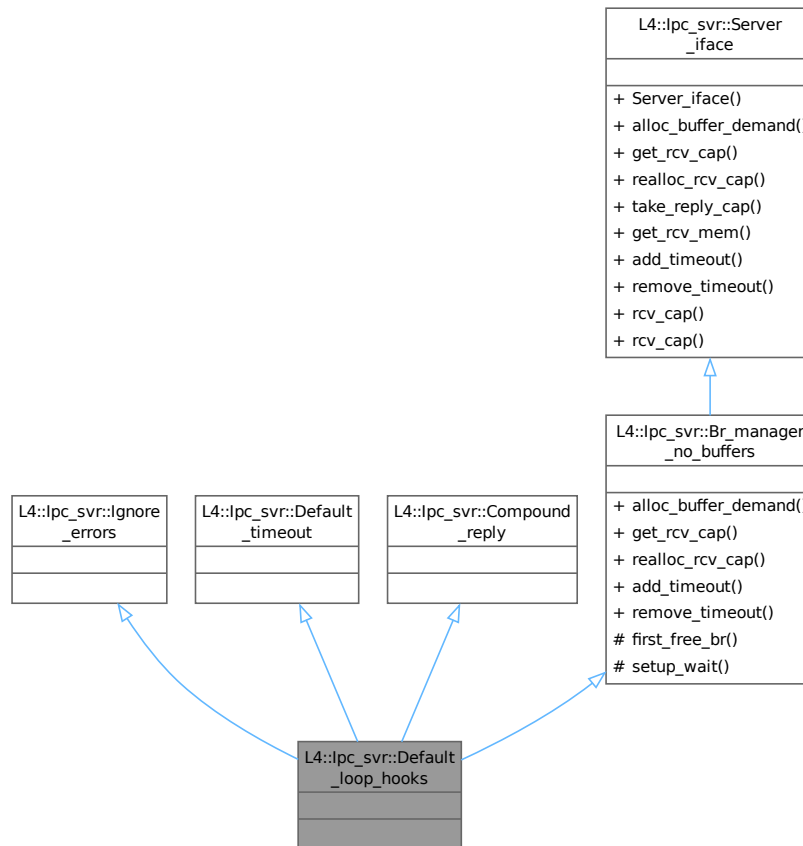
Default LOOP_HOOKS.

```
#include <ipc_server_loop>
```

Inheritance diagram for L4::ipc_svr::Default_loop_hooks:



Collaboration diagram for L4::lpc_svr::Default_loop_hooks:



Additional Inherited Members

Public Types inherited from L4::lpc_svr::Server_iface

- using **Demand** = L4::Type_info::Demand
Data type expressing server-side demand for receive buffers.

Public Member Functions inherited from L4::lpc_svr::Br_manager_no_buffers

- int **alloc_buffer_demand** (Demand const &demand) override
Tells the server to allocate buffers for the given demand.
- L4::Cap< void > **get_rcv_cap** (int) const override
Returns L4::Cap<void>::Invalid, we have no buffer management.
- int **realloc_rcv_cap** (int) override
Returns -L4_ENOMEM, we have no buffer management.
- int **add_timeout** (Timeout *, l4_kernel_clock_t) override
Returns -L4_ENOSYS, we have no timeout queue.
- int **remove_timeout** (Timeout *) override
Returns -L4_ENOSYS, we have no timeout queue.

Public Member Functions inherited from [L4::lpc_svr::Server_iface](#)

- **Server_iface** ()
Make a server interface.
- virtual [cxx::Result](#)< [L4::Reply_cap](#) > **take_reply_cap** () noexcept
Take the currently used reply capability.
- virtual [cxx::Result](#)< [Mem_window](#) > **get_rcv_mem** () noexcept
Take the current memory receive window.
- template<typename T>
[L4::Cap](#)< T > **rcv_cap** (int index) const
Get given receive buffer as typed capability.
- [L4::Cap](#)< void > **rcv_cap** (int index) const
Get receive cap with the given index as generic (void) type.

Protected Member Functions inherited from [L4::lpc_svr::Br_manager_no_buffers](#)

- unsigned **first_free_br** () const
Returns 1 as first free buffer.
- void **setup_wait** ([l4_utcb_t](#) *utcb, [L4::lpc_svr::Reply_mode](#))
Setup wait function for the server loop ([Server](#)<>).

15.151.1 Detailed Description

Default LOOP_HOOKS.

Combination of [Ignore_errors](#), [Default_timeout](#), [Compound_reply](#), and [Br_manager_no_buffers](#).

Definition at line 285 of file [ipc_server_loop](#).

The documentation for this struct was generated from the following file:

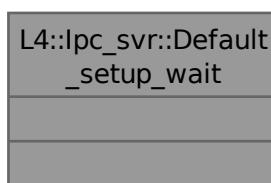
- [l4/sys/cxx/ipc_server_loop](#)

15.152 [L4::lpc_svr::Default_setup_wait](#) Struct Reference

Mix in for LOOP_HOOKS for setup_wait no op.

```
#include <ipc_server_loop>
```

Collaboration diagram for [L4::lpc_svr::Default_setup_wait](#):



15.152.1 Detailed Description

Mix in for LOOP_HOOKS for setup_wait no op.

Definition at line 84 of file [ipc_server_loop](#).

The documentation for this struct was generated from the following file:

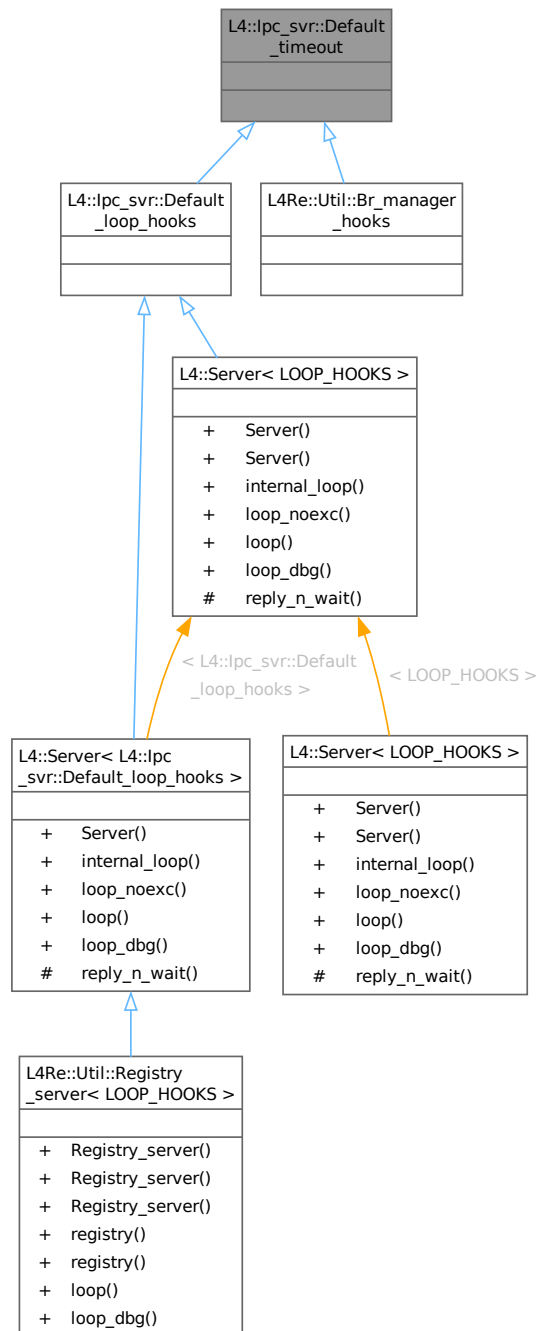
- l4/sys/cxx/ipc_server_loop

15.153 L4::lpc_svr::Default_timeout Struct Reference

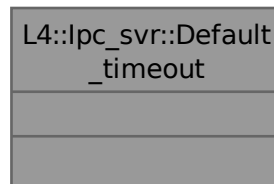
Mix in for LOOP_HOOKS to use a 0 send and an infinite receive timeout.

```
#include <ipc_server_loop>
```

Inheritance diagram for L4::lpc_svr::Default_timeout:



Collaboration diagram for L4::lpc_svr::Default_timeout:



15.153.1 Detailed Description

Mix in for LOOP_HOOKS to use a 0 send and an infinite receive timeout.

Definition at line 65 of file [ipc_server_loop](#).

The documentation for this struct was generated from the following file:

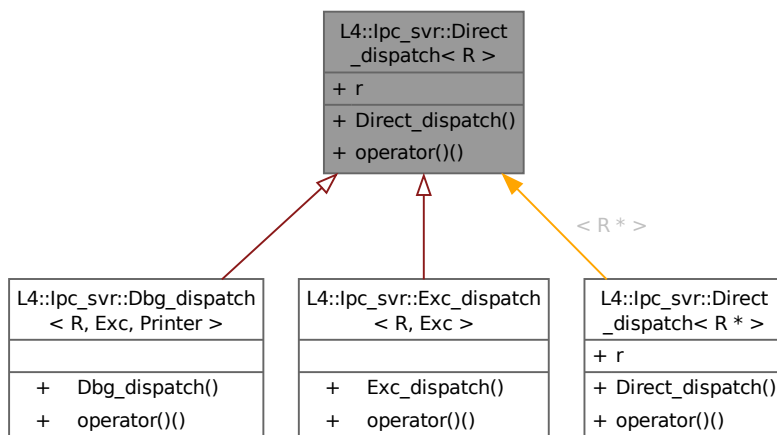
- l4/sys/cxx/ipc_server_loop

15.154 L4::lpc_svr::Direct_dispatch< R > Struct Template Reference

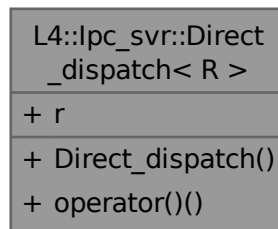
Direct dispatch helper, for forwarding dispatch calls to a registry *R*.

```
#include <ipc_server_loop>
```

Inheritance diagram for L4::lpc_svr::Direct_dispatch< R >:



Collaboration diagram for L4::lpc_svr::Direct_dispatch< R >:



Public Member Functions

- **Direct_dispatch** (R &r)
Make a direct dispatcher.
- **l4_msgtag_t operator()** (l4_msgtag_t tag, l4_umword_t obj, l4_utcb_t *utcb)
call operator forwarding to r.dispatch()

Data Fields

- R & r
stores a reference to the registry object

15.154.1 Detailed Description

```
template<typename R>
struct L4::lpc_svr::Direct_dispatch< R >
```

Direct dispatch helper, for forwarding dispatch calls to a registry *R*.

Template Parameters

| | |
|----------|---|
| <i>R</i> | Data type of the registry that is used for dispatching to different server objects, usually based on the protected IPC label. |
|----------|---|

Definition at line 95 of file [ipc_server_loop](#).

The documentation for this struct was generated from the following file:

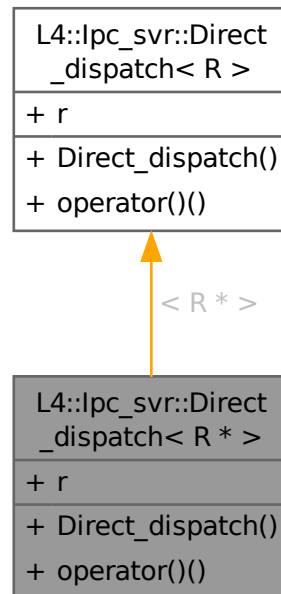
- l4/sys/cxx/ipc_server_loop

15.155 L4::lpc_svr::Direct_dispatch< R * > Struct Template Reference

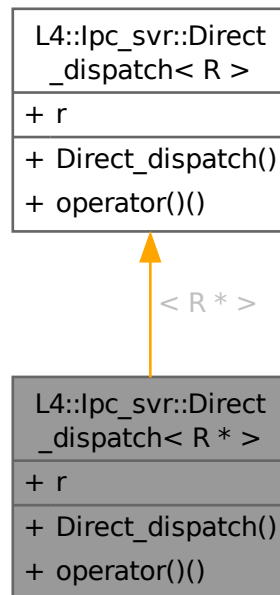
Direct dispatch helper, for forwarding dispatch calls via a pointer to a registry *R*.

```
#include <ipc_server_loop>
```

Inheritance diagram for L4::lpc_svr::Direct_dispatch< R * >:



Collaboration diagram for L4::lpc_svr::Direct_dispatch< R * >:



Public Member Functions

- **Direct_dispatch** (R *r)
Make a direct dispatcher.
- **l4_msgtag_t operator()** (l4_msgtag_t tag, l4_umword_t obj, l4_utcb_t *utcb)
call operator forwarding to r->dispatch()

Data Fields

- R * **r**
stores a pointer to the registry object

15.155.1 Detailed Description

```
template<typename R>
struct L4::lpc_svr::Direct_dispatch< R * >
```

Direct dispatch helper, for forwarding dispatch calls via a pointer to a registry *R*.

Template Parameters

| | |
|----------|---|
| <i>R</i> | Data type of the registry that is used for dispatching to different server objects, usually based on the protected IPC label. |
|----------|---|

Definition at line 116 of file [ipc_server_loop](#).

The documentation for this struct was generated from the following file:

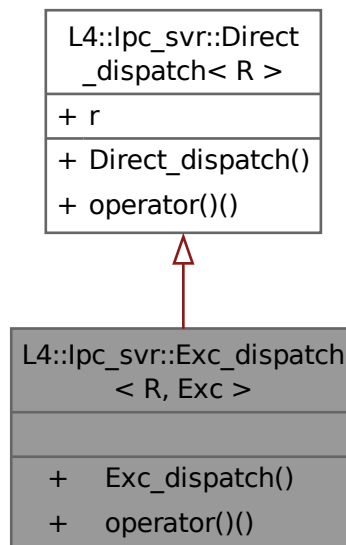
- l4/sys/cxx/ipc_server_loop

15.156 L4::lpc_svr::Exc_dispatch< R, Exc > Struct Template Reference

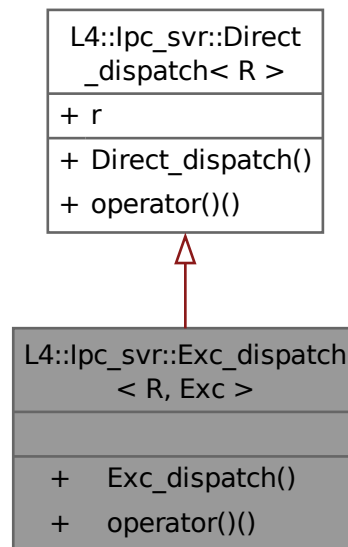
Dispatch helper wrapping try {} catch {} around the dispatch call.

```
#include <ipc_server_loop>
```

Inheritance diagram for L4::lpc_svr::Exc_dispatch< R, Exc >:



Collaboration diagram for `L4::ipc_svr::Exc_dispatch< R, Exc >`:



Public Member Functions

- **Exc_dispatch** (`R r`)
Make an exception handling dispatcher.
- `l4_msgtag_t operator()` (`l4_msgtag_t tag`, `l4_umword_t obj`, `l4_utcb_t *utcb`)
Dispatch the call via [Direct_dispatch<R>\(\)](#) and handle exceptions.

15.156.1 Detailed Description

```
template<typename R, typename Exc>
struct L4::ipc_svr::Exc_dispatch< R, Exc >
```

Dispatch helper wrapping `try {} catch {}` around the dispatch call.

Template Parameters

| | |
|------------|--|
| <i>R</i> | Data type of the registry used for dispatching to objects. |
| <i>Exc</i> | Data type of the exceptions that shall be caught. This data type must provide a member <code>err_no()</code> that returns the negative integer (int) error code for the exception. |

This dispatcher wraps `Direct_dispatch<R>` with a try-catch (`Exc`).

Definition at line 140 of file [ipc_server_loop](#).

The documentation for this struct was generated from the following file:

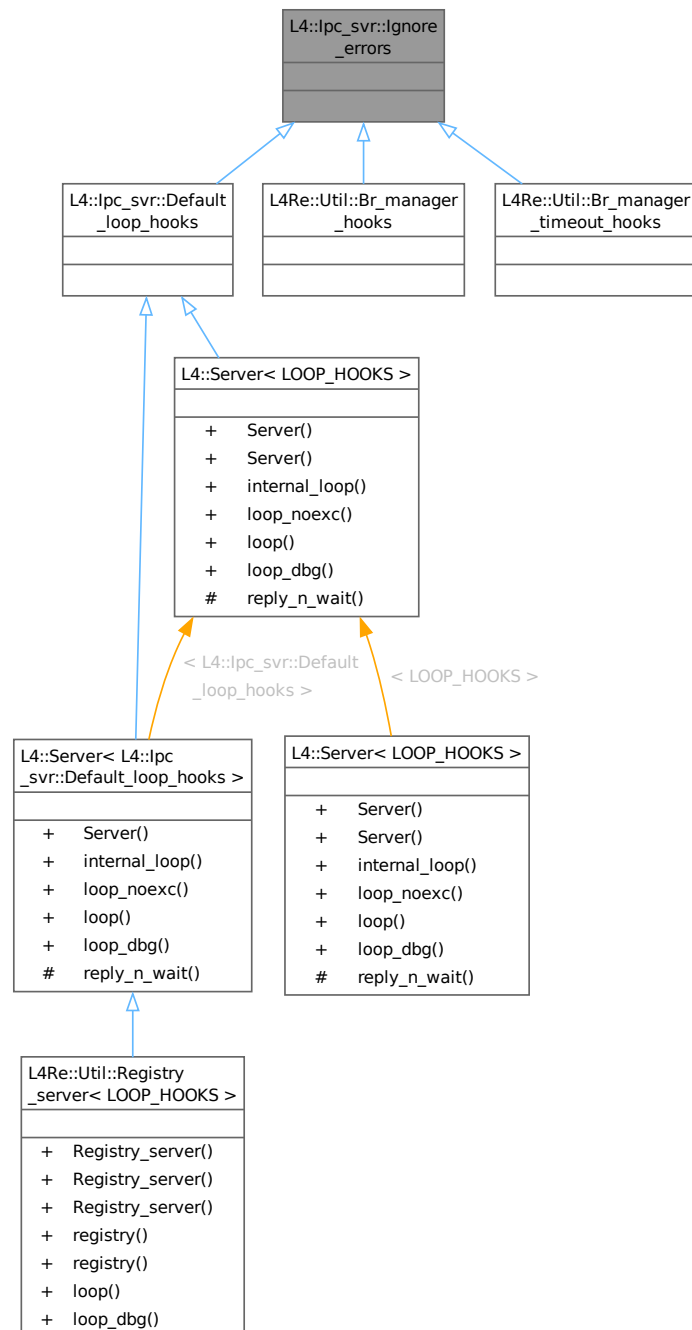
- `l4/sys/cxx/ipc_server_loop`

15.157 L4::lpc_svr::Ignore_errors Struct Reference

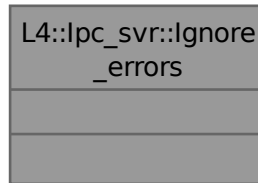
Mix in for LOOP_HOOKS to ignore IPC errors.

```
#include <ipc_server_loop>
```

Inheritance diagram for L4::lpc_svr::Ignore_errors:



Collaboration diagram for L4::lpc_svr::Ignore_errors:



15.157.1 Detailed Description

Mix in for LOOP_HOOKS to ignore IPC errors.

Definition at line 57 of file [ipc_server_loop](#).

The documentation for this struct was generated from the following file:

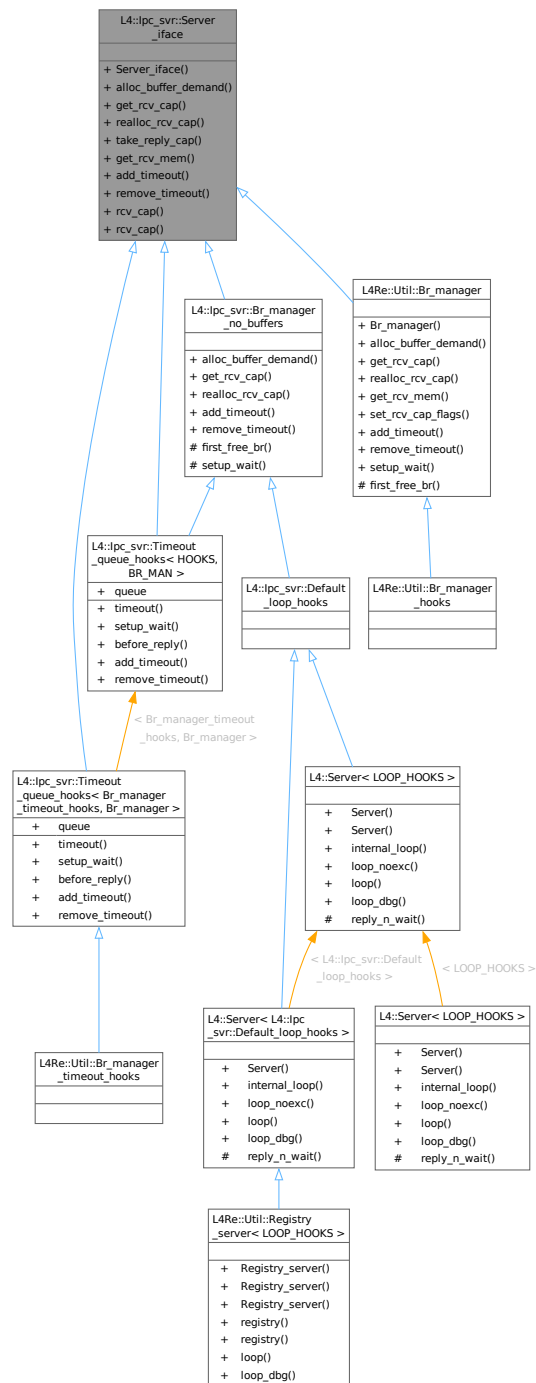
- `l4/sys/cxx/ipc_server_loop`

15.158 L4::lpc_svr::Server_iface Class Reference

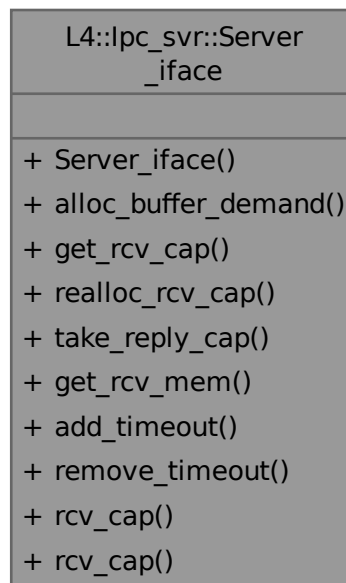
Interface for server-loop related functions.

```
#include <ipc_epiface>
```


Inheritance diagram for L4::lpc_svr::Server_iface:



Collaboration diagram for L4::lpc_svr::Server_iface:



Data Structures

- class [Mem_window](#)
Memory receive window.

Public Types

- using **Demand** = [L4::Type_info::Demand](#)
Data type expressing server-side demand for receive buffers.

Public Member Functions

- **Server_iface** ()
Make a server interface.
- virtual int [alloc_buffer_demand](#) ([Demand](#) const &demand)=0
Tells the server to allocate buffers for the given demand.
- virtual [L4::Cap](#)< void > [get_rcv_cap](#) (int index) const =0
Get capability slot allocated to the given receive buffer.
- virtual int [realloc_rcv_cap](#) (int index)=0
Allocate a new capability for the given receive buffer.
- virtual [cxx::Result](#)< [L4::Reply_cap](#) > [take_reply_cap](#) () noexcept
Take the currently used reply capability.
- virtual [cxx::Result](#)< [Mem_window](#) > [get_rcv_mem](#) () noexcept
Take the current memory receive window.

- virtual int [add_timeout](#) ([Timeout](#) *timeout, [l4_kernel_clock_t](#) time)=0
Add a timeout to the server internal timeout queue.
- virtual int [remove_timeout](#) ([Timeout](#) *timeout)=0
Remove the given timeout from the timer queue.
- template<typename T>
[L4::Cap](#)< T > [rcv_cap](#) (int index) const
Get given receive buffer as typed capability.
- [L4::Cap](#)< void > [rcv_cap](#) (int index) const
Get receive cap with the given index as generic (void) type.

15.158.1 Detailed Description

Interface for server-loop related functions.

This interface provides access to high-level server-loop related functions, such as management of receive buffers and timeouts.

Definition at line 37 of file [ipc_epiface](#).

15.158.2 Member Function Documentation

15.158.2.1 add_timeout()

```
virtual int L4::Ipc_svr::Server_iface::add_timeout (
    Timeout * timeout,
    l4\_kernel\_clock\_t time) [pure virtual]
```

Add a timeout to the server internal timeout queue.

Parameters

| | |
|----------------|--|
| <i>timeout</i> | The timeout object to register. |
| <i>time</i> | The time (absolute) at which the timeout shall expire. |

Precondition

timeout must not be in any queue.

Returns

0 on success, 1 if timeout is already expired, < 0 on error.

Implemented in [L4::lpc_svr::Br_manager_no_buffers](#), [L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >](#), [L4::lpc_svr::Timeout_queue_hooks< Br_manager_timeout_hooks, Br_manager >](#), and [L4Re::Util::Br_manager](#).

15.158.2.2 alloc_buffer_demand()

```
virtual int L4::Ipc_svr::Server_iface::alloc_buffer_demand (
    Demand const & demand) [pure virtual]
```

Tells the server to allocate buffers for the given demand.

Parameters

| | |
|---------------|--|
| <i>demand</i> | The total server-side demand of receive buffers needed for a given interface, see Demand . |
|---------------|--|

This function is not called by user applications directly. Usually the server implementation or the registry implementation calls this function whenever a new object is registered at the server.

Implemented in [L4::lpc_svr::Br_manager_no_buffers](#), and [L4Re::Util::Br_manager](#).

15.158.2.3 get_rcv_cap()

```
virtual L4::Cap< void > L4::Ipc_svr::Server_iface::get_rcv_cap (
    int index) const [pure virtual]
```

Get capability slot allocated to the given receive buffer.

Parameters

| | |
|--------------|---|
| <i>index</i> | The receive buffer index of the expected capability argument ($0 \leq \text{index} < \text{caps}$ registered with alloc_buffer_demand()). |
|--------------|---|

Precondition

$0 \leq \text{index} < \text{caps}$ registered with [alloc_buffer_demand\(\)](#)

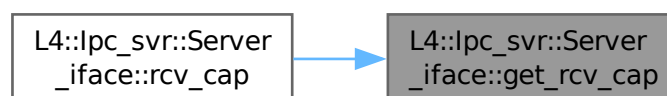
Returns

Capability slot currently allocated to the given receive buffer.

Implemented in [L4::lpc_svr::Br_manager_no_buffers](#), and [L4Re::Util::Br_manager](#).

Referenced by [rcv_cap\(\)](#).

Here is the caller graph for this function:



15.158.2.4 get_rcv_mem()

```
virtual cxx::Result< Mem\_window > L4::Ipc_svr::Server_iface::get_rcv_mem () [inline], [virtual],
[noexcept]
```

Take the current memory receive window.

The caller takes the ownership of the memory receive window. A new receive window will be allocated for the next call.

Attention

The caller has to ensure that the received IPC call actually mapped some memory flexpage. Still, the IPC client can revoke the pages at any time. Thus, any access into the memory window could create unresolved page faults.

Reimplemented in [L4Re::Util::Br_manager](#).

Definition at line 195 of file [ipc_epiface](#).

References [L4_ENOSYS](#).

15.158.2.5 rcv_cap() [1/2]

```
L4::Cap< void > L4::Ipc_svr::Server_iface::rcv_cap (
    int index) const [inline]
```

Get receive cap with the given index as generic (void) type.

Parameters

| | |
|--------------|--|
| <i>index</i> | The index of the cap receive buffer of the expected capability. (0 ≤ index < caps registered with alloc_buffer_demand() .) |
|--------------|--|

Returns

Capability slot currently allocated to the given capability buffer.

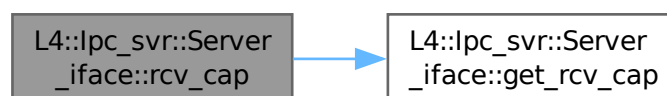
Note

This is a convenience wrapper for [get_rcv_cap\(\)](#).

Definition at line 238 of file [ipc_epiface](#).

References [get_rcv_cap\(\)](#).

Here is the call graph for this function:



15.158.2.6 rcv_cap() [2/2]

```
template<typename T>
L4::Cap< T > L4::Ipc_svr::Server_iface::rcv_cap (
    int index) const [inline]
```

Get given receive buffer as typed capability.

See also

[get_rcv_cap\(\)](#)

Parameters

| | |
|--------------|--|
| <i>index</i> | The receive buffer index of the expected capability argument. ($0 \leq \text{index} < \text{caps}$ registered with alloc_buffer_demand() .) |
|--------------|--|

Precondition

$0 \leq \text{index} < \text{caps}$ registered with [alloc_buffer_demand\(\)](#)

Returns

Capability slot currently allocated to the given receive buffer.

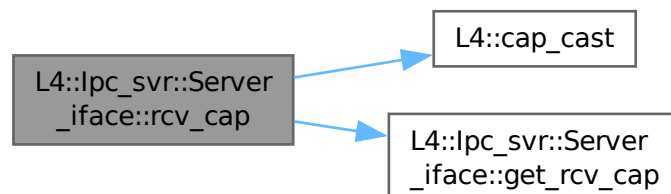
Note

This is a convenience wrapper for [get_rcv_cap\(\)](#) to avoid [L4::cap_cast<>\(\)](#).

Definition at line 226 of file [ipc_epiface](#).

References [L4::cap_cast\(\)](#), and [get_rcv_cap\(\)](#).

Here is the call graph for this function:



15.158.2.7 realloc_rcv_cap()

```
virtual int L4::Ipc_svr::Server_iface::realloc_rcv_cap (
    int index) [pure virtual]
```

Allocate a new capability for the given receive buffer.

Parameters

| | |
|--------------|---|
| <i>index</i> | The receive buffer index of the expected capability argument ($0 \leq \text{index} < \text{caps}$ registered with alloc_buffer_demand()). |
|--------------|---|

Precondition

$0 \leq \text{index} < \text{caps}$ registered with [alloc_buffer_demand\(\)](#)

Returns

0 on success, < 0 on error.

Implemented in [L4::lpc_svr::Br_manager_no_buffers](#), and [L4Re::Util::Br_manager](#).

15.158.2.8 remove_timeout()

```
virtual int L4::Ipc_svr::Server_iface::remove_timeout (
    Timeout * timeout) [pure virtual]
```

Remove the given timeout from the timer queue.

Parameters

| | |
|----------------|-------------------------------|
| <i>timeout</i> | The timeout object to remove. |
|----------------|-------------------------------|

Returns

0 on success, < 0 on error.

Implemented in [L4::lpc_svr::Br_manager_no_buffers](#), [L4::lpc_svr::Timeout_queue_hooks<HOOKS, BR_MAN>](#), [L4::lpc_svr::Timeout_queue_hooks<Br_manager_timeout_hooks, Br_manager>](#), and [L4Re::Util::Br_manager](#).

15.158.2.9 take_reply_cap()

```
virtual cxx::Result< L4::Reply_cap > L4::Ipc_svr::Server_iface::take_reply_cap () [inline],
[virtual], [noexcept]
```

Take the currently used reply capability.

The caller takes ownership of the reply capability. A new reply capability is allocated for the next call.

Definition at line 181 of file [ipc_epiface](#).

References [L4_ENOSYS](#).

The documentation for this class was generated from the following file:

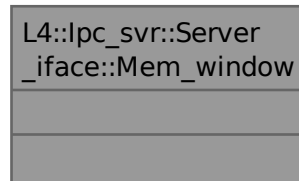
- [l4/sys/cxx/ipc_epiface](#)

15.159 L4::lpc_svr::Server_iface::Mem_window Class Reference

Memory receive window.

```
#include <ipc_epiface>
```

Collaboration diagram for L4::lpc_svr::Server_iface::Mem_window:



15.159.1 Detailed Description

Memory receive window.

Represents a region in the virtual address space in which an IPC call *potentially* mapped pages.

Definition at line 50 of file [ipc_epiface](#).

The documentation for this class was generated from the following file:

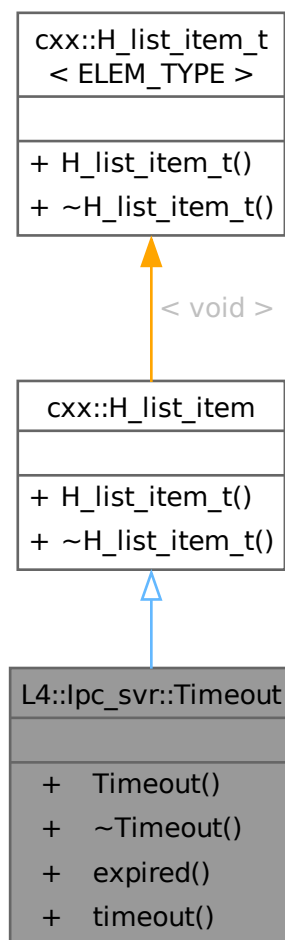
- `l4/sys/cxx/ipc_epiface`

15.160 L4::lpc_svr::Timeout Class Reference

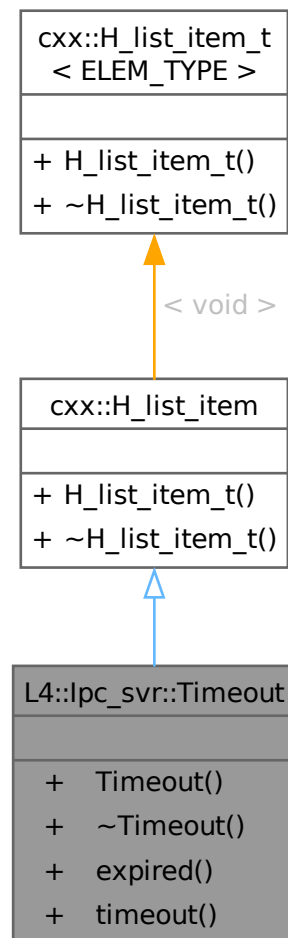
Callback interface for [Timeout_queue](#).

```
#include <ipc_timeout_queue>
```


Inheritance diagram for L4::lpc_svr::Timeout:



Collaboration diagram for L4::lpc_svr::Timeout:



Public Member Functions

- **Timeout ()**
Make a timeout.
- virtual **~Timeout ()=0**
Destroy a timeout.
- virtual void **expired ()=0**
callback function to be called when timeout happened
- **l4_kernel_clock_t timeout () const**
return absolute timeout of this callback.

Public Member Functions inherited from cxx::H_list_item_t< void >

- **H_list_item_t ()**
Constructor.
- **~H_list_item_t () noexcept**
Destructor.

15.160.1 Detailed Description

Callback interface for [Timeout_queue](#).

Definition at line 18 of file [ipc_timeout_queue](#).

15.160.2 Member Function Documentation

15.160.2.1 expired()

```
virtual void L4::Ipc_svr::Timeout::expired () [pure virtual]
```

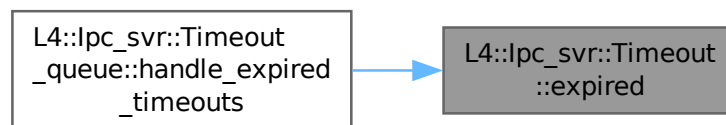
callback function to be called when timeout happened

Note

The timeout object is already dequeued when this function is called, this means the timeout may be safely queued again within the [expired\(\)](#) function.

Referenced by [L4::lpc_svr::Timeout_queue::handle_expired_timeouts\(\)](#).

Here is the caller graph for this function:



15.160.2.2 timeout()

```
l4_kernel_clock_t L4::Ipc_svr::Timeout::timeout () const [inline]
```

return absolute timeout of this callback.

Returns

absolute timeout for this instance of the timeout.

Precondition

The timeout object must have been in a queue before, otherwise the timeout is not set.

Definition at line 42 of file [ipc_timeout_queue](#).

The documentation for this class was generated from the following file:

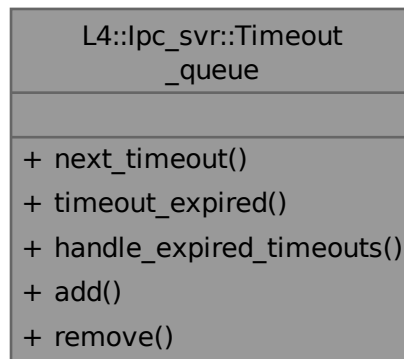
- `I4/cxx/ipc_timeout_queue`

15.161 L4::lpc_svr::Timeout_queue Class Reference

[Timeout](#) queue to be used in l4re server loop.

```
#include <ipc_timeout_queue>
```

Collaboration diagram for L4::lpc_svr::Timeout_queue:



Public Types

- typedef [L4::lpc_svr::Timeout](#) **Timeout**
Provide a local definition of [Timeout](#) for backward compatibility.

Public Member Functions

- [l4_kernel_clock_t](#) [next_timeout](#) () const
Get the time for the next timeout.
- bool [timeout_expired](#) ([l4_kernel_clock_t](#) now) const
Determine if a timeout has happened.
- void [handle_expired_timeouts](#) ([l4_kernel_clock_t](#) now)
run the callbacks of expired timeouts
- void [add](#) ([Timeout](#) *timeout, [l4_kernel_clock_t](#) time)
Add a timeout to the queue.
- void [remove](#) ([Timeout](#) *timeout)
Remove timeout from the queue.

15.161.1 Detailed Description

[Timeout](#) queue to be used in l4re server loop.

Definition at line 55 of file [ipc_timeout_queue](#).

15.161.2 Member Function Documentation

15.161.2.1 add()

```
void L4::Ipc_svr::Timeout_queue::add (
    Timeout * timeout,
    l4_kernel_clock_t time) [inline]
```

Add a timeout to the queue.

Parameters

| | |
|----------------|-----------------------------------|
| <i>timeout</i> | timeout object to add |
| <i>time</i> | the time when the timeout expires |

Precondition

timeout must not be in any queue already

Definition at line 111 of file [ipc_timeout_queue](#).

15.161.2.2 handle_expired_timeouts()

```
void L4::Ipc_svr::Timeout_queue::handle_expired_timeouts (
    l4_kernel_clock_t now) [inline]
```

run the callbacks of expired timeouts

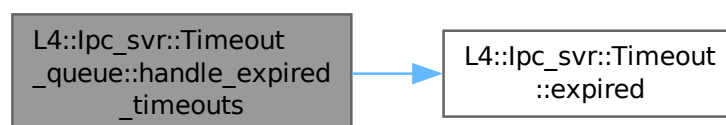
Parameters

| | |
|------------|-------------------|
| <i>now</i> | the current time. |
|------------|-------------------|

Definition at line 91 of file [ipc_timeout_queue](#).

References [L4::lpc_svr::Timeout::expired\(\)](#).

Here is the call graph for this function:



15.161.2.3 next_timeout()

```
l4_kernel_clock_t L4::Ipc_svr::Timeout_queue::next_timeout () const [inline]
```

Get the time for the next timeout.

Returns

the time for the next timeout or 0 if there is none

Definition at line 65 of file [ipc_timeout_queue](#).

Referenced by [timeout_expired\(\)](#).

Here is the caller graph for this function:



15.161.2.4 remove()

```
void L4::Ipc_svr::Timeout_queue::remove (
    Timeout * timeout) [inline]
```

Remove *timeout* from the queue.

Parameters

| | |
|----------------|--------------------------------------|
| <i>timeout</i> | timeout to remove from timeout queue |
|----------------|--------------------------------------|

Precondition

timeout must be in this queue

Definition at line 126 of file [ipc_timeout_queue](#).

15.161.2.5 timeout_expired()

```
bool L4::lpc_svr::Timeout_queue::timeout_expired (
    l4_kernel_clock_t now) const [inline]
```

Determine if a timeout has happened.

Parameters

| | |
|------------|-------------------|
| <i>now</i> | The current time. |
|------------|-------------------|

Return values

| | |
|-------------|---|
| <i>true</i> | There is at least one expired timeout in the queue. <i>false</i> No expired timeout in the queue. |
|-------------|---|

Definition at line 81 of file [ipc_timeout_queue](#).

References [next_timeout\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

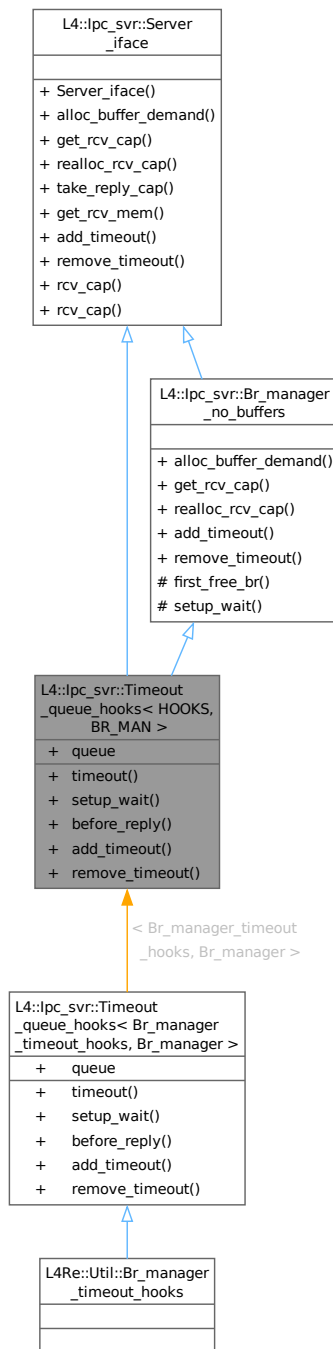
- `l4/cxx/ipc_timeout_queue`

15.162 L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN > Class Template Reference

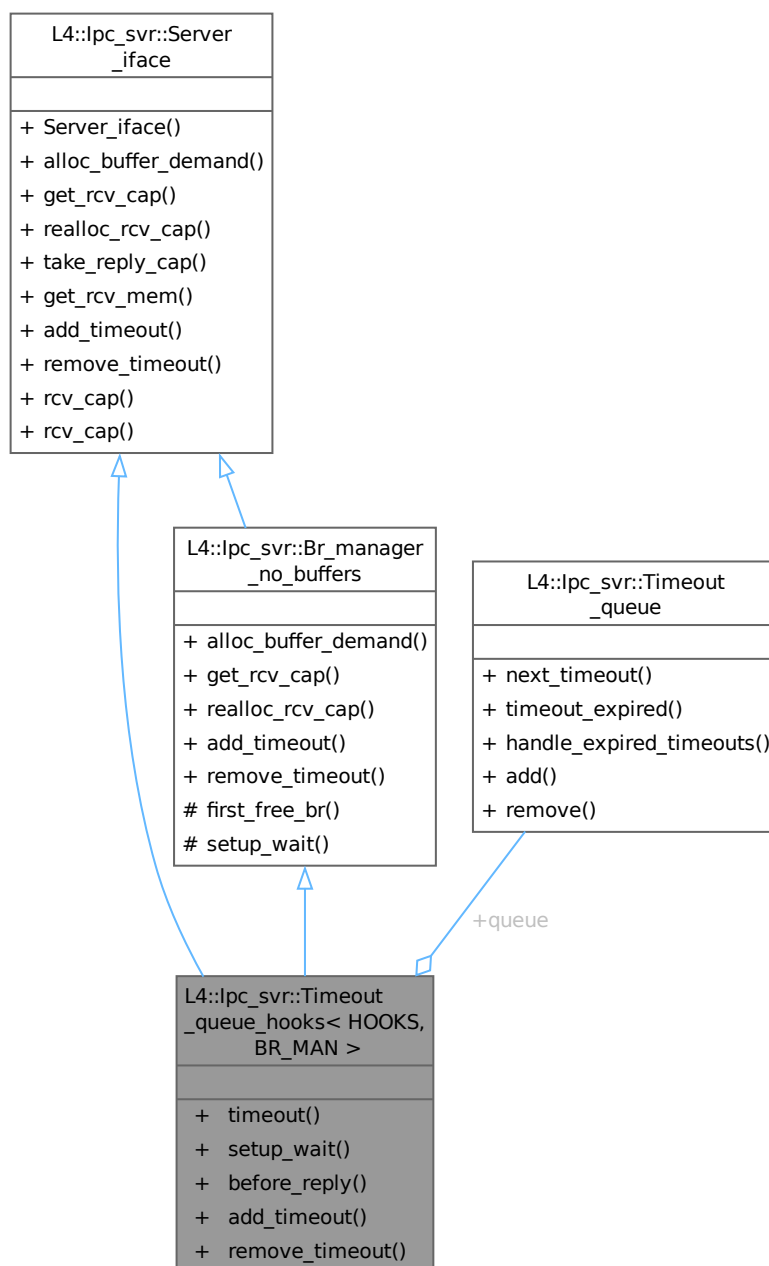
Loop hooks mixin for integrating a timeout queue into the server loop.

```
#include <ipc_timeout_queue>
```

Inheritance diagram for L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >:



Collaboration diagram for L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >:



Public Member Functions

- [l4_timeout_t](#) **timeout** ()
get the time for the next timeout
- void **setup_wait** ([l4_utcb_t](#) *utcb, [L4::lpc_svr::Reply_mode](#) mode)
setup_wait() for the server loop
- [L4::lpc_svr::Reply_mode](#) **before_reply** ([l4_msgtag_t](#), [l4_utcb_t](#) *)
server loop hook

- int [add_timeout](#) ([Timeout](#) *[timeout](#), [l4_kernel_clock_t](#) [time](#)) override
Add a timeout to the queue for time time.
- int [remove_timeout](#) ([Timeout](#) *[timeout](#)) override
Remove timeout from the queue.

Public Member Functions inherited from [L4::lpc_svr::Server_iface](#)

- [Server_iface](#) ()
Make a server interface.
- virtual [cxx::Result](#)< [L4::Reply_cap](#) > [take_reply_cap](#) () noexcept
Take the currently used reply capability.
- virtual [cxx::Result](#)< [Mem_window](#) > [get_rcv_mem](#) () noexcept
Take the current memory receive window.
- template<typename T>
[L4::Cap](#)< T > [rcv_cap](#) (int [index](#)) const
Get given receive buffer as typed capability.
- [L4::Cap](#)< void > [rcv_cap](#) (int [index](#)) const
Get receive cap with the given index as generic (void) type.

Public Member Functions inherited from [L4::lpc_svr::Br_manager_no_buffers](#)

- int [alloc_buffer_demand](#) ([Demand](#) const &[demand](#)) override
Tells the server to allocate buffers for the given demand.
- [L4::Cap](#)< void > [get_rcv_cap](#) (int) const override
Returns [L4::Cap<void>::Invalid](#), we have no buffer management.
- int [realloc_rcv_cap](#) (int) override
Returns -L4_ENOMEM, we have no buffer management.

Data Fields

- [Timeout_queue](#) [queue](#)
Use this timeout queue.

Additional Inherited Members

Public Types inherited from [L4::lpc_svr::Server_iface](#)

- using [Demand](#) = [L4::Type_info::Demand](#)
Data type expressing server-side demand for receive buffers.

Protected Member Functions inherited from [L4::lpc_svr::Br_manager_no_buffers](#)

- unsigned [first_free_br](#) () const
Returns 1 as first free buffer.
- void [setup_wait](#) ([l4_utcb_t](#) *[utcb](#), [L4::lpc_svr::Reply_mode](#))
Setup wait function for the server loop ([Server<>](#)).

15.162.1 Detailed Description

```
template<typename HOOKS, typename BR_MAN = Br_manager_no_buffers>
class L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >
```

Loop hooks mixin for integrating a timeout queue into the server loop.

Template Parameters

| | |
|---------------|---|
| <i>HOOKS</i> | has to inherit from Timeout_queue_hooks<> and provide the functions <code>now()</code> that has to return the current time. |
| <i>BR_MAN</i> | This used as a base class for and provides the API for selecting the buffer register (BR) that is used to store the timeout value. This is usually L4Re::Util::Br_manager or L4::lpc_svr::Br_manager_no_buffers . |

Definition at line 150 of file [ipc_timeout_queue](#).

15.162.2 Member Function Documentation

15.162.2.1 add_timeout()

```
template<typename HOOKS, typename BR_MAN = Br_manager_no_buffers>
int L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >::add_timeout (
    Timeout * timeout,
    l4_kernel_clock_t time) [inline], [override], [virtual]
```

Add a timeout to the queue for time *time*.

Parameters

| | |
|----------------|--|
| <i>timeout</i> | The timeout object to add into the queue (must not be in any queue currently). |
| <i>time</i> | The time when the timeout shall expire. |

Precondition

`timeout` must not be in any queue.

Note

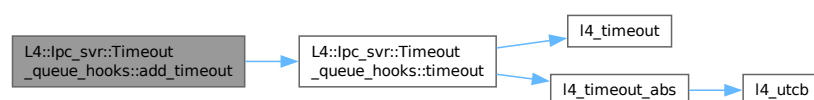
The timeout is automatically dequeued before the [Timeout::expired\(\)](#) function is called

Implements [L4::lpc_svr::Server_iface](#).

Definition at line 202 of file [ipc_timeout_queue](#).

References [queue](#), and [timeout\(\)](#).

Here is the call graph for this function:



15.162.2.2 remove_timeout()

```
template<typename HOOKS, typename BR_MAN = Br_manager_no_buffers>
int L4::Ipc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >::remove_timeout (
    Timeout * timeout) [inline], [override], [virtual]
```

Remove timeout from the queue.

Parameters

| | |
|----------------|--|
| <i>timeout</i> | The timeout object to be removed from the queue. |
|----------------|--|

Note

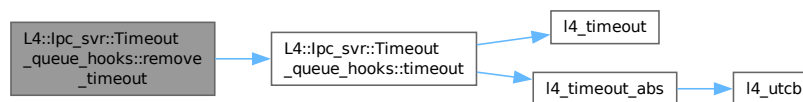
This function may be safely called even if the timeout is not currently enqueued.
in [Timeout::expired\(\)](#) the timeout is already dequeued!

Implements [L4::Ipc_svr::Server_iface](#).

Definition at line 215 of file [ipc_timeout_queue](#).

References [queue](#), and [timeout\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

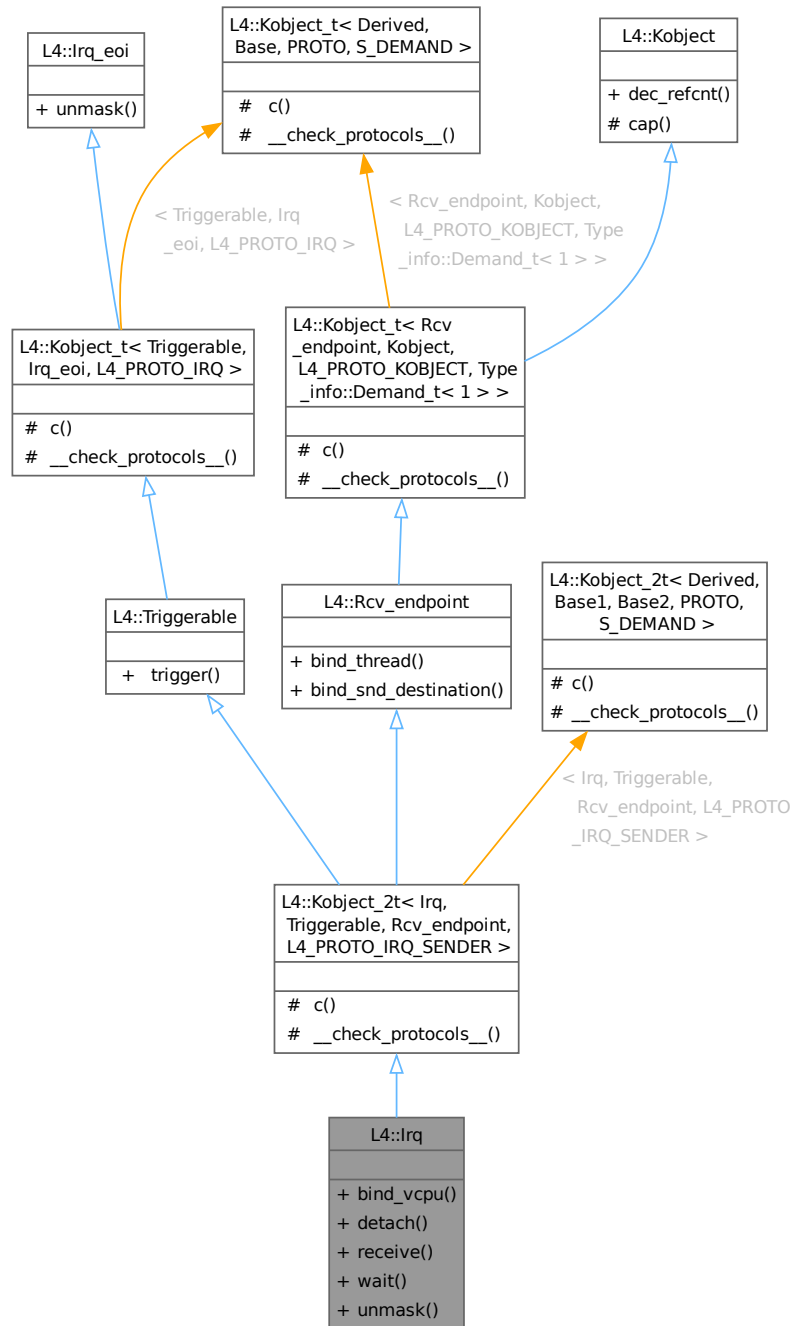
- [I4/cxx/ipc_timeout_queue](#)

15.163 L4::Irq Class Reference

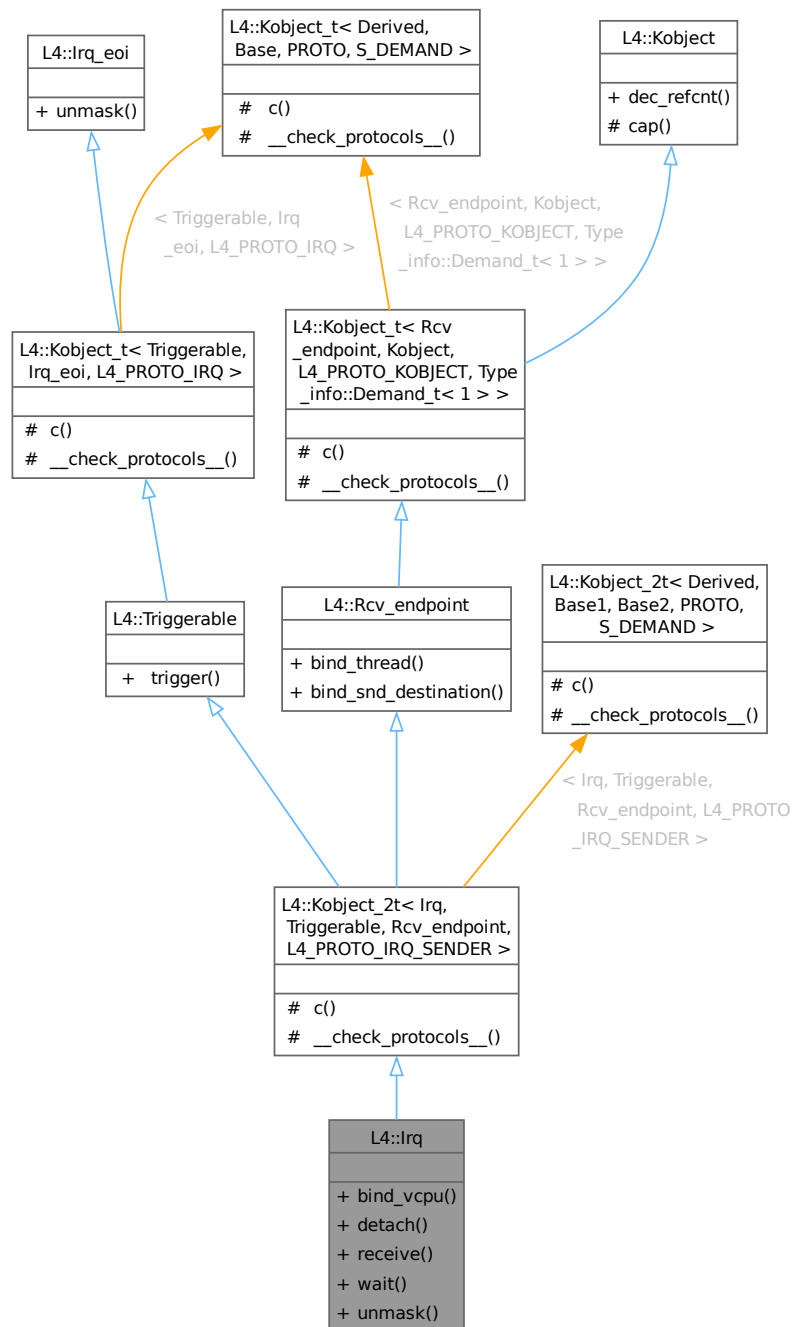
C++ [Irq](#) interface, see [IRQs](#) for the C interface.

```
#include <irq>
```

Inheritance diagram for L4::Irq:



Collaboration diagram for L4::Irq:



Public Member Functions

- **[l4_msgtag_t bind_vcpu](#)** (**[L4::Cap](#)**< **[Thread](#)** > const &thread, **[l4_umword_t](#)** cfg, **[l4_utcb_t](#)** *utcb=**[l4_utcb\(\)](#)**) noexcept
Bind a thread to this **[Irq](#)** for vCPU interrupt forwarding.
- **[l4_msgtag_t detach](#)** (**[l4_utcb_t](#)** *utcb=**[l4_utcb\(\)](#)**) noexcept
Detach from this interrupt.

- `l4_msgtag_t receive (l4_timeout_t timeout=L4_IPC_NEVER, l4_utcb_t *utcb=l4_utcb()) noexcept`
Unmask and wait for this IRQ.
- `l4_msgtag_t wait (l4_umword_t *label, l4_timeout_t timeout=L4_IPC_NEVER, l4_utcb_t *utcb=l4_utcb()) noexcept`
Unmask IRQ and (open) wait for any message.
- `l4_msgtag_t unmask (l4_utcb_t *utcb=l4_utcb()) noexcept`
Unmask this IRQ.

Public Member Functions inherited from L4::Triggerable

- `l4_msgtag_t trigger (l4_utcb_t *utcb=l4_utcb()) noexcept`
Trigger the object.

Public Member Functions inherited from L4::lrq_eoi

- `l4_msgtag_t unmask` (unsigned irqnum, `l4_umword_t` *label=0, `l4_timeout_t` to=L4_IPC_NEVER, `l4_utcb_t` *utcb=`l4_utcb()`) noexcept
Unmask the given interrupt line.

Public Member Functions inherited from L4::Rcv_endpoint

- `l4_msgtag_t bind_thread (lpc::Cap< Thread > t, l4_umword_t label)`
Bind the IPC receive endpoint to a thread.
- `l4_msgtag_t bind_snd_destination (Cap< Snd_destination > snd_dst, l4_umword_t label)`
Bind a send destination (a thread or thread group) to an IPC receive endpoint.

Public Member Functions inherited from L4::Kobject

- `l4_msgtag_t dec_refcnt (l4_mword_t diff, l4_utcb_t *utcb=l4_utcb())`
Decrement the in kernel reference counter for the object.

Additional Inherited Members

Protected Types inherited from

L4::Kobject_2t < Irq, Triggerable, Rcv_endpoint, L4_PROTO_IRQ_SENDER >

- `typedef Irq Class`
The target interface type (inheriting from `Kobject_t`).
- `typedef Typeid::Iface< PROTO, Irq > __Iface`
The interface description for the derived class.
- `typedef Typeid::Merge_list< Typeid::Iface_list< __Iface >, Typeid::Merge_list< typename Triggerable::__↵
Iface_list, typename Rcv_endpoint::__Iface_list > > __Iface_list`
The list of all RPC interfaces provided directly or through inheritance.

Protected Types inherited from [L4::Kobject_t< Triggerable, Irq_eoi, L4_PROTO_IRQ >](#)

- typedef [Triggerable](#) **Class**
The target interface type (inheriting from [Kobject_t](#)).
- typedef Typeid::Iface< PROTO, [Triggerable](#) > **__Iface**
The interface description for the derived class.
- typedef Typeid::Merge_list< Typeid::Iface_list< **__Iface** >, typename Irq_eoi::__Iface_list > **__Iface_list**
The list of all RPC interfaces provided directly or through inheritance.

Protected Types inherited from

[L4::Kobject_t< Rcv_endpoint, Kobject, L4_PROTO_KOBJECT, Type_info::Demand_t< 1 > >](#)

- typedef [Rcv_endpoint](#) **Class**
The target interface type (inheriting from [Kobject_t](#)).
- typedef Typeid::Iface< PROTO, [Rcv_endpoint](#) > **__Iface**
The interface description for the derived class.
- typedef Typeid::Merge_list< Typeid::Iface_list< **__Iface** >, typename Kobject::__Iface_list > **__Iface_list**
The list of all RPC interfaces provided directly or through inheritance.

Protected Member Functions inherited from

[L4::Kobject_2t< Irq, Triggerable, Rcv_endpoint, L4_PROTO_IRQ_SENDER >](#)

- [L4::Cap< Class > c](#) () const noexcept
Get the capability to ourselves.

Protected Member Functions inherited from

[L4::Kobject_t< Triggerable, Irq_eoi, L4_PROTO_IRQ >](#)

- [L4::Cap< Class > c](#) () const noexcept
Get the capability to ourselves.

Protected Member Functions inherited from

[L4::Kobject_t< Rcv_endpoint, Kobject, L4_PROTO_KOBJECT, Type_info::Demand_t< 1 > >](#)

- [L4::Cap< Class > c](#) () const noexcept
Get the capability to ourselves.

Protected Member Functions inherited from [L4::Kobject](#)

- [l4_cap_idx_t cap](#) () const noexcept
Return capability selector.

Static Protected Member Functions inherited from

[L4::Kobject_2t< Irq, Triggerable, Rcv_endpoint, L4_PROTO_IRQ_SENDER >](#)

- static void [__check_protocols__](#) () noexcept
Helper to check for protocol conflicts.

Static Protected Member Functions inherited from [L4::Kobject_t< Triggerable, Irq_eoi, L4_PROTO_IRQ >](#)

- static void `__check_protocols__()` noexcept
Helper to check for protocol conflicts.

Static Protected Member Functions inherited from [L4::Kobject_t< Rcv_endpoint, Kobject, L4_PROTO_KOBJECT, Type_info::Demand_t< 1 > >](#)

- static void `__check_protocols__()` noexcept
Helper to check for protocol conflicts.

15.163.1 Detailed Description

C++ [Irq](#) interface, see [IRQs](#) for the C interface.

Note

"IRQ" is short for "interrupt request". This is often used interchangeably for "interrupt"

The [Irq](#) class provides access to abstract interrupts provided by the microkernel. Interrupts may be

- hardware interrupts provided by the platform interrupt controller,
- virtual device interrupts provided by the microkernel's virtual devices (virtual serial or trace buffer) or
- virtual interrupts that can be triggered by user programs (IRQs) via the inherited method [L4::Triggerable::trigger\(\)](#).

For hardware and virtual device interrupts the [Irq](#) object must be bound to an interrupt source, see [L4::Icu](#). To receive interrupts, the [Irq](#) object must be bound to a thread, see [L4::Rcv_endpoint](#).

[Irq](#) objects can be created using a factory, see the [L4::Factory](#) API ([L4::Factory::create\(\)](#)).

Include File

```
#include <l4/sys/irq>
```

For the C interface refer to the [IRQs](#) API for an overview.

Examples

[examples/libs/l4re/c++/shared_ds/ds_clnt.cc](#).

Definition at line 120 of file [irq](#).

15.163.2 Member Function Documentation

15.163.2.1 bind_vcpu()

```
l4_msgtag_t L4::Irq::bind_vcpu (
    L4::Cap< Thread > const & thread,
    l4_umword_t cfg,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
```

Bind a thread to this [lrq](#) for vCPU interrupt forwarding.

If the interrupt is triggered, the kernel will directly inject the interrupt into the guest. This requires that the thread is currently in extended vCPU user mode. Otherwise the interrupt will stay pending and gets injected on the next vCPU user mode transition. Optionally a doorbell [lrq](#) can be registered on the thread (see [Thread::register_doorbell_irq\(\)](#)) that is triggered in this case.

If a guest has acknowledged the interrupt but has not yet issued an EOI (i.e. the interrupt is in "active" state), it is not possible to bind the [lrq](#) to a new thread object. Either wait for the guest to issue the EOI or [detach\(\)](#) from the current thread. In this case the interrupt will stay active in the guest and it is the responsibility of the VMM to handle the eventual EOI of the guest.

Parameters

| | |
|---------------|--|
| <i>thread</i> | Thread object this lrq shall be bound to. |
| <i>cfg</i> | Architecture specific interrupt configuration. |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |

Returns

Syscall return tag

Return values

| | |
|----------------------------|---|
| -L4_EPERM | Insufficient permissions; see precondition. |
| -L4_EBUSY | Cannot bind to the new thread because interrupt is active on previous thread and guest has to issue end-of-interrupt first. |
| -L4_ENOSYS | The kernel does not support direct interrupt forwarding. |

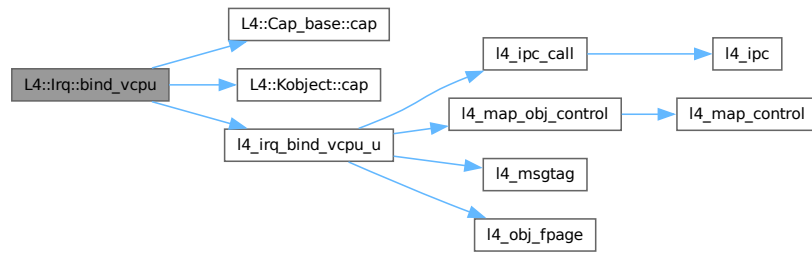
Precondition

The invoked [lrq](#) capability and the capability `thread` both must have the permission [L4_CAP_FPAGE_S](#).

Definition at line 158 of file [irq](#).

References [L4::Cap_base::cap\(\)](#), [L4::Kobject::cap\(\)](#), and [l4_irq_bind_vcpu_u\(\)](#).

Here is the call graph for this function:



15.163.2.2 detach()

```
l4_msgtag_t L4::Irq::detach (
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
```

Detach from this interrupt.

Parameters

| | |
|-------------|--|
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |
|-------------|--|

Returns

Syscall return tag

Return values

| | |
|-----------|--|
| 0 | Successfully detached, there was no interrupt pending. |
| 1 | Successfully detached, there was an interrupt pending. |
| 2 | Successfully detached, an active vIRQ was abandoned. |
| -L4_EPERM | Insufficient permissions; see precondition. |

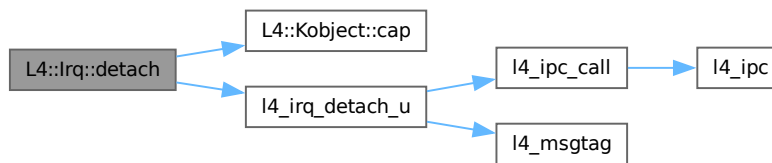
Precondition

The invoked [Irq](#) capability must have the permission [L4_CAP_FPAGE_S](#).

Definition at line 176 of file [irq](#).

References [L4::Kobject::cap\(\)](#), and [l4_irq_detach_u\(\)](#).

Here is the call graph for this function:



15.163.2.3 receive()

```

l4_msgtag_t L4::Irq::receive (
    l4_timeout_t timeout = L4_IPC_NEVER,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
  
```

Unmask and wait for this IRQ.

Parameters

| | |
|----------------|--|
| <i>timeout</i> | Timeout. |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |

Returns

Syscall return tag

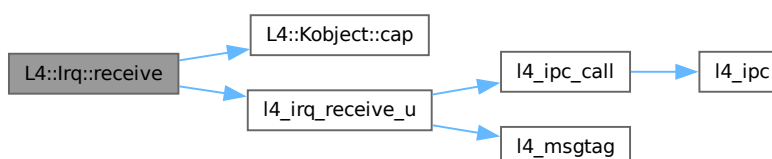
Note

If this is the function normally used for your IRQs consider using [L4::Semaphore](#) instead of [L4::Irq](#).

Definition at line 191 of file [irq](#).

References [L4::Kobject::cap\(\)](#), [L4_IPC_NEVER](#), and [I4_irq_receive_u\(\)](#).

Here is the call graph for this function:



15.163.2.4 unmask()

```
l4_msgtag_t L4::Irq::unmask (
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
```

Unmask this IRQ.

Parameters

| | |
|-------------|--|
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |
|-------------|--|

Returns

Syscall return tag for a send-only operation, this means there is no return value except [L4_MSGTAG_ERROR](#) indicating success or failure of the send operation. Use [l4_ipc_error\(\)](#) to check for errors and **do not** use [l4_error\(\)](#).

[Irq::wait\(\)](#) and [Irq::receive\(\)](#) operations already include an [unmask\(\)](#), do not use an extra [unmask\(\)](#) in these cases.

Definition at line 221 of file [irq](#).

References [L4_IPC_NEVER](#), and [unmask\(\)](#).

Referenced by [unmask\(\)](#), and [wait\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.163.2.5 wait()

```
l4_msgtag_t L4::Irq::wait (
    l4_umword_t * label,
    l4_timeout_t timeout = L4_IPC_NEVER,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
```

Unmask IRQ and (open) wait for any message.

Parameters

| | |
|----------------|--|
| <i>label</i> | The <i>protected label</i> shall be received here. |
| <i>timeout</i> | Timeout. |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |

Returns

Syscall return tag

Definition at line [204](#) of file [irq](#).

References [L4_IPC_NEVER](#), and [unmask\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

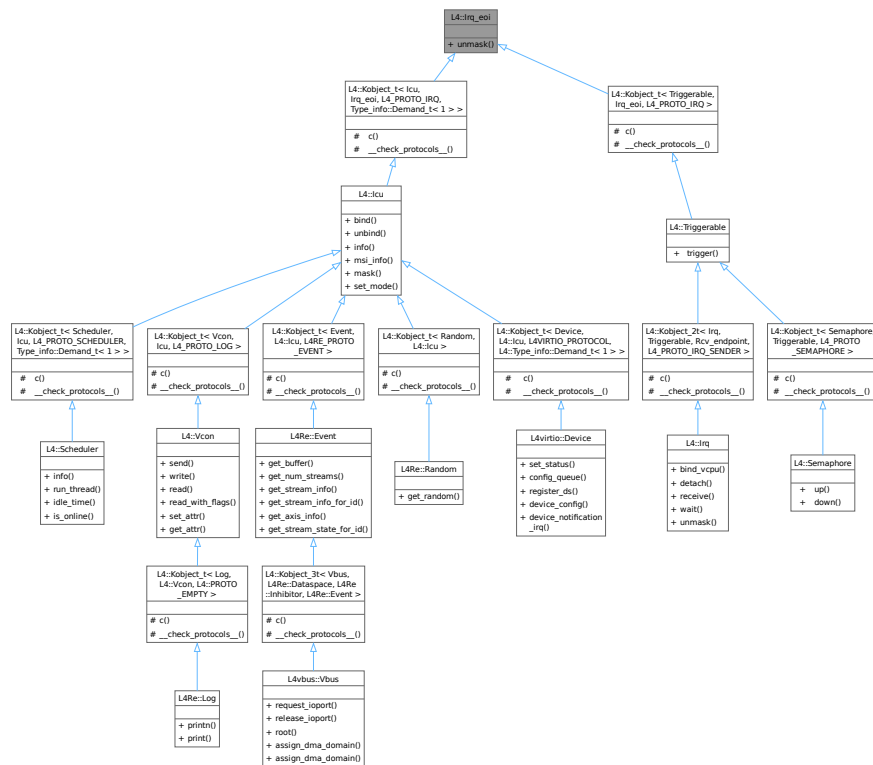
- [l4/sys/irq](#)

15.164 L4::Irq_eoi Class Reference

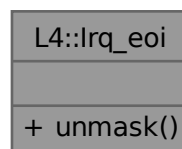
Interface for sending an unmask message to an object.

```
#include <irq>
```

Inheritance diagram for L4::Irq_eoi:



Collaboration diagram for L4::Irq_eoi:



Public Member Functions

- [l4_msgtag_t unmask](#) (unsigned irqnum, [l4_umword_t](#) *label=0, [l4_timeout_t](#) to=L4_IPC_NEVER, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) noexcept
Unmask the given interrupt line.

15.164.1 Detailed Description

Interface for sending an unmask message to an object.

The object is usually an ICU or an IRQ.

When the kernel receives an IRQ, it masks the interrupt line at the interrupt controller and immediately acknowledges the interrupt. This interface is used to let the kernel know that userspace has dealt with the IRQ. The kernel will unmask the interrupt line and further IRQs can then be delivered.

See also

[L4::lcu](#), [L4::irq](#)

Definition at line 37 of file [irq](#).

15.164.2 Member Function Documentation

15.164.2.1 unmask()

```
l4_msgtag_t L4::Irq_eoi::unmask (
    unsigned irqnum,
    l4_umword_t * label = 0,
    l4_timeout_t to = L4_IPC_NEVER,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
```

Unmask the given interrupt line.

When the object is an IRQ, the given interrupt line is ignored and instead the line which the IRQ is bound to (if any) is unmasked.

Its counterpart for explicitly masking an interrupt line is [L4::lcu::mask\(\)](#).

Parameters

| | | |
|-----|---------------|---|
| | <i>irqnum</i> | The interrupt line that shall be unmasked. Ignored if the object is an IRQ. |
| out | <i>label</i> | If NULL, this is a send-only unmask. If not NULL, this operation enters an open wait and the <i>protected label</i> shall be received here. |
| | <i>to</i> | The timeout-pair (send and receive) that shall be used for this operation. The receive timeout is used with a non-NULL <i>label</i> only. |
| | <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |

Returns

Syscall return tag. If *label* is NULL, this function performs an IPC send-only operation and there is no return value except [L4_MSGTAG_ERROR](#) indicating success or failure of the send operation. In this case use [l4_ipc_error\(\)](#) to check for errors and **do not** use [l4_error\(\)](#).

Definition at line 64 of file [irq](#).

References [L4::Kobject::cap\(\)](#), and [L4_IPC_NEVER](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

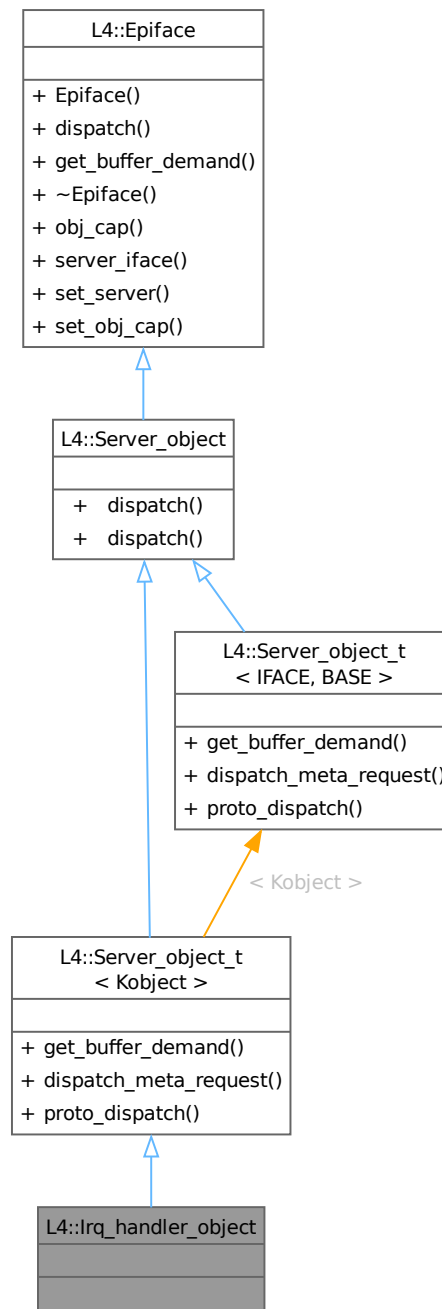
- `l4/sys/irq`

15.165 L4::lrq_handler_object Struct Reference

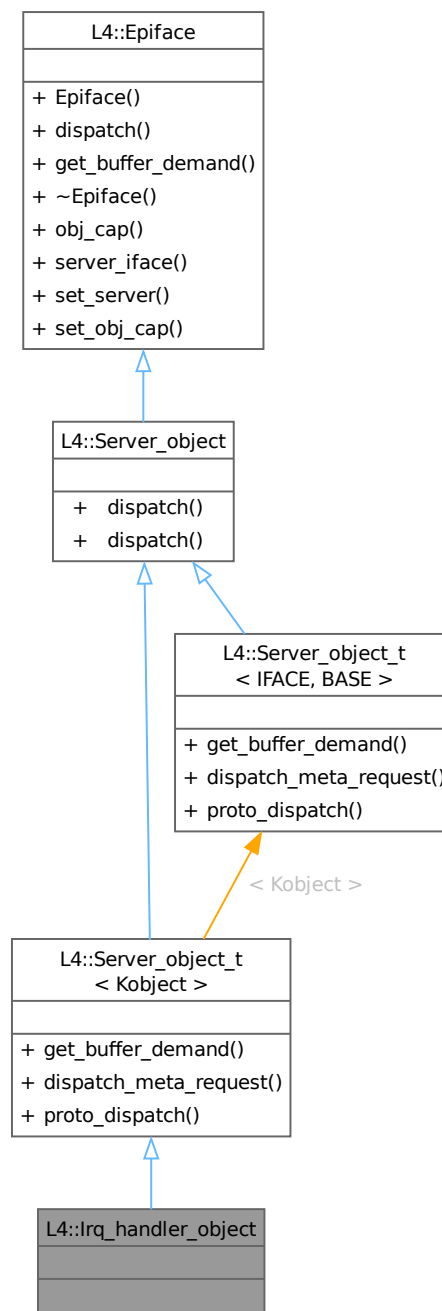
[Server](#) object base class for handling IRQ messages.

```
#include <ipc_server>
```

Inheritance diagram for L4::lrq_handler_object:



Collaboration diagram for L4::lrq_handler_object:



Additional Inherited Members

Public Types inherited from **L4::Server_object_t< Kobject >**

- typedef **Kobject** Interface

Data type of the IPC interface definition.

Public Types inherited from [L4::Epiface](#)

- using **Server_iface** = [lpc_svr::Server_iface](#)
Type for abstract server interface.
- using **Demand** = [lpc_svr::Server_iface::Demand](#)
Type for server-side receive buffer demand.

Public Member Functions inherited from [L4::Server_object_t< Kobject >](#)

- [L4::Server_object::Demand get_buffer_demand](#) () const override
- int [dispatch_meta_request](#) ([L4::lpc::lostream](#) &ios)
Implementation of the meta protocol based on IFACE.

Public Member Functions inherited from [L4::Server_object](#)

- virtual int [dispatch](#) (unsigned long rights, [lpc::lostream](#) &ios)=0
The abstract handler for client requests to the object.
- [l4_msgtag_t](#) [dispatch](#) ([l4_msgtag_t](#) tag, unsigned rights, [l4_utcb_t](#) *utcb) override
The abstract handler for client requests to the object.

Public Member Functions inherited from [L4::Epiface](#)

- **Epiface** ()
Make a server object.
- virtual ~**Epiface** ()=0
Destroy the object.
- Stored_cap [obj_cap](#) () const
Get the capability to the kernel object belonging to this object.
- [Server_iface](#) * [server_iface](#) () const
Get pointer to server interface at which the object is currently registered.
- int [set_server](#) ([Server_iface](#) *srv, [Cap](#)< void > cap, bool managed=false)
Set server registration info for the object.
- void [set_obj_cap](#) ([Cap](#)< void > const &cap)
Deprecated server registration function.

Static Public Member Functions inherited from [L4::Server_object_t< Kobject >](#)

- static int [proto_dispatch](#) (THIS *self, [l4_umword_t](#) rights, [L4::lpc::lostream](#) &ios)
Implementation of protocol-based dispatch for this server object.

15.165.1 Detailed Description

[Server](#) object base class for handling IRQ messages.

This server object base class implements the empty interface ([L4::Kobject](#)). The implementation of [Server_object::dispatch\(\)](#) must return [-L4_ENOREPLY](#), because IRQ messages do not handle replies.

Examples

[examples/libs/l4re/c++/shared_ds/ds_srv.cc](#).

Definition at line 161 of file [ipc_server](#).

The documentation for this struct was generated from the following file:

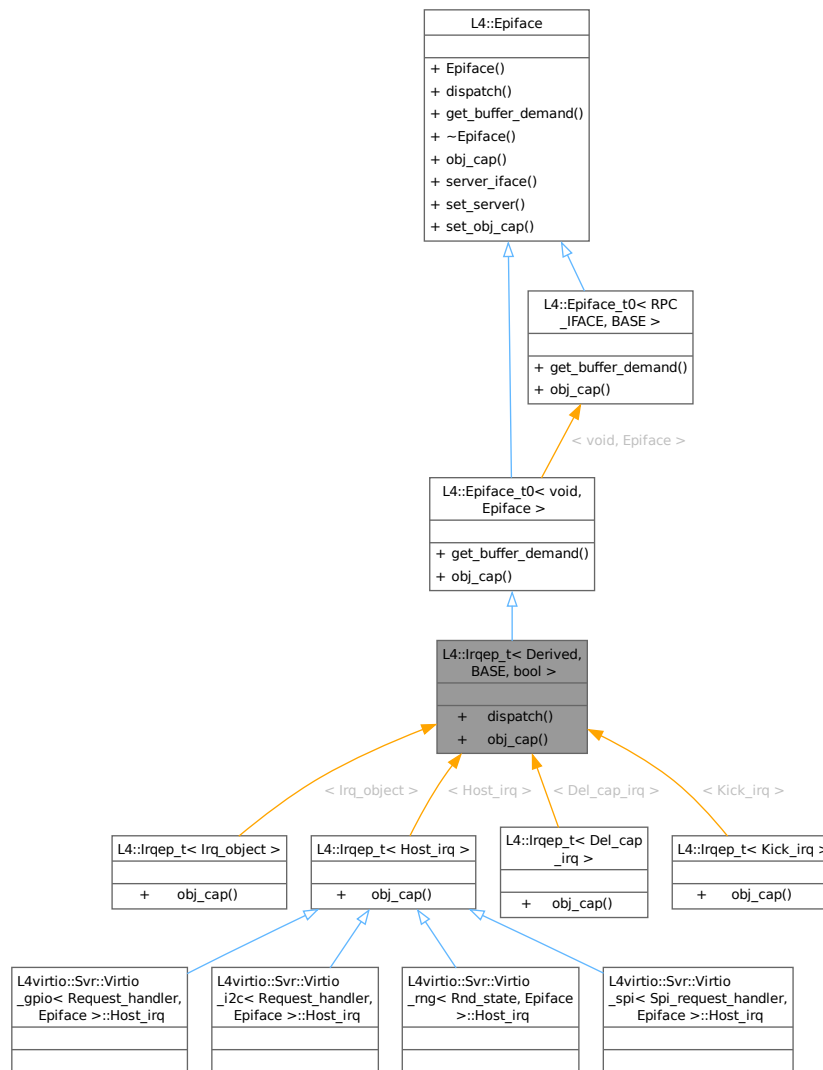
- [l4/cxx/ipc_server](#)

15.166 L4::lrqep_t< Derived, BASE, bool > Struct Template Reference

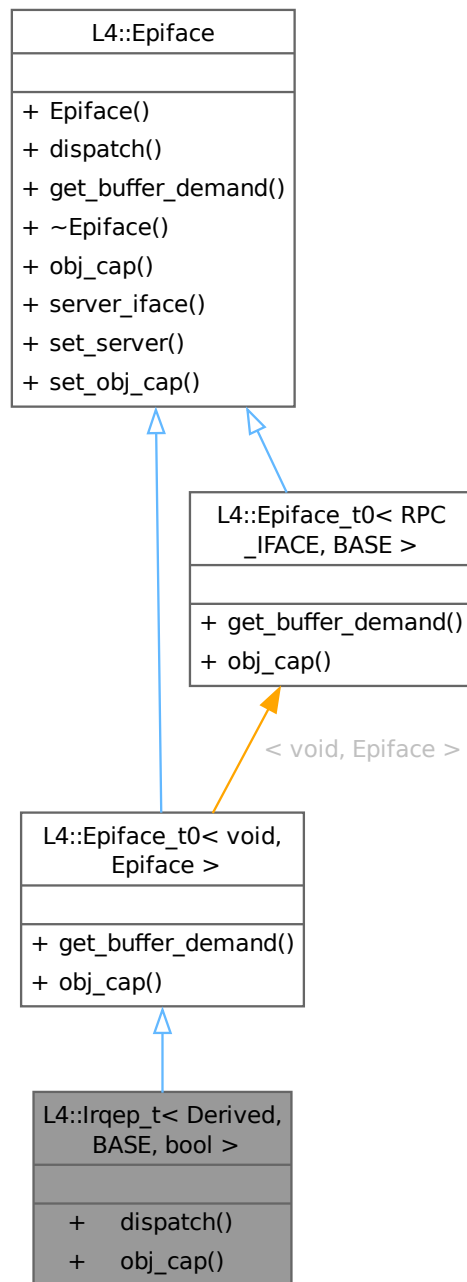
[Epiface](#) implementation for interrupt handlers.

```
#include <ipc_epiface>
```

Inheritance diagram for L4::lrqep_t< Derived, BASE, bool >:



Collaboration diagram for L4::lrqep_t< Derived, BASE, bool >:



Public Member Functions

- `l4_msgtag_t dispatch (l4_msgtag_t, unsigned, l4_utcb_t *)` final
The abstract handler for client requests to the object.
- `Cap< L4::lrq > obj_cap ()` const
Get the (typed) capability to this object.

Public Member Functions inherited from L4::Epiface_t0< void, Epiface >

- [Type_info::Demand](#) **get_buffer_demand** () const
Get the server-side buffer demand based in IFACE.
- [Cap](#)< void > **obj_cap** () const
Get the (typed) capability to this object.

Public Member Functions inherited from L4::Epiface

- **Epiface** ()
Make a server object.
- virtual **~Epiface** ()=0
Destroy the object.
- Stored_cap [obj_cap](#) () const
Get the capability to the kernel object belonging to this object.
- [Server_iface](#) * **server_iface** () const
Get pointer to server interface at which the object is currently registered.
- int **set_server** ([Server_iface](#) *srv, [Cap](#)< void > cap, bool managed=false)
Set server registration info for the object.
- void **set_obj_cap** ([Cap](#)< void > const &cap)
Deprecated server registration function.

Additional Inherited Members**Public Types inherited from L4::Epiface_t0< void, Epiface >**

- using **Interface**
Data type of the IPC interface definition.

Public Types inherited from L4::Epiface

- using **Server_iface** = [lpc_svr::Server_iface](#)
Type for abstract server interface.
- using **Demand** = [lpc_svr::Server_iface::Demand](#)
Type for server-side receive buffer demand.

15.166.1 Detailed Description

```
template<typename Derived, typename BASE = Epiface, bool = cxx::is_polymorphic<BASE>::value>
struct L4::Irqep_t< Derived, BASE, bool >
```

[Epiface](#) implementation for interrupt handlers.

Template Parameters

| | |
|----------------|--|
| <i>Derived</i> | Irq handler implementation class. The class must provide a single function <code>handle_irq()</code> . |
|----------------|--|

| | |
|-------------|-------------------------------------|
| <i>BASE</i> | Base Epiface class. |
|-------------|-------------------------------------|

Definition at line [394](#) of file [ipc_epiface](#).

15.166.2 Member Function Documentation

15.166.2.1 `dispatch()`

```
template<typename Derived, typename BASE = Epiface, bool = cxx::is_polymorphic<BASE>::value>
l4_msgtag_t L4::Irqep_t< Derived, BASE, bool >::dispatch (
    l4_msgtag_t tag,
    unsigned rights,
    l4_utcb_t * utcb) [inline], [final], [virtual]
```

The abstract handler for client requests to the object.

Parameters

| | |
|---------------|--|
| <i>tag</i> | The message tag for this invocation. |
| <i>rights</i> | The rights bits in the invoked capability. |
| <i>utcb</i> | The UTCB used for the invocation. |

Return values

| | |
|---------------------------|-----------------------------------|
| <code>-L4_ENOREPLY</code> | No reply message is sent. |
| <code>< 0</code> | Error, reply with error code. |
| <code>>= 0</code> | Success, reply with return value. |

This function must be implemented by application specific server objects.

Implements [L4::Epiface](#).

Definition at line [396](#) of file [ipc_epiface](#).

References [L4_ENOREPLY](#), and [l4_msgtag\(\)](#).

Here is the call graph for this function:



15.166.2.2 obj_cap()

```
template<typename Derived, typename BASE = Epiface, bool = cxx::is_polymorphic<BASE>::value>  
Cap< L4::Irq > L4::Irqep_t< Derived, BASE, bool >::obj_cap () const [inline]
```

Get the (typed) capability to this object.

Returns

[Irq](#) capability for the kernel object behind the server.

Definition at line 406 of file [ipc_epiface](#).

References [L4::cap_cast\(\)](#).

Here is the call graph for this function:



The documentation for this struct was generated from the following file:

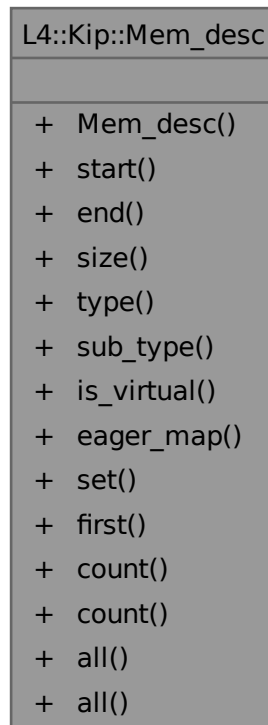
- [l4/sys/cxx/ipc_epiface](#)

15.167 L4::Kip::Mem_desc Class Reference

Memory descriptors stored in the kernel interface page.

```
#include <kip>
```

Collaboration diagram for L4::Kip::Mem_desc:



Public Types

- enum [Mem_type](#) {
[Undefined](#) = 0x0 , [Conventional](#) = 0x1 , [Reserved](#) = 0x2 , [Dedicated](#) = 0x3 ,
[Shared](#) = 0x4 , [Info](#) = 0xd , [Bootloader](#) = 0xe , [Arch](#) = 0xf }
Memory types.
- enum [Info_sub_type](#) { [Info_acpi_rsd](#) = 0 , [Reserved_kernel](#) = 0 , [Reserved_heap](#) = 1 , [Reserved_mmio](#) = 2
}

Memory sub types for the [Mem_type::Info](#) type.
- enum [Arch_sub_type_common](#) { [Arch_acpi_tables](#) = 3 , [Arch_acpi_nvs](#) = 4 }
Common sub types across all architectures for the [Mem_type::Arch](#) type.

Public Member Functions

- [Mem_desc](#) (unsigned long [start](#), unsigned long [end](#), [Mem_type](#) t, unsigned char st=0, bool virt=false, bool eager=false) noexcept
Initialize memory descriptor.
- unsigned long [start](#) () const noexcept
Return start address of memory descriptor.
- unsigned long [end](#) () const noexcept
Return end address of memory descriptor.

- unsigned long [size](#) () const noexcept
Return size of region described by the memory descriptor.
- [Mem_type](#) type () const noexcept
Return type of the memory descriptor.
- unsigned char [sub_type](#) () const noexcept
Return sub-type of the memory descriptor.
- unsigned [is_virtual](#) () const noexcept
Return whether the memory descriptor describes a virtual or physical region.
- unsigned [eager_map](#) () const noexcept
Return whether the region shall be eligible for eager mapping in sigma0 or the root task.
- void [set](#) (unsigned long [start](#), unsigned long [end](#), [Mem_type](#) t, unsigned char st=0, bool virt=false, bool eager=false) noexcept
Set values of a memory descriptor.

Static Public Member Functions

- static [Mem_desc](#) * [first](#) ([l4_kernel_info_t](#) *kip) noexcept
Get first memory descriptor.
- static unsigned long [count](#) ([l4_kernel_info_t](#) const *kip) noexcept
Return number of memory descriptors stored in the kernel info page.
- static void [count](#) ([l4_kernel_info_t](#) *kip, unsigned count) noexcept
Set number of memory descriptors.
- static [cxx::static_vector](#)< [Mem_desc](#) const > [all](#) ([l4_kernel_info_t](#) const *kip)
Return enumerable list of memory descriptors.
- static [cxx::static_vector](#)< [Mem_desc](#) > [all](#) ([l4_kernel_info_t](#) *kip)
Return enumerable list of memory descriptors.

15.167.1 Detailed Description

Memory descriptors stored in the kernel interface page.

Include File

```
#include <l4/sys/kip>
```

Definition at line 42 of file [kip](#).

15.167.2 Member Enumeration Documentation

15.167.2.1 Arch_sub_type_common

```
enum L4::Kip::Mem_desc::Arch_sub_type_common
```

Common sub types across all architectures for the [Mem_type::Arch](#) type.

Enumerator

| | |
|------------------|----------------------------------|
| Arch_acpi_tables | Firmware ACPI tables. |
| Arch_acpi_nvs | Firmware reserved address space. |

Definition at line 76 of file [kip](#).

15.167.2.2 Info_sub_type

```
enum L4::Kip::Mem_desc::Info_sub_type
```

Memory sub types for the [Mem_type::Info](#) type.

Enumerator

| | |
|-----------------|--|
| Info_acpi_rsdp | Physical address of the ACPI root pointer. |
| Reserved_kernel | Kernel image. |
| Reserved_heap | Kernel heap. |
| Reserved_mmio | MMIO range reserved by kernel. |

Definition at line 64 of file [kip](#).

15.167.2.3 Mem_type

```
enum L4::Kip::Mem_desc::Mem_type
```

Memory types.

Enumerator

| | |
|--------------|--|
| Undefined | Undefined memory. |
| Conventional | Conventional memory. |
| Reserved | Reserved region, do not use this memory. |
| Dedicated | Dedicated. |
| Shared | Shared. |
| Info | Info by boot loader. |
| Bootloader | Memory belongs to the boot loader. |
| Arch | Architecture specific memory. |

Definition at line 48 of file [kip](#).

15.167.3 Constructor & Destructor Documentation

15.167.3.1 Mem_desc()

```
L4::Kip::Mem_desc::Mem_desc (
    unsigned long start,
    unsigned long end,
    Mem_type t,
    unsigned char st = 0,
    bool virt = false,
    bool eager = false) [inline], [noexcept]
```

Initialize memory descriptor.

Parameters

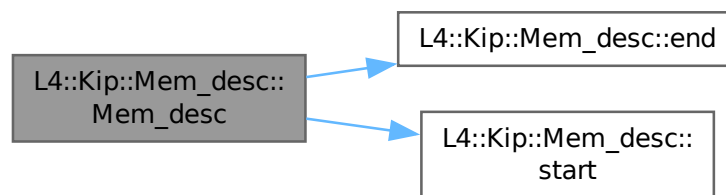
| | |
|--------------|--|
| <i>start</i> | Start address |
| <i>end</i> | End address |
| <i>t</i> | Memory type |
| <i>st</i> | Memory subtype, defaults to 0 |
| <i>virt</i> | True for virtual memory, false for physical memory, defaults to physical |
| <i>eager</i> | The region shall be eligible for eager mapping in sigma0 or the root task. This is just an optimization to prevent on-demand paging. |

Definition at line 162 of file [kip](#).

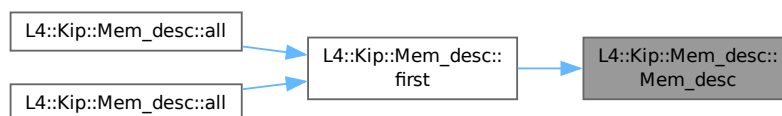
References [end\(\)](#), and [start\(\)](#).

Referenced by [first\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.167.4 Member Function Documentation

15.167.4.1 all() [1/2]

```
cxx::static_vector< Mem_desc > L4::Kip::Mem_desc::all (
    l4_kernel_info_t * kip) [inline], [static]
```

Return enumerable list of memory descriptors.

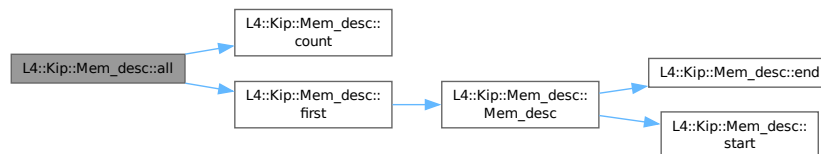
Parameters

| | |
|------------|----------------------------------|
| <i>kip</i> | Pointer to the kernel info page. |
|------------|----------------------------------|

Definition at line 143 of file [kip](#).

References [count\(\)](#), and [first\(\)](#).

Here is the call graph for this function:



15.167.4.2 all() [2/2]

```
cxx::static_vector< Mem_desc const > L4::Kip::Mem_desc::all (
    l4_kernel_info_t const * kip) [inline], [static]
```

Return enumerable list of memory descriptors.

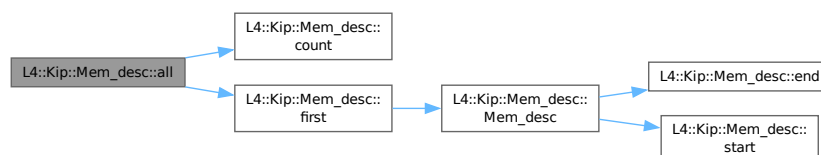
Parameters

| | |
|------------|----------------------------------|
| <i>kip</i> | Pointer to the kernel info page. |
|------------|----------------------------------|

Definition at line 132 of file [kip](#).

References [count\(\)](#), and [first\(\)](#).

Here is the call graph for this function:



15.167.4.3 count() [1/2]

```
void L4::Kip::Mem_desc::count (
    l4_kernel_info_t * kip,
    unsigned count) [inline], [static], [noexcept]
```

Set number of memory descriptors.

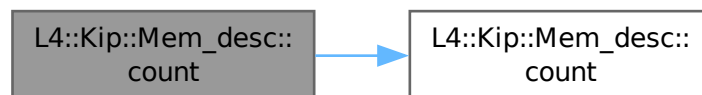
Parameters

| | |
|--------------|---------------------------------|
| <i>kip</i> | Pointer to the kernel info page |
| <i>count</i> | Number of memory descriptors |

Definition at line 122 of file [kip](#).

References [count\(\)](#).

Here is the call graph for this function:



15.167.4.4 count() [2/2]

```
unsigned long L4::Kip::Mem_desc::count (
    l4_kernel_info_t const * kip) [inline], [static], [noexcept]
```

Return number of memory descriptors stored in the kernel info page.

Parameters

| | |
|------------|---------------------------------|
| <i>kip</i> | Pointer to the kernel info page |
|------------|---------------------------------|

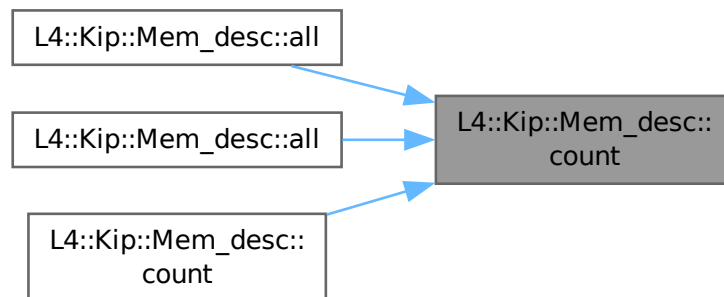
Returns

Number of memory descriptors in the kernel info page.

Definition at line 111 of file [kip](#).

Referenced by [all\(\)](#), [all\(\)](#), and [count\(\)](#).

Here is the caller graph for this function:

**15.167.4.5 end()**

```
unsigned long L4::Kip::Mem_desc::end () const [inline], [noexcept]
```

Return end address of memory descriptor.

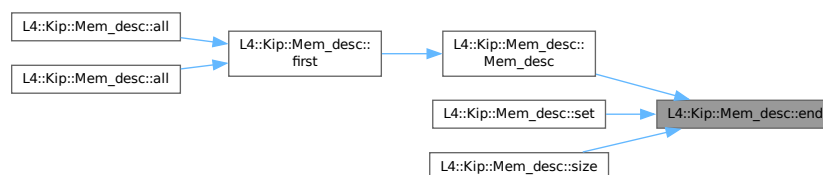
Returns

End address of memory descriptor

Definition at line 181 of file [kip](#).

Referenced by [Mem_desc\(\)](#), [set\(\)](#), and [size\(\)](#).

Here is the caller graph for this function:



15.167.4.6 first()

```
Mem_desc * L4::Kip::Mem_desc::first (
    l4_kernel_info_t * kip) [inline], [static], [noexcept]
```

Get first memory descriptor.

Parameters

| | |
|------------|---------------------------------|
| <i>kip</i> | Pointer to the kernel info page |
|------------|---------------------------------|

Returns

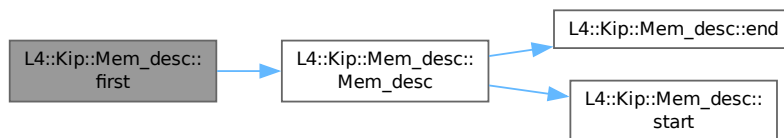
First memory descriptor stored in the kernel info page

Definition at line 93 of file [kip](#).

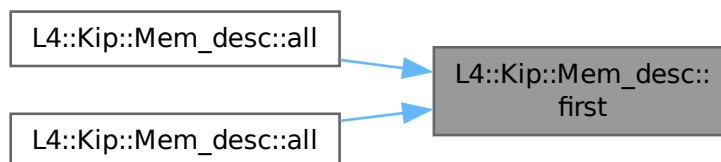
References [Mem_desc\(\)](#).

Referenced by [all\(\)](#), and [all\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.167.4.7 is_virtual()

```
unsigned L4::Kip::Mem_desc::is_virtual () const [inline], [noexcept]
```

Return whether the memory descriptor describes a virtual or physical region.

Returns

True for virtual region, false for physical region.

Definition at line 213 of file [kip](#).

15.167.4.8 set()

```
void L4::Kip::Mem_desc::set (
    unsigned long start,
    unsigned long end,
    Mem_type t,
    unsigned char st = 0,
    bool virt = false,
    bool eager = false) [inline], [noexcept]
```

Set values of a memory descriptor.

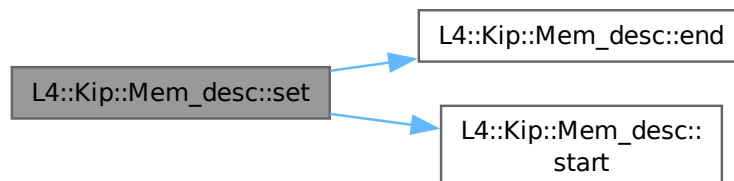
Parameters

| | |
|--------------|--|
| <i>start</i> | Start address |
| <i>end</i> | End address |
| <i>t</i> | Memory type |
| <i>st</i> | Sub-type, defaults to 0 |
| <i>virt</i> | Virtual or physical memory region, defaults to physical |
| <i>eager</i> | The region shall be eligible for eager mapping in sigma0 or the root task. This is just an optimization to prevent on-demand paging. |

Definition at line 233 of file [kip](#).

References [end\(\)](#), and [start\(\)](#).

Here is the call graph for this function:



15.167.4.9 size()

```
unsigned long L4::Kip::Mem_desc::size () const [inline], [noexcept]
```

Return size of region described by the memory descriptor.

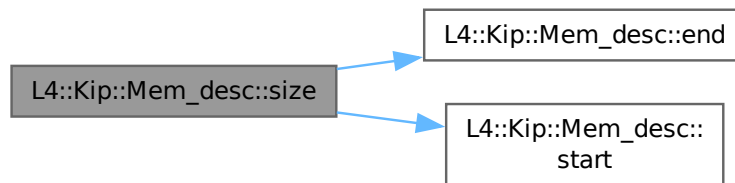
Returns

Size of the region described by the memory descriptor

Definition at line 188 of file [kip](#).

References [end\(\)](#), and [start\(\)](#).

Here is the call graph for this function:

**15.167.4.10 start()**

```
unsigned long L4::Kip::Mem_desc::start () const [inline], [noexcept]
```

Return start address of memory descriptor.

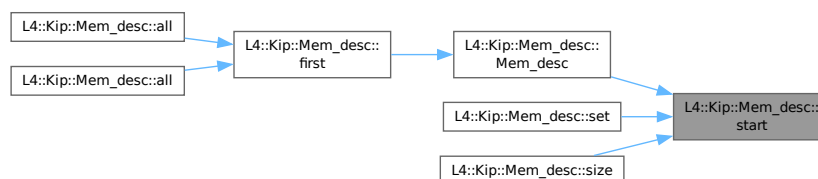
Returns

Start address of memory descriptor

Definition at line 174 of file [kip](#).

Referenced by [Mem_desc\(\)](#), [set\(\)](#), and [size\(\)](#).

Here is the caller graph for this function:



Public Member Functions

- `l4_msgtag_t dec_refcnt (l4_mword_t diff, l4_utcb_t *utcb=l4_utcb())`
Decrement the in kernel reference counter for the object.

Protected Member Functions

- `l4_cap_idx_t cap () const noexcept`
Return capability selector.

15.168.1 Detailed Description

Base class for all kinds of kernel objects and remote objects, referenced by capabilities.

Include File

```
#include <l4/sys/capability>
```

This is the base class for all remote objects accessible using RPC. However, subclasses do not directly inherit from `L4::Kobject` but *must* use `L4::Kobject_t` (`L4::Kobject_0t`, `L4::Kobject_2t`, `L4::Kobject_3t`, or `L4::Kobject_x`) for inheritance, otherwise these classes cannot be used as RPC interfaces.

Attention

Objects derived from `Kobject` *must* never add any data to those objects. Kobjects can act only as proxy object for encapsulating object invocations.

Definition at line 36 of file `kobject`.

15.168.2 Member Function Documentation

15.168.2.1 cap()

```
l4_cap_idx_t L4::Kobject::cap () const [inline], [protected], [noexcept]
```

Return capability selector.

Returns

Capability selector.

This method is for derived classes to gain access to the actual capability selector.

Definition at line 69 of file [kobject](#).

Referenced by [L4::Thread_group::add\(\)](#), [L4::Debugger::add_image_info\(\)](#), [L4::Task::add_ku_mem\(\)](#), [L4Re::Mem_alloc::alloc\(\)](#), [L4vbus::Vbus::assign_dma_domain\(\)](#), [L4vbus::Vbus::assign_dma_domain\(\)](#), [L4::lcu::bind\(\)](#), [L4::Rcv_endpoint::bind_snd_destination\(\)](#), [L4::Irq::bind_vcpu\(\)](#), [L4::Task::cap_equal\(\)](#), [L4::Task::cap_valid\(\)](#), [L4::Thread::control\(\)](#), [L4::Factory::create\(\)](#), [L4::Factory::create\(\)](#), [L4::Factory::create_factory\(\)](#), [L4::Factory::create_gate\(\)](#), [L4::Factory::create_task\(\)](#), [L4::Factory::create_thread_group\(\)](#), [L4Re::Video::Goos::create_view\(\)](#), [dec_refcnt\(\)](#), [L4::Task::delete_obj\(\)](#), [L4::Irq::detach\(\)](#), [L4::Semaphore::down\(\)](#), [L4::Thread::ex_regs\(\)](#), [L4::Thread::ex_regs\(\)](#), [L4::Vcon::get_attr\(\)](#), [L4::Debugger::get_object_name\(\)](#), [L4::Debugger::global_id\(\)](#), [L4::lcu::info\(\)](#), [L4::Scheduler::is_online\(\)](#), [L4::Debugger::kobj_to_id\(\)](#), [L4::Task::map\(\)](#), [L4::lcu::mask\(\)](#), [L4::Thread::modify_senders\(\)](#), [L4::Debugger::query_log_name\(\)](#), [L4::Debugger::query_log_typeid\(\)](#), [L4::Debugger::query_object_name\(\)](#), [L4::Vcon::read\(\)](#), [L4::Vcon::read_with_flags\(\)](#), [L4::Irq::receive\(\)](#), [L4::Thread::register_del_irq\(\)](#), [L4::Thread::register_doorbell_irq\(\)](#), [L4::Task::release_cap\(\)](#), [L4vbus::Vbus::release_ioport\(\)](#), [L4::Thread_group::remove\(\)](#), [L4vbus::Vbus::request_ioport\(\)](#), [L4vbus::Vbus::root\(\)](#), [L4::Vcon::send\(\)](#), [L4::Vcon::set_attr\(\)](#), [L4::lcu::set_mode\(\)](#), [L4::Debugger::set_object_name\(\)](#), [L4::Thread::stats_time\(\)](#), [L4::Debugger::switch_log\(\)](#), [L4::Thread::switch_to\(\)](#), [L4::Triggerable::trigger\(\)](#), [L4::lcu::unbind\(\)](#), [L4::Task::unmap\(\)](#), [L4::Task::unmap_batch\(\)](#), [L4::Irq_eoi::unmask\(\)](#), [L4::Thread::vcpu_control\(\)](#), [L4::Thread::vcpu_control_ext\(\)](#), [L4::Thread::vcpu_resume_commit\(\)](#), [L4Re::Video::Goos::view\(\)](#), and [L4::Vcon::write\(\)](#).

[illegible]

```

14_msgtag_t L4::Kobject::dec_refcnt (
    14_mword_t diff,
    14_utcb_t * utcb = 14_utcb()) [inline]

```

Decrement the in kernel reference counter for the object.

Parameters

| | |
|-------------|--|
| <i>diff</i> | The delta that shall be subtracted from the reference count. |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |

Returns

Syscall return tag

This function is intended for servers to be able to remove the servers own capability from the counted references. This leads to the semantics that the kernel will delete the object even if the capability of the server is valid. The server can detect the deletion by polling its capabilities or by using the IPC-gate deletion IRQs. And to cleanup if the clients dropped the last reference (capability) to the object.

This function only succeeds on a kernel object of type [L4::lpc_gate](#) which has the server right ([L4_FPAGE_C_IPCGATE_SVR](#)). For other kernel objects, -L4_ENOSYS is returned.

Definition at line 100 of file [kobject](#).

References [cap\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

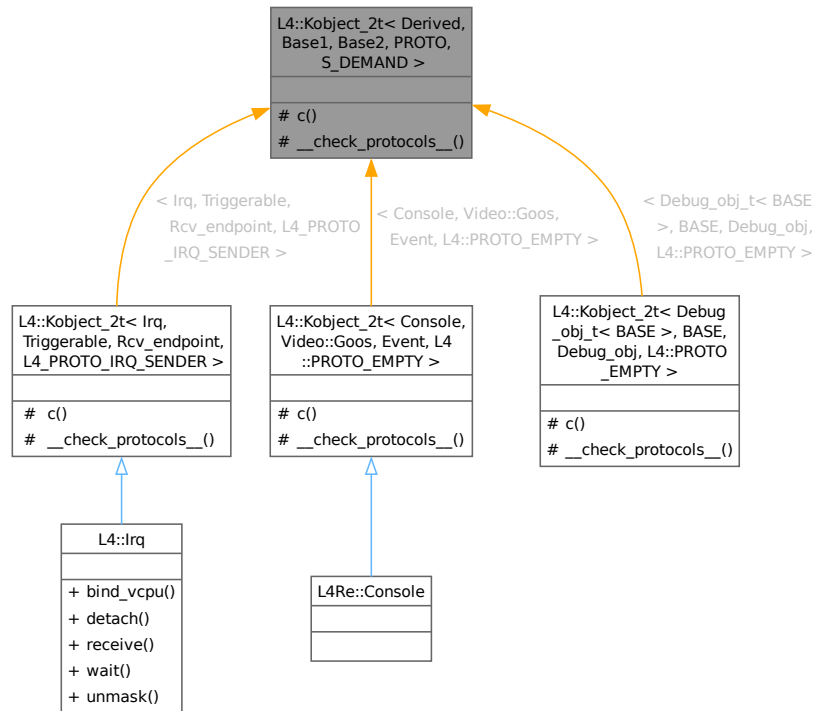
- `l4/sys/kobject`

15.169 L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND > Class Template Reference

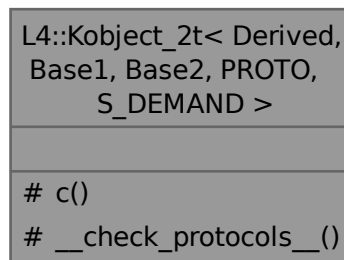
Helper class to create an [L4Re](#) interface class that is derived from two base classes (see [L4::Kobject_t](#)).

```
#include <l4/sys/capability>
```


Inheritance diagram for L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >:



Collaboration diagram for L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >:



Protected Types

- typedef Derived [Class](#)
The target interface type (inheriting from [Kobject_t](#)).
- typedef Typeid::Iface< PROTO, Derived > [__Iface](#)
The interface description for the derived class.
- typedef Typeid::Merge_list< Typeid::Iface_list< [__Iface](#) >, Typeid::Merge_list< typename Base1::__Iface<__Iface_list, typename Base2::__Iface_list > > > [__Iface_list](#)
The list of all RPC interfaces provided directly or through inheritance.

Protected Member Functions

- `L4::Cap< Class > c ()` const noexcept
Get the capability to ourselves.

Static Protected Member Functions

- static void `__check_protocols__ ()` noexcept
Helper to check for protocol conflicts.

15.169.1 Detailed Description

```
template<typename Derived, typename Base1, typename Base2, l4_proto_t PROTO = PROTO_ANY, type-
name S_DEMAND = Type_info::Demand_t<>>
class L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >
```

Helper class to create an [L4Re](#) interface class that is derived from two base classes (see [L4::Kobject_t](#)).

Template Parameters

| | |
|-----------------|---|
| <i>Derived</i> | is the name of the new interface. |
| <i>Base1</i> | is the name of the interface's first base class. |
| <i>Base2</i> | is the name of the interface's second base class. |
| <i>PROTO</i> | may be set to the statically assigned protocol number used to communicate with this interface. |
| <i>S_DEMAND</i> | type defining the demand of server-side resources for this interface, usually a L4::Type_info::Demand_t . This value must describe the server-side resources needed by the interface itself, the resource demand of the base interfaces (Base1 and Base2) are automatically included. |

The typical usage pattern is shown in the following code snippet. The semantics of this example is an interface `My_iface` that is derived from [L4::Icu](#) and [L4Re::Dataspace](#).

```
class My_iface : public L4::Kobject_2t<My_iface, L4::Icu, L4Re::Dataspace>
{
    ...
};
```

Definition at line 827 of file [__typeinfo.h](#).

15.169.2 Member Typedef Documentation

15.169.2.1 `__Iface`

```
template<typename Derived, typename Base1, typename Base2, l4_proto_t PROTO = PROTO_ANY, typename
S_DEMAND = Type_info::Demand_t<>>
typedef Typeid::Iface<PROTO, Derived> L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND
>::__Iface [protected]
```

The interface description for the derived class.

Definition at line 833 of file [__typeinfo.h](#).

15.169.2.2 __Iface_list

```
template<typename Derived, typename Base1, typename Base2, l4_proto_t PROTO = PROTO_ANY, typename
S_DEMAND = Type_info::Demand_t<>>
typedef Typeid::Merge_list< Typeid::Iface_list<__Iface>, Typeid::Merge_list< typename Base1↵
::__Iface_list, typename Base2::__Iface_list > > L4::Kobject_2t< Derived, Base1, Base2, PROTO,
S_DEMAND >::__Iface_list [protected]
```

The list of all RPC interfaces provided directly or through inheritance.

Definition at line 841 of file [__typeinfo.h](#).

15.169.2.3 Class

```
template<typename Derived, typename Base1, typename Base2, l4_proto_t PROTO = PROTO_ANY, typename
S_DEMAND = Type_info::Demand_t<>>
typedef Derived L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >::Class [protected]
```

The target interface type (inheriting from [Kobject_t](#)).

Definition at line 831 of file [__typeinfo.h](#).

15.169.3 Member Function Documentation

15.169.3.1 __check_protocols__()

```
template<typename Derived, typename Base1, typename Base2, l4_proto_t PROTO = PROTO_ANY, typename
S_DEMAND = Type_info::Demand_t<>>
void L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >::__check_protocols__ () [inline],
[static], [protected], [noexcept]
```

Helper to check for protocol conflicts.

Definition at line 844 of file [__typeinfo.h](#).

15.169.3.2 c()

```
template<typename Derived, typename Base1, typename Base2, l4_proto_t PROTO = PROTO_ANY, typename
S_DEMAND = Type_info::Demand_t<>>
L4::Cap< Class > L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >::c () const [inline],
[protected], [noexcept]
```

Get the capability to ourselves.

Definition at line 863 of file [__typeinfo.h](#).

The documentation for this class was generated from the following file:

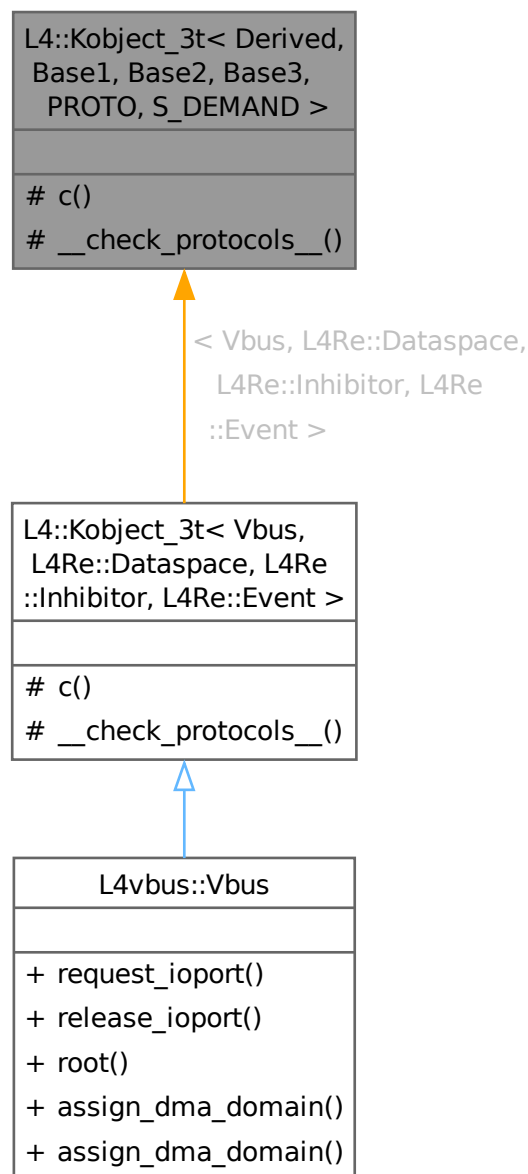
- [l4/sys/__typeinfo.h](#)

15.170 L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND > Struct Template Reference

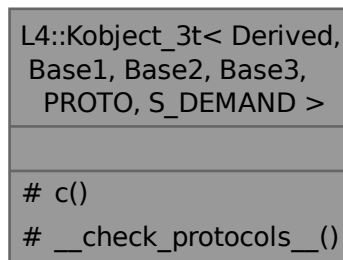
Helper class to create an [L4Re](#) interface class that is derived from three base classes (see [L4::Kobject_t](#)).

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >:



Collaboration diagram for L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >:



Protected Types

- typedef Derived [Class](#)
The target interface type (inheriting from [Kobject_t](#)).
- typedef Typeid::Iface< PROTO, Derived > [__Iface](#)
The interface description for the derived class.
- typedef Typeid::Merge_list< Typeid::Iface_list< [__Iface](#) >, Typeid::Merge_list< typename Base1::__Iface_list, Typeid::Merge_list< typename Base2::__Iface_list, typename Base3::__Iface_list > > > [__Iface_list](#)
The list of all RPC interfaces provided directly or through inheritance.

Protected Member Functions

- [L4::Cap](#)< [Class](#) > [c](#) () const noexcept
Get the capability to ourselves.

Static Protected Member Functions

- static void [__check_protocols__](#) () noexcept
Helper to check for protocol conflicts.

15.170.1 Detailed Description

```
template<typename Derived, typename Base1, typename Base2, typename Base3, l4\_proto\_t PROTO =
PROTO_ANY, typename S_DEMAND = Type_info::Demand_t<>>
struct L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >
```

Helper class to create an [L4Re](#) interface class that is derived from three base classes (see [L4 : Kobject_t](#)).

Template Parameters

| | |
|-----------------|---|
| <i>Derived</i> | is the name of the new interface. |
| <i>Base1</i> | is the name of the interface's first base class. |
| <i>Base2</i> | is the name of the interface's second base class. |
| <i>Base3</i> | is the name of the interfaces third base class. |
| <i>PROTO</i> | may be set to the statically assigned protocol number used to communicate with this interface. |
| <i>S_DEMAND</i> | type defining the demand on server-side resources for this interface, usually a L4::Type_info::Demand_t . This value must describe the server-side resources needed by the interface itself, the resource demand of the base interfaces (Base1 and Base2) are automatically included. |

See also

[L4::Kobject_t](#), [L4::Kobject_2t](#), [L4::Kobject_0t](#), [L4::Kobject_x](#)

Definition at line 930 of file [__typeinfo.h](#).

15.170.2 Member Typedef Documentation

15.170.2.1 __Iface

```
template<typename Derived, typename Base1, typename Base2, typename Base3, l4\_proto\_t PROTO =
PROTO_ANY, typename S_DEMAND = Type_info::Demand_t<>>
typedef Typeid::Iface<PROTO, Derived> L4::Kobject\_3t< Derived, Base1, Base2, Base3, PROTO,
S_DEMAND >::__Iface [protected]
```

The interface description for the derived class.

Definition at line 936 of file [__typeinfo.h](#).

15.170.2.2 __Iface_list

```
template<typename Derived, typename Base1, typename Base2, typename Base3, l4\_proto\_t PROTO =
PROTO_ANY, typename S_DEMAND = Type_info::Demand_t<>>
typedef Typeid::Merge_list< Typeid::Iface_list<\_\_Iface>, Typeid::Merge_list< typename Base1↔
::__Iface_list, Typeid::Merge_list< typename Base2::__Iface_list, typename Base3::__Iface↔
_list > > > L4::Kobject\_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >::__Iface_list
[protected]
```

The list of all RPC interfaces provided directly or through inheritance.

Definition at line 947 of file [__typeinfo.h](#).

15.170.2.3 Class

```
template<typename Derived, typename Base1, typename Base2, typename Base3, l4\_proto\_t PROTO =
PROTO_ANY, typename S_DEMAND = Type_info::Demand_t<>>
typedef Derived L4::Kobject\_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >::Class [protected]
```

The target interface type (inheriting from [Kobject_t](#)).

Definition at line 934 of file [__typeinfo.h](#).

15.170.3 Member Function Documentation

15.170.3.1 __check_protocols__()

```
template<typename Derived, typename Base1, typename Base2, typename Base3, l4_proto_t PROTO =
PROTO_ANY, typename S_DEMAND = Type_info::Demand_t<>>
void L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >::__check_protocols__ ()
[inline], [static], [protected], [noexcept]
```

Helper to check for protocol conflicts.

Definition at line 950 of file [__typeinfo.h](#).

15.170.3.2 c()

```
template<typename Derived, typename Base1, typename Base2, typename Base3, l4_proto_t PROTO =
PROTO_ANY, typename S_DEMAND = Type_info::Demand_t<>>
L4::Cap< Class > L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >::c () const
[inline], [protected], [noexcept]
```

Get the capability to ourselves.

Definition at line 978 of file [__typeinfo.h](#).

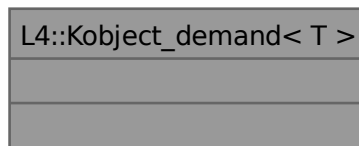
The documentation for this struct was generated from the following file:

- [l4/sys/__typeinfo.h](#)

15.171 L4::Kobject_demand< T > Struct Template Reference

Get the combined server-side resource requirements for all type T...

Collaboration diagram for L4::Kobject_demand< T >:



15.171.1 Detailed Description

```
template<typename ... T>
struct L4::Kobject_demand< T >
```

Get the combined server-side resource requirements for all type T...

Template Parameters

| | |
|----------|---|
| <i>T</i> | List of IPC interface types for which the combined server-side resource requirements shall be calculated. |
|----------|---|

Definition at line 1031 of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

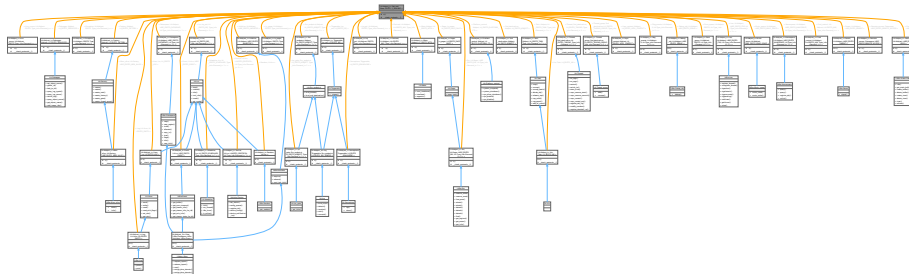
- [l4/sys/__typeinfo.h](#)

15.172 L4::Kobject_t< Derived, Base, PROTO, S_DEMAND > Class Template Reference

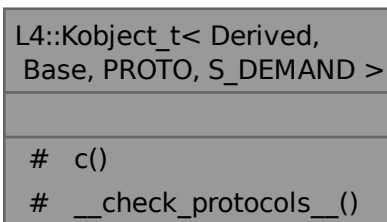
Helper class to create an [L4Re](#) interface class that is derived from a single base class.

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Kobject_t< Derived, Base, PROTO, S_DEMAND >:



Collaboration diagram for L4::Kobject_t< Derived, Base, PROTO, S_DEMAND >:



Protected Types

- typedef Derived **Class**
The target interface type (inheriting from [Kobject_t](#)).
- typedef Typeid::Iface< PROTO, Derived > **__Iface**
The interface description for the derived class.
- typedef Typeid::Merge_list< Typeid::Iface_list< **__Iface** >, typename Base::__Iface_list > **__Iface_list**
The list of all RPC interfaces provided directly or through inheritance.

Protected Member Functions

- [L4::Cap](#)< **Class** > **c** () const noexcept
Get the capability to ourselves.

Static Protected Member Functions

- static void **__check_protocols** () noexcept
Helper to check for protocol conflicts.

15.172.1 Detailed Description

```
template<typename Derived, typename Base, l4\_proto\_t PROTO = PROTO_ANY, typename S_DEMAND =
Type_info::Demand_t<>>
class L4::Kobject_t< Derived, Base, PROTO, S_DEMAND >
```

Helper class to create an [L4Re](#) interface class that is derived from a single base class.

Template Parameters

| | |
|-----------------|---|
| <i>Derived</i> | is the name of the new interface. |
| <i>Base</i> | is the name of the interfaces single base class. |
| <i>PROTO</i> | may be set to the statically assigned protocol number used to communicate with this interface. |
| <i>S_DEMAND</i> | type defining the demand on server-side resources for this interface, usually a L4::Type_info::Demand_t . This value must describe the server-side resources needed by the interface itself, the resource demand of the base interface <i>Base</i> is automatically included. |

The typical usage pattern is shown in the following code snippet. The semantics of this example is an interface `My_iface` that is derived from [L4::Kobject](#).

```
class My_iface : public L4::Kobject_t<My_iface, L4::Kobject>
{
    ...
};
```

Examples

[examples/clntsrv/src/shared.h](#).

Definition at line 749 of file [__typeinfo.h](#).

The documentation for this class was generated from the following file:

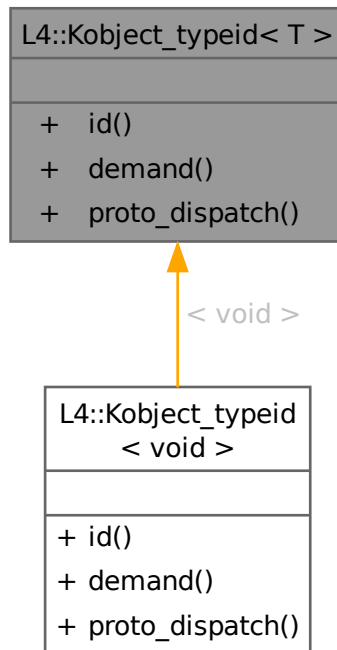
- [l4/sys/__typeinfo.h](#)

15.173 L4::Kobject_typeid< T > Struct Template Reference

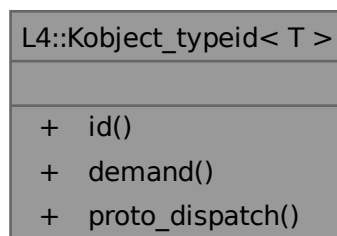
[Meta](#) object for handling access to type information of Kobjects.

```
#include <__typeinfo.h>
```

Inheritance diagram for L4::Kobject_typeid< T >:



Collaboration diagram for L4::Kobject_typeid< T >:



Public Types

- typedef T::__Kobject_typeid::Demand [Demand](#)

Data type expressing the static demand of receive buffers in a server.

Static Public Member Functions

- static [Type_info](#) const * [id](#) () noexcept
Get a pointer to the [Kobject](#) type information of T.
- static [Type_info::Demand](#) [demand](#) () noexcept
Get the receive-buffer demand for the server providing the interface T.
- template<typename THIS, typename A1, typename A2>
static int [proto_dispatch](#) (THIS *self, [i4_proto_t](#) proto, A1 a1, A2 &a2)
Protocol based server-side dispatch function.

15.173.1 Detailed Description

```
template<typename T>
struct L4::Kobject_typeid< T >
```

[Meta](#) object for handling access to type information of Kobjects.

Template Parameters

| | |
|----------|--|
| <i>T</i> | The data type derived from Kobject , usually using Kobject_t . |
|----------|--|

Definition at line 610 of file [__typeinfo.h](#).

15.173.2 Member Typedef Documentation

15.173.2.1 Demand

```
template<typename T>
typedef T::__Kobject_typeid::Demand L4::Kobject\_typeid< T >::Demand
```

Data type expressing the static demand of receive buffers in a server.

This information is the combined demand of all base interfaces for T and the buffer demand of T itself. The buffer demand of T is usually specified as the S_DEMAND argument of the [Kobject_t](#) or [Kobject_2t](#) inheritance helpers. S_DEMAND is usually of type [L4::Type_info::Demand_t](#), or [L4::Type_info::Demand_union_t](#).

Definition at line 622 of file [__typeinfo.h](#).

15.173.3 Member Function Documentation

15.173.3.1 demand()

```
template<typename T>
Type_info::Demand L4::Kobject_typeid< T >::demand () [inline], [static], [noexcept]
```

Get the receive-buffer demand for the server providing the interface T.

Returns

A demand value describing the minimum receive buffers needed for handling server side requests for interface T.

Definition at line 639 of file [__typeinfo.h](#).

15.173.3.2 id()

```
template<typename T>
Type_info const * L4::Kobject_typeid< T >::id () [inline], [static], [noexcept]
```

Get a pointer to the [Kobject](#) type information of T.

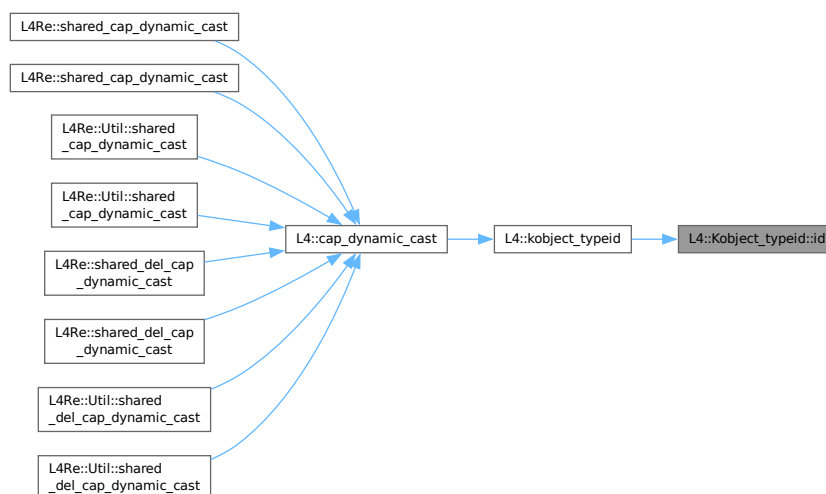
Returns

a pointer to the [Kobject](#) typeinfo of T.

Definition at line 630 of file [__typeinfo.h](#).

Referenced by [L4::kobject_typeid\(\)](#).

Here is the caller graph for this function:



15.173.3.3 proto_dispatch()

```

template<typename T>
template<typename THIS, typename A1, typename A2>
int L4::Kobject_typeid< T >::proto_dispatch (
    THIS * self,
    l4_proto_t proto,
    A1 a1,
    A2 & a2) [inline], [static]

```

Protocol based server-side dispatch function.

Template Parameters

| | |
|-------------|---|
| <i>THIS</i> | Data type of the server-side object implementing the interface T. |
| <i>A1</i> | Data type of second argument for p_dispatch() |
| <i>A2</i> | Data type of third argument for p_dispatch() |

Parameters

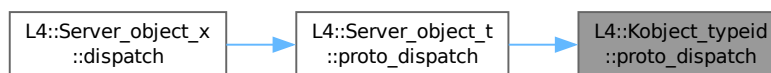
| | |
|--------------|--|
| <i>self</i> | The pointer to the server object |
| <i>proto</i> | The protocol number used by the caller |
| <i>a1</i> | The second argument passed to self->p_dispatch() |
| <i>a2</i> | The third argument passed to self->p_dispatch() |

This function forwards the call to the overloaded p_dispatch() function of self. The data type of the first argument for p_dispatch is determined by the given protocol number.

Definition at line 660 of file [__typeinfo.h](#).

Referenced by [L4::Server_object_t< IFACE, BASE >::proto_dispatch\(\)](#).

Here is the caller graph for this function:



The documentation for this struct was generated from the following file:

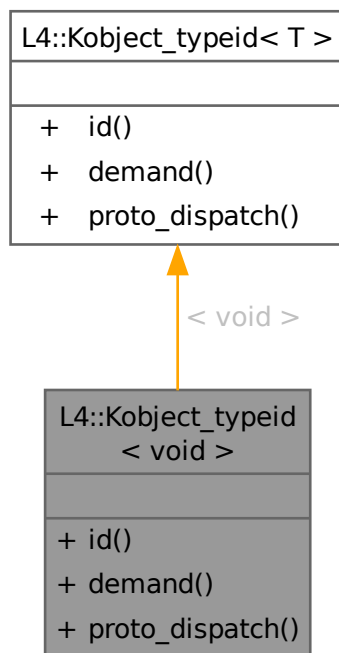
- [l4/sys/__typeinfo.h](#)

15.174 L4::Kobject_typeid< void > Struct Reference

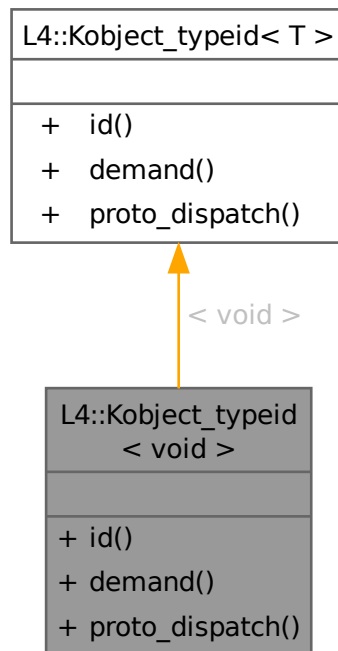
Minimalistic ID for void interface.

```
#include <__typeinfo.h>
```

Inheritance diagram for L4::Kobject_typeid< void >:



Collaboration diagram for L4::Kobject_typeid< void >:



Static Public Member Functions

- static `Type_info` const * `id` () noexcept
Get a pointer to the *Kobject* type information of *T*.
- static `Type_info::Demand` `demand` () noexcept
Get the receive-buffer demand for the server providing the interface *T*.
- static int `proto_dispatch` (THIS *self, `l4_proto_t` proto, A1 a1, A2 &a2)
Protocol based server-side dispatch function.

15.174.1 Detailed Description

Minimalistic ID for `void` interface.

Definition at line 667 of file `__typeinfo.h`.

15.174.2 Member Function Documentation

15.174.2.1 demand()

```
Type_info::Demand L4::Kobject_typeid< void >::demand () [inline], [static], [noexcept]
```

Get the receive-buffer demand for the server providing the interface *T*.

Returns

A demand value describing the minimum receive buffers needed for handling server side requests for interface T.

Definition at line 639 of file [__typeinfo.h](#).

15.174.2.2 id()

```
Type_info const * L4::Kobject_typeid< void >::id () [inline], [static], [noexcept]
```

Get a pointer to the [Kobject](#) type information of T.

Returns

a pointer to the [Kobject](#) typeinfo of T.

Definition at line 630 of file [__typeinfo.h](#).

15.174.2.3 proto_dispatch()

```
int L4::Kobject_typeid< void >::proto_dispatch (
    THIS * self,
    l4_proto_t proto,
    A1 a1,
    A2 & a2) [inline], [static]
```

Protocol based server-side dispatch function.

Template Parameters

| | |
|-------------|---|
| <i>THIS</i> | Data type of the server-side object implementing the interface T. |
| <i>A1</i> | Data type of second argument for p_dispatch() |
| <i>A2</i> | Data type of third argument for p_dispatch() |

Parameters

| | |
|--------------|--|
| <i>self</i> | The pointer to the server object |
| <i>proto</i> | The protocol number used by the caller |
| <i>a1</i> | The second argument passed to self->p_dispatch() |
| <i>a2</i> | The third argument passed to self->p_dispatch() |

This function forwards the call to the overloaded p_dispatch() function of self. The data type of the first argument for p_dispatch is determined by the given protocol number.

Definition at line 660 of file [__typeinfo.h](#).

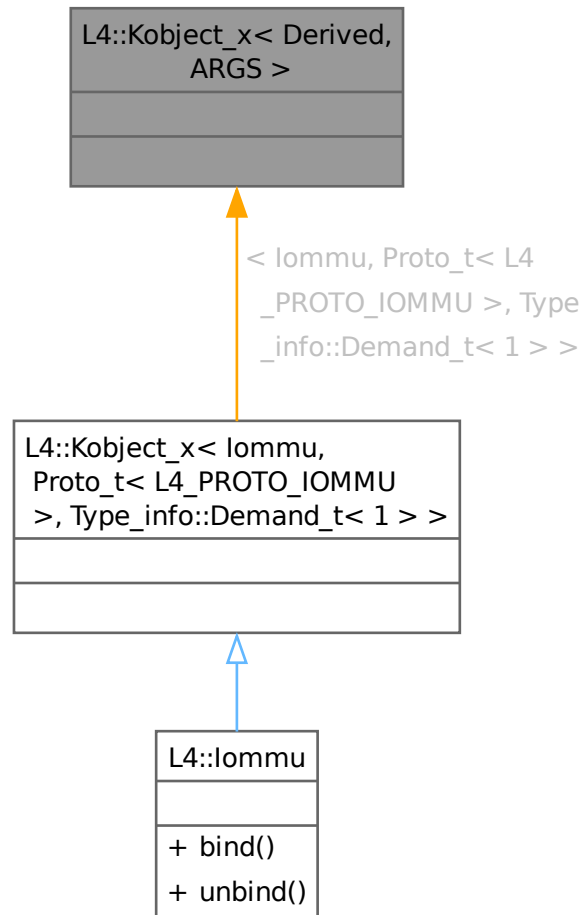
The documentation for this struct was generated from the following file:

- [l4/sys/__typeinfo.h](#)

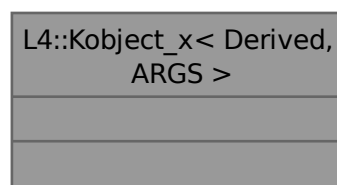
15.175 L4::Kobject_x< Derived, ARGS > Struct Template Reference

Generic [Kobject](#) inheritance template.

Inheritance diagram for L4::Kobject_x< Derived, ARGS >:



Collaboration diagram for L4::Kobject_x< Derived, ARGS >:



15.175.1 Detailed Description

```
template<typename Derived, typename ... ARGS>
struct L4::Kobject_x< Derived, ARGS >
```

Generic [Kobject](#) inheritance template.

Template Parameters

| | |
|----------------|---|
| <i>Derived</i> | The class name that derives from Kobject_x . |
| <i>ARGS</i> | An optional protocol number via L4::Proto_t , followed by an optional server-side requirement passed as L4::Type_info::Demand_t , followed by the list of base classes. |

Definition at line 1197 of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

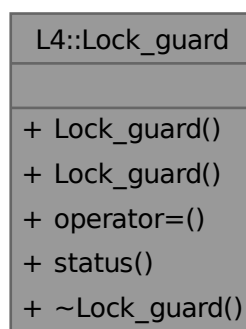
- [l4/sys/__typeinfo.h](#)

15.176 L4::Lock_guard Class Reference

Basic lock guard implementation that prevents forgotten unlocks on exit paths from a method or a block of code.

```
#include <lock_guard.h>
```

Collaboration diagram for L4::Lock_guard:



Public Member Functions

- [Lock_guard](#) (pthread_mutex_t &lock)
Construct the lock guard and lock the associated mutex.
- [Lock_guard](#) (Lock_guard &&guard)
Move constructor from other lock guard.
- Lock_guard & [operator=](#) (Lock_guard &&guard)
Move assignment from other lock guard.
- int [status](#) () const
Return last lock/unlock operation error status.
- [~Lock_guard](#) ()
Lock guard destructor.

15.176.1 Detailed Description

Basic lock guard implementation that prevents forgotten unlocks on exit paths from a method or a block of code.

Targeting pthread_mutex_t.

An instance of lock guard cannot be copied, but it can be moved.

The typical usage pattern of the lock guard is:

```
pthread_mutex_t mtx = PTHREAD_MUTEX_INITIALIZER;

{
    auto guard = L4Re::Lock_guard(mtx);

    // Correctness check.
    assert(guard.status() == 0);

    // Critical section protected by mtx.

    // The mtx is automatically unlocked when guard goes out of scope.
}
```

Definition at line 44 of file [lock_guard.h](#).

15.176.2 Constructor & Destructor Documentation

15.176.2.1 Lock_guard() [1/2]

```
L4::Lock_guard::Lock_guard (
    pthread_mutex_t & lock) [inline], [explicit]
```

Construct the lock guard and lock the associated mutex.

The error condition of the locking operation can be checked by the [status\(\)](#) method.

Parameters

| | |
|-------------|--------------------------------|
| <i>lock</i> | Associated mutex to be locked. |
|-------------|--------------------------------|

Definition at line 59 of file [lock_guard.h](#).

15.176.2.2 Lock_guard() [2/2]

```
L4::Lock_guard::Lock_guard (
    Lock_guard && guard) [inline]
```

Move constructor from other lock guard.

The mutex associated with the other lock guard is kept locked.

Parameters

| | |
|--------------|-------------------------|
| <i>guard</i> | Lock guard to be moved. |
|--------------|-------------------------|

Definition at line 71 of file [lock_guard.h](#).

15.176.2.3 ~Lock_guard()

```
L4::Lock_guard::~~Lock_guard () [inline]
```

Lock guard destructor.

The associated mutex (if any) is unlocked.

There is no mechanism for indicating any error conditions. However, if the mutex has been previously locked successfully by this class and if the implementation of the mutex behaves according to the POSIX specification, the construction of this class guarantees that the unlock operation does not fail.

Definition at line 126 of file [lock_guard.h](#).

15.176.3 Member Function Documentation

15.176.3.1 operator=()

```
Lock_guard & L4::Lock_guard::operator= (
    Lock_guard && guard) [inline]
```

Move assignment from other lock guard.

The mutex currently associated with this lock guard is unlocked. The mutex associated with the other lock guard is kept locked.

There is no mechanism for indicating any error conditions of the unlocking operation. However, if the mutex has been previously locked successfully by this class and if the implementation of the mutex behaves according to the POSIX specification, the construction of this class guarantees that the unlock operation does not fail.

Parameters

| | |
|--------------|-------------------------|
| <i>guard</i> | Lock guard to be moved. |
|--------------|-------------------------|

Definition at line 90 of file [lock_guard.h](#).

15.176.3.2 status()

```
int L4::Lock_guard::status () const [inline]
```

Return last lock/unlock operation error status.

Returns

Zero indicating no errors, any other value indicating an error.

Definition at line 110 of file [lock_guard.h](#).

The documentation for this class was generated from the following file:

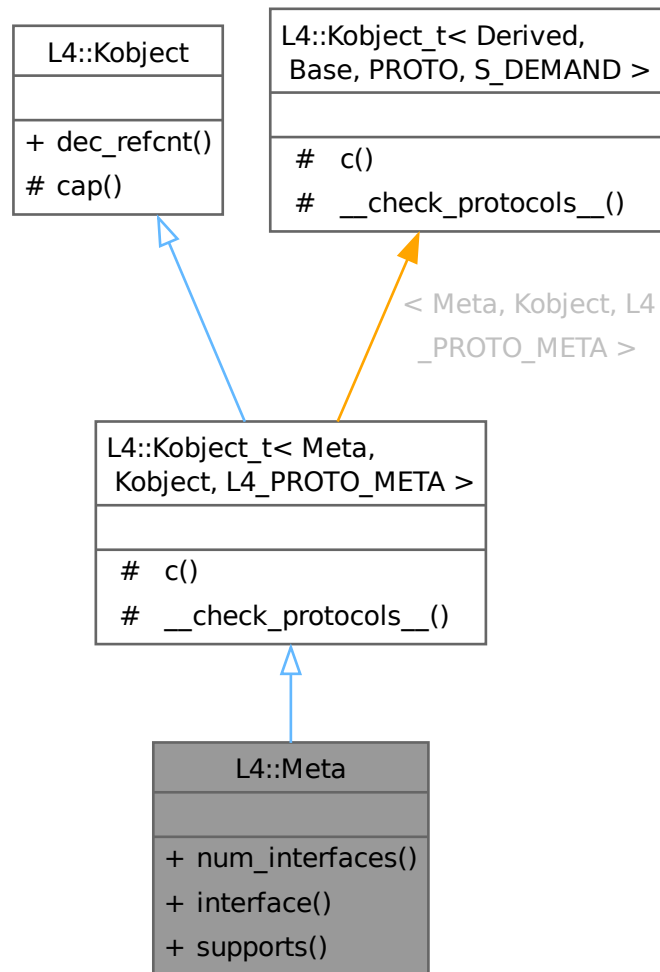
- [l4/cxx/lock_guard.h](#)

15.177 L4::Meta Class Reference

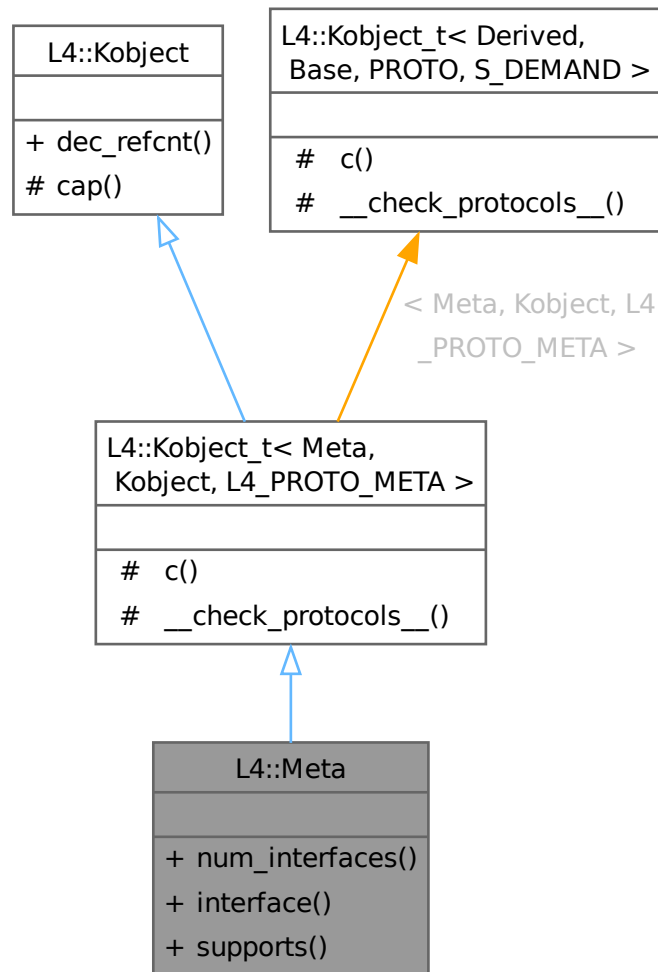
[Meta](#) interface that shall be implemented by each [L4Re](#) object and gives access to the dynamic type information for [L4Re](#) objects.

```
#include <meta>
```

Inheritance diagram for L4::Meta:



Collaboration diagram for L4::Meta:



Public Member Functions

- [l4_msgtag_t num_interfaces \(\)](#)
Get the number of interfaces implemented by this object.
- [l4_msgtag_t interface \(l4_umword_t idx, long *proto, L4::lpc::String< char > *name\)](#)
Get the protocol number that must be used for the interface with the number *idx*.
- [l4_msgtag_t supports \(l4_mword_t protocol\)](#)
Figure out if the object supports the given protocol (number).

Public Member Functions inherited from [L4::Kobject](#)

- [l4_msgtag_t dec_refcnt \(l4_mword_t diff, l4_utcb_t *utcb=l4_utcb\(\)\)](#)
Decrement the in kernel reference counter for the object.

Additional Inherited Members

Protected Types inherited from [L4::Kobject_t< Meta, Kobject, L4_PROTO_META >](#)

- typedef [Meta](#) **Class**
The target interface type (inheriting from [Kobject_t](#)).
- typedef Typeid::Iface< PROTO, [Meta](#) > **__iface**
The interface description for the derived class.
- typedef Typeid::Merge_list< Typeid::Iface_list< **__iface** >, typename Kobject::__iface_list > **__iface_list**
The list of all RPC interfaces provided directly or through inheritance.

Protected Member Functions inherited from [L4::Kobject_t< Meta, Kobject, L4_PROTO_META >](#)

- [L4::Cap< Class > c](#) () const noexcept
Get the capability to ourselves.

Protected Member Functions inherited from [L4::Kobject](#)

- [l4_cap_idx_t](#) [cap](#) () const noexcept
Return capability selector.

Static Protected Member Functions inherited from [L4::Kobject_t< Meta, Kobject, L4_PROTO_META >](#)

- static void [__check_protocols__](#) () noexcept
Helper to check for protocol conflicts.

15.177.1 Detailed Description

[Meta](#) interface that shall be implemented by each [L4Re](#) object and gives access to the dynamic type information for [L4Re](#) objects.

Definition at line 26 of file [meta](#).

15.177.2 Member Function Documentation

15.177.2.1 interface()

```
l4_msgtag_t L4::Meta::interface (
    l4_umword_t idx,
    long * proto,
    L4::Ipc::String< char > * name)
```

Get the protocol number that must be used for the interface with the number `idx`.

Parameters

| | | |
|-----|--------------|--|
| | <i>idx</i> | The index of the interface to get the protocol number for. <code>idx</code> must be ≥ 0 and $<$ the return value of <code>num_interfaces()</code> . |
| out | <i>proto</i> | The protocol number for interface <code>idx</code> . |
| out | <i>name</i> | The protocol name for interface <code>idx</code> . |

Return values

| | |
|--|--|
| <code>l4_msgtag_t::label() == 0</code> | Successful; see <code>`proto`</code> and <code>`name`</code> . |
| <code>l4_msgtag_t::label() < 0</code> | Error code. |

Referenced by `num_interfaces()`.

Here is the caller graph for this function:



15.177.2.2 num_interfaces()

```
l4_msgtag_t L4::Meta::num_interfaces ()
```

Get the number of interfaces implemented by this object.

Return values

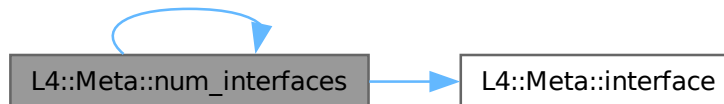
| | |
|---|-------------------------------------|
| <code>l4_msgtag_t::label() >= 0</code> | The number of supported interfaces. |
|---|-------------------------------------|

| | |
|--|-----------------------------------|
| <code>l4_msgtag_t::label() < 0</code> | Error code of the occurred error. |
|--|-----------------------------------|

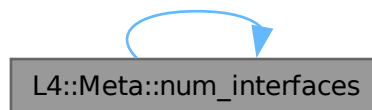
References [interface\(\)](#), and [num_interfaces\(\)](#).

Referenced by [num_interfaces\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.177.2.3 supports()

```
l4_msgtag_t L4::Meta::supports (
    l4_mword_t protocol)
```

Figure out if the object supports the given protocol (number).

Parameters

| | |
|-----------------|-----------------------------------|
| <i>protocol</i> | The protocol number to check for. |
|-----------------|-----------------------------------|

Return values

| | |
|--|----------------------------|
| <code>l4_msgtag_t::label() == 1</code> | protocol is supported. |
| <code>l4_msgtag_t::label() == 0</code> | protocol is not supported. |

This method is intended to be used for statically assigned protocol numbers.

References [supports\(\)](#).

Referenced by [supports\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

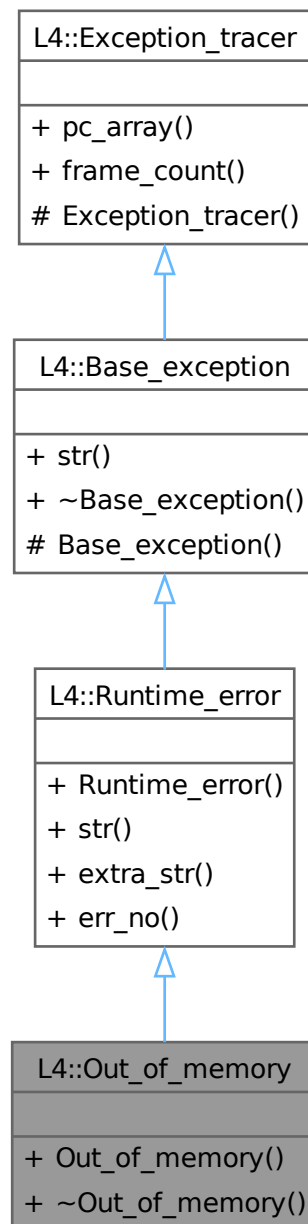
- `l4/sys/meta`

15.178 L4::Out_of_memory Class Reference

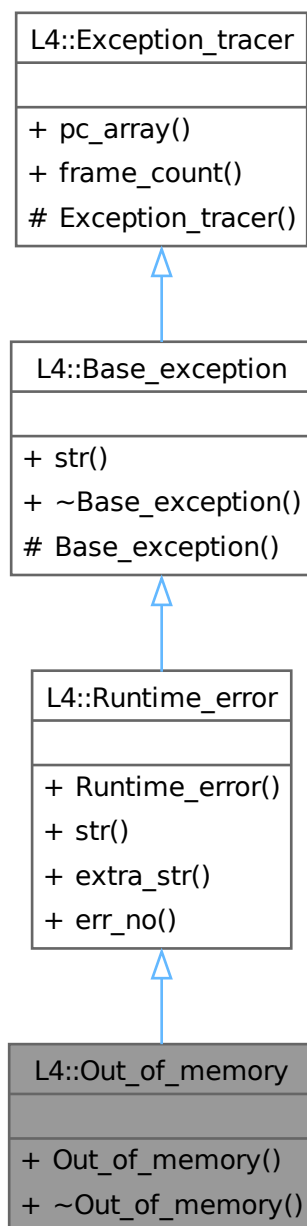
[Exception](#) signalling insufficient memory.

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Out_of_memory:



Collaboration diagram for L4::Out_of_memory:



Public Member Functions

- **Out_of_memory** (char const *extra="") noexcept
Create an out-of-memory exception.
- **~Out_of_memory** () noexcept
Destruction.

Public Member Functions inherited from [L4::Runtime_error](#)

- [Runtime_error](#) (long [err_no](#), char const *extra=0) noexcept
Create a new [Runtime_error](#).
- char const * [str](#) () const noexcept override
Return a human readable string for the exception.
- char const * [extra_str](#) () const
Get the description text for this runtime error.
- long [err_no](#) () const noexcept
Get the error value for this runtime error.

Public Member Functions inherited from [L4::Base_exception](#)

- virtual ~[Base_exception](#) () noexcept
Destruction.

Public Member Functions inherited from [L4::Exception_tracer](#)

- void const *const * [pc_array](#) () const noexcept
Get the array containing the call trace.
- int [frame_count](#) () const noexcept
Get the number of entries that are valid in the call trace.

Additional Inherited Members

Protected Member Functions inherited from [L4::Base_exception](#)

- [Base_exception](#) () noexcept
Create a base exception.

Protected Member Functions inherited from [L4::Exception_tracer](#)

- [Exception_tracer](#) () noexcept
Create a back trace.

15.178.1 Detailed Description

[Exception](#) signalling insufficient memory.

Definition at line 177 of file [exceptions](#).

The documentation for this class was generated from the following file:

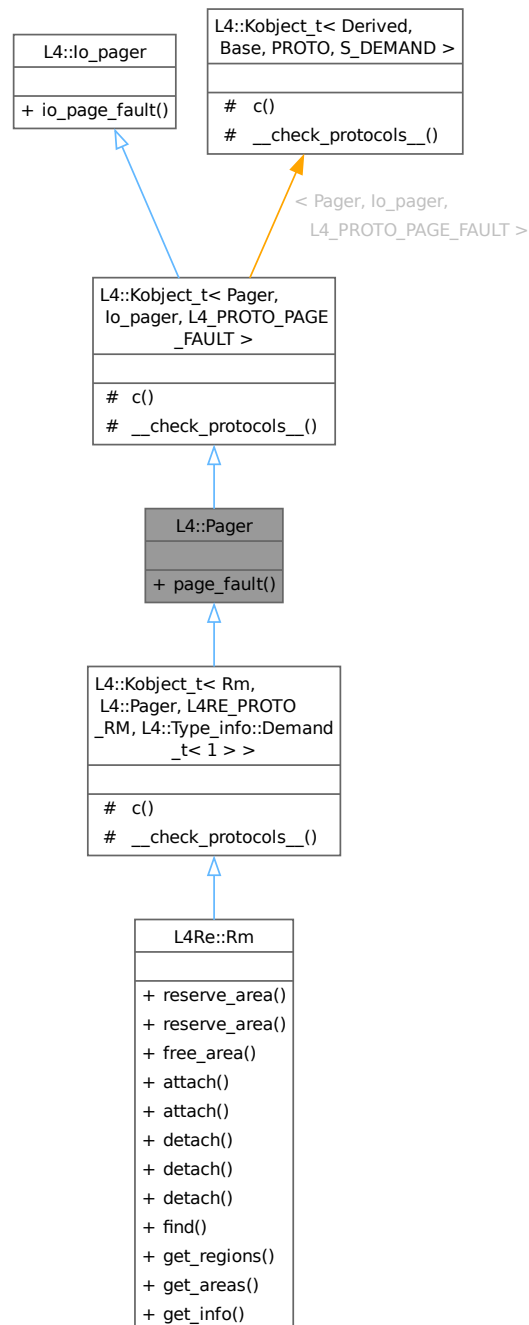
- [l4/cxx/exceptions](#)

15.179 L4::Pager Class Reference

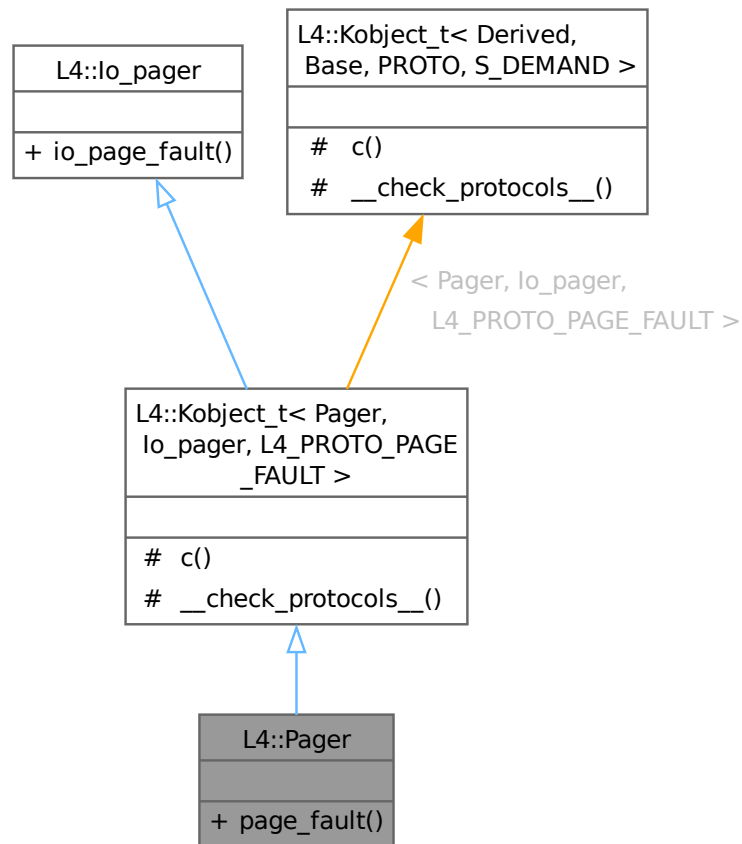
[Pager](#) interface including the [io_pager](#) interface.

```
#include <pager>
```

Inheritance diagram for L4::Pager:



Collaboration diagram for L4::Pager:



Public Member Functions

- `l4_msgtag_t page_fault (l4_umword_t pfa, l4_umword_t pc, L4::lpc::Rcv_fpage rwin, L4::lpc::Opt< L4::lpc::Snd_fpage & > fp)`

Page-fault protocol message.

Public Member Functions inherited from `L4::io_pager`

- `l4_msgtag_t io_page_fault (l4_fpage_t io_pfa, l4_umword_t pc, L4::lpc::Rcv_fpage rwin, L4::lpc::Opt< L4::lpc::Snd_fpage & > fp)`

IO page fault protocol message.

Additional Inherited Members

Protected Types inherited from

`L4::Kobject_t< Pager, io_pager, L4_PROTO_PAGE_FAULT >`

- typedef `Pager` **Class**

The target interface type (inheriting from [Kobject_t](#)).

- typedef Typeid::Iface< PROTO, [Pager](#) > __Iface

The interface description for the derived class.

- typedef Typeid::Merge_list< Typeid::Iface_list< __Iface >, typename lo_pager::__Iface_list > __Iface_list

The list of all RPC interfaces provided directly or through inheritance.

Protected Member Functions inherited from [L4::Kobject_t< Pager, lo_pager, L4_PROTO_PAGE_FAULT >](#)

- [L4::Cap< Class > c \(\)](#) const noexcept

Get the capability to ourselves.

Static Protected Member Functions inherited from [L4::Kobject_t< Pager, lo_pager, L4_PROTO_PAGE_FAULT >](#)

- static void [__check_protocols__ \(\)](#) noexcept

Helper to check for protocol conflicts.

15.179.1 Detailed Description

[Pager](#) interface including the [lo_pager](#) interface.

This class defines the interface for handling page fault IPC. If a thread causes a page fault, the microkernel synthesises a page fault IPC message and sends it to the thread's page fault handler (pager). The pager can then handle the message, for example by establishing a suitable page mapping.

The page fault handler is set with the [L4::Thread::control](#) interface.

Definition at line 87 of file [pager](#).

15.179.2 Member Function Documentation

15.179.2.1 [page_fault\(\)](#)

```
l4_msgtag_t L4::Pager::page_fault (
    l4_umword_t pfa,
    l4_umword_t pc,
    L4::Ipc::Rcv_fpage rwin,
    L4::Ipc::Opt< L4::Ipc::Snd_fpage & > fp)
```

Page-fault protocol message.

Parameters

| | |
|----------------------|---|
| pfa | Faulting address including failure reason: bits [0:2]. |
| pc | Faulting program counter. |
| rwin | Receive window for a flexpage mapping resolving the page fault. |

| | | |
|-----|----|---|
| out | fp | Optional: flexpage descriptor to send to the task raising the page fault. |
|-----|----|---|

Returns

System call message tag; use [l4_error\(\)](#) to check for errors.

Page-fault messages are usually generated by the kernel and need to be handled by an appropriate handler function, potentially filling in `fp` for the reply.

`pfa` encoding is as shown:

| [63/31 .. 3] | 2 | 1 | 0 |
|--------------|---|---|---|
| PFA | X | W | r |

- **PFA** Bits 63/31..3 of `pfa` are the page fault address bits 63/31 to 3, bits 2..0 are masked.
- **X** Bit 2 of `pfa` if set, indicates a page fault during instruction fetch. Note, this bit is implementation-defined and might always be clear. Therefore, if this bit is clear it does not imply that the page fault is not due to an instruction fetch.
- **W** Bit 1 of `pfa` is set to 1 for a page fault due to a write operation.
- **r** Bit0: reserved, undefined.

The documentation for this class was generated from the following file:

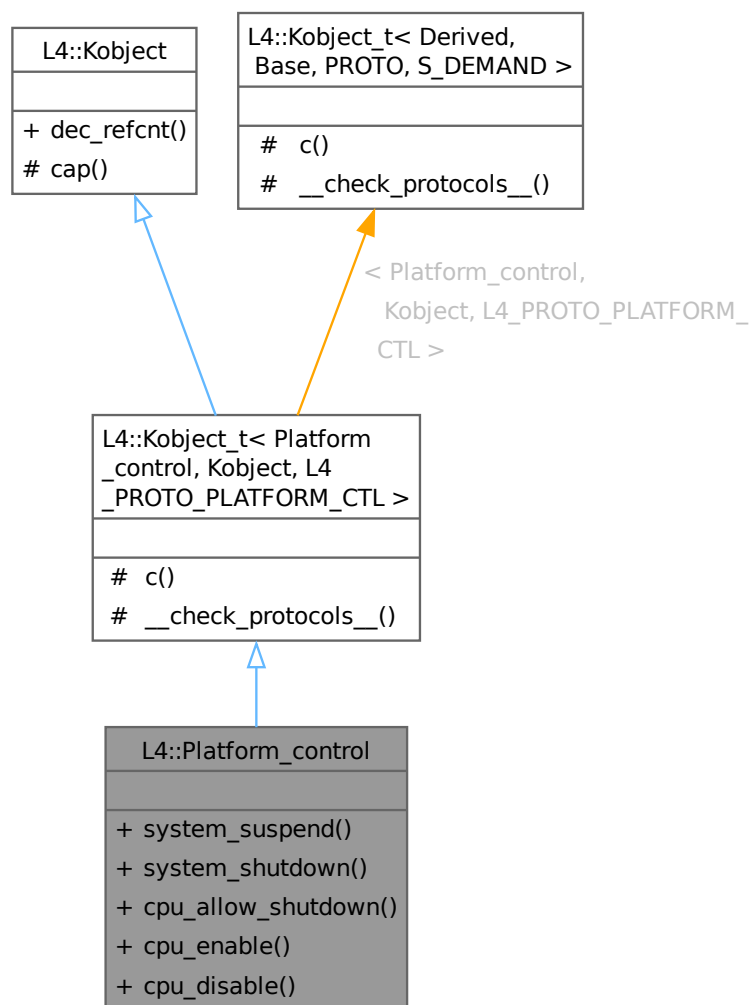
- [l4/sys/pager](#)

15.180 L4::Platform_control Class Reference

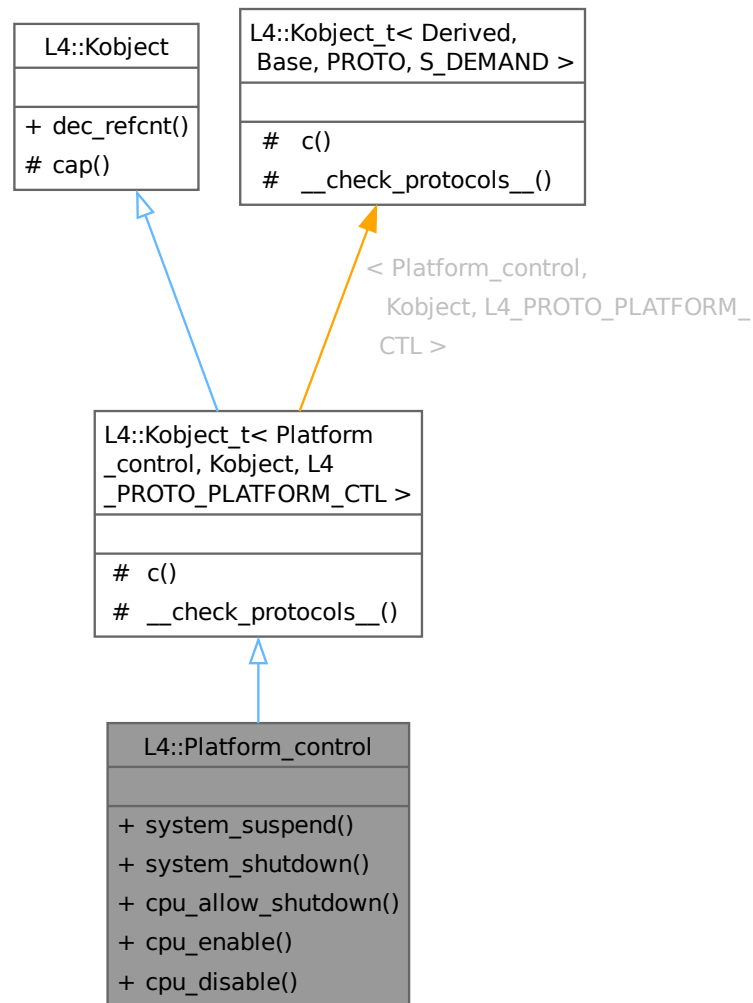
[L4](#) C++ interface for controlling platform-wide properties, see [Platform Control C API](#) for the C interface.

```
#include <platform_control>
```

Inheritance diagram for L4::Platform_control:



Collaboration diagram for L4::Platform_control:



Public Member Functions

- `l4_msgtag_t system_suspend (l4_umword_t extras)`
Enter suspend to RAM.
- `l4_msgtag_t system_shutdown (l4_umword_t reboot)`
Shutdown/Reboot the system.
- `l4_msgtag_t cpu_allow_shutdown (l4_umword_t phys_id, l4_umword_t enable)`
Allow CPU shutdown.
- `l4_msgtag_t cpu_enable (l4_umword_t phys_id)`
Enable an offline CPU.
- `l4_msgtag_t cpu_disable (l4_umword_t phys_id)`
Disable an online CPU.

Public Member Functions inherited from [L4::Kobject](#)

- [l4_msgtag_t dec_refcnt](#) ([l4_mword_t diff](#), [l4_utcb_t *utcb=l4_utcb\(\)](#))
Decrement the in kernel reference counter for the object.

Additional Inherited Members

Protected Types inherited from

[L4::Kobject_t](#)< [Platform_control](#), [Kobject](#), [L4_PROTO_PLATFORM_CTL](#) >

- typedef [Platform_control](#) **Class**
The target interface type (inheriting from [Kobject_t](#)).
- typedef [Typeid::Iface](#)< [PROTO](#), [Platform_control](#) > **__Iface**
The interface description for the derived class.
- typedef [Typeid::Merge_list](#)< [Typeid::Iface_list](#)< **__Iface** >, [typename Kobject::__Iface_list](#) > **__Iface_list**
The list of all RPC interfaces provided directly or through inheritance.

Protected Member Functions inherited from

[L4::Kobject_t](#)< [Platform_control](#), [Kobject](#), [L4_PROTO_PLATFORM_CTL](#) >

- [L4::Cap](#)< [Class](#) > **c** () const noexcept
Get the capability to ourselves.

Protected Member Functions inherited from [L4::Kobject](#)

- [l4_cap_idx_t cap](#) () const noexcept
Return capability selector.

Static Protected Member Functions inherited from

[L4::Kobject_t](#)< [Platform_control](#), [Kobject](#), [L4_PROTO_PLATFORM_CTL](#) >

- static void **__check_protocols** () noexcept
Helper to check for protocol conflicts.

15.180.1 Detailed Description

[L4](#) C++ interface for controlling platform-wide properties, see [Platform Control C API](#) for the C interface.

Add

```
#include <l4/sys/platform_control>
```

to your code to use the platform control functions. The API allows a client to suspend, reboot or shutdown the system.

For the C interface refer to the [Platform Control C API](#).

Definition at line 36 of file [platform_control](#).

15.180.2 Member Function Documentation

15.180.2.1 `cpu_allow_shutdown()`

```
l4_msgtag_t L4::Platform_control::cpu_allow_shutdown (
    l4_umword_t phys_id,
    l4_umword_t enable)
```

Allow CPU shutdown.

Parameters

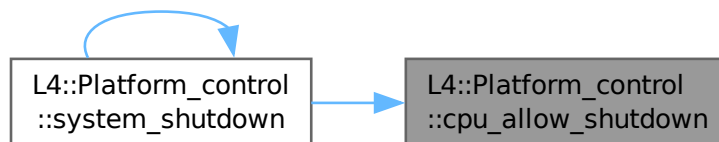
| | |
|----------------|---|
| <i>phys_id</i> | Physical CPU id of CPU (e.g. local APIC id) to disable. |
| <i>enable</i> | Allow shutdown when 1, disallow when 0. |

Sets or unsets a hint that a CPU that is not currently used may be powered down.

References [L4_PLATFORM_CTL_CPU_ENABLE_OP](#).

Referenced by [system_shutdown\(\)](#).

Here is the caller graph for this function:



15.180.2.2 `cpu_disable()`

```
l4_msgtag_t L4::Platform_control::cpu_disable (
    l4_umword_t phys_id)
```

Disable an online CPU.

Parameters

| | |
|----------------|---|
| <i>phys_id</i> | Physical CPU id of CPU (e.g. local APIC id) to disable. |
|----------------|---|

Returns

System call message tag

This function is currently only supported on the ARM EXYNOS platform.

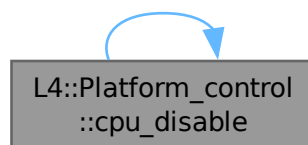
References [cpu_disable\(\)](#).

Referenced by [cpu_disable\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**15.180.2.3 cpu_enable()**

```
l4_msgtag_t L4::Platform_control::cpu_enable (  
    l4_umword_t phys_id)
```

Enable an offline CPU.

Parameters

| | |
|----------------|--|
| <i>phys_id</i> | Physical CPU id of CPU (e.g. local APIC id) to enable. |
|----------------|--|

Returns

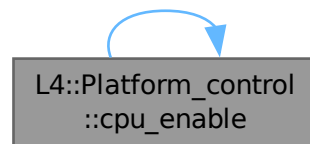
System call message tag

This function is currently only supported on the ARM EXYNOS platform.

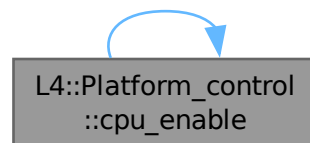
References [cpu_enable\(\)](#), and [L4_PLATFORM_CTL_CPU_DISABLE_OP](#).

Referenced by [cpu_enable\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**15.180.2.4 system_shutdown()**

```
l4_msgtag_t L4::Platform_control::system_shutdown (
    l4_umword_t reboot)
```

Shutdown/Reboot the system.

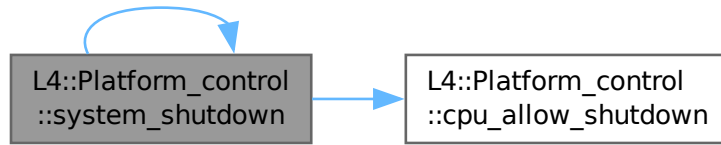
Parameters

| | |
|---------------|-------------------------------|
| <i>reboot</i> | 1 for reboot, 0 for power off |
|---------------|-------------------------------|

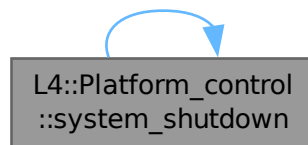
References [cpu_allow_shutdown\(\)](#), [L4_PLATFORM_CTL_CPU_ALLOW_SHUTDOWN_OP](#), and [system_shutdown\(\)](#).

Referenced by [system_shutdown\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.180.2.5 system_suspend()

```
l4_msgtag_t L4::Platform_control::system_suspend (
    l4_umword_t extras)
```

Enter suspend to RAM.

Precondition

Must only be invoked on the boot CPU. Furthermore it must be ensured that the invoking thread is not migrated to a different CPU during the suspend.

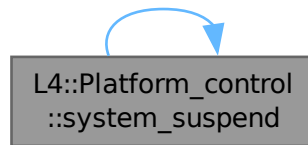
Parameters

| | |
|---------------|---|
| <i>extras</i> | Some extra platform-specific information needed to enter suspend to RAM. On x86 platforms and when using the Platform_control object provided by Fiasco, the value defines the sleep state. The sleep states are defined in the ACPI table. Other platforms as well as lo's Platform_control object don't make use of this value at the moment. |
|---------------|---|

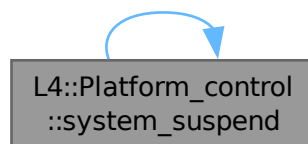
References [L4_PLATFORM_CTL_SYS_SHUTDOWN_OP](#), and [system_suspend\(\)](#).

Referenced by [system_suspend\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

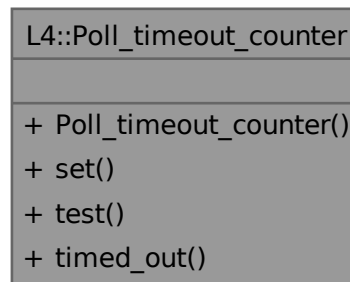
- [l4/sys/platform_control](#)

15.181 L4::Poll_timeout_counter Class Reference

Evaluate an expression for a maximum number of times.

```
#include <poll_timeout_counter.h>
```

Collaboration diagram for L4::Poll_timeout_counter:



Public Member Functions

- [Poll_timeout_counter](#) (unsigned counter_val)
Constructor.
- void [set](#) (unsigned counter_val)
Set the counter to a certain value.
- bool [test](#) (bool expression=true)
Evaluate the expression for a maximum number of times.
- bool [timed_out](#) () const
Indicator if the maximum number of tests was required.

15.181.1 Detailed Description

Evaluate an expression for a maximum number of times.

A typical use case is testing for a bit change in a hardware register for a maximum number of times (polling). For example:

```
Mmio_register_block regs;
Poll_timeout_counter i(3000000);
while (i.test(!(regs.read<uint32_t>(0x04) & 1)))
    ;
```

The following usage is **wrong**:

```
...
Poll_timeout_counter i(3000000);
while (!i.test((regs.read<uint32_t>(0x04) & 1)))
    ;
```

This loop would never terminate if the hardware register doesn't change!

Definition at line 34 of file [poll_timeout_counter.h](#).

15.181.2 Constructor & Destructor Documentation

15.181.2.1 Poll_timeout_counter()

```
L4::Poll_timeout_counter::Poll_timeout_counter (
    unsigned counter_val) [inline]
```

Constructor.

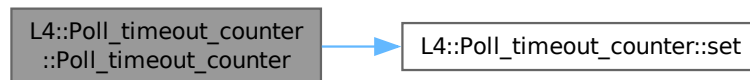
Parameters

| | |
|--------------------|---|
| <i>counter_val</i> | Maximum number of times to repeat the test. |
|--------------------|---|

Definition at line 42 of file [poll_timeout_counter.h](#).

References [set\(\)](#).

Here is the call graph for this function:



15.181.3 Member Function Documentation

15.181.3.1 set()

```
void L4::Poll_timeout_counter::set (
    unsigned counter_val) [inline]
```

Set the counter to a certain value.

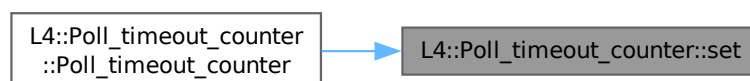
Parameters

| | |
|--------------------|---|
| <i>counter_val</i> | New counter value for maximum number of times to repeat the test. |
|--------------------|---|

Definition at line 53 of file [poll_timeout_counter.h](#).

Referenced by [Poll_timeout_counter\(\)](#).

Here is the caller graph for this function:



15.181.3.2 timed_out()

```
bool L4::Poll_timeout_counter::timed_out () const [inline]
```

Indicator if the maximum number of tests was required.

Return values

| | |
|-----------------|---|
| <i>true, if</i> | the maximum number of tests was required or if the counter was initialized to zero. |
|-----------------|---|

Definition at line 81 of file [poll_timeout_counter.h](#).

The documentation for this class was generated from the following file:

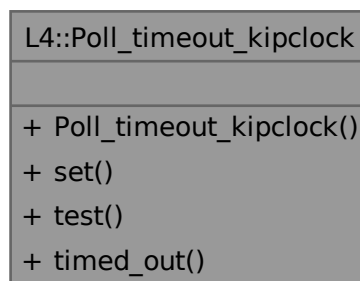
- [pkg/drivers-frst/include/poll_timeout_counter.h](#)

15.182 L4::Poll_timeout_kipclock Class Reference

A polling timeout based on the [L4Re](#) clock.

```
#include <poll_timeout_kipclock>
```

Collaboration diagram for L4::Poll_timeout_kipclock:



Public Member Functions

- [Poll_timeout_kipclock](#) (unsigned poll_time_us)
Initialise relative timeout in microseconds.
- void [set](#) (unsigned poll_time_us)
(Re-)Set relative timeout in microseconds
- bool [test](#) (bool expression=true)
Test whether timeout has expired.
- bool [timed_out](#) () const
Query whether timeout has expired.

15.182.1 Detailed Description

A polling timeout based on the [L4Re](#) clock.

This class allows to conveniently add a timeout to a polling loop.

The original

```
while (device.read(State) & Busy)
;
```

is converted to

```
Poll_timeout_kipclock timeout(10000);
while (timeout.test(device.read(State) & Busy))
;
if (timeout.timed_out())
    printf("ERROR: Device does not respond.\n");
```

Definition at line 35 of file [poll_timeout_kipclock](#).

15.182.2 Constructor & Destructor Documentation

15.182.2.1 Poll_timeout_kipclock()

```
L4::Poll_timeout_kipclock::Poll_timeout_kipclock (
    unsigned poll_time_us) [inline]
```

Initialise relative timeout in microseconds.

Parameters

| | |
|---------------------|----------------------------------|
| <i>poll_time_us</i> | Polling timeout in microseconds. |
|---------------------|----------------------------------|

Definition at line 42 of file [poll_timeout_kipclock](#).

References [set\(\)](#).

Here is the call graph for this function:



15.182.3 Member Function Documentation

15.182.3.1 set()

```
void L4::Poll_timeout_kipclock::set (
    unsigned poll_time_us) [inline]
```

(Re-)Set relative timeout in microseconds

Parameters

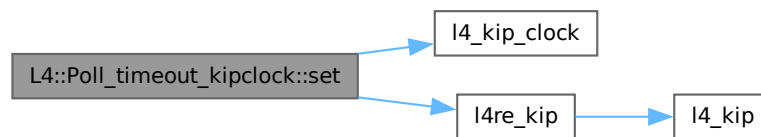
| | |
|---------------------|----------------------------------|
| <i>poll_time_us</i> | Polling timeout in microseconds. |
|---------------------|----------------------------------|

Definition at line 51 of file [poll_timeout_kipclock](#).

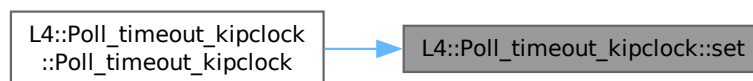
References [l4_kip_clock\(\)](#), and [l4re_kip\(\)](#).

Referenced by [Poll_timeout_kipclock\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.182.3.2 test()

```
bool L4::Poll_timeout_kipclock::test (
    bool expression = true) [inline]
```

Test whether timeout has expired.

Parameters

| | |
|-------------------|----------------------|
| <i>expression</i> | Optional expression. |
|-------------------|----------------------|

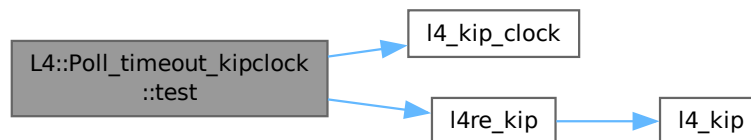
Return values

| | |
|--------------|---|
| <i>false</i> | The timeout has expired or the given expression returned false. |
| <i>true</i> | The timeout has not expired and the optionally given expression returns true. |

Definition at line 65 of file [poll_timeout_kipclock](#).

References [l4_kip_clock\(\)](#), and [l4re_kip\(\)](#).

Here is the call graph for this function:

**15.182.3.3 timed_out()**

```
bool L4::Poll_timeout_kipclock::timed_out () const [inline]
```

Query whether timeout has expired.

Returns

Expiry state of timeout

Definition at line 77 of file [poll_timeout_kipclock](#).

The documentation for this class was generated from the following file:

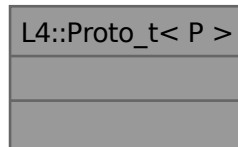
- `l4/re/util/poll_timeout_kipclock`

15.183 L4::Proto_t< P > Struct Template Reference

Data type for defining protocol numbers.

```
#include <__typeinfo.h>
```

Collaboration diagram for L4::Proto_t< P >:



15.183.1 Detailed Description

```
template<I4\_proto\_t P = PROTO_EMPTY>
struct L4::Proto_t< P >
```

Data type for defining protocol numbers.

Template Parameters

| | |
|----------|----------------------------|
| <i>P</i> | The protocol number itself |
|----------|----------------------------|

This type must be used when specifying a protocol number with [Kobject_x](#).

Definition at line [1181](#) of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

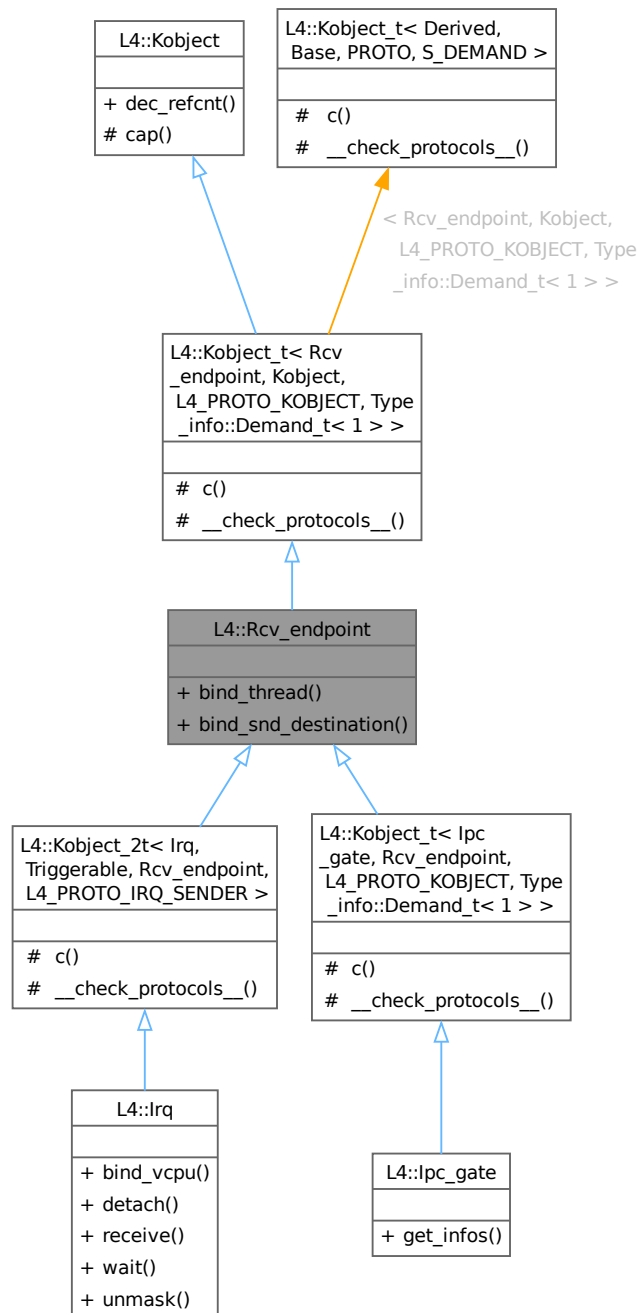
- [I4/sys/__typeinfo.h](#)

15.184 L4::Rcv_endpoint Class Reference

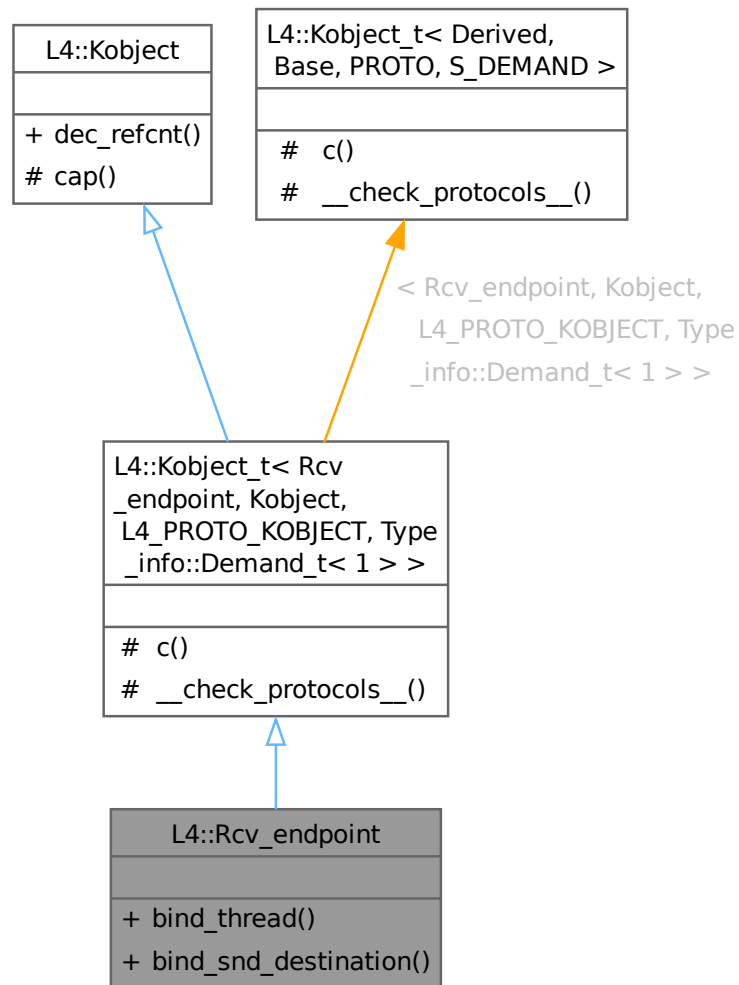
Interface for kernel objects that allow to receive IPC from them.

```
#include <rcv_endpoint>
```

Inheritance diagram for L4::Rcv_endpoint:



Collaboration diagram for L4::Rcv_endpoint:



Public Member Functions

- `l4_msgtag_t bind_thread (lpc::Cap< Thread > t, l4_umword_t label)`
Bind the IPC receive endpoint to a thread.
- `l4_msgtag_t bind_snd_destination (Cap< Snd_destination > snd_dst, l4_umword_t label)`
Bind a send destination (a thread or thread group) to an IPC receive endpoint.

Public Member Functions inherited from L4::Kobject

- `l4_msgtag_t dec_refcnt (l4_mword_t diff, l4_utcb_t *utcb=l4_utcb())`
Decrement the in kernel reference counter for the object.

Additional Inherited Members

Protected Types inherited from

[L4::Kobject_t< Rcv_endpoint, Kobject, L4_PROTO_KOBJECT, Type_info::Demand_t< 1 > >](#)

- typedef [Rcv_endpoint](#) **Class**
The target interface type (inheriting from [Kobject_t](#)).
- typedef Typeid::Iface< PROTO, [Rcv_endpoint](#) > **__iface**
The interface description for the derived class.
- typedef Typeid::Merge_list< Typeid::Iface_list< **__iface** >, typename Kobject::__iface_list > **__iface_list**
The list of all RPC interfaces provided directly or through inheritance.

Protected Member Functions inherited from

[L4::Kobject_t< Rcv_endpoint, Kobject, L4_PROTO_KOBJECT, Type_info::Demand_t< 1 > >](#)

- [L4::Cap< Class > c](#) () const noexcept
Get the capability to ourselves.

Protected Member Functions inherited from [L4::Kobject](#)

- [l4_cap_idx_t](#) [cap](#) () const noexcept
Return capability selector.

Static Protected Member Functions inherited from

[L4::Kobject_t< Rcv_endpoint, Kobject, L4_PROTO_KOBJECT, Type_info::Demand_t< 1 > >](#)

- static void [__check_protocols__](#) () noexcept
Helper to check for protocol conflicts.

15.184.1 Detailed Description

Interface for kernel objects that allow to receive IPC from them.

Such an object is for example an [lpc_gate](#) (with server rights) or an [lrq](#). Those objects can be bound to a thread that shall receive IPC from these objects via [bind_thread\(\)](#) or [bind_snd_destination\(\)](#).

Definition at line 30 of file [rcv_endpoint](#).

15.184.2 Member Function Documentation

15.184.2.1 bind_snd_destination()

```
l4_msgtag_t L4::Rcv_endpoint::bind_snd_destination (
    Cap< Snd_destination > snd_dst,
    l4_umword_t label) [inline]
```

Bind a send destination (a thread or thread group) to an IPC receive endpoint.

Parameters

| | |
|----------------|---|
| <i>snd_dst</i> | Snd_destination object (a thread or thread group) that shall be bound to this receive endpoint. See bind_thread() and bind_snd_destination() for binding a thread or thread group object. |
| <i>label</i> | Label to assign to this receive endpoint. For IPC gates, the two least significant bits must be set to zero. |

Returns

Syscall return tag containing one of the following return codes.

Return values

| | |
|-------------------|---|
| <i>L4_EOK</i> | Operation successful. |
| <i>-L4_EINVAL</i> | snd_dst is not a thread or thread group object or other arguments were malformed. |
| <i>-L4_EPERM</i> | No L4_CAP_FPAGE_S right on snd_dst or the capability used to invoke this operation. |

Precondition

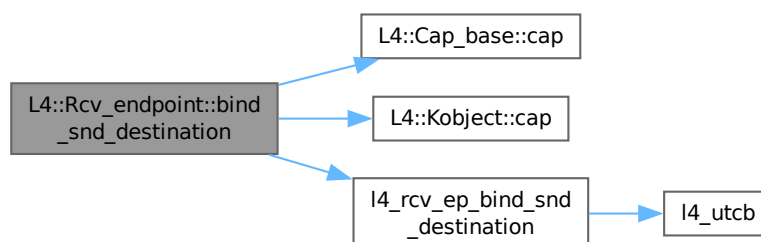
If this operation is invoked using an IPC gate capability without the [L4_FPAGE_C_IPCGATE_SVR](#) right, the kernel will not perform the operation. Instead, the underlying IPC message will be forwarded to the thread bound to the IPC gate, blocking the caller if no thread or thread group is bound yet.

The specified *label* is passed to the receiver of the incoming IPC. It is possible to re-bind a receive endpoint to the same or a different thread or thread group. In this case, IPC already in flight will be delivered with the old label to the previously bound thread or thread group unless [L4::Thread::modify_senders\(\)](#) is used to change these labels.

Definition at line 101 of file [rcv_endpoint](#).

References [L4::Cap_base::cap\(\)](#), [L4::Kobject::cap\(\)](#), and [l4_rcv_ep_bind_snd_destination\(\)](#).

Here is the call graph for this function:



15.184.2.2 `bind_thread()`

```
l4_msgtag_t L4::Rcv_endpoint::bind_thread (
    ipc::Cap< Thread > t,
    l4_umword_t label)
```

Bind the IPC receive endpoint to a thread.

Parameters

| | |
|--------------|---|
| <i>t</i> | Thread object this receive endpoint shall be bound to. |
| <i>label</i> | Label to assign to <code>this</code> receive endpoint. For IPC gates, the two least significant bits must be set to zero. |

Returns

Syscall return tag containing one of the following return codes.

Return values

| | |
|-------------------|--|
| <i>L4_EOK</i> | Operation successful. |
| <i>-L4_EINVAL</i> | <i>t</i> is not a thread object or other arguments were malformed. |
| <i>-L4_EPERM</i> | Insufficient permissions; see precondition. |

Precondition

The invoked capability and the capability *t* both must have the permission [L4_CAP_FPAGE_S](#).

Deprecated Use [bind_snd_destination\(\)](#) instead.

Precondition

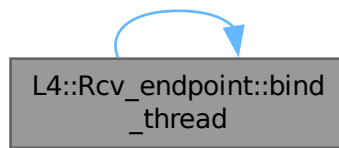
If this operation is invoked using an IPC gate capability without the [L4_FPAGE_C_IPCGATE_SVR](#) right, the kernel will not perform the operation. Instead, the underlying IPC message will be forwarded to the thread the IPC gate is bound to, blocking the caller if the IPC gate was not bound yet.

The specified *label* is passed to the receiver of the incoming IPC. It is possible to re-bind a receive endpoint to the same or a different thread. In this case, IPC already in flight will be delivered with the old label to the previously bound thread unless [L4::Thread::modify_senders\(\)](#) is used to change these labels.

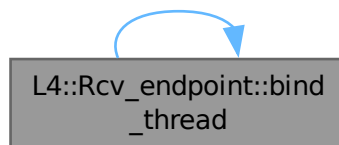
References [bind_thread\(\)](#).

Referenced by [bind_thread\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

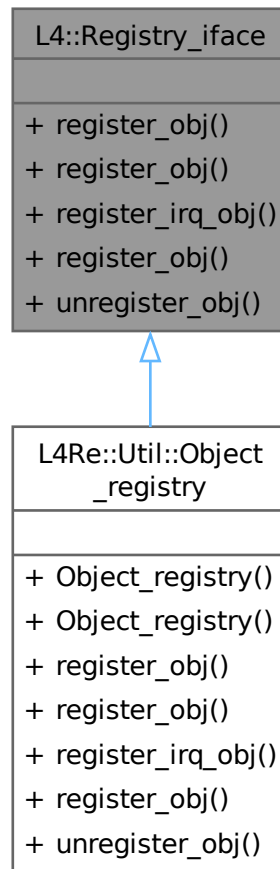
- [l4/sys/rcv_endpoint](#)

15.185 L4::Registry_iface Class Reference

Abstract interface for object registries.

```
#include <ipc_epiface>
```

Inheritance diagram for L4::Registry_iface:



Collaboration diagram for L4::Registry_iface:



Public Member Functions

- virtual [L4::Cap< void >](#) [register_obj](#) ([L4::Epiface](#) *o, char const *service)=0
Register an [L4::Epiface](#) for an IPC gate available in the applications environment under the name `service`.
- virtual [L4::Cap< void >](#) [register_obj](#) ([L4::Epiface](#) *o)=0
Register o as server-side object for synchronous RPC.
- virtual [L4::Cap< L4::Irq >](#) [register_irq_obj](#) ([L4::Epiface](#) *o)=0
Register o as server-side object for asynchronous IRQs.
- virtual [L4::Cap< L4::Rcv_endpoint >](#) [register_obj](#) ([L4::Epiface](#) *o, [L4::Cap< L4::Rcv_endpoint >](#) ep)=0
Register o as server-side object for a pre-allocated capability.
- virtual void [unregister_obj](#) ([L4::Epiface](#) *o, bool unmap=true)=0
Unregister the given object o from the server.

15.185.1 Detailed Description

Abstract interface for object registries.

An object registry allows to register [L4::Epiface](#) objects at a server loop either for synchronous RPC messages or for asynchronous IRQ messages.

Definition at line 434 of file [ipc_epiface](#).

15.185.2 Member Function Documentation

15.185.2.1 [register_irq_obj\(\)](#)

```
virtual L4::Cap< L4::Irq > L4::Registry\_iface::register\_irq\_obj (  
    L4::Epiface * o) [pure virtual]
```

Register o as server-side object for asynchronous IRQs.

Parameters

| | |
|-------------------|---|
| o | Pointer to an Epiface object that shall be registered as server-side object for IRQs. |
|-------------------|---|

Return values

| | |
|---|--|
| L4::Cap<L4::Irq> | Capability to a new IRQ object on success. |
| L4::Cap<L4::Irq>::Invalid | The allocation of the IRQ has failed. |

After successful registration `o->obj_cap()` will be the capability of the allocated IRQ object.

The function may allocate a capability slot for the object. In that case [unregister_obj\(\)](#) is responsible for freeing the slot as well.

Implemented in [L4Re::Util::Object_registry](#).

15.185.2.2 register_obj() [1/3]

```
virtual L4::Cap< void > L4::Registry_iface::register_obj (
    L4::Epiface * o) [pure virtual]
```

Register *o* as server-side object for synchronous RPC.

Parameters

| | |
|----------|--|
| <i>o</i> | Pointer to an Epiface object that shall be registered as server-side object for RPC. |
|----------|--|

Return values

| | |
|--|--|
| L4::Cap< void > | A valid capability to a new IPC gate. |
| L4::Cap< void >::Invalid | The allocation of the IPC gate has failed. |

After successful registration *o*→*obj_cap*() will be the capability of the allocated IPC gate.

The function may allocate a capability slot for the object. In that case [unregister_obj\(\)](#) is responsible for freeing the slot as well.

Implemented in [L4Re::Util::Object_registry](#).

15.185.2.3 register_obj() [2/3]

```
virtual L4::Cap< void > L4::Registry_iface::register_obj (
    L4::Epiface * o,
    char const * service) [pure virtual]
```

Register an [L4::Epiface](#) for an IPC gate available in the applications environment under the name *service*.

Parameters

| | |
|----------------|---|
| <i>o</i> | Pointer to an Epiface object that shall be registered. |
| <i>service</i> | Name of the capability that shall be used to connect <i>o</i> to as a server-side object. |

Return values

| | |
|--|--|
| L4::Cap< void > | The capability known as <i>service</i> on success. |
| L4::Cap< void >::Invalid | No capability with the given name found. |

After a successful call to this function *o*→*obj_cap*() is equal to the capability in the environment with the name given by *service*.

Implemented in [L4Re::Util::Object_registry](#).

15.185.2.4 register_obj() [3/3]

```
virtual L4::Cap< L4::Rcv_endpoint > L4::Registry_iface::register_obj (
    L4::Epiface * o,
    L4::Cap< L4::Rcv_endpoint > ep) [pure virtual]
```

Register *o* as server-side object for a pre-allocated capability.

Parameters

| | |
|-----------|---|
| <i>o</i> | Pointer to an Epiface object that shall be registered as server-side object. |
| <i>ep</i> | Capability to an already allocated capability where <i>o</i> shall be attached as server-side handler. The capability may point to an IPC gate or an IRQ. |

Return values

| | |
|--|--------------------------------------|
| L4::Cap<L4::Rcv_endpoint> | Capability <i>ep</i> on success. |
| L4::Cap<L4::Rcv_endpoint>::Invalid | The IRQ attach operation has failed. |

After successful registration *o*→*obj_cap*() will be equal to *ep*.

Implemented in [L4Re::Util::Object_registry](#).

15.185.2.5 unregister_obj()

```
virtual void L4::Registry_iface::unregister_obj (
    L4::Epiface * o,
    bool unmap = true) [pure virtual]
```

Unregister the given object *o* from the server.

Parameters

| | |
|--------------|---|
| <i>o</i> | Pointer to the Epiface object that shall be unregistered. The object must have been registered with any of the register methods if Registry_iface . |
| <i>unmap</i> | If true the capability <i>o</i> → <i>obj_cap</i> () shall be unmapped from the local object space. |

The function always unmaps and frees the capability if it was allocated by either [Registry_iface::register_irq_obj\(L4::Epiface *\)](#), or by [Registry_iface::register_obj\(L4::Epiface *\)](#).

Implemented in [L4Re::Util::Object_registry](#).

The documentation for this class was generated from the following file:

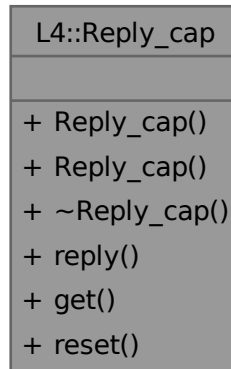
- `l4/sys/cxx/ipc_epiface`

15.186 L4::Reply_cap Class Reference

An explicit reply capability.

```
#include <capability.h>
```

Collaboration diagram for L4::Reply_cap:



Public Member Functions

- constexpr **Reply_cap** () noexcept=default
Construct an invalid reply capability.
- constexpr **Reply_cap** (**Reply_cap_idx** cap, **Reply_cap_alloc** *alloc) noexcept
Construct a valid explicit reply capability.
- ~**Reply_cap** ()
Destroy reply capability.
- **l4_ret_t** **reply** (**l4_msgtag_t** tag, **l4_utcb_t** *utcb=**l4_utcb**()) noexcept
Reply with this reply capability.
- constexpr **Reply_cap_idx** **get** () const noexcept
Get reply capability index.
- void **reset** (**Reply_cap_idx** cap=**Reply_cap_idx**(), **Reply_cap_alloc** *alloc=nullptr) noexcept
Replace reply capability.

15.186.1 Detailed Description

An explicit reply capability.

Represents a use-once, explicit reply capability. If valid, the class will ensure that the caller will always get a reply. Once used, the reply capability slot will be immediately released to the associated reply capability allocator.

Definition at line 529 of file [capability.h](#).

15.186.2 Constructor & Destructor Documentation

15.186.2.1 Reply_cap()

```
L4::Reply_cap::Reply_cap (
    Reply_cap_idx cap,
    Reply_cap_alloc * alloc) [inline], [constexpr], [noexcept]
```

Construct a valid explicit reply capability.

The object will take ownership of the reply capability slot. It is returned to the allocator after being used.

Parameters

| | |
|--------------|--|
| <i>cap</i> | The reply capability index |
| <i>alloc</i> | The associated reply capability allocator. |

Definition at line 544 of file [capability.h](#).

References [Reply_cap\(\)](#).

Here is the call graph for this function:



15.186.2.2 ~Reply_cap()

```
L4::Reply_cap::~~Reply_cap () [inline]
```

Destroy reply capability.

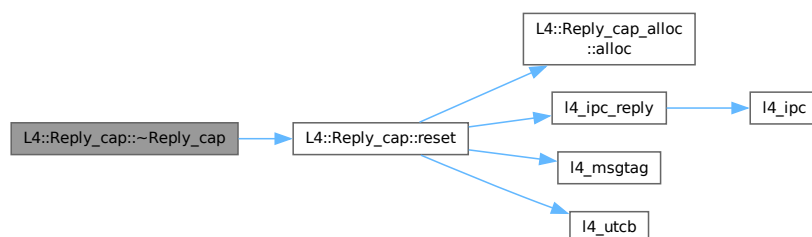
Attention

If the reply capability is still valid, an `L4_EDROPREPLY` error reply is sent. See [Reply_cap::reset\(\)](#).

Definition at line 554 of file [capability.h](#).

References [reset\(\)](#).

Here is the call graph for this function:



15.186.3 Member Function Documentation

15.186.3.1 `get()`

`Reply_cap_idx` `L4::Reply_cap::get () const [inline], [constexpr], [noexcept]`

Get reply capability index.

This method must not be used to send a reply. Use `Reply_cap::reply()` instead that does the proper reply capability lifetime management.

Definition at line 612 of file `capability.h`.

15.186.3.2 `reply()`

```
l4_ret_t L4::Reply_cap::reply (
    l4_msgtag_t tag,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
```

Reply with this reply capability.

Disposes the reply capability after the reply was sent. Can be used only once. If the reply object is not valid, no message is sent.

Parameters

| | |
|-------------|--|
| <i>tag</i> | Message tag for reply. The UTCB must have been prepared. |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |

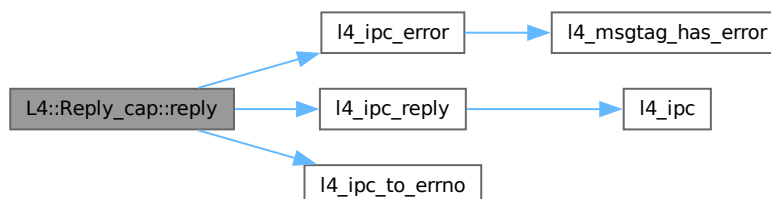
Return values

| | |
|--------------------------------------|----------------------------|
| 0 | Success |
| -(L4_EIPC_LO + L4_IPC_ENOT_EXISTENT) | Reply capability not valid |
| <0 | Other IPC error |

Definition at line 589 of file `capability.h`.

References [L4_IPC_BOTH_TIMEOUT_0](#), [L4_IPC_ENOT_EXISTENT](#), [l4_ipc_error\(\)](#), [l4_ipc_reply\(\)](#), and [l4_ipc_to_errno\(\)](#).

Here is the call graph for this function:



15.186.3.3 reset()

```
void L4::Reply_cap::reset (
    Reply_cap_idx cap = Reply_cap_idx(),
    Reply_cap_alloc * alloc = nullptr) [inline], [noexcept]
```

Replace reply capability.

Attention

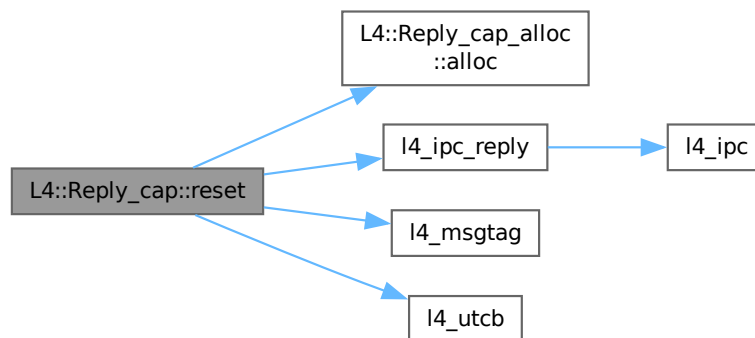
If the current reply capability is valid, an L4_EDROPREPLY error reply is sent to the caller!

Definition at line 639 of file [capability.h](#).

References [L4::Reply_cap_alloc::alloc\(\)](#), [L4_EDROPREPLY](#), [L4_IPC_BOTH_TIMEOUT_0](#), [l4_ipc_reply\(\)](#), [l4_msgtag\(\)](#), and [l4_utcb\(\)](#).

Referenced by [~Reply_cap\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

- `l4/sys/cxx/capability.h`

15.187 L4::Reply_cap_alloc Class Reference

Interface to a reply capability allocator.

```
#include <capability.h>
```

Collaboration diagram for L4::Reply_cap_alloc:

| L4::Reply_cap_alloc | |
|---------------------|---------|
| | |
| + | alloc() |
| # | free() |

Public Member Functions

- virtual Reply_cap [alloc](#) () noexcept=0
Allocate new reply capability slot.

Protected Member Functions

- virtual void [free](#) (Reply_cap_idx cap) noexcept=0
Free reply capability slot.

15.187.1 Detailed Description

Interface to a reply capability allocator.

Definition at line [488](#) of file [capability.h](#).

15.187.2 Member Function Documentation

15.187.2.1 alloc()

```
virtual Reply_cap L4::Reply_cap_alloc::alloc () [pure virtual], [noexcept]
```

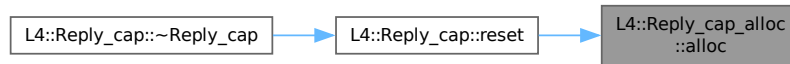
Allocate new reply capability slot.

Returns

Reply capability index or invalid index if exhausted.

Referenced by [L4::Reply_cap::reset\(\)](#).

Here is the caller graph for this function:

**15.187.2.2 free()**

```
virtual void L4::Reply_cap_alloc::free (
    Reply_cap_idx cap) [protected], [pure virtual], [noexcept]
```

Free reply capability slot.

The slot must have been allocated by the same allocator.

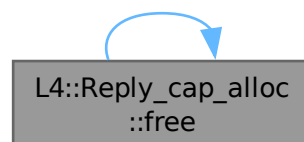
Parameters

| | |
|------------|-----------------------------|
| <i>cap</i> | The reply cap slot to free. |
|------------|-----------------------------|

References [free\(\)](#).

Referenced by [free\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

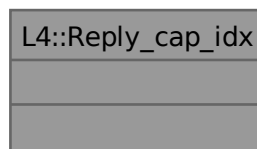
- `l4/sys/cxx/capability.h`

15.188 L4::Reply_cap_idx Class Reference

Value class for a reply capability index.

```
#include <capability.h>
```

Collaboration diagram for L4::Reply_cap_idx:



15.188.1 Detailed Description

Value class for a reply capability index.

Ensures that the `L4_REPLY_CAP_BIT` is always set to make this an index into the reply capability space.

Definition at line [458](#) of file [capability.h](#).

The documentation for this class was generated from the following file:

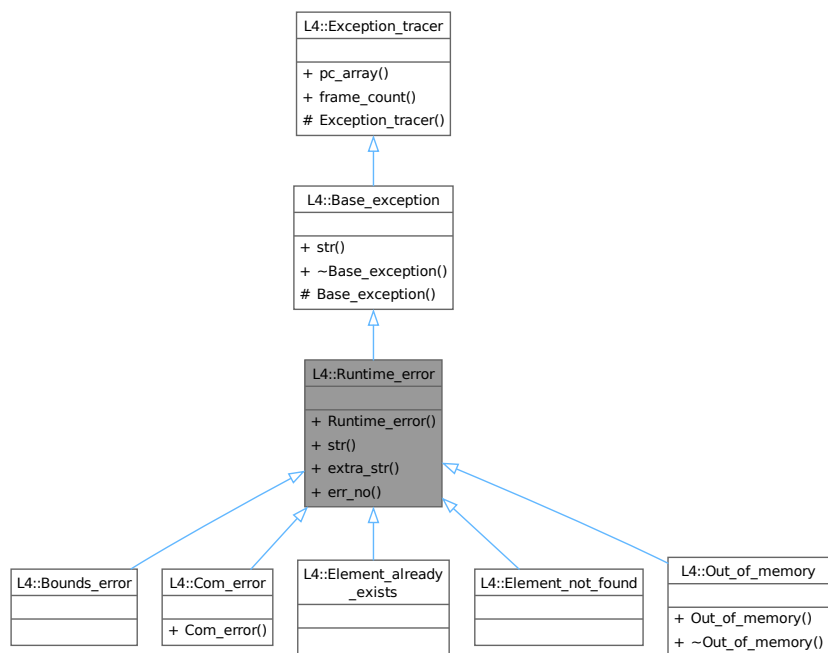
- `l4/sys/cxx/capability.h`

15.189 L4::Runtime_error Class Reference

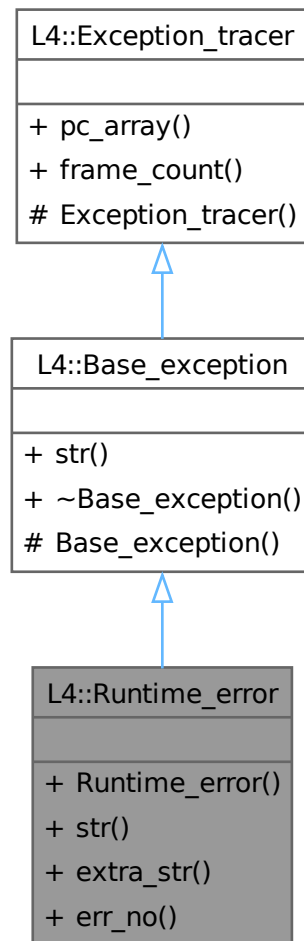
[Exception](#) for an abstract runtime error.

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Runtime_error:



Collaboration diagram for L4::Runtime_error:



Public Member Functions

- [Runtime_error](#) (long [err_no](#), char const *extra=0) noexcept
Create a new [Runtime_error](#).
- char const * [str](#) () const noexcept override
Return a human readable string for the exception.
- char const * [extra_str](#) () const
Get the description text for this runtime error.
- long [err_no](#) () const noexcept
Get the error value for this runtime error.

Public Member Functions inherited from [L4::Base_exception](#)

- virtual `~Base_exception ()` noexcept
Destruction.

Public Member Functions inherited from [L4::Exception_tracer](#)

- `void const *const * pc_array () const noexcept`
Get the array containing the call trace.
- `int frame_count () const noexcept`
Get the number of entries that are valid in the call trace.

Additional Inherited Members

Protected Member Functions inherited from [L4::Base_exception](#)

- `Base_exception () noexcept`
Create a base exception.

Protected Member Functions inherited from [L4::Exception_tracer](#)

- `Exception_tracer () noexcept`
Create a back trace.

15.189.1 Detailed Description

[Exception](#) for an abstract runtime error.

This is the base class for a set of exceptions that cover all errors that have a C error value (see [l4_error_code_t](#)).

Definition at line [128](#) of file [exceptions](#).

15.189.2 Constructor & Destructor Documentation

15.189.2.1 Runtime_error()

```
L4::Runtime_error::Runtime_error (
    long err_no,
    char const * extra = 0) [inline], [explicit], [noexcept]
```

Create a new [Runtime_error](#).

Parameters

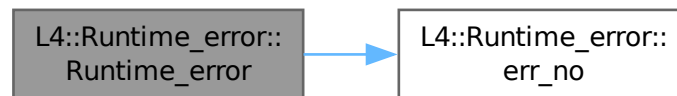
| | |
|---------------|---|
| <i>err_no</i> | Error value for this runtime error. |
| <i>extra</i> | Description of what happened when the error occurred. |

Definition at line [141](#) of file [exceptions](#).

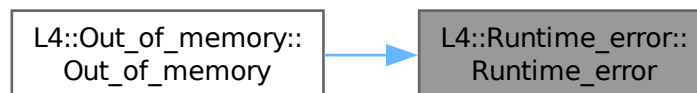
References [err_no\(\)](#).

Referenced by [L4::Out_of_memory::Out_of_memory\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.189.3 Member Function Documentation

15.189.3.1 `err_no()`

```
long L4::Runtime_error::err_no () const [inline], [noexcept]
```

Get the error value for this runtime error.

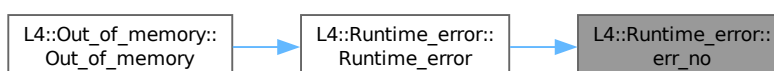
Returns

Error value.

Definition at line 170 of file [exceptions](#).

Referenced by [Runtime_error\(\)](#).

Here is the caller graph for this function:



15.189.3.2 extra_str()

```
char const * L4::Runtime_error::extra_str () const [inline]
```

Get the description text for this runtime error.

Returns

Pointer to the description string.

Definition at line 162 of file [exceptions](#).

The documentation for this class was generated from the following file:

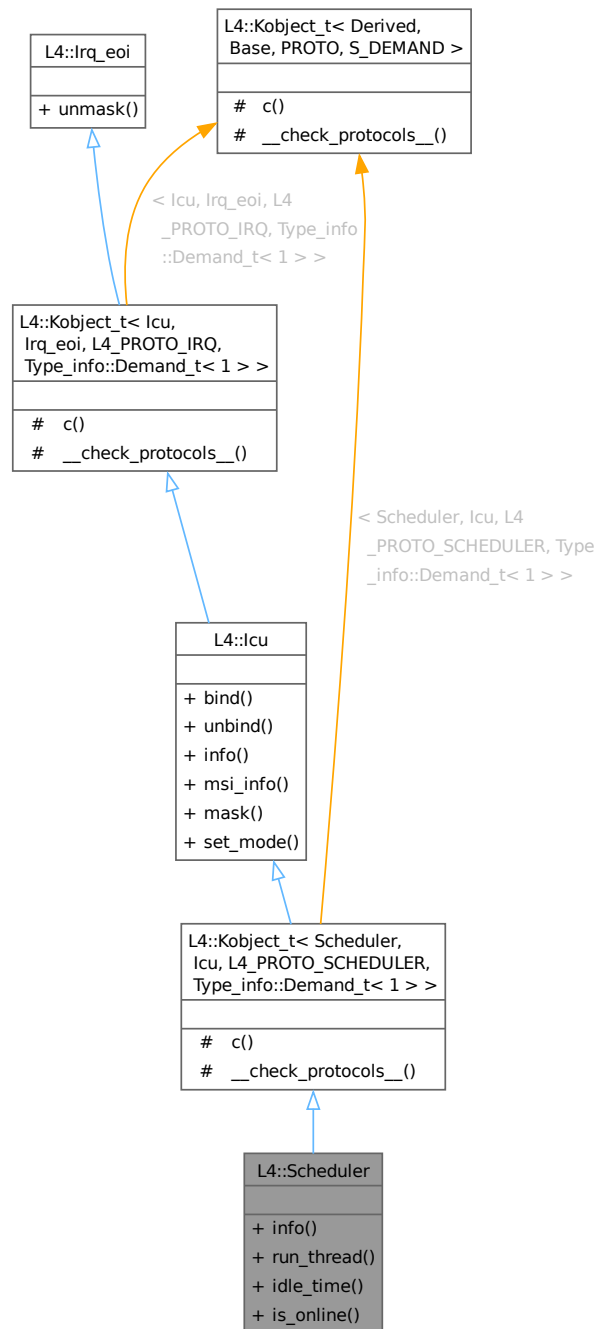
- [l4/cxx/exceptions](#)

15.190 L4::Scheduler Class Reference

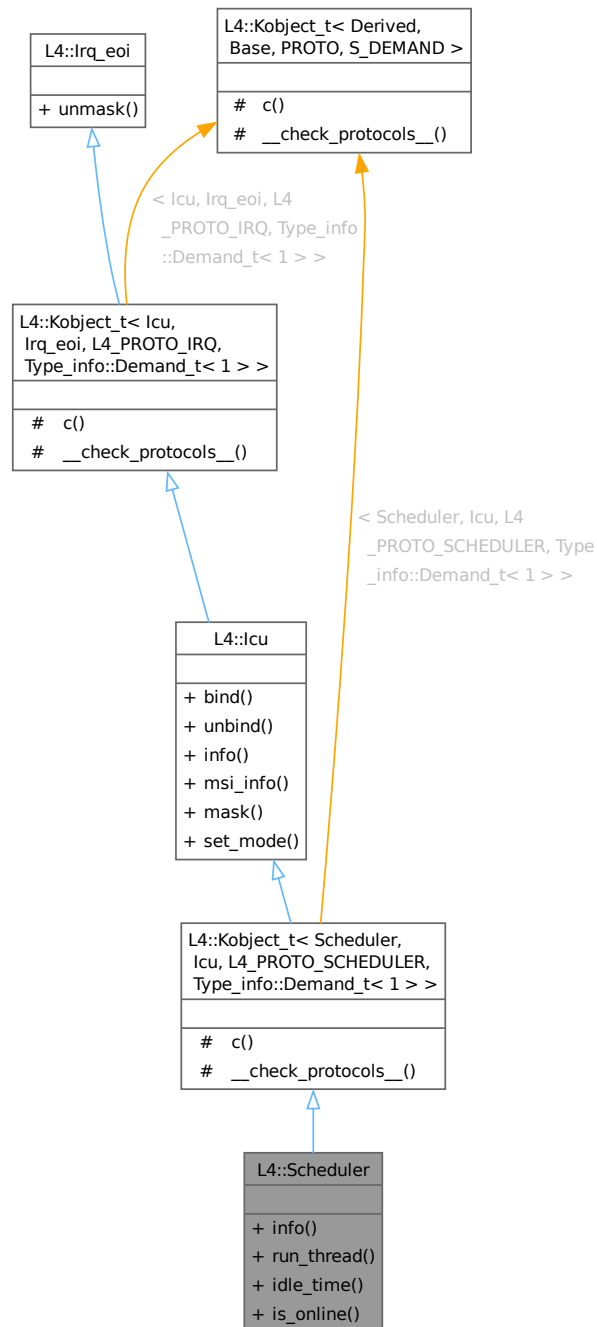
C++ interface of the [Scheduler](#) kernel object, see [Scheduler](#) for the C interface.

```
#include <scheduler>
```

Inheritance diagram for L4::Scheduler:



Collaboration diagram for L4::Scheduler:



Public Member Functions

- `l4_msgtag_t info (l4_umword_t *cpu_max, l4_sched_cpu_set_t *cpus, l4_umword_t *sched_classes=nullptr, l4_utcb_t *utcb=l4_utcb()) const noexcept`
Get scheduler information.
- `l4_msgtag_t run_thread (lpc::Cap< Thread > thread, l4_sched_param_t const &sp)`
Run a thread on a *Scheduler*.

- `l4_msgtag_t idle_time` (`l4_sched_cpu_set_t` const &cpus, `l4_kernel_clock_t` *us)
Query the idle time (in μ s) of a CPU.
- `bool is_online` (`l4_umword_t` cpu, `l4_utcb_t` *utcb=`l4_utcb`()) const noexcept
Query if a CPU is online.

Public Member Functions inherited from `L4::lcu`

- `l4_msgtag_t bind` (unsigned irqnum, `L4::Cap`< `Triggerable` > irq, `l4_utcb_t` *utcb=`l4_utcb`()) noexcept
Bind an interrupt line of an interrupt controller to an interrupt object.
- `l4_msgtag_t unbind` (unsigned irqnum, `L4::Cap`< `Triggerable` > irq, `l4_utcb_t` *utcb=`l4_utcb`()) noexcept
Remove binding of an interrupt line from the interrupt controller object.
- `l4_msgtag_t info` (`l4_icu_info_t` *info, `l4_utcb_t` *utcb=`l4_utcb`()) noexcept
Get information about the ICU features.
- `l4_msgtag_t msi_info` (`l4_umword_t` irqnum, `l4_uint64_t` source, `l4_icu_msi_info_t` *msi_info)
Get MSI info about IRQ.
- `l4_msgtag_t mask` (unsigned irqnum, `l4_umword_t` *label=0, `l4_timeout_t` to=`L4_IPC_NEVER`, `l4_utcb_t` *utcb=`l4_utcb`()) noexcept
Mask an IRQ line.
- `l4_msgtag_t set_mode` (unsigned irqnum, `l4_umword_t` mode, `l4_utcb_t` *utcb=`l4_utcb`()) noexcept
Set interrupt mode.

Public Member Functions inherited from `L4::lrq_eoi`

- `l4_msgtag_t unmask` (unsigned irqnum, `l4_umword_t` *label=0, `l4_timeout_t` to=`L4_IPC_NEVER`, `l4_utcb_t` *utcb=`l4_utcb`()) noexcept
Unmask the given interrupt line.

Additional Inherited Members

Protected Types inherited from

`L4::Kobject_t`< `Scheduler`, `lcu`, `L4_PROTO_SCHEDULER`, `Type_info::Demand_t`< 1 > >

- typedef `Scheduler` **Class**
The target interface type (inheriting from `Kobject_t`).
- typedef `Typeid::Iface`< `PROTO`, `Scheduler` > **__Iface**
The interface description for the derived class.
- typedef `Typeid::Merge_list`< `Typeid::Iface_list`< **__Iface** >, typename `lcu::__Iface_list` > **__Iface_list**
The list of all RPC interfaces provided directly or through inheritance.

Protected Types inherited from

`L4::Kobject_t`< `lcu`, `lrq_eoi`, `L4_PROTO_IRQ`, `Type_info::Demand_t`< 1 > >

- typedef `lcu` **Class**
The target interface type (inheriting from `Kobject_t`).
- typedef `Typeid::Iface`< `PROTO`, `lcu` > **__Iface**
The interface description for the derived class.
- typedef `Typeid::Merge_list`< `Typeid::Iface_list`< **__Iface** >, typename `lrq_eoi::__Iface_list` > **__Iface_list**
The list of all RPC interfaces provided directly or through inheritance.

Protected Member Functions inherited from**L4::Kobject_t< Scheduler, Icu, L4_PROTO_SCHEDULER, Type_info::Demand_t< 1 > >**

- **L4::Cap< Class > c ()** const noexcept

*Get the capability to ourselves.***Protected Member Functions inherited from****L4::Kobject_t< Icu, Irq_eoi, L4_PROTO_IRQ, Type_info::Demand_t< 1 > >**

- **L4::Cap< Class > c ()** const noexcept

*Get the capability to ourselves.***Static Protected Member Functions inherited from****L4::Kobject_t< Scheduler, Icu, L4_PROTO_SCHEDULER, Type_info::Demand_t< 1 > >**

- **static void __check_protocols__ ()** noexcept

*Helper to check for protocol conflicts.***Static Protected Member Functions inherited from****L4::Kobject_t< Icu, Irq_eoi, L4_PROTO_IRQ, Type_info::Demand_t< 1 > >**

- **static void __check_protocols__ ()** noexcept

*Helper to check for protocol conflicts.***15.190.1 Detailed Description**

C++ interface of the [Scheduler](#) kernel object, see [Scheduler](#) for the C interface.

The [Scheduler](#) interface allows a client to manage CPU resources. The API provides functions to query scheduler information, check the online state of CPUs, query CPU idle time and to start threads on defined CPU sets.

The scheduler offers IRQ number 0, which triggers when the number of online cores changes, e.g. due to hotplug events.

The [Scheduler](#) interface inherits from [L4::Icu](#) and [L4::Irq_eoi](#) for managing the scheduler virtual device IRQ which, in contrast to hardware IRQs, implements a limited functionality:

- Only IRQ line 0 is supported, no MSI vectors.
- The IRQ is edge-triggered and the IRQ mode cannot be changed.
- As the IRQ is edge-triggered, it does not have to be explicitly unmasked.

It depends on the platform, which hotplug events actually trigger the IRQ. Many platforms only support triggering the IRQ when a CPU core different from the boot CPU goes online.

Include File

```
#include <l4/sys/scheduler>
```

Definition at line 46 of file [scheduler](#).

15.190.2 Member Function Documentation

15.190.2.1 idle_time()

```
l4_msgtag_t L4::Scheduler::idle_time (
    l4_sched_cpu_set_t const & cpus,
    l4_kernel_clock_t * us)
```

Query the idle time (in μ s) of a CPU.

Parameters

| | | |
|-----|-------------|---|
| | <i>cpus</i> | Set of CPUs to query. Only the idle time of the first selected CPU in <code>cpus.map</code> is queried. |
| out | <i>us</i> | Idle time of queried CPU in μ s. |

Return values

| | |
|------------|-----------------------------------|
| 0 | Success. |
| -L4_EINVAL | Invalid CPU requested in cpu set. |

This function retrieves the idle time in μ s of the first selected CPU in `cpus.map`. The idle time is the accumulated time a CPU has spent in the idle thread since its last reset. To calculate a load estimate l one has to retrieve the idle time at the beginning ($i1$) and the end ($i2$) of a known time interval t . The load is then calculated as $l = 1 - (i2 - i1)/t$.

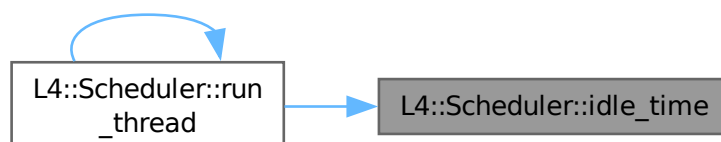
The idle time is only defined for online CPUs. Reading the idle time from offline CPUs is undefined and may result in either getting -L4_EINVAL or calculating an estimated (incorrect) load of 1.

Note

The idle time statistics of remote CPUs is updated on context switch events only, hence may not be up-to-date when requested cross-CPU. To get up-to-date idle time you should use a thread running on the same CPU of which the idle time is requested.

Referenced by [run_thread\(\)](#).

Here is the caller graph for this function:



15.190.2.2 info()

```
l4_msgtag_t L4::Scheduler::info (
    l4_umword_t * cpu_max,
    l4_sched_cpu_set_t * cpus,
    l4_umword_t * sched_classes = nullptr,
    l4_utcb_t * utcb = l4_utcb()) const [inline], [noexcept]
```

Get scheduler information.

Parameters

| | | |
|---------|----------------------|---|
| out | <i>cpu_max</i> | Maximum number of CPUs ever available. Optional, can be nullptr. |
| in, out | <i>cpus</i> | <i>cpus.offset</i> is first CPU of interest. <i>cpus.granularity</i> (see l4_sched_cpu_set_t). <i>cpus.map</i> Bitmap of online CPUs. Must not be nullptr. |
| out | <i>sched_classes</i> | A bitmap of available scheduling classes (see L4_scheduler_classes). Pass nullptr to omit this information. |
| | <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb . |

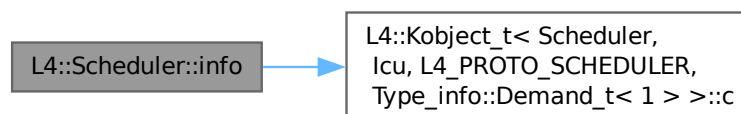
Return values

| | |
|------------|---|
| 0 | Success. |
| -L4_ERANGE | The given CPU offset is larger than the maximum number of CPUs. |

Definition at line 74 of file [scheduler](#).

References [L4::Kobject_t< Scheduler, Icu, L4_PROTO_SCHEDULER, Type_info::Demand_t< 1 > >::c\(\)](#), [l4_sched_cpu_set_t::gran_offset](#), and [l4_sched_cpu_set_t::map](#).

Here is the call graph for this function:



15.190.2.3 is_online()

```
bool L4::Scheduler::is_online (
    l4_umword_t cpu,
    l4_utcb_t * utcb = l4_utcb()) const [inline], [noexcept]
```

Query if a CPU is online.

Parameters

| | |
|-------------|--|
| <i>cpu</i> | CPU number whose online status should be queried. |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb . |

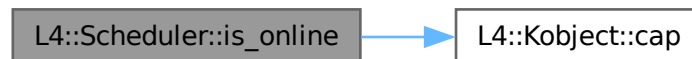
Return values

| | |
|--------------|--------------------|
| <i>true</i> | The CPU is online. |
| <i>false</i> | The CPU is offline |

Definition at line 154 of file [scheduler](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:

**15.190.2.4 run_thread()**

```

l4_msgtag_t L4::Scheduler::run_thread (
    Ipc::Cap< Thread > thread,
    l4_sched_param_t const & sp)
  
```

Run a thread on a [Scheduler](#).

Parameters

| | |
|---------------|----------------------------------|
| <i>thread</i> | Capability of the thread to run. |
| <i>sp</i> | Scheduling parameters. |

Return values

| | |
|-------------------|---|
| <i>0</i> | Success. |
| <i>-L4_EINVAL</i> | Invalid size of the scheduling parameter. |

This function launches a thread on a CPU determined by the scheduling parameter `sp.affinity`. A thread can be intentionally stopped by migrating it on an offline or an invalid CPU. The thread is only guaranteed to run if the CPU it is migrated to is currently online.

Note

If the target CPU is currently not online, there is no guarantee that the thread will ever run, even if the CPU comes online later on.

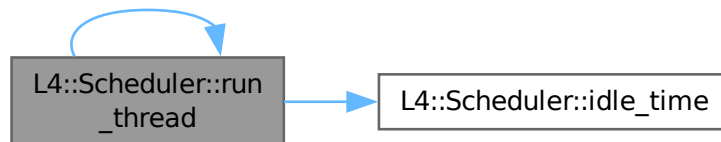
A scheduler may impose a policy with regard to selecting CPUs. However the scheduler is required to ensure the following two properties:

- Two threads with disjoint CPU sets must be scheduled to different CPUs.
- Two threads with identical CPU sets selecting only a single CPU must be scheduled to the same CPU.

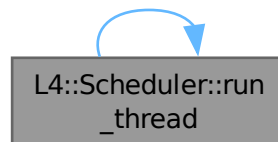
References [idle_time\(\)](#), [L4_SCHEDULER_IDLE_TIME_OP](#), and [run_thread\(\)](#).

Referenced by [run_thread\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

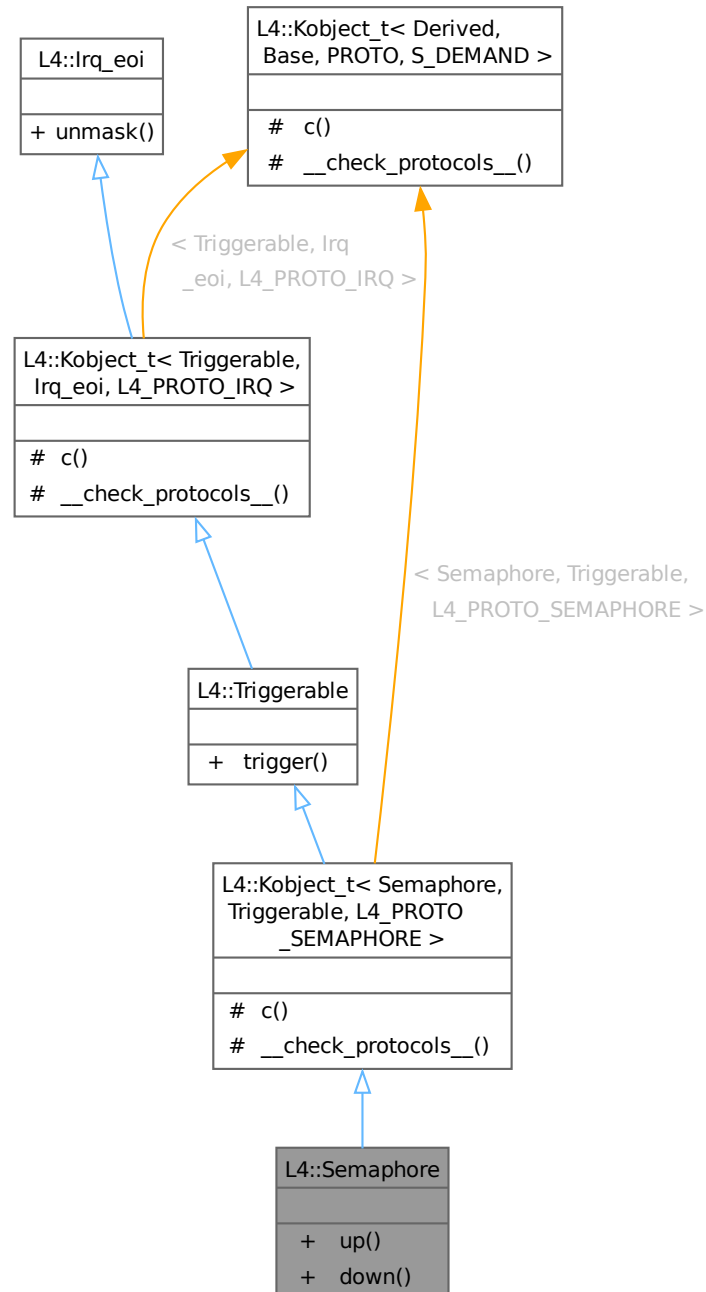
- [l4/sys/scheduler](#)

15.191 L4::Semaphore Struct Reference

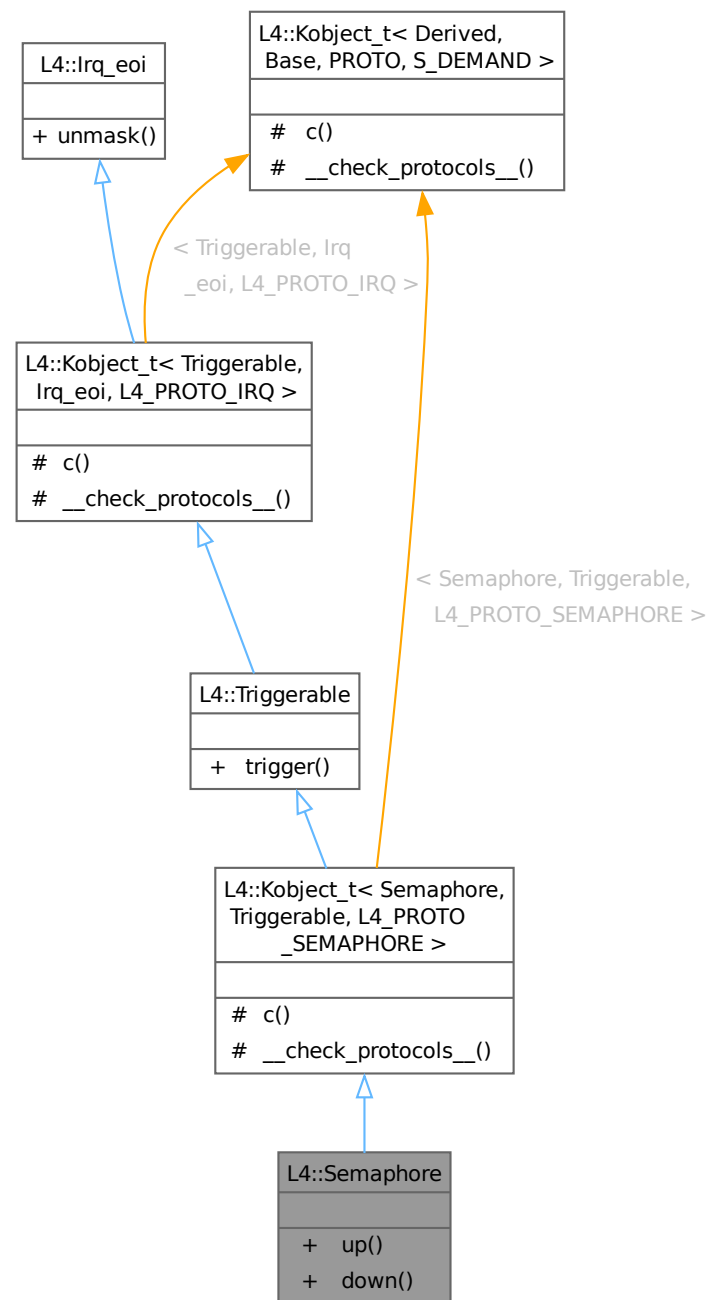
C++ Kernel-provided semaphore interface, see [Kernel-provided semaphore](#) for the C interface.

```
#include <semaphore>
```

Inheritance diagram for L4::Semaphore:



Collaboration diagram for L4::Semaphore:



Public Member Functions

- [l4_msgtag_t up \(l4_utcb_t *utcb=l4_utcb\(\)\) noexcept](#)
Semaphore up operation (wrapper for [trigger\(\)](#)).
- [l4_msgtag_t down \(l4_timeout_t timeout=L4_IPC_NEVER, l4_utcb_t *utcb=l4_utcb\(\)\) noexcept](#)
Semaphore down operation.

Public Member Functions inherited from [L4::Triggerable](#)

- [l4_msgtag_t trigger](#) ([l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) noexcept
Trigger the object.

Public Member Functions inherited from [L4::Irq_eoi](#)

- [l4_msgtag_t unmask](#) (unsigned irqnum, [l4_umword_t](#) *label=0, [l4_timeout_t](#) to=[L4_IPC_NEVER](#), [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) noexcept
Unmask the given interrupt line.

Additional Inherited Members

Protected Types inherited from [L4::Kobject_t](#)< [Semaphore](#), [Triggerable](#), [L4_PROTO_SEMAPHORE](#) >

- typedef [Semaphore](#) **Class**
The target interface type (inheriting from [Kobject_t](#)).
- typedef Typeid::Iface< [PROTO](#), [Semaphore](#) > **__Iface**
The interface description for the derived class.
- typedef Typeid::Merge_list< Typeid::Iface_list< **__Iface** >, typename [Triggerable](#)::__Iface_list > **__Iface_list**
The list of all RPC interfaces provided directly or through inheritance.

Protected Types inherited from [L4::Kobject_t](#)< [Triggerable](#), [Irq_eoi](#), [L4_PROTO_IRQ](#) >

- typedef [Triggerable](#) **Class**
The target interface type (inheriting from [Kobject_t](#)).
- typedef Typeid::Iface< [PROTO](#), [Triggerable](#) > **__Iface**
The interface description for the derived class.
- typedef Typeid::Merge_list< Typeid::Iface_list< **__Iface** >, typename [Irq_eoi](#)::__Iface_list > **__Iface_list**
The list of all RPC interfaces provided directly or through inheritance.

Protected Member Functions inherited from [L4::Kobject_t](#)< [Semaphore](#), [Triggerable](#), [L4_PROTO_SEMAPHORE](#) >

- [L4::Cap](#)< [Class](#) > **c** () const noexcept
Get the capability to ourselves.

Protected Member Functions inherited from [L4::Kobject_t](#)< [Triggerable](#), [Irq_eoi](#), [L4_PROTO_IRQ](#) >

- [L4::Cap](#)< [Class](#) > **c** () const noexcept
Get the capability to ourselves.

Static Protected Member Functions inherited from**[L4::Kobject_t](#) < [Semaphore](#), [Triggerable](#), [L4_PROTO_SEMAPHORE](#) >**

- static void `__check_protocols__()` noexcept
Helper to check for protocol conflicts.

Static Protected Member Functions inherited from**[L4::Kobject_t](#) < [Triggerable](#), [Irq_eoi](#), [L4_PROTO_IRQ](#) >**

- static void `__check_protocols__()` noexcept
Helper to check for protocol conflicts.

15.191.1 Detailed Description

C++ Kernel-provided semaphore interface, see [Kernel-provided semaphore](#) for the C interface.

This is the interface for kernel-provided semaphore objects. The object provides the classical functions `up()` and `down()` for counting the semaphore and blocking. The semaphore is a [Triggerable](#) with respect to the `up()` function, this means that a semaphore can be bound to an interrupt line at an ICU ([L4::lcu](#)) and incoming interrupts increment the semaphore counter.

The `down()` method decrements the semaphore counter and blocks if the counter is already zero. Blocking on a semaphore may—as all blocking operations—either return successfully, or be aborted due to an expired timeout provided to the `down()` operation, or due to an [L4::Thread::ex_regs\(\)](#) operation with the [L4_THREAD_EX_REGS_CANCEL](#) flag set.

A semaphore object is initialized with counter value 0.

The main reason for using a semaphore instead of an [L4::Irq](#) is to ensure that incoming trigger signals do not interfere with any open-wait operations, as used for example in a server loop.

Note that this is a kernel-level semaphore primitive that shall be used to implement user-level, application-usable synchronization primitives. For example, use `pthread_mutex` functions in applications if possible. When implementing a synchronization primitive, please ensure to only use [L4::Semaphore](#) in the case of contention, and use atomic operations for the non-contended case.

Definition at line 51 of file [semaphore](#).

15.191.2 Member Function Documentation**15.191.2.1 `down()`**

```
l4_msgtag_t L4::Semaphore::down (
    l4_timeout_t timeout = L4_IPC_NEVER,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
```

[Semaphore](#) down operation.

Parameters

| | |
|----------------|---|
| <i>timeout</i> | Timeout for blocking the semaphore down operation. Note: The receive timeout of this timeout-pair is significant for blocking, the send part is usually non-blocking. |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |

Returns

Syscall return tag. Use [l4_error\(\)](#) to check for errors.

Return values

| | |
|------------------------|---|
| <code>-L4_EPERM</code> | Insufficient permissions; see precondition. |
|------------------------|---|

Precondition

The invoked [Semaphore](#) capability must have the permission [L4_CAP_FPAGE_S](#).

This method decrements the semaphore counter by one, or blocks if the counter is already zero, until either a timeout or cancel condition hits or the counter is increased by an [up\(\)](#) operation.

Definition at line 89 of file [semaphore](#).

References [L4::Kobject::cap\(\)](#), and [L4_IPC_NEVER](#).

Here is the call graph for this function:



15.191.2.2 up()

```

l4_msgtag_t L4::Semaphore::up (
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
  
```

[Semaphore](#) up operation (wrapper for [trigger\(\)](#)).

Parameters

| | |
|-------------|--|
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |
|-------------|--|

Returns

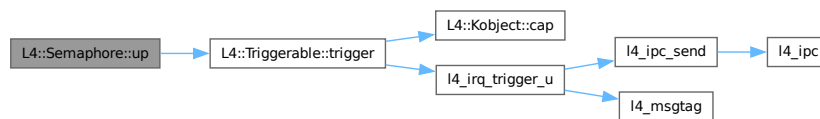
Syscall return tag for a send-only operation, this means there is no return value except [L4_MSGTAG_ERROR](#) indicating success or failure of the send operation. Use [l4_ipc_error\(\)](#) to check for errors and **do not** use [l4_error\(\)](#).

Increases the semaphore counter by one if it is smaller than an unspecified limit. The unspecified limit is guaranteed to be at least $2^{31}-1$.

Definition at line 67 of file [semaphore](#).

References [L4::Triggerable::trigger\(\)](#).

Here is the call graph for this function:



The documentation for this struct was generated from the following file:

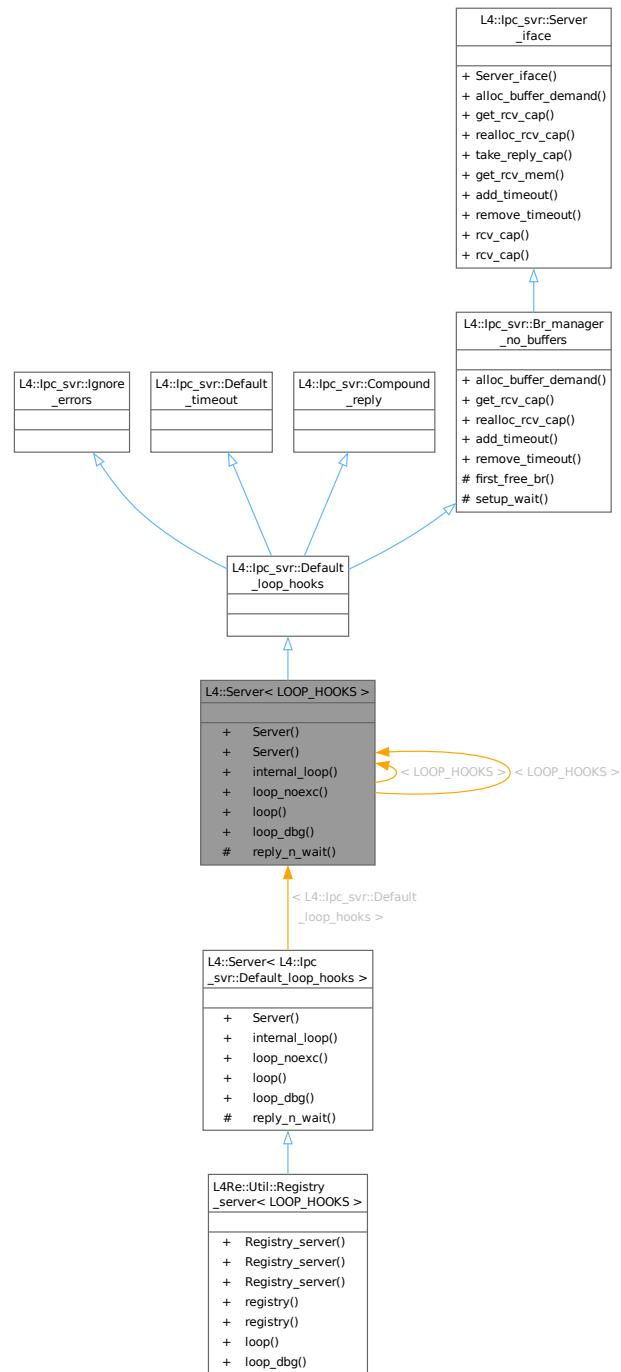
- [l4/sys/semaphore](#)

15.192 L4::Server< LOOP_HOOKS > Class Template Reference

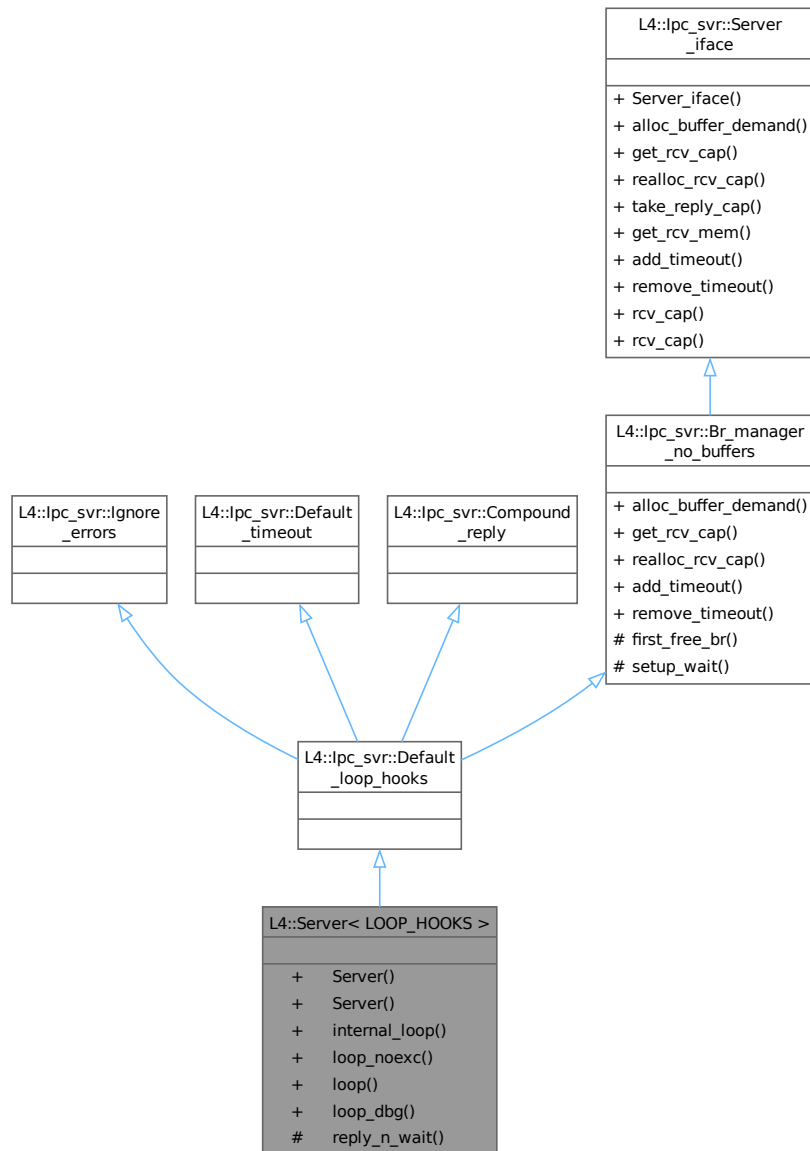
Basic server loop for handling client requests.

```
#include <ipc_server_loop>
```

Inheritance diagram for L4::Server< LOOP_HOOKS >:



Collaboration diagram for L4::Server< LOOP_HOOKS >:



Public Member Functions

- [Server](#) ([l4_utcb_t](#) *)
Initializes the server loop.
- **Server** ()
Initializes the server loop.
- `template<typename DISPATCH>`
[L4_NORETURN](#) void [internal_loop](#) (`DISPATCH` dispatch, [l4_utcb_t](#) *)
The server loop.
- `template<typename R>`
[L4_NORETURN](#) void **loop_noexc** (`R` r, [l4_utcb_t](#) *u=[l4_utcb](#)())
Server loop without exception handling.

- `template<typename EXC, typename R>`
`L4::NORETURN void loop (R r, l4_utcb_t *u=l4_utcb())`
Server loop with internal exception handling.
- `template<typename EXC, typename R, typename Printer>`
`L4::NORETURN void loop_dbg (R r, Printer p, l4_utcb_t *u=l4_utcb())`
Server loop with internal exception handling including message printing.

Public Member Functions inherited from `L4::lpc_svr::Br_manager_no_buffers`

- `int alloc_buffer_demand (Demand const &demand) override`
Tells the server to allocate buffers for the given demand.
- `L4::Cap< void > get_rcv_cap (int) const override`
Returns `L4::Cap<void>::Invalid`, we have no buffer management.
- `int realloc_rcv_cap (int) override`
Returns `-L4_ENOMEM`, we have no buffer management.
- `int add_timeout (Timeout *, l4_kernel_clock_t) override`
Returns `-L4_ENOSYS`, we have no timeout queue.
- `int remove_timeout (Timeout *) override`
Returns `-L4_ENOSYS`, we have no timeout queue.

Public Member Functions inherited from `L4::lpc_svr::Server_iface`

- `Server_iface ()`
Make a server interface.
- `virtual cxx::Result< L4::Reply_cap > take_reply_cap () noexcept`
Take the currently used reply capability.
- `virtual cxx::Result< Mem_window > get_rcv_mem () noexcept`
Take the current memory receive window.
- `template<typename T>`
`L4::Cap< T > rcv_cap (int index) const`
Get given receive buffer as typed capability.
- `L4::Cap< void > rcv_cap (int index) const`
Get receive cap with the given index as generic (void) type.

Protected Member Functions

- `l4_msgtag_t reply_n_wait (l4_msgtag_t reply, l4_umword_t *p, l4_utcb_t *)`
Internal implementation for reply and wait.

Protected Member Functions inherited from `L4::lpc_svr::Br_manager_no_buffers`

- `unsigned first_free_br () const`
Returns 1 as first free buffer.
- `void setup_wait (l4_utcb_t *utcb, L4::lpc_svr::Reply_mode)`
Setup wait function for the server loop (`Server<>`).

Additional Inherited Members

Public Types inherited from [L4::lpc_svr::Server_iface](#)

- using **Demand** = [L4::Type_info::Demand](#)
Data type expressing server-side demand for receive buffers.

15.192.1 Detailed Description

```
template<typename LOOP_HOOKS = lpc_svr::Default_loop_hooks>
class L4::Server< LOOP_HOOKS >
```

Basic server loop for handling client requests.

Parameters

| | |
|-------------------|--|
| <i>LOOP_HOOKS</i> | the server inherits from LOOP_HOOKS and calls the hooks defined in LOOP_HOOKS in the server loop. See lpc_svr::Default_loop_hooks , lpc_svr::Ignore_errors , lpc_svr::Default_timeout , lpc_svr::Compound_reply , and lpc_svr::Br_manager_no_buffers . |
|-------------------|--|

This is basically a simple server loop that uses a single message buffer for receiving requests and sending replies. The dispatcher determines how incoming messages are handled.

Definition at line 307 of file [ipc_server_loop](#).

15.192.2 Constructor & Destructor Documentation

15.192.2.1 Server()

```
template<typename LOOP_HOOKS = lpc_svr::Default_loop_hooks>
L4::Server< LOOP_HOOKS >::Server (
    l4_utcb_t * ) [inline], [explicit]
```

Initializes the server loop.

Note that this variant is deprecated. The UTCB can be specified with the server loop function ([l4_utcb\(\)](#) is the default). (2021-10)

Definition at line 319 of file [ipc_server_loop](#).

15.192.3 Member Function Documentation

15.192.3.1 internal_loop()

```
template<typename L>
template<typename DISPATCH>
L4_NORETURN void L4::Server< L >::internal_loop (
    DISPATCH dispatch,
    l4_utcb_t * utcb) [inline]
```

The server loop.

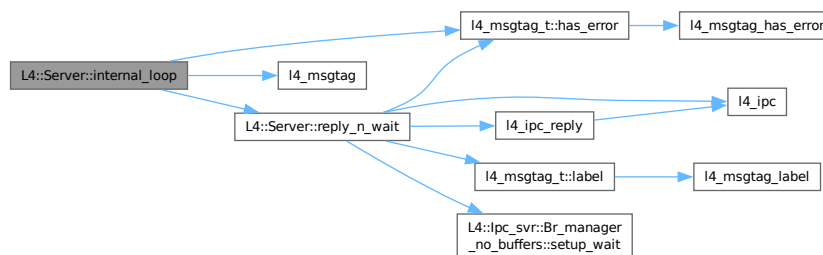
This function usually never returns, it waits for incoming messages calls the dispatcher, sends a reply and waits again.

Definition at line 424 of file [ipc_server_loop](#).

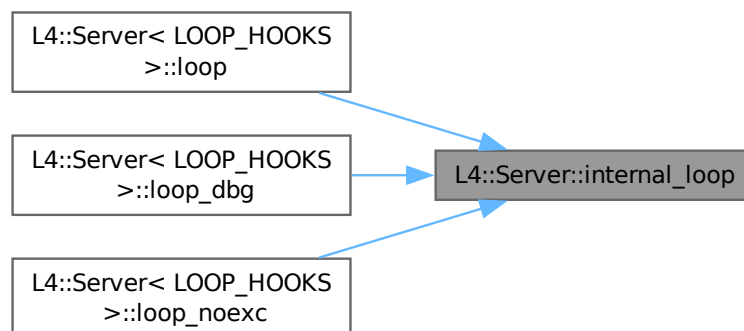
References [l4_msgtag_t::has_error\(\)](#), [L4_ENOREPLY](#), [l4_msgtag\(\)](#), and [reply_n_wait\(\)](#).

Referenced by [loop\(\)](#), [loop_dbg\(\)](#), and [loop_noexc\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.192.3.2 loop()

```
template<typename LOOP_HOOKS = Ipc_svr::Default_loop_hooks>
template<typename EXC, typename R>
L4_NORETURN void L4::Server< LOOP_HOOKS >::loop (
    R r,
    l4_utcb_t * u = l4_utcb()) [inline]
```

Server loop with internal exception handling.

This server loop translates [L4::Runtime_error](#) exceptions into negative error return codes sent to the caller.

Definition at line 355 of file [ipc_server_loop](#).

15.192.3.3 loop_dbg()

```
template<typename LOOP_HOOKS = Ipc_svr::Default_loop_hooks>
template<typename EXC, typename R, typename Printer>
L4_NORETURN void L4::Server< LOOP_HOOKS >::loop_dbg (
    R r,
    Printer p,
    l4_utcb_t * u = l4_utcb()) [inline]
```

Server loop with internal exception handling including message printing.

Exceptions are translated into error codes, just like in [loop\(\)](#). In addition, error messages are printed using the Printer argument.

Definition at line 368 of file [ipc_server_loop](#).

The documentation for this class was generated from the following file:

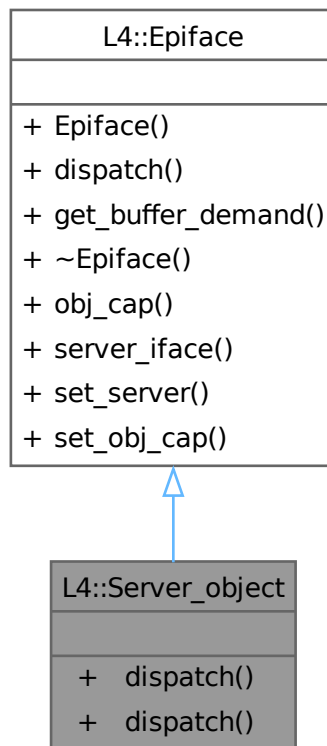
- [l4/sys/cxx/ipc_server_loop](#)

15.193 L4::Server_object Class Reference

Abstract server object to be used with [L4::Server](#) and [L4::Basic_registry](#).

```
#include <ipc_server>
```


Collaboration diagram for L4::Server_object:



Public Member Functions

- virtual int [dispatch](#) (unsigned long rights, [lpc::lostream](#) &ios)=0
The abstract handler for client requests to the object.
- [l4_msgtag_t](#) [dispatch](#) ([l4_msgtag_t](#) tag, unsigned rights, [l4_utcb_t](#) *utcb) override
The abstract handler for client requests to the object.

Public Member Functions inherited from [L4::Epiface](#)

- **Epiface** ()
Make a server object.
- virtual [Demand](#) [get_buffer_demand](#) () const =0
Get the server-side receive buffer demand for this object.
- virtual **~Epiface** ()=0
Destroy the object.
- Stored_cap [obj_cap](#) () const
Get the capability to the kernel object belonging to this object.
- [Server_iface](#) * [server_iface](#) () const
Get pointer to server interface at which the object is currently registered.
- int [set_server](#) ([Server_iface](#) *srv, [Cap](#)< void > cap, bool managed=false)
Set server registration info for the object.
- void **set_obj_cap** ([Cap](#)< void > const &cap)
Deprecated server registration function.

Additional Inherited Members

Public Types inherited from [L4::Epiface](#)

- using **Server_iface** = [lpc_svr::Server_iface](#)
Type for abstract server interface.
- using **Demand** = [lpc_svr::Server_iface::Demand](#)
Type for server-side receive buffer demand.

15.193.1 Detailed Description

Abstract server object to be used with [L4::Server](#) and [L4::Basic_registry](#).

Note

Usually [L4::Server_object_t](#) is used as a base class when writing server objects.

This server object provides an abstract interface that is used by the [L4::Registry_dispatcher](#) model. You can derive subclasses from this interface and implement application specific server objects.

Definition at line 38 of file [ipc_server](#).

15.193.2 Member Function Documentation

15.193.2.1 `dispatch()` [1/2]

```
l4_msgtag_t L4::Server_object::dispatch (
    l4_msgtag_t tag,
    unsigned rights,
    l4_utcb_t * utcb) [inline], [override], [virtual]
```

The abstract handler for client requests to the object.

Parameters

| | |
|---------------|--|
| <i>tag</i> | The message tag for this invocation. |
| <i>rights</i> | The rights bits in the invoked capability. |
| <i>utcb</i> | The UTCB used for the invocation. |

Return values

| | |
|---------------------------|-----------------------------------|
| <code>-L4_ENOREPLY</code> | No reply message is sent. |
| <code><0</code> | Error, reply with error code. |
| <code>>=0</code> | Success, reply with return value. |

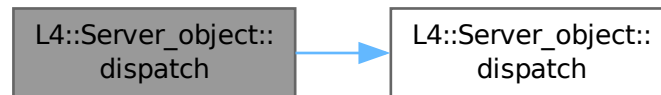
This function must be implemented by application specific server objects.

Implements [L4::Epiface](#).

Definition at line 60 of file [ipc_server](#).

References [dispatch\(\)](#).

Here is the call graph for this function:



15.193.2.2 dispatch() [2/2]

```
virtual int L4::Server_object::dispatch (
    unsigned long rights,
    Ipc::Iostream & ios) [pure virtual]
```

The abstract handler for client requests to the object.

Parameters

| | |
|---------------|--|
| <i>rights</i> | The rights bits in the invoked capability. |
| <i>ios</i> | The Ipc::Iostream for reading the request and writing the reply. |

Return values

| | |
|---------------------------|--|
| <code>-L4_ENOREPLY</code> | Instructs the server loop to not send a reply. |
| <code><0</code> | Error, reply with error code. |
| <code>>=0</code> | Success, reply with return value. |

This function must be implemented by application specific server objects. The implementation must unmarshall data from the stream (`ios`) and create a reply by marshalling to the stream (`ios`). For details about the IPC stream see [IPC stream operators](#).

Note

You need to extract the complete message from the `ios` stream before inserting any reply data or before doing any function call that may use the UTCB. Otherwise, the incoming message may get lost.

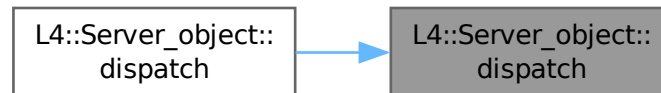
Implemented in [L4::Server_object_x< Derived, IFACE, BASE >](#).

Examples

[examples/libs/l4re/c++/shared_ds/ds_srv.cc](#), and [examples/libs/l4re/streammap/server.cc](#).

Referenced by [dispatch\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

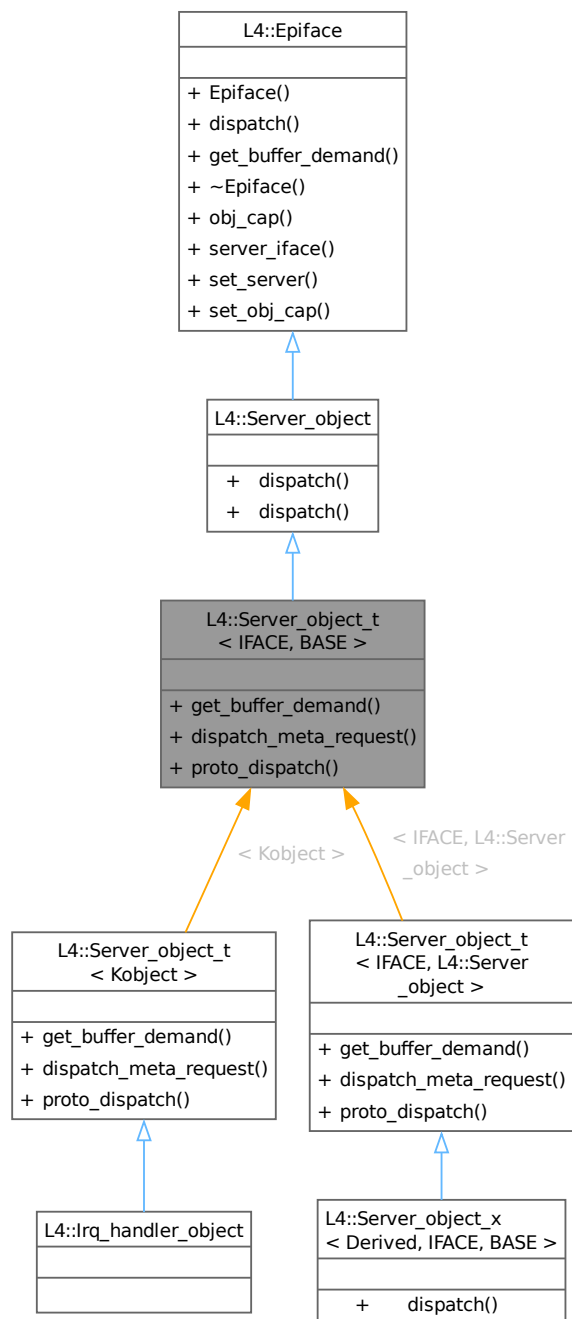
- [l4/cxx/ipc_server](#)

15.194 L4::Server_object_t< IFACE, BASE > Struct Template Reference

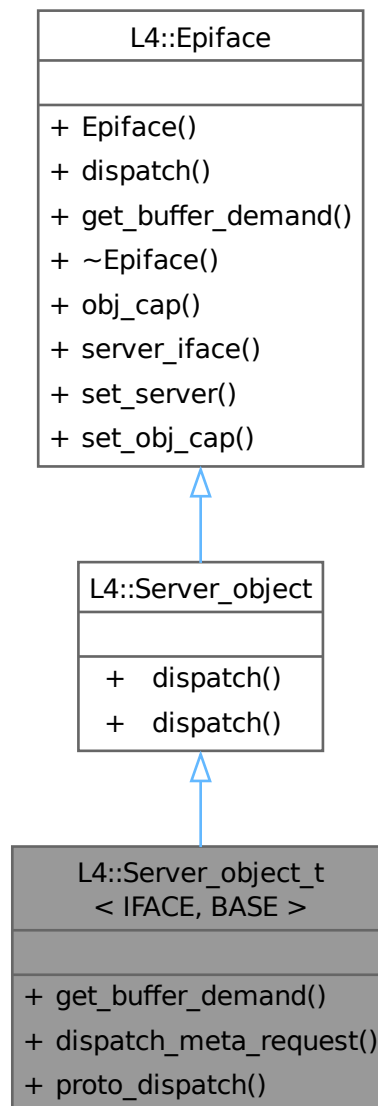
Base class (template) for server implementing server objects.

```
#include <ipc_server>
```


Inheritance diagram for L4::Server_object_t< IFACE, BASE >:



Collaboration diagram for L4::Server_object_t< IFACE, BASE >:



Public Types

- typedef IFACE **Interface**
Data type of the IPC interface definition.

Public Types inherited from L4::Epiface

- using **Server_iface** = [lpc_svr::Server_iface](#)
Type for abstract server interface.
- using **Demand** = [lpc_svr::Server_iface::Demand](#)
Type for server-side receive buffer demand.

Public Member Functions

- BASE::Demand [get_buffer_demand](#) () const override
- int [dispatch_meta_request](#) (L4::lpc::lostream &ios)

Implementation of the meta protocol based on IFACE.

Public Member Functions inherited from L4::Server_object

- virtual int [dispatch](#) (unsigned long rights, lpc::lostream &ios)=0
The abstract handler for client requests to the object.
- [l4_msgtag_t](#) [dispatch](#) ([l4_msgtag_t](#) tag, unsigned rights, [l4_utcb_t](#) *utcb) override
The abstract handler for client requests to the object.

Public Member Functions inherited from L4::Epiface

- **Epiface** ()
Make a server object.
- virtual ~**Epiface** ()=0
Destroy the object.
- Stored_cap [obj_cap](#) () const
Get the capability to the kernel object belonging to this object.
- [Server_iface](#) * [server_iface](#) () const
Get pointer to server interface at which the object is currently registered.
- int [set_server](#) ([Server_iface](#) *srv, [Cap](#)< void > cap, bool managed=false)
Set server registration info for the object.
- void [set_obj_cap](#) ([Cap](#)< void > const &cap)
Deprecated server registration function.

Static Public Member Functions

- template<typename THIS>
static int [proto_dispatch](#) (THIS *self, [l4_umword_t](#) rights, L4::lpc::lostream &ios)
Implementation of protocol-based dispatch for this server object.

15.194.1 Detailed Description

```
template<typename IFACE, typename BASE = L4::Server_object>
struct L4::Server_object_t< IFACE, BASE >
```

Base class (template) for server implementing server objects.

Template Parameters

| | |
|--------------|---|
| <i>IFACE</i> | The IPC interface class that defines the interface that shall be implemented. |
| <i>BASE</i> | The server object base class (usually L4::Server_object). |

Examples

[examples/libs/l4re/c++/shared_ds/ds_srv.cc](#), and [examples/libs/l4re/streammap/server.cc](#).

Definition at line 80 of file [ipc_server](#).

15.194.2 Member Function Documentation

15.194.2.1 `dispatch_meta_request()`

```
template<typename IFACE, typename BASE = L4::Server_object>
int L4::Server_object_t< IFACE, BASE >::dispatch_meta_request (
    L4::Ipc::Iostream & ios) [inline]
```

Implementation of the meta protocol based on *IFACE*.

Parameters

| | |
|------------|---|
| <i>ios</i> | The IO stream used for receiving the message. |
|------------|---|

This function can be used to handle incoming [L4_PROTO_META](#) protocol requests. The implementation uses the [L4::Type_info](#) of *IFACE* to handle the requests. Call this function in the implementation of [Server_object::dispatch\(\)](#) when the received message tag has protocol [L4_PROTO_META](#) ([L4::Meta::Protocol](#)).

Definition at line 99 of file [ipc_server](#).

15.194.2.2 `get_buffer_demand()`

```
template<typename IFACE, typename BASE = L4::Server_object>
BASE::Demand L4::Server_object_t< IFACE, BASE >::get_buffer_demand () const [inline], [override],
[virtual]
```

Returns

the server-side buffer demand based in *IFACE*.

Implements [L4::Epiface](#).

Definition at line 86 of file [ipc_server](#).

15.194.2.3 `proto_dispatch()`

```
template<typename IFACE, typename BASE = L4::Server_object>
template<typename THIS>
int L4::Server_object_t< IFACE, BASE >::proto_dispatch (
    THIS * self,
    l4_umword_t rights,
    L4::Ipc::Iostream & ios) [inline], [static]
```

Implementation of protocol-based dispatch for this server object.

Parameters

| | |
|-------------|---|
| <i>self</i> | The this pointer for the object (inherits from Server_object_t). |
|-------------|---|

| | |
|---------------|---|
| <i>rights</i> | The rights from the received IPC (forwarded to p_dispatch()). |
| <i>ios</i> | The message stream for the incoming and the reply message. |

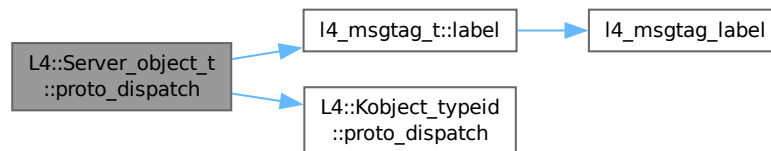
Server objects may call this function from their [dispatch\(\)](#) function. This function reads the protocol ID from the message tag and uses the p_dispatch code to dispatch to overloaded p_dispatch functions of self.

Definition at line 114 of file [ipc_server](#).

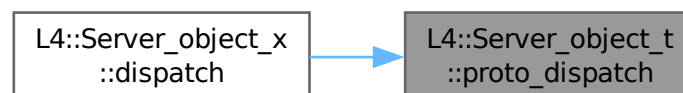
References [l4_msgtag_t::label\(\)](#), and [L4::Kobject_typeid< T >::proto_dispatch\(\)](#).

Referenced by [L4::Server_object_x< Derived, IFACE, BASE >::dispatch\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this struct was generated from the following file:

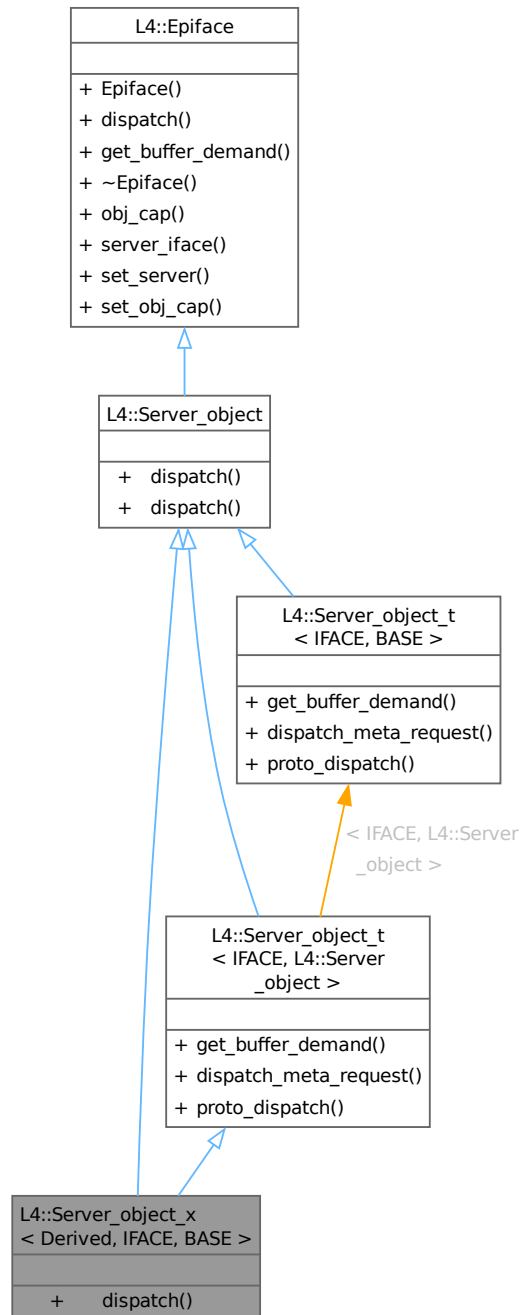
- [l4/cxx/ipc_server](#)

15.195 L4::Server_object_x< Derived, IFACE, BASE > Struct Template Reference

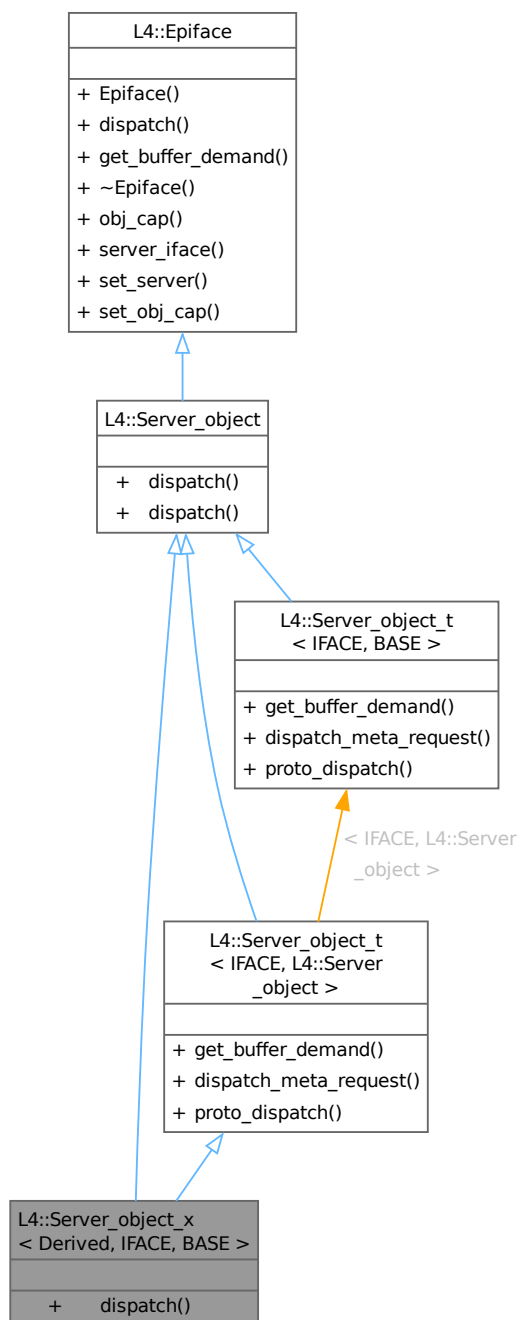
Helper class to implement p_dispatch based server objects.

```
#include <ipc_server>
```

Inheritance diagram for L4::Server_object_x< Derived, IFACE, BASE >:



Collaboration diagram for L4::Server_object_x< Derived, IFACE, BASE >:



Public Member Functions

- `int dispatch (l4_umword_t r, L4::lpc::lostream &ios)`
Implementation forwarding to `p_dispatch()`.

Public Member Functions inherited from L4::Server_object

- `l4_msgtag_t dispatch (l4_msgtag_t tag, unsigned rights, l4_utcb_t *utcb)` override

The abstract handler for client requests to the object.

Public Member Functions inherited from [L4::Epiface](#)

- **Epiface** ()
Make a server object.
- virtual **~Epiface** ()=0
Destroy the object.
- Stored_cap **obj_cap** () const
Get the capability to the kernel object belonging to this object.
- **Server_iface** * **server_iface** () const
Get pointer to server interface at which the object is currently registered.
- int **set_server** (**Server_iface** *srv, **Cap**< void > cap, bool managed=false)
Set server registration info for the object.
- void **set_obj_cap** (**Cap**< void > const &cap)
Deprecated server registration function.

Public Member Functions inherited from [L4::Server_object_t< IFACE, L4::Server_object >](#)

- [L4::Server_object::Demand](#) **get_buffer_demand** () const override
- int **dispatch_meta_request** ([L4::lpc::lostream](#) &ios)
Implementation of the meta protocol based on IFACE.

Additional Inherited Members

Public Types inherited from [L4::Epiface](#)

- using **Server_iface** = [lpc_svr::Server_iface](#)
Type for abstract server interface.
- using **Demand** = [lpc_svr::Server_iface::Demand](#)
Type for server-side receive buffer demand.

Public Types inherited from [L4::Server_object_t< IFACE, L4::Server_object >](#)

- typedef IFACE **Interface**
Data type of the IPC interface definition.

Static Public Member Functions inherited from [L4::Server_object_t< IFACE, L4::Server_object >](#)

- static int **proto_dispatch** (THIS *self, [l4_umword_t](#) rights, [L4::lpc::lostream](#) &ios)
Implementation of protocol-based dispatch for this server object.

15.195.1 Detailed Description

```
template<typename Derived, typename IFACE, typename BASE = L4::Server_object>
struct L4::Server_object_x< Derived, IFACE, BASE >
```

Helper class to implement p_dispatch based server objects.

Template Parameters

| | |
|----------------|---|
| <i>Derived</i> | The data type of your server object class. |
| <i>IFACE</i> | The data type providing the interface definition for the object. |
| <i>BASE</i> | Optional data-type of the base server object (usually L4::Server_object) |

This class implements the standard [dispatch\(\)](#) function of [L4::Server_object](#) and forwards incoming messages to a set of overloaded p_dispatch() functions. There must be a p_dispatch() function in Derived for each interface provided by IFACE with the signature

```
int p_dispatch(Iface *, unsigned rights, L4::Ipc::Iostream &)
```

that is called for messages with protocol == Iface::Protocol.

Example signature for [L4Re::Dataspace](#) is:

```
int p_dispatch(L4Re::Dataspace *, unsigned, L4::Ipc::Iostream &)
```

Definition at line [143](#) of file [ipc_server](#).

The documentation for this struct was generated from the following file:

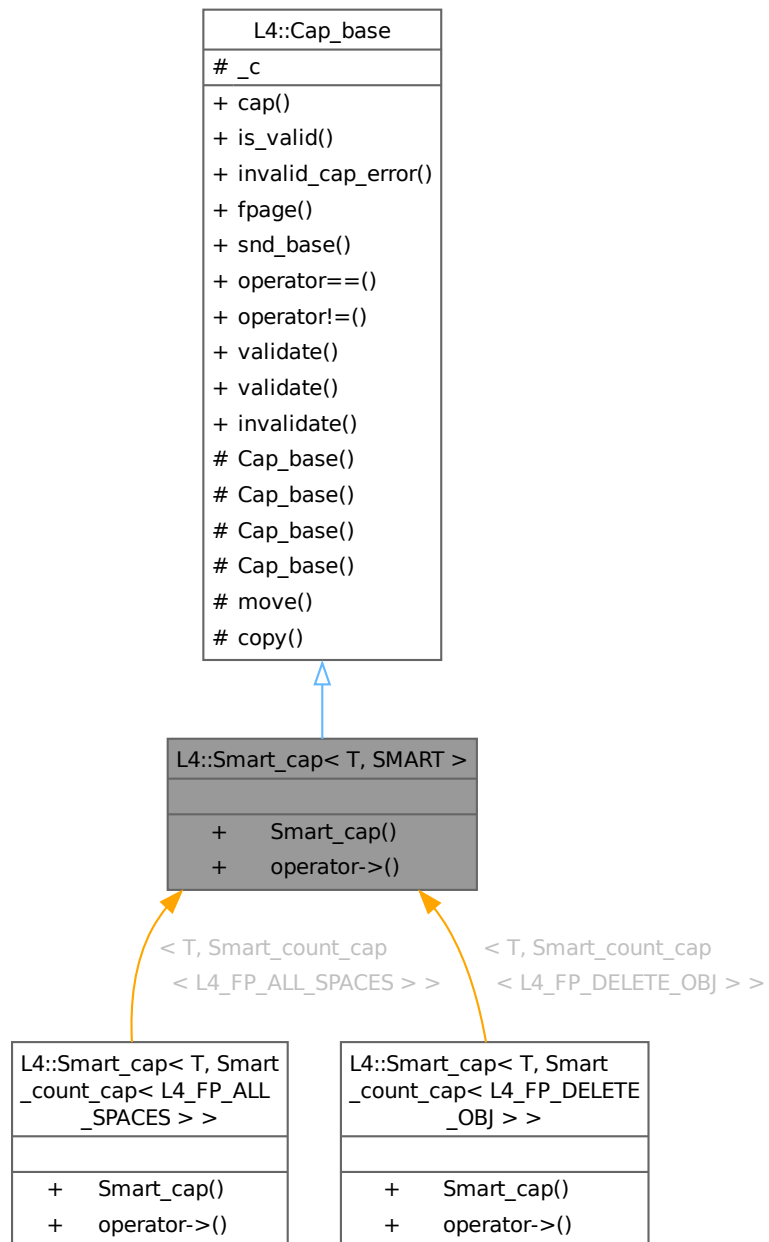
- [l4/cxx/ipc_server](#)

15.196 L4::Smart_cap< T, SMART > Class Template Reference

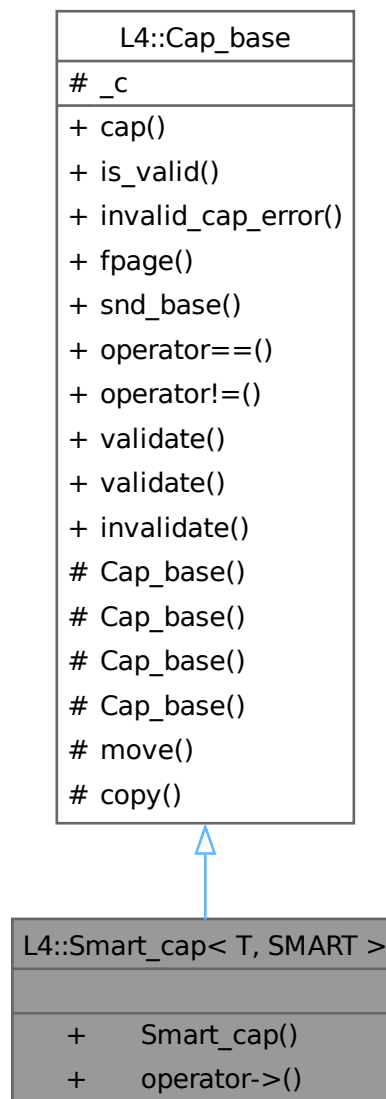
Smart capability class.

```
#include <smart_capability>
```

Inheritance diagram for L4::Smart_cap< T, SMART >:



Collaboration diagram for L4::Smart_cap< T, SMART >:



Public Member Functions

- `template<typename O>`
`Smart_cap (Cap< O > const &p) noexcept`
Internal constructor, use to generate a capability from a `this` pointer.
- `Cap< T > operator-> () const noexcept`
Member access of a `T`.

Public Member Functions inherited from [L4::Cap_base](#)

- `l4_cap_idx_t cap () const noexcept`

- Return capability selector.*

 - bool **is_valid** () const noexcept

Test whether the capability is a valid capability index (i.e., not L4_INVALID_CAP).
- int **invalid_cap_error** () const noexcept

Return the transported error code in an invalid capability index.
- **l4_fpage_t** **fpage** (unsigned rights=L4_CAP_FPAGE_RWS) const noexcept

Return flexpage for the capability.
- **l4_umword_t** **snd_base** (unsigned grant=L4_MAP_ITEM_MAP, **l4_cap_idx_t** base=L4_INVALID_CAP) const noexcept

Return send base.
- bool **operator==** (**Cap_base** const &o) const noexcept

Test if two capabilities are equal.
- bool **operator!=** (**Cap_base** const &o) const noexcept

Test if two capabilities are not equal.
- **l4_msgtag_t** **validate** (**l4_utcb_t** *u=**l4_utcb**()) const noexcept

Check whether a capability is present (refers to an object).
- **l4_msgtag_t** **validate** (**Cap**< **Task** > task, **l4_utcb_t** *u=**l4_utcb**()) const noexcept

Check whether a capability is present (refers to an object).
- void **invalidate** () noexcept

Set this capability to invalid (L4_INVALID_CAP).

Additional Inherited Members

Public Types inherited from L4::Cap_base

- enum **No_init_type** { **No_init** }
- Special value for uninitialized capability objects.*
- enum **Cap_type** { **Invalid** = L4_INVALID_CAP }
- Invalid capability type.*

Protected Member Functions inherited from L4::Cap_base

- **Cap_base** (**l4_cap_idx_t** c) noexcept
- Generate a capability from its C representation.*
- **Cap_base** (**Cap_type** cap) noexcept
- Constructor to create an invalid capability.*
- **Cap_base** (**l4_default_caps_t** cap) noexcept
- Initialize capability with one of the default capabilities.*
- **Cap_base** () noexcept
- Create an uninitialized instance.*
- void **move** (**Cap_base** const &src) const
- Replace this capability with the contents of src.*
- void **copy** (**Cap_base** const &src) const
- Copy a capability.*

Protected Attributes inherited from L4::Cap_base

- **l4_cap_idx_t** **_c**
- The C representation of a capability selector.*

15.196.1 Detailed Description

```
template<typename T, typename SMART>
class L4::Smart_cap< T, SMART >
```

Smart capability class.

Definition at line 25 of file [smart_capability](#).

15.196.2 Constructor & Destructor Documentation

15.196.2.1 Smart_cap()

```
template<typename T, typename SMART>
template<typename O>
L4::Smart_cap< T, SMART >::Smart_cap (
    Cap< O > const & p) [inline], [noexcept]
```

Internal constructor, use to generate a capability from a `this` pointer.

Attention

This constructor is only useful to generate a capability from the `this` pointer of an object that is an [L4::Kobject](#). Do `never` use this constructor for something else!

Parameters

| | |
|----------|--|
| <i>p</i> | The <code>this</code> pointer of the Kobject or derived object |
|----------|--|

Definition at line 62 of file [smart_capability](#).

The documentation for this class was generated from the following file:

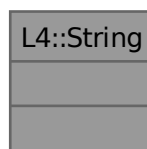
- [l4/sys/smart_capability](#)

15.197 L4::String Class Reference

A null-terminated string container class.

```
#include <string.h>
```

Collaboration diagram for L4::String:



15.197.1 Detailed Description

A null-terminated string container class.

Definition at line 22 of file [string.h](#).

The documentation for this class was generated from the following file:

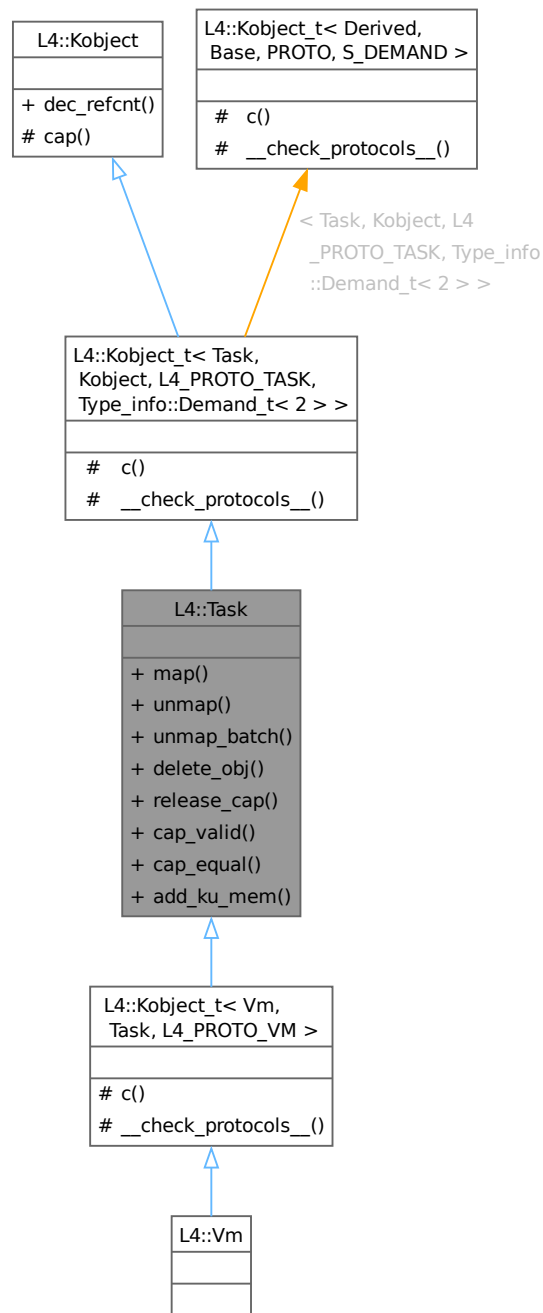
- [l4/cxx/string.h](#)

15.198 L4::Task Class Reference

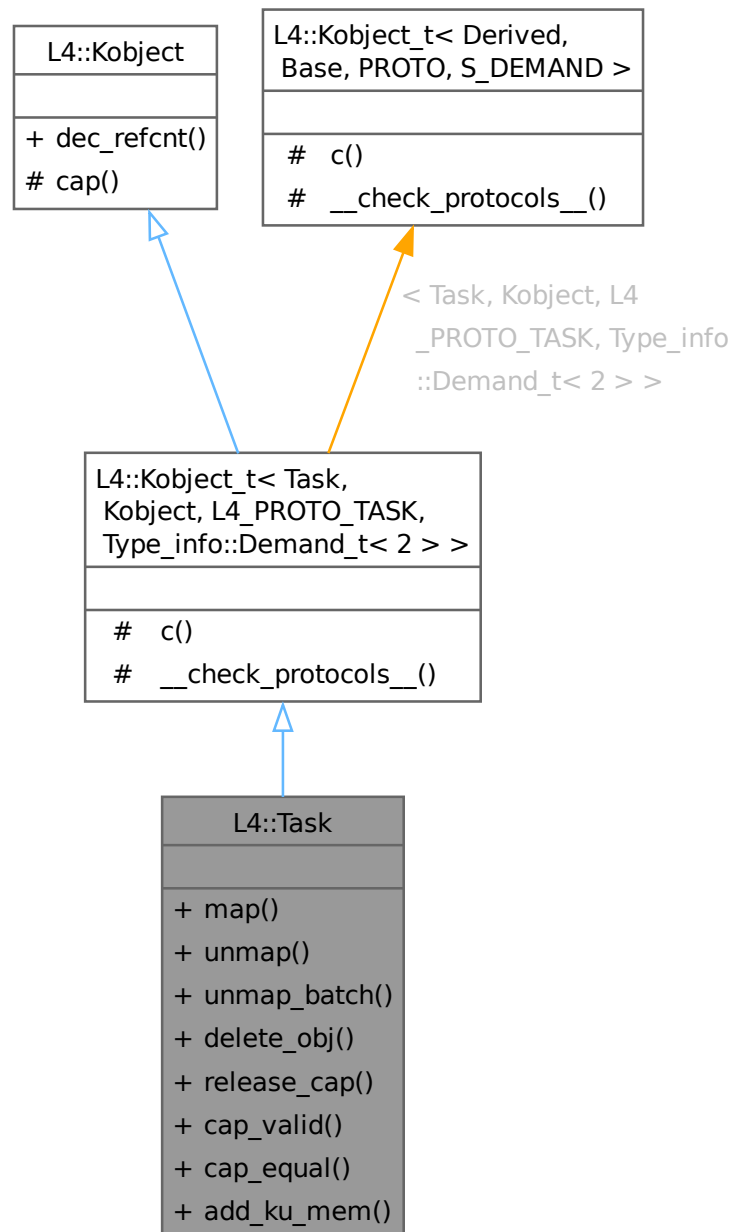
C++ interface of the [Task](#) kernel object, see [Task](#) for the C interface.

```
#include <task>
```

Inheritance diagram for L4::Task:



Collaboration diagram for L4::Task:



Public Member Functions

- `l4_msgtag_t map` (`Cap< Task > const &src_task`, `l4_fpage_t const &snd_fpage`, `l4_umword_t snd_base`, `l4_utcb_t *utcb=l4_utcb()`) noexcept

Map resources available in the source task to a destination task.

- `l4_msgtag_t unmap` (`l4_fpage_t const &fpage`, `l4_umword_t map_mask`, `l4_utcb_t *utcb=l4_utcb()`) noexcept

Revoke rights from the task.

- [l4_msgtag_t unmap_batch](#) ([l4_fpage_t](#) const *fpages, unsigned num_fpages, [l4_umword_t](#) map_mask, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) noexcept
Revoke rights from a task.
- [l4_msgtag_t delete_obj](#) ([L4::Cap](#)< void > obj, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) noexcept
Release capability and delete object.
- [l4_msgtag_t release_cap](#) ([L4::Cap](#)< void > cap, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) noexcept
Release object capability.
- [l4_msgtag_t cap_valid](#) ([Cap](#)< void > const &cap, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) noexcept
Check whether a capability is present (refers to an object).
- [l4_msgtag_t cap_equal](#) ([Cap](#)< void > const &cap_a, [Cap](#)< void > const &cap_b, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) noexcept
Test whether two capabilities point to the same object with the same permissions (only considering selected permissions).
- [l4_msgtag_t add_ku_mem](#) ([l4_fpage_t](#) *fpage, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) noexcept
Add kernel-user memory.

Public Member Functions inherited from [L4::Kobject](#)

- [l4_msgtag_t dec_refcnt](#) ([l4_mword_t](#) diff, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#))
Decrement the in kernel reference counter for the object.

Additional Inherited Members

Protected Types inherited from

[L4::Kobject_t](#)< [Task](#), [Kobject](#), [L4_PROTO_TASK](#), [Type_info::Demand_t](#)< 2 > >

- typedef [Task](#) **Class**
The target interface type (inheriting from [Kobject_t](#)).
- typedef [Typeid::Iface](#)< [PROTO](#), [Task](#) > **__Iface**
The interface description for the derived class.
- typedef [Typeid::Merge_list](#)< [Typeid::Iface_list](#)< **__Iface** >, typename [Kobject::__Iface_list](#) > **__Iface_list**
The list of all RPC interfaces provided directly or through inheritance.

Protected Member Functions inherited from

[L4::Kobject_t](#)< [Task](#), [Kobject](#), [L4_PROTO_TASK](#), [Type_info::Demand_t](#)< 2 > >

- [L4::Cap](#)< [Class](#) > **c** () const noexcept
Get the capability to ourselves.

Protected Member Functions inherited from [L4::Kobject](#)

- [l4_cap_idx_t](#) **cap** () const noexcept
Return capability selector.

Static Protected Member Functions inherited from

[L4::Kobject_t](#)< [Task](#), [Kobject](#), [L4_PROTO_TASK](#), [Type_info::Demand_t](#)< 2 > >

- static void **__check_protocols** () noexcept
Helper to check for protocol conflicts.

15.198.1 Detailed Description

C++ interface of the [Task](#) kernel object, see [Task](#) for the C interface.

The [L4::Task](#) class represents a combination of the address spaces provided by the [L4Re](#) micro kernel. A task consists of at least a memory address space and an object address space. On IA32 there is also an IO-port address space associated with an [L4::Task](#).

[L4::Task](#) objects are created using the [L4::Factory](#) interface.

Include File

```
#include <l4/sys/task>
```

Definition at line 33 of file [task](#).

15.198.2 Member Function Documentation

15.198.2.1 add_ku_mem()

```
l4_msgtag_t L4::Task::add_ku_mem (
    l4_fpage_t * fpage,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
```

Add kernel-user memory.

Parameters

| | | |
|----------------|--------------|---|
| <i>in, out</i> | <i>fpage</i> | Flexpage describing the virtual area the memory goes to. On systems without MMU, the flexpage is adjusted to reflect the actually allocated physical address. |
| | <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb . |

Returns

Syscall return tag

Kernel-user memory (ku_mem) is memory that is shared between the kernel and user-space. It is needed for the UTCB area of threads (see [L4::Thread::Attr::bind\(\)](#)) and for (extended) vCPU state. Note that existing kernel-user memory cannot be unmapped or mapped somewhere else.

Note

The amount of kernel-user memory that can be allocated at once is limited by the used kernel implementation. The minimum allocatable amount is one page ([L4_PAGESIZE](#)). A portable implementation should not depend on allocations greater than 16KiB to succeed.

This function is only guaranteed to work on [L4::Task](#) objects. It might or might not work on [L4::Vm](#) objects or on [L4Re::Dma_space](#) objects but there is no practical use for adding kernel-user memory to [L4::Vm](#) objects or to [L4Re::Dma_space](#) objects.

Definition at line 281 of file [task](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:

**15.198.2.2 cap_equal()**

```

l4_msgtag_t L4::Task::cap_equal (
    Cap< void > const & cap_a,
    Cap< void > const & cap_b,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
  
```

Test whether two capabilities point to the same object with the same permissions (only considering selected permissions).

Parameters

| | |
|------------------------|--|
| <i>cap_a</i> | Capability selector for the first capability to compare. |
| <i>cap_b</i> | Capability selector for the second capability to compare. |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb . |

Return values

| | |
|---------------------------------|--|
| <i>l4_msgtag_t::label()</i> = 1 | The compared capabilities point to the same object with same considered permission. |
| <i>l4_msgtag_t::label()</i> = 0 | The compared capabilities do not point to the same object or differ in the considered permission. |

- For [L4::lpc_gate](#) objects, only the permissions [L4_CAP_FPAGE_W](#), [L4_CAP_FPAGE_S](#), and [L4_FPAGE_C_OBJ_RIGHT1](#) are considered for the comparison. Differences in other permissions are ignored.

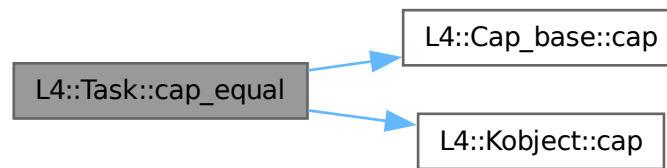
- For other objects, only the permissions [L4_CAP_FPAGE_W](#) and [L4_CAP_FPAGE_S](#) are considered for the comparison. Differences in other permissions are ignored.

Note that having the [L4_CAP_FPAGE_R](#) permission is implicit in possessing the capability.

Definition at line 251 of file [task](#).

References [L4::Cap_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



15.198.2.3 cap_valid()

```

l4_msgtag_t L4::Task::cap_valid (
    Cap< void > const & cap,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
  
```

Check whether a capability is present (refers to an object).

Parameters

| | |
|-------------|--|
| <i>cap</i> | Valid capability to check for presence. |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb . |

Return values

| | |
|----------------------------------|--|
| <i>l4_msgtag_t::label()</i> > 0 | Capability is present (refers to an object). |
| <i>l4_msgtag_t::label()</i> == 0 | No capability present (void object). |

A capability is considered present when it refers to an existing kernel object.

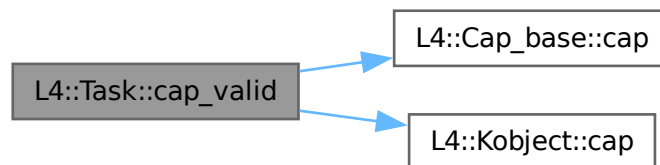
Precondition

`cap` must be a valid capability (i.e. `cap.is_valid() == true`). If you are unsure about the validity of your capability use [L4::Cap.validate\(\)](#) instead.

Definition at line 222 of file [task](#).

References [L4::Cap_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:

**15.198.2.4 delete_obj()**

```

l4_msgtag_t L4::Task::delete_obj (
    L4::Cap< void > obj,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
  
```

Release capability and delete object.

Parameters

| | |
|-------------|--|
| <i>obj</i> | Capability index of the object to delete. |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb . |

Returns

Syscall return tag

If `obj` has the delete permission, initiates the deletion of the object. This implies that all capabilities for that object are gone afterwards. However, kernel-internally, objects are not destroyed until all other kernel objects holding a reference to it drop the reference. Hence, quota used by that object might not be freed immediately.

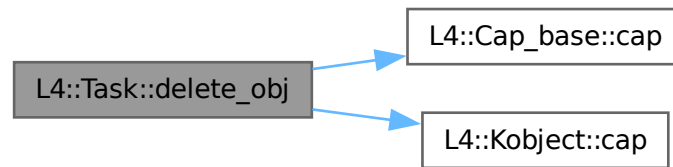
If `obj` does not have the delete permission, no error will be reported and only the capability `obj` is removed. (Note that, depending on the object's reference counter, this might still imply initiation of deletion.)

This operation is equivalent to [unmap\(\)](#) with [L4_FP_DELETE_OBJ](#) flag.

Definition at line 180 of file [task](#).

References [L4::Cap_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



15.198.2.5 map()

```

l4_msgtag_t L4::Task::map (
    Cap< Task > const & src_task,
    l4_fpage_t const & snd_fpage,
    l4_umword_t snd_base,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
  
```

Map resources available in the source task to a destination task.

Parameters

| | |
|------------------|--|
| <i>src_task</i> | Capability selector of the source task. |
| <i>snd_fpage</i> | Send flexpage that describes an area in the address space or object space of the source task. |
| <i>snd_base</i> | Send base that describes an offset in the receive window of the destination task. The lower bits contain additional map control flags (see l4_fpage_cacheability_opt_t for memory mappings, Attributes and additional permissions for object send items for object mappings, and L4_MAP_ITEM_GRANT ; also see l4_map_control() and l4_map_obj_control()). |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb . |

Returns

Syscall return tag. The function [l4_error\(\)](#) shall be used to test if the map operation was successful.

Return values

| | |
|----------------------------------|--|
| <code>L4_EOK</code> | Operation successful (but see notes below). |
| <code>-L4_EPERM</code> | Insufficient permissions; see precondition. |
| <code>-L4_EINVAL</code> | Invalid source task capability. |
| <code>-L4_IPC_SEMAPFAILED</code> | The map operation failed due to limited quota. |

Precondition

The invoked [Task](#) capability must have the permission [L4_CAP_FPAGE_W](#).

This method allows for asynchronous transfer of capabilities, memory mappings, and IO-port mappings (on IA32) from one task to another. The destination task is the task referenced by the capability on which the map is invoked, and the receive window is the whole address space of that task. By specifying proper rights in the `snd_fpage` and `snd_base`, it is possible to remove rights during transfer.

Note

If the send flexpage is of type [L4_FPAGE_OBJ](#), the [L4_CAP_FPAGE_S](#) right is removed from the transferred capability unless both the source and destination task capabilities possess the [L4_CAP_FPAGE_S](#) right themselves.

Even with [l4_error\(\)](#) returning `L4_EOK` there might be cases where not all pages of the send flexpage were mapped respectively granted to the destination task, for instance, if the corresponding mapping in the destination task does already exist.

For more information on spaces and mappings, see [Spaces and Mappings](#). The flexpage API is described in more detail at [Flexpages](#).

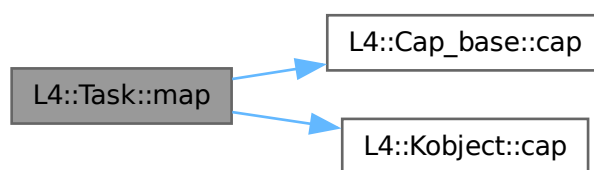
Note

For peculiarities when using grant, see [L4_MAP_ITEM_GRANT](#).

Definition at line 85 of file [task](#).

References [L4::Cap_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:

**15.198.2.6 release_cap()**

```

l4_msgtag_t L4::Task::release_cap (
    L4::Cap< void > cap,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
  
```

Release object capability.

Parameters

| | |
|-------------|--|
| <i>cap</i> | Capability selector of the object to release. |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb . |

Returns

Syscall return tag.

This operation unmaps the capability from `this` task. This operation is equivalent to unmapping a single object capability by specifying all object rights as unmap mask.

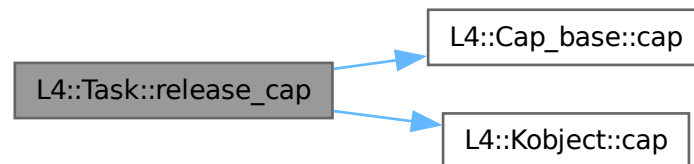
Note

If the reference counter of the kernel object referenced by [cap](#) goes down to zero, the deletion of the object is initiated. Objects are not destroyed until all other kernel objects holding a reference to it drop the reference.

Definition at line 201 of file [task](#).

References [L4::Cap_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:

**15.198.2.7 unmap()**

```

l4_msgtag_t L4::Task::unmap (
    l4_fpage_t const & fpage,
    l4_unword_t map_mask,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
  
```

Revoke rights from the task.

Parameters

| | |
|-----------------|---|
| <i>fpage</i> | Flexpage that describes an area in one capability space of <code>this</code> task and the rights to revoke. |
| <i>map_mask</i> | Unmap mask, see l4_unmap_flags_t |

| | |
|-------------|--|
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb . |
|-------------|--|

Returns

Syscall return tag

This method allows to revoke rights from the destination task. The rights to revoke are specified in the flexpage, see [l4_fpage_rights\(\)](#). For a flexpage describing IO ports or memory, it also revokes rights from all the tasks that got the rights delegated from the destination task (i.e., this operation does a recursive rights revocation). The capability is unmapped if certain rights are specified, see below for details. It is guaranteed that the rights revocation is completed before this function returns.

Note that this function cannot be used to revoke the reference counting permission (see [L4_FPAGE_C_REF_CNT](#)) or the IPC-gate server permission (see [L4_FPAGE_C_IPCGATE_SVR](#)) from object capabilities.

It depends on the platform and the object type which rights need to be specified in the `rights` field of `fpage` to unmap a capability:

- An object capability is unmapped if and only if the [L4_CAP_FPAGE_R](#) right bit is set.
- An IO port is unmapped if and only if any right bit is set.
- Memory is unmapped if and only if the [L4_FPAGE_RO](#) right bit is set.

Note

Depending on the page-table features supported by the hardware, revocation of certain rights from a memory capability can be a no-op (i.e., the rights are not revoked). Further, revocation of certain rights may grant other rights which were not present before. For instance, on an architecture without support for NX, revoking X does nothing. For another example, revoking only X from an execute-only page grants read permission (because the mapping remains present in the page table).

If the reference counter of a kernel object referenced in `fpage` goes down to zero (as a result of deleting capabilities), the deletion of the object is initiated. Objects are not destroyed until all other kernel objects holding a reference to it drop the reference.

Definition at line [135](#) of file [task](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



15.198.2.8 unmap_batch()

```
l4_msgtag_t L4::Task::unmap_batch (
    l4_fpage_t const * fpages,
    unsigned num_fpages,
    l4_umword_t map_mask,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
```

Revoke rights from a task.

Parameters

| | |
|-------------------|--|
| <i>fpages</i> | An array of flexpages. Each item describes an area in one capability space of <code>this</code> task. |
| <i>num_fpages</i> | Number of fpages in the <code>fpages</code> array. |
| <i>map_mask</i> | Unmap mask, see l4_unmap_flags_t . |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb . |

Revoke rights for an array of flexpages, see [unmap](#) for details.

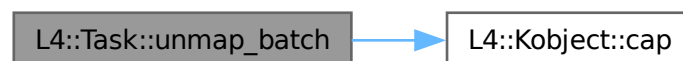
Precondition

The caller needs to take care that `num_fpages` is not bigger than `L4_UTCB_GENERIC_DATA_SIZE - 2`.

Definition at line 154 of file [task](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

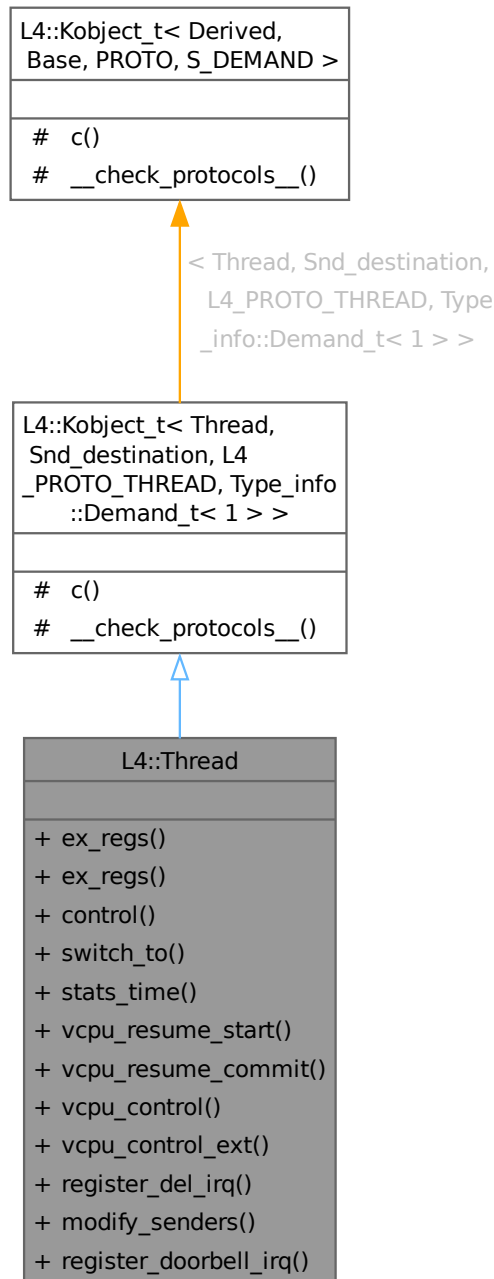
- [l4/sys/task](#)

15.199 L4::Thread Class Reference

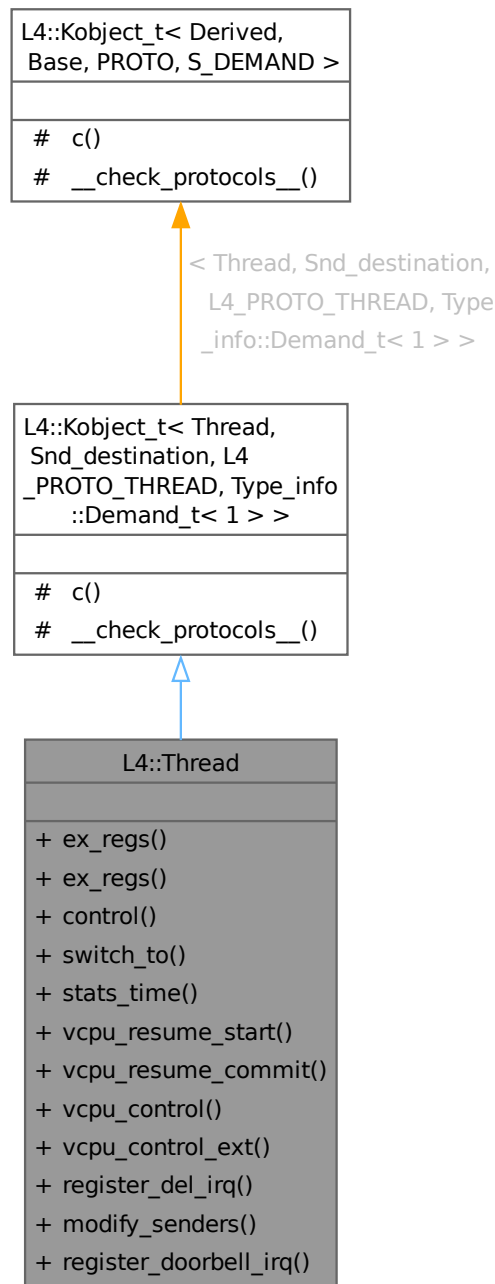
C++ [L4](#) kernel thread interface, see [Thread](#) for the C interface.

```
#include <thread>
```

Inheritance diagram for L4::Thread:



Collaboration diagram for L4::Thread:



Data Structures

- class [Attr](#)
Thread attributes used for `control()`.
- class [Modify_senders](#)
Class wrapping a list of rules which modify the sender label of IPC messages inbound to this thread.

Public Member Functions

- [l4_msgtag_t ex_regs](#) ([l4_addr_t](#) ip, [l4_addr_t](#) sp, [l4_umword_t](#) flags, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) noexcept
Exchange basic thread registers.
- [l4_msgtag_t ex_regs](#) ([l4_addr_t](#) *ip, [l4_addr_t](#) *sp, [l4_umword_t](#) *flags, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) noexcept
Exchange basic thread registers and return previous values.
- [l4_msgtag_t control](#) ([Attr](#) const &attr) noexcept
Commit the given thread-attributes object.
- [l4_msgtag_t switch_to](#) ([l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) noexcept
Switch execution to this thread.
- [l4_msgtag_t stats_time](#) ([l4_kernel_clock_t](#) *us, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) noexcept
Get consumed time of thread in us.
- [l4_msgtag_t vcpu_resume_start](#) ([l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) noexcept
Resume from vCPU asynchronous IPC handler, start.
- [l4_msgtag_t vcpu_resume_commit](#) ([l4_msgtag_t](#) tag, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) noexcept
Resume from vCPU asynchronous IPC handler, commit.
- [l4_msgtag_t vcpu_control](#) ([l4_addr_t](#) vcpu_state, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) noexcept
Enable the vCPU feature for the thread.
- [l4_msgtag_t vcpu_control_ext](#) ([l4_addr_t](#) ext_vcpu_state, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) noexcept
Enable the extended vCPU feature for the thread.
- [l4_msgtag_t register_del_irq](#) ([Cap](#) < [Irq](#) > irq, [l4_utcb_t](#) *u=[l4_utcb\(\)](#)) noexcept
Register an IRQ that will trigger upon deletion events.
- [l4_msgtag_t modify_senders](#) ([Modify_senders](#) const &todo) noexcept
Apply sender modification rules.
- [l4_msgtag_t register_doorbell_irq](#) ([Cap](#) < [Irq](#) > irq, [l4_utcb_t](#) *u=[l4_utcb\(\)](#)) noexcept
Register an IRQ that will trigger when a forwarded virtual interrupt is pending.

Additional Inherited Members

Protected Types inherited from

[L4::Kobject_t](#) < [Thread](#), [Snd_destination](#), [L4_PROTO_THREAD](#), [Type_info::Demand_t](#) < 1 > >

- typedef [Thread](#) **Class**
The target interface type (inheriting from [Kobject_t](#)).
- typedef [Typeid::Iface](#) < [PROTO](#), [Thread](#) > **__lfac**
The interface description for the derived class.
- typedef [Typeid::Merge_list](#) < [Typeid::Iface_list](#) < **__lfac** >, typename [Snd_destination::__lfac_list](#) > **__lfac_list**
The list of all RPC interfaces provided directly or through inheritance.

Protected Member Functions inherited from

[L4::Kobject_t](#) < [Thread](#), [Snd_destination](#), [L4_PROTO_THREAD](#), [Type_info::Demand_t](#) < 1 > >

- [L4::Cap](#) < [Class](#) > **c** () const noexcept
Get the capability to ourselves.

Static Protected Member Functions inherited from

[L4::Kobject_t](#) < [Thread](#), [Snd_destination](#), [L4_PROTO_THREAD](#), [Type_info::Demand_t](#) < 1 > >

- static void **__check_protocols** () noexcept
Helper to check for protocol conflicts.

15.199.1 Detailed Description

C++ [L4](#) kernel thread interface, see [Thread](#) for the C interface.

The [Thread](#) class defines a thread of execution in the [L4](#) context. Usually user-level and kernel threads are mapped 1:1 to each other. [Thread](#) kernel objects are created using a factory, see the [L4::Factory](#) API ([L4::Factory::create\(\)](#)).

Amongst other things an [L4::Thread](#) encapsulates:

- CPU state
 - General-purpose registers
 - Program counter
 - Stack pointer
- FPU state
- Scheduling parameters, see the [L4::Scheduler](#) API
- Execution state
 - Blocked, Runnable, Running

[Thread](#) objects provide an API for

- [Thread](#) configuration and manipulation
- [Thread](#) switching.

On ARM newly created threads run in EL0 by default and the exception level can be changed there with [ex_regs\(\)](#).

Include File

```
#include <l4/sys/thread>
```

For the C interface see the [Thread](#) API. For more elaborated documentation on the vCPU feature see [vCPU API](#).

Definition at line 52 of file [thread](#).

15.199.2 Member Function Documentation

15.199.2.1 control()

```
l4_msgtag_t L4::Thread::control (
    Attr const & attr) [inline], [noexcept]
```

Commit the given thread-attributes object.

Parameters

| | |
|-------------|---|
| <i>attr</i> | the attribute object to commit to the thread. |
|-------------|---|

Returns

Syscall return tag containing one of the following return codes.

Return values

| | |
|-------------------------|---|
| <code>L4_EOK</code> | Operation successful. |
| <code>-L4_EPERM</code> | Insufficient permissions; see precondition. |
| <code>-L4_EINVAL</code> | Malformed thread-attributes. |

Precondition

The invoked [Thread](#) capability must have the permission [L4_CAP_FPAGE_S](#). When using [Attr::bind\(\)](#), also the respective [Task](#) capability must have the permission [L4_CAP_FPAGE_S](#).

Definition at line 243 of file [thread](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:

**15.199.2.2 ex_regs() [1/2]**

```

l4_msgtag_t L4::Thread::ex_regs (
    l4_addr_t * ip,
    l4_addr_t * sp,
    l4_umword_t * flags,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
  
```

Exchange basic thread registers and return previous values.

Parameters

| | | |
|---------|--------------|--|
| in, out | <i>ip</i> | New instruction pointer, use ~0UL to leave the instruction pointer unchanged, return previous instruction pointer. |
| in, out | <i>sp</i> | New stack pointer, use ~0UL to leave the stack pointer unchanged, returns previous stack pointer. |
| in, out | <i>flags</i> | Ex-regs flags, see L4_thread_ex_regs_flags , return previous CPU flags of the thread. |

| | | |
|--|-------------|--|
| | <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb . |
|--|-------------|--|

Returns

System call return tag. [out] parameters are only valid if the function returns successfully. Use [l4_error\(\)](#) to check.

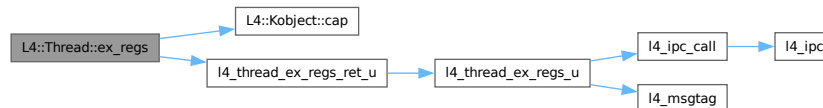
This method allows to manipulate and start a thread. The basic functionality is to set the instruction pointer and the stack pointer of a thread. Additionally, this method allows also to cancel ongoing IPC operations and to force the thread to raise an artificial exception (see [flags](#)). If the thread is in an IPC operation or if [L4_THREAD_EX_REGS_TRIGGER_EXCEPTION](#) forces an IPC then changes in IP and SP take effect directly after returning from this IPC. On ARM this method allows to change the exception level, see [L4_thread_ex_regs_flags_arm](#) and [L4_thread_ex_regs_flags_arm64](#).

The thread is started using [L4::Scheduler::run_thread\(\)](#). However, if at the time [L4::Scheduler::run_thread\(\)](#) is called, the instruction pointer of the thread is invalid, a later call to [ex_regs\(\)](#) with a valid instruction pointer might start the thread.

Definition at line 119 of file [thread](#).

References [L4::Kobject::cap\(\)](#), and [l4_thread_ex_regs_ret_u\(\)](#).

Here is the call graph for this function:



15.199.2.3 ex_regs() [2/2]

```

l4_msgtag_t L4::Thread::ex_regs (
    l4_addr_t ip,
    l4_addr_t sp,
    l4_umword_t flags,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]

```

Exchange basic thread registers.

Parameters

| | |
|--------------|--|
| <i>ip</i> | New instruction pointer, use ~0UL to leave the instruction pointer unchanged. |
| <i>sp</i> | New stack pointer, use ~0UL to leave the stack pointer unchanged. |
| <i>flags</i> | Ex-regs flags, see L4_thread_ex_regs_flags . |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb . |

Returns

System call return tag.

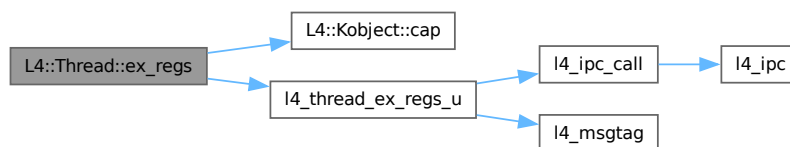
This method allows to manipulate and start a thread. The basic functionality is to set the instruction pointer and the stack pointer of a thread. Additionally, this method allows also to cancel ongoing IPC operations and to force the thread to raise an artificial exception (see `flags`). If the thread is in an IPC operation or if `L4_THREAD_EX_REGS_TRIGGER_EXCEPTION` forces an IPC then changes in IP and SP take effect directly after returning from this IPC. On ARM this method allows to change the exception level, see `L4_thread_ex_regs_flags_arm` and `L4_thread_ex_regs_flags_arm64`.

The thread is started using `L4::Scheduler::run_thread()`. However, if at the time `L4::Scheduler::run_thread()` is called, the instruction pointer of the thread is invalid, a later call to `ex_regs()` with a valid instruction pointer might start the thread.

Definition at line 84 of file `thread`.

References `L4::Kobject::cap()`, and `l4_thread_ex_regs_u()`.

Here is the call graph for this function:

**15.199.2.4 modify_senders()**

```
l4_msgtag_t L4::Thread::modify_senders (
    Modify_senders const & todo) [inline], [noexcept]
```

Apply sender modification rules.

Parameters

| | |
|-------------|-------------------------------------|
| <i>todo</i> | Prepared sender modification rules. |
|-------------|-------------------------------------|

Returns

System call return tag.

The modification rules are applied to all IPCs to the thread (whether directly or by IPC gate) that are already in flight, that is that the sender is already blocking on.

See `Modify_senders` for a detailed description when applying sender modification rules is required.

Note

Modifying the senders of a thread running on a different CPU core is not supported.

To ensure that no in-flight senders are missed, either the thread itself must execute `modify_senders`, or the thread executing the `modify_senders` must synchronize with the target thread. This synchronization must ensure the following:

1. Before `modify_senders` is executed the target thread must execute at least shortly (so that pending DRQs are handled).
2. The target thread must pause its IPC dispatch, until `modify_senders` is completed. In other words, the target thread must not be receive ready, because otherwise an IPC message with an unmodified label can be transferred to its UTCB or vCPU state.

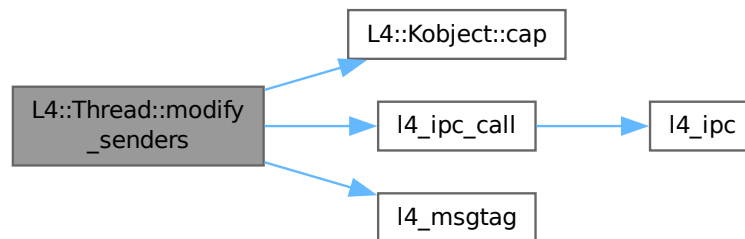
See also

[l4_thread_modify_sender_commit\(\)](#)

Definition at line 525 of file [thread](#).

References [L4::Kobject::cap\(\)](#), [l4_ipc_call\(\)](#), [L4_IPC_NEVER](#), [l4_msgtag\(\)](#), and [L4_PROTO_THREAD](#).

Here is the call graph for this function:

**15.199.2.5 register_del_irq()**

```

l4_msgtag_t L4::Thread::register_del_irq (
    Cap< Irq > irq,
    l4_utcb_t * u = l4_utcb()) [inline], [noexcept]
  
```

Register an IRQ that will trigger upon deletion events.

Parameters

| | |
|------------|--|
| <i>irq</i> | Capability selector for the IRQ object to be triggered. |
| <i>u</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb . |

Returns

System call return tag containing the return code.

Return values

| | |
|------------------------|---|
| <code>-L4_BUSY</code> | A deletion IRQ is already bound to this thread. |
| <code>-L4_EPERM</code> | Insufficient permissions; see precondition. |

Precondition

The capability `irq` must have the permission `L4_CAP_FPAGE_W`.

In case the `irq` is already bound to an interrupt source, it is unbound first. When `irq` is deleted, it will be deregistered first. A registered deletion `irq` can only be deregistered by deleting the `irq` or the thread.

List of deletion events:

- deletion of one or several IPC gates bound to this thread.

When the deletion event is delivered, there is no indication about which IPC gate was deleted.

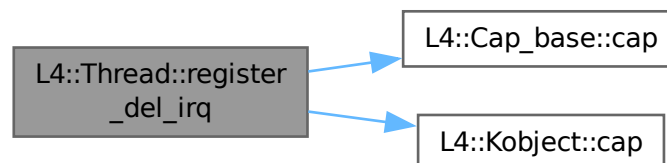
See also

[l4_thread_register_del_irq](#)

Definition at line 427 of file [thread](#).

References [L4::Cap_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



15.199.2.6 register_doorbell_irq()

```
l4_msgtag_t L4::Thread::register_doorbell_irq (
    Cap< Irq > irq,
    l4_utcb_t * u = l4_utcb()) [inline], [noexcept]
```

Register an IRQ that will trigger when a forwarded virtual interrupt is pending.

Parameters

| | |
|------------|--|
| <i>irq</i> | Capability selector for the IRQ object to be triggered. |
| <i>u</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb . |

Returns

System call return tag containing the return code.

Return values

| | |
|-----------|---|
| -L4_BUSY | A doorbell IRQ is already bound to this thread. |
| -L4_EPERM | Insufficient permissions; see precondition. |

Precondition

The capability `irq` must have the permission [L4_CAP_FPAGE_W](#).

See [Irq::bind_vcpu\(\)](#) for more details about how interrupts can be forwarded directly by the kernel to extended vCPU user mode.

In case the `irq` is already bound to an interrupt source, it is unbound first. When `irq` is deleted, it will be deregistered first. A registered doorbell [irq](#) can only be deregistered by deleting the [irq](#) or the thread.

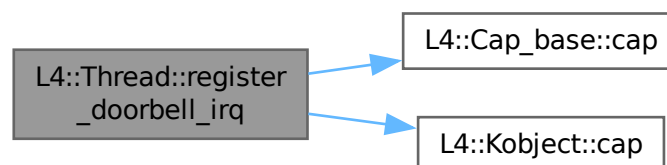
See also

[l4_thread_register_doorbell_irq](#)

Definition at line 553 of file [thread](#).

References [L4::Cap_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



15.199.2.7 stats_time()

```
l4_msgtag_t L4::Thread::stats_time (
    l4_kernel_clock_t * us,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
```

Get consumed time of thread in us.

Parameters

| | | |
|-----|-------------|--|
| out | <i>us</i> | Consumed time in μ s. |
| | <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb . |

Returns

Syscall return tag.

Definition at line [264](#) of file [thread](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



15.199.2.8 switch_to()

```
l4_msgtag_t L4::Thread::switch_to (
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
```

Switch execution to this thread.

Parameters

| | |
|-------------|--|
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb . |
|-------------|--|

Note

The current time slice is inherited to this thread.

Definition at line 253 of file [thread](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:

**15.199.2.9 vcpu_control()**

```

l4_msgtag_t L4::Thread::vcpu_control (
    l4_addr_t vcpu_state,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
  
```

Enable the vCPU feature for the thread.

Parameters

| | |
|-------------------|--|
| <i>vcpu_state</i> | A virtual address pointing to a l4_vcpu_state_t . It must be a valid kernel-user-memory address (see L4::Task::add_ku_mem()). |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb . |

Returns

Syscall return tag.

This function enables the vCPU feature of `this` thread

The kernel-user memory starting at `vcpu_state` must be at least 128-byte aligned and must cover the size of [l4_vcpu_state_t](#).

The asynchronous IPC handling is described at [vCPU API](#).

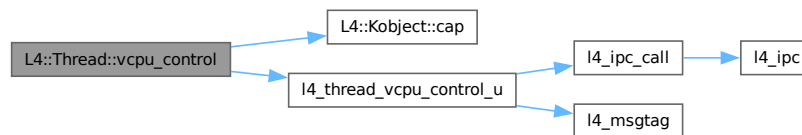
Note

Disabling of the vCPU feature is optional and currently not supported.

Definition at line 358 of file [thread](#).

References [L4::Kobject::cap\(\)](#), and [l4_thread_vcpu_control_u\(\)](#).

Here is the call graph for this function:

**15.199.2.10 vcpu_control_ext()**

```

l4_msgtag_t L4::Thread::vcpu_control_ext (
    l4_addr_t ext_vcpu_state,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
  
```

Enable the extended vCPU feature for the thread.

Parameters

| | |
|-----------------------|--|
| <i>ext_vcpu_state</i> | The virtual address where the kernel shall store the vCPU state in case of vCPU exits. The address must be a valid kernel-user-memory address (see L4::Task::add_ku_mem()). |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb . |

Returns

Syscall return tag.

The extended vCPU feature allows the use of hardware-virtualization features such as Intel's VT-x (VMX) or AMD's AMD-V (SVM).

This function enables the extended vCPU feature of `this` thread. Enabling the extended vCPU feature also enables the vCPU feature.

The kernel-user memory area starting at `ext_vcpu_state` must be at least 4 KiB aligned and must cover a size of `L4_PAGESIZE`. It includes the data of [l4_vcpu_state_t](#) at offset 0, the extended vCPU state at offset `L4_VCPU_OFFSET_EXT_STATE`, and, on some platforms, the extended vCPU information at offset `L4_VCPU_OFFSET_EXT_INFOS`.

On Intel's VT-x (VMX), the extended vCPU state is [l4_vm_vmx_vcpu_vmcs_t](#) and the extended vCPU information is [l4_vm_vmx_vcpu_infos_t](#). Furthermore, the extended vCPU state needs to be associated with a vCPU context (see [l4_vm_vmx_set_hw_vmcs\(\)](#)).

On AMD's AMD-V (SVM), the extended vCPU state is [l4_vm_svm_vmcb_t](#).

Note

Enabling the extended vCPU feature for a thread running on a different CPU core is currently not supported.

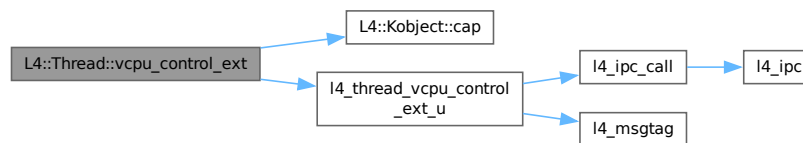
Disabling of the extended vCPU feature is currently not supported.

Upgrading from non-extended vCPU feature to extended vCPU feature is currently not supported.

Definition at line 398 of file [thread](#).

References [L4::Kobject::cap\(\)](#), and [l4_thread_vcpu_control_ext_u\(\)](#).

Here is the call graph for this function:

**15.199.2.11 vcpu_resume_commit()**

```

l4_msgtag_t L4::Thread::vcpu_resume_commit (
    l4_msgtag_t tag,
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
  
```

Resume from vCPU asynchronous IPC handler, commit.

Parameters

| | |
|-------------|--|
| <i>tag</i> | Tag to use, returned by l4_thread_vcpu_resume_start() . |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb . |

Returns

Syscall return tag containing one of the following return codes.

Return values

| | |
|------------|--|
| 0 | Indicates a VM exit, provided that <code>thread</code> is in extended vCPU mode with virtual interrupts cleared. |
| 1 | Indicates an incoming IPC message, provided that the <code>thread</code> is in extended vCPU mode with virtual interrupts cleared. |
| -L4_EPERM | The user task capability set in the vCPU state is missing the L4_CAP_FPAGE_S right. On Intel's VT-x (VMX): The vCPU context capability set in the extended vCPU state is missing the L4_CAP_FPAGE_S right. |
| -L4_ENOENT | The user task capability set in the vCPU state is invalid. |

| | |
|-------------------------|---|
| <code>-L4_EINVAL</code> | <code>thread</code> is not the current running thread, or does not have the vCPU feature enabled. On Intel's VT-x (VMX): No vCPU context associated with the extended vCPU state. |
| <code>-L4_EBUSY</code> | On Intel's VT-x (VMX): The vCPU context associated with the extended vCPU state is already active on a different CPU. |
| <code>-L4_ENODEV</code> | On Intel's VT-x (VMX): The vCPU context associated with the extended vCPU state cannot be initialized or activated. |
| <code><0</code> | A supplied mapping failed. |

All flexpages in the UTCB (added with [l4_sndfpage_add\(\)](#) after [l4_thread_vcpu_resume_start\(\)](#)) are unconditionally mapped into the user task configured in the vCPU state.

To resume into another address space, the capability to the target [Task](#) (or [L4::Vm](#)) must be set in [l4_vcpu_state_t::user_task](#) together with [L4_VCPU_F_USER_MODE](#). The capability selector must have all lower bits clear (see [L4_CAP_MASK](#)). The kernel adds the [L4_SYSF_SEND](#) flag there to indicate that the capability has been referenced in the kernel. Consecutive resumes will not reference the task capability again until all lower bits are cleared again. To release a task use a different task capability or use an invalid capability with the [L4_SYSF_REPLY](#) flag set.

The asynchronous IPC handling is described at [vCPU API](#).

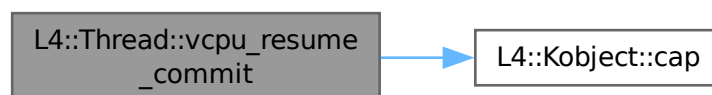
See also

[l4_thread_vcpu_resume_commit](#)

Definition at line 334 of file [thread](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



15.199.2.12 vcpu_resume_start()

```
l4_msgtag_t L4::Thread::vcpu_resume_start (
    l4_utcb_t * utcb = l4_utcb()) [inline], [noexcept]
```

Resume from vCPU asynchronous IPC handler, start.

Parameters

| | |
|-------------|--|
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb . |
|-------------|--|

Returns

Message tag to be used for [l4_sndfpage_add\(\)](#) and [l4_thread_vcpu_resume_commit\(\)](#)

The vCPU resume functionality is split in multiple functions to allow the specification of additional send-flexpages using [l4_sndfpage_add\(\)](#).

The asynchronous IPC handling is described at [vCPU API](#).

See also

[l4_thread_vcpu_resume_start](#)

Definition at line 283 of file [thread](#).

The documentation for this class was generated from the following file:

- [l4/sys/thread](#)

15.200 L4::Thread::Attr Class Reference

[Thread](#) attributes used for [control\(\)](#).

```
#include <thread>
```

Collaboration diagram for L4::Thread::Attr:

| L4::Thread::Attr |
|---|
| <ul style="list-style-type: none"> + Attr() + pager() + pager() + exc_handler() + exc_handler() + bind() + alien() |

Public Member Functions

- [Attr](#) ([l4_utcb_t](#) *utcb=[l4_utcb](#)()) noexcept
Create a thread-attribute object with the given UTCB.
- void [pager](#) ([Cap](#)< void > const &pager) noexcept
Set the pager capability selector.
- [Cap](#)< void > [pager](#) () noexcept
Get the capability selector used for page-fault messages.
- void [exc_handler](#) ([Cap](#)< void > const &exc_handler) noexcept
Set the exception-handler capability selector.
- [Cap](#)< void > [exc_handler](#) () noexcept
Get the capability selector used for exception messages.
- void [bind](#) ([l4_utcb_t](#) *thread_utcb, [Cap](#)< [Task](#) > const &task) noexcept
Bind the thread to a task.
- void [alien](#) (int on) noexcept
Enable alien mode.

Friends

- class [L4::Thread](#)

15.200.1 Detailed Description

[Thread](#) attributes used for [control\(\)](#).

This class is responsible for initializing various attributes of a thread in a UTCB for the [control\(\)](#) method.

Note

Instantiation of this class starts the preparation of the UTCB. Do not invoke any non-Attr functions between the instantiation and the call to [L4::Thread::control\(\)](#).

See also

[Thread control](#) for some more details.

Definition at line 137 of file [thread](#).

15.200.2 Constructor & Destructor Documentation

15.200.2.1 Attr()

```
L4::Thread::Attr::Attr (
    l4_utcb_t * utcb = l4_utcb()) [inline], [explicit], [noexcept]
```

Create a thread-attribute object with the given UTCB.

Parameters

| | |
|-------------|---|
| <i>utcb</i> | The UTCB to use for the later L4::Thread::control() function. Usually this is the UTCB of the calling thread. See l4_utcb() . |
|-------------|---|

Definition at line 151 of file [thread](#).

15.200.3 Member Function Documentation

15.200.3.1 alien()

```
void L4::Thread::Attr::alien (
    int on) [inline], [noexcept]
```

Enable alien mode.

Parameters

| | |
|-----------|--|
| <i>on</i> | Boolean value defining the state of the feature. |
|-----------|--|

For a thread in alien mode the kernel produces just an exception IPC for each IPC and exception caused by the alien thread instead of handling these events regularly. (Page faults of alien threads and interrupts occurring while the alien thread is running are always handled regularly.) While the alien thread is blocking, the exception handler can inspect and modify the state of the alien thread and potentially also the system call arguments. If the exception handler replies with [L4_PROTO_ALLOW_SYSCALL](#) as message tag, the kernel handles the next IPC or exception of the alien thread in a regular way. If the exception handler leaves certain thread state unchanged (in particular the instruction pointer), this will be the IPC or exception that caused the call of the exception handler. For a regularly processed IPC or exception of the alien thread the kernel also performs an exception IPC on kernel exit.

This feature can be used to attach a debugger to a thread and trace all object invocations and their results. It could also be used to handle other systems that use the same syscall instruction as [L4Re](#).

Definition at line 224 of file [thread](#).

15.200.3.2 bind()

```
void L4::Thread::Attr::bind (
    l4_utcb_t * thread_utcb,
    Cap< Task > const & task) [inline], [noexcept]
```

Bind the thread to a task.

Parameters

| | |
|--------------------|---|
| <i>thread_utcb</i> | The thread's UTCB address within the task it shall be bound to. The address must be aligned (architecture dependent; at least word aligned) and it must point to at least L4_UTCB_OFFSET bytes of kernel-user memory. |
| <i>task</i> | The task the thread shall be bound to. |

Precondition

The thread must not be bound to a task yet.

The capability `task` must have the permission [L4_CAP_FPAGE_S](#), otherwise the later call to [L4::Thread::control\(\)](#) with this [Attr](#) object will fail with [L4_EPERM](#).

A thread may execute code in the context of a task if and only if the thread is bound to the task. To actually start execution, [L4::Thread::ex_regs\(\)](#) needs to be used. Execution in the context of the task means that the code has access to all the task's resources (and only those). The executed code itself must be one of those resources. A thread can be bound at most once to a task.

Note

The UTCBs of different threads in the same task should not overlap in order to prevent data corruption.

Definition at line [218](#) of file [thread](#).

15.200.3.3 exc_handler() [1/2]

```
Cap< void > L4::Thread::Attr::exc_handler () [inline], [noexcept]
```

Get the capability selector used for exception messages.

Returns

The capability selector used to send exception messages. The selector is valid in the task the thread is bound to.

Definition at line [189](#) of file [thread](#).

15.200.3.4 exc_handler() [2/2]

```
void L4::Thread::Attr::exc_handler (
    Cap< void > const & exc_handler) [inline], [noexcept]
```

Set the exception-handler capability selector.

Parameters

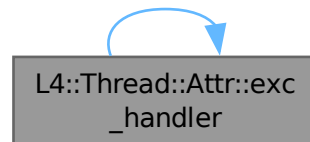
| | |
|--------------------|---|
| <i>exc_handler</i> | The capability selector that shall be used for exception messages. This capability selector must be valid within the task the thread is bound to. |
|--------------------|---|

Definition at line 180 of file [thread](#).

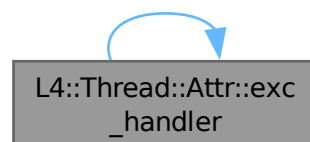
References [exc_handler\(\)](#).

Referenced by [exc_handler\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**15.200.3.5 pager() [1/2]**

```
Cap< void > L4::Thread::Attr::pager () [inline], [noexcept]
```

Get the capability selector used for page-fault messages.

Returns

The capability selector used to send page-fault messages. The selector is valid in the task the thread is bound to.

Definition at line 170 of file [thread](#).

15.200.3.6 pager() [2/2]

```
void L4::Thread::Attr::pager (
    Cap< void > const & pager) [inline], [noexcept]
```

Set the pager capability selector.

Parameters

| | |
|--------------|--|
| <i>pager</i> | The capability selector that shall be used for page-fault messages. This capability selector must be valid within the task the thread is bound to. |
|--------------|--|

Definition at line 161 of file [thread](#).

References [pager\(\)](#).

Referenced by [pager\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

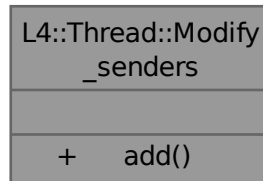
- [l4/sys/thread](#)

15.201 L4::Thread::Modify_senders Class Reference

[Class](#) wrapping a list of rules which modify the sender label of IPC messages inbound to this thread.

```
#include <thread>
```

Collaboration diagram for L4::Thread::Modify_senders:



Public Member Functions

- `int add (l4_umword_t match_mask, l4_umword_t match, l4_umword_t del_bits, l4_umword_t add_bits) noexcept`
Add a rule.

15.201.1 Detailed Description

[Class](#) wrapping a list of rules which modify the sender label of IPC messages inbound to this thread.

Use the [add\(\)](#) function to add modification rules, and use [modify_senders\(\)](#) to commit. Do not use the UTCB in between as it is used by [add\(\)](#) and [modify_senders\(\)](#).

This mechanism shall be used to change the source object labels of every pending IPC of an IPC gate or an IRQ if the labels in such pending IPC become invalid for the receiving thread, potentially because:

- an IPC gate / IRQ was unbound from a thread, or
- an IPC gate / IRQ was removed, or
- the label of an IPC gate / IRQ bound to a thread was changed.

It is not required to perform the `modify_sender` mechanism after an IPC gate or an IRQ was bound to a thread for the first time.

Definition at line [448](#) of file [thread](#).

15.201.2 Member Function Documentation

15.201.2.1 add()

```
int L4::Thread::Modify_senders::add (
    14_umword_t match_mask,
    14_umword_t match,
    14_umword_t del_bits,
    14_umword_t add_bits) [inline], [noexcept]
```

Add a rule.

Parameters

| | |
|-------------------|--|
| <i>match_mask</i> | Bitmask of bits to match the label. |
| <i>match</i> | Bitmask that must be equal to the label after applying match_mask. |
| <i>del_bits</i> | Bits to be deleted from the label. |
| <i>add_bits</i> | Bits to be added to the label. |

Returns

0 on success, <0 on error

In pseudo code: if ((sender_label & match_mask) == match) { sender_label = (sender_label & ~del_bits) | add_bits; }

Only the first match is applied.

See also

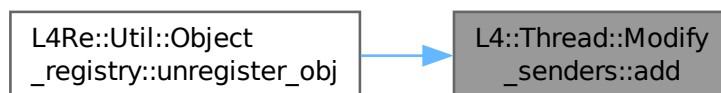
[l4_thread_modify_sender_add\(\)](#)

Definition at line 481 of file [thread](#).

References [L4_ENOMEM](#), and [l4_msg_regs_t::mr](#).

Referenced by [L4Re::Util::Object_registry::unregister_obj\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

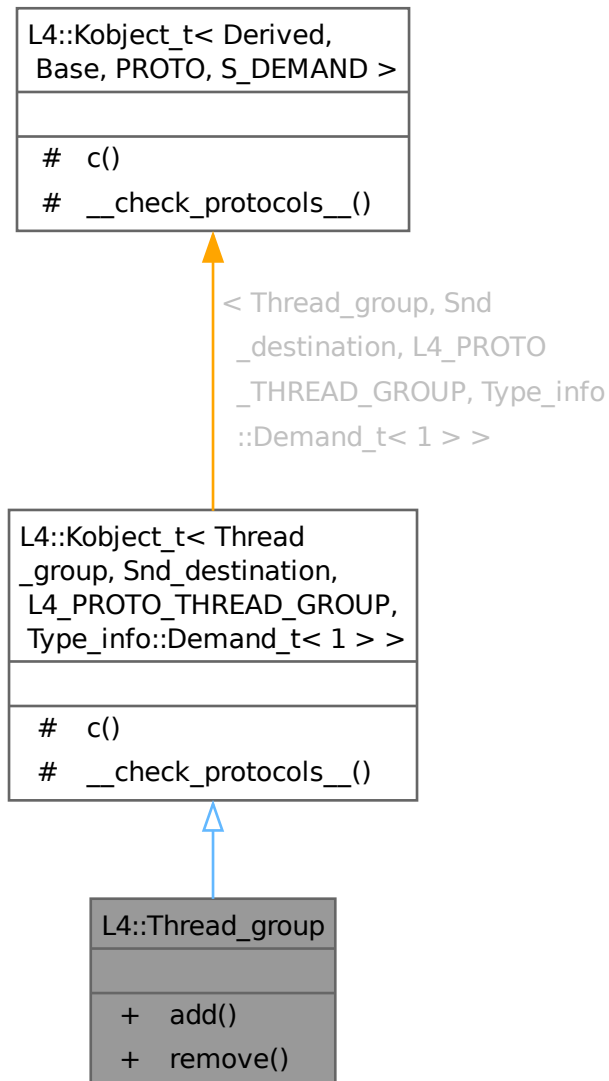
- [l4/sys/thread](#)

15.202 L4::Thread_group Class Reference

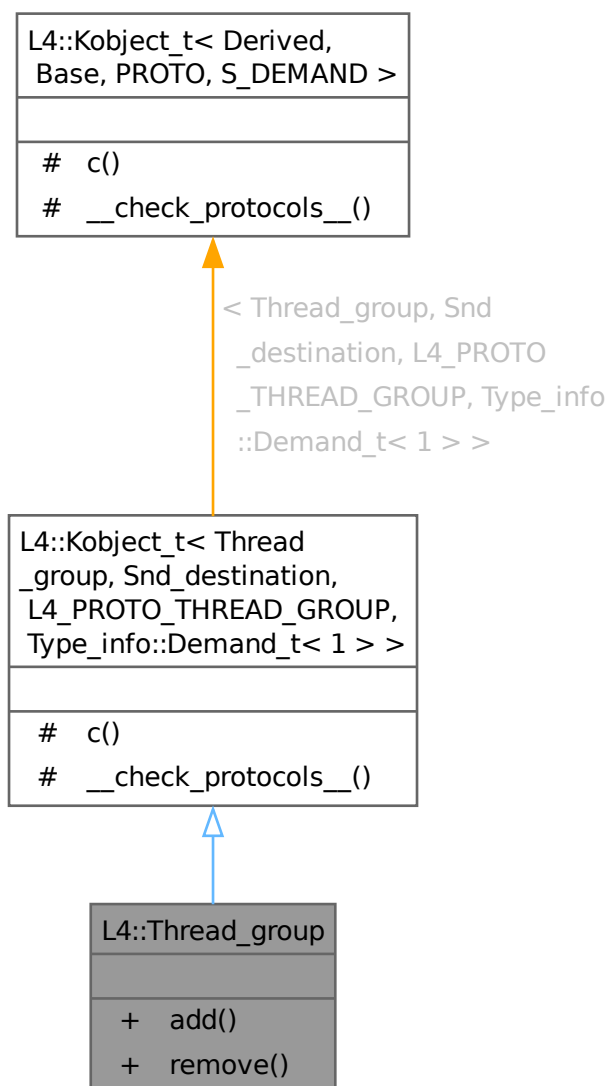
C++ [L4](#) kernel thread group interface, see [Thread groups](#) for the C interface.

```
#include <thread_group>
```

Inheritance diagram for L4::Thread_group:



Collaboration diagram for L4::Thread_group:



Public Member Functions

- `l4_msgtag_t add (Cap< Thread > thread, l4_utcb_t *utcb=l4_utcb()) noexcept`
Add thread to a thread group.
- `l4_msgtag_t remove (Cap< Thread > thread, l4_utcb_t *utcb=l4_utcb()) noexcept`
Remove thread from a thread group.

Additional Inherited Members

Protected Types inherited from

L4::Kobject_t< Thread_group, Snd_destination, L4_PROTO_THREAD_GROUP, Type_info::Demand_t<

- `typedef Thread_group Class`

The target interface type (inheriting from [Kobject_t](#)).

- typedef Typeid::Iface< PROTO, [Thread_group](#) > __Iface

The interface description for the derived class.

- typedef Typeid::Merge_list< Typeid::Iface_list< __Iface >, typename Snd_destination::__Iface_list > __↵
Iface_list

The list of all RPC interfaces provided directly or through inheritance.

Protected Member Functions inherited from

[L4::Kobject_t< Thread_group, Snd_destination, L4_PROTO_THREAD_GROUP, Type_info::Demand_t<](#)

- [L4::Cap< Class > c \(\)](#) const noexcept

Get the capability to ourselves.

Static Protected Member Functions inherited from

[L4::Kobject_t< Thread_group, Snd_destination, L4_PROTO_THREAD_GROUP, Type_info::Demand_t<](#)

- static void [__check_protocols__ \(\)](#) noexcept

Helper to check for protocol conflicts.

15.202.1 Detailed Description

C++ [L4](#) kernel thread group interface, see [Thread groups](#) for the C interface.

An [L4](#) thread group is a collection of threads used as indirection for IPC gate and IRQ objects such that these objects can have multiple receivers, from which the kernel selects one according to a policy.

The primary use case for thread groups are multi-threaded servers and CPU core local IRQ / IPC delivery.

A thread can be bound to at most one thread group. Before binding a thread to a thread group, the thread must be bound to a task. All threads bound to the same thread group must belong to the same task.

Definition at line 34 of file [thread_group](#).

15.202.2 Member Function Documentation

15.202.2.1 add()

```
l4\_msgtag\_t L4::Thread_group::add (
    Cap< Thread > thread,
    l4\_utcb\_t * utcb = l4\_utcb\(\)) [inline], [noexcept]
```

Add thread to a thread group.

Parameters

| | |
|---------------|--|
| <i>thread</i> | Thread to add to the thread group. |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb . |

Returns

Syscall return tag containing one of the following return codes.

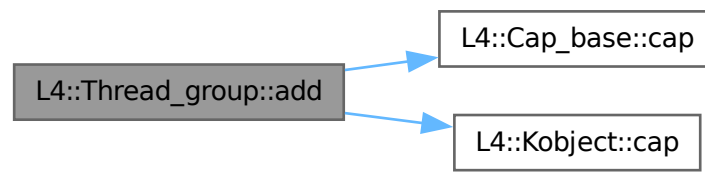
Return values

| | |
|-------------------|--|
| <i>L4_EOK</i> | No error occurred. |
| <i>-L4_EINVAL</i> | <i>thread</i> is not a thread object. |
| <i>-L4_EEXIST</i> | <i>thread</i> already bound to this thread group. |
| <i>-L4_EBUSY</i> | <i>thread</i> already bound to a different thread group. |
| <i>-L4_ENOENT</i> | Thread group doesn't exist. |

Definition at line 53 of file [thread_group](#).

References [L4::Cap_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:

**15.202.2.2 remove()**

```

14_msgtag_t L4::Thread_group::remove (
    Cap< Thread > thread,
    14_utcb_t * utcb = 14_utcb()) [inline], [noexcept]
  
```

Remove thread from a thread group.

Parameters

| | |
|---------------|--|
| <i>thread</i> | Thread to remove from the thread group. |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to 14_utcb . |

Returns

Syscall return tag containing one of the following return codes.

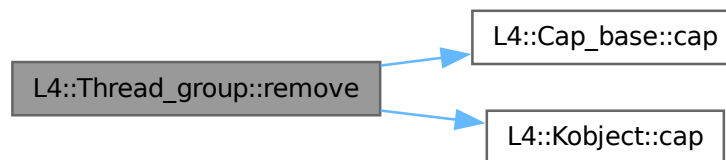
Return values

| | |
|-------------------------|---|
| <code>L4_EOK</code> | No error occurred. |
| <code>-L4_EINVAL</code> | <code>thread</code> is not a thread object. |

Definition at line 67 of file [thread_group](#).

References [L4::Cap_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

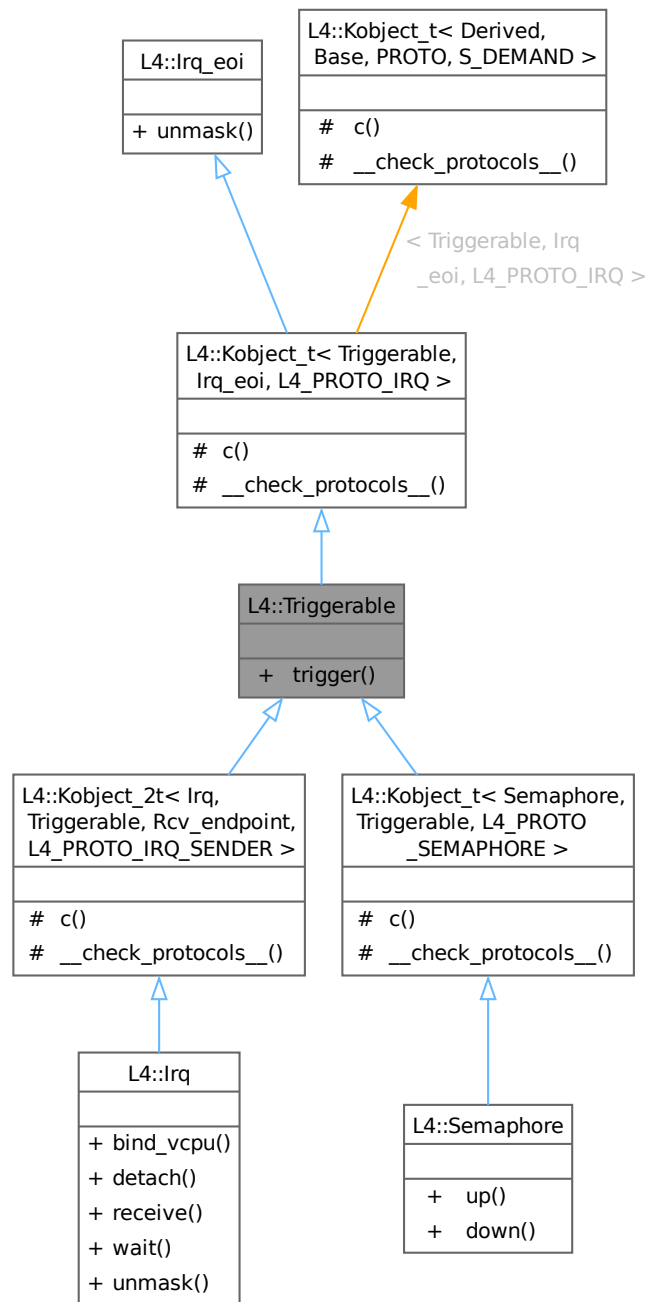
- [l4/sys/thread_group](#)

15.203 L4::Triggerable Struct Reference

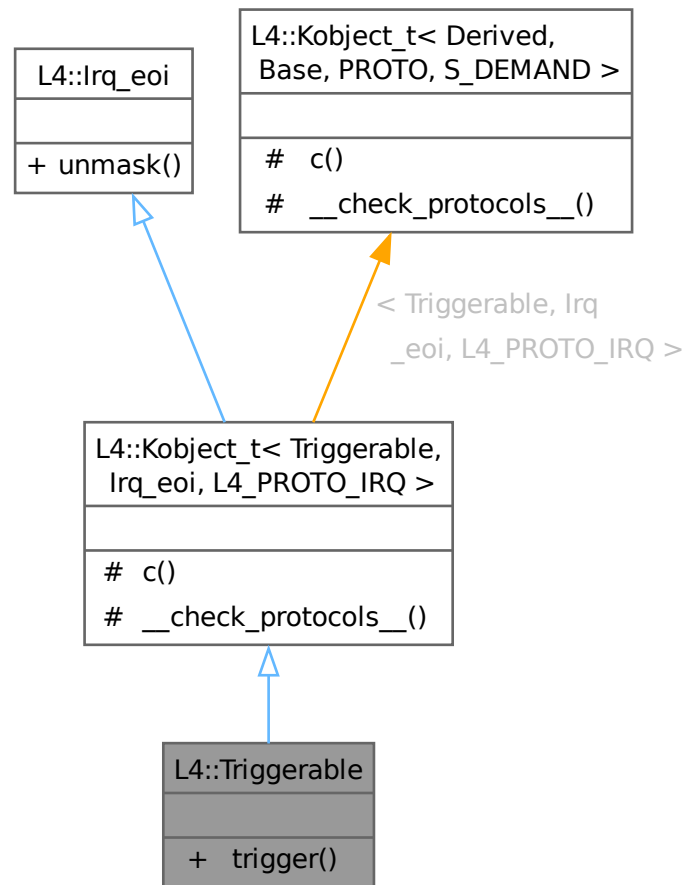
Interface that allows an object to be triggered by some source.

```
#include <irq>
```

Inheritance diagram for L4::Triggerable:



Collaboration diagram for L4::Triggerable:



Public Member Functions

- `l4_msgtag_t trigger (l4_utcb_t *utcb=l4_utcb()) noexcept`
Trigger the object.

Public Member Functions inherited from L4::Irq_eoi

- `l4_msgtag_t unmask (unsigned irqnum, l4_umword_t *label=0, l4_timeout_t to=L4_IPC_NEVER, l4_utcb_t *utcb=l4_utcb()) noexcept`
Unmask the given interrupt line.

Additional Inherited Members

Protected Types inherited from L4::Kobject_t< Triggerable, Irq_eoi, L4_PROTO_IRQ >

- typedef `Triggerable` Class

The target interface type (inheriting from [Kobject_t](#)).

- typedef Typeid::Iface< PROTO, [Triggerable](#) > [__Iface](#)

The interface description for the derived class.

- typedef Typeid::Merge_list< Typeid::Iface_list< [__Iface](#) >, typename Irq_eoi::__Iface_list > [__Iface_list](#)

The list of all RPC interfaces provided directly or through inheritance.

Protected Member Functions inherited from [L4::Kobject_t](#)< [Triggerable](#), [Irq_eoi](#), [L4_PROTO_IRQ](#) >

- [L4::Cap](#)< [Class](#) > [c](#) () const noexcept

Get the capability to ourselves.

Static Protected Member Functions inherited from [L4::Kobject_t](#)< [Triggerable](#), [Irq_eoi](#), [L4_PROTO_IRQ](#) >

- static void [__check_protocols__](#) () noexcept

Helper to check for protocol conflicts.

15.203.1 Detailed Description

Interface that allows an object to be triggered by some source.

The interface specifies no semantics for the trigger operation, this is defined by derived objects.

This interface is usually used in conjunction with [L4::lcu](#).

Definition at line 79 of file [irq](#).

15.203.2 Member Function Documentation

15.203.2.1 trigger()

```
l4\_msgtag\_t L4::Triggerable::trigger (
    l4\_utcb\_t * utcb = l4\_utcb() ) [inline], [noexcept]
```

Trigger the object.

Parameters

| | |
|-------------|--|
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |
|-------------|--|

Returns

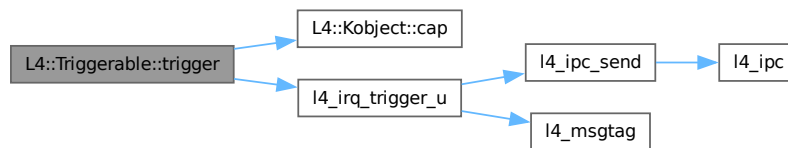
Syscall return tag for a send-only operation, this means there is no return value except `L4_MSGTAG_ERROR` indicating success or failure of the send operation. Use `l4_ipc_error()` to check for errors and **do not** use `l4_error()`.

Definition at line 91 of file `irq`.

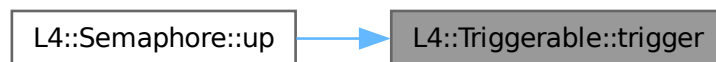
References `L4::Kobject::cap()`, and `l4_irq_trigger_u()`.

Referenced by `L4::Semaphore::up()`.

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this struct was generated from the following file:

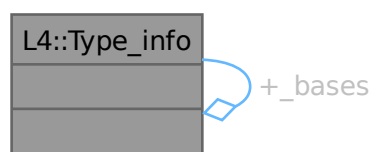
- `l4/sys/irq`

15.204 L4::Type_info Struct Reference

Dynamic Type Information for `L4Re` Interfaces.

```
#include <l4/sys/capability>
```

Collaboration diagram for `L4::Type_info`:



Data Structures

- class [Demand](#)
Data type for expressing the needed receive buffers at the server-side of an interface.
- struct [Demand_t](#)
Template type statically describing demand of receive buffers.
- struct [Demand_union_t](#)
Template type statically describing the combination of two [Demand](#) object.

15.204.1 Detailed Description

Dynamic Type Information for [L4Re](#) Interfaces.

This class represents the runtime-dynamic type information for [L4Re](#) interfaces, and is not intended to be used directly by applications.

Note

The interface of is subject to changes.

The main use for this info is to be used by the implementation of the [L4::cap_dynamic_cast\(\)](#) function.

Definition at line 499 of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

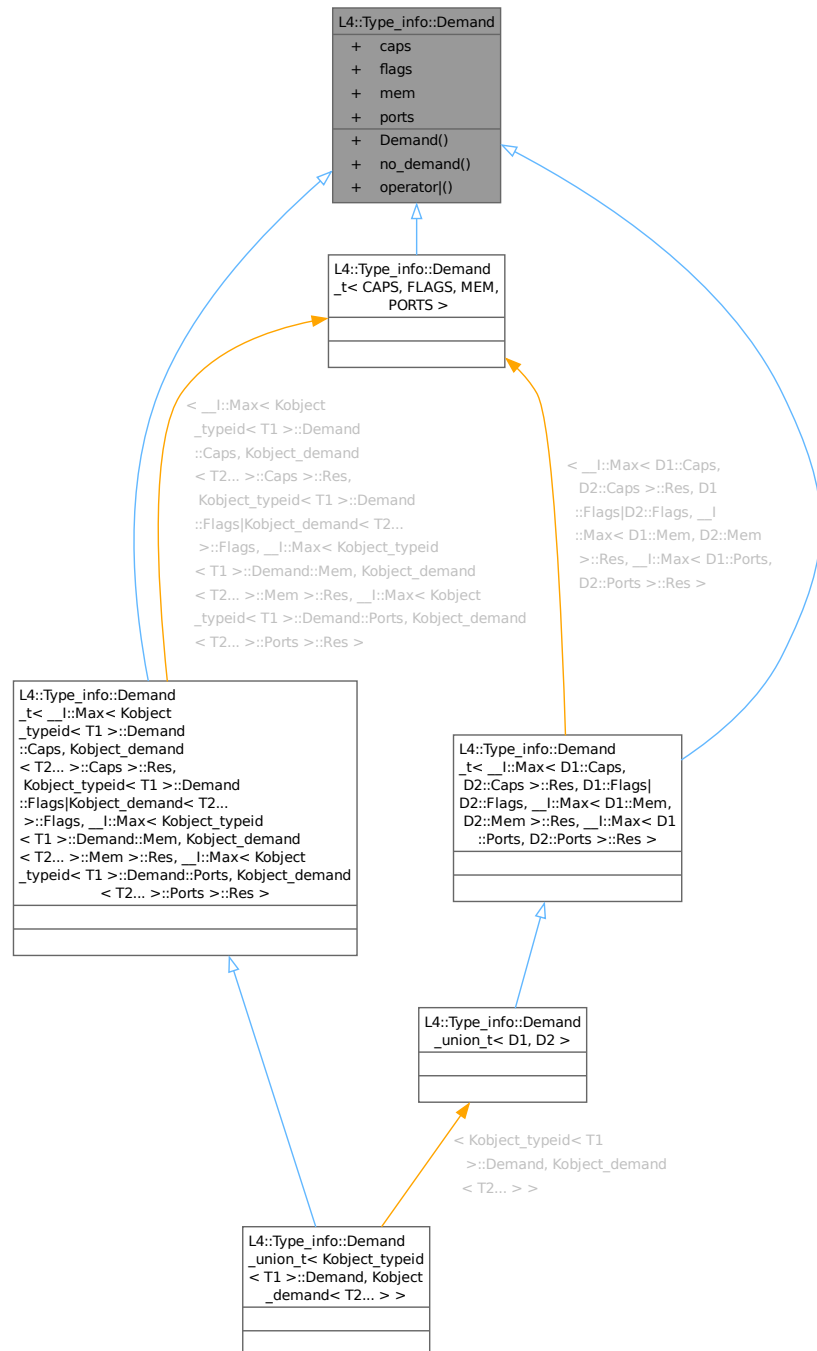
- [l4/sys/__typeinfo.h](#)

15.205 L4::Type_info::Demand Class Reference

Data type for expressing the needed receive buffers at the server-side of an interface.

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Type_info::Demand:



Collaboration diagram for L4::Type_info::Demand:

| L4::Type_info::Demand |
|-----------------------|
| + caps |
| + flags |
| + mem |
| + ports |
| + Demand() |
| + no_demand() |
| + operator () |

Public Member Functions

- [Demand](#) (unsigned char [caps](#)=0, unsigned char [flags](#)=0, unsigned char [mem](#)=0, unsigned char [ports](#)=0) noexcept
Make [Demand](#) object.
- bool [no_demand](#) () const noexcept
- [Demand operator|](#) ([Demand](#) const &rhs) const noexcept
get the combined demand of this and rhs

Data Fields

- unsigned char **caps**
number of capability receive buffers.
- unsigned char **flags**
flags, such as the need for timeouts (TBD).
- unsigned char **mem**
size (2^{mem} bytes) of memory receive buffer.
- unsigned char **ports**
number of IO-port receive buffers.

15.205.1 Detailed Description

Data type for expressing the needed receive buffers at the server-side of an interface.

Definition at line 506 of file [__typeinfo.h](#).

15.205.2 Constructor & Destructor Documentation

15.205.2.1 Demand()

```
L4::Type_info::Demand::Demand (
    unsigned char caps = 0,
    unsigned char flags = 0,
    unsigned char mem = 0,
    unsigned char ports = 0) [inline], [explicit], [noexcept]
```

Make [Demand](#) object.

Parameters

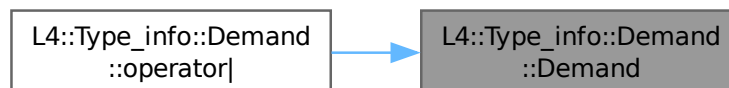
| | |
|--------------|--|
| <i>caps</i> | number of capability receive buffers |
| <i>flags</i> | flags, such as the need for timeouts (TBD). |
| <i>mem</i> | size (2^{mem} bytes) of memory receive window. |
| <i>ports</i> | number of IO-port receive windows. |

Definition at line [527](#) of file [__typeinfo.h](#).

References [caps](#), [flags](#), [mem](#), and [ports](#).

Referenced by [operator|\(\)](#).

Here is the caller graph for this function:



15.205.3 Member Function Documentation

15.205.3.1 no_demand()

```
bool L4::Type_info::Demand::no_demand () const [inline], [noexcept]
```

Returns

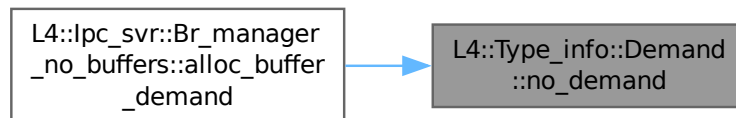
true if there is no demand at all

Definition at line 532 of file [__typeinfo.h](#).

References [caps](#), [flags](#), [mem](#), and [ports](#).

Referenced by [L4::lpc_svr::Br_manager_no_buffers::alloc_buffer_demand\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

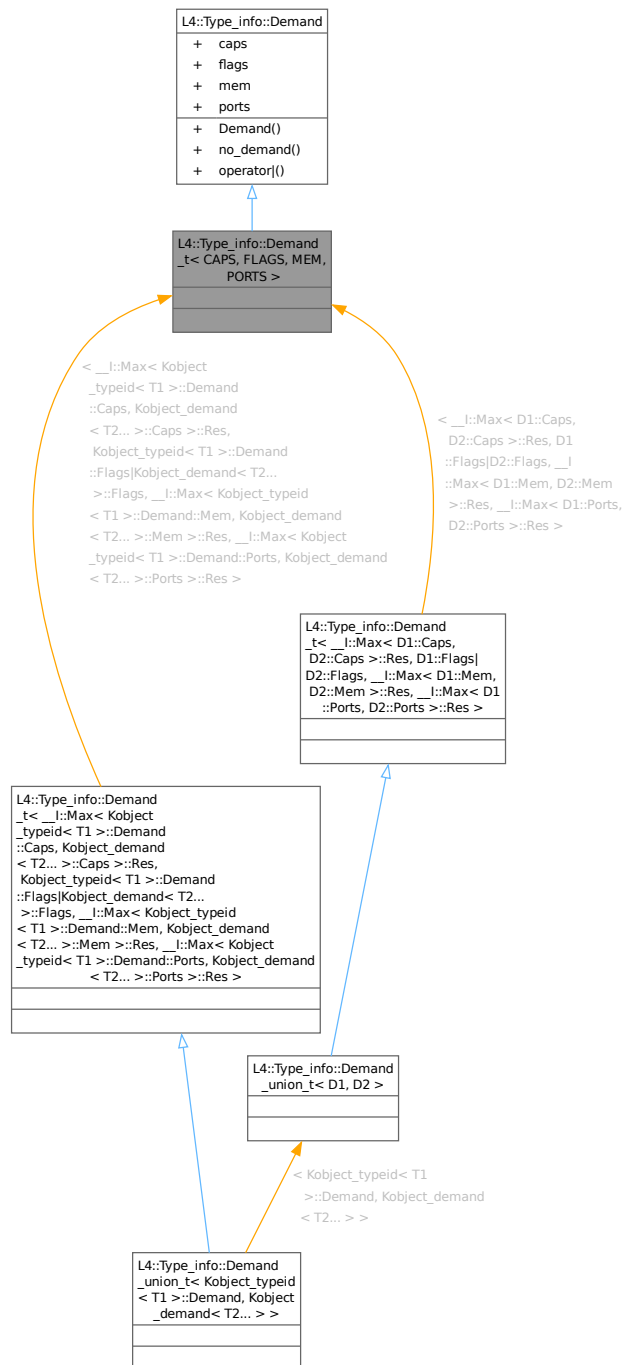
- [l4/sys/__typeinfo.h](#)

15.206 L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS > Struct Template Reference

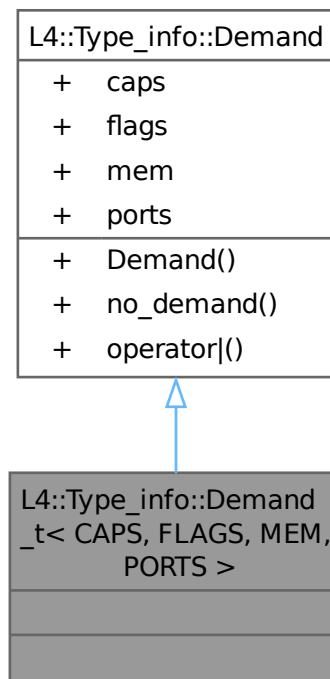
Template type statically describing demand of receive buffers.

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS >:



Collaboration diagram for L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS >:



Public Types

- enum { `Caps` = CAPS , `Flags` = FLAGS , `Mem` = MEM , `Ports` = PORTS }

Additional Inherited Members

Public Member Functions inherited from L4::Type_info::Demand

- `Demand` (unsigned char `caps`=0, unsigned char `flags`=0, unsigned char `mem`=0, unsigned char `ports`=0) noexcept
Make `Demand` object.
- bool `no_demand` () const noexcept
- `Demand operator|` (`Demand` const &rhs) const noexcept
get the combined demand of this and rhs

Data Fields inherited from L4::Type_info::Demand

- unsigned char `caps`
number of capability receive buffers.
- unsigned char `flags`
flags, such as the need for timeouts (TBD).
- unsigned char `mem`
size (2^{mem} bytes) of memory receive buffer.
- unsigned char `ports`
number of IO-port receive buffers.

15.206.1 Detailed Description

```
template<unsigned char CAPS = 0, unsigned char FLAGS = 0, unsigned char MEM = 0, unsigned char
PORTS = 0>
struct L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS >
```

Template type statically describing demand of receive buffers.

Template Parameters

| | |
|--------------|---|
| <i>CAPS</i> | number of capability receive buffers needed. |
| <i>FLAGS</i> | flags, such as the need for timeouts (TBD). |
| <i>MEM</i> | size (2^{MEM} bytes) of memory receive window needed. |
| <i>PORTS</i> | number of IO-port receive windows needed. |

Definition at line 553 of file [__typeinfo.h](#).

15.206.2 Member Enumeration Documentation

15.206.2.1 anonymous enum

```
template<unsigned char CAPS = 0, unsigned char FLAGS = 0, unsigned char MEM = 0, unsigned char
PORTS = 0>
anonymous enum
```

Enumerator

| | |
|-------|--|
| Caps | number of capability receive buffers. |
| Flags | flags, such as the need for timeouts. |
| Mem | size (2^{MEM} bytes) of memory receive window. |
| Ports | number of IO-port receive windows. |

Definition at line 555 of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

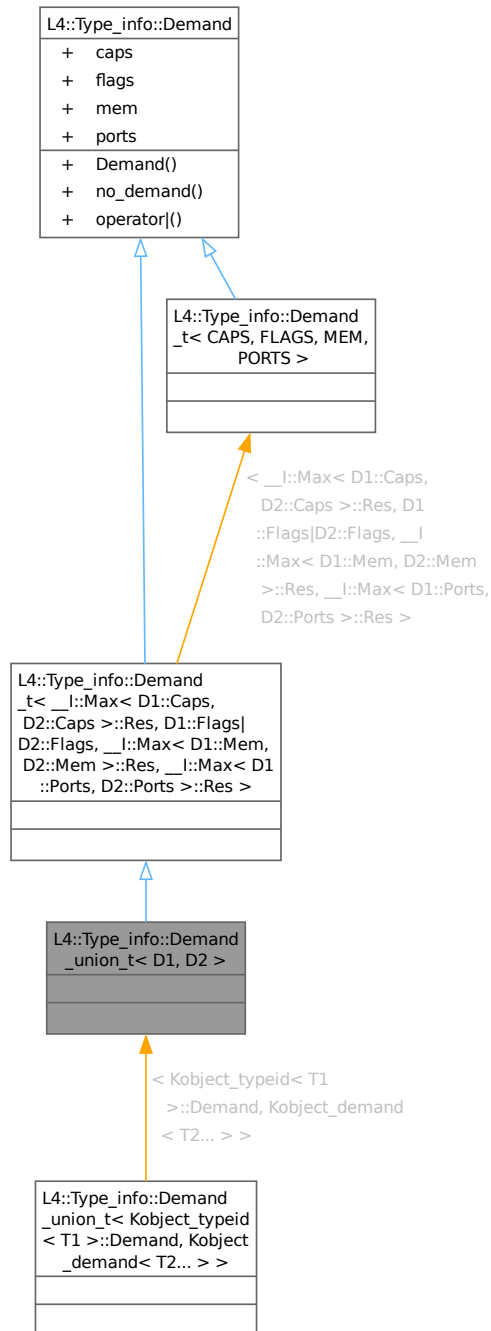
- [l4/sys/__typeinfo.h](#)

15.207 L4::Type_info::Demand_union_t< D1, D2 > Struct Template Reference

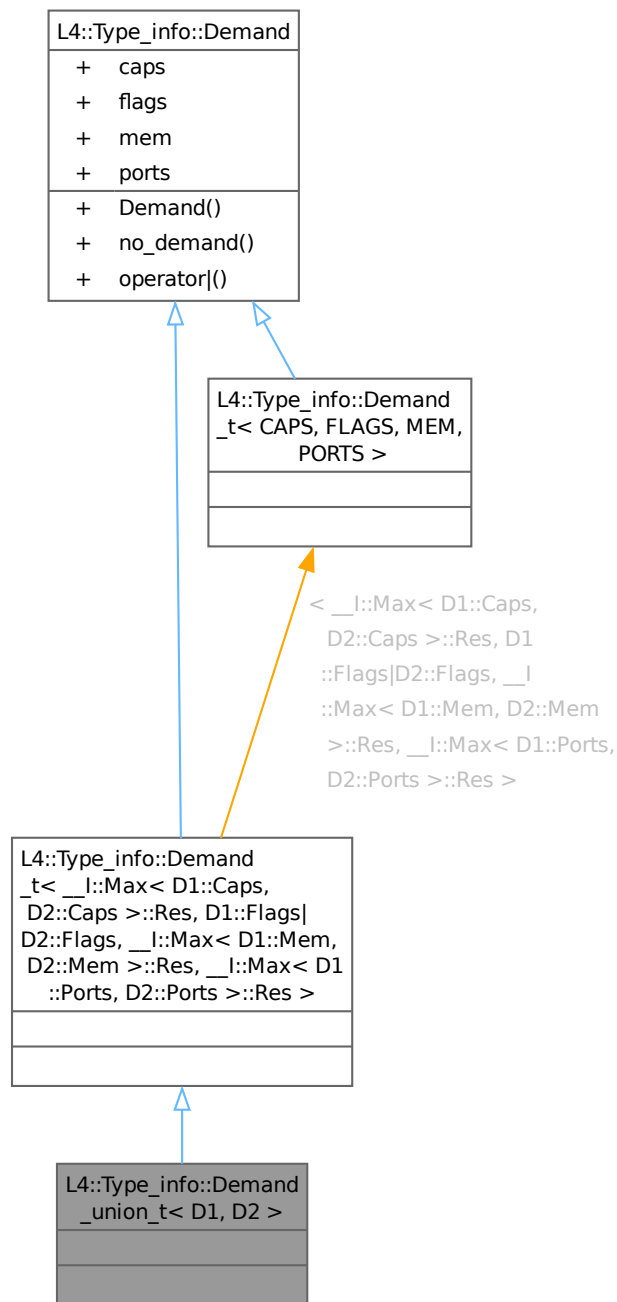
Template type statically describing the combination of two [Demand](#) object.

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Type_info::Demand_union_t< D1, D2 >:



Collaboration diagram for L4::Type_info::Demand_union_t< D1, D2 >:



Additional Inherited Members

Public Types inherited from

[L4::Type_info::Demand_t< __I::Max< D1::Caps, D2::Caps >::Res, D1::Flags|D2::Flags, __I::Max< D1::](#)

Public Member Functions inherited from [L4::Type_info::Demand](#)

- [Demand](#) (unsigned char [caps](#)=0, unsigned char [flags](#)=0, unsigned char [mem](#)=0, unsigned char [ports](#)=0) noexcept
Make [Demand](#) object.
- bool [no_demand](#) () const noexcept
- [Demand](#) **operator**| ([Demand](#) const &rhs) const noexcept
get the combined demand of this and rhs

Data Fields inherited from [L4::Type_info::Demand](#)

- unsigned char **caps**
number of capability receive buffers.
- unsigned char **flags**
flags, such as the need for timeouts (TBD).
- unsigned char **mem**
size (2^{\wedge} mem bytes) of memory receive buffer.
- unsigned char **ports**
number of IO-port receive buffers.

15.207.1 Detailed Description

```
template<typename D1, typename D2>
struct L4::Type_info::Demand_union_t< D1, D2 >
```

Template type statically describing the combination of two [Demand](#) object.

Template Parameters

| | |
|-----------|-----------------------|
| <i>D1</i> | first demand object. |
| <i>D2</i> | second demand object. |

Definition at line [573](#) of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

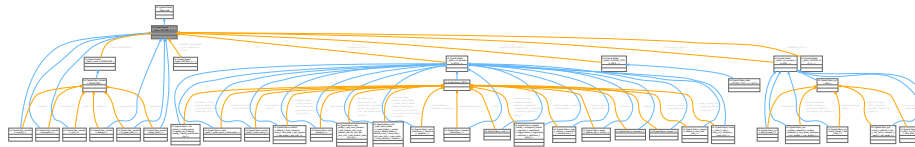
- [l4/sys/__typeinfo.h](#)

15.208 L4::Typeid::Detail::_Rpc< OPCODE, O, X > Struct Template Reference

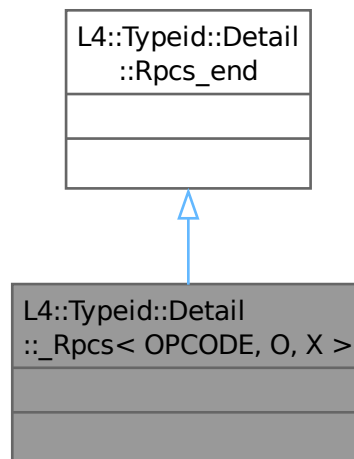
Empty list of RPCs.

```
#include <__typeinfo.h>
```

Inheritance diagram for L4::Typeid::Detail::_Rpc< OPCODE, O, X >:



Collaboration diagram for L4::Typeid::Detail::_Rpc< OPCODE, O, X >:



15.208.1 Detailed Description

```
template<typename OPCODE, unsigned O, typename ... X>
struct L4::Typeid::Detail::_Rpc< OPCODE, O, X >
```

Empty list of RPCs.

Definition at line 365 of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

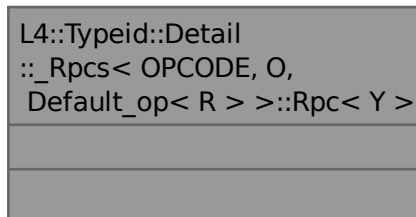
- [l4/sys/__typeinfo.h](#)

15.209 L4::Typeid::Detail::_Rpc< OPCODE, O, Default_op< R > >::Rpc< Y > Struct Template Reference

Find the given RPC in the list.

```
#include <__typeinfo.h>
```

Collaboration diagram for L4::Typeid::Detail::_Rpc< OPCODE, O, Default_op< R > >::Rpc< Y >:



15.209.1 Detailed Description

```
template<typename OPCODE, unsigned O, typename R>
template<typename Y>
struct L4::Typeid::Detail::_Rpc< OPCODE, O, Default_op< R > >::Rpc< Y >
```

Find the given RPC in the list.

Definition at line 399 of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

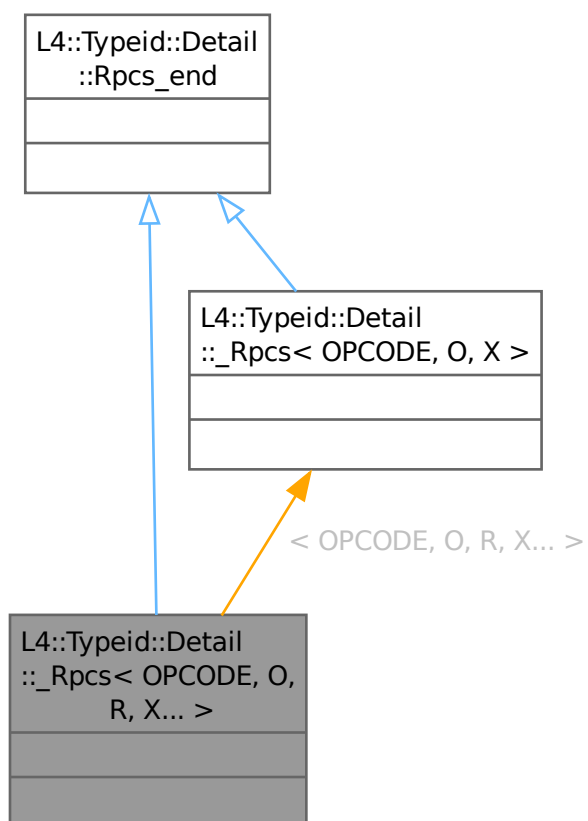
- [l4/sys/__typeinfo.h](#)

15.210 L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... > Struct Template Reference

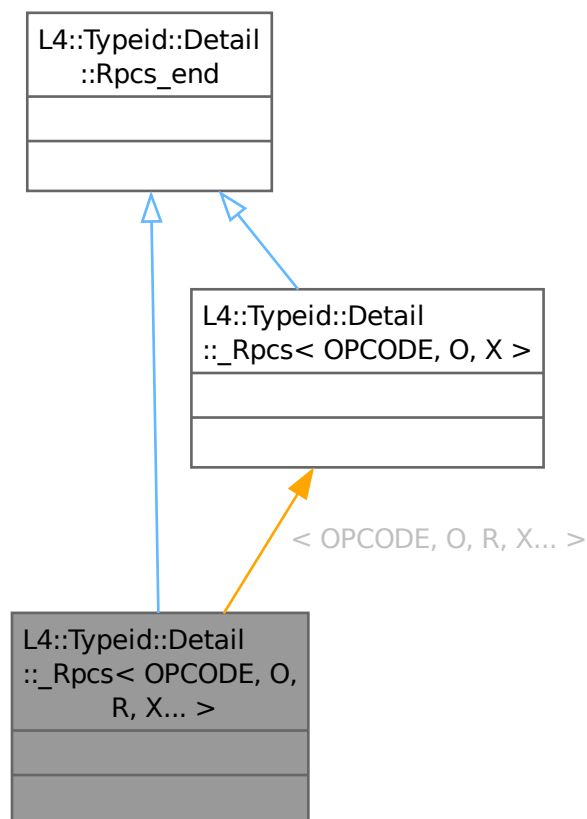
Non-empty list of RPCs.

```
#include <__typeinfo.h>
```

Inheritance diagram for L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... >:



Collaboration diagram for L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... >:



Data Structures

- struct [Rpc](#)

Find the given RPC in the list.

Public Types

- enum

The opcode value to use for this RPC, may be bogus if the [opcode_type](#) is void.

- typedef [_Rpc](#) type

The list element itself.

- typedef OPCODE [opcode_type](#)

The data type for the opcode.

- typedef R [rpc](#)

The RPC type `L4::lpc::Msg::Rpc_call` or `L4::lpc::Msg::Rpc_inline_call`.

- typedef [_Rpc](#)< OPCODE, [_Get_opcode](#)< R, O >::value+1, X... >::type next

The next RPC in the list or [Rpc_end](#) if this is the last.

15.210.1 Detailed Description

```
template<typename OPCODE, unsigned O, typename R, typename ... X>
struct L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... >
```

Non-empty list of RPCs.

Definition at line 369 of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

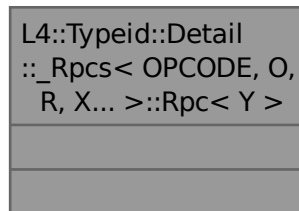
- [l4/sys/__typeinfo.h](#)

15.211 L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... >::Rpc< Y > Struct Template Reference

Find the given RPC in the list.

```
#include <__typeinfo.h>
```

Collaboration diagram for L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... >::Rpc< Y >:



15.211.1 Detailed Description

```
template<typename OPCODE, unsigned O, typename R, typename ... X>
template<typename Y>
struct L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... >::Rpc< Y >
```

Find the given RPC in the list.

Definition at line 382 of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

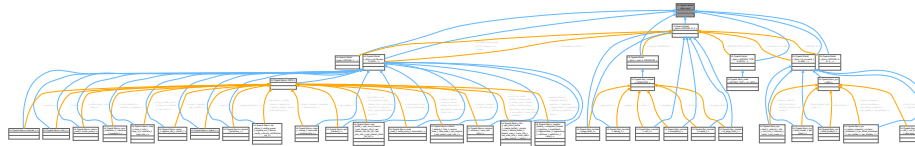
- [l4/sys/__typeinfo.h](#)

15.212 L4::Typeid::Detail::Rpc_end Struct Reference

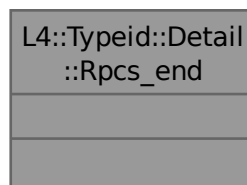
Internal end-of-list marker.

```
#include <__typeinfo.h>
```

Inheritance diagram for L4::Typeid::Detail::Rpc_end:



Collaboration diagram for L4::Typeid::Detail::Rpc_end:



15.212.1 Detailed Description

Internal end-of-list marker.

Definition at line 317 of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

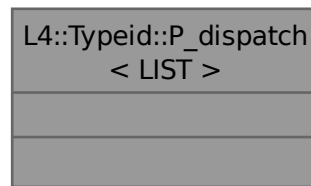
- [l4/sys/__typeinfo.h](#)

15.213 L4::Typeid::P_dispatch< LIST > Struct Template Reference

Use for protocol based dispatch stage.

```
#include <__typeinfo.h>
```

Collaboration diagram for L4::Typeid::P_dispatch< LIST >:



15.213.1 Detailed Description

```
template<typename LIST>
struct L4::Typeid::P_dispatch< LIST >
```

Use for protocol based dispatch stage.

Definition at line 308 of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

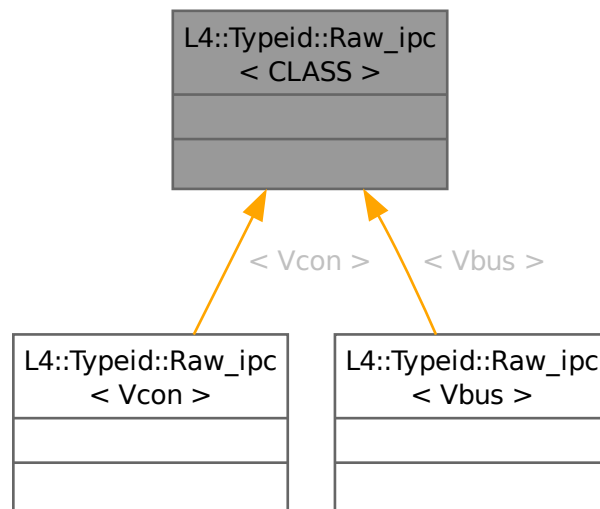
- [l4/sys/__typeinfo.h](#)

15.214 L4::Typeid::Raw_ipc< CLASS > Struct Template Reference

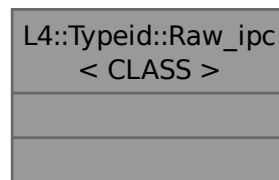
RPCs list for passing raw incoming IPC to the server object.

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Typeid::Raw_ipc< CLASS >:



Collaboration diagram for L4::Typeid::Raw_ipc< CLASS >:



15.214.1 Detailed Description

```

template<typename CLASS>
struct L4::Typeid::Raw_ipc< CLASS >

```

RPCs list for passing raw incoming IPC to the server object.

Template Parameters

| | |
|--------------|--|
| CLASS | The type of the interface (e.g., L4::lcu) |
|--------------|--|

This template allows to have fully handcrafted IPC protocols.

Definition at line 412 of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

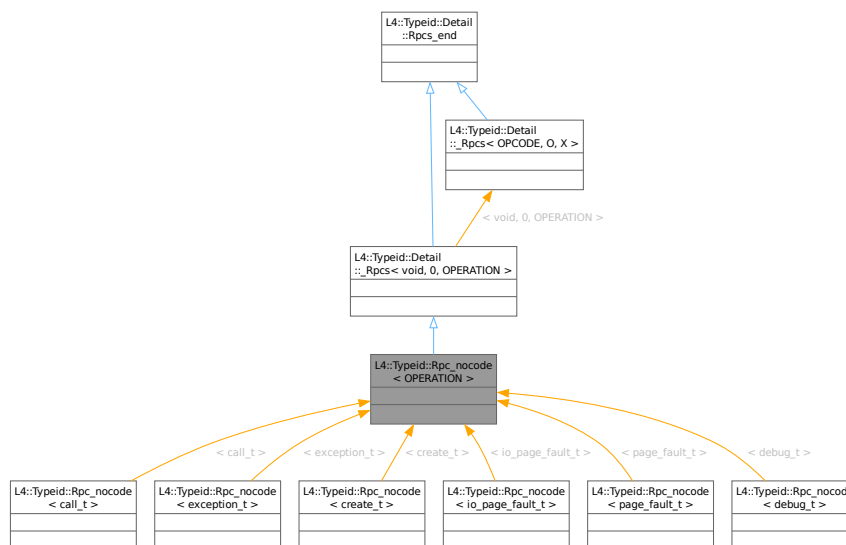
- [l4/sys/__typeinfo.h](#)

15.215 L4::Typeid::Rpc_nocode< OPERATION > Struct Template Reference

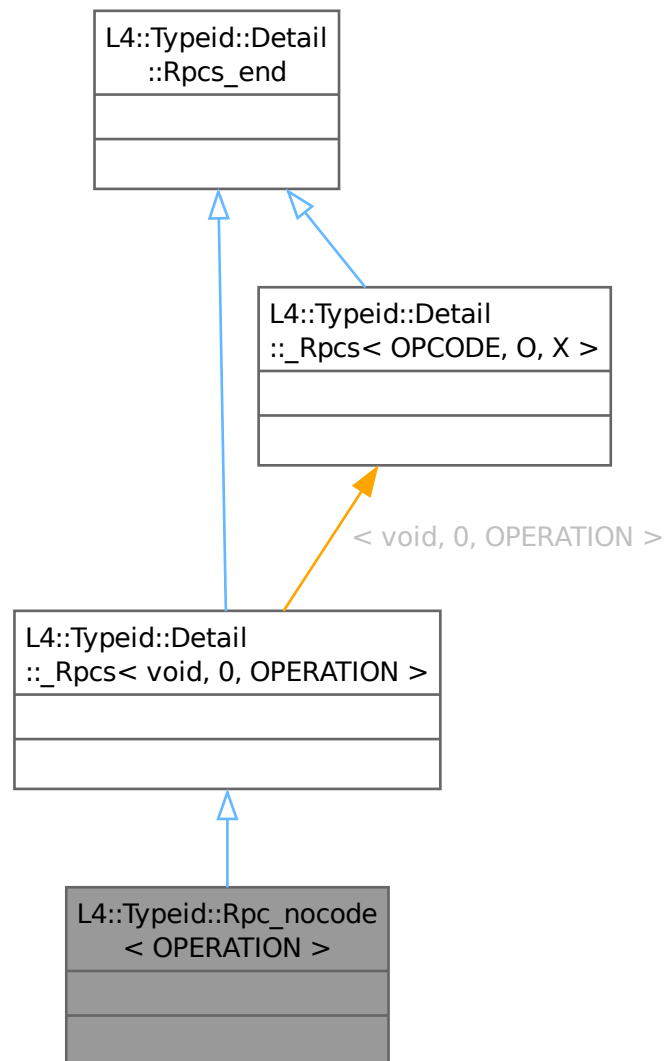
List of RPCs of an interface using a single operation without an opcode.

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Typeid::Rpc_nocode< OPERATION >:



Collaboration diagram for L4::Typeid::Rpc_nocode< OPERATION >:



15.215.1 Detailed Description

```
template<typename OPERATION>
struct L4::Typeid::Rpc_nocode< OPERATION >
```

List of RPCs of an interface using a single operation without an opcode.

Template Parameters

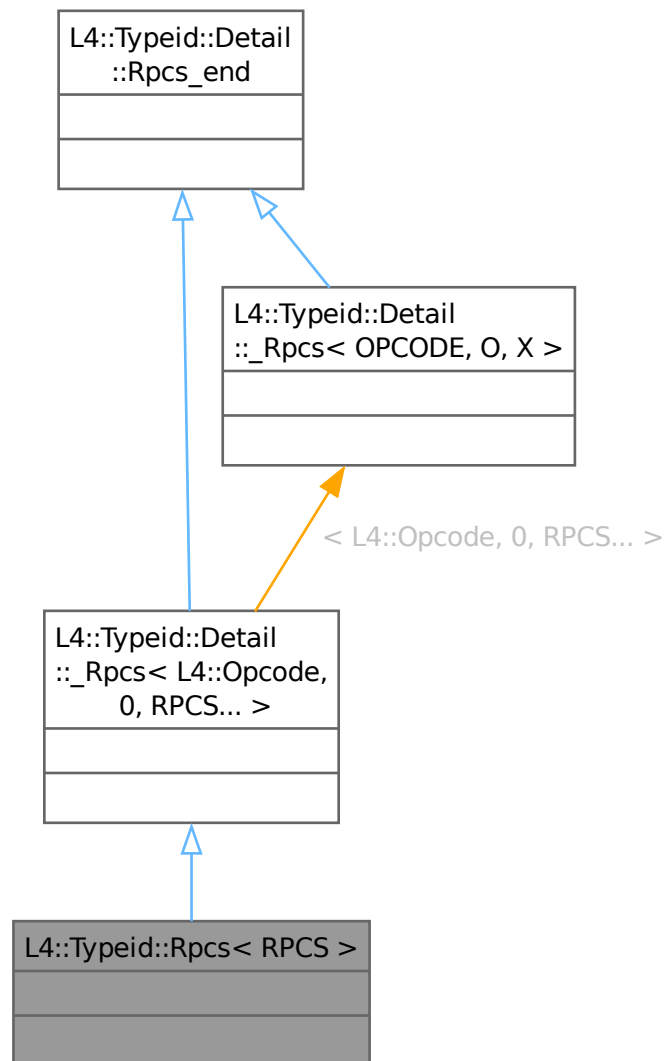
| | |
|------------------|---|
| <i>OPERATION</i> | The RPC operation as defined by L4_RPC etc. |
|------------------|---|

The documentation for this struct was generated from the following file:

- ## 15.216 L4::Typeid::Rpc< RPCS > Struct Template Reference

```
#include <linux/sys/capability>
```


Collaboration diagram for L4::Typeid::Rpc< RPCS >:



15.216.1 Detailed Description

```
template<typename ... RPCS>
struct L4::Typeid::Rpc< RPCS >
```

Standard list of RPCs of an interface.

Template Parameters

| | |
|-------------|---|
| <i>RPCS</i> | list of RPC types as defined by L4_RPC etc. |
|-------------|---|

This is the default list for RPC functions of an interface, it uses [L4::Opcode](#) as opcode type and uses opcodes starting from 0.

Examples

[examples/clntsrv/src/shared.h](#).

Definition at line 428 of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

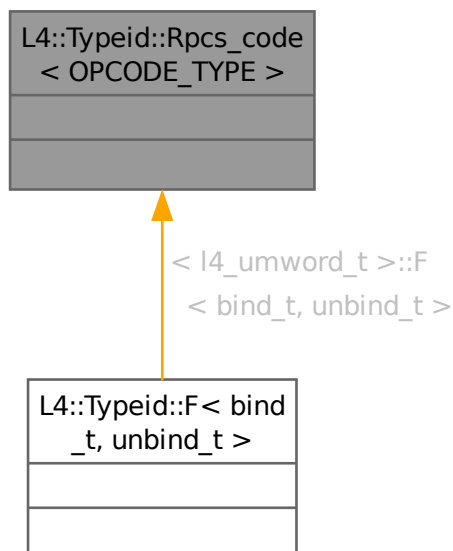
- [l4/sys/__typeinfo.h](#)

15.217 L4::Typeid::Rpc_code< OPCODE_TYPE > Struct Template Reference

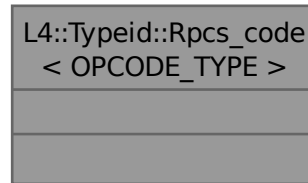
List of RPCs of an interface using a special opcode type.

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Typeid::Rpc_code< OPCODE_TYPE >:



Collaboration diagram for L4::Typeid::Rpc_code< OPCODE_TYPE >:



Data Structures

- struct [F](#)

15.217.1 Detailed Description

```
template<typename OPCODE_TYPE>
struct L4::Typeid::Rpc_code< OPCODE_TYPE >
```

List of RPCs of an interface using a special opcode type.

Template Parameters

| | |
|--------------------|------------------------------|
| <i>OPCODE_TYPE</i> | The data type of the opcode. |
|--------------------|------------------------------|

List for RPC functions of an interface, using OPCODE_TYPE as data type for the opcode, opcodes starting from 0.

Definition at line [439](#) of file [__typeinfo.h](#).

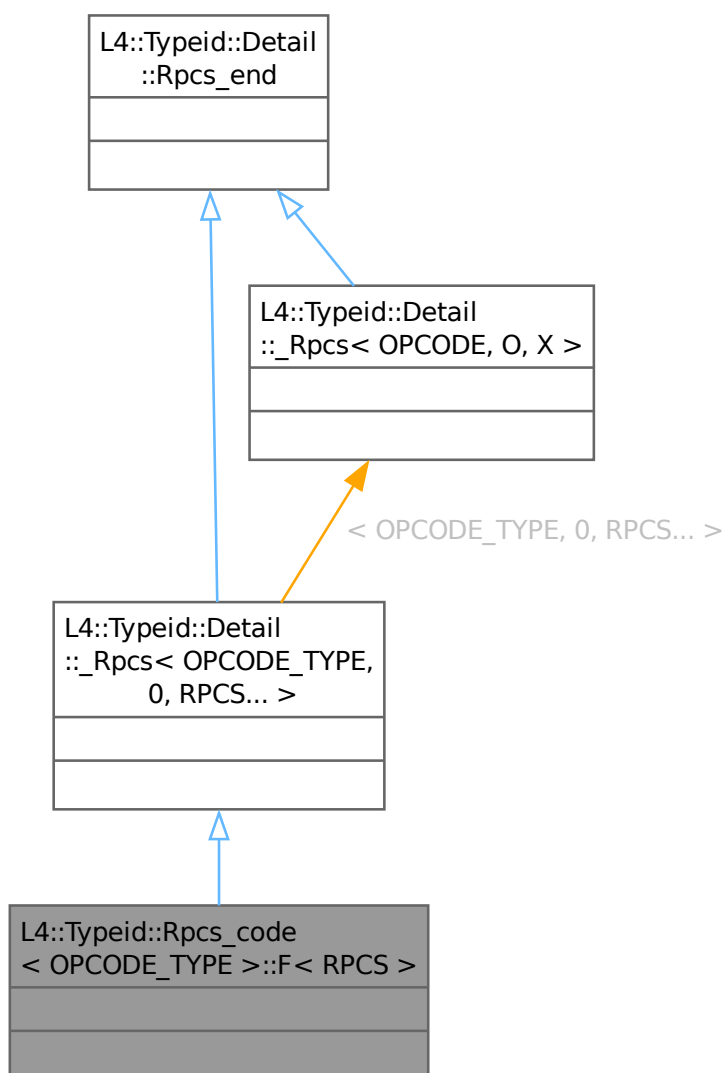
The documentation for this struct was generated from the following file:

- [l4/sys/__typeinfo.h](#)

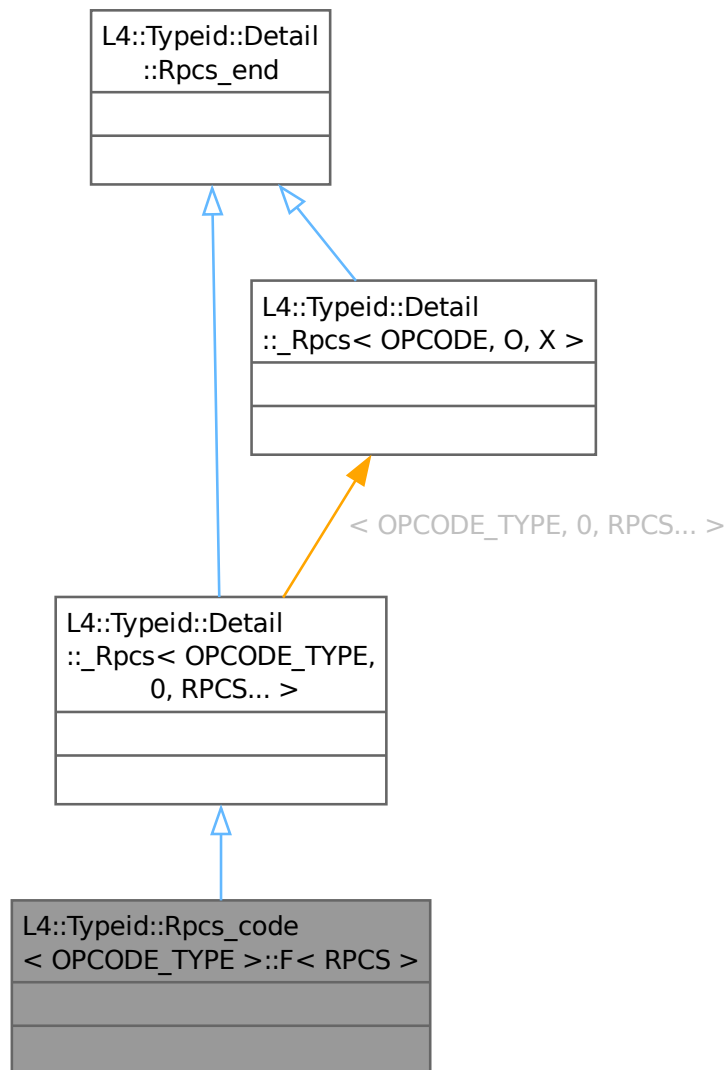
15.218 L4::Typeid::Rpc_code< OPCODE_TYPE >::F< RPCS > Struct Template Reference

```
#include <__typeinfo.h>
```

Inheritance diagram for L4::Typeid::Rpc_code< OPCODE_TYPE >::F< RPCS >:



Collaboration diagram for L4::Typeid::Rpc_code< OPCODE_TYPE >::F< RPCS >:



15.218.1 Detailed Description

```

template<typename OPCODE_TYPE>
template<typename ... RPCS>
struct L4::Typeid::Rpc_code< OPCODE_TYPE >::F< RPCS >

```

Template Parameters

| | |
|-------------|---|
| <i>RPCS</i> | list of RPC types as defined by L4_RPC etc. |
|-------------|---|

Definition at line 445 of file `__typeinfo.h`.

The documentation for this struct was generated from the following file:

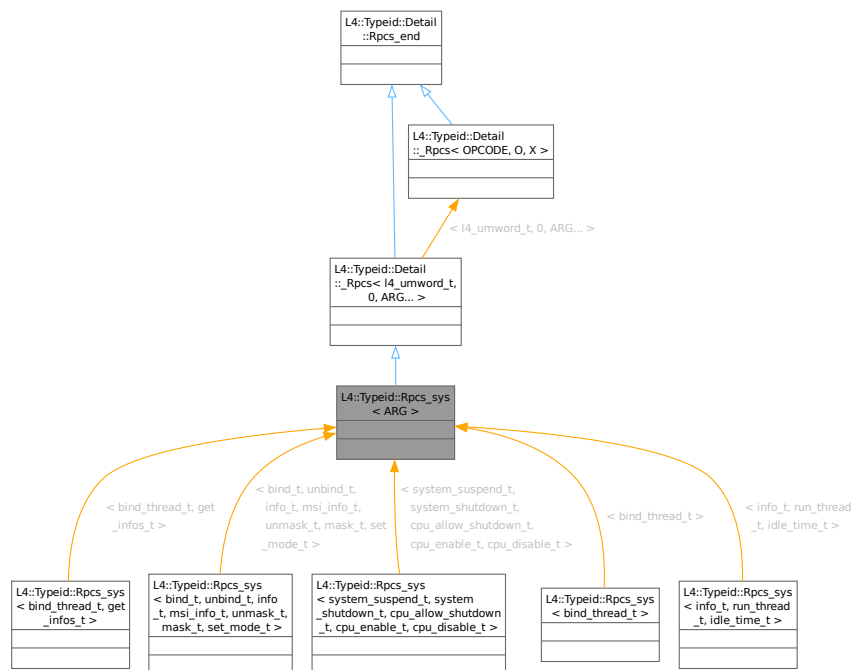
- [l4/sys/__typeinfo.h](#)

15.219 L4::Typeid::Rpcsys< ARG > Struct Template Reference

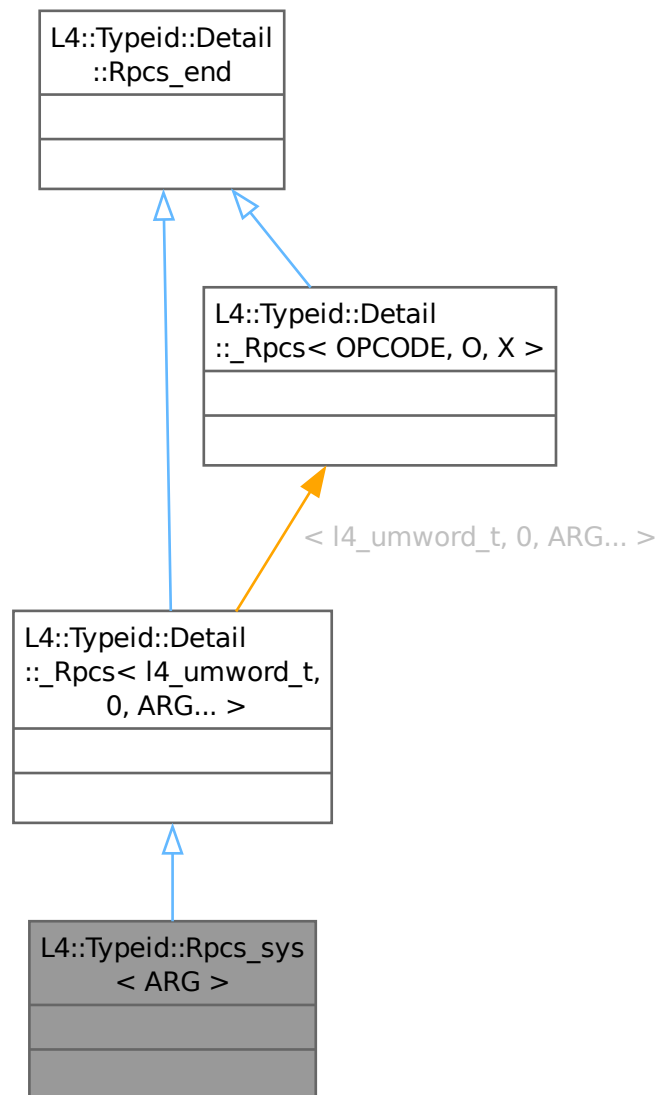
List of RPCs typically used for kernel interfaces.

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Typeid::Rpcsys< ARG >:



Collaboration diagram for L4::Typeid::Rpcsys< ARG >:



15.219.1 Detailed Description

```
template<typename ... ARG>
struct L4::Typeid::Rpcsys< ARG >
```

List of RPCs typically used for kernel interfaces.

Template Parameters

| | |
|-------------|---|
| <i>RPCS</i> | list of RPC types as defined by L4_RPC etc. |
|-------------|---|

This list of RPC functions uses [l4_umword_t](#) as type for the opcode as most kernel protocol do.

Definition at line [465](#) of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

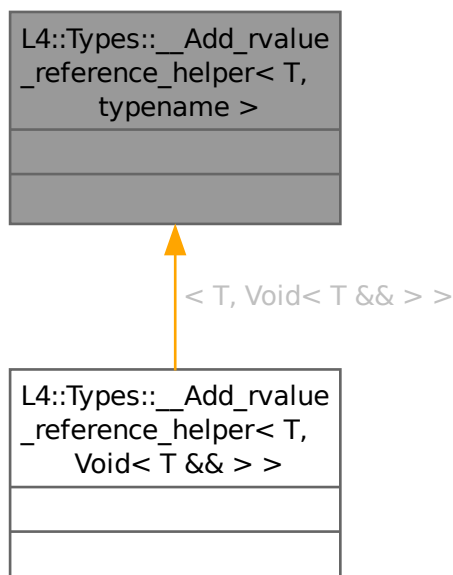
- [l4/sys/__typeinfo.h](#)

15.220 L4::Types::__Add_rvalue_reference_helper< T, typename > Struct Template Reference

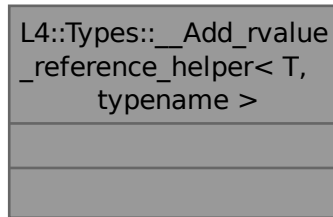
Helper template for [Add_rvalue_reference](#).

```
#include <types>
```

Inheritance diagram for L4::Types::__Add_rvalue_reference_helper< T, typename >:



Collaboration diagram for L4::Types::__Add_rvalue_reference_helper< T, typename >:



15.220.1 Detailed Description

```
template<typename T, typename = void>
struct L4::Types::__Add_rvalue_reference_helper< T, typename >
```

Helper template for [Add_rvalue_reference](#).

Definition at line 283 of file [types](#).

The documentation for this struct was generated from the following file:

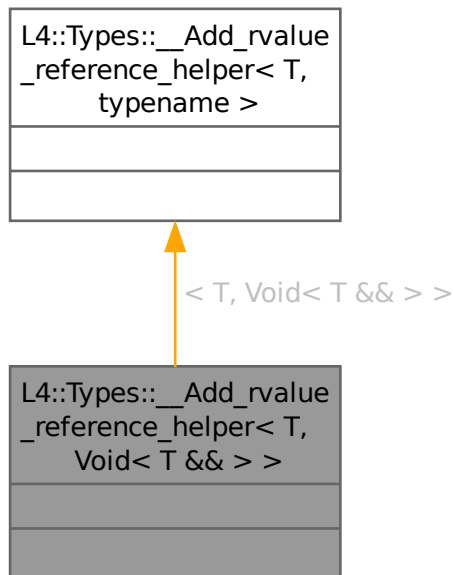
- [l4/sys/cxx/types](#)

15.221 L4::Types::__Add_rvalue_reference_helper< T, Void< T && > > Struct Template Reference

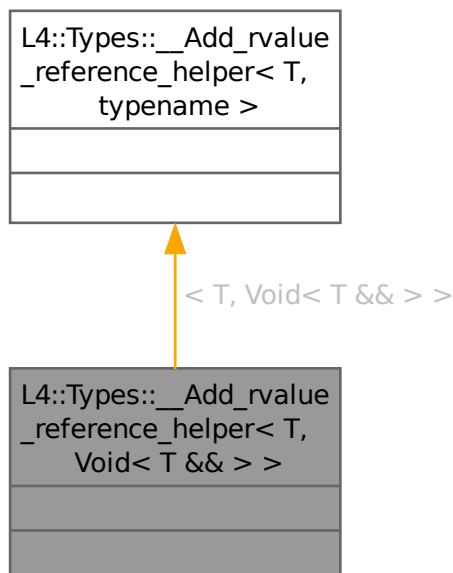
Helper template for [Add_rvalue_reference](#).

```
#include <types>
```

Inheritance diagram for L4::Types::__Add_rvalue_reference_helper< T, Void< T && > >:



Collaboration diagram for L4::Types::__Add_rvalue_reference_helper< T, Void< T && > >:



15.221.1 Detailed Description

```
template<typename T>
struct L4::Types::__Add_rvalue_reference_helper< T, Void< T && > >
```

Helper template for [Add_rvalue_reference](#).

Definition at line 287 of file [types](#).

The documentation for this struct was generated from the following file:

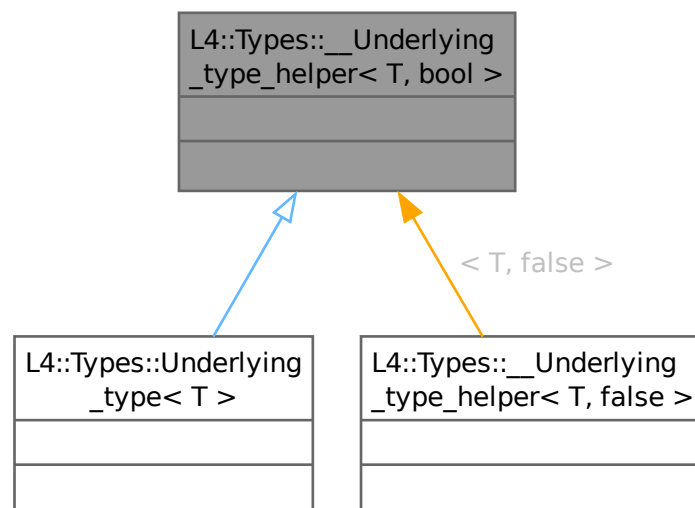
- [l4/sys/cxx/types](#)

15.222 L4::Types::__Underlying_type_helper< T, bool > Struct Template Reference

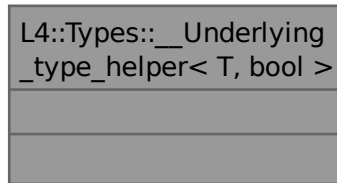
Helper template for [Underlying_type](#).

```
#include <types>
```

Inheritance diagram for L4::Types::__Underlying_type_helper< T, bool >:



Collaboration diagram for L4::Types::__Underlying_type_helper< T, bool >:



15.222.1 Detailed Description

```
template<typename T, bool = Is_enum<T>::value>
struct L4::Types::__Underlying_type_helper< T, bool >
```

Helper template for [Underlying_type](#).

Note

The implementation relies on the intrinsic `__underlying_type()` compiler type trait that is provided by the mainstream C++ compilers. There is no fully portable and future-proof way of implementing this template (the only fragile and unmaintainable possibility is to evaluate all possible underlying types).

Template Parameters

| | |
|----------|---|
| <i>T</i> | Enumeration type to get the underlying type of. |
|----------|---|

Definition at line [362](#) of file [types](#).

The documentation for this struct was generated from the following file:

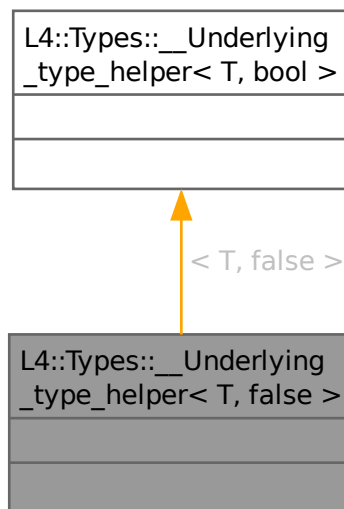
- [l4/sys/cxx/types](#)

15.223 L4::Types::__Underlying_type_helper< T, false > Struct Template Reference

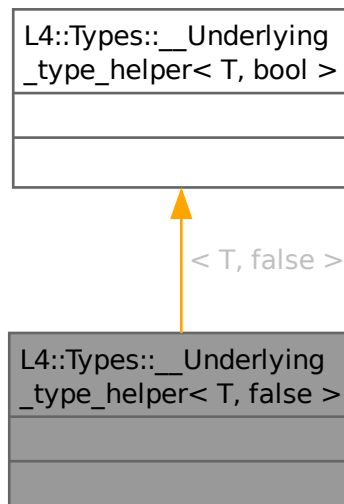
Helper template for [Underlying_type](#).

```
#include <types>
```

Inheritance diagram for L4::Types::__Underlying_type_helper< T, false >:



Collaboration diagram for L4::Types::__Underlying_type_helper< T, false >:



15.223.1 Detailed Description

```
template<typename T>
struct L4::Types::__Underlying_type_helper< T, false >
```

Helper template for [Underlying_type](#).

Definition at line 365 of file [types](#).

The documentation for this struct was generated from the following file:

- [l4/sys/cxx/types](#)

15.224 L4::Types::Add_rvalue_reference< T > Struct Template Reference

Create an rvalue reference of the given type.

```
#include <types>
```

Collaboration diagram for L4::Types::Add_rvalue_reference< T >:

| |
|--|
| L4::Types::Add_rvalue _reference< T > |
| |
| |

15.224.1 Detailed Description

```
template<typename T>
struct L4::Types::Add_rvalue_reference< T >
```

Create an rvalue reference of the given type.

Definition at line 291 of file [types](#).

The documentation for this struct was generated from the following file:

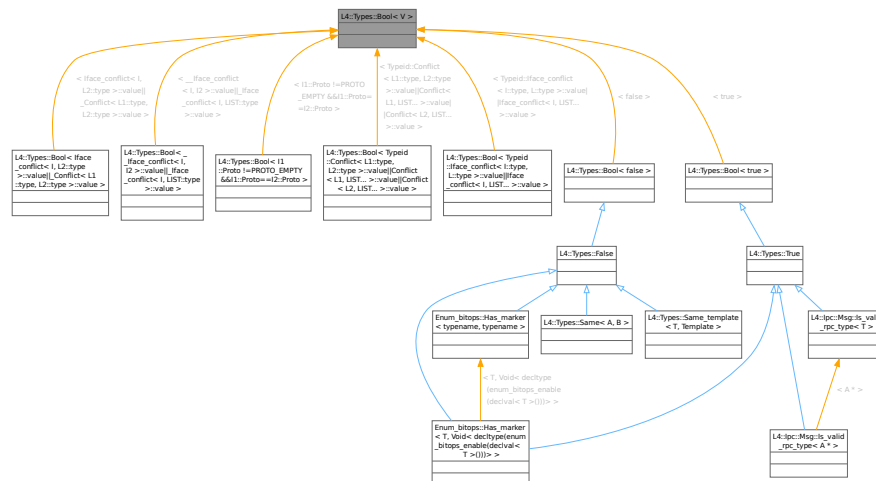
- [l4/sys/cxx/types](#)

15.225 L4::Types::Bool< V > Struct Template Reference

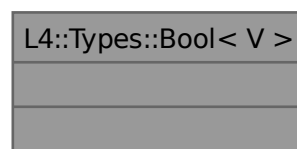
Boolean meta type.

```
#include <types>
```

Inheritance diagram for L4::Types::Bool< V >:



Collaboration diagram for L4::Types::Bool< V >:



Public Types

- using **type** = Bool<V>
The meta type itself.

15.225.1 Detailed Description

```
template<bool V>
struct L4::Types::Bool< V >
```

Boolean meta type.

Template Parameters

| | |
|----------|-------------------|
| <i>V</i> | The boolean value |
|----------|-------------------|

Definition at line 312 of file [types](#).

The documentation for this struct was generated from the following file:

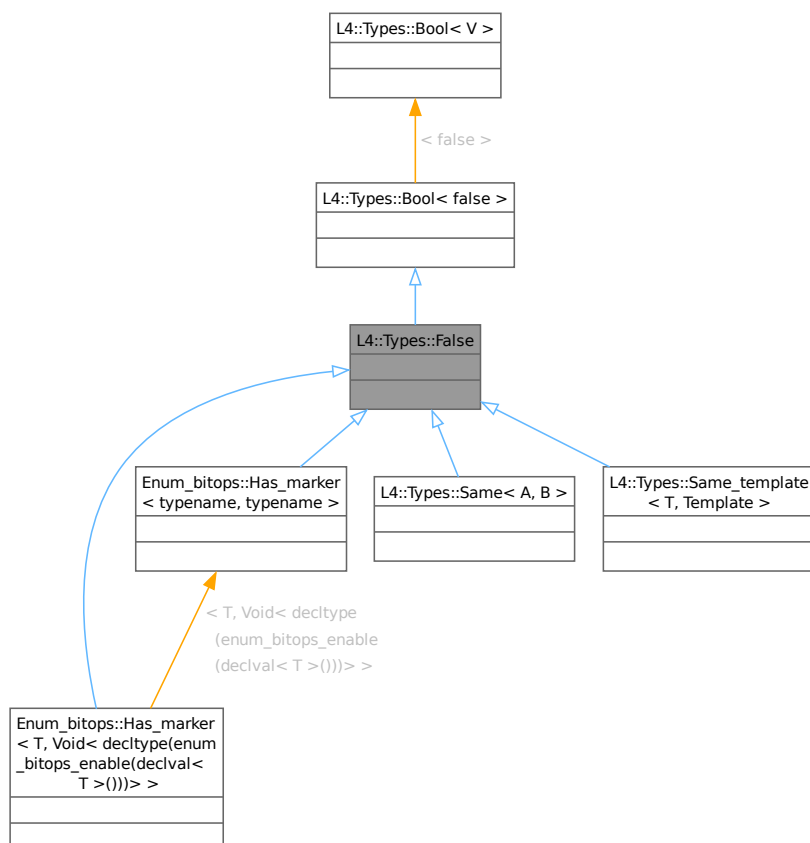
- [l4/sys/cxx/types](#)

15.226 L4::Types::False Struct Reference

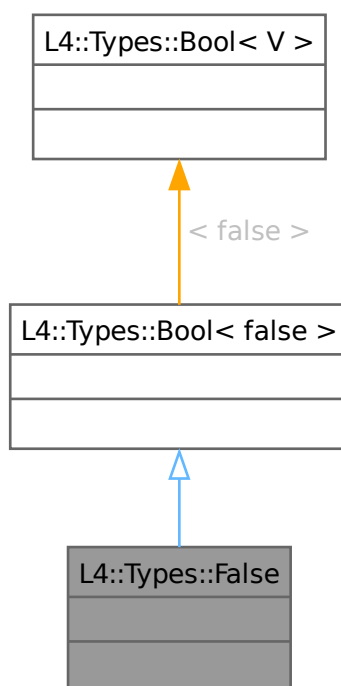
[False](#) meta value.

```
#include <types>
```

Inheritance diagram for L4::Types::False:



Collaboration diagram for L4::Types::False:



Additional Inherited Members

Public Types inherited from `L4::Types::Bool< false >`

- using **type**
The meta type itself.

15.226.1 Detailed Description

`False` meta value.

Definition at line 320 of file `types`.

The documentation for this struct was generated from the following file:

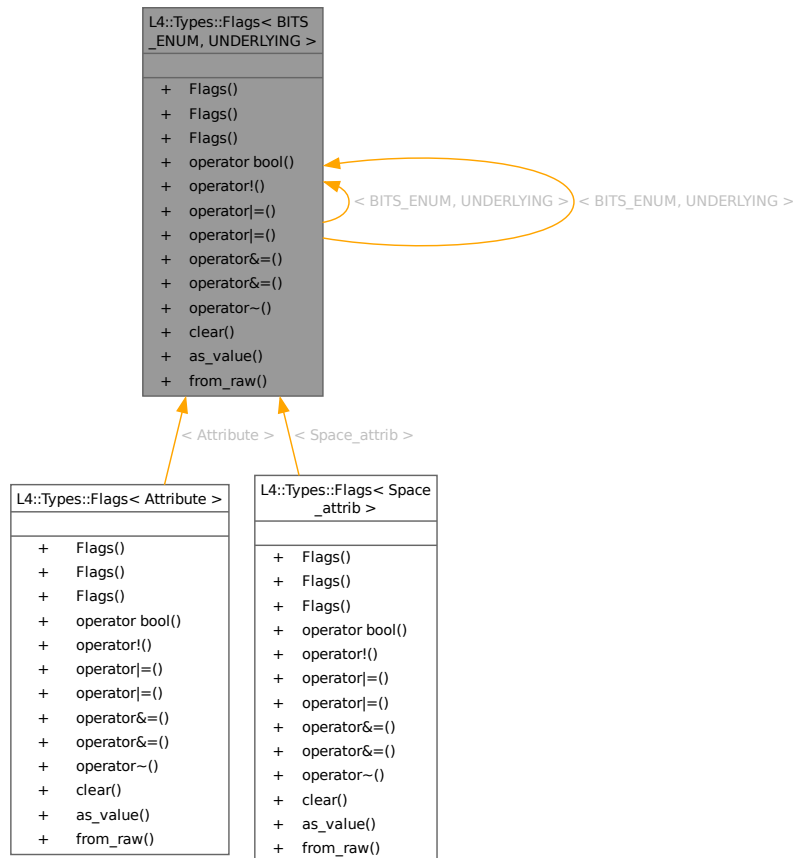
- `l4/sys/cxx/types`

15.227 L4::Types::Flags< BITS_ENUM, UNDERLYING > Class Template Reference

Template for defining typical [Flags](#) bitmaps.

```
#include <types>
```

Inheritance diagram for L4::Types::Flags< BITS_ENUM, UNDERLYING >:



Collaboration diagram for L4::Types::Flags< BITS_ENUM, UNDERLYING >:

| L4::Types::Flags< BITS_ENUM, UNDERLYING > |
|---|
| <ul style="list-style-type: none"> + Flags() + Flags() + Flags() + operator bool() + operator!() + operator =() + operator =() + operator&=() + operator&=() + operator~() + clear() + as_value() + from_raw() |

Public Types

- enum [None_type](#) { [None](#) }
The none type to get an empty bitmap.
- using **value_type** = UNDERLYING
type of the underlying value
- using **bits_enum_type** = BITS_ENUM
enum type defining a name for each bit
- using **type** = Flags<BITS_ENUM, UNDERLYING>
the Flags<> type itself

Public Member Functions

- constexpr [Flags](#) ([None_type](#)) noexcept
Make an empty bitmap.
- constexpr **Flags** () noexcept
Make default [Flags](#).
- constexpr [Flags](#) (BITS_ENUM e) noexcept
Make flags from bit name.
- constexpr **operator bool** () const noexcept
Support for `if (flags)` syntax (test for non-empty flags).

- constexpr bool **operator!** () const noexcept
Support for `if (!flags)` syntax (test for empty flags).
- constexpr **type** & **operator|**= (type rhs) noexcept
Support |= of two compatible [Flags](#) types.
- constexpr **type** & **operator|**= (bits_enum_type rhs)
Support |= of [Flags](#) type and bit name.
- constexpr **type** & **operator&=** (type rhs)
Support &= of two compatible [Flags](#) types.
- constexpr **type** & **operator&=** (bits_enum_type rhs)
Support &= of [Flags](#) type and bit name.
- constexpr **type** **operator~** () const noexcept
Support ~ for [Flags](#) types.
- constexpr **type** & **clear** (bits_enum_type flag) noexcept
Clear the given flag.
- constexpr **value_type** **as_value** () const noexcept
Get the underlying value.

Static Public Member Functions

- static constexpr **type from_raw** (value_type v) noexcept
Make flags from a raw value of [value_type](#).

Friends

- constexpr **type operator|** (type lhs, type rhs) noexcept
Support | of two compatible [Flags](#) types.
- constexpr **type operator|** (type lhs, bits_enum_type rhs) noexcept
Support | of [Flags](#) type and bit name.
- constexpr **type operator&** (type lhs, type rhs) noexcept
Support & of two compatible [Flags](#) types.
- constexpr **type operator&** (type lhs, bits_enum_type rhs) noexcept
Support & of [Flags](#) type and bit name.

15.227.1 Detailed Description

```
template<typename BITS_ENUM, typename UNDERLYING = unsigned long>
class L4::Types::Flags< BITS_ENUM, UNDERLYING >
```

Template for defining typical [Flags](#) bitmaps.

Template Parameters

| | |
|-------------------|--|
| <i>BITS_ENUM</i> | enum type that defines a name for each bit in the bitmap. The values of the enum members must be the number of the bit (<i>not</i> a mask). |
| <i>UNDERLYING</i> | The underlying data type used to represent the bitmap. |

The resulting data type provides a type-safe version that allows bitwise `and` and `or` operations with the `BITS_ENUM` members. As well as, test for `0` or `!0`.

Example:

```
enum Test_flag
{
    Do_weak_tests,
    Do_strong_tests
};

using Test_flags = L4::Types::Flags<Test_flag>;

Test_flags x = Do_weak_tests;

if (x & Do_strong_tests) { ... }
x |= Do_strong_tests;
if (x & Do_strong_tests) { ... }
```

Definition at line 52 of file [types](#).

15.227.2 Member Enumeration Documentation

15.227.2.1 None_type

```
template<typename BITS_ENUM, typename UNDERLYING = unsigned long>
enum L4::Types::Flags::None_type
```

The none type to get an empty bitmap.

Enumerator

| | |
|------|----------------------------------|
| None | Use this to get an empty bitmap. |
|------|----------------------------------|

Definition at line 68 of file [types](#).

15.227.3 Constructor & Destructor Documentation

15.227.3.1 Flags() [1/2]

```
template<typename BITS_ENUM, typename UNDERLYING = unsigned long>
L4::Types::Flags< BITS_ENUM, UNDERLYING >::Flags (
    None_type ) [inline], [constexpr], [noexcept]
```

Make an empty bitmap.

Usually used for implicit conversion from [Flags::None](#).

```
Flags x = Flags::None;
```

Definition at line 78 of file [types](#).

15.227.3.2 Flags() [2/2]

```
template<typename BITS_ENUM, typename UNDERLYING = unsigned long>
L4::Types::Flags< BITS_ENUM, UNDERLYING >::Flags (
    BITS_ENUM e) [inline], [constexpr], [noexcept]
```

Make flags from bit name.

Usually used for implicit conversion for a bit name.

```
Test_flags f = Do_strong_tests;
```

Definition at line 91 of file [types](#).

15.227.4 Member Function Documentation

15.227.4.1 clear()

```
template<typename BITS_ENUM, typename UNDERLYING = unsigned long>
type & L4::Types::Flags< BITS_ENUM, UNDERLYING >::clear (
    bits_enum_type flag) [inline], [constexpr], [noexcept]
```

Clear the given flag.

Parameters

| | |
|-------------|---------------------------------|
| <i>flag</i> | The flag that shall be cleared. |
|-------------|---------------------------------|

`flags.clear(The_flag)` is a shortcut for `flags &= ~Flags(The_flag)`.

Definition at line 149 of file [types](#).

15.227.4.2 from_raw()

```
template<typename BITS_ENUM, typename UNDERLYING = unsigned long>
constexpr type L4::Types::Flags< BITS_ENUM, UNDERLYING >::from_raw (
    value_type v) [inline], [static], [constexpr], [noexcept]
```

Make flags from a raw value of [value_type](#).

This function may be used for example in C wrapper code.

Definition at line 100 of file [types](#).

The documentation for this class was generated from the following file:

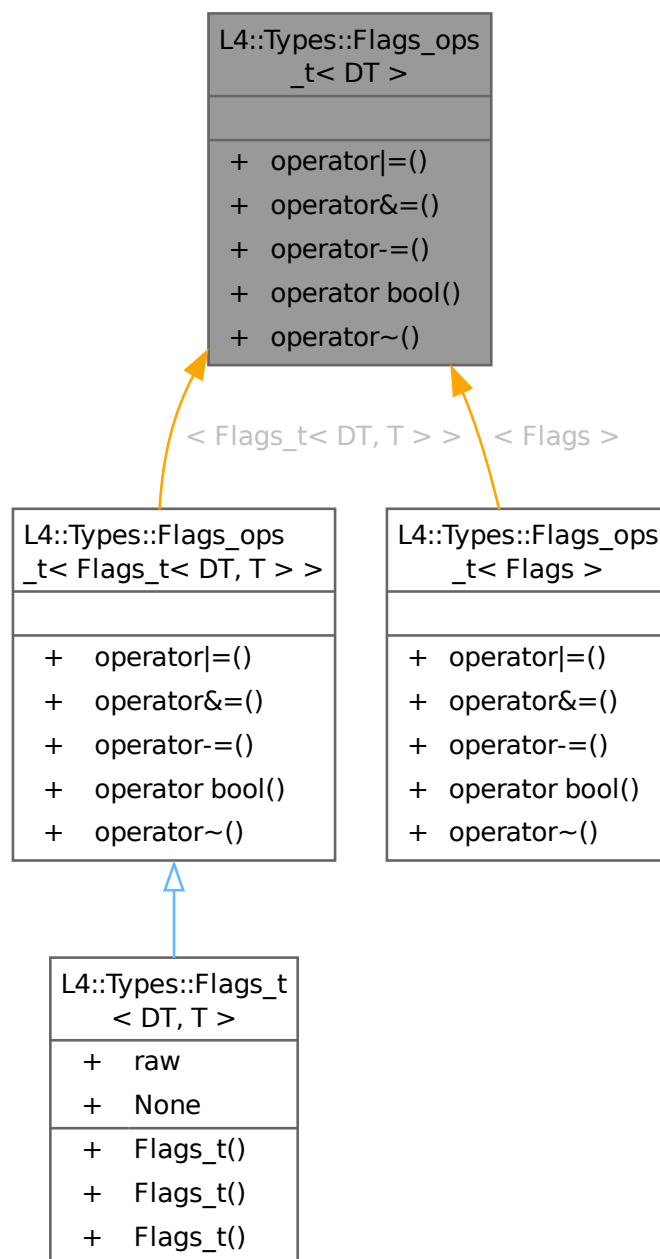
- [l4/sys/cxx/types](#)

15.228 L4::Types::Flags_ops_t< DT > Struct Template Reference

Mixin class to define a set of friend bitwise operators on `DT`.

```
#include <types>
```

Inheritance diagram for L4::Types::Flags_ops_t< DT >:



Collaboration diagram for L4::Types::Flags_ops_t< DT >:

| L4::Types::Flags_ops_t< DT > |
|---|
| <ul style="list-style-type: none"> + operator =() + operator&=() + operator-=() + operator bool() + operator~() |

Public Member Functions

- constexpr DT & **operator|**=(DT const r)
bitwise or assignment for DT
- constexpr DT & **operator&=** (DT const r)
bitwise and assignment for DT
- constexpr DT & **operator-=** (DT const r)
Bitwise difference (clear bits) assignment for DT.
- constexpr **operator bool** () const
explicit conversion to bool for tests
- constexpr DT **operator~** () const
bitwise negation for DT

Friends

- constexpr DT **operator|** (DT l, DT r)
bitwise or for DT
- constexpr DT **operator&** (DT l, DT r)
bitwise and for DT
- constexpr DT **operator-** (DT l, DT r)
Bitwise difference (clear bits) for DT.
- constexpr bool **operator==** (DT l, DT r)
equality for DT
- constexpr bool **operator!=** (DT l, DT r)
inequality for DT

15.228.1 Detailed Description

```
template<typename DT>
struct L4::Types::Flags_ops_t< DT >
```

Mixin class to define a set of friend bitwise operators on `DT`.

Template Parameters

| | |
|-----------|---|
| <i>DT</i> | The type usually inheriting from Flags_ops_t with a member <i>raw</i> of enum or integral type. |
|-----------|---|

Definition at line 203 of file [types](#).

The documentation for this struct was generated from the following file:

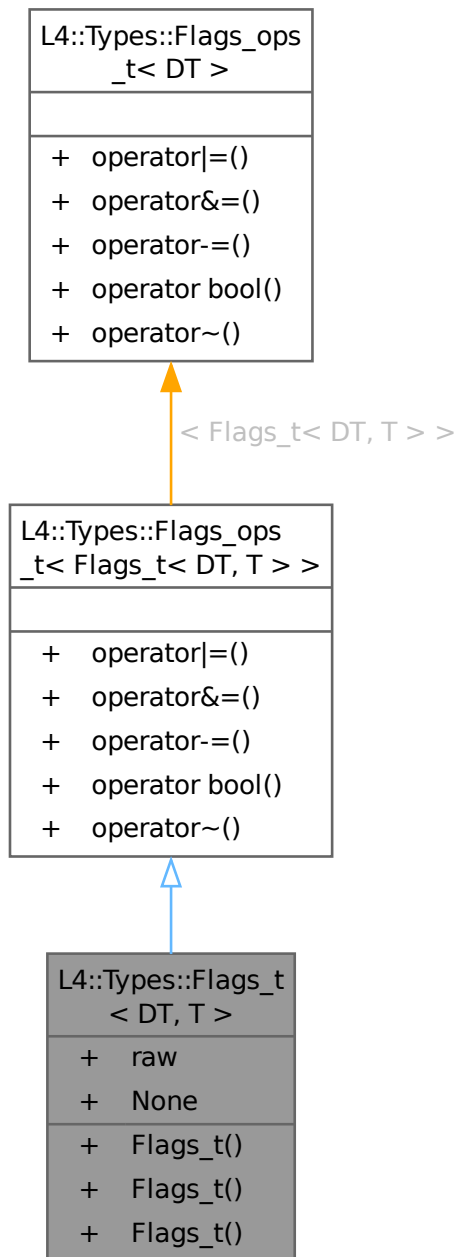
- [l4/sys/cxx/types](#)

15.229 L4::Types::Flags_t< DT, T > Struct Template Reference

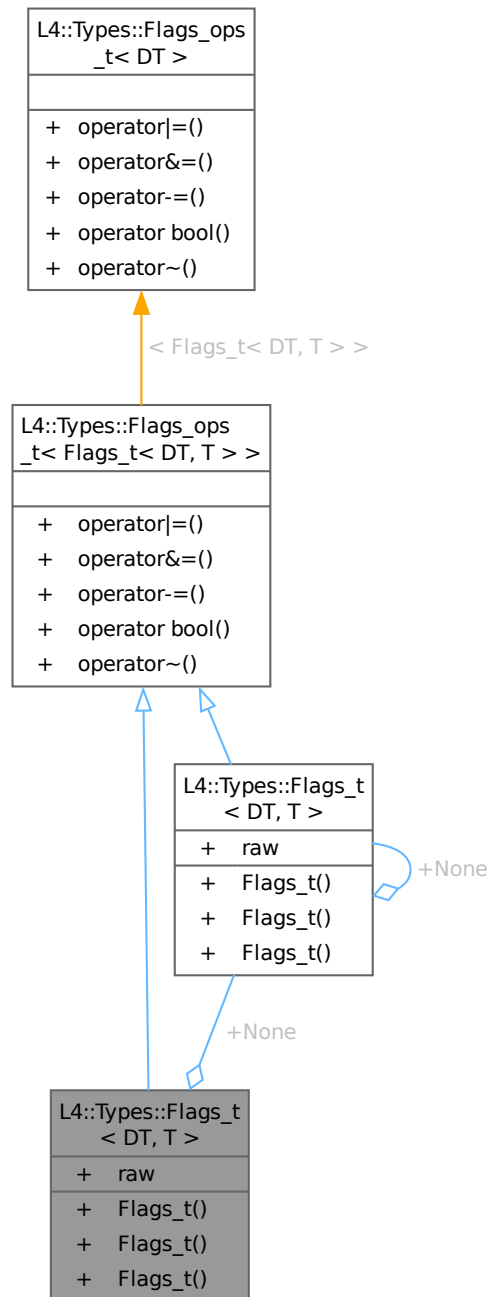
Template type to define a flags type with bitwise operations.

```
#include <types>
```

Inheritance diagram for L4::Types::Flags_t< DT, T >:



Collaboration diagram for L4::Types::Flags_t< DT, T >:



Public Member Functions

- constexpr **Flags_t** ()=default
Default (uninitializing) constructor.
- constexpr **Flags_t** (DT f)
Initialization from determinator type.
- constexpr **Flags_t** (T f)
Explicit initialization from the underlying type.

Public Member Functions inherited from [L4::Types::Flags_ops_t< Flags_t< DT, T > >](#)

- constexpr DT & **operator**|= (DT const r)
bitwise or assignment for DT
- constexpr DT & **operator**&= (DT const r)
bitwise and assignment for DT
- constexpr DT & **operator**-= (DT const r)
Bitwise difference (clear bits) assignment for DT.
- constexpr **operator** bool () const
explicit conversion to bool for tests
- constexpr DT **operator**~ () const
bitwise negation for DT

Data Fields

- T raw = 0
Raw integral value.

Static Public Attributes

- static constexpr [Flags_t](#) None = [Flags_t](#)()
Empty flags literal.

15.229.1 Detailed Description

```
template<typename DT, typename T>
struct L4::Types::Flags_t< DT, T >
```

Template type to define a flags type with bitwise operations.

Template Parameters

| | |
|-----------|---|
| <i>DT</i> | determinator type to make the resulting type unique (unused). |
| <i>T</i> | underlying type used to store the bits, usually an integral type. |

Definition at line [264](#) of file [types](#).

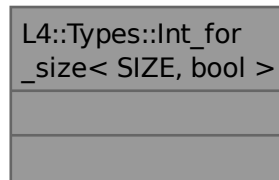
The documentation for this struct was generated from the following file:

- [l4/sys/cxx/types](#)

15.230 L4::Types::Int_for_size< SIZE, bool > Struct Template Reference

Metafunction to get an unsigned integral type for the given size.

Collaboration diagram for L4::Types::Int_for_size< SIZE, bool >:



15.230.1 Detailed Description

```
template<unsigned SIZE, bool = true>
struct L4::Types::Int_for_size< SIZE, bool >
```

Metafunction to get an unsigned integral type for the given size.

Template Parameters

| | |
|-------------|-----------------------------------|
| <i>SIZE</i> | The size of the integer in bytes. |
|-------------|-----------------------------------|

Definition at line 161 of file [types](#).

The documentation for this struct was generated from the following file:

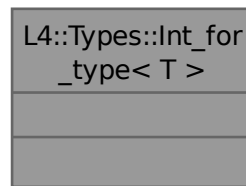
- [l4/sys/cxx/types](#)

15.231 L4::Types::Int_for_type< T > Struct Template Reference

Metafunction to get an integral type of the same size as `T`.

```
#include <types>
```

Collaboration diagram for L4::Types::Int_for_type< T >:



Public Types

- using **type** = typename [Int_for_size](#)<sizeof(T)>::type
The resulting unsigned integer type with the size like T.

15.231.1 Detailed Description

```
template<typename T>
struct L4::Types::Int_for_type< T >
```

Metafunction to get an integral type of the same size as T.

Template Parameters

| | |
|----------|--|
| <i>T</i> | The type for which an unsigned integral type with the same size is needed. |
|----------|--|

Definition at line [188](#) of file [types](#).

The documentation for this struct was generated from the following file:

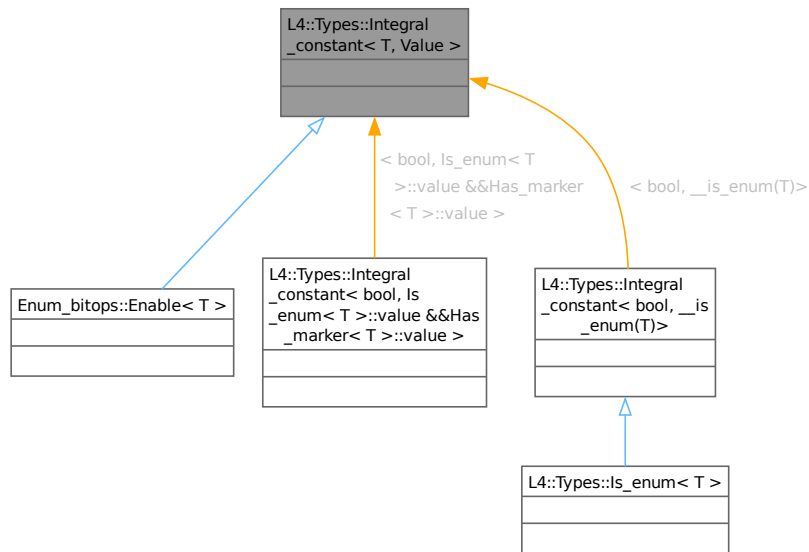
- [l4/sys/cxx/types](#)

15.232 L4::Types::Integral_constant< T, Value > Struct Template Reference

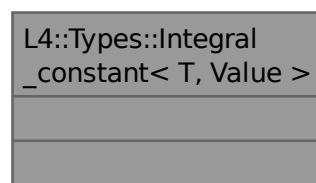
Wrapper for a static constant of the given type.

```
#include <types>
```

Inheritance diagram for L4::Types::Integral_constant< T, Value >:



Collaboration diagram for L4::Types::Integral_constant< T, Value >:



15.232.1 Detailed Description

```
template<typename T, T Value>
struct L4::Types::Integral_constant< T, Value >
```

Wrapper for a static constant of the given type.

Definition at line 328 of file [types](#).

The documentation for this struct was generated from the following file:

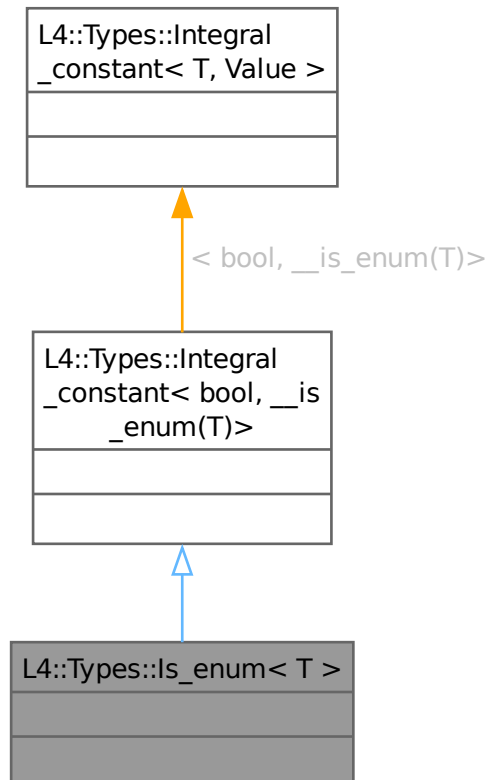
- [l4/sys/cxx/types](#)

15.233 L4::Types::ls_enum< T > Struct Template Reference

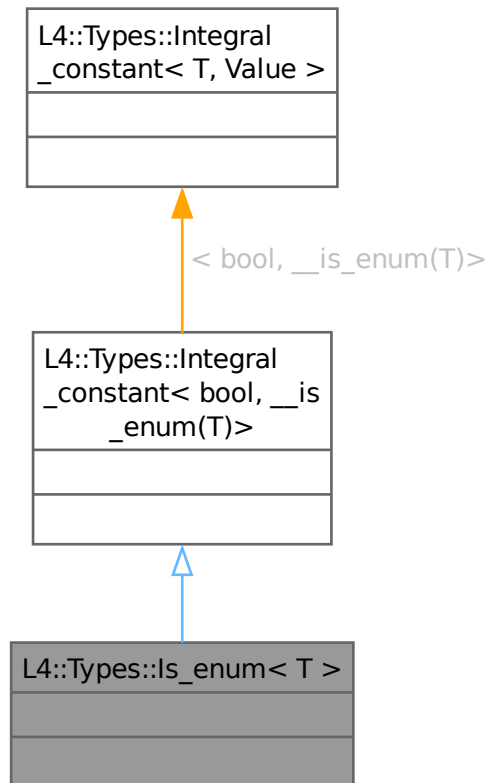
Check whether the given type is an enumeration type.

```
#include <types>
```

Inheritance diagram for L4::Types::ls_enum< T >:



Collaboration diagram for L4::Types::ls_enum< T >:



15.233.1 Detailed Description

```
template<typename T>
struct L4::Types::ls_enum< T >
```

Check whether the given type is an enumeration type.

Note

The implementation relies on the intrinsic `__is_enum()` compiler type trait that is provided by the mainstream C++ compilers. There is no fully portable and future-proof way of implementing this template (the only fragile and unmaintainable possibility is to exclude all other types to identify the enumeration type).

Template Parameters

| | |
|----------|--|
| <i>T</i> | Type to check whether it is an enumeration type. |
|----------|--|

Definition at line 348 of file [types](#).

The documentation for this struct was generated from the following file:

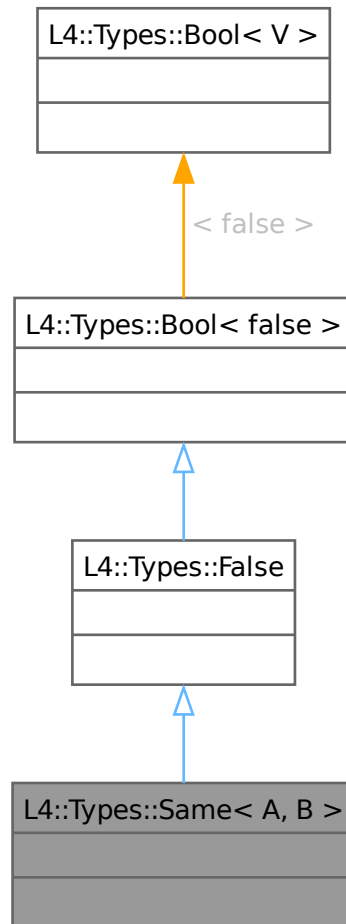
- [I4/sys/cxx/types](#)

15.234 L4::Types::Same< A, B > Struct Template Reference

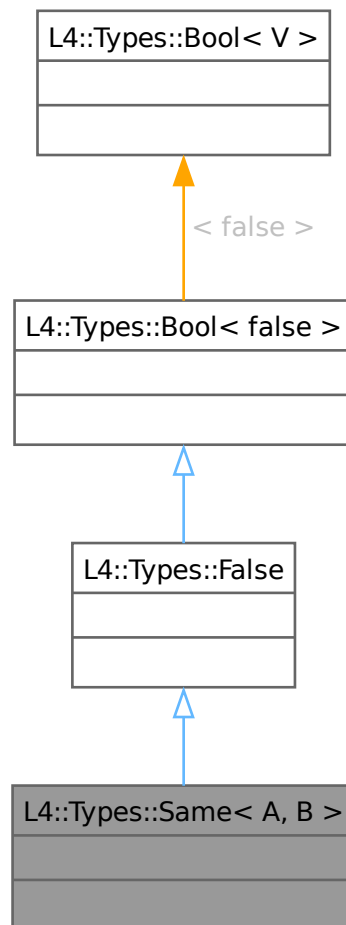
Compare two data types for equality.

```
#include <types>
```

Inheritance diagram for L4::Types::Same< A, B >:



Collaboration diagram for L4::Types::Same< A, B >:



Additional Inherited Members

Public Types inherited from **L4::Types::Bool< false >**

- using **type**
The meta type itself.

15.234.1 Detailed Description

```
template<typename A, typename B>
struct L4::Types::Same< A, B >
```

Compare two data types for equality.

Template Parameters

| | |
|----------|----------------------|
| <i>A</i> | The first data type |
| <i>B</i> | The second data type |

The result is the boolean [True](#) if A and B are the same types.

Definition at line [384](#) of file [types](#).

The documentation for this struct was generated from the following file:

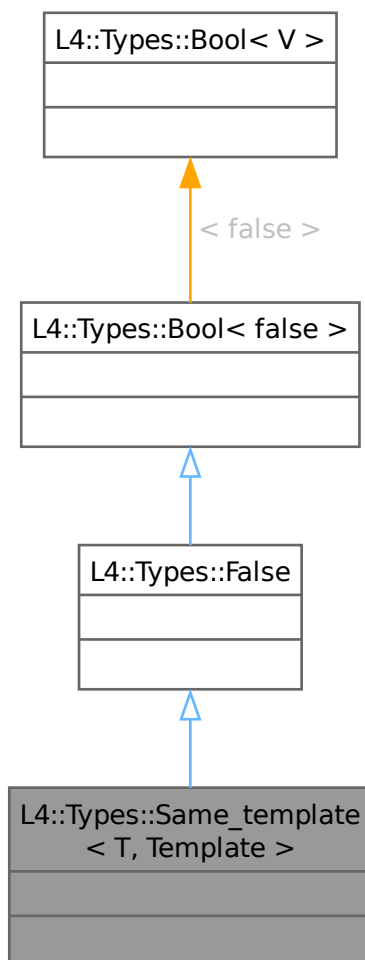
- [l4/sys/cxx/types](#)

15.235 **L4::Types::Same_template< T, Template > Struct Template Reference**

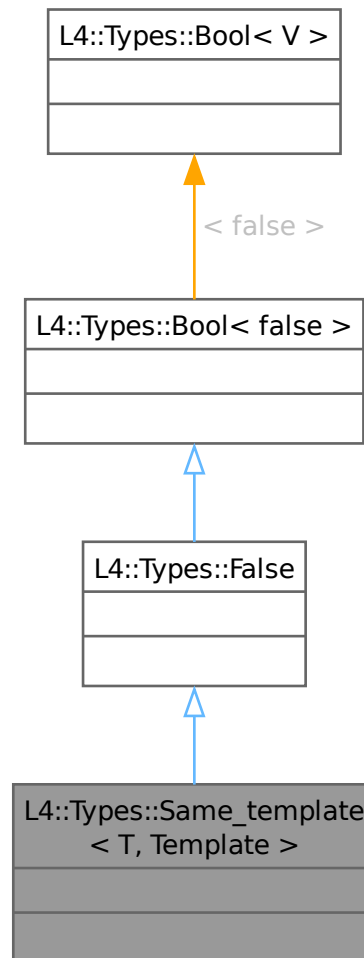
Check if a type T is an instantiation of a given template.

```
#include <types>
```

Inheritance diagram for L4::Types::Same_template< T, Template >:



Collaboration diagram for L4::Types::Same_template< T, Template >:



Additional Inherited Members

Public Types inherited from **L4::Types::Bool< false >**

- using **type**
The meta type itself.

15.235.1 Detailed Description

```
template<typename T, template< typename... > typename Template>
struct L4::Types::Same_template< T, Template >
```

Check if a type T is an instantiation of a given template.

Definition at line 394 of file [types](#).

The documentation for this struct was generated from the following file:

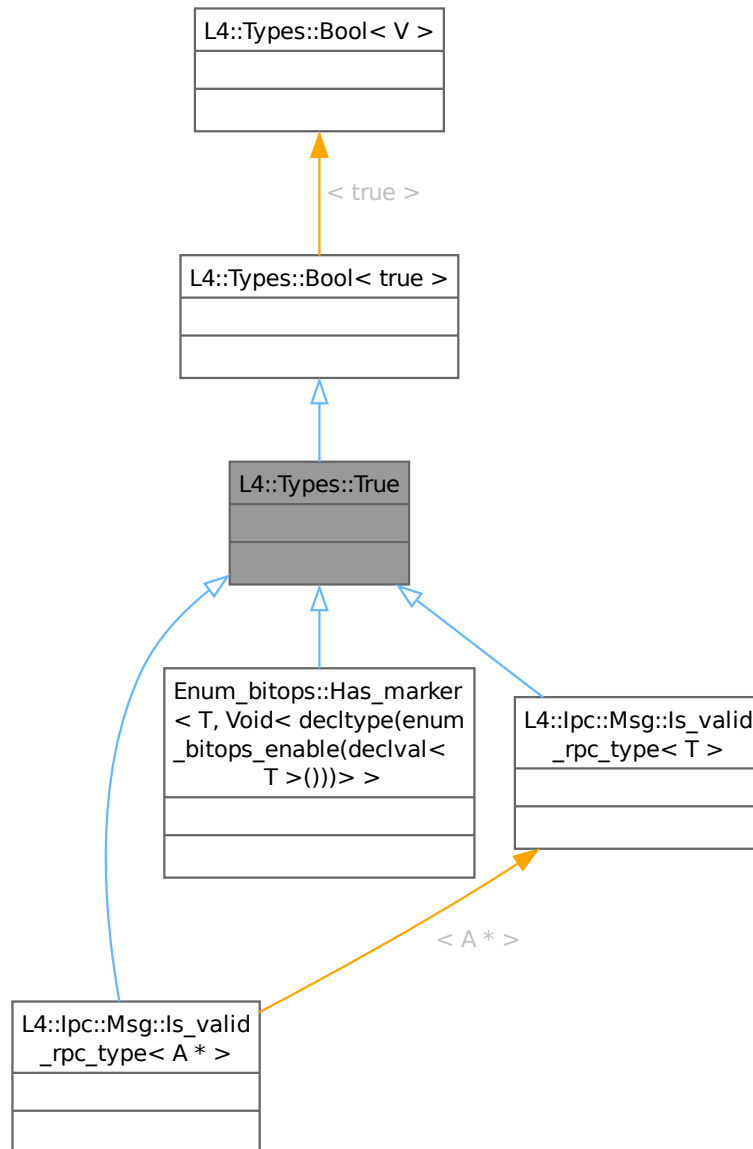
- [l4/sys/cxx/types](#)

15.236 L4::Types::True Struct Reference

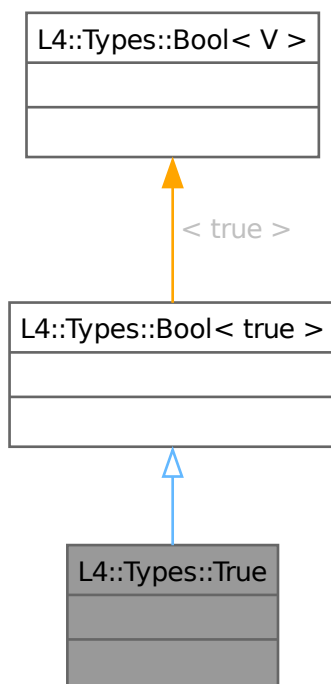
[True](#) meta value.

```
#include <types>
```

Inheritance diagram for L4::Types::True:



Collaboration diagram for L4::Types::True:



Additional Inherited Members

Public Types inherited from `L4::Types::Bool< true >`

- using **type**
The meta type itself.

15.236.1 Detailed Description

`True` meta value.

Definition at line 324 of file `types`.

The documentation for this struct was generated from the following file:

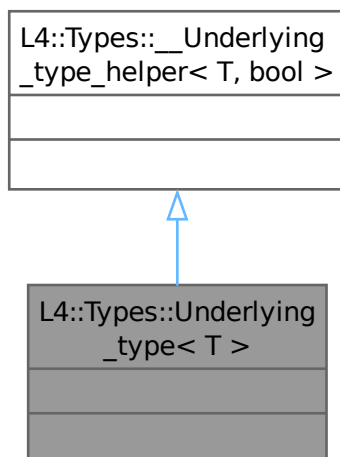
- `l4/sys/cxx/types`

15.237 L4::Types::__Underlying_type< T > Struct Template Reference

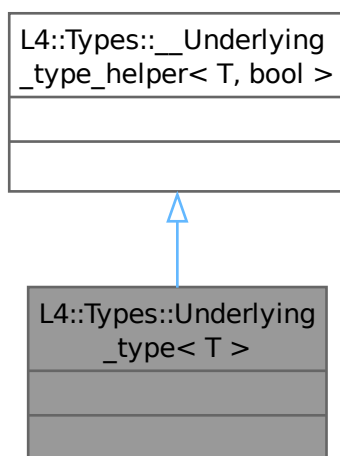
Get an underlying type of an enumeration type.

```
#include <types>
```

Inheritance diagram for L4::Types::__Underlying_type< T >:



Collaboration diagram for L4::Types::__Underlying_type< T >:



15.237.1 Detailed Description

```
template<typename T>  
struct L4::Types::Underlying_type< T >
```

Get an underlying type of an enumeration type.

Definition at line 368 of file [types](#).

The documentation for this struct was generated from the following file:

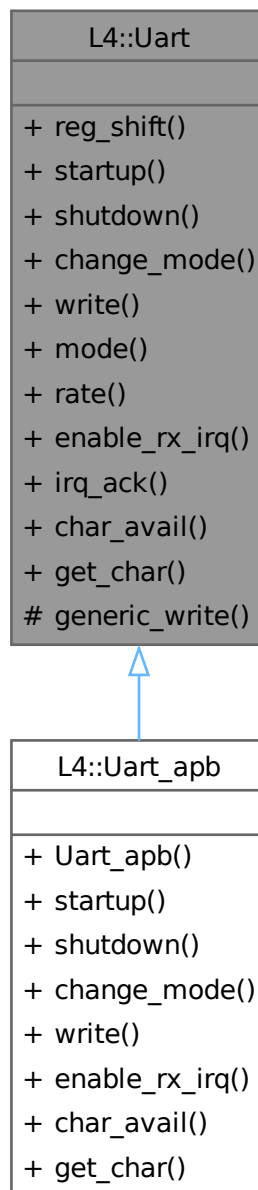
- [l4/sys/cxx/types](#)

15.238 L4::Uart Class Reference

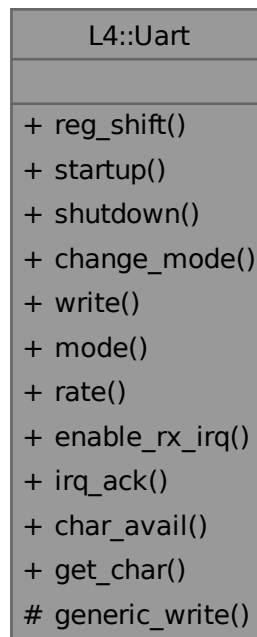
[Uart](#) driver abstraction.

```
#include <uart_base.h>
```

Inheritance diagram for L4::Uart:



Collaboration diagram for L4::Uart:



Public Member Functions

- virtual bool [reg_shift](#) (unsigned char *shift) const
Provide the reg_shift value for the UART's registers.
- virtual bool [startup](#) (lo_register_block const *regs)=0
Start the UART driver.
- virtual void [shutdown](#) ()=0
Terminate the UART driver.
- virtual bool [change_mode](#) (Transfer_mode m, Baud_rate r)=0
Set certain parameters of the UART.
- virtual int [write](#) (char const *s, unsigned long count, bool blocking=true) const =0
Transmit a number of characters.
- Transfer_mode [mode](#) () const
Return the transfer mode.
- Baud_rate [rate](#) () const
Return the baud rate.
- virtual bool [enable_rx_irq](#) (bool=true)
Enable the receive IRQ.
- virtual void [irq_ack](#) ()
Acknowledge a received interrupt.
- virtual int [char_avail](#) () const =0
Check if there is at least one character available for reading from the UART.
- virtual int [get_char](#) (bool blocking=true) const =0
Read a character from the UART.

Protected Member Functions

- `template<typename Uart_driver, bool Timeout_guard = true>`
`int generic_write (char const *s, unsigned long count, bool blocking=true) const`
Internal function transmitting each character one-after-another and finally waiting that the transmission did actually finish.

15.238.1 Detailed Description

[Uart](#) driver abstraction.

Definition at line 22 of file [uart_base.h](#).

15.238.2 Member Function Documentation

15.238.2.1 `change_mode()`

```
virtual bool L4::Uart::change_mode (  
    Transfer_mode m,  
    Baud_rate r) [pure virtual]
```

Set certain parameters of the UART.

Parameters

| | |
|----------|-------------------------------------|
| <i>m</i> | UART mode. Depends on the hardware. |
| <i>r</i> | Baud rate. |

Return values

| | |
|--------------|---|
| <i>true</i> | Mode setting succeeded (or was not performed at all). |
| <i>false</i> | Mode setting failed for some reason. |

Note

Some drivers don't perform any mode setting at all and just return true.

Implemented in [L4::Uart_apb](#).

15.238.2.2 `char_avail()`

```
virtual int L4::Uart::char_avail () const [pure virtual]
```

Check if there is at least one character available for reading from the UART.

Returns

0 if there is no character available for reading, !=0 otherwise.

Implemented in [L4::Uart_apb](#).

15.238.2.3 enable_rx_irq()

```
virtual bool L4::Uart::enable_rx_irq (
    bool = true) [inline], [virtual]
```

Enable the receive IRQ.

Return values

| | |
|--------------|--|
| <i>true</i> | The RX IRQ was successfully enabled / disabled. |
| <i>false</i> | The RX IRQ couldn't be enabled / disabled. The driver does not support this operation. |

Reimplemented in [L4::Uart_apb](#).

Definition at line 115 of file [uart_base.h](#).

15.238.2.4 generic_write()

```
template<typename Uart_driver, bool Timeout_guard = true>
int L4::Uart::generic_write (
    char const * s,
    unsigned long count,
    bool blocking = true) const [inline], [protected]
```

Internal function transmitting each character one-after-another and finally waiting that the transmission did actually finish.

Parameters

| | |
|-----------------|--|
| <i>s</i> | Buffer containing the characters. |
| <i>count</i> | The number of characters to transmit. |
| <i>blocking</i> | If true, wait until there is space in the transmit buffer and also wait until every character was successful transmitted. Otherwise do not wait. |

Returns

The number of successful written characters.

Definition at line 155 of file [uart_base.h](#).

References [L4::Poll_timeout_counter::test\(\)](#).

Here is the call graph for this function:



15.238.2.5 get_char()

```
virtual int L4::Uart::get_char (
    bool blocking = true) const [pure virtual]
```

Read a character from the UART.

Parameters

| | |
|-----------------|--|
| <i>blocking</i> | If true, wait until a character is available for reading. Otherwise do not wait and just return -1 if no character is available. |
|-----------------|--|

Returns

The actual character read from the UART.

Implemented in [L4::Uart_apb](#).

15.238.2.6 mode()

```
Transfer_mode L4::Uart::mode () const [inline]
```

Return the transfer mode.

Returns

The transfer mode.

Definition at line [97](#) of file [uart_base.h](#).

15.238.2.7 rate()

```
Baud_rate L4::Uart::rate () const [inline]
```

Return the baud rate.

Returns

The baud rate.

Definition at line [104](#) of file [uart_base.h](#).

15.238.2.8 reg_shift()

```
virtual bool L4::Uart::reg_shift (
    unsigned char * shift) const [inline], [virtual]
```

Provide the reg_shift value for the UART's registers.

Parameters

| | | |
|-----|--------------|---|
| out | <i>shift</i> | Register shift to be used for the UART's registers. |
|-----|--------------|---|

Return values

| | |
|--------------|-----------------------------------|
| <i>false</i> | The output argument is not valid. |
| <i>true</i> | The output argument is valid. |

Definition at line 47 of file [uart_base.h](#).

15.238.2.9 shutdown()

```
virtual void L4::Uart::shutdown () [pure virtual]
```

Terminate the UART driver.

This includes disabling of interrupts.

Implemented in [L4::Uart_apb](#).

15.238.2.10 startup()

```
virtual bool L4::Uart::startup (
    Io_register_block const * regs) [pure virtual]
```

Start the UART driver.

Parameters

| | |
|-------------|--------------------------------|
| <i>regs</i> | IO register block of the UART. |
|-------------|--------------------------------|

Return values

| | |
|--------------|--------------------|
| <i>true</i> | Startup succeeded. |
| <i>false</i> | Startup failed. |

Implemented in [L4::Uart_apb](#).

15.238.2.11 write()

```
virtual int L4::Uart::write (  
    char const * s,  
    unsigned long count,  
    bool blocking = true) const [pure virtual]
```

Transmit a number of characters.

Parameters

| | |
|-----------------|--|
| <i>s</i> | Buffer containing the characters. |
| <i>count</i> | Number of characters to transmit. |
| <i>blocking</i> | If true, wait until there is space in the transmit buffer and also wait until every character was successful transmitted. Otherwise do not wait. |

Returns

The number of successfully written characters.

Implemented in [L4::Uart_apb](#).

The documentation for this class was generated from the following file:

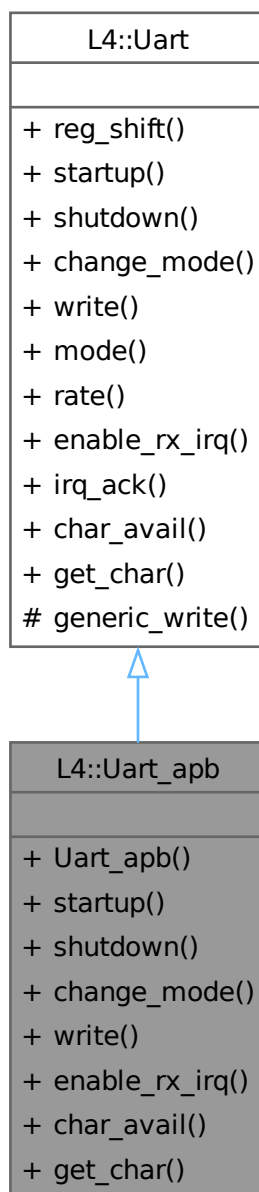
- pkg/drivers-frst/uart/include/uart_base.h

15.239 L4::Uart_apb Class Reference

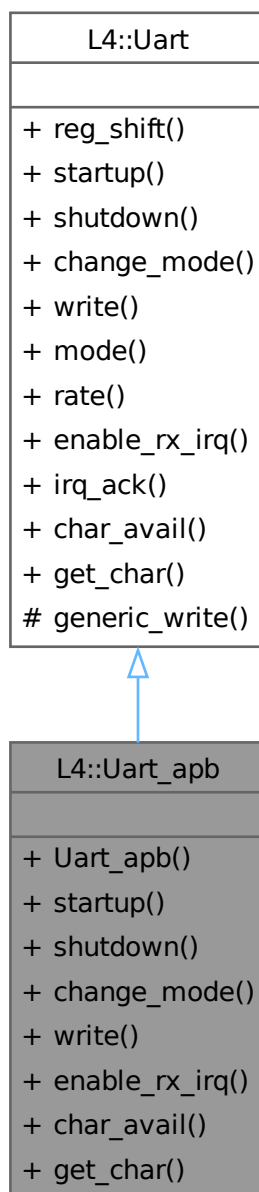
Driver for the Advanced Peripheral Bus (APB) UART from the Cortex-M System Design Kit (CMSDK).

```
#include <uart_apb.h>
```

Inheritance diagram for L4::Uart_apb:



Collaboration diagram for L4::Uart_apb:



Public Member Functions

- **Uart_apb** (unsigned freq)
freq == 0 means unknown and don't change baud rate
- bool **startup** (lo_register_block const *) override
Start the UART driver.
- void **shutdown** () override
Terminate the UART driver.

- bool `change_mode` (Transfer_mode m, Baud_rate r) override
Set certain parameters of the UART.
- int `write` (char const *s, unsigned long count, bool blocking=true) const override
Transmit a number of characters.
- bool `enable_rx_irq` (bool enable) override
Enable the receive IRQ.
- int `char_avail` () const override
Check if there is at least one character available for reading from the UART.
- int `get_char` (bool blocking=true) const override
Read a character from the UART.

Public Member Functions inherited from `L4::Uart`

- virtual bool `reg_shift` (unsigned char *shift) const
Provide the reg_shift value for the UART's registers.
- Transfer_mode `mode` () const
Return the transfer mode.
- Baud_rate `rate` () const
Return the baud rate.
- virtual void `irq_ack` ()
Acknowledge a received interrupt.

Additional Inherited Members

Protected Member Functions inherited from `L4::Uart`

- template<typename Uart_driver, bool Timeout_guard = true>
int `generic_write` (char const *s, unsigned long count, bool blocking=true) const
Internal function transmitting each character one-after-another and finally waiting that the transmission did actually finish.

15.239.1 Detailed Description

Driver for the Advanced Peripheral Bus (APB) UART from the Cortex-M System Design Kit (CMSDK).

Definition at line 17 of file `uart_apb.h`.

15.239.2 Member Function Documentation

15.239.2.1 change_mode()

```
bool L4::Uart_apb::change_mode (
    Transfer_mode m,
    Baud_rate r) [override], [virtual]
```

Set certain parameters of the UART.

Parameters

| | |
|----------|-------------------------------------|
| <i>m</i> | UART mode. Depends on the hardware. |
| <i>r</i> | Baud rate. |

Return values

| | |
|--------------|---|
| <i>true</i> | Mode setting succeeded (or was not performed at all). |
| <i>false</i> | Mode setting failed for some reason. |

Note

Some drivers don't perform any mode setting at all and just return true.

Implements [L4::Uart](#).

15.239.2.2 char_avail()

```
int L4::Uart_apb::char_avail () const [override], [virtual]
```

Check if there is at least one character available for reading from the UART.

Returns

0 if there is no character available for reading, !=0 otherwise.

Implements [L4::Uart](#).

15.239.2.3 enable_rx_irq()

```
bool L4::Uart_apb::enable_rx_irq (  
    bool ) [override], [virtual]
```

Enable the receive IRQ.

Return values

| | |
|--------------|--|
| <i>true</i> | The RX IRQ was successfully enabled / disabled. |
| <i>false</i> | The RX IRQ couldn't be enabled / disabled. The driver does not support this operation. |

Reimplemented from [L4::Uart](#).

15.239.2.4 get_char()

```
int L4::Uart_apb::get_char (  
    bool blocking = true) const [override], [virtual]
```

Read a character from the UART.

Parameters

| | |
|-----------------|--|
| <i>blocking</i> | If true, wait until a character is available for reading. Otherwise do not wait and just return -1 if no character is available. |
|-----------------|--|

Returns

The actual character read from the UART.

Implements [L4::Uart](#).

15.239.2.5 shutdown()

```
void L4::Uart_apb::shutdown () [override], [virtual]
```

Terminate the UART driver.

This includes disabling of interrupts.

Implements [L4::Uart](#).

15.239.2.6 startup()

```
bool L4::Uart_apb::startup (
    Io_register_block const * regs) [override], [virtual]
```

Start the UART driver.

Parameters

| | |
|-------------|--------------------------------|
| <i>regs</i> | IO register block of the UART. |
|-------------|--------------------------------|

Return values

| | |
|--------------|--------------------|
| <i>true</i> | Startup succeeded. |
| <i>false</i> | Startup failed. |

Implements [L4::Uart](#).

15.239.2.7 write()

```
int L4::Uart_apb::write (
    char const * s,
    unsigned long count,
    bool blocking = true) const [override], [virtual]
```

Transmit a number of characters.

Parameters

| | |
|-----------------|--|
| <i>s</i> | Buffer containing the characters. |
| <i>count</i> | Number of characters to transmit. |
| <i>blocking</i> | If true, wait until there is space in the transmit buffer and also wait until every character was successful transmitted. Otherwise do not wait. |

Returns

The number of successfully written characters.

Implements [L4::Uart](#).

The documentation for this class was generated from the following file:

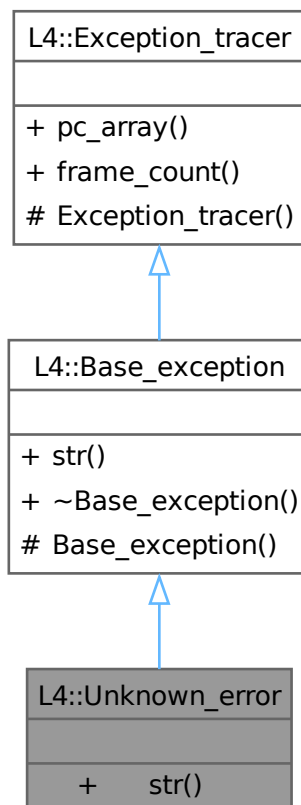
- pkg/drivers-frst/uart/include/uart_apb.h

15.240 L4::Unknown_error Class Reference

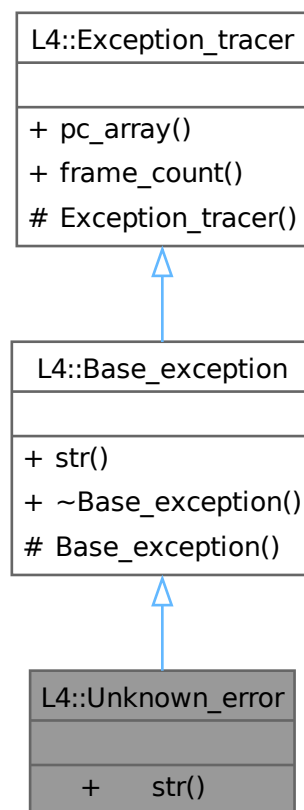
[Exception](#) for an unknown condition.

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Unknown_error:



Collaboration diagram for L4::Unknown_error:



Public Member Functions

- `char const * str ()` `const noexcept` override
Return a human readable string for the exception.

Public Member Functions inherited from [L4::Base_exception](#)

- `virtual ~Base_exception ()` `noexcept`
Destruction.

Public Member Functions inherited from [L4::Exception_tracer](#)

- `void const *const * pc_array ()` `const noexcept`
Get the array containing the call trace.
- `int frame_count ()` `const noexcept`
Get the number of entries that are valid in the call trace.

Additional Inherited Members

Protected Member Functions inherited from [L4::Base_exception](#)

- **Base_exception** () noexcept
Create a base exception.

Protected Member Functions inherited from [L4::Exception_tracer](#)

- **Exception_tracer** () noexcept
Create a back trace.

15.240.1 Detailed Description

[Exception](#) for an unknown condition.

This error is usually used when a server returns an unknown return state to the client, this may indicate incompatible messages used by the client and the server.

Definition at line [208](#) of file [exceptions](#).

The documentation for this class was generated from the following file:

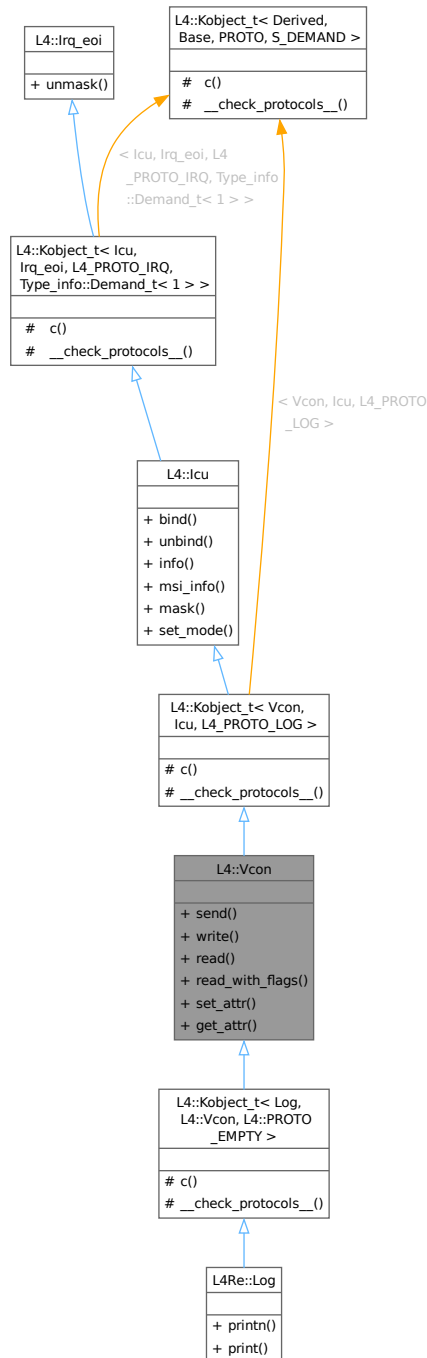
- [l4/cxx/exceptions](#)

15.241 L4::Vcon Class Reference

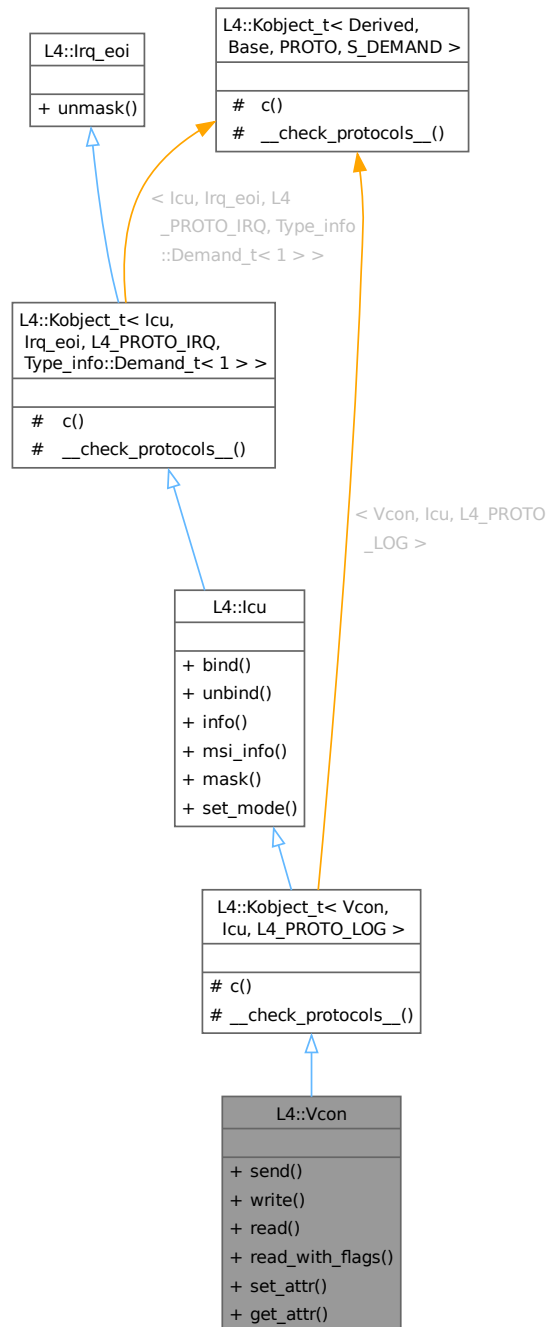
C++ [L4 Vcon](#) interface, see [Virtual Console](#) for the C interface.

```
#include <vcon>
```

Inheritance diagram for L4::Vcon:



Collaboration diagram for L4::Vcon:



Public Member Functions

- `l4_msgtag_t send` (char const *buf, unsigned size, `l4_utcb_t *utcb=l4_utcb()`) const noexcept
Send data to *this* virtual console.
- `long write` (char const *buf, unsigned size, `l4_utcb_t *utcb=l4_utcb()`) const noexcept
Write data to *this* virtual console.
- `int read` (char *buf, unsigned size, `l4_utcb_t *utcb=l4_utcb()`) const noexcept

- *Read data from `this` virtual console.*
 • `int read_with_flags (char *buf, unsigned size, l4_utcb_t *utcb=l4_utcb\(\)) const noexcept`
Read data from `this` virtual console which also returns flags.
- `l4_msgtag_t set_attr (l4_vcon_attr_t const *attr, l4_utcb_t *utcb=l4_utcb\(\)) const noexcept`
Set the attributes of `this` virtual console.
- `l4_msgtag_t get_attr (l4_vcon_attr_t *attr, l4_utcb_t *utcb=l4_utcb\(\)) const noexcept`
Get attributes of `this` virtual console.

Public Member Functions inherited from [L4::Icu](#)

- `l4_msgtag_t bind (unsigned irqnum, L4::Cap< Triggerable > irq, l4_utcb_t *utcb=l4_utcb\(\)) noexcept`
Bind an interrupt line of an interrupt controller to an interrupt object.
- `l4_msgtag_t unbind (unsigned irqnum, L4::Cap< Triggerable > irq, l4_utcb_t *utcb=l4_utcb\(\)) noexcept`
Remove binding of an interrupt line from the interrupt controller object.
- `l4_msgtag_t info (l4_icu_info_t *info, l4_utcb_t *utcb=l4_utcb\(\)) noexcept`
Get information about the ICU features.
- `l4_msgtag_t msi_info (l4_umword_t irqnum, l4_uint64_t source, l4_icu_msi_info_t *msi_info)`
Get MSI info about IRQ.
- `l4_msgtag_t mask (unsigned irqnum, l4_umword_t *label=0, l4_timeout_t to=L4_IPC_NEVER, l4_utcb_t *utcb=l4_utcb\(\)) noexcept`
Mask an IRQ line.
- `l4_msgtag_t set_mode (unsigned irqnum, l4_umword_t mode, l4_utcb_t *utcb=l4_utcb\(\)) noexcept`
Set interrupt mode.

Public Member Functions inherited from [L4::Irq_eoi](#)

- `l4_msgtag_t unmask (unsigned irqnum, l4_umword_t *label=0, l4_timeout_t to=L4_IPC_NEVER, l4_utcb_t *utcb=l4_utcb\(\)) noexcept`
Unmask the given interrupt line.

Additional Inherited Members

Protected Types inherited from [L4::Kobject_t](#)< [Vcon](#), [Icu](#), [L4_PROTO_LOG](#) >

- typedef [Vcon](#) Class
The target interface type (inheriting from [Kobject_t](#)).
- typedef `Typeid::Iface< PROTO, Vcon > __iface`
The interface description for the derived class.
- typedef `Typeid::Merge_list< Typeid::Iface_list< __iface >, typename Icu::__iface_list > __iface_list`
The list of all RPC interfaces provided directly or through inheritance.

Protected Types inherited from [L4::Kobject_t](#)< [Icu](#), [Irq_eoi](#), [L4_PROTO_IRQ](#), [Type_info::Demand_t](#)< 1 > >

- typedef [Icu](#) Class
The target interface type (inheriting from [Kobject_t](#)).
- typedef `Typeid::Iface< PROTO, Icu > __iface`
The interface description for the derived class.
- typedef `Typeid::Merge_list< Typeid::Iface_list< __iface >, typename Irq_eoi::__iface_list > __iface_list`
The list of all RPC interfaces provided directly or through inheritance.

Protected Member Functions inherited from [L4::Kobject_t< Vcon, lcu, L4_PROTO_LOG >](#)

- [L4::Cap< Class > c \(\)](#) const noexcept
Get the capability to ourselves.

Protected Member Functions inherited from [L4::Kobject_t< lcu, Irq_eoi, L4_PROTO_IRQ, Type_info::Demand_t< 1 > >](#)

- [L4::Cap< Class > c \(\)](#) const noexcept
Get the capability to ourselves.

Static Protected Member Functions inherited from [L4::Kobject_t< Vcon, lcu, L4_PROTO_LOG >](#)

- static void [__check_protocols__ \(\)](#) noexcept
Helper to check for protocol conflicts.

Static Protected Member Functions inherited from [L4::Kobject_t< lcu, Irq_eoi, L4_PROTO_IRQ, Type_info::Demand_t< 1 > >](#)

- static void [__check_protocols__ \(\)](#) noexcept
Helper to check for protocol conflicts.

15.241.1 Detailed Description

C++ [L4 Vcon](#) interface, see [Virtual Console](#) for the C interface.

[L4::Vcon](#) is a virtual console for simple character-based input and output. The interrupt for read events is provided by the virtual key interrupt.

The [Vcon](#) interface inherits from [L4::lcu](#) and [L4::Irq_eoi](#) for managing the virtual key interrupt which, in contrast to hardware IRQs, implements a limited functionality:

- Only IRQ line 0 is supported, no MSI vectors.
- The IRQ is edge-triggered and the IRQ mode cannot be changed.
- As the IRQ is edge-triggered, it does not have to be explicitly unmasked.

A server implementing the virtual console protocol has a queue for input events. When the first input event is added to the empty queue, the virtual key interrupt is triggered. Further events are added to the queue without generating further interrupts. The queue is emptied when a client reads all queued input events.

Include File

```
#include <l4/sys/vcon>
```

See the [Virtual Console](#) for the C interface.

Definition at line 45 of file [vcon](#).

15.241.2 Member Function Documentation

15.241.2.1 get_attr()

```
l4_msgtag_t L4::Vcon::get_attr (
    l4_vcon_attr_t * attr,
    l4_utcb_t * utcb = l4_utcb()) const [inline], [noexcept]
```

Get attributes of this virtual console.

Parameters

| | | |
|-----|-------------|--|
| out | <i>attr</i> | Attribute structure. Contains the attributes after a successful call of this function. |
| | <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb . |

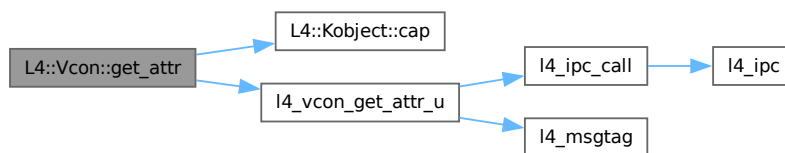
Returns

Syscall return tag.

Definition at line 151 of file [vcon](#).

References [L4::Kobject::cap\(\)](#), and [l4_vcon_get_attr_u\(\)](#).

Here is the call graph for this function:



15.241.2.2 read()

```
int L4::Vcon::read (
    char * buf,
    unsigned size,
    l4_utcb_t * utcb = l4_utcb()) const [inline], [noexcept]
```

Read data from this virtual console.

Parameters

| | | |
|-----|-------------|-----------------------------------|
| out | <i>buf</i> | Pointer to data buffer. |
| | <i>size</i> | Size of the data buffer in bytes. |

| | | |
|--|-------------|--|
| | <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb . |
|--|-------------|--|

Return values

| | |
|------------------------|--|
| <code>-L4_EPERM</code> | Insufficient permissions; see precondition. |
| <code>>size</code> | More bytes to read, <code>size</code> bytes are in the buffer <code>buf</code> . |
| <code><=size</code> | Number of bytes read. |

Precondition

The invoked [Vcon](#) capability must have the permission [L4_CAP_FPAGE_W](#).

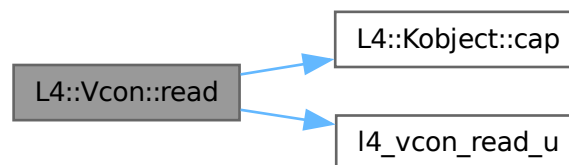
Note

Size must not exceed [L4_VCON_READ_SIZE](#).

Definition at line [98](#) of file [vcon](#).

References [L4::Kobject::cap\(\)](#), and [l4_vcon_read_u\(\)](#).

Here is the call graph for this function:



15.241.2.3 read_with_flags()

```

int L4::Vcon::read_with_flags (
    char * buf,
    unsigned size,
    l4_utcb_t * utcb = l4_utcb()) const [inline], [noexcept]

```

Read data from `this` virtual console which also returns flags.

Parameters

| | | |
|-----|------------|-------------------------|
| out | <i>buf</i> | Pointer to data buffer. |
|-----|------------|-------------------------|

| | |
|-------------|--|
| <i>size</i> | Size of the data buffer in bytes. |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb . |

Return values

| | |
|------------------------|--|
| <code>-L4_EPERM</code> | Insufficient permissions; see precondition. |
| <code>>size</code> | More bytes to read, <i>size</i> bytes are in the buffer <i>buf</i> . |
| <code><=size</code> | Number of bytes read. |

Precondition

The invoked [Vcon](#) capability must have the permission [L4_CAP_FPAGE_W](#).

If this function returns a positive value the caller can check the [L4_VCON_READ_STAT_BREAK](#) flag bit for a break condition. The bytes read can be obtained by masking the return value with [L4_VCON_READ_SIZE_MASK](#).

If a break condition is signaled, it is always the first event in the transmitted content, i.e. all characters supplied by this read call follow the break condition.

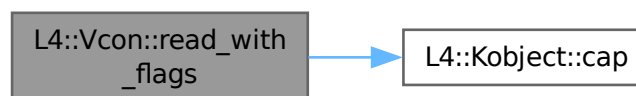
Note

Size must not exceed [L4_VCON_READ_SIZE](#).

Definition at line 125 of file [vcon](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



15.241.2.4 send()

```

l4_msgtag_t L4::Vcon::send (
    char const * buf,
    unsigned size,
    l4_utcb_t * utcb = l4_utcb()) const [inline], [noexcept]
  
```

Send data to this virtual console.

Parameters

| | |
|-------------|--|
| <i>buf</i> | Pointer to the data buffer. |
| <i>size</i> | Size of the data buffer in bytes. |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb . |

Returns

Syscall return tag

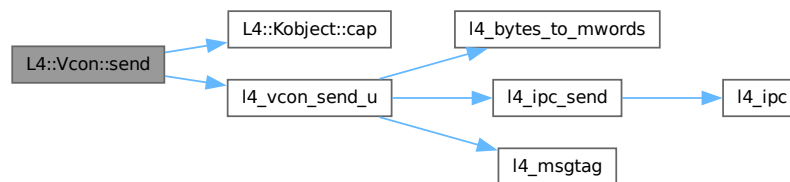
Note

Size must not exceed [L4_VCON_WRITE_SIZE](#), a proper value of the `size` parameter is NOT checked. Also, this function is a send only operation, this means there is no return value except for a failed send operation. Use [l4_ipc_error\(\)](#) to check for send errors, do not use [l4_error\(\)](#), as [l4_error\(\)](#) will always return an error.

Definition at line 65 of file [vcon](#).

References [L4::Kobject::cap\(\)](#), and [l4_vcon_send_u\(\)](#).

Here is the call graph for this function:

**15.241.2.5 set_attr()**

```

l4_msgtag_t L4::Vcon::set_attr (
    l4_vcon_attr_t const * attr,
    l4_utcb_t * utcb = l4_utcb()) const [inline], [noexcept]

```

Set the attributes of `this` virtual console.

Parameters

| | |
|-------------|--|
| <i>attr</i> | Attribute structure with the attributes for the virtual console. |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb . |

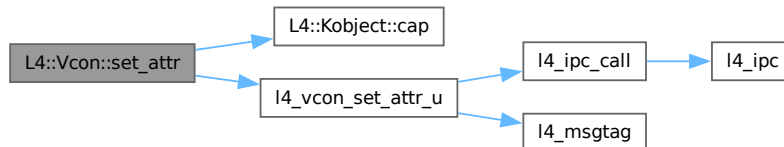
Returns

Syscall return tag.

Definition at line 138 of file [vcon](#).

References [L4::Kobject::cap\(\)](#), and [l4_vcon_set_attr_u\(\)](#).

Here is the call graph for this function:

**15.241.2.6 write()**

```

long L4::Vcon::write (
    char const * buf,
    unsigned size,
    l4_utcb_t * utcb = l4_utcb()) const [inline], [noexcept]

```

Write data to this virtual console.

Parameters

| | |
|-------------|--|
| <i>buf</i> | Pointer to the data buffer. |
| <i>size</i> | Size of the data buffer in bytes. |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. Defaults to l4_utcb . |

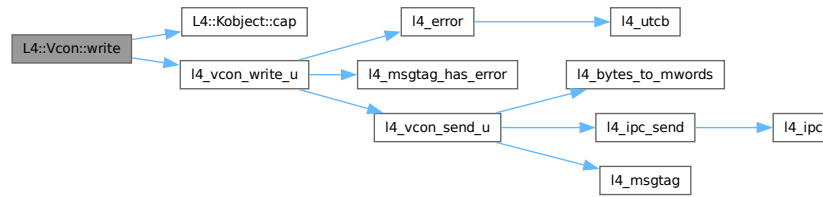
Return values

| | |
|---------------|---|
| <i><0</i> | Error. |
| <i>>=0</i> | Number of bytes written to the virtual console. |

Definition at line 79 of file [vcon](#).

References [L4::Kobject::cap\(\)](#), and [l4_vcon_write_u\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

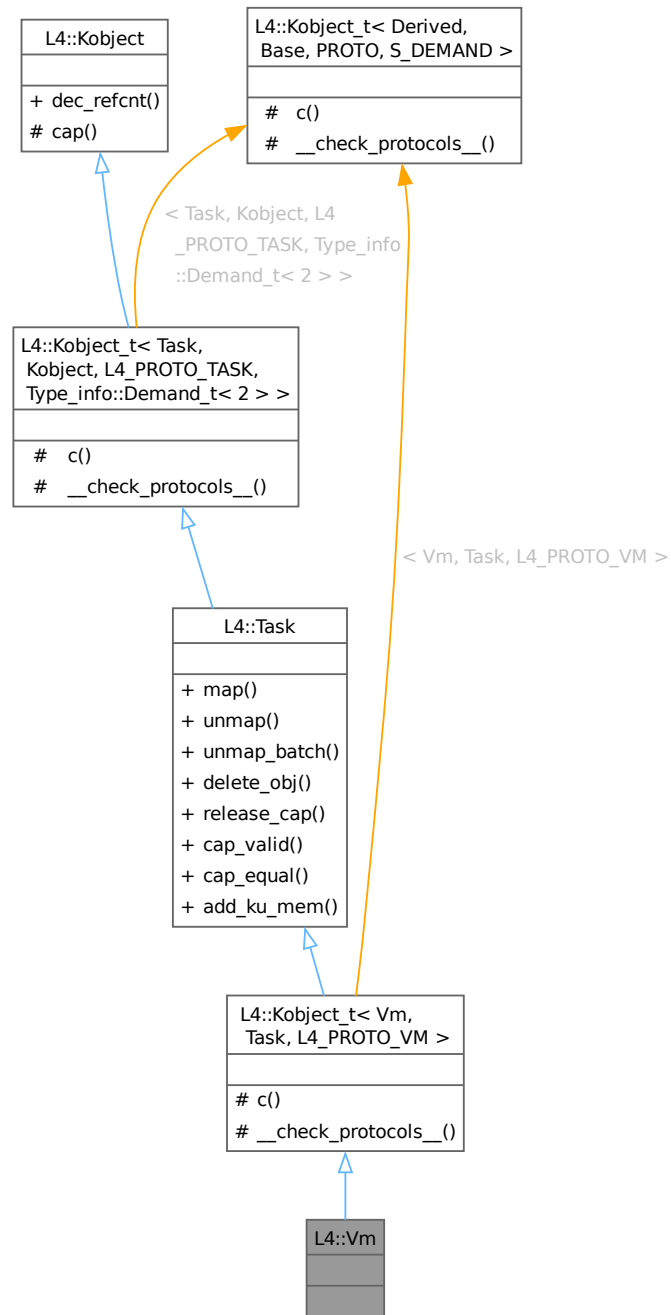
- [l4/sys/vcon](#)

15.242 L4::Vm Class Reference

Virtual machine host address space.

```
#include <vm>
```

Inheritance diagram for L4::Vm:



- Map resources available in the source task to a destination task.*

 - `l4_msgtag_t unmap (l4_fpage_t const &fpage, l4_umword_t map_mask, l4_utcb_t *utcb=l4_utcb()) noexcept`

Revoke rights from the task.
- `l4_msgtag_t unmap_batch (l4_fpage_t const *fpages, unsigned num_fpages, l4_umword_t map_mask, l4_utcb_t *utcb=l4_utcb()) noexcept`

Revoke rights from a task.
- `l4_msgtag_t delete_obj (L4::Cap< void > obj, l4_utcb_t *utcb=l4_utcb()) noexcept`

Release capability and delete object.
- `l4_msgtag_t release_cap (L4::Cap< void > cap, l4_utcb_t *utcb=l4_utcb()) noexcept`

Release object capability.
- `l4_msgtag_t cap_valid (Cap< void > const &cap, l4_utcb_t *utcb=l4_utcb()) noexcept`

Check whether a capability is present (refers to an object).
- `l4_msgtag_t cap_equal (Cap< void > const &cap_a, Cap< void > const &cap_b, l4_utcb_t *utcb=l4_utcb()) noexcept`

Test whether two capabilities point to the same object with the same permissions (only considering selected permissions).
- `l4_msgtag_t add_ku_mem (l4_fpage_t *fpage, l4_utcb_t *utcb=l4_utcb()) noexcept`

Add kernel-user memory.

Public Member Functions inherited from L4::Kobject

- `l4_msgtag_t dec_refcnt (l4_mword_t diff, l4_utcb_t *utcb=l4_utcb())`

Decrement the in kernel reference counter for the object.

Protected Types inherited from L4::Kobject_t< Vm, Task, L4_PROTO_VM >

- typedef **Vm Class**
- The target interface type (inheriting from Kobject_t).*
- typedef Typeid::Iface< PROTO, Vm > **__Iface**
- The interface description for the derived class.*
- typedef Typeid::Merge_list< Typeid::Iface_list< __Iface >, typename Task::__Iface_list > **__Iface_list**
- The list of all RPC interfaces provided directly or through inheritance.*

Protected Types inherited from L4::Kobject_t< Task, Kobject, L4_PROTO_TASK, Type_info::Demand_t< 2 > >

- typedef **Task Class**
- The target interface type (inheriting from Kobject_t).*
- typedef Typeid::Iface< PROTO, Task > **__Iface**
- The interface description for the derived class.*
- typedef Typeid::Merge_list< Typeid::Iface_list< __Iface >, typename Kobject::__Iface_list > **__Iface_list**
- The list of all RPC interfaces provided directly or through inheritance.*

Protected Member Functions inherited from L4::Kobject_t< Vm, Task, L4_PROTO_VM >

- `L4::Cap< Class > c () const noexcept`

Get the capability to ourselves.

Protected Member Functions inherited from

[L4::Kobject_t](#)< [Task](#), [Kobject](#), [L4_PROTO_TASK](#), [Type_info::Demand_t](#)< 2 > >

- [L4::Cap](#)< [Class](#) > [c](#) () const noexcept

Get the capability to ourselves.

Protected Member Functions inherited from [L4::Kobject](#)

- [l4_cap_idx_t](#) [cap](#) () const noexcept

Return capability selector.

Static Protected Member Functions inherited from

[L4::Kobject_t](#)< [Vm](#), [Task](#), [L4_PROTO_VM](#) >

- static void [__check_protocols__](#) () noexcept

Helper to check for protocol conflicts.

Static Protected Member Functions inherited from

[L4::Kobject_t](#)< [Task](#), [Kobject](#), [L4_PROTO_TASK](#), [Type_info::Demand_t](#)< 2 > >

- static void [__check_protocols__](#) () noexcept

Helper to check for protocol conflicts.

15.242.1 Detailed Description

Virtual machine host address space.

[L4::Vm](#) is a specialisation of [L4::Task](#), used for virtual machines. The microkernel employs an appropriate page-table format for hosting VMs, such as ePT on VT-x. On Arm, it offers a call to make the virtual GICC area available to the VM.

Definition at line 30 of file [vm](#).

The documentation for this class was generated from the following file:

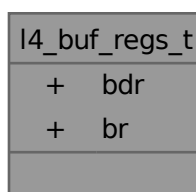
- [l4/sys/vm](#)

15.243 l4_buf_regs_t Struct Reference

Encapsulation of the buffer-registers block in the UTCB.

```
#include <utcb.h>
```

Collaboration diagram for [l4_buf_regs_t](#):



Data Fields

- [l4_umword_t](#) bdr
Buffer descriptor.
- [l4_umword_t](#) br [L4_UTCB_GENERIC_BUFFERS_SIZE]
Buffer registers.

15.243.1 Detailed Description

Encapsulation of the buffer-registers block in the UTCB.

Definition at line 147 of file [utcb.h](#).

15.243.2 Field Documentation

15.243.2.1 bdr

```
l4\_umword\_t l4_buf_regs_t::bdr
```

[Buffer](#) descriptor.

See also

[l4_bdr\(\)](#)

Definition at line 150 of file [utcb.h](#).

Referenced by [l4util_ioport_map\(\)](#), [L4::lpc_svr::Br_manager_no_buffers::setup_wait\(\)](#), and [L4Re::Util::Br_manager::setup_wait\(\)](#).

The documentation for this struct was generated from the following file:

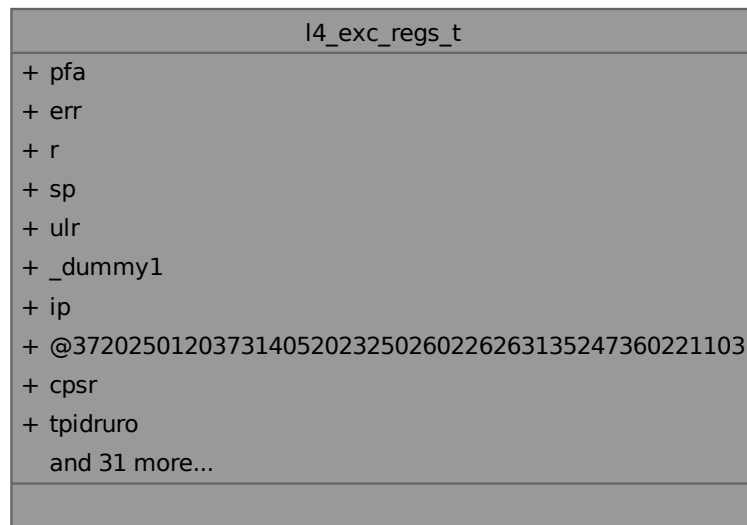
- [l4/sys/utcb.h](#)

15.244 l4_exc_regs_t Struct Reference

UTCB structure for exceptions.

```
#include <utcb.h>
```

Collaboration diagram for l4_exc_regs_t:



Data Fields

- [l4_umword_t](#) **pfa**
page fault address
- [l4_umword_t](#) **err**
error code
- [l4_umword_t](#) **r** [13]
registers
- [l4_umword_t](#) **sp**
stack pointer
- [l4_umword_t](#) **ulr**
ulr
- [l4_umword_t](#) **_dummy1**
dummy
- union {
};

aliases for PC
- [l4_umword_t](#) **cpsr**
cpsr
- [l4_umword_t](#) **tpidruro**
Thread-ID register.
- [l4_umword_t](#) **tpidrurw**
Thread-ID register.
- [l4_umword_t](#) **r15**
r15
- [l4_umword_t](#) **r14**

- r14*
 - [l4_umword_t](#) **r13**
- r13*
 - [l4_umword_t](#) **r12**
- r12*
 - [l4_umword_t](#) **r11**
- r11*
 - [l4_umword_t](#) **r10**
- r10*
 - [l4_umword_t](#) **r9**
- r9*
 - [l4_umword_t](#) **r8**
- r8*
 - [l4_umword_t](#) **rdi**
- rdi*
 - [l4_umword_t](#) **rsi**
- rsi*
 - [l4_umword_t](#) **rbp**
- rbp*
 - [l4_umword_t](#) **rbx**
- rbx*
 - [l4_umword_t](#) **rdx**
- rdx*
 - [l4_umword_t](#) **rcx**
- rcx*
 - [l4_umword_t](#) **rax**
- rax*
 - [l4_umword_t](#) **trapno**
- trap number*
 - [l4_umword_t](#) **dummy1**
- dummy*
 - [l4_umword_t](#) **ss**
- stack segment register*
 - [l4_umword_t](#) **es**
- es register*
 - [l4_umword_t](#) **ds**
- ds register*
 - [l4_umword_t](#) **gs**
- gs register*
 - [l4_umword_t](#) **fs**
- fs register*
 - [l4_umword_t](#) **edi**
- edi register*
 - [l4_umword_t](#) **esi**
- esi register*
 - [l4_umword_t](#) **ebp**
- ebp register*
 - [l4_umword_t](#) **ebx**
- ebx register*
 - [l4_umword_t](#) **edx**
- edx register*

- [l4_umword_t ecx](#)
ecx register
- [l4_umword_t eax](#)
eax register

15.244.1 Detailed Description

UTCB structure for exceptions.

Examples

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), and [examples/sys/start-with-exc/main.c](#).

Definition at line 27 of file [utcb.h](#).

15.244.2 Field Documentation

15.244.2.1 flags

```
l4\_umword\_t l4_exc_regs_t::flags
```

rflags

eflags

Definition at line 37 of file [utcb.h](#).

15.244.2.2 ss

```
l4\_umword\_t l4_exc_regs_t::ss
```

stack segment register

ss register

Definition at line 53 of file [utcb.h](#).

The documentation for this struct was generated from the following files:

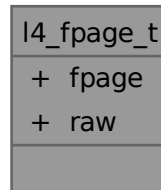
- [arm/l4/sys/arch/utcb.h](#)
- [arm64/l4/sys/arch/utcb.h](#)
- [amd64/l4/sys/arch/utcb.h](#)
- [x86/l4/sys/arch/utcb.h](#)
- [riscv/l4/sys/arch/utcb.h](#)

15.245 l4_fpage_t Union Reference

[L4](#) flexpage type.

```
#include <__l4_fpage.h>
```

Collaboration diagram for l4_fpage_t:



Data Fields

- [l4_umword_t](#) **fpage**
Raw value.
- [l4_umword_t](#) **raw**
Raw value.

15.245.1 Detailed Description

[L4](#) flexpage type.

Definition at line [76](#) of file [__l4_fpage.h](#).

The documentation for this union was generated from the following file:

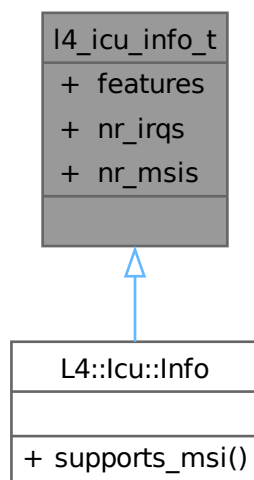
- l4/sys/__l4_fpage.h

15.246 l4_icu_info_t Struct Reference

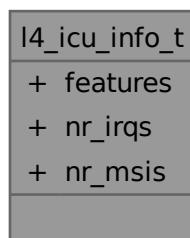
Info structure for an ICU.

```
#include <icu.h>
```

Inheritance diagram for `l4_icu_info_t`:



Collaboration diagram for `l4_icu_info_t`:



Data Fields

- unsigned `features`
Feature flags.
- unsigned `nr_irqs`
The number of IRQ lines supported by the ICU,.
- unsigned `nr_msis`
The number of MSI vectors supported by the ICU,.

15.246.1 Detailed Description

Info structure for an ICU.

This structure contains information about the features of an ICU.

See also

[l4_icu_info\(\)](#).

Definition at line 163 of file [icu.h](#).

15.246.2 Field Documentation

15.246.2.1 features

```
unsigned l4_icu_info_t::features
```

Feature flags.

If [L4_ICU_FLAG_MSI](#) is set the ICU supports MSIs.

Definition at line 170 of file [icu.h](#).

Referenced by [L4::Icu::Info::supports_msi\(\)](#).

The documentation for this struct was generated from the following file:

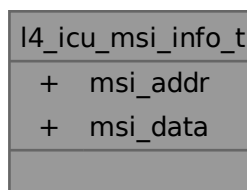
- [l4/sys/icu.h](#)

15.247 l4_icu_msi_info_t Struct Reference

Info to use for a specific MSI.

```
#include <icu.h>
```

Collaboration diagram for `l4_icu_msi_info_t`:



Data Fields

- [l4_uint64_t](#) **msi_addr**
Value to use as address when sending this MSI.
- [l4_uint32_t](#) **msi_data**
Value to use as data written to msi_addr, when sending this MSI.

15.247.1 Detailed Description

Info to use for a specific MSI.

Definition at line 184 of file [icu.h](#).

The documentation for this struct was generated from the following file:

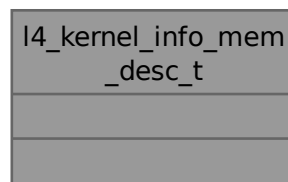
- [l4/sys/icu.h](#)

15.248 l4_kernel_info_mem_desc_t Struct Reference

Memory descriptor data structure.

```
#include <memdesc.h>
```

Collaboration diagram for `l4_kernel_info_mem_desc_t`:



15.248.1 Detailed Description

Memory descriptor data structure.

Note

This data type is opaque, and must be accessed by the accessor functions defined in this module.

Definition at line 77 of file [memdesc.h](#).

The documentation for this struct was generated from the following file:

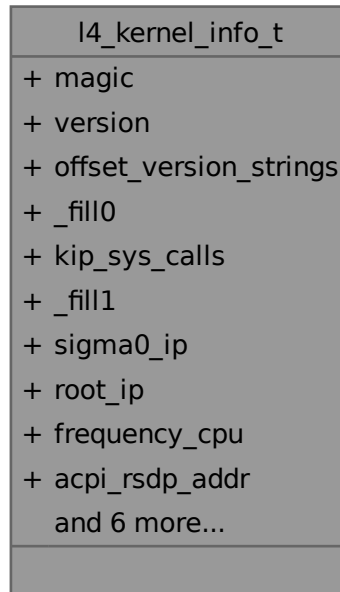
- [l4/sys/memdesc.h](#)

15.249 l4_kernel_info_t Struct Reference

[L4 Kernel Interface Page](#).

```
#include <kip.h>
```

Collaboration diagram for l4_kernel_info_t:



Data Fields

- [l4_uint32_t](#) **magic**
Kernel Info Page identifier ("L4μK").
- [l4_uint32_t](#) **version**
Kernel version.
- [l4_uint8_t](#) **offset_version_strings**
offset to version string
- [l4_uint8_t](#) **_fill0** [3]
reserved
- [l4_uint8_t](#) **kip_sys_calls**
pointer to system calls
- [l4_uint8_t](#) **_fill1** [2]
reserved
- [l4_uint64_t](#) **sigma0_ip**
Sigma0 instruction pointer.
- [l4_uint64_t](#) **root_ip**
Root task instruction pointer.
- [l4_uint64_t](#) **frequency_cpu**

- [l4_uint64_t](#) **acpi_rsdp_addr**
ACPI RSDP/XSDP.
- [l4_uint64_t](#) **dt_addr**
Device Tree.
- [l4_uint64_t](#) **user_ptr**
user_ptr
- [l4_uint32_t](#) **scheduler_granularity**
for rounding time slices
- [l4_uint32_t](#) **mem_descs**
memory descriptors relative to Kip
- [l4_uint32_t](#) **mem_descs_num**
number of memory descriptors
- [l4_uint64_t](#) **_res2** [2]
internal - spare space

15.249.1 Detailed Description

[L4 Kernel Interface Page.](#)

32-bit architecture may assume that the upper 32 bits of addresses is 0

Definition at line 36 of file [kip.h](#).

The documentation for this struct was generated from the following file:

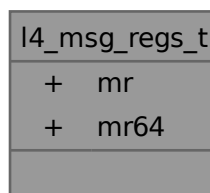
- [l4/sys/kip.h](#)

15.250 l4_msg_regs_t Union Reference

Encapsulation of the message-register block in the UTCB.

```
#include <utcb.h>
```

Collaboration diagram for `l4_msg_regs_t`:



Data Fields

- [l4_umword_t](#) **mr** [L4_UTCB_GENERIC_DATA_SIZE]
Message registers.
- [l4_uint64_t](#) **mr64** [L4_UTCB_GENERIC_DATA_SIZE/(sizeof([l4_uint64_t](#))/sizeof([l4_umword_t](#)))]
Message registers 64bit alias.

15.250.1 Detailed Description

Encapsulation of the message-register block in the UTCB.

Examples

[examples/sys/utcb-ipc/main.c](#).

Definition at line 132 of file [utcb.h](#).

The documentation for this union was generated from the following file:

- l4/sys/[utcb.h](#)

15.251 l4_msgtag_t Struct Reference

Message tag data structure.

```
#include <types.h>
```

Collaboration diagram for l4_msgtag_t:

| l4_msgtag_t |
|----------------------|
| + raw |
| + label() |
| + label() |
| + words() |
| + items() |
| + flags() |
| + is_page_fault() |
| + is_exception() |
| + is_sigma0() |
| + is_io_page_fault() |
| + has_error() |

Public Member Functions

- long **label** () const [L4_NOTHROW](#)
Get the protocol value.
- void **label** (long v) [L4_NOTHROW](#)
Set the protocol value.
- unsigned **words** () const [L4_NOTHROW](#)
Get the number of untyped words.
- unsigned **items** () const [L4_NOTHROW](#)
Get the number of typed items.
- unsigned **flags** () const [L4_NOTHROW](#)
Get the flags value.
- bool **is_page_fault** () const [L4_NOTHROW](#)
Test if protocol indicates page-fault protocol.
- bool **is_exception** () const [L4_NOTHROW](#)
Test if protocol indicates exception protocol.
- bool **is_sigma0** () const [L4_NOTHROW](#)
Test if protocol indicates sigma0 protocol.
- bool **is_io_page_fault** () const [L4_NOTHROW](#)
Test if protocol indicates IO-page-fault protocol.
- bool **has_error** () const [L4_NOTHROW](#)
Test if flags indicate an error.

Data Fields

- [l4_mword_t](#) **raw**
raw value

15.251.1 Detailed Description

Message tag data structure.

Include File

```
#include <l4/sys/types.h>
```

Describes the details of an IPC operation, in particular which parts of the UTCB have to be transmitted, and also flags to enable real-time and FPU extensions.

The message tag also contains a user-defined label that could be used to specify a protocol ID. Some negative values are reserved for kernel protocols such as page faults and exceptions.

The type must be treated completely opaque.

Examples

[examples/libs/l4re/c++/shared_ds/ds_srv.cc](#), [examples/libs/l4re/streammap/server.cc](#), [examples/sys/aliens/main.c](#), [examples/sys/ipc/ipc_example.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 265 of file [types.h](#).

15.251.2 Member Function Documentation

15.251.2.1 flags()

```
unsigned l4_msgtag_t::flags () const [inline]
```

Get the flags value.

The flags are a combination of the flags defined by [L4_msgtag_flags](#).

Definition at line 283 of file [types.h](#).

References [l4_msgtag_flags\(\)](#), and [L4_NOTHROW](#).

Here is the call graph for this function:



15.251.2.2 has_error()

```
bool l4_msgtag_t::has_error () const [inline]
```

Test if flags indicate an error.

If true, the error code is stored in the UTCB, see [l4_utcb_tcr\(\)->error](#).

Definition at line 299 of file [types.h](#).

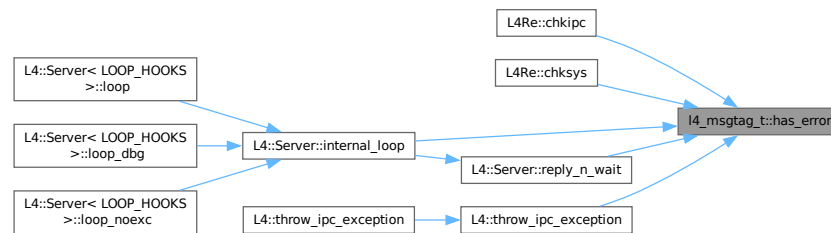
References [l4_msgtag_has_error\(\)](#), and [L4_NOTHROW](#).

Referenced by [L4Re::chkipc\(\)](#), [L4Re::chksys\(\)](#), [L4::Server< LOOP_HOOKS >::internal_loop\(\)](#), [L4::Server< LOOP_HOOKS >::reply](#) and [L4::throw_ipc_exception\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this struct was generated from the following file:

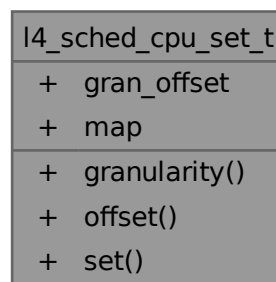
- [l4/sys/types.h](#)

15.252 l4_sched_cpu_set_t Struct Reference

CPU sets.

```
#include <scheduler.h>
```

Collaboration diagram for l4_sched_cpu_set_t:



Public Member Functions

- unsigned char [granularity](#) () const
- unsigned [offset](#) () const
- void [set](#) (unsigned char [granularity](#), unsigned [offset](#))
Set offset and granularity.

Data Fields

- [l4_umword_t gran_offset](#)
Combination of granularity and offset.
- [l4_umword_t map](#)
Bitmap of CPUs.

15.252.1 Detailed Description

CPU sets.

Examples

[examples/sys/migrate/thread_migrate.cc](#).

Definition at line 58 of file [scheduler.h](#).

15.252.2 Member Function Documentation

15.252.2.1 granularity()

```
unsigned char l4_sched_cpu_set_t::granularity () const [inline]
```

Returns

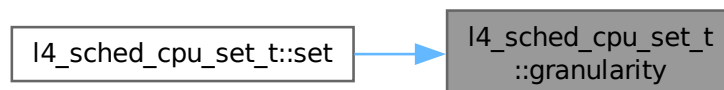
Get granularity value

Definition at line 81 of file [scheduler.h](#).

References [gran_offset](#).

Referenced by [set\(\)](#).

Here is the caller graph for this function:



15.252.2.2 `offset()`

```
unsigned l4_sched_cpu_set_t::offset () const [inline]
```

Returns

Get offset value

Definition at line 83 of file [scheduler.h](#).

References [gran_offset](#).

Referenced by [set\(\)](#).

Here is the caller graph for this function:



15.252.2.3 `set()`

```
void l4_sched_cpu_set_t::set (  
    unsigned char granularity,  
    unsigned offset) [inline]
```

Set offset and granularity.

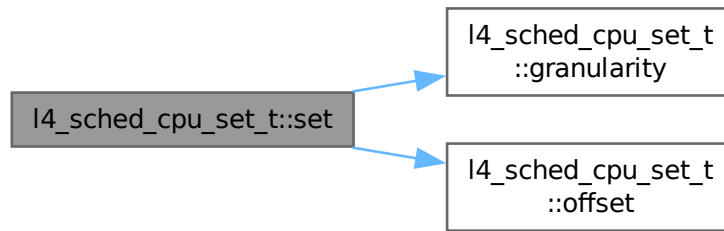
Parameters

| | |
|--------------------|--|
| <i>granularity</i> | Granularity in log2 notation. |
| <i>offset</i> | Offset. Must be a multiple of $2^{\text{granularity}}$. |

Definition at line 90 of file [scheduler.h](#).

References [gran_offset](#), [granularity\(\)](#), and [offset\(\)](#).

Here is the call graph for this function:



15.252.3 Field Documentation

15.252.3.1 gran_offset

`l4_umword_t l4_sched_cpu_set_t::gran_offset`

Combination of granularity and offset.

The granularity defines how many CPUs each bit in map describes. And the offset is the number of the first CPU described by the first bit in the bitmap.

Precondition

offset must be a multiple of $2^{\text{granularity}}$.

| MSB | LSB |
|------------------|-----------------|
| 8bit granularity | 24bit offset .. |

Definition at line 72 of file [scheduler.h](#).

Referenced by [granularity\(\)](#), [L4::Scheduler::info\(\)](#), [l4_sched_cpu_set\(\)](#), [offset\(\)](#), and [set\(\)](#).

The documentation for this struct was generated from the following file:

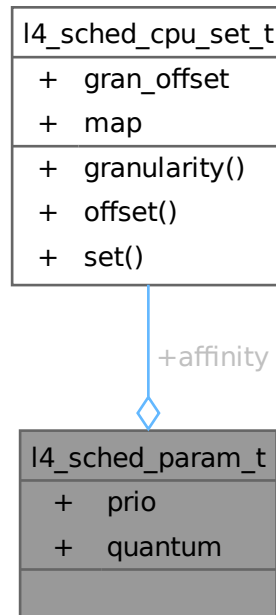
- [l4/sys/scheduler.h](#)

15.253 l4_sched_param_t Struct Reference

Scheduler parameter set.

```
#include <scheduler.h>
```

Collaboration diagram for l4_sched_param_t:



Data Fields

- [l4_sched_cpu_set_t](#) **affinity**
CPU affinity.
- [l4_umword_t](#) **prio**
Priority for scheduling.
- [l4_umword_t](#) **quantum**
Timeslice in micro seconds.

15.253.1 Detailed Description

Scheduler parameter set.

Examples

[examples/sys/aliens/main.c](#), [examples/sys/migrate/thread_migrate.cc](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 173 of file [scheduler.h](#).

15.253.2 Field Documentation

15.253.2.1 prio

```
l4_umword_t l4_sched_param_t::prio
```

Priority for scheduling.

The kernel supports priorities for userland threads in the range of 1..255. Priority 0 is reserved for the kernel.

Definition at line 182 of file [scheduler.h](#).

Referenced by [l4_sched_param\(\)](#).

The documentation for this struct was generated from the following file:

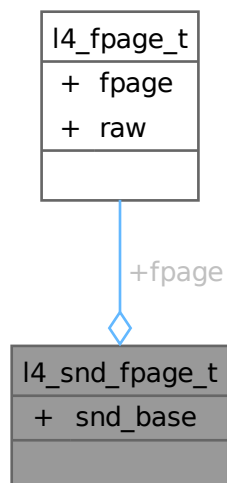
- [l4/sys/scheduler.h](#)

15.254 l4_snd_fpage_t Struct Reference

Send-flexpage types.

```
#include <__l4_fpage.h>
```

Collaboration diagram for l4_snd_fpage_t:



Data Fields

- [l4_umword_t](#) **snd_base**
Offset in receive window (send base).
- [l4_fpage_t](#) **fpage**
Source flexpage descriptor.

15.254.1 Detailed Description

Send-flexpage types.

Definition at line 99 of file [__l4_fpage.h](#).

The documentation for this struct was generated from the following file:

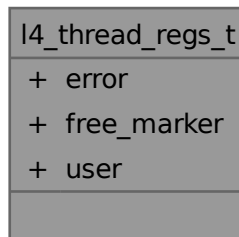
- [l4/sys/__l4_fpage.h](#)

15.255 l4_thread_regs_t Struct Reference

Encapsulation of the thread-control-register block of the UTCB.

```
#include <utcb.h>
```

Collaboration diagram for l4_thread_regs_t:



Data Fields

- [l4_umword_t](#) `error`
System call error code (see [l4_ipc_tcr_error_t](#)).
- [l4_umword_t](#) `free_marker`
Kernel free marker.
- [l4_umword_t](#) `user` [3]
User values (ignored and preserved by the kernel).

15.255.1 Detailed Description

Encapsulation of the thread-control-register block of the UTCB.

Definition at line 164 of file [utcb.h](#).

15.255.2 Field Documentation

15.255.2.1 error

`l4_umword_t l4_thread_regs_t::error`

System call error code (see [l4_ipc_tcr_error_t](#)).

If the kernel indicates an error in the message tag (see [l4_msgtag_has_error\(\)](#) and [l4_msgtag_t::has_error\(\)](#)), the kernel writes the error code to this field.

Definition at line 171 of file [utcb.h](#).

15.255.2.2 free_marker

`l4_umword_t l4_thread_regs_t::free_marker`

Kernel free marker.

The kernel sets this field to zero as soon as it is guaranteed that the kernel does not use the UTCB anymore for the bound thread. This usually happens while a thread is deleted. However, it is not defined when exactly the kernel sets the field. In particular, the point in time is not necessarily related to any IPC.

Userland may use this field for determining if a UTCB can be re-used for another thread. Note that, in order to make use of that feature, userland has to set this field to a non-zero value when a thread is bound with this UTCB.

Definition at line 185 of file [utcb.h](#).

The documentation for this struct was generated from the following file:

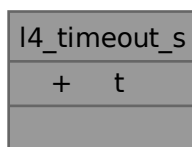
- [l4/sys/utcb.h](#)

15.256 l4_timeout_s Struct Reference

Basic timeout specification.

```
#include <__timeout.h>
```

Collaboration diagram for `l4_timeout_s`:



Data Fields

- [l4_uint16_t t](#)
timeout value

15.256.1 Detailed Description

Basic timeout specification.

If bit 15 == 0, basically a floating point number with 10 bits mantissa and 5 bits exponent ($t = m \cdot 2^e$).

If the mantissa is zero, the exponent encodes special values, see [L4_IPC_TIMEOUT_0](#) and [L4_IPC_TIMEOUT_NEVER](#).

If bit 15 == 1 the timeout is absolute and the lower 6 bits encode the index of the UTCB buffer register(s) holding the absolute 64-bit timeout value. On 32-bit systems, two consecutive UTCB buffer registers are used.

Definition at line 40 of file [__timeout.h](#).

The documentation for this struct was generated from the following file:

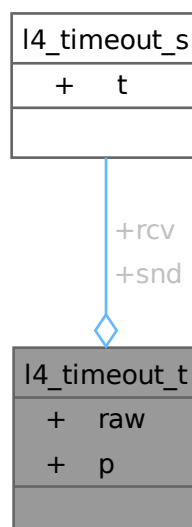
- [l4/sys/__timeout.h](#)

15.257 l4_timeout_t Union Reference

Timeout pair.

```
#include <__timeout.h>
```

Collaboration diagram for `l4_timeout_t`:



Data Fields

- [l4_uint32_t](#) **raw**
raw value
- struct {
 [l4_timeout_s](#) **rcv**
 receive timeout
 [l4_timeout_s](#) **snd**
 send timeout
} **p**

combined timeout

15.257.1 Detailed Description

Timeout pair.

For IPC there are usually a send and a receive timeout. So this structure contains a pair of timeouts.

Definition at line 52 of file [__timeout.h](#).

The documentation for this union was generated from the following file:

- l4/sys/__timeout.h

15.258 l4_vcon_attr_t Struct Reference

Vcon attribute structure.

```
#include <vcon.h>
```

Collaboration diagram for l4_vcon_attr_t:

| l4_vcon_attr_t |
|----------------|
| + i_flags |
| + o_flags |
| + l_flags |
| + set_raw() |

Public Member Functions

- void [set_raw](#) ()
Set terminal attributes to disable all special processing.

Data Fields

- [l4_umword_t i_flags](#)
input flags
- [l4_umword_t o_flags](#)
output flags
- [l4_umword_t l_flags](#)
local flags

15.258.1 Detailed Description

Vcon attribute structure.

The flags members can be a combination of their respective enums.

See also

[L4_vcon_i_flags](#)
[L4_vcon_o_flags](#)
[L4_vcon_l_flags](#)

Examples

[examples/sys/isr/main.c](#).

Definition at line [187](#) of file [vcon.h](#).

15.258.2 Member Function Documentation

15.258.2.1 set_raw()

```
void l4_vcon_attr_t::set_raw () [inline]
```

Set terminal attributes to disable all special processing.

Removes all flags that would mangle the read or written characters. Also disables echoing and any special processing of characters.

Definition at line [450](#) of file [vcon.h](#).

References [l4_vcon_set_attr_raw\(\)](#).

Here is the call graph for this function:



The documentation for this struct was generated from the following file:

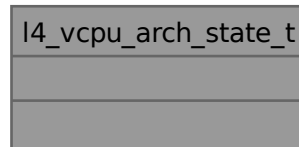
- [l4/sys/vcon.h](#)

15.259 l4_vcpu_arch_state_t Struct Reference

Architecture-specific vCPU state.

```
#include <__vcpu-arch.h>
```

Collaboration diagram for l4_vcpu_arch_state_t:



15.259.1 Detailed Description

Architecture-specific vCPU state.

Definition at line 74 of file [__vcpu-arch.h](#).

The documentation for this struct was generated from the following files:

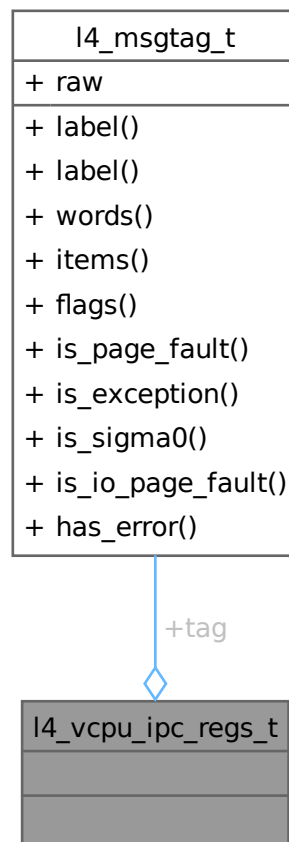
- [arm/l4/sys/__vcpu-arch.h](#)
- [arm64/l4/sys/__vcpu-arch.h](#)
- [amd64/l4/sys/__vcpu-arch.h](#)
- [riscv/l4/sys/__vcpu-arch.h](#)

15.260 l4_vcpu_ipc_regs_t Struct Reference

vCPU message registers.

```
#include <__vcpu-arch.h>
```

Collaboration diagram for `l4_vcpu_ipc_regs_t`:



15.260.1 Detailed Description

vCPU message registers.

Definition at line 83 of file [__vcpu-arch.h](#).

The documentation for this struct was generated from the following files:

- [arm/l4/sys/__vcpu-arch.h](#)
- [arm64/l4/sys/__vcpu-arch.h](#)
- [amd64/l4/sys/__vcpu-arch.h](#)
- [x86/l4/sys/__vcpu-arch.h](#)
- [riscv/l4/sys/__vcpu-arch.h](#)

15.261 l4_vcpu_regs_t Struct Reference

vCPU registers.

```
#include <__vcpu-arch.h>
```

Collaboration diagram for l4_vcpu_regs_t:

| l4_vcpu_regs_t |
|----------------|
| + pfa |
| + err |
| + sp |
| + ip |
| + flags |
| + tpidruro |
| + tpidrurw |
| + r15 |
| + r14 |
| + r13 |
| and 20 more... |

Data Fields

- [l4_umword_t](#) **pfa**
page fault address
- [l4_umword_t](#) **err**
error code
- [l4_umword_t](#) **sp**
stack pointer
- [l4_umword_t](#) **ip**
instruction pointer
- [l4_umword_t](#) **flags**
eflags
- [l4_umword_t](#) **tpidruro**
Thread-ID register.
- [l4_umword_t](#) **tpidrurw**
Thread-ID register.
- [l4_umword_t](#) **r15**
r15 register
- [l4_umword_t](#) **r14**

- r14 register*
- [l4_umword_t r13](#)
- r13 register*
- [l4_umword_t r12](#)
- r12 register*
- [l4_umword_t r11](#)
- r11 register*
- [l4_umword_t r10](#)
- r10 register*
- [l4_umword_t r9](#)
- r9 register*
- [l4_umword_t r8](#)
- r8 register*
- [l4_umword_t di](#)
- rdi register*
- [l4_umword_t si](#)
- rsi register*
- [l4_umword_t bp](#)
- rbp register*
- [l4_umword_t bx](#)
- rbx register*
- [l4_umword_t dx](#)
- rdx register*
- [l4_umword_t cx](#)
- rcx register*
- [l4_umword_t ax](#)
- rax register*
- [l4_umword_t trapno](#)
- trap number*
- [l4_umword_t cs](#)
- dummy*
- [l4_umword_t ss](#)
- ss register*
- [l4_umword_t es](#)
- es register*
- [l4_umword_t ds](#)
- ds register*
- [l4_umword_t gs](#)
- gs register*
- [l4_umword_t fs](#)
- fs register*
- [l4_umword_t dummy1](#)
- dummy*

15.261.1 Detailed Description

vCPU registers.

Definition at line 55 of file [__vcpu-arch.h](#).

15.261.2 Field Documentation

15.261.2.1 ax

`l4_umword_t l4_vcpu_regs_t::ax`

rax register

eax register

Definition at line 77 of file [__vcpu-arch.h](#).

15.261.2.2 bp

`l4_umword_t l4_vcpu_regs_t::bp`

rbp register

ebp register

Definition at line 72 of file [__vcpu-arch.h](#).

15.261.2.3 bx

`l4_umword_t l4_vcpu_regs_t::bx`

rbx register

ebx register

Definition at line 74 of file [__vcpu-arch.h](#).

15.261.2.4 cx

`l4_umword_t l4_vcpu_regs_t::cx`

rcx register

ecx register

Definition at line 76 of file [__vcpu-arch.h](#).

15.261.2.5 di

`l4_umword_t l4_vcpu_regs_t::di`

rdi register

edi register

Definition at line 70 of file [__vcpu-arch.h](#).

15.261.2.6 dx

```
l4_umword_t l4_vcpu_regs_t::dx
```

rdx register

edx register

Definition at line 75 of file [__vcpu-arch.h](#).

15.261.2.7 si

```
l4_umword_t l4_vcpu_regs_t::si
```

rsi register

esi register

Definition at line 71 of file [__vcpu-arch.h](#).

The documentation for this struct was generated from the following files:

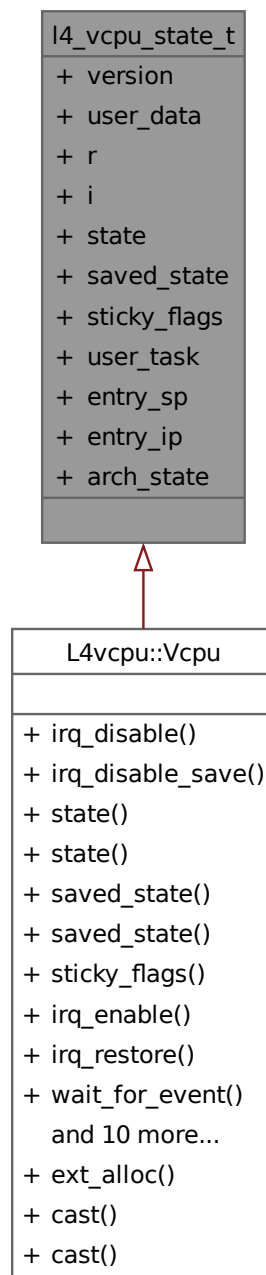
- [arm/l4/sys/__vcpu-arch.h](#)
- [amd64/l4/sys/__vcpu-arch.h](#)
- [x86/l4/sys/__vcpu-arch.h](#)

15.262 l4_vcpu_state_t Struct Reference

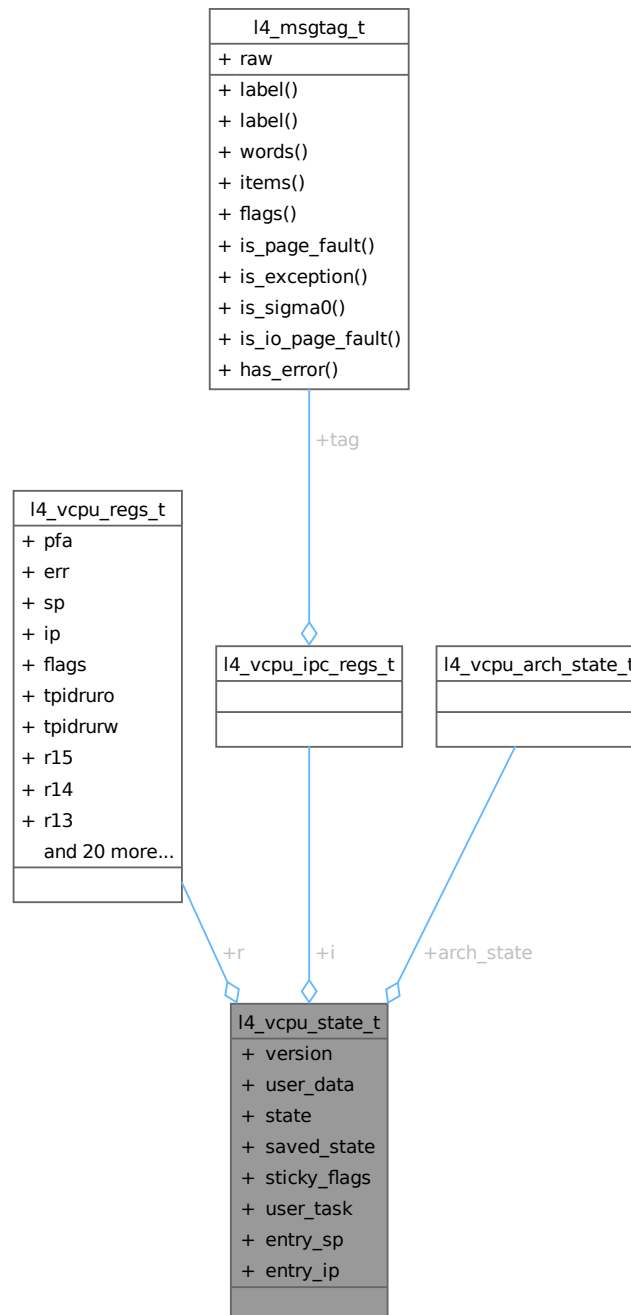
State of a vCPU.

```
#include <vcpu.h>
```

Inheritance diagram for l4_vcpu_state_t:



Collaboration diagram for `l4_vcpu_state_t`:



Data Fields

- [l4_umword_t version](#)
vCPU ABI version.
- [l4_umword_t user_data](#) [7]
User-specific data.
- [l4_vcpu_regs_t r](#)

- Register state.*
- [l4_vcpu_ipc_regs_t](#) **i**
IPC state.
- [l4_uint16_t](#) **state**
Current vCPU state. See [L4_vcpu_state_flags](#).
- [l4_uint16_t](#) **saved_state**
Saved vCPU state. See [L4_vcpu_state_flags](#).
- [l4_uint16_t](#) **sticky_flags**
Pending flags. See [L4_vcpu_sticky_flags](#).
- [l4_cap_idx_t](#) **user_task**
User task to use.
- [l4_umword_t](#) **entry_sp**
Stack pointer for entry (when coming from user task).
- [l4_umword_t](#) **entry_ip**
IP for entry.
- [l4_vcpu_arch_state_t](#) **arch_state**
Architecture-specific state.

15.262.1 Detailed Description

State of a vCPU.

Definition at line 75 of file [vcpu.h](#).

15.262.2 Field Documentation

15.262.2.1 version

```
l4\_umword\_t l4_vcpu_state_t::version
```

vCPU ABI version.

Set by the kernel and must be checked by the user for equality with [L4_VCPU_STATE_VERSION](#).

Definition at line 77 of file [vcpu.h](#).

The documentation for this struct was generated from the following file:

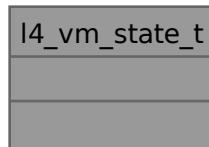
- [l4/sys/vcpu.h](#)

15.263 l4_vm_state_t Struct Reference

L4 extended vCPU state for RISC-V.

```
#include <vm.h>
```

Collaboration diagram for l4_vm_state_t:



15.263.1 Detailed Description

L4 extended vCPU state for RISC-V.

Contains the additional RISC-V guest state accompanying the [l4_vcpu_state_t](#).

Definition at line 50 of file [vm.h](#).

The documentation for this struct was generated from the following file:

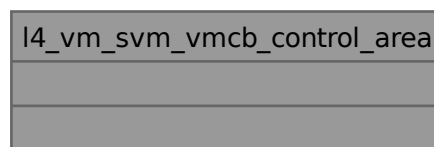
- riscv/l4/sys/vm.h

15.264 l4_vm_svm_vmcb_control_area Struct Reference

VMCB structure for SVM VMs.

```
#include <__vm-svm.h>
```

Collaboration diagram for l4_vm_svm_vmcb_control_area:



15.264.1 Detailed Description

VMCB structure for SVM VMs.

Definition at line 28 of file [__vm-svm.h](#).

The documentation for this struct was generated from the following file:

- l4/sys/__vm-svm.h

15.265 l4_vm_svm_vmcb_state_save_area Struct Reference

State save area structure for SVM VMs.

```
#include <__vm-svm.h>
```

Collaboration diagram for l4_vm_svm_vmcb_state_save_area:



15.265.1 Detailed Description

State save area structure for SVM VMs.

Definition at line 85 of file [__vm-svm.h](#).

The documentation for this struct was generated from the following file:

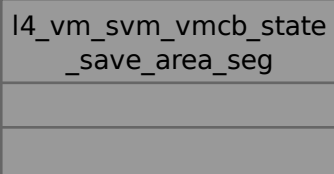
- `l4/sys/__vm-svm.h`

15.266 `l4_vm_svm_vmcb_state_save_area_seg` Struct Reference

State save area segment selector struct.

```
#include <__vm-svm.h>
```

Collaboration diagram for `l4_vm_svm_vmcb_state_save_area_seg`:



15.266.1 Detailed Description

State save area segment selector struct.

Definition at line 73 of file [__vm-svm.h](#).

The documentation for this struct was generated from the following file:

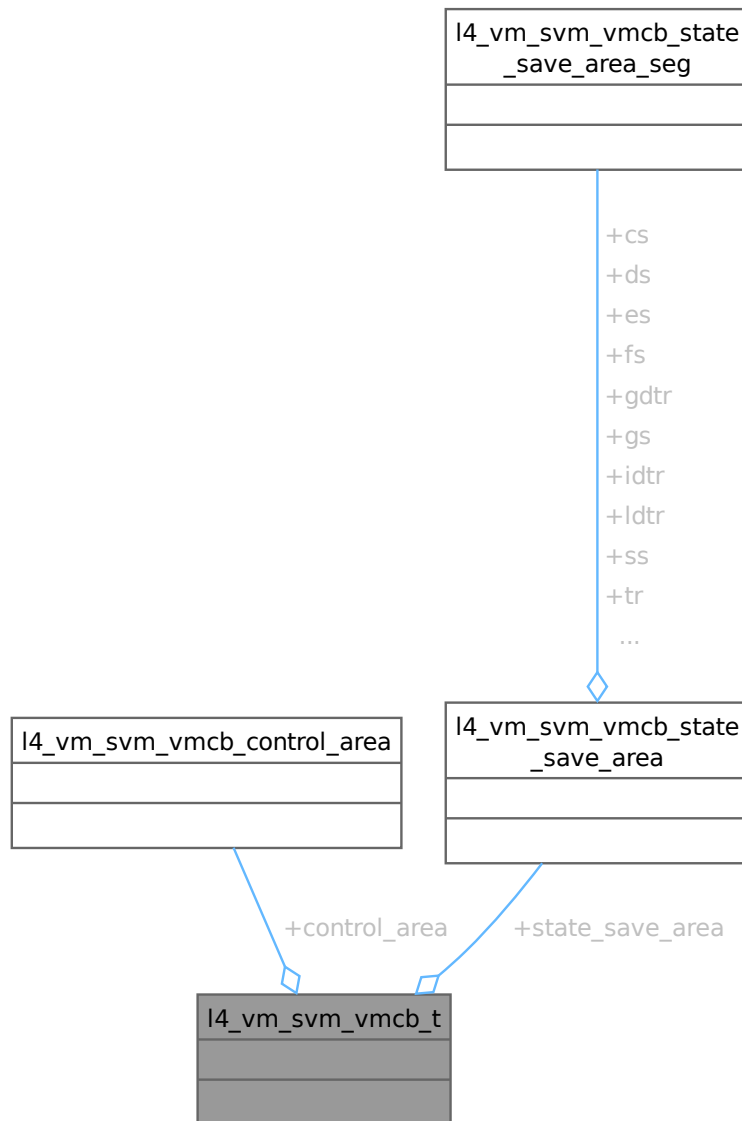
- `l4/sys/__vm-svm.h`

15.267 l4_vm_svm_vmcb_t Struct Reference

Control structure for SVM VMs.

```
#include <__vm-svm.h>
```

Collaboration diagram for l4_vm_svm_vmcb_t:



15.267.1 Detailed Description

Control structure for SVM VMs.

Definition at line 154 of file [__vm-svm.h](#).

The documentation for this struct was generated from the following file:

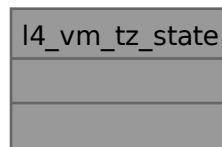
- l4/sys/__vm-svm.h

15.268 l4_vm_tz_state Struct Reference

state structure for TrustZone VMs

```
#include <vm.h>
```

Collaboration diagram for l4_vm_tz_state:



15.268.1 Detailed Description

state structure for TrustZone VMs

Definition at line 44 of file [vm.h](#).

The documentation for this struct was generated from the following file:

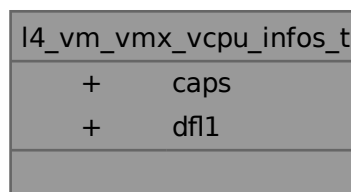
- [arm/l4/sys/vm.h](#)

15.269 l4_vm_vmx_vcpu_infos_t Struct Reference

VMX information members.

```
#include <__vm-vmx.h>
```

Collaboration diagram for l4_vm_vmx_vcpu_infos_t:



Data Fields

- [l4_uint64_t caps](#) [[L4_VM_VMX_NUM_CAPS_REGS](#)]
Exported VMX capability registers. See [L4_vm_vmx_caps_regs](#).
- [l4_uint32_t dfl1](#) [[L4_VM_VMX_NUM_DFL1_REGS](#)]
Exported VMX capability registers (default to 1 bits).

15.269.1 Detailed Description

VMX information members.

Definition at line 239 of file [__vm-vmx.h](#).

15.269.2 Field Documentation

15.269.2.1 dfl1

```
l4\_uint32\_t l4_vm_vmx_vcpu_infos_t::dfl1 [L4\_VM\_VMX\_NUM\_DFL1\_REGS]
```

Exported VMX capability registers (default to 1 bits).

See [L4_vm_vmx_dfl1_regs](#).

Definition at line 246 of file [__vm-vmx.h](#).

The documentation for this struct was generated from the following file:

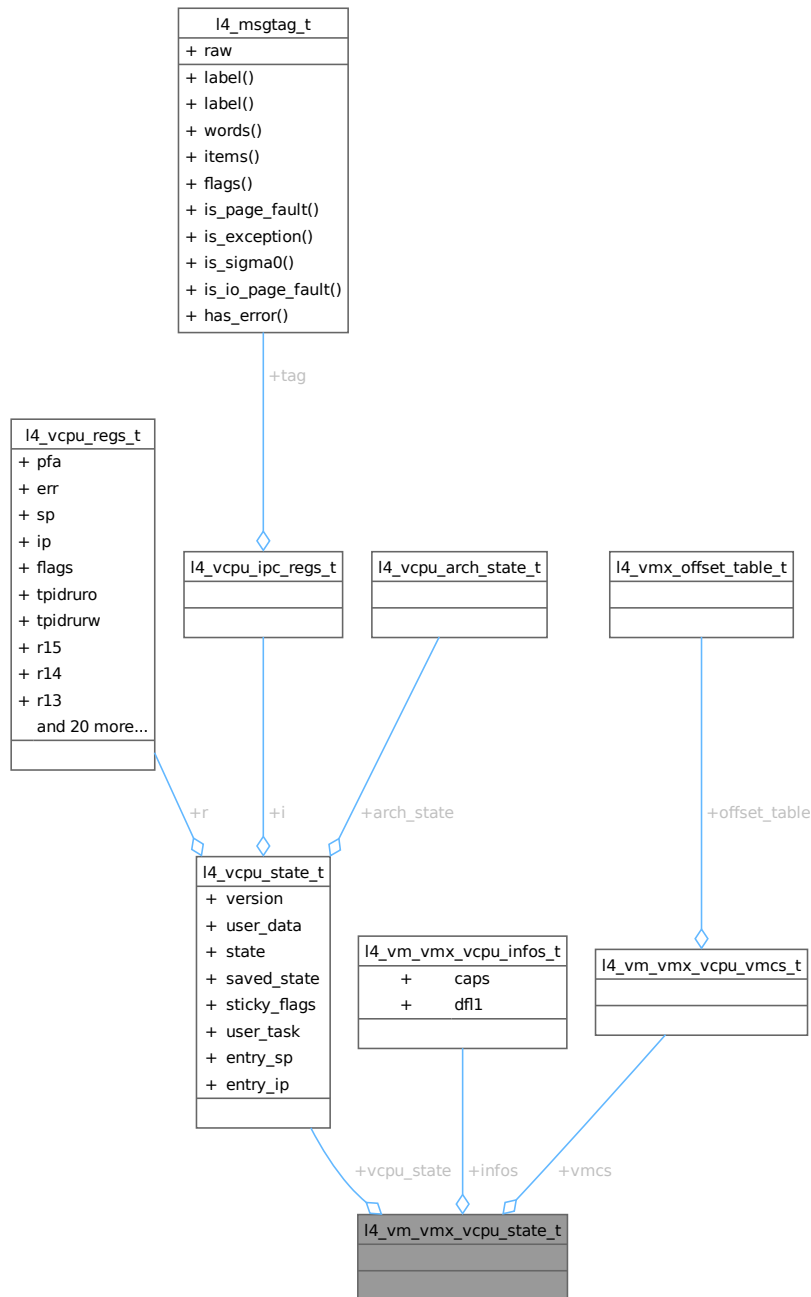
- [l4/sys/__vm-vmx.h](#)

15.270 l4_vm_vmx_vcpu_state_t Struct Reference

VMX vCPU state.

```
#include <\_\_vm-vmx.h>
```

Collaboration diagram for `l4_vm_vmx_vcpu_state_t`:



15.270.1 Detailed Description

VMX vCPU state.

This is a specialization of the generic vCPU state for VMX. This data structure represents the following memory layout:

- 0x000 - 0x1ff: Standard vCPU state (with padding). See [l4_vcpu_state_t](#).

- 0x200 - 0x3ff: VMX information members (with padding). See [l4_vm_vmx_vcpu_infos_t](#).
- 0x400 - 0xfff: VMX software VMCS. See [l4_vm_vmx_vcpu_vmcs_t](#).

Note

The memory layout is documented here for reference purposes. However, the users are strongly discouraged from accessing the data structure directly. The API functions defined in this file are the preferred way of achieving the functionality.

Definition at line 267 of file [__vm-vmx.h](#).

The documentation for this struct was generated from the following file:

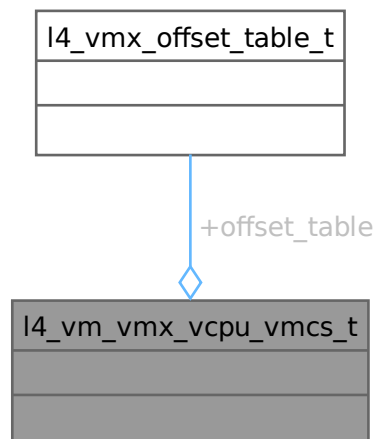
- l4/sys/__vm-vmx.h

15.271 l4_vm_vmx_vcpu_vmcs_t Struct Reference

VMX software VMCS.

```
#include <__vm-vmx.h>
```

Collaboration diagram for l4_vm_vmx_vcpu_vmcs_t:



15.271.1 Detailed Description

VMX software VMCS.

This data structure represents the following memory layout:

- 0x000 - 0x007: Reserved (ignored by the kernel). In the hardware VMCS, the revision identifier and the abort indicator are stored in this area. Hereby we simply ignore these two entries.
- 0x008 - 0x00f: User space data (ignored by the kernel). This currently stores the pointer to a different software VMCS whose content has been loaded to this software VMCS.
- 0x010 - 0x013: VMCS field index of the software-defined CR2 field in the software VMCS.
- 0x014 - 0x017: Reserved.
- 0x018 - 0x01f: Capability of the vCPU context, i.e. the hardware VMCS object (with padding).
- 0x020 - 0x047: Software VMCS field offset table. See [l4_vmx_offset_table_t](#).
- 0x048 - 0x0bf: Reserved.
- 0x0c0 - 0xabf: Software VMCS fields (with padding).
- 0xac0 - 0xbff: Software VMCS fields dirty bitmap (with padding).

Note

The memory layout is documented here for reference purposes. However, the users are strongly discouraged from accessing the data structure directly. The API functions defined in this file are the preferred way of achieving the functionality.

Definition at line 205 of file [__vm-vmx.h](#).

The documentation for this struct was generated from the following file:

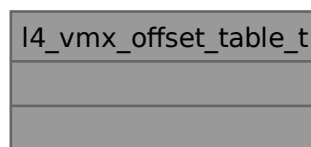
- l4/sys/__vm-vmx.h

15.272 l4_vmx_offset_table_t Struct Reference

Software VMCS field offset table.

```
#include <__vm-vmx.h>
```

Collaboration diagram for l4_vmx_offset_table_t:



15.272.1 Detailed Description

Software VMCS field offset table.

This data structure represents the following memory layout:

- 0x00 - 0x02: 3 offsets for 16-bit fields.
- 0x03: Reserved.
- 0x04 - 0x06: 3 offsets for 64-bit fields.
- 0x07: Reserved.
- 0x08 - 0x0a: 3 offsets for 32-bit fields.
- 0x0b: Reserved.
- 0x0c - 0x0e: 3 offsets for natural-width fields.
- 0x0f: Reserved.
- 0x10 - 0x12: 3 limits for 16-bit fields.
- 0x13: Reserved.
- 0x14 - 0x16: 3 limits for 64-bit fields.
- 0x17: Reserved.
- 0x18 - 0x1a: 3 limits for 32-bit fields.
- 0x1b: Reserved.
- 0x1c - 0x1e: 3 limits for natural-width fields.
- 0x1f: Reserved.
- 0x20 - 0x23: 4 index shifts.
- 0x24: Offset of the first software VMCS field.
- 0x25: Size of the software VMCS fields.
- 0x26 - 0x27: Reserved.

The offsets/limits in each size category are in the following order:

- Control fields.
- Read-only fields.
- Guest fields.

The index shifts are in the following order:

- 16-bit.
- 64-bit.
- 32-bit.
- Natural-width.

All offsets/limits/sizes are represented in a 64-byte granule.

The offsets (after being multiplied by 64) are indexes in the values array in [l4_vm_vmx_vcpu_vmcs_t](#) and bit indexes in the dirty_bitmap array in [l4_vm_vmx_vcpu_vmcs_t](#).

The limits (after being multiplied by 64) represent the range of the available indexes.

Note

The memory layout is documented here for reference purposes. However, the users are strongly discouraged from accessing the data structure directly. The API functions defined in this file are the preferred way of achieving the functionality.

Definition at line 155 of file [__vm-vmx.h](#).

The documentation for this struct was generated from the following file:

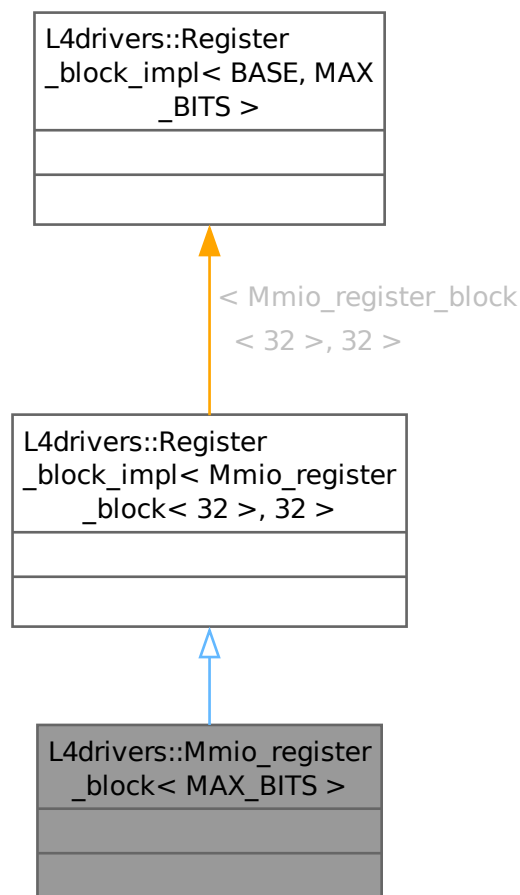
- [l4/sys/__vm-vmx.h](#)

15.273 L4drivers::Mmio_register_block< MAX_BITS > Struct Template Reference

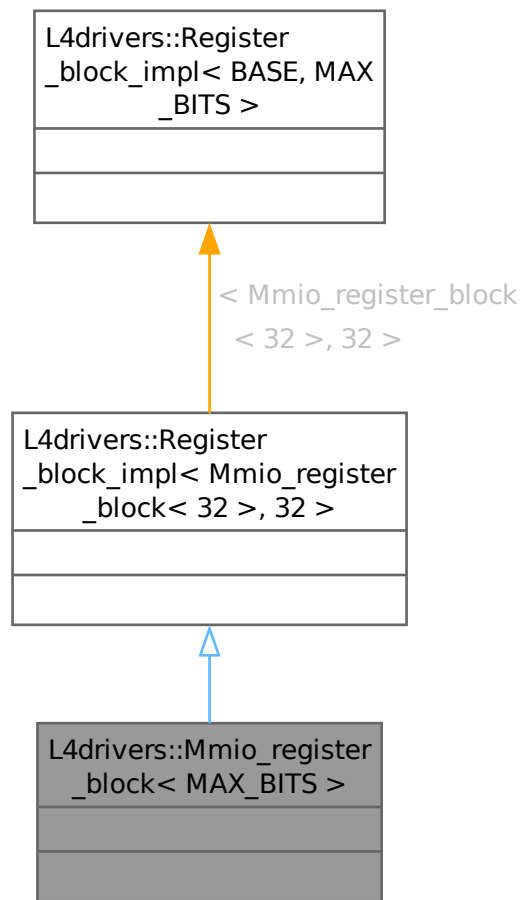
An MMIO block with up to 64-bit register access (32-bit default) and little endian byte order.

```
#include <hw_mmio_register_block>
```

Inheritance diagram for L4drivers::Mmio_register_block< MAX_BITS >:



Collaboration diagram for L4drivers::Mmio_register_block< MAX_BITS >:



15.273.1 Detailed Description

```
template<unsigned MAX_BITS = 32>
struct L4drivers::Mmio_register_block< MAX_BITS >
```

An MMIO block with up to 64-bit register access (32-bit default) and little endian byte order.

Definition at line 45 of file [hw_mmio_register_block](#).

The documentation for this struct was generated from the following file:

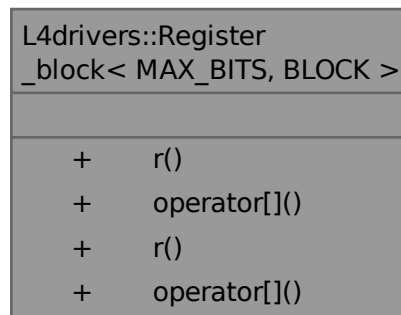
- pkg/drivers-frst/include/hw_mmio_register_block

15.274 L4drivers::Register_block< MAX_BITS, BLOCK > Class Template Reference

Handles a reference to a register block of the given maximum access width.

```
#include <hw_register_block>
```

Collaboration diagram for L4drivers::Register_block< MAX_BITS, BLOCK >:



Public Member Functions

- template<unsigned BITS>
Ro_register_tmpl< BITS, Block > r (unsigned offset) const
Read only access to register at offset offset.
- Ro_register operator[] (unsigned offset) const
Read only access to register at offset offset.
- template<unsigned BITS>
Register_tmpl< BITS, Block > r (unsigned offset)
Read/write access to register at offset offset.
- Register operator[] (unsigned offset)
Read/write access to register at offset offset.

15.274.1 Detailed Description

```
template<unsigned MAX_BITS, typename BLOCK = Register_block_tmpl< Register_block_base<MAX_BITS> >>
class L4drivers::Register_block< MAX_BITS, BLOCK >
```

Handles a reference to a register block of the given maximum access width.

Register block.

Template Parameters

| | |
|-----------------|---|
| <i>MAX_BITS</i> | Maximum access width for the registers in this block. |
| <i>BLOCK</i> | Type implementing the register accesses (<code>read<>()</code> , <code>write<>()</code> , <code>modify<>()</code> , <code>set<>()</code> , and <code>clear<>()</code>). |

Provides access to registers in this block via `r<WIDTH>()` and `operator[]()`.

Example usage:

```
void test()
{
    // create a register block reference for max. 16bit accesses, using a
    // MMIO register block implementation (at address 0x1000).
    Hw::Register_block<16> regs = new Hw::Mmio_register_block<16>(0x1000);

    // Alternatively it is allowed to use an implementation that allows
    // wider access than actually needed.
    Hw::Register_block<16> regs = new Hw::Mmio_register_block<32>(0x1000);

    // read a 16bit register at offset 8byte
    unsigned short x = regs.r<16>(8);
    unsigned short x1 = regs[8];           // alternative

    // read an 8bit register at offset 0byte
    unsigned v = regs.r<8>(0);

    // do a 16bit write to register at offset 2byte (four variants)
    regs[2] = 22;
    regs.r<16>(2) = 22;
    regs[2].write(22);
    regs.r<16>().write(22);

    // do an 8bit write (two variants)
    regs.r<8>(0) = 9;
    regs.r<8>(0).write(9);

    // do 16bit read-modify-write (two variants)
    regs[4].modify(0xf, 3); // clear 4 lowest bits and set them to 3
    regs.r<16>(4).modify(0xf, 3);

    // do 8bit read-modify-write
    regs.r<8>(0).modify(0xf, 3);

    // fails to compile, because of too wide access
    // (32 bit access but regs is Hw::Register_block<16>)
    unsigned long v = regs.r<32>(4)
}
```

Definition at line 330 of file `hw_register_block`.

15.274.2 Member Function Documentation

15.274.2.1 `operator[]()` [1/2]

```
template<unsigned MAX_BITS, typename BLOCK = Register_block_tmpl< Register_block_base<MAX_BITS> >>
Register L4drivers::Register_block< MAX_BITS, BLOCK >::operator[] (
    unsigned offset) [inline]
```

Read/write access to register at offset *offset*.

Parameters

| | |
|---------------|--|
| <i>offset</i> | The offset of the register within the register file. |
|---------------|--|

Returns

register object allowing read and write access with width *MAX_BITS*.

Definition at line 385 of file [hw_register_block](#).

References [r\(\)](#).

Here is the call graph for this function:

**15.274.2.2 operator[]() [2/2]**

```

template<unsigned MAX_BITS, typename BLOCK = Register_block_tmpl< Register_block_base<MAX_BITS> >>
Ro_register L4drivers::Register_block< MAX_BITS, BLOCK >::operator[] (
    unsigned offset) const [inline]
  
```

Read only access to register at offset *offset*.

Parameters

| | |
|---------------|--|
| <i>offset</i> | The offset of the register within the register file. |
|---------------|--|

Returns

register object allowing read only access with width *MAX_BITS*.

Definition at line 365 of file [hw_register_block](#).

References [r\(\)](#).

Here is the call graph for this function:



15.274.2.3 `r()` [1/2]

```
template<unsigned MAX_BITS, typename BLOCK = Register_block_tmpl< Register_block_base<MAX_BITS> >>
template<unsigned BITS>
Register_tmpl< BITS, Block > L4drivers::Register_block< MAX_BITS, BLOCK >::r (
    unsigned offset) [inline]
```

Read/write access to register at offset *offset*.

Template Parameters

| | |
|-------------|--|
| <i>BITS</i> | the access width in bits for the register. |
|-------------|--|

Parameters

| | |
|---------------|--|
| <i>offset</i> | The offset of the register within the register file. |
|---------------|--|

Returns

register object allowing read and write access with width *BITS*.

Definition at line 376 of file [hw_register_block](#).

15.274.2.4 `r()` [2/2]

```
template<unsigned MAX_BITS, typename BLOCK = Register_block_tmpl< Register_block_base<MAX_BITS> >>
template<unsigned BITS>
Ro_register_tmpl< BITS, Block > L4drivers::Register_block< MAX_BITS, BLOCK >::r (
    unsigned offset) const [inline]
```

Read only access to register at offset *offset*.

Template Parameters

| | |
|-------------|--|
| <i>BITS</i> | the access width in bits for the register. |
|-------------|--|

Parameters

| | |
|---------------|--|
| <i>offset</i> | The offset of the register within the register file. |
|---------------|--|

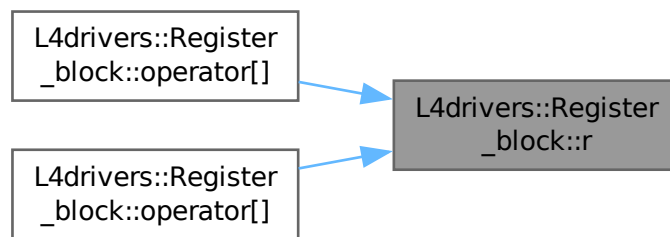
Returns

register object allowing read only access with width *BITS*.

Definition at line 357 of file [hw_register_block](#).

Referenced by [operator\[\]\(\)](#), and [operator\[\]\(\)](#).

Here is the caller graph for this function:



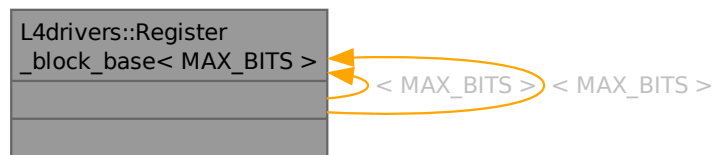
The documentation for this class was generated from the following file:

- `pkg/drivers-frst/include/hw_register_block`

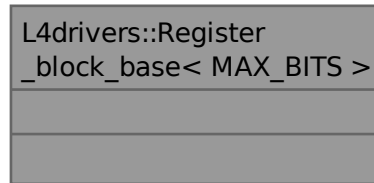
15.275 L4drivers::Register_block_base< MAX_BITS > Struct Template Reference

Abstract register block interface.

Inheritance diagram for L4drivers::Register_block_base< MAX_BITS >:



Collaboration diagram for L4drivers::Register_block_base< MAX_BITS >:



15.275.1 Detailed Description

```
template<unsigned MAX_BITS = 32>
struct L4drivers::Register_block_base< MAX_BITS >
```

Abstract register block interface.

Template Parameters

| | |
|-----------------|---|
| <i>MAX_BITS</i> | The maximum access width for the registers. |
|-----------------|---|

This interfaces is based on virtual `do_read_<xx>` and `do_write_<xx>` methods that have to be implemented up to the maximum access width.

Definition at line 72 of file [hw_register_block](#).

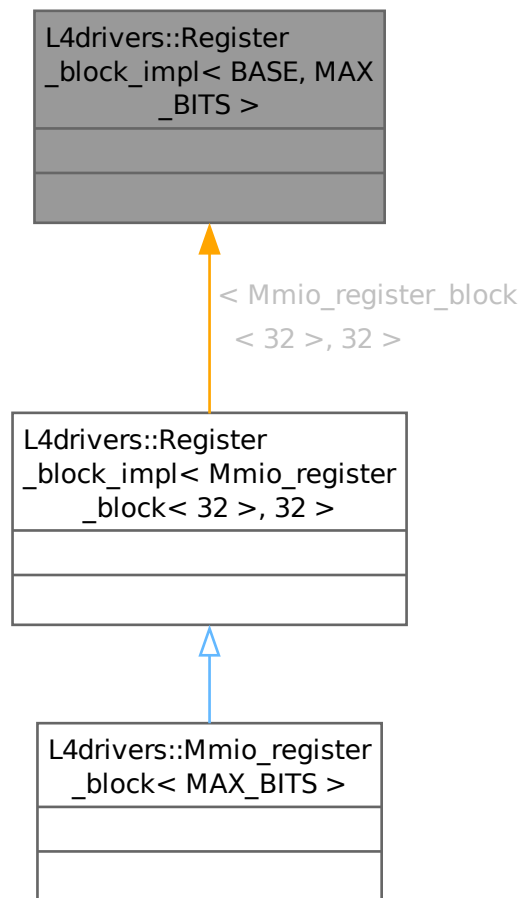
The documentation for this struct was generated from the following file:

- `pkg/drivers-frst/include/hw_register_block`

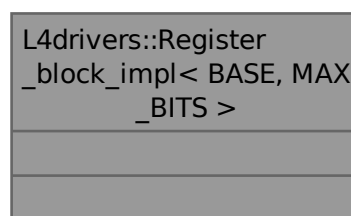
15.276 L4drivers::Register_block_impl< BASE, MAX_BITS > Struct Template Reference

Implementation helper for register blocks.

Inheritance diagram for L4drivers::Register_block_impl< BASE, MAX_BITS >:



Collaboration diagram for L4drivers::Register_block_impl< BASE, MAX_BITS >:



15.276.1 Detailed Description

```
template<typename BASE, unsigned MAX_BITS = 32>
struct L4drivers::Register_block_tmpl< BASE, MAX_BITS >
```

Implementation helper for register blocks.

Parameters

| | |
|-----------------|--|
| <i>BASE</i> | The class implementing read<> and write<> template functions for accessing the registers. This class must inherit from Register_block_impl . |
| <i>MAX_BITS</i> | The maximum access width for the register file. Supported values are 8, 16, 32, or 64. |

This template allows easy implementation of register files by providing read<> and write<> template functions, see [Mmio_register_block](#) as an example.

Definition at line 455 of file [hw_register_block](#).

The documentation for this struct was generated from the following file:

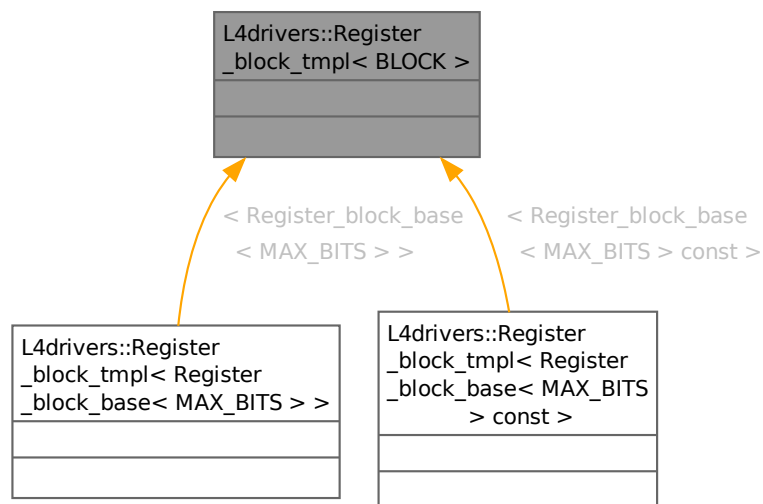
- [pkg/drivers-frst/include/hw_register_block](#)

15.277 L4drivers::Register_block_tmpl< BLOCK > Class Template Reference

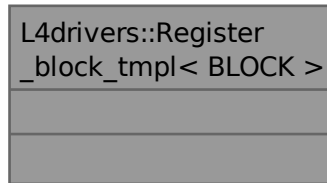
Helper template that translates to the [Register_block_base](#) interface.

```
#include <hw_register_block>
```

Inheritance diagram for L4drivers::Register_block_tmpl< BLOCK >:



Collaboration diagram for L4drivers::Register_block_tmpl< BLOCK >:



15.277.1 Detailed Description

```
template<typename BLOCK>
class L4drivers::Register_block_tmpl< BLOCK >
```

Helper template that translates to the [Register_block_base](#) interface.

Template Parameters

| | |
|--------------|---|
| <i>BLOCK</i> | The type of the Register_block_base interface to use. |
|--------------|---|

This helper translates `read<T>()`, `write<T>()`, `set<T>()`, `clear<T>()`, and `modify<T>()` calls to `BLOCK::do_read_<xx>` and `BLOCK::do_write_<xx>`.

Definition at line 156 of file [hw_register_block](#).

The documentation for this class was generated from the following file:

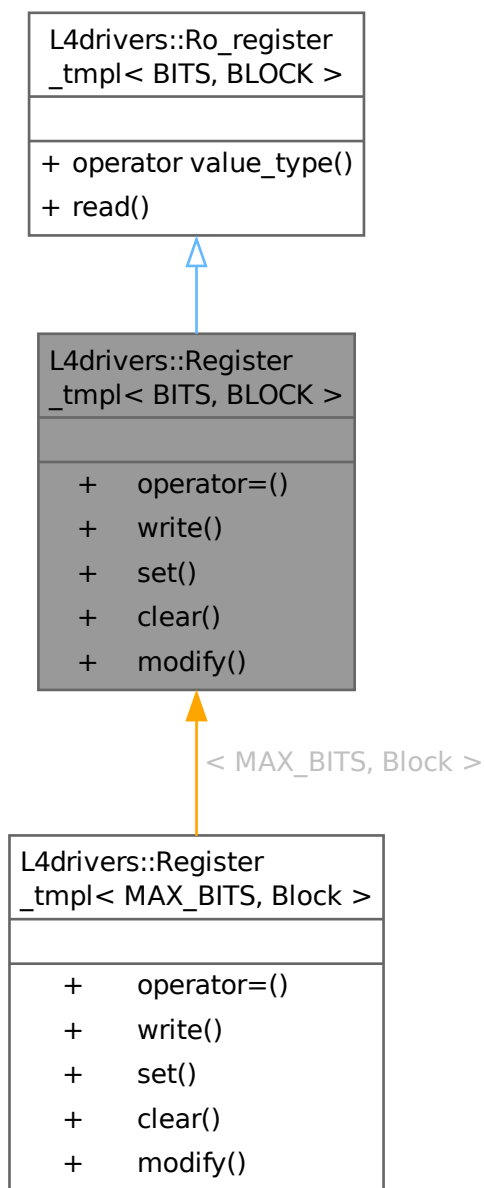
- `pkg/drivers-frst/include/hw_register_block`

15.278 L4drivers::Register_tmpl< BITS, BLOCK > Class Template Reference

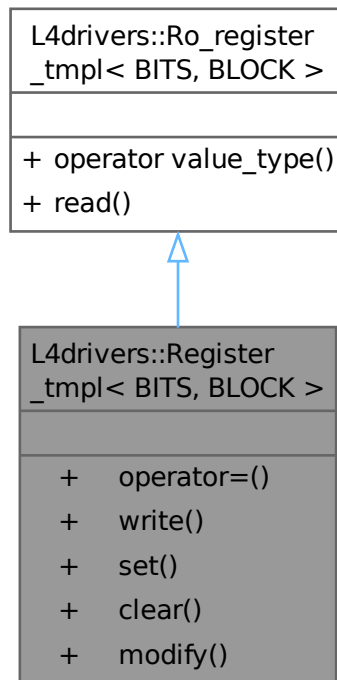
Single hardware register inside a [Register_block_base](#) interface.

```
#include <hw_register_block>
```

Inheritance diagram for L4drivers::Register_tmpl< BITS, BLOCK >:



Collaboration diagram for L4drivers::Register_tmpl< BITS, BLOCK >:



Public Member Functions

- `Register_tmpl & operator= (value_type val)`
write val into the hardware register.
- `void write (value_type val)`
write val into the hardware register.
- `value_type set (value_type set_bits)`
set bits in set_bits in the hardware register.
- `value_type clear (value_type clear_bits)`
clears bits in clear_bits in the hardware register.
- `value_type modify (value_type clear_bits, value_type set_bits)`
clears bits in clear_bits and sets bits in set_bits in the hardware register.

Public Member Functions inherited from **L4drivers::Ro_register_tmpl< BITS, BLOCK >**

- `operator value_type () const`
read the value from the hardware register.
- `value_type read () const`
read the value from the hardware register.

15.278.1 Detailed Description

template<unsigned BITS, typename BLOCK>
class L4drivers::Register_tmpl< BITS, BLOCK >

Single hardware register inside a [Register_block_base](#) interface.

Template Parameters

| | |
|--------------|--|
| <i>BITS</i> | The access width for the register in bits. |
| <i>BLOCK</i> | the type of the Register_block_base interface. |

Note

Objects of this type must be used only in temporary contexts not in global, class, or object scope.

Definition at line 237 of file [hw_register_block](#).

15.278.2 Member Function Documentation

15.278.2.1 clear()

```
template<unsigned BITS, typename BLOCK>
value_type L4drivers::Register_tmpl< BITS, BLOCK >::clear (
    value_type clear_bits) [inline]
```

clears bits in *clear_bits* in the hardware register.

Parameters

| | |
|-------------------|--|
| <i>clear_bits</i> | bits to be cleared within the hardware register. |
|-------------------|--|

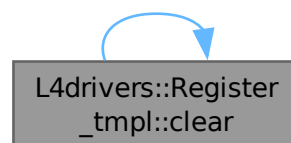
This is a read-modify-write function that does a logical and of the old value from the register with the negated value of *clear_bits*.

```
unsigned old_value = read();
write(old_value & ~clear_bits);
```

Definition at line 290 of file [hw_register_block](#).

Referenced by [L4drivers::Register_tmpl< MAX_BITS, Block >::clear\(\)](#).

Here is the caller graph for this function:



15.278.2.2 modify()

```
template<unsigned BITS, typename BLOCK>
value_type L4drivers::Register_tmpl< BITS, BLOCK >::modify (
    value_type clear_bits,
    value_type set_bits) [inline]
```

clears bits in *clear_bits* and sets bits in *set_bits* in the hardware register.

Parameters

| | |
|-------------------|--|
| <i>clear_bits</i> | bits to be cleared within the hardware register. |
| <i>set_bits</i> | bits to set in the hardware register. |

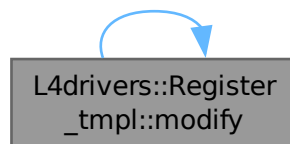
This is a read-modify-write function that first does a logical and of the old value from the register with the negated value of *clear_bits* and then does a logical or with *set_bits*.

```
unsigned old_value = read();
write((old_value & ~clear_bits) | set_bits);
```

Definition at line 308 of file [hw_register_block](#).

Referenced by [L4drivers::Register_tmpl< MAX_BITS, Block >::modify\(\)](#).

Here is the caller graph for this function:



15.278.2.3 operator=()

```
template<unsigned BITS, typename BLOCK>
Register_tmpl & L4drivers::Register_tmpl< BITS, BLOCK >::operator= (
    value_type val) [inline]
```

write *val* into the hardware register.

Parameters

| | |
|------------|--|
| <i>val</i> | the value to write into the hardware register. |
|------------|--|

Definition at line 252 of file [hw_register_block](#).

15.278.2.4 set()

```
template<unsigned BITS, typename BLOCK>
value_type L4drivers::Register_tmpl< BITS, BLOCK >::set (
    value_type set_bits) [inline]
```

set bits in *set_bits* in the hardware register.

Parameters

| | |
|-----------------|--|
| <i>set_bits</i> | bits to be set within the hardware register. |
|-----------------|--|

This is a read-modify-write function that does a logical or of the old value from the register with *set_bits*.

```
unsigned old_value = read();
write(old_value | set_bits);
```

Definition at line 274 of file [hw_register_block](#).

Referenced by [L4drivers::Register_tmpl< MAX_BITS, Block >::set\(\)](#).

Here is the caller graph for this function:



15.278.2.5 write()

```
template<unsigned BITS, typename BLOCK>
void L4drivers::Register_tmpl< BITS, BLOCK >::write (
    value_type val) [inline]
```

write *val* into the hardware register.

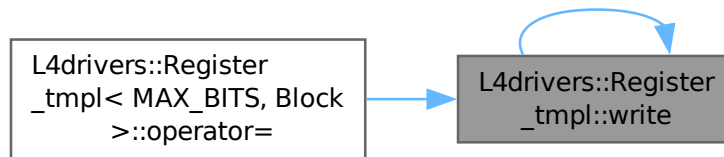
Parameters

| | |
|------------|--|
| <i>val</i> | the value to write into the hardware register. |
|------------|--|

Definition at line 259 of file [hw_register_block](#).

Referenced by [L4drivers::Register_tmpl< MAX_BITS, Block >::operator=\(\)](#), and [L4drivers::Register_tmpl< MAX_BITS, Block >::write\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

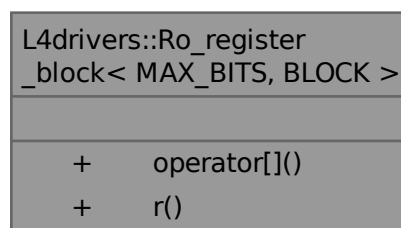
- pkg/drivers-frst/include/hw_register_block

15.279 L4drivers::Ro_register_block< MAX_BITS, BLOCK > Class Template Reference

Handles a reference to a read only register block of the given maximum access width.

```
#include <hw_register_block>
```

Collaboration diagram for L4drivers::Ro_register_block< MAX_BITS, BLOCK >:



Public Member Functions

- Ro_register [operator\[\]](#) (unsigned offset) const
Read only access to register at offset offset.
- template<unsigned BITS>
[Ro_register_tmpl](#)< BITS, Block > [r](#) (unsigned offset) const
Read only access to register at offset offset.

15.279.1 Detailed Description

```
template<unsigned MAX_BITS, typename BLOCK = Register_block_tmpl< Register_block_base<MAX_BITS> const >>
class L4drivers::Ro_register_block< MAX_BITS, BLOCK >
```

Handles a reference to a read only register block of the given maximum access width.

Template Parameters

| | |
|-----------------|---|
| <i>MAX_BITS</i> | Maximum access width for the registers in this block. |
| <i>BLOCK</i> | Type implementing the register accesses (read<>()), |

Provides read only access to registers in this block via [r<WIDTH>\(\)](#) and [operator\[\]\(\)](#).

Definition at line [404](#) of file [hw_register_block](#).

15.279.2 Member Function Documentation

15.279.2.1 operator[]()

```
template<unsigned MAX_BITS, typename BLOCK = Register_block_tmpl< Register_block_base<MAX_BITS> const >>
Ro_register L4drivers::Ro_register_block< MAX_BITS, BLOCK >::operator[] (
    unsigned offset) const [inline]
```

Read only access to register at offset *offset*.

Parameters

| | |
|---------------|--|
| <i>offset</i> | The offset of the register within the register file. |
|---------------|--|

Returns

register object allowing read only access with width *MAX_BITS*.

Definition at line [426](#) of file [hw_register_block](#).

15.279.2.2 r()

```
template<unsigned MAX_BITS, typename BLOCK = Register_block_tmpl< Register_block_base<MAX_BITS> const >>
template<unsigned BITS>
Ro_register_tmpl< BITS, Block > L4drivers::Ro_register_block< MAX_BITS, BLOCK >::r (
    unsigned offset) const [inline]
```

Read only access to register at offset *offset*.

Template Parameters

| | |
|-------------|--|
| <i>BITS</i> | the access width in bits for the register. |
|-------------|--|

Parameters

| | |
|---------------|--|
| <i>offset</i> | The offset of the register within the register file. |
|---------------|--|

Returns

register object allowing read only access with width *BITS*.

Definition at line 436 of file [hw_register_block](#).

The documentation for this class was generated from the following file:

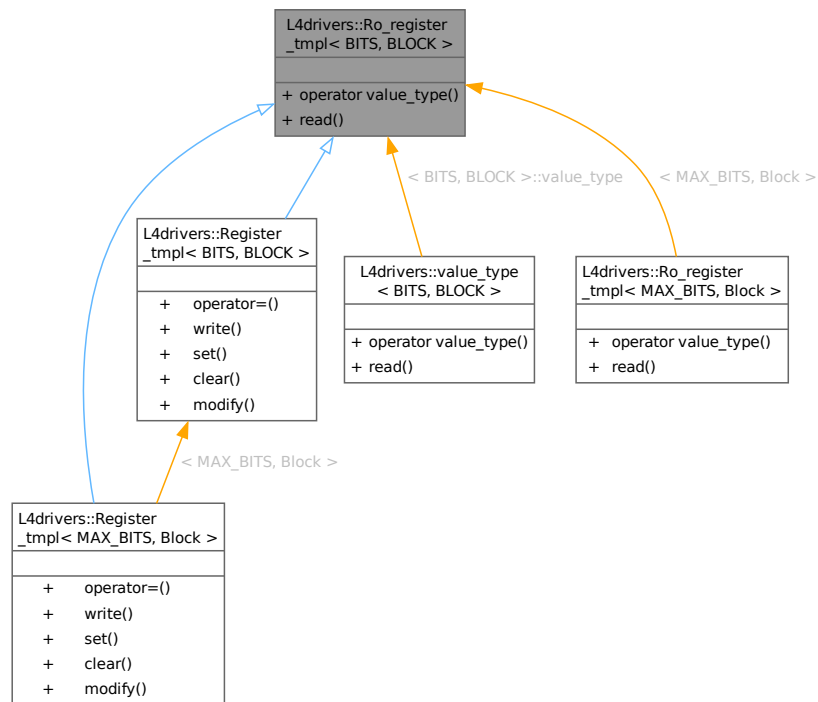
- pkg/drivers-frst/include/hw_register_block

15.280 L4drivers::Ro_register_tmpl< BITS, BLOCK > Class Template Reference

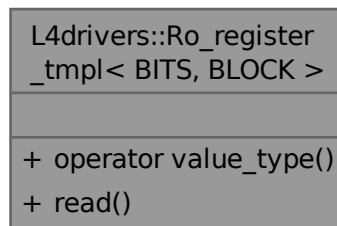
Single read only register inside a [Register_block_base](#) interface.

```
#include <hw_register_block>
```

Inheritance diagram for L4drivers::Ro_register_tmpl< BITS, BLOCK >:



Collaboration diagram for L4drivers::Ro_register_tmpl< BITS, BLOCK >:



Public Member Functions

- [operator value_type \(\)](#) const
read the value from the hardware register.
- [value_type read \(\)](#) const
read the value from the hardware register.

15.280.1 Detailed Description

```
template<unsigned BITS, typename BLOCK>
class L4drivers::Ro_register_tmpl< BITS, BLOCK >
```

Single read only register inside a [Register_block_base](#) interface.

Template Parameters

| | |
|--------------|---|
| <i>BITS</i> | The access with of the register in bits. |
| <i>BLOCK</i> | The type for the Register_block_base interface. |

Note

Objects of this type must be used only in temporary contexts not in global, class, or object scope.

Allows simple read only access to a hardware register.

Definition at line 201 of file [hw_register_block](#).

15.280.2 Member Function Documentation

15.280.2.1 operator value_type()

```
template<unsigned BITS, typename BLOCK>
L4drivers::Ro_register_tmpl< BITS, BLOCK >::operator value_type () const [inline]
```

read the value from the hardware register.

Returns

value read from the hardware register.

Definition at line 217 of file [hw_register_block](#).

15.280.2.2 read()

```
template<unsigned BITS, typename BLOCK>
value_type L4drivers::Ro_register_tmpl< BITS, BLOCK >::read () const [inline]
```

read the value from the hardware register.

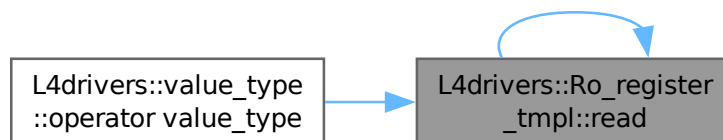
Returns

value from the hardware register.

Definition at line 224 of file [hw_register_block](#).

Referenced by [L4drivers::value_type< BITS, BLOCK >::operator value_type\(\)](#), and [L4drivers::value_type< BITS, BLOCK >::read\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

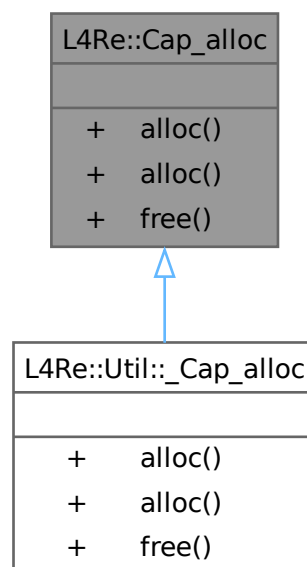
- `pkg/drivers-frst/include/hw_register_block`

15.281 L4Re::Cap_alloc Class Reference

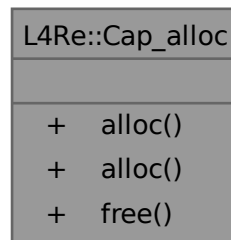
Capability allocator interface.

```
#include <cap_alloc>
```

Inheritance diagram for L4Re::Cap_alloc:



Collaboration diagram for L4Re::Cap_alloc:



Public Member Functions

- virtual [L4::Cap](#)< void > [alloc](#) () noexcept=0
Allocate a capability.
- template<typename T>
[L4::Cap](#)< T > [alloc](#) () noexcept
Allocate a capability.
- virtual void [free](#) ([L4::Cap](#)< void > cap, [l4_cap_idx_t](#) task=[L4_INVALID_CAP](#), unsigned unmap_↔
flags=[L4_FP_ALL_SPACES](#)) noexcept=0
Free a capability.

15.281.1 Detailed Description

Capability allocator interface.

Definition at line 30 of file [cap_alloc](#).

15.281.2 Member Function Documentation

15.281.2.1 alloc() [1/2]

```
template<typename T>
L4::Cap< T > L4Re::Cap_alloc::alloc () [inline], [noexcept]
```

Allocate a capability.

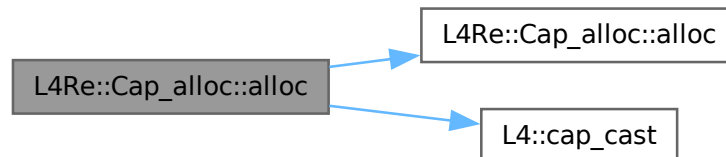
Returns

Capability of type T

Definition at line 55 of file [cap_alloc](#).

References [alloc\(\)](#), and [L4::cap_cast\(\)](#).

Here is the call graph for this function:

**15.281.2.2 alloc() [2/2]**

```
virtual L4::Cap< void > L4Re::Cap_alloc::alloc () [pure virtual], [noexcept]
```

Allocate a capability.

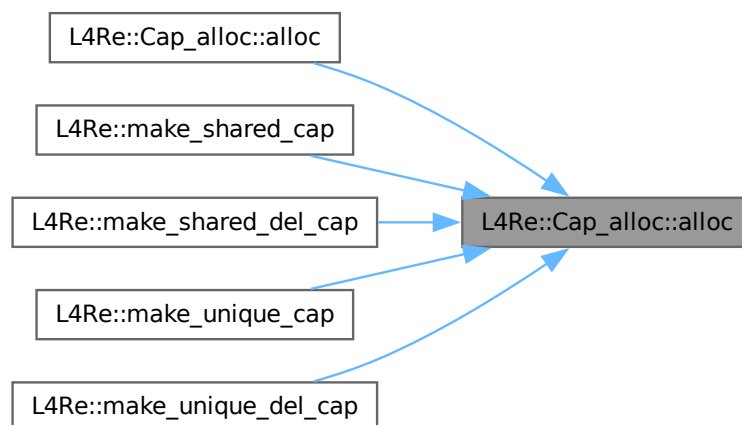
Returns

Capability of type void

Implemented in [L4Re::Util::_Cap_alloc](#), and [L4Re::Util::_Cap_alloc](#).

Referenced by [alloc\(\)](#), [L4Re::make_shared_cap\(\)](#), [L4Re::make_shared_del_cap\(\)](#), [L4Re::make_unique_cap\(\)](#), and [L4Re::make_unique_del_cap\(\)](#).

Here is the caller graph for this function:



15.281.2.3 free()

```
virtual void L4Re::Cap_alloc::free (
    L4::Cap< void > cap,
    l4_cap_idx_t task = L4_INVALID_CAP,
    unsigned unmap_flags = L4_FP_ALL_SPACES) [pure virtual], [noexcept]
```

Free a capability.

Parameters

| | |
|--------------------|---|
| <i>cap</i> | Capability to free. |
| <i>task</i> | If set, task to unmap the capability from. |
| <i>unmap_flags</i> | Flags for unmap, see l4_unmap_flags_t . |

Implemented in [L4Re::Util::_Cap_alloc](#).

References [L4Re::L4_FP_ALL_SPACES](#), and [L4_INVALID_CAP](#).

The documentation for this class was generated from the following file:

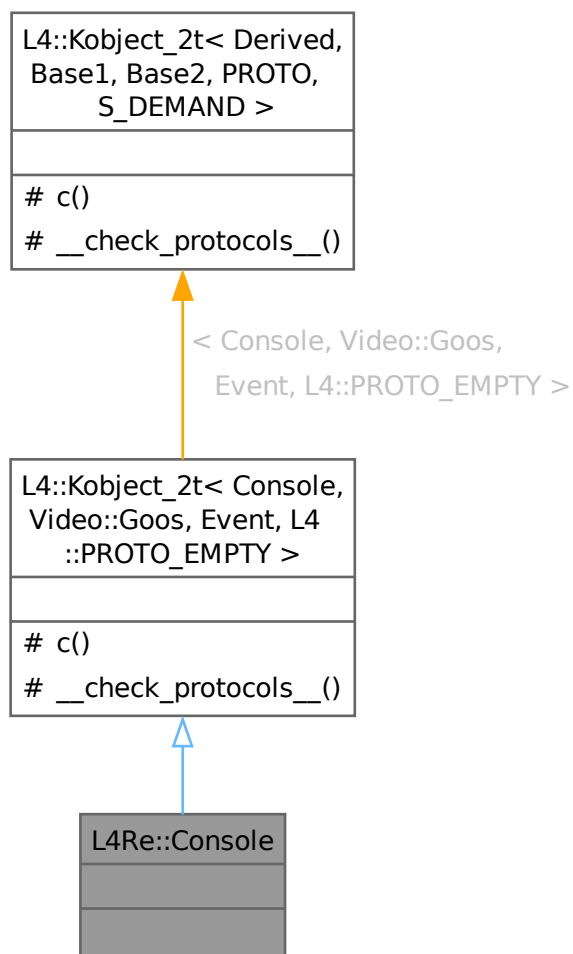
- [l4/re/cap_alloc](#)

15.282 L4Re::Console Class Reference

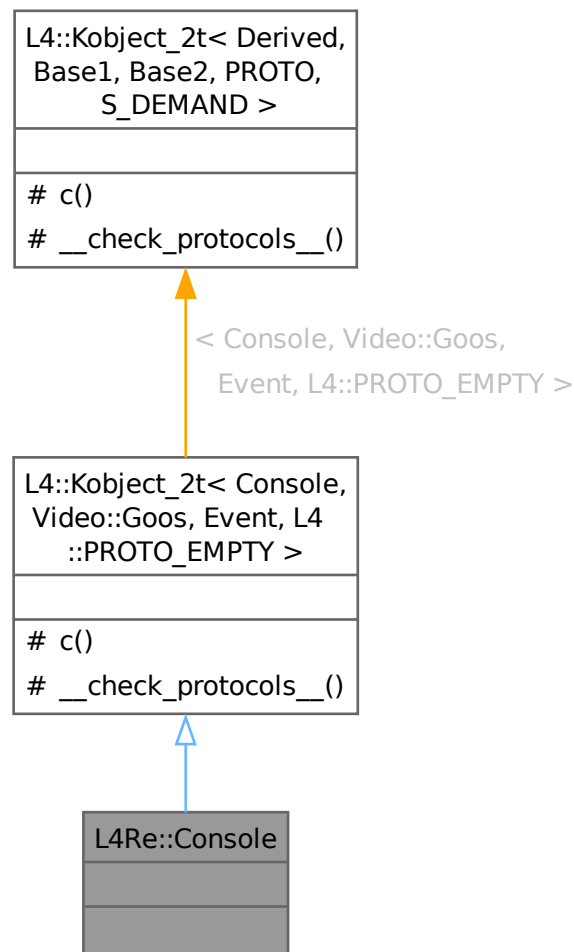
[Console](#) class.

```
#include <console>
```

Inheritance diagram for L4Re::Console:



Collaboration diagram for L4Re::Console:



Additional Inherited Members

Protected Types inherited from

L4::Kobject_2t< Console, Video::Goos, Event, L4::PROTO_EMPTY >

- typedef Console [Class](#)
The target interface type (inheriting from [Kobject_t](#)).
- typedef Typeid::Iface< PROTO, Console > [__Iface](#)
The interface description for the derived class.
- typedef Typeid::Merge_list< Typeid::Iface_list< [__Iface](#) >, Typeid::Merge_list< typename Video::Goos::_↔
[_Iface_list](#), typename Event::_Iface_list > > [__Iface_list](#)
The list of all RPC interfaces provided directly or through inheritance.

Protected Member Functions inherited from**[L4::Kobject_2t< Console, Video::Goos, Event, L4::PROTO_EMPTY >](#)**

- [L4::Cap< Class > c \(\)](#) const noexcept

*Get the capability to ourselves.***Static Protected Member Functions inherited from****[L4::Kobject_2t< Console, Video::Goos, Event, L4::PROTO_EMPTY >](#)**

- static void [__check_protocols__ \(\)](#) noexcept

*Helper to check for protocol conflicts.***15.282.1 Detailed Description**

[Console](#) class.

Definition at line 28 of file [console](#).

The documentation for this class was generated from the following file:

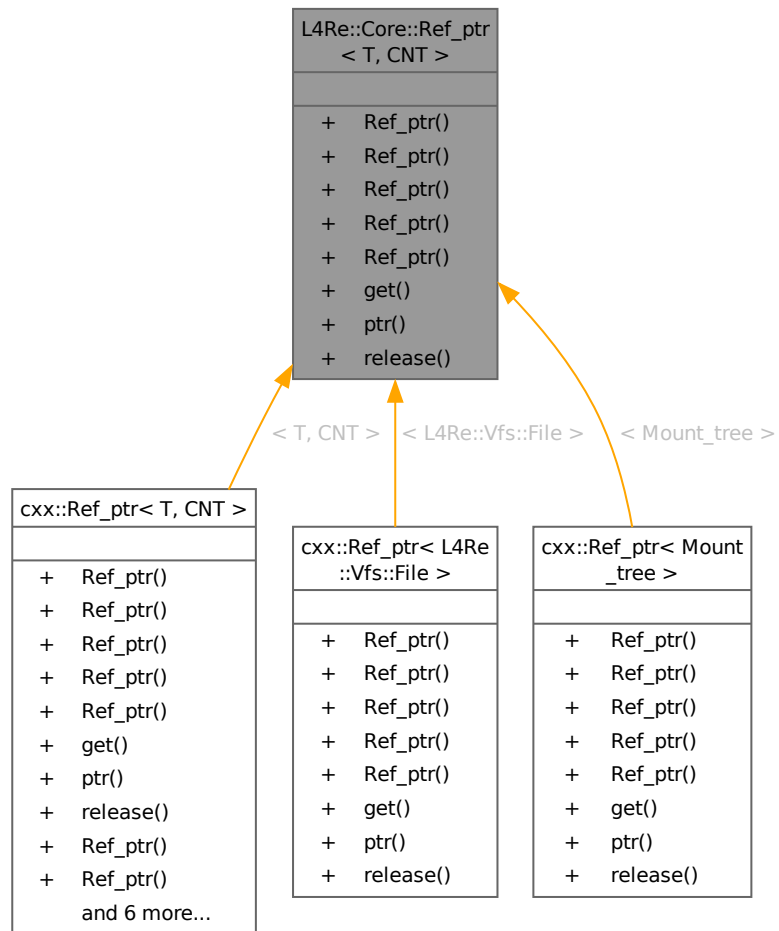
- [l4/re/console](#)

15.283 L4Re::Core::Ref_ptr< T, CNT > Class Template Reference

A reference-counting pointer with automatic cleanup.

```
#include <ref_ptr>
```

Inheritance diagram for L4Re::Core::Ref_ptr< T, CNT >:



Collaboration diagram for L4Re::Core::Ref_ptr< T, CNT >:

| L4Re::Core::Ref_ptr < T, CNT > | |
|-----------------------------------|-----------|
| | |
| + | Ref_ptr() |
| + | Ref_ptr() |
| + | Ref_ptr() |
| + | Ref_ptr() |
| + | Ref_ptr() |
| + | get() |
| + | ptr() |
| + | release() |

Public Member Functions

- **Ref_ptr** () noexcept
Default constructor creates a pointer with no managed object.
- **Ref_ptr** (Wp const &o) noexcept
Create a shared pointer from a weak pointer.
- **Ref_ptr** (decltype(nullptr) n) noexcept
allow creation from `nullptr`
- template<typename X>
Ref_ptr (X *o) noexcept
Create a shared pointer from a raw pointer.
- **Ref_ptr** (T *o, bool d) noexcept
Create a shared pointer from a raw pointer without creating a new reference.
- T * **get** () const noexcept
Return a raw pointer to the object this shared pointer points to.
- T * **ptr** () const noexcept
Return a raw pointer to the object this shared pointer points to.
- T * **release** () noexcept
Release the shared pointer without removing the reference.

15.283.1 Detailed Description

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
class L4Re::Core::Ref_ptr< T, CNT >
```

A reference-counting pointer with automatic cleanup.

Template Parameters

| | |
|------------|--|
| <i>T</i> | Type of object the pointer points to. |
| <i>CNT</i> | Type of management class that manages the life time of the object. |

This pointer is similar to the standard C++-11 `shared_ptr` but it does the reference counting directly in the object being pointed to, so that no additional management structures need to be allocated from the heap.

Classes that use this pointer type must implement two functions:

```
int remove_ref()
```

is called when a reference is removed and must return 0 when there are no further references to the object.

```
void add_ref()
```

is called when another `ref_ptr` to the object is created.

`Ref_obj` provides a simple implementation of this interface from which classes may inherit.

Definition at line 70 of file [ref_ptr](#).

15.283.2 Constructor & Destructor Documentation

15.283.2.1 `Ref_ptr()` [1/3]

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
cxx::Ref_ptr< T, CNT >::Ref_ptr (
    Wp const & o) [inline], [noexcept]
```

Create a shared pointer from a weak pointer.

Increases references.

Definition at line 88 of file [ref_ptr](#).

15.283.2.2 `Ref_ptr()` [2/3]

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
template<typename X>
cxx::Ref_ptr< T, CNT >::Ref_ptr (
    X * o) [inline], [explicit], [noexcept]
```

Create a shared pointer from a raw pointer.

In contrast to C++11 `shared_ptr` it is safe to use this constructor multiple times and have the same reference counter.

Definition at line 101 of file [ref_ptr](#).

15.283.2.3 Ref_ptr() [3/3]

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
cxx::Ref_ptr< T, CNT >::Ref_ptr (
    T * o,
    bool d) [inline], [noexcept]
```

Create a shared pointer from a raw pointer without creating a new reference.

Parameters

| | |
|----------|---|
| <i>o</i> | Pointer to the object. |
| <i>d</i> | Dummy parameter to select this constructor at compile time. The value may be true or false. |

This is the counterpart to [release\(\)](#).

Definition at line [114](#) of file [ref_ptr](#).

15.283.3 Member Function Documentation**15.283.3.1 get()**

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
T * cxx::Ref_ptr< T, CNT >::get () const [inline], [noexcept]
```

Return a raw pointer to the object this shared pointer points to.

This does not release the pointer or decrease the reference count.

Definition at line [121](#) of file [ref_ptr](#).

15.283.3.2 ptr()

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
T * cxx::Ref_ptr< T, CNT >::ptr () const [inline], [noexcept]
```

Return a raw pointer to the object this shared pointer points to.

This does not release the pointer or decrease the reference count.

Definition at line [127](#) of file [ref_ptr](#).

15.283.3.3 release()

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
T * cxx::Ref_ptr< T, CNT >::release () [inline], [noexcept]
```

Release the shared pointer without removing the reference.

Returns

A raw pointer to the managed object.

Definition at line [138](#) of file [ref_ptr](#).

The documentation for this class was generated from the following file:

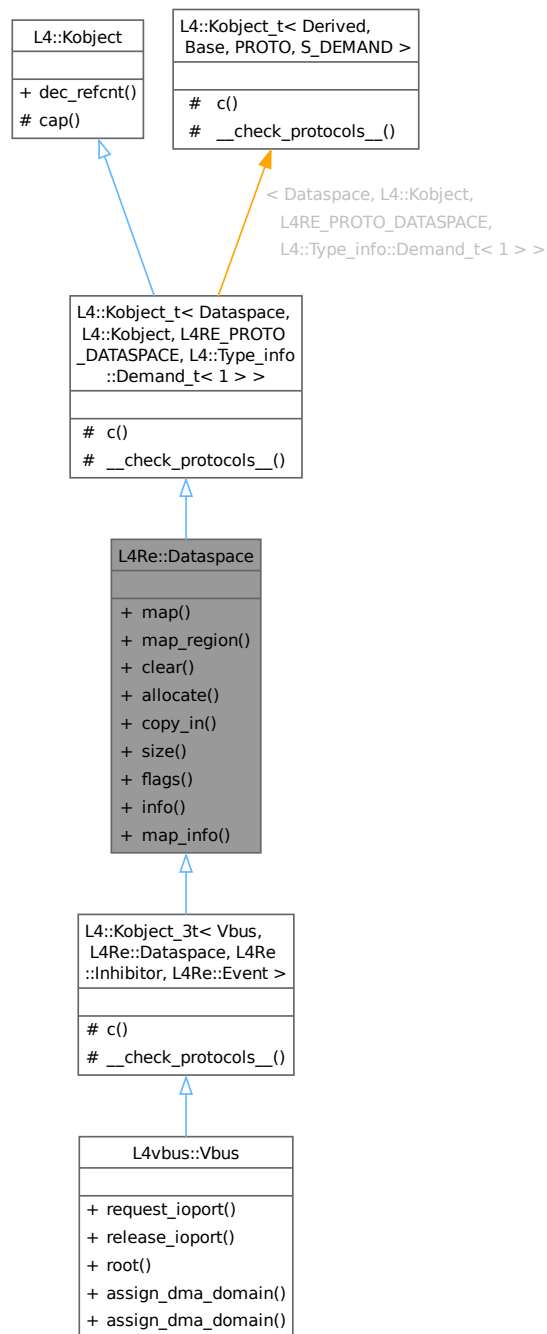
- [I4/cxx/ref_ptr](#)

15.284 L4Re::Dataspace Class Reference

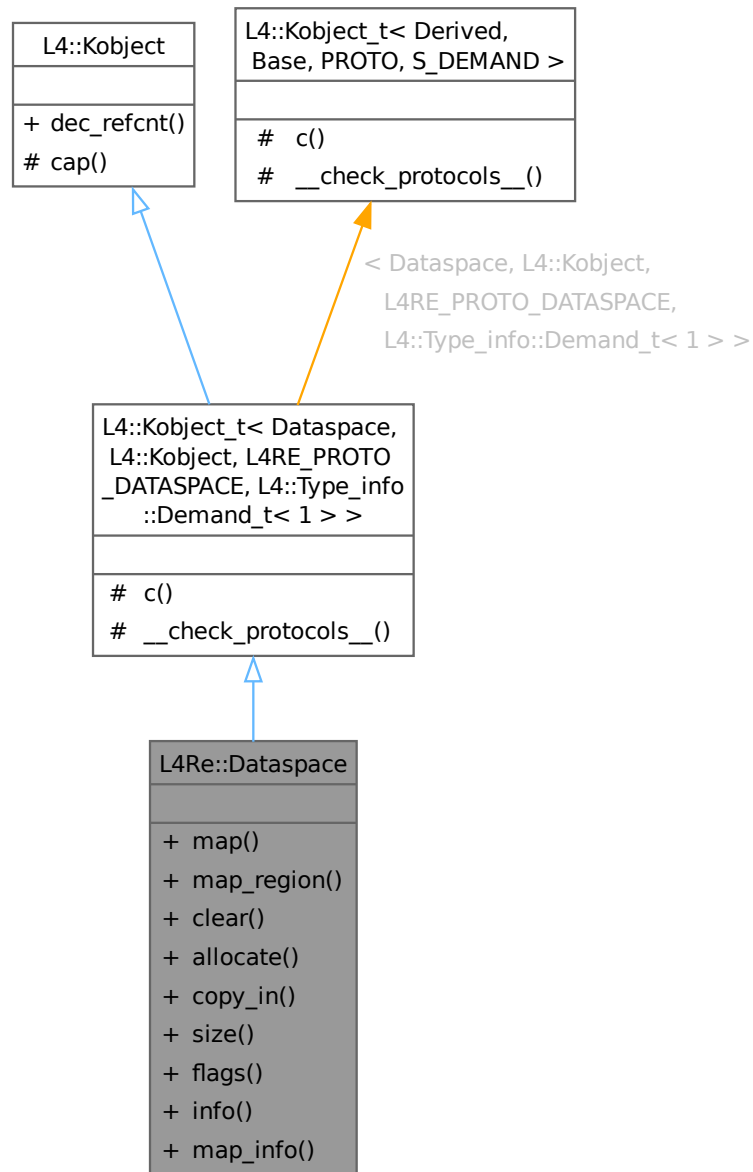
Interface for memory-like objects.

```
#include <dataspace>
```

Inheritance diagram for L4Re::Dataspace:



Collaboration diagram for L4Re::Dataspace:



Data Structures

- struct [F](#)
Dataspace flags definitions.
- struct [Stats](#)
Information about the dataspace.

Public Member Functions

- [l4_ret_t map](#) (Offset `offset`, Flags `flags`, Map_addr `local_addr`, Map_addr `min_addr`, Map_addr `max_addr`, `L4::Cap< L4::Task > dst=L4::Cap< L4::Task >::Invalid`) const noexcept

- Request a flexpage mapping from the dataspace.*

 - [l4_ret_t map_region](#) (Offset offset, Flags [flags](#), Map_addr min_addr, Map_addr max_addr, [L4::Cap](#)< [L4::Task](#) > dst=[L4::Cap](#)< [L4::Task](#) >::Invalid) const noexcept

Map a part of a dataspace into a local memory area.
- [l4_ret_t clear](#) (Offset offset, Size [size](#))

Clear parts of a dataspace.
- [l4_ret_t allocate](#) (Offset offset, Size [size](#))

Allocate a range in the dataspace.
- [l4_ret_t copy_in](#) (Offset dst_offs, [L4::lpc::Cap](#)< Dataspace > src, Offset src_offs, Size [size](#))

Copy contents from another dataspace.
- Size [size](#) () const noexcept

Get size of a dataspace.
- Flags [flags](#) () const noexcept

Get flags of the dataspace.
- [l4_ret_t info](#) ([Stats](#) *stats)

Get information on the dataspace.
- long [map_info](#) ([l4_addr_t](#) *start_addr, [l4_addr_t](#) *end_addr)

Get mapping range of dataspace.

Public Member Functions inherited from [L4::Kobject](#)

- [l4_msgtag_t dec_refcnt](#) ([l4_mword_t](#) diff, [l4_utcb_t](#) *utcb=[l4_utcb](#)())

Decrement the in kernel reference counter for the object.

Additional Inherited Members

Protected Types inherited from

[L4::Kobject_t](#)< [Dataspace](#), [L4::Kobject](#), [L4RE_PROTO_DATASPACE](#), [L4::Type_info::Demand_t](#)< 1 > >

- typedef Dataspace **Class**
- The target interface type (inheriting from [Kobject_t](#)).*
- typedef Typeid::Iface< PROTO, Dataspace > **__Iface**
- The interface description for the derived class.*
- typedef Typeid::Merge_list< Typeid::Iface_list< [__Iface](#) >, typename [L4::Kobject::__Iface_list](#) > **__Iface_list**
- The list of all RPC interfaces provided directly or through inheritance.*

Protected Member Functions inherited from

[L4::Kobject_t](#)< [Dataspace](#), [L4::Kobject](#), [L4RE_PROTO_DATASPACE](#), [L4::Type_info::Demand_t](#)< 1 > >

- [L4::Cap](#)< [Class](#) > **c** () const noexcept
- Get the capability to ourselves.*

Protected Member Functions inherited from [L4::Kobject](#)

- [l4_cap_idx_t cap](#) () const noexcept
- Return capability selector.*

Static Protected Member Functions inherited from**L4::Kobject_t< Dataspace, L4::Kobject, L4RE_PROTO_DATASPACE, L4::Type_info::Demand_t< 1 > >**

- static void **__check_protocols__** () noexcept
Helper to check for protocol conflicts.

15.284.1 Detailed Description

Interface for memory-like objects.

Dataspaces are a central abstraction provided by [L4Re](#). A dataspace is an abstraction for any thing that is available via usual memory access instructions. A dataspace can be a file, as well as the memory-mapped registers of a device, or anonymous memory, such as a heap.

The dataspace interface defines a set of methods that allow any kind of dataspace to be attached (mapped) to the virtual address space of an [L4](#) task and then be accessed via memory-access instructions. The [L4Re::Rm](#) interface can be used to attach a dataspace to a virtual address space of a task paged by a certain instance of a region map.

Include File

```
#include <l4/re/dataspace>
```

Examples

[examples/libs/l4re/c++/mem_alloc/ma+rm.cc](#), [examples/libs/l4re/c++/shared_ds/ds_clnt.cc](#), and [examples/libs/l4re/c++/shared](#)

Definition at line 50 of file [dataspace](#).

15.284.2 Member Function Documentation**15.284.2.1 allocate()**

```
l4_ret_t L4Re::Dataspace::allocate (
    Offset offset,
    Size size)
```

Allocate a range in the dataspace.

Parameters

| | |
|---------------|------------------------------------|
| <i>offset</i> | Offset in the dataspace, in bytes. |
| <i>size</i> | Size of the range, in bytes. |

Return values

| | |
|---------------|---------|
| <i>L4_EOK</i> | Success |
|---------------|---------|

| | |
|-------------------------|--|
| <code>-L4_ERANGE</code> | Given range is outside the dataspace. (A dataspace provider may also silently ignore areas outside the dataspace.) |
| <code>-L4_ENOMEM</code> | Not enough memory available. |
| <code>< 0</code> | IPC errors |

On success, at least the given range is guaranteed to be allocated. The dataspace manager may also allocate more memory due to page granularity.

The memory is allocated with the same rights as the dataspace capability.

References [allocate\(\)](#), [copy_in\(\)](#), and [size\(\)](#).

Referenced by [allocate\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.284.2.2 clear()

```

l4_ret_t L4Re::Dataspace::clear (
    Offset offset,
    Size size)

```

Clear parts of a dataspace.

Parameters

| | |
|---------------|-------------------------------------|
| <i>offset</i> | Offset within dataspace (in bytes). |
| <i>size</i> | Size of region to clear (in bytes). |

Return values

| | |
|--------------------------|--|
| ≥ 0 | Success. |
| <code>-L4_ERANGE</code> | Given range is outside the dataspace. (A dataspace provider may also silently ignore areas outside the dataspace.) |
| <code>-L4_EACCESS</code> | No <code>L4_CAP_FPAGE_W</code> right on dataspace capability. |
| < 0 | IPC errors |

Zeroes out the memory. Depending on the type of memory the memory could also be deallocated and replaced by a shared zero-page.

References [clear\(\)](#), and [size\(\)](#).

Referenced by [clear\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**15.284.2.3 copy_in()**

```

l4_ret_t L4Re::Dataspace::copy_in (
    Offset dst_offs,
    L4::Ipc::Cap< Dataspace > src,
    Offset src_offs,
    Size size)

```

Copy contents from another dataspace.

Parameters

| | |
|-----------------|----------------------------------|
| <i>dst_offs</i> | Offset in destination dataspace. |
| <i>src</i> | Source dataspace to copy from. |
| <i>src_offs</i> | Offset in the source dataspace. |
| <i>size</i> | Size to copy (in bytes). |

Return values

| | |
|--------------------|---|
| <i>L4_EOK</i> | Success |
| <i>-L4_EACCESS</i> | No L4_CAP_FPAGE_W right on the destination dataspace. |
| <i>-L4_EINVAL</i> | Invalid parameter supplied. |
| <i>< 0</i> | IPC errors |

The copy operation may use copy-on-write mechanisms. The operation may also fail if both dataspaces are not from the same dataspace manager or the dataspace managers do not cooperate.

References [flags\(\)](#), and [size\(\)](#).

Referenced by [allocate\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**15.284.2.4 flags()**

```
Dataspace::Flags Dataspace::flags () const [noexcept]
```

Get flags of the dataspace.

Return values

| | |
|----------|------------------------|
| ≥ 0 | Flags of the dataspace |
| < 0 | IPC errors |

See also

[L4Re::Dataspace::F::Flags](#)

Definition at line 112 of file [dataspace_impl.h](#).

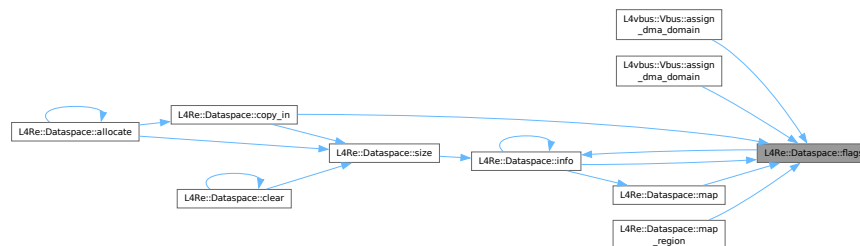
References [L4Re::Dataspace::Stats::flags](#), and [info\(\)](#).

Referenced by [L4vbus::Vbus::assign_dma_domain\(\)](#), [L4vbus::Vbus::assign_dma_domain\(\)](#), [copy_in\(\)](#), [info\(\)](#), [map\(\)](#), and [map_region\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**15.284.2.5 info()**

```
l4_ret_t L4Re::Dataspace::info (
    Stats * stats)
```

Get information on the dataspace.

Parameters

| | | |
|------------------|--------------------|-----------------------|
| <code>out</code> | <code>stats</code> | Dataspace information |
|------------------|--------------------|-----------------------|

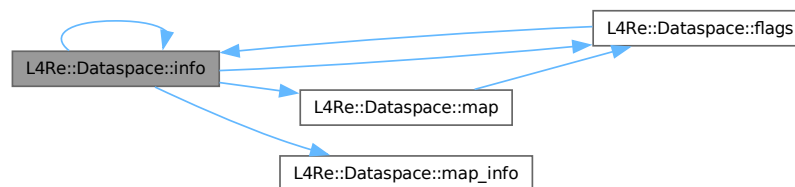
Return values

| | |
|--------------------|------------|
| <code>0</code> | Success |
| <code><0</code> | IPC errors |

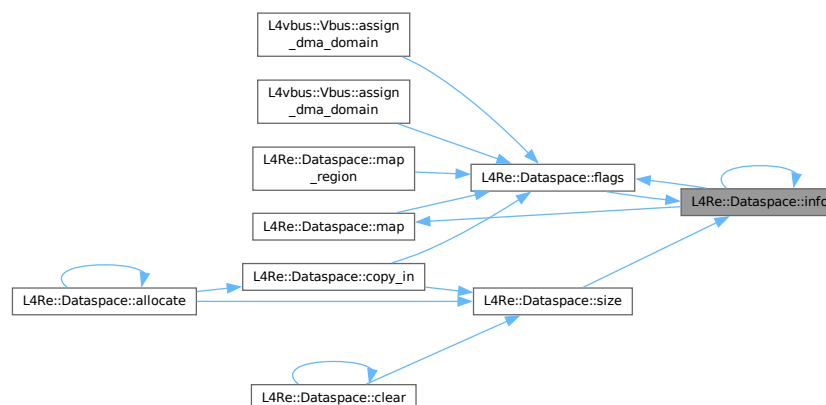
References [flags\(\)](#), [info\(\)](#), [L4_RPC](#), [L4_RPC_NF](#), [map\(\)](#), and [map_info\(\)](#).

Referenced by [flags\(\)](#), [info\(\)](#), and [size\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**15.284.2.6 map()**

```

l4_ret_t Dataspace::map (
    Dataspace::Offset offset,

```

```

Dataspace::Flags flags,
Dataspace::Map_addr local_addr,
Dataspace::Map_addr min_addr,
Dataspace::Map_addr max_addr,
L4::Cap< L4::Task > dst = L4::Cap<L4::Task>::Invalid) const [noexcept]

```

Request a flexpage mapping from the dataspace.

Parameters

| | |
|-------------------|---|
| <i>offset</i> | Offset to start within dataspace |
| <i>flags</i> | Dataspace flags, see L4Re::Dataspace::F::Flags . |
| <i>local_addr</i> | Local address to map to. |
| <i>min_addr</i> | Defines start of receive window. (Rounded down to page size.) |
| <i>max_addr</i> | Defines end of receive window. (Rounded up to page size.) |
| <i>dst</i> | Optional destination task of the mapping. If invalid, the callers task is implicitly the destination. |

Return values

| | |
|-------------------|---|
| <i>L4_EOK</i> | Success |
| <i>-L4_ERANGE</i> | Invalid offset. |
| <i>-L4_EPERM</i> | Insufficient permission to map with requested rights. |
| <i><0</i> | IPC errors |

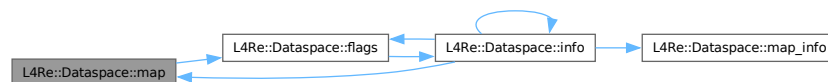
The map call will attempt to map the largest possible flexpage that covers the given local address and still fits into the region defined by *min_addr* and *max_addr*. If the given region is invalid or does not overlap the local address, the smallest valid page size is used.

Definition at line 84 of file [dataspace_impl.h](#).

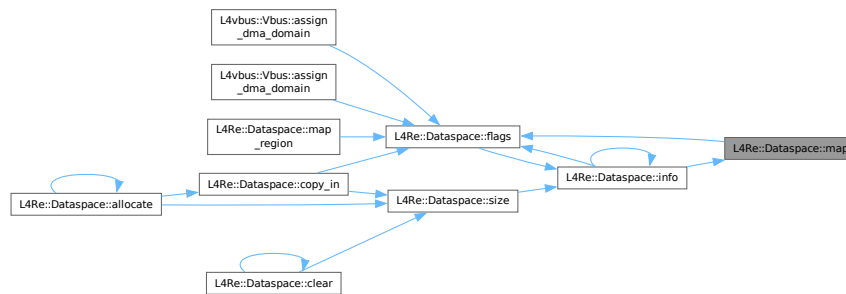
References [flags\(\)](#), and [L4_LOG2_PAGESIZE](#).

Referenced by [info\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.284.2.7 map_info()

```
long L4Re::Dataspace::map_info (
    l4_addr_t * start_addr,
    l4_addr_t * end_addr) [inline]
```

Get mapping range of dataspace.

In case of a MMU-less system, the dataspace must be mapped at the correct address in the task because virtual and physical address must match. This method returns the start and end address of the physically contiguous buffer backing the dataspace.

On MMU-enabled system any page aligned address is permissible. On such systems the method is just a stub.

Parameters

| | | |
|-----|-------------------|---------------------------------------|
| out | <i>start_addr</i> | Start address of dataspace. |
| out | <i>end_addr</i> | End address (inclusive) of dataspace. |

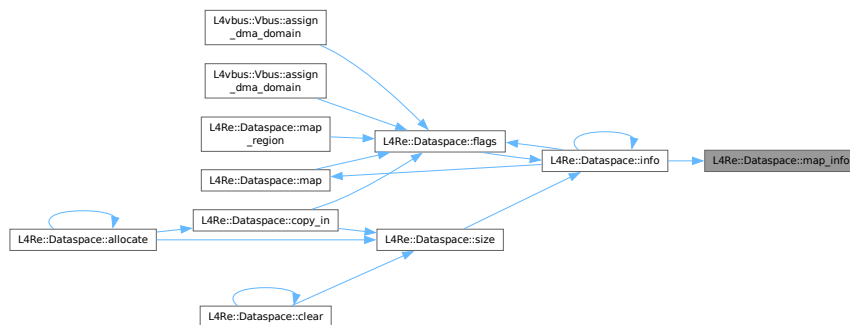
Return values

| | |
|-----------|---|
| >0 | Start/end address have been set and need to be obeyed. |
| 0 | No constraint of mapping address. |
| -L4_EPERM | Cannot infer mapping address. Dataspace not mappable. |
| <0 | IPC errors. |

Definition at line 306 of file [dataspace](#).

Referenced by [info\(\)](#).

Here is the caller graph for this function:



15.284.2.8 map_region()

```

14_ret_t Dataspace::map_region (
    Dataspace::Offset offset,
    Dataspace::Flags flags,
    Dataspace::Map_addr min_addr,
    Dataspace::Map_addr max_addr,
    L4::Cap< L4::Task > dst = L4::Cap<L4::Task>::Invalid) const [noexcept]

```

Map a part of a dataspace into a local memory area.

Parameters

| | |
|-----------------|---|
| <i>offset</i> | Offset to start within dataspace. |
| <i>flags</i> | Dataspace flags, see L4Re::Dataspace::F::Flags . |
| <i>min_addr</i> | (Inclusive) start of the receive area. |
| <i>max_addr</i> | (Exclusive) end of receive area. |
| <i>dst</i> | Optional destination task of the mapping. If invalid, the callers task is implicitly the destination. |

Return values

| | |
|-------------------|---|
| <i>L4_EOK</i> | Success |
| <i>-L4_ERANGE</i> | Invalid offset or receive area larger than the dataspace. |
| <i>-L4_EPERM</i> | Insufficient permission to map with requested rights. |
| <i><0</i> | IPC errors |

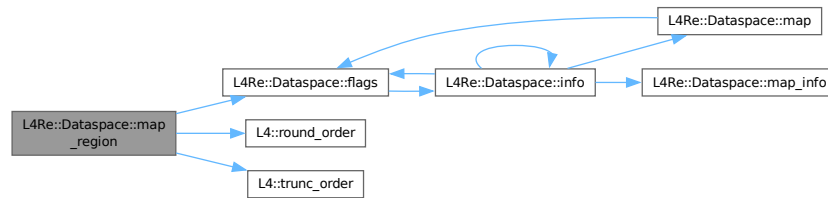
This is a convenience function which maps flexpages consecutively into the given memory area in the local task. The area is expected to be filled completely. If the dataspace is not large enough to provide the mappings for the entire size of the area, then an error is returned. Mappings may or may not have been already established at that point.

offset and min_addr are rounded down to the next L4_PAGESIZE boundary when necessary. max_addr is rounded up to the page boundary. If the resulting maximum address is less or equal than the minimum address, then the function is a noop.

Definition at line 45 of file [dataspace_impl.h](#).

References [flags\(\)](#), [L4_LOG2_PAGESIZE](#), [L4_UNLIKELY](#), [L4::round_order\(\)](#), and [L4::trunc_order\(\)](#).

Here is the call graph for this function:



15.284.2.9 size()

```
Dataspace::Size Dataspace::size () const [noexcept]
```

Get size of a dataspace.

Returns

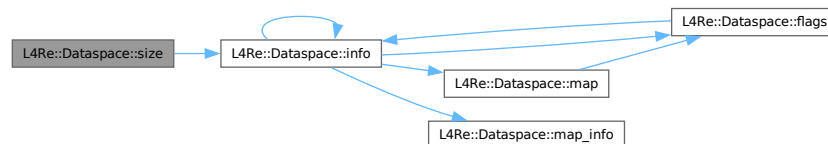
Size of the dataspace in bytes.

Definition at line 100 of file [dataspace_impl.h](#).

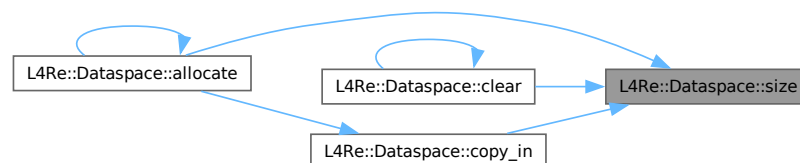
References [info\(\)](#), and [L4Re::Dataspace::Stats::size](#).

Referenced by [allocate\(\)](#), [clear\(\)](#), and [copy_in\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following files:

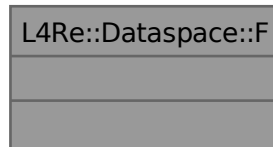
- [l4/re/dataspace](#)
- [l4/re/impl/dataspace_impl.h](#)

15.285 L4Re::Dataspace::F Struct Reference

[Dataspace](#) flags definitions.

```
#include <dataspace>
```

Collaboration diagram for L4Re::Dataspace::F:



Public Types

- enum { [Caching_shift](#) = 4 }
- enum [Flags](#) {

[R](#) = L4_FPAGE_RO , [Ro](#) = L4_FPAGE_RO , [RW](#) = L4_FPAGE_RW , [W](#) = L4_FPAGE_W ,

[X](#) = L4_FPAGE_X , [RX](#) = L4_FPAGE_RX , [RWX](#) = L4_FPAGE_RWX , [Rights_mask](#) = 0x0f ,

[Normal](#) = 0x00 , [Cacheable](#) = Normal , [Bufferable](#) = 0x10 , [Uncacheable](#) = 0x20 ,

[Caching_mask](#) = 0x30 }

[Flags](#) for map operations.

15.285.1 Detailed Description

[Dataspace](#) flags definitions.

Definition at line 57 of file [dataspace](#).

15.285.2 Member Enumeration Documentation

15.285.2.1 anonymous enum

anonymous enum

Enumerator

| | |
|---------------|-------------------------------|
| Caching_shift | shift value for caching flags |
|---------------|-------------------------------|

Definition at line 59 of file [dataspace](#).

15.285.2.2 Flags

enum [L4Re::Dataspace::F::Flags](#)

[Flags](#) for map operations.

A dataspace implementation must check the requested flags during the map and other operations against the dataspace rights.

Enumerator

| | |
|--------------|---|
| R | Request read-only mapping. |
| Ro | Request read-only mapping. |
| RW | Request read-write mapping. |
| W | Request write-only mapping. |
| X | Request execute-only mapping. |
| RX | Request read-execute mapping. |
| RWX | Request read-write-execute mapping. |
| Rights_mask | All rights bits available for mappings. |
| Normal | Request normal (cached) memory mapping. This is the default if no other cache-related flag was specified. |
| Cacheable | Request normal memory mapping. |
| Bufferable | Request bufferable (write buffered) mappings. |
| Uncacheable | Request uncacheable memory mappings. |
| Caching_mask | Mask for caching flags. |

Definition at line 70 of file [dataspace](#).

The documentation for this struct was generated from the following file:

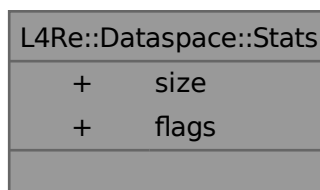
- [l4/re/dataspace](#)

15.286 L4Re::Dataspace::Stats Struct Reference

Information about the dataspace.

```
#include <dataspace>
```

Collaboration diagram for L4Re::Dataspace::Stats:



Data Fields

- Size **size**
size
- Flags **flags**
flags

15.286.1 Detailed Description

Information about the dataspace.

Definition at line 128 of file [dataspace](#).

The documentation for this struct was generated from the following file:

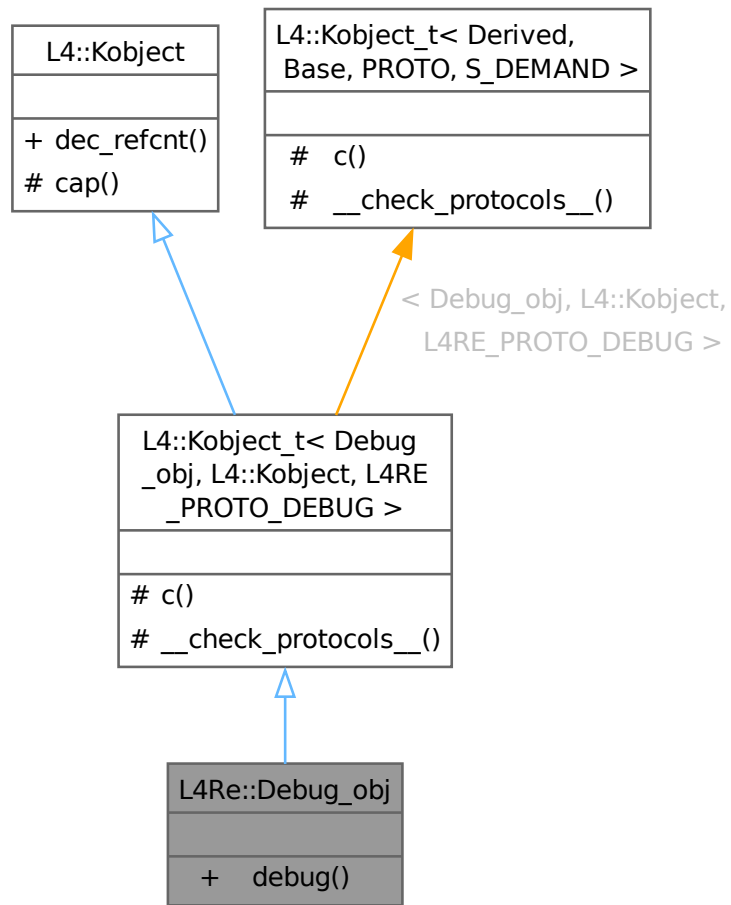
- [l4/re/dataspace](#)

15.287 L4Re::Debug_obj Class Reference

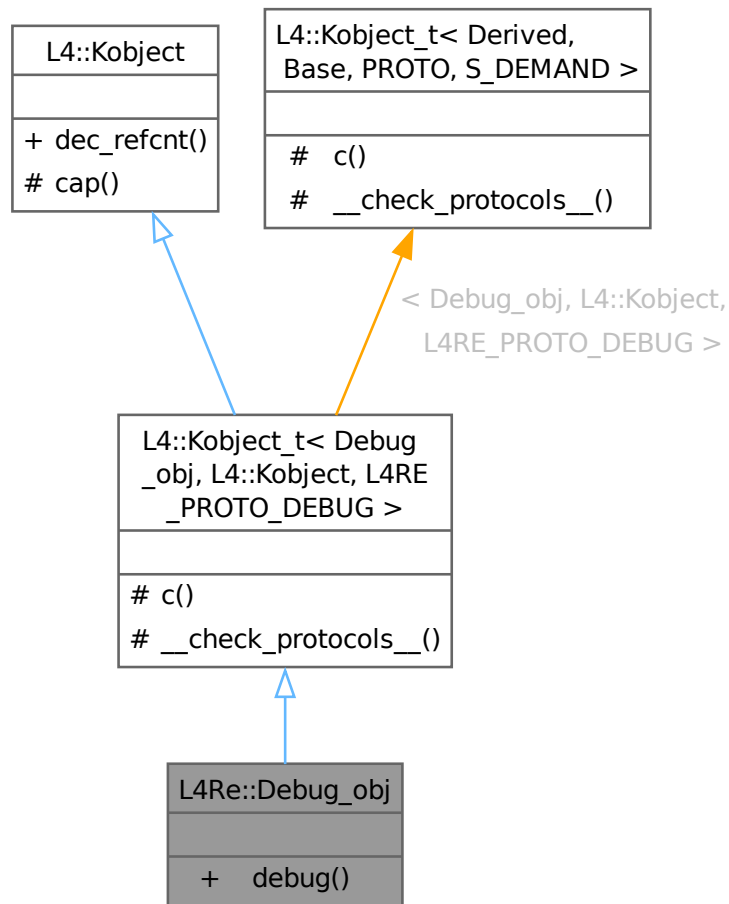
Debug interface.

```
#include <debug>
```

Inheritance diagram for L4Re::Debug_obj:



Collaboration diagram for L4Re::Debug_obj:



Public Member Functions

- [l4_ret_t debug](#) (unsigned long function)
Debug call.

Public Member Functions inherited from [L4::Kobject](#)

- [l4_msgtag_t dec_refcnt](#) ([l4_mword_t](#) diff, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#))
Decrement the in kernel reference counter for the object.

Additional Inherited Members

Protected Types inherited from

[L4::Kobject_t< Debug_obj, L4::Kobject, L4RE_PROTO_DEBUG >](#)

- typedef `Debug_obj` **Class**

The target interface type (inheriting from [Kobject_t](#)).

- typedef Typeid::Iface< PROTO, Debug_obj > **__Iface**

The interface description for the derived class.

- typedef Typeid::Merge_list< Typeid::Iface_list< [__Iface](#) >, typename L4::Kobject::__Iface_list > **__Iface_list**

The list of all RPC interfaces provided directly or through inheritance.

Protected Member Functions inherited from [L4::Kobject_t< Debug_obj, L4::Kobject, L4RE_PROTO_DEBUG >](#)

- [L4::Cap< Class > c](#) () const noexcept

Get the capability to ourselves.

Protected Member Functions inherited from [L4::Kobject](#)

- [l4_cap_idx_t cap](#) () const noexcept

Return capability selector.

Static Protected Member Functions inherited from [L4::Kobject_t< Debug_obj, L4::Kobject, L4RE_PROTO_DEBUG >](#)

- static void [__check_protocols__](#) () noexcept

Helper to check for protocol conflicts.

15.287.1 Detailed Description

Debug interface.

See also

[Debugging API](#) .

Definition at line 40 of file [debug](#).

15.287.2 Member Function Documentation

15.287.2.1 debug()

```
l4_ret_t L4Re::Debug_obj::debug (
    unsigned long function)
```

Debug call.

Parameters

| | |
|-----------------|-------------------|
| <i>function</i> | Function to call. |
|-----------------|-------------------|

Returns

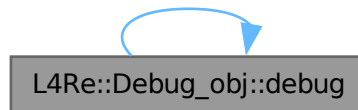
- L4_EOK
- IPC errors

An object can provide a number of debug functions, each identified by some integer. This method is used to fan out to these functions from a common entry point.

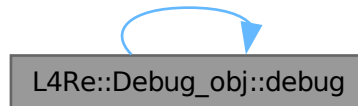
References [debug\(\)](#).

Referenced by [debug\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

- l4/re/[debug](#)

15.288 L4Re::Default_event_payload Struct Reference

Default event stream payload.

```
#include <event>
```

Collaboration diagram for L4Re::Default_event_payload:

| L4Re::Default_event_payload | |
|-----------------------------|-----------|
| + | type |
| + | code |
| + | value |
| + | stream_id |
| | |

Data Fields

- unsigned short **type**
Type of event.
- unsigned short **code**
Code of event.
- int **value**
Value of event.
- [l4_umword_t](#) **stream_id**
Stream ID.

15.288.1 Detailed Description

Default event stream payload.

Definition at line [234](#) of file [event](#).

The documentation for this struct was generated from the following file:

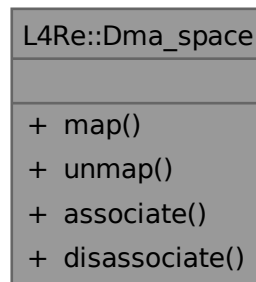
- [l4/re/event](#)

15.289 L4Re::Dma_space Class Reference

Managed DMA Address Space.

```
#include <dma_space>
```


Collaboration diagram for L4Re::Dma_space:



Public Types

- enum [Direction](#) { [Bidirectional](#) , [To_device](#) , [From_device](#) , [None](#) }
Direction of the DMA transfers.
- enum [Attribute](#) { [No_sync](#) }
Attributes used for the memory region during the transfer.
- enum [Space_attrib](#) { [Coherent](#) , [Phys_space](#) }
Attributes assigned to the DMA space when associated with a specific device.
- typedef [l4_uint64_t](#) [Dma_addr](#)
Data type for DMA addresses.
- typedef [L4::Types::Flags](#)< [Attribute](#) > [Attributes](#)
Attributes for DMA mappings.
- typedef [L4::Types::Flags](#)< [Space_attrib](#) > [Space_attribs](#)
Attributes used when configuring the DMA space.

Public Member Functions

- [l4_ret_t](#) [map](#) ([L4::lpc::Cap](#)< [L4Re::Dataspace](#) > src, [L4Re::Dataspace::Offset](#) offset, [L4::lpc::In_out](#)< [l4_size_t](#) * > size, [Attributes](#) attrs, [Direction](#) dir, [Dma_addr](#) *dma_addr)
Map the given part of this data space into the DMA address space.
- [l4_ret_t](#) [unmap](#) ([Dma_addr](#) dma_addr, [l4_size_t](#) size, [Attributes](#) attrs, [Direction](#) dir)
Unmap the given part of this data space from the DMA address space.
- [l4_ret_t](#) [associate](#) ([L4::lpc::Opt](#)< [L4::lpc::Cap](#)< [L4::Task](#) > > dma_task, [Space_attribs](#) attr)
Associate a (kernel) DMA space for a device to this DMA space.
- [l4_ret_t](#) [disassociate](#) ()
Disassociate the (kernel) DMA space from this DMA space.

15.289.1 Detailed Description

Managed DMA Address Space.

A managed [Dma_space](#) represents the [L4Re](#) abstraction of an DMA address space of one or several devices. Devices are assigned to a managed [Dma_space](#) by binding the [Dma_space](#) to the respective DMA domain (see [L4vbus::Vbus::assign_dma_domain\(\)](#)), which might link the [Dma_space](#) with a kernel [DMA space](#). Note that several DMA domains can be bound to the same [Dma_space](#). Whenever a device needs direct access to parts of an [L4Re::Dataspace](#), that part of the data space must be mapped to the managed [Dma_space](#) that is assigned to that device. Binding to DMA domains must happen before mapping. After the DMA accesses to the memory are finished the memory must be unmapped from the device's DMA address space.

Mapping to a managed DMA address space, using [map\(\)](#), makes the given parts of the data space visible to the associated device at the returned DMA address. As long as the memory is mapped into a DMA space it is 'pinned' and cannot be subject to dynamic memory management such as swapping. Additionally, [map\(\)](#) is responsible for the necessary syncing operations before the DMA.

[unmap\(\)](#) is the reverse operation to [map\(\)](#) and unmaps the given data-space part for the DMA address space. [unmap\(\)](#) is responsible for the necessary sync operations after the DMA.

Definition at line 52 of file [dma_space](#).

15.289.2 Member Typedef Documentation

15.289.2.1 Attributes

```
typedef L4::Types::Flags<Attribute> L4Re::Dma_space::Attributes
```

[Attributes](#) for DMA mappings.

See also

[Attribute](#)

Definition at line 97 of file [dma_space](#).

15.289.3 Member Enumeration Documentation

15.289.3.1 Attribute

```
enum L4Re::Dma_space::Attribute
```

[Attributes](#) used for the memory region during the transfer.

See also

[Attributes](#)

Enumerator

| | |
|---------|---|
| No_sync | Do not sync the memory hierarchy. When this flag is <i>not set</i> (default) the memory region shall be made coherent to the point-of-coherency of the device associated with this Dma_space . When using this attribute the client is responsible for syncing the memory hierarchy for DMA. This can either be done using the cache API or by another map() or unmap() operation of the same part of the data space (without the No_sync attribute). |
|---------|---|

Definition at line 76 of file [dma_space](#).

15.289.3.2 Direction

```
enum L4Re::Dma_space::Direction
```

[Direction](#) of the DMA transfers.

Enumerator

| | |
|---------------|--|
| Bidirectional | device reads and writes to the memory |
| To_device | device reads the memory |
| From_device | device writes to the memory |
| None | device is coherently connected to the memory |

Definition at line 64 of file [dma_space](#).

15.289.3.3 Space_attrib

```
enum L4Re::Dma_space::Space_attrib
```

[Attributes](#) assigned to the DMA space when associated with a specific device.

See also

[Space_attribs](#)

Enumerator

| | |
|------------|---|
| Coherent | The device is connected coherently with the cache. This means that the map() and unmap() do not need to sync CPU caches before and after DMA. |
| Phys_space | The DMA space has no DMA task assigned and uses the CPUs physical memory. |

Definition at line 104 of file [dma_space](#).

15.289.4 Member Function Documentation

15.289.4.1 associate()

```
l4_ret_t L4Re::Dma_space::associate (
    L4::Ipc::Opt< L4::Ipc::Cap< L4::Task > > dma_task,
    Space_attribs attr)
```

Associate a (kernel) [DMA space](#) for a device to this [Dma_space](#).

Parameters

| | | |
|----|-----------------|---|
| in | <i>dma_task</i> | The (kernel) DMA space used for the device that shall be associated with this DMA space. In case no IOMMU is present or configured, the <i>dma_task</i> might be an invalid capability when L4Re::Dma_space::Phys_space is set in <i>attr</i> , in this case the CPUs physical memory is used as DMA address space. |
| in | <i>attr</i> | Attributes for this DMA space. See L4Re::Dma_space::Space_attrb . |

Return values

| | |
|-------------------|---|
| <i>L4_EOK</i> | Operation successful. |
| <i>-L4_EPERM</i> | Insufficient permissions; see precondition. |
| <i>-L4_EINVAL</i> | |
| <i>-L4_ENOENT</i> | |

Precondition

The invoked [Dma_space](#) capability must have the permission [L4_CAP_FPAGE_W](#).

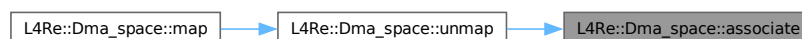
References [disassociate\(\)](#).

Referenced by [unmap\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.289.4.2 disassociate()

```
l4_ret_t L4Re::Dma_space::disassociate ()
```

Disassociate the (kernel) [DMA space](#) from this [Dma_space](#).

Return values

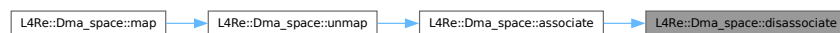
| | |
|-------------------|---|
| <i>L4_EOK</i> | Operation successful. |
| <i>-L4_EPERM</i> | Insufficient permissions; see precondition. |
| <i>-L4_ENOENT</i> | |

Precondition

The invoked [Dma_space](#) capability must have the permission [L4_CAP_FPAGE_W](#).

Referenced by [associate\(\)](#).

Here is the caller graph for this function:



15.289.4.3 map()

```
l4_ret_t L4Re::Dma_space::map (
    L4::Ipc::Cap< L4Re::Dataspace > src,
    L4Re::Dataspace::Offset offset,
    L4::Ipc::In_out< l4_size_t * > size,
    Attributes attrs,
    Direction dir,
    Dma_addr * dma_addr)
```

Map the given part of this data space into the DMA address space.

Parameters

| | | |
|---------|---------------|---|
| in | <i>src</i> | Source data space (that describes the memory). Caller needs write right to the data space. |
| in | <i>offset</i> | The offset (bytes) within <i>src</i> . |
| in, out | <i>size</i> | The size (bytes) of the region to be mapped for DMA, after successful mapping the size returned is the size mapped for DMA as a single block. This size might be smaller than the original input size, in this case the caller might call map() again with a new offset and the remaining size. |
| in | <i>attrs</i> | The attributes used for this DMA mapping (a combination of Dma_space::Attribute values). |

| | | |
|-----|-----------------|--|
| in | <i>dir</i> | The direction of the DMA transfer issued with this mapping. The same value must later be passed to unmap() . |
| out | <i>dma_addr</i> | The DMA address to use for DMA with the associated device. |

Return values

| | |
|-------------------|--|
| <i>L4_EOK</i> | Operation successful. |
| <i>-L4_EPERM</i> | Insufficient permissions; see precondition. |
| <i>-L4_EINVAL</i> | The capability <i>src</i> is invalid or does not refer to a valid dataspace. |
| <i>-L4_EEXIST</i> | The specified region overlaps an existing mapping. |
| <i>-L4_ENOMEM</i> | Not enough memory to allocate internal datastructures. |
| <i>-L4_ERANGE</i> | <i>offset</i> is larger than the size of the dataspace. |

Precondition

The capability *src* must have the permission [L4_CAP_FPAGE_W](#).

Note

[associate\(\)](#) must be called prior to mapping memory. Usually this is done implicitly when binding the managed [Dma_space](#) to a DMA domain (see [L4vbus::Vbus::assign_dma_domain\(\)](#)).

References [unmap\(\)](#).

Here is the call graph for this function:



15.289.4.4 unmap()

```

l4_ret_t L4Re::Dma_space::unmap (
    Dma_addr dma_addr,
    l4_size_t size,
    Attributes attrs,
    Direction dir)
  
```

Unmap the given part of this data space from the DMA address space.

Parameters

| | |
|-----------------|--|
| <i>dma_addr</i> | The DMA address (returned by Dma_space::map()). |
| <i>size</i> | The size (bytes) of the memory region to unmap. |

| | |
|--------------|--|
| <i>attrs</i> | The attributes for the unmap (currently none). |
| <i>dir</i> | The direction of the finished DMA operation. |

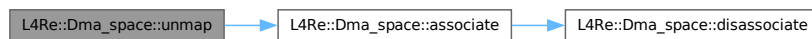
Returns

0 in the case of success, a negative error code otherwise.

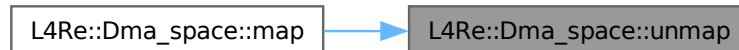
References [associate\(\)](#).

Referenced by [map\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

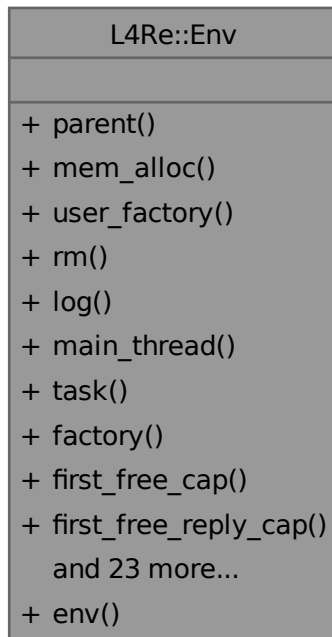
- [l4/re/dma_space](#)

15.290 L4Re::Env Class Reference

C++ interface of the initial environment that is provided to an [L4](#) task.

```
#include <env>
```

Collaboration diagram for L4Re::Env:



Public Types

- typedef [l4re_env_cap_entry_t](#) **Cap_entry**
C++ type for an entry in the initial objects array.

Public Member Functions

- [L4::Cap](#)< [Parent](#) > [parent](#) () const noexcept
Object-capability to the parent.
- [L4::Cap](#)< [Mem_alloc](#) > [mem_alloc](#) () const noexcept
Object-capability to the memory allocator.
- [L4::Cap](#)< [L4::Factory](#) > **user_factory** () const noexcept
Object-capability to the user-level object factory.
- [L4::Cap](#)< [Rm](#) > [rm](#) () const noexcept
Object-capability to the region map.
- [L4::Cap](#)< [Log](#) > [log](#) () const noexcept
Object-capability to the logging service.
- [L4::Cap](#)< [L4::Thread](#) > [main_thread](#) () const noexcept
Object-capability of the first user thread.
- [L4::Cap](#)< [L4::Task](#) > [task](#) () const noexcept
Object-capability of the user task.
- [L4::Cap](#)< [L4::Factory](#) > [factory](#) () const noexcept
Object-capability to the factory object available to the task.

- [l4_cap_idx_t first_free_cap](#) () const noexcept
First available capability selector.
- [l4_umword_t first_free_reply_cap](#) () const noexcept
First available reply capability index.
- [l4_fpage_t utcb_area](#) () const noexcept
UTCB area of the task.
- [l4_addr_t first_free_utcb](#) () const noexcept
First free UTCB.
- [Cap_entry](#) const * [initial_caps](#) () const noexcept
Get a pointer to the first entry in the initial objects array.
- [Cap_entry](#) const * [get](#) (char const *name, unsigned l) const noexcept
Get the [Cap_entry](#) for the object named name.
- template<typename T>
[L4::Cap](#)< T > [get_cap](#) (char const *name, unsigned l) const noexcept
Get the capability selector for the object named name.
- template<typename T>
[L4::Cap](#)< T > [get_cap](#) (char const *name) const noexcept
Get the capability selector for the object named name.
- void [parent](#) ([L4::Cap](#)< [Parent](#) > const &c) noexcept
Set parent object-capability.
- void [mem_alloc](#) ([L4::Cap](#)< [Mem_alloc](#) > const &c) noexcept
Set memory allocator object-capability.
- void [rm](#) ([L4::Cap](#)< [Rm](#) > const &c) noexcept
Set region map object-capability.
- void [log](#) ([L4::Cap](#)< [Log](#) > const &c) noexcept
Set log object-capability.
- void [main_thread](#) ([L4::Cap](#)< [L4::Thread](#) > const &c) noexcept
Set object-capability of first user thread.
- void [factory](#) ([L4::Cap](#)< [L4::Factory](#) > const &c) noexcept
Set factory object-capability.
- void [first_free_cap](#) ([l4_cap_idx_t](#) c) noexcept
Set first available capability selector.
- void [first_free_reply_cap](#) ([l4_umword_t](#) c) noexcept
Set first available reply capability index.
- void [utcb_area](#) ([l4_fpage_t](#) utcb) noexcept
Set UTCB area of the task.
- void [first_free_utcb](#) ([l4_addr_t](#) u) noexcept
Set first free UTCB.
- [L4::Cap](#)< [L4::Scheduler](#) > [scheduler](#) () const noexcept
Get the scheduler capability for the task.
- void [scheduler](#) ([L4::Cap](#)< [L4::Scheduler](#) > const &c) noexcept
Set the scheduler capability.
- [L4::Cap](#)< [Itas](#) > [itas](#) () const noexcept
Object-capability to the ITAS services.
- void [itas](#) ([L4::Cap](#)< [Itas](#) > const &c) noexcept
Set the ITAS capability.
- [L4::Cap](#)< [Dbg_events](#) > [dbg_events](#) () const noexcept
Object-capability to a debugger events service.
- void [dbg_events](#) ([L4::Cap](#)< [Dbg_events](#) > const &dbg_events) noexcept
Set the dbg_events capability.
- void [initial_caps](#) ([Cap_entry](#) *first) noexcept
Set the pointer to the first [Cap_entry](#) in the initial objects array.

Static Public Member Functions

- static [Env](#) const * [env](#) () noexcept
Returns the initial environment for the current task.

15.290.1 Detailed Description

C++ interface of the initial environment that is provided to an [L4](#) task.

The initial environment is provided to each [L4](#) task that is started by an [L4Re](#) conform loader, such as the Moe root task. The initial environment provides access to a set of initial capabilities and some additional information about the available resources, such as free UTCBs (see [Virtual Registers](#)) and available entries in capability table (provided by the micro kernel).

Each of the initial capabilities is stored at a fixed index in the task's capability table and the [L4](#) runtime environment provides convenience functions to retrieve the capabilities. See the table below for an comprehensive overview.

| Name | Object Type | Convenience Function |
|--------------|-------------------------------|---|
| parent | L4Re::Parent | L4Re::Env::parent() |
| user_factory | L4::Factory | L4Re::Env::user_factory() |
| log | L4Re::Log | L4Re::Env::log() |
| main_thread | L4::Thread | L4Re::Env::main_thread() |
| rm | L4Re::Rm | L4Re::Env::rm() |
| factory | L4::Factory | L4Re::Env::factory() |
| task | L4::Task | L4Re::Env::task() |
| scheduler | L4::Scheduler | L4Re::Env::scheduler() |
| itas | L4Re::Itas | L4Re::Env::itas() |

Additional information found in the initial environment is:

- First free entry in capability table
- The [UTCB](#) area (as flexpage)
- First free UTCB (address in the UTCB area)

Include File

```
#include <l4/re/env>
```

For an explanation of the default task capabilities see [l4_default_caps_t](#).

For the C interface refer to [Initial Environment](#).

Definition at line 78 of file [env](#).

15.290.2 Member Function Documentation

15.290.2.1 dbg_events() [1/2]

`L4::Cap< Dbg_events > L4Re::Env::dbg_events () const [inline], [noexcept]`

Object-capability to a debugger events service.

Returns

Dbg_events object-capability

This capability can be invalid.

Definition at line 328 of file [env](#).

Referenced by [dbg_events\(\)](#).

Here is the caller graph for this function:



15.290.2.2 dbg_events() [2/2]

```

void L4Re::Env::dbg_events (
    L4::Cap< Dbg_events > const & dbg_events) [inline], [noexcept]
  
```

Set the dbg_events capability.

Parameters

| | |
|-------------------------|---|
| <code>dbg_events</code> | is the capability to be set for the debug events service. |
|-------------------------|---|

Note that the capability can be invalid.

Definition at line 338 of file [env](#).

References [dbg_events\(\)](#).

Here is the call graph for this function:



15.290.2.3 env()

```
Env const * L4Re::Env::env () [inline], [static], [noexcept]
```

Returns the initial environment for the current task.

Returns

Pointer to the initial environment class.

A typical use of this function is `L4Re::Env::env()-><member>()`

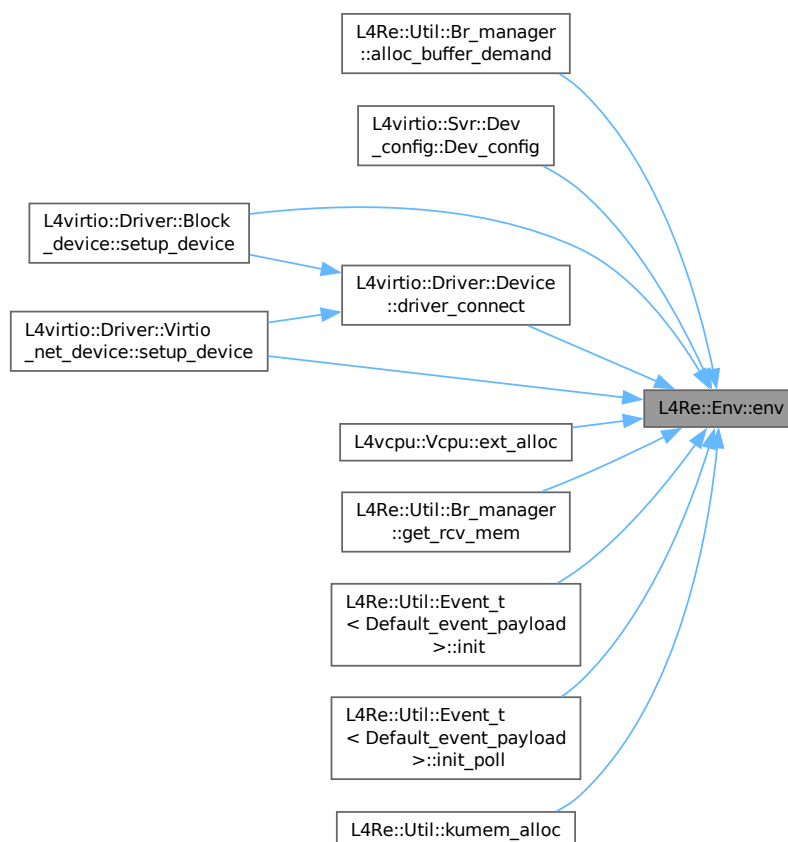
Examples

`examples/clntsrv/src/client.cc`, `examples/libs/l4re/c++/mem_alloc/ma+rm.cc`, `examples/libs/l4re/c++/shared_ds/ds_clnt.cc`, `examples/libs/l4re/c++/shared_ds/ds_srv.cc`, `examples/libs/l4re/streammap/client.cc`, and `examples/sys/migrate/thread_migrate.cc`

Definition at line 96 of file `env`.

Referenced by `L4Re::Util::Br_manager::alloc_buffer_demand()`, `L4virtio::Svr::Dev_config::Dev_config()`, `L4virtio::Driver::Device::driver_connect()`, `L4vcpu::Vcpu::ext_alloc()`, `L4Re::Util::Br_manager::get_rcv_mem()`, `L4Re::Util::Event_t< Default_event_payload >::init()`, `L4Re::Util::Event_t< Default_event_payload >::init_poll()`, `L4Re::Util::kumem_alloc()`, `L4virtio::Driver::Block_device::setup_device()`, and `L4virtio::Driver::Virtio_net_device::setup_device()`.

Here is the caller graph for this function:



15.290.2.4 factory() [1/2]

```
L4::Cap< L4::Factory > L4Re::Env::factory () const [inline], [noexcept]
```

Object-capability to the factory object available to the task.

Returns

Factory object-capability

Definition at line 148 of file [env](#).

15.290.2.5 factory() [2/2]

```
void L4Re::Env::factory (  
    L4::Cap< L4::Factory > const & c) [inline], [noexcept]
```

Set factory object-capability.

Parameters

| | |
|----------|---------------------------|
| <i>c</i> | Factory object-capability |
|----------|---------------------------|

Definition at line 261 of file [env](#).

15.290.2.6 first_free_cap() [1/2]

```
l4_cap_idx_t L4Re::Env::first_free_cap () const [inline], [noexcept]
```

First available capability selector.

Returns

First capability selector.

First capability selector available for use for in the application.

Definition at line 156 of file [env](#).

15.290.2.7 first_free_cap() [2/2]

```
void L4Re::Env::first_free_cap (
    l4_cap_idx_t c) [inline], [noexcept]
```

Set first available capability selector.

Parameters

| | |
|----------|---|
| c | First capability selector available to the application. |
|----------|---|

Definition at line 267 of file [env](#).

15.290.2.8 first_free_reply_cap() [1/2]

```
l4_umword_t L4Re::Env::first_free_reply_cap () const [inline], [noexcept]
```

First available reply capability index.

Returns

First reply capability index.

First reply capability selector index for use for in the application.

Definition at line 164 of file [env](#).

15.290.2.9 first_free_reply_cap() [2/2]

```
void L4Re::Env::first_free_reply_cap (
    l4_umword_t c) [inline], [noexcept]
```

Set first available reply capability index.

Parameters

| | |
|----------|--|
| c | First reply capability index available to the application. |
|----------|--|

Definition at line 273 of file [env](#).

15.290.2.10 first_free_utcb() [1/2]

```
l4_addr_t L4Re::Env::first_free_utcb () const [inline], [noexcept]
```

First free UTCB.

Returns

object-capability

First free UTCB within the UTCB area available for the application to use.

Definition at line 179 of file [env](#).

15.290.2.11 first_free_utcb() [2/2]

```
void L4Re::Env::first_free_utcb (
    l4_addr_t u) [inline], [noexcept]
```

Set first free UTCB.

Parameters

| | |
|----------|--|
| <i>u</i> | First UTCB available for the application to use. |
|----------|--|

Definition at line 285 of file [env](#).

15.290.2.12 get()

```
Cap_entry const * L4Re::Env::get (
    char const * name,
    unsigned l) const [inline], [noexcept]
```

Get the [Cap_entry](#) for the object named *name*.

Parameters

| | |
|-------------|---|
| <i>name</i> | is the name of the object. |
| <i>l</i> | is the length of the name, thus <i>name</i> might not be zero terminated. |

Returns

A pointer to the [Cap_entry](#) for the object named *name*, or NULL if no such object was found.

Definition at line 197 of file [env](#).

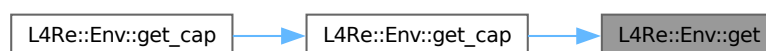
References [l4re_env_get_cap_l\(\)](#).

Referenced by [get_cap\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.290.2.13 `get_cap()` [1/2]

```
template<typename T>
L4::Cap< T > L4Re::Env::get_cap (
    char const * name) const [inline], [noexcept]
```

Get the capability selector for the object named *name*.

Parameters

| | |
|-------------|--|
| <i>name</i> | is the name of the object (zero terminated). |
|-------------|--|

Returns

A capability selector for the object named *name*, or an invalid capability selector if no such object was found.

Definition at line 224 of file [env](#).

References [get_cap\(\)](#).

Here is the call graph for this function:

**15.290.2.14** `get_cap()` [2/2]

```
template<typename T>
L4::Cap< T > L4Re::Env::get_cap (
    char const * name,
    unsigned l) const [inline], [noexcept]
```

Get the capability selector for the object named *name*.

Parameters

| | |
|-------------|---|
| <i>name</i> | is the name of the object. |
| <i>l</i> | is the length of the name, thus <i>name</i> might not be zero terminated. |

Returns

A capability selector for the object named *name*, or an invalid capability selector if no such object was found.

Examples

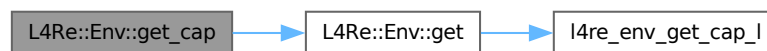
[examples/clntsrv/src/client.cc](#), [examples/libs/l4re/c++/shared_ds/ds_clnt.cc](#), and [examples/libs/l4re/streammap/client.cc](#).

Definition at line 209 of file [env](#).

References [get\(\)](#), and [L4_ENOENT](#).

Referenced by [get_cap\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**15.290.2.15 initial_caps() [1/2]**

```
Cap_entry const * L4Re::Env::initial_caps () const [inline], [noexcept]
```

Get a pointer to the first entry in the initial objects array.

Returns

A pointer to the first entry in the initial objects array.

Definition at line 186 of file [env](#).

15.290.2.16 initial_caps() [2/2]

```
void L4Re::Env::initial_caps (
    Cap_entry * first) [inline], [noexcept]
```

Set the pointer to the first [Cap_entry](#) in the initial objects array.

Parameters

| | |
|--------------|------------------------------------|
| <i>first</i> | is the first element in the array. |
|--------------|------------------------------------|

Definition at line [345](#) of file [env](#).

15.290.2.17 itas() [1/2]

```
L4::Cap< Itas > L4Re::Env::itas () const [inline], [noexcept]
```

Object-capability to the ITAS services.

Returns

ITAS object-capability

Attention: this capability might be invalid, depending on the system configuration. Regular applications must not use it directly as it is an implementation detail of the [L4Re](#) libc that is subject to change without notice!

Definition at line [312](#) of file [env](#).

15.290.2.18 itas() [2/2]

```
void L4Re::Env::itas (
    L4::Cap< Itas > const & c) [inline], [noexcept]
```

Set the ITAS capability.

Parameters

| | |
|----------|--------------------------------------|
| <i>c</i> | is the capability to be set as ITAS. |
|----------|--------------------------------------|

Definition at line [319](#) of file [env](#).

15.290.2.19 log() [1/2]

```
L4::Cap< Log > L4Re::Env::log () const [inline], [noexcept]
```

Object-capability to the logging service.

Might be invalid if there is no logging service available to the application!

Returns

[Log](#) object-capability

Definition at line [130](#) of file [env](#).

15.290.2.20 log() [2/2]

```
void L4Re::Env::log (
    L4::Cap< Log > const & c) [inline], [noexcept]
```

Set log object-capability.

Parameters

| | |
|----------------|-----------------------|
| <code>c</code> | Log object-capability |
|----------------|-----------------------|

Definition at line 249 of file [env](#).

15.290.2.21 main_thread() [1/2]

```
L4::Cap< L4::Thread > L4Re::Env::main_thread () const [inline], [noexcept]
```

Object-capability of the first user thread.

Returns

Object-capability of the first user thread.

Definition at line 136 of file [env](#).

15.290.2.22 main_thread() [2/2]

```
void L4Re::Env::main_thread (
    L4::Cap< L4::Thread > const & c) [inline], [noexcept]
```

Set object-capability of first user thread.

Parameters

| | |
|----------------|----------------------------------|
| <code>c</code> | First thread's object-capability |
|----------------|----------------------------------|

Definition at line 255 of file [env](#).

15.290.2.23 mem_alloc() [1/2]

```
L4::Cap< Mem_alloc > L4Re::Env::mem_alloc () const [inline], [noexcept]
```

Object-capability to the memory allocator.

Returns

Memory allocator object-capability

Examples

[examples/libs/l4re/c++/shared_ds/ds_srv.cc](#).

Definition at line 109 of file [env](#).

15.290.2.24 mem_alloc() [2/2]

```
void L4Re::Env::mem_alloc (  
    L4::Cap< Mem_alloc > const & c) [inline], [noexcept]
```

Set memory allocator object-capability.

Parameters

| | |
|----------|------------------------------------|
| <i>c</i> | Memory allocator object-capability |
|----------|------------------------------------|

Definition at line 237 of file [env](#).

15.290.2.25 parent() [1/2]

```
L4::Cap< Parent > L4Re::Env::parent () const [inline], [noexcept]
```

Object-capability to the parent.

Returns

[Parent](#) object-capability

Definition at line 103 of file [env](#).

15.290.2.26 parent() [2/2]

```
void L4Re::Env::parent (  
    L4::Cap< Parent > const & c) [inline], [noexcept]
```

Set parent object-capability.

Parameters

| | |
|----------|--|
| <i>c</i> | Parent object-capability |
|----------|--|

Definition at line 231 of file [env](#).

15.290.2.27 `rm()` [1/2]

```
L4::Cap< Rm > L4Re::Env::rm () const [inline], [noexcept]
```

Object-capability to the region map.

Returns

Region map object-capability

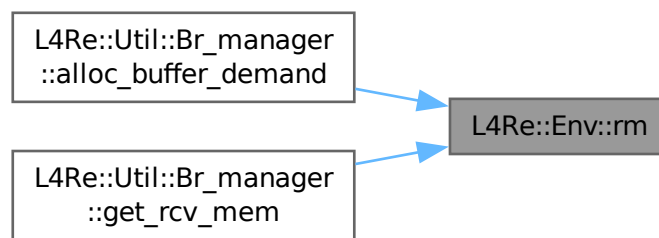
Examples

[examples/libs/l4re/c++/shared_ds/ds_clnt.cc](#), and [examples/libs/l4re/c++/shared_ds/ds_srv.cc](#).

Definition at line 120 of file [env](#).

Referenced by [L4Re::Util::Br_manager::alloc_buffer_demand\(\)](#), and [L4Re::Util::Br_manager::get_rcv_mem\(\)](#).

Here is the caller graph for this function:

**15.290.2.28** `rm()` [2/2]

```
void L4Re::Env::rm (
    L4::Cap< Rm > const & c) [inline], [noexcept]
```

Set region map object-capability.

Parameters

| | |
|----------------|------------------------------|
| <code>c</code> | Region map object-capability |
|----------------|------------------------------|

Definition at line 243 of file [env](#).

15.290.2.29 scheduler() [1/2]

```
L4::Cap< L4::Scheduler > L4Re::Env::scheduler () const [inline], [noexcept]
```

Get the scheduler capability for the task.

Returns

The capability selector for the default scheduler used for this task.

Definition at line 293 of file [env](#).

15.290.2.30 scheduler() [2/2]

```
void L4Re::Env::scheduler (
    L4::Cap< L4::Scheduler > const & c) [inline], [noexcept]
```

Set the scheduler capability.

Parameters

| | |
|----------|---|
| <i>c</i> | is the capability to be set as scheduler. |
|----------|---|

Definition at line 300 of file [env](#).

15.290.2.31 task()

```
L4::Cap< L4::Task > L4Re::Env::task () const [inline], [noexcept]
```

Object-capability of the user task.

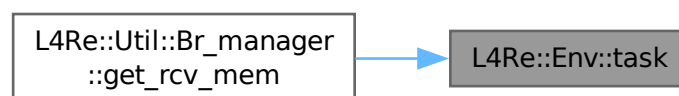
Returns

Object-capability of the user task.

Definition at line 142 of file [env](#).

Referenced by [L4Re::Util::Br_manager::get_rcv_mem\(\)](#).

Here is the caller graph for this function:



15.290.2.32 utcb_area() [1/2]

```
l4_fpage_t L4Re::Env::utcb_area () const [inline], [noexcept]
```

UTCB area of the task.

Returns

UTCB area

Definition at line 170 of file [env](#).

15.290.2.33 utcb_area() [2/2]

```
void L4Re::Env::utcb_area (
    l4_fpage_t utcbs) [inline], [noexcept]
```

Set UTCB area of the task.

Parameters

| | |
|--------------|-----------|
| <i>utcbs</i> | UTCB area |
|--------------|-----------|

Definition at line 279 of file [env](#).

The documentation for this class was generated from the following file:

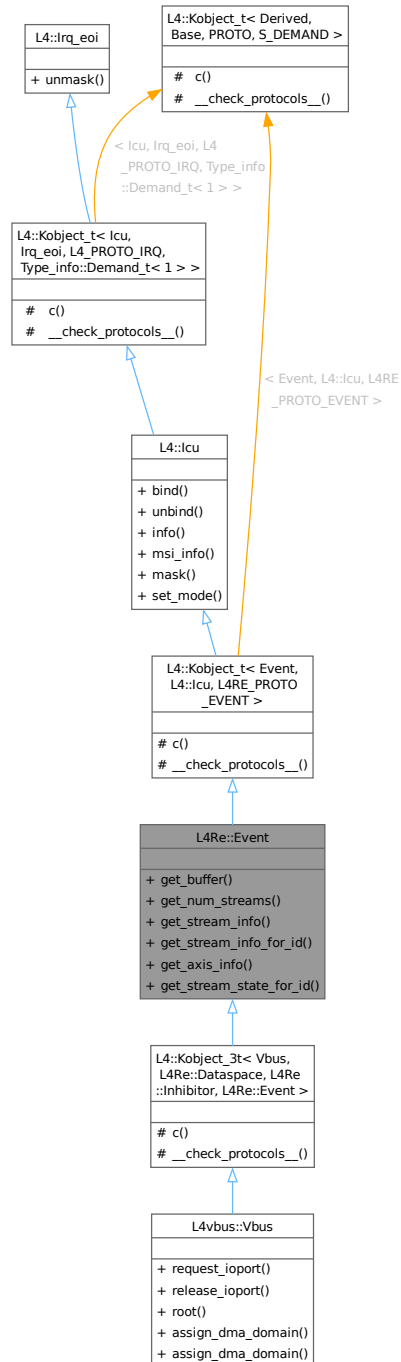
- [l4/re/env](#)

15.291 L4Re::Event Class Reference

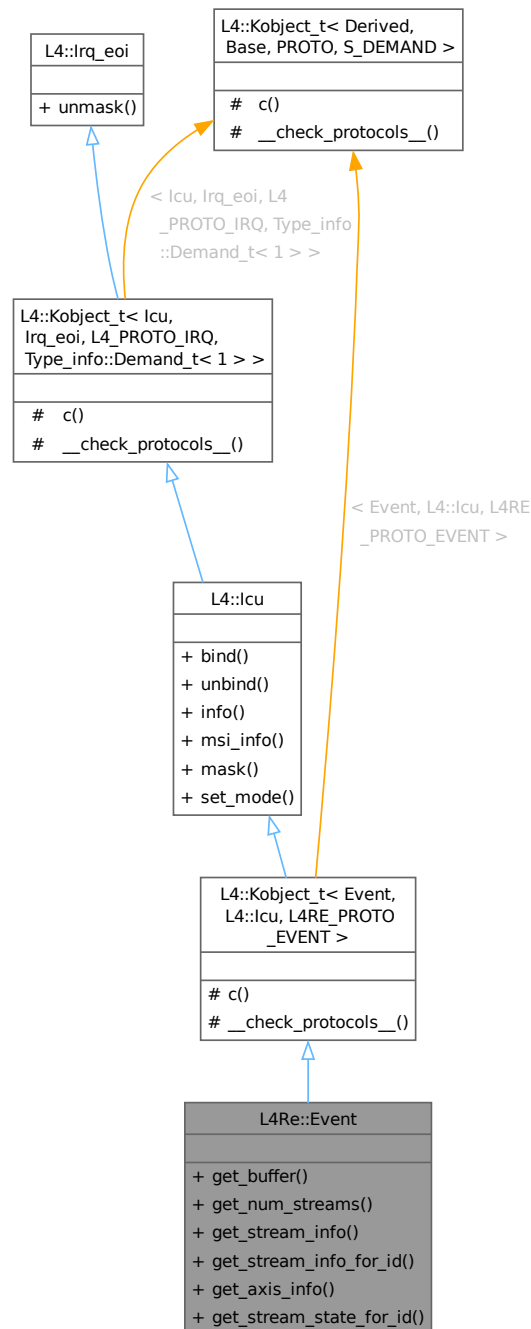
[Event](#) class.

```
#include <event>
```

Inheritance diagram for L4Re::Event:



Collaboration diagram for L4Re::Event:



Public Member Functions

- `l4_ret_t get_buffer (L4::lpc::Out< L4::Cap< Dataspace > > ds)`
Get event signal buffer.
- `l4_ret_t get_num_streams ()`
Get number of event streams.
- `l4_ret_t get_stream_info (int idx, Event_stream_info *info)`

Get event stream infos.

- [l4_ret_t get_stream_info_for_id](#) ([l4_umword_t](#) stream_id, [Event_stream_info](#) *info)

Get event stream infos.

- [l4_ret_t get_axis_info](#) ([l4_umword_t](#) stream_id, unsigned naxes, unsigned const *axis, [Event_absinfo](#) *info) const noexcept

Get event stream axis infos.

- [l4_ret_t get_stream_state_for_id](#) ([l4_umword_t](#) stream_id, [Event_stream_state](#) *state)

Get event stream state.

Public Member Functions inherited from [L4::lcu](#)

- [l4_msgtag_t bind](#) (unsigned irqnum, [L4::Cap](#)< [Triggerable](#) > irq, [l4_utcb_t](#) *utcb=[l4_utcb](#)()) noexcept

Bind an interrupt line of an interrupt controller to an interrupt object.

- [l4_msgtag_t unbind](#) (unsigned irqnum, [L4::Cap](#)< [Triggerable](#) > irq, [l4_utcb_t](#) *utcb=[l4_utcb](#)()) noexcept

Remove binding of an interrupt line from the interrupt controller object.

- [l4_msgtag_t info](#) ([l4_icu_info_t](#) *info, [l4_utcb_t](#) *utcb=[l4_utcb](#)()) noexcept

Get information about the ICU features.

- [l4_msgtag_t msi_info](#) ([l4_umword_t](#) irqnum, [l4_uint64_t](#) source, [l4_icu_msi_info_t](#) *msi_info)

Get MSI info about IRQ.

- [l4_msgtag_t mask](#) (unsigned irqnum, [l4_umword_t](#) *label=0, [l4_timeout_t](#) to=[L4_IPC_NEVER](#), [l4_utcb_t](#) *utcb=[l4_utcb](#)()) noexcept

Mask an IRQ line.

- [l4_msgtag_t set_mode](#) (unsigned irqnum, [l4_umword_t](#) mode, [l4_utcb_t](#) *utcb=[l4_utcb](#)()) noexcept

Set interrupt mode.

Public Member Functions inherited from [L4::lrq_eoi](#)

- [l4_msgtag_t unmask](#) (unsigned irqnum, [l4_umword_t](#) *label=0, [l4_timeout_t](#) to=[L4_IPC_NEVER](#), [l4_utcb_t](#) *utcb=[l4_utcb](#)()) noexcept

Unmask the given interrupt line.

Additional Inherited Members

Protected Types inherited from [L4::Kobject_t](#)< [Event](#), [L4::lcu](#), [L4RE_PROTO_EVENT](#) >

- typedef [Event](#) **Class**

The target interface type (inheriting from [Kobject_t](#)).

- typedef [Typeid::Iface](#)< [PROTO](#), [Event](#) > **__Iface**

The interface description for the derived class.

- typedef [Typeid::Merge_list](#)< [Typeid::Iface_list](#)< **__Iface** >, typename [L4::lcu::__Iface_list](#) > **__Iface_list**

The list of all RPC interfaces provided directly or through inheritance.

Protected Types inherited from

[L4::Kobject_t](#)< [lcu](#), [lrq_eoi](#), [L4_PROTO_IRQ](#), [Type_info::Demand_t](#)< 1 > >

- typedef [lcu](#) **Class**

The target interface type (inheriting from [Kobject_t](#)).

- typedef [Typeid::Iface](#)< [PROTO](#), [lcu](#) > **__Iface**

The interface description for the derived class.

- typedef [Typeid::Merge_list](#)< [Typeid::Iface_list](#)< **__Iface** >, typename [lrq_eoi::__Iface_list](#) > **__Iface_list**

The list of all RPC interfaces provided directly or through inheritance.

Protected Member Functions inherited from [L4::Kobject_t< Event, L4::lcu, L4RE_PROTO_EVENT >](#)

- [L4::Cap< Class > c\(\)](#) const noexcept
Get the capability to ourselves.

Protected Member Functions inherited from [L4::Kobject_t< lcu, Irq_eoi, L4_PROTO_IRQ, Type_info::Demand_t< 1 > >](#)

- [L4::Cap< Class > c\(\)](#) const noexcept
Get the capability to ourselves.

Static Protected Member Functions inherited from [L4::Kobject_t< Event, L4::lcu, L4RE_PROTO_EVENT >](#)

- static void [__check_protocols__\(\)](#) noexcept
Helper to check for protocol conflicts.

Static Protected Member Functions inherited from [L4::Kobject_t< lcu, Irq_eoi, L4_PROTO_IRQ, Type_info::Demand_t< 1 > >](#)

- static void [__check_protocols__\(\)](#) noexcept
Helper to check for protocol conflicts.

15.291.1 Detailed Description

[Event](#) class.

See also

[L4Re Event API](#)

Definition at line 137 of file [event](#).

15.291.2 Member Function Documentation

15.291.2.1 [get_axis_info\(\)](#)

```
l4_ret_t L4Re::Event::get_axis_info (
    l4_umword_t stream_id,
    unsigned naxes,
    unsigned const * axis,
    Event_absinfo * info) const [inline], [noexcept]
```

Get event stream axis infos.

Parameters

| | | |
|-----|-------------------------------|------------------------------------|
| | <i>stream</i> ↔ <i>_id</i> | ID of the event stream. |
| in | <i>naxes</i> | Number of axis IDs and axis infos. |
| in | <i>axis</i> | Array of axis IDs. |
| out | <i>info</i> | Array of axis infos. |

Return values

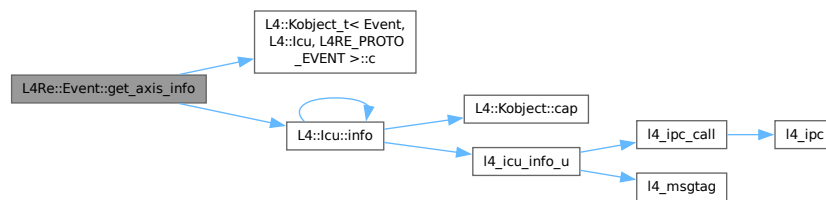
| | |
|----------|--------------------------------|
| ≥ 0 | Number of returned axes infos. |
| < 0 | Error code. |

Definition at line 200 of file [event](#).

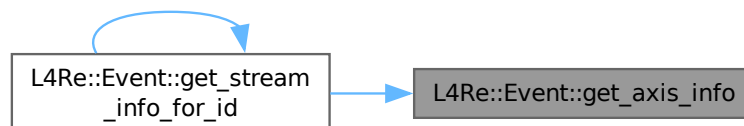
References [L4::Kobject_t< Event, L4::lcu, L4RE_PROTO_EVENT >::c\(\)](#), and [L4::lcu::info\(\)](#).

Referenced by [get_stream_info_for_id\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.291.2.2 `get_buffer()`

```
l4_ret_t L4Re::Event::get_buffer (
    L4::Ipc::Out< L4::Cap< Dataspace > > ds)
```

Get event signal buffer.

Parameters

| | | |
|-----|----|---------------|
| out | ds | Event buffer. |
|-----|----|---------------|

Return values

| | |
|----|---------|
| 0 | Success |
| <0 | Error |

References [get_buffer\(\)](#).

Referenced by [get_buffer\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.291.2.3 `get_num_streams()`

```
l4_ret_t L4Re::Event::get_num_streams ()
```

Get number of event streams.

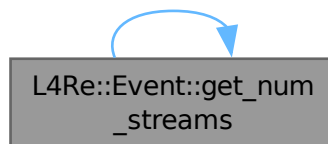
Return values

| | |
|----------|--------------------|
| ≥ 0 | Number of streams. |
| < 0 | Error code. |

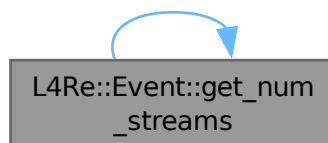
References [get_num_streams\(\)](#).

Referenced by [get_num_streams\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.291.2.4 `get_stream_info()`

```
l4_ret_t L4Re::Event::get_stream_info (  
    int idx,  
    Event_stream_info * info)
```

Get event stream infos.

Deprecated. Use [get_stream_info_for_id\(\)](#).

Parameters

| | | |
|-----|-------------|------------------------------------|
| | <i>idx</i> | ID of the event stream. |
| out | <i>info</i> | Event stream info. |

Return values

| | |
|----|---------|
| 0 | Success |
| <0 | Error |

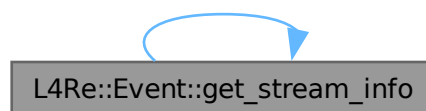
References [get_stream_info\(\)](#), and [L4::lcu::info\(\)](#).

Referenced by [get_stream_info\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.291.2.5 get_stream_info_for_id()

```

l4_ret_t L4Re::Event::get_stream_info_for_id (
    l4_umword_t stream_id,
    Event_stream_info * info)

```

Get event stream infos.

Parameters

| | | |
|-----|-------------------------------|------------------------------------|
| | <i>stream</i> ↔ <i>_id</i> | ID of the event stream. |
| out | <i>info</i> | Event stream info. |

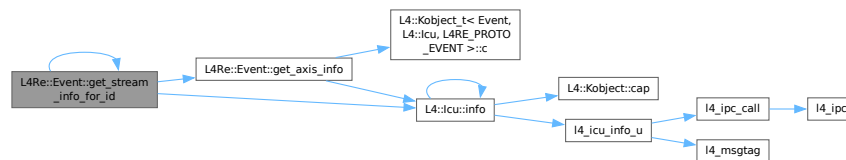
Return values

| | |
|----|---------|
| 0 | Success |
| <0 | Error |

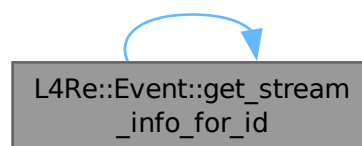
References [get_axis_info\(\)](#), [get_stream_info_for_id\(\)](#), and [L4::lcu::info\(\)](#).

Referenced by [get_stream_info_for_id\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**15.291.2.6 get_stream_state_for_id()**

```

l4_ret_t L4Re::Event::get_stream_state_for_id (
    l4_umword_t stream_id,
    Event_stream_state * state)

```

Get event stream state.

Parameters

| | | |
|-----|-------------------------------|-------------------------------------|
| | <i>stream</i> ↔ <i>_id</i> | ID of the event stream. |
| out | <i>state</i> | Event stream state. |

Return values

| | |
|----|---------|
| 0 | Success |
| <0 | Error |

The documentation for this class was generated from the following file:

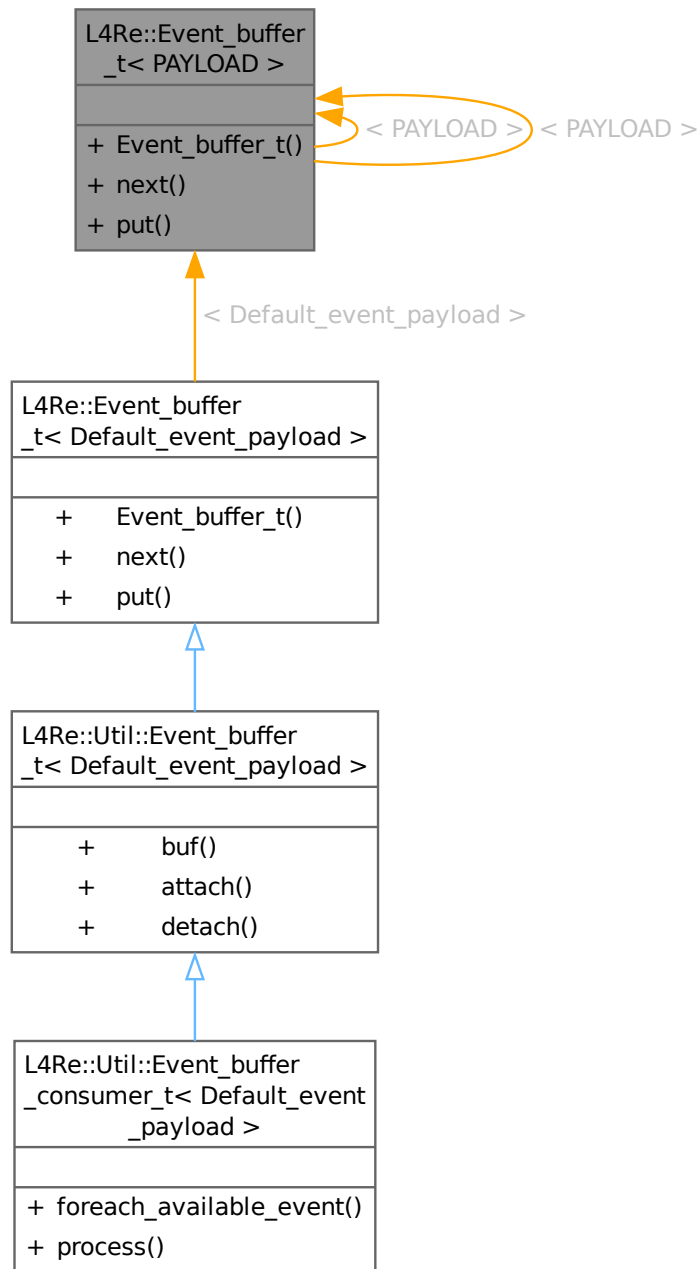
- l4/re/event

15.292 L4Re::Event_buffer_t< PAYLOAD > Class Template Reference

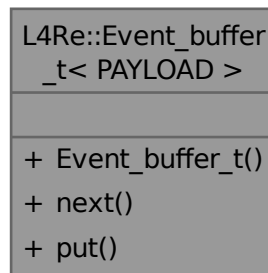
[Event](#) buffer class.

```
#include <event>
```

Inheritance diagram for L4Re::Event_buffer_t< PAYLOAD >:



Collaboration diagram for L4Re::Event_buffer_t< PAYLOAD >:



Data Structures

- struct [Event](#)
[Event](#) structure used in buffer.

Public Member Functions

- [Event_buffer_t](#) (void *buffer, [l4_addr_t](#) size)
Initialize event buffer.
- Event * [next](#) () noexcept
Next event in buffer.
- bool [put](#) (Event const &ev) noexcept
Put event into buffer at current position.

15.292.1 Detailed Description

```
template<typename PAYLOAD = Default_event_payload>
class L4Re::Event_buffer_t< PAYLOAD >
```

[Event](#) buffer class.

Definition at line [248](#) of file [event](#).

15.292.2 Constructor & Destructor Documentation

15.292.2.1 Event_buffer_t()

```
template<typename PAYLOAD = Default_event_payload>
L4Re::Event_buffer_t< PAYLOAD >::Event_buffer_t (
    void * buffer,
    l4_addr_t size) [inline]
```

Initialize event buffer.

Parameters

| | |
|---------------|--------------------------|
| <i>buffer</i> | Pointer to buffer. |
| <i>size</i> | Size of buffer in bytes. |

Definition at line 295 of file [event](#).

15.292.3 Member Function Documentation

15.292.3.1 next()

```
template<typename PAYLOAD = Default_event_payload>
Event * L4Re::Event_buffer_t< PAYLOAD >::next () [inline], [noexcept]
```

Next event in buffer.

Returns

0 if no event available, event otherwise.

Definition at line 305 of file [event](#).

15.292.3.2 put()

```
template<typename PAYLOAD = Default_event_payload>
bool L4Re::Event_buffer_t< PAYLOAD >::put (
    Event const & ev) [inline], [noexcept]
```

Put event into buffer at current position.

Parameters

| | |
|-----------|---|
| <i>ev</i> | Event to put into the buffer. |
|-----------|---|

Returns

false if buffer is full and entry could not be added.

Definition at line 322 of file [event](#).

The documentation for this class was generated from the following file:

- [l4/re/event](#)

15.293 L4Re::Event_buffer_t< PAYLOAD >::Event Struct Reference

[Event](#) structure used in buffer.

```
#include <event>
```

Collaboration diagram for L4Re::Event_buffer_t< PAYLOAD >::Event:

| L4Re::Event_buffer _t< PAYLOAD >::Event | |
|--|--------|
| + | time |
| + | free() |

Public Member Functions

- void **free** () noexcept
Free the entry.

Data Fields

- long long **time**
[Event](#) time stamp.

15.293.1 Detailed Description

```
template<typename PAYLOAD = Default_event_payload>  
struct L4Re::Event_buffer_t< PAYLOAD >::Event
```

[Event](#) structure used in buffer.

Definition at line 255 of file [event](#).

The documentation for this struct was generated from the following file:

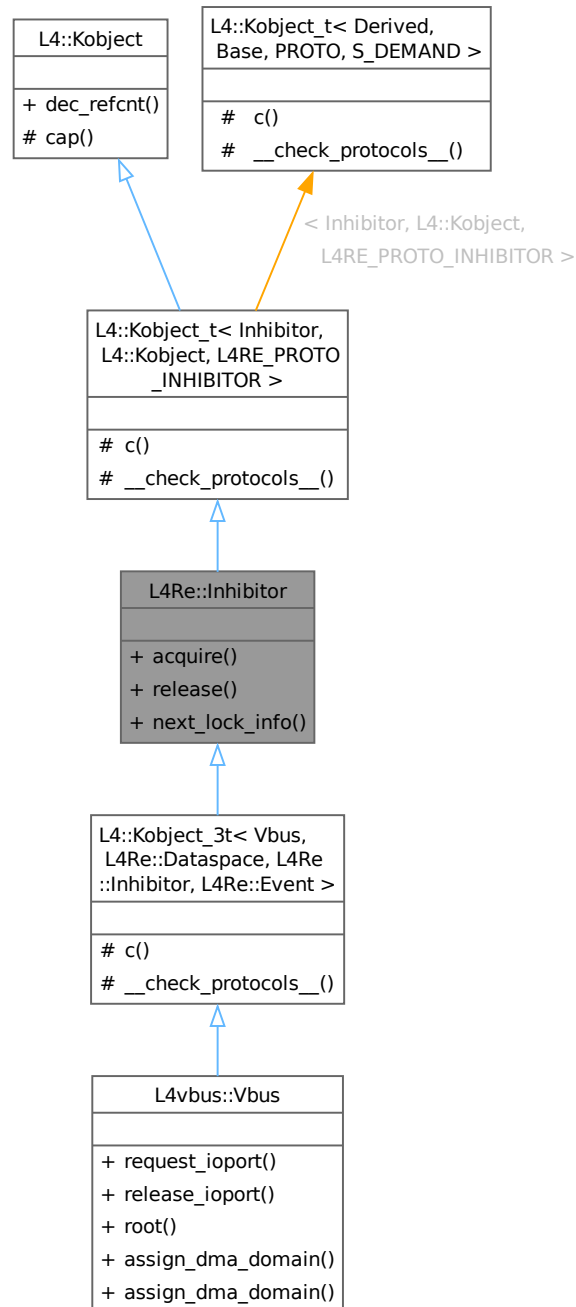
- l4/re/event

15.294 L4Re::Inhibitor Class Reference

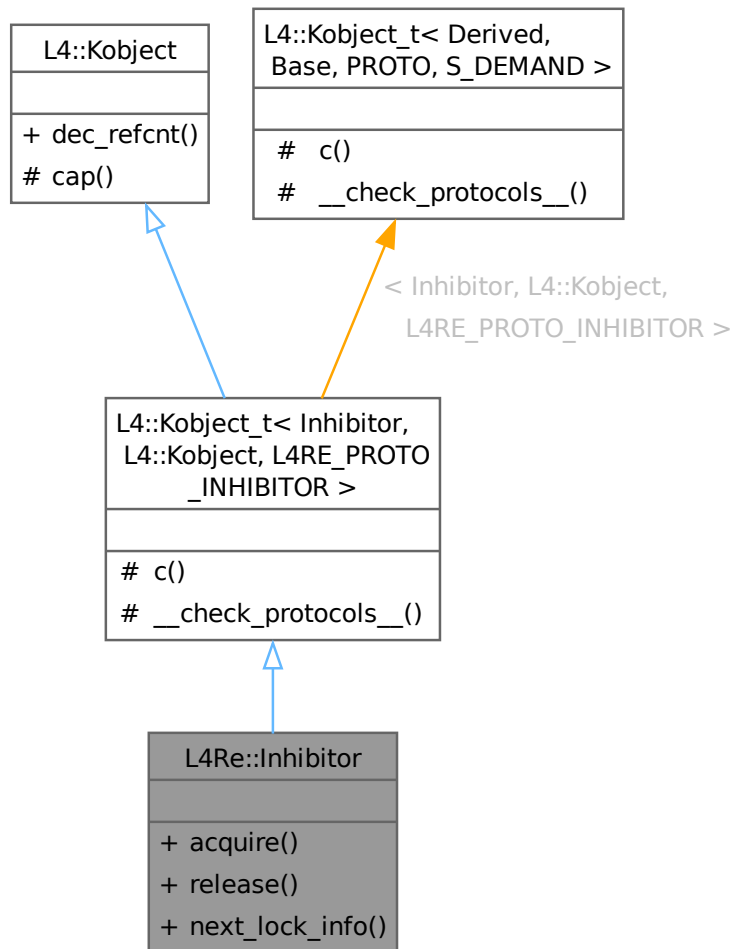
Set of inhibitor locks, which inhibit specific actions when held.

```
#include <inhibitor>
```

Inheritance diagram for L4Re::Inhibitor:



Collaboration diagram for L4Re::Inhibitor:



Public Types

- enum { `Name_max` = 20 }

Public Member Functions

- `l4_ret_t acquire (l4_umword_t id, L4::lpc::String<> reason)`
Acquire a specific inhibitor lock.
- `l4_ret_t release (l4_umword_t id)`
Release a specific inhibitor lock.
- `l4_ret_t next_lock_info (char *name, unsigned len, l4_mword_t current_id=-1, l4_utcb_t *utcb=l4_utcb())`
Get information for the next available inhibitor lock.

Public Member Functions inherited from [L4::Kobject](#)

- [l4_msgtag_t dec_refcnt](#) ([l4_mword_t diff](#), [l4_utcb_t *utcb=l4_utcb\(\)](#))

Decrement the in kernel reference counter for the object.

Additional Inherited Members

Protected Types inherited from

[L4::Kobject_t< Inhibitor, L4::Kobject, L4RE_PROTO_INHIBITOR >](#)

- typedef Inhibitor **Class**

The target interface type (inheriting from [Kobject_t](#)).

- typedef Typeid::Iface< PROTO, Inhibitor > **__iface**

The interface description for the derived class.

- typedef Typeid::Merge_list< Typeid::Iface_list< [__iface](#) >, typename L4::Kobject::__iface_list > **__iface_list**

The list of all RPC interfaces provided directly or through inheritance.

Protected Member Functions inherited from

[L4::Kobject_t< Inhibitor, L4::Kobject, L4RE_PROTO_INHIBITOR >](#)

- [L4::Cap< Class > c](#) () const noexcept

Get the capability to ourselves.

Protected Member Functions inherited from [L4::Kobject](#)

- [l4_cap_idx_t cap](#) () const noexcept

Return capability selector.

Static Protected Member Functions inherited from

[L4::Kobject_t< Inhibitor, L4::Kobject, L4RE_PROTO_INHIBITOR >](#)

- static void [__check_protocols__](#) () noexcept

Helper to check for protocol conflicts.

15.294.1 Detailed Description

Set of inhibitor locks, which inhibit specific actions when held.

This interface provides access to a set of inhibitor locks, each determined by an ID that is specific to the [Inhibitor](#) object. Each individual lock shall prevent, a specific (implementation defined) action to be executed, as long as the lock is held.

For example there can be an inhibitor lock to prevent a transition to suspend-to-RAM state and a different one to prevent shutdown.

A client shall take an inhibitor lock if it needs to execute code before the action is taken. For example a lock-screen application shall grab an inhibitor lock for the suspend action to be able to lock the screen before the system goes to sleep.

[Inhibitor](#) locks are usually closely related to specific events. Usually a server automatically subscribes a client holding a lock to the corresponding event. The server shall send the event to inform the client that an action is pending. Upon reception of the event, the client is supposed to release the corresponding inhibitor lock.

Definition at line 38 of file [inhibitor](#).

15.294.2 Member Enumeration Documentation

15.294.2.1 anonymous enum

anonymous enum

Enumerator

| | |
|----------|--------------------------------------|
| Name_max | The maximum length of a lock's name. |
|----------|--------------------------------------|

Definition at line 42 of file [inhibitor](#).

15.294.3 Member Function Documentation

15.294.3.1 acquire()

```
l4_ret_t L4Re::Inhibitor::acquire (
    l4_umword_t id,
    L4::Ipc::String<> reason)
```

Acquire a specific inhibitor lock.

Parameters

| | |
|---------------|---|
| <i>id</i> | ID of the inhibitor lock that the client intends to acquire |
| <i>reason</i> | The reason why you need the lock. Used for informing the user or debugging. |

Return values

| | |
|------------|---|
| 0 | Success |
| -L4_ENODEV | The specified <i>id</i> does not exist. |

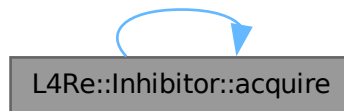
References [acquire\(\)](#).

Referenced by [acquire\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.294.3.2 next_lock_info()

```

l4_ret_t L4Re::Inhibitor::next_lock_info (
    char * name,
    unsigned len,
    l4_mword_t current_id = -1,
    l4_utcb_t * utcb = l4_utcb()) [inline]
  
```

Get information for the next available inhibitor lock.

Parameters

| | |
|-------------------|--|
| <i>name</i> | A pointer to a buffer for the name of the lock. |
| <i>len</i> | The length of the available buffer (usually Name_max is used). |
| <i>current_id</i> | The ID of the last available lock, use -1 to get the first lock. |
| <i>utcb</i> | The UTCB to use for the message. |

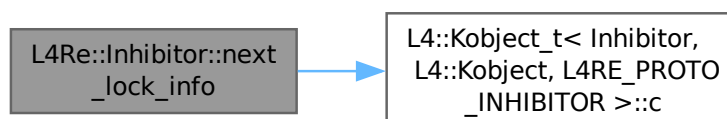
Return values

| | |
|-------------------|--|
| <i>>0</i> | The ID of the next available lock if there is one (in this case <i>name</i> shall contain the name of the inhibitor lock). |
| <i>-L4_ENODEV</i> | There are no more locks. |

Definition at line 84 of file [inhibitor](#).

References [L4::Kobject_t< Inhibitor, L4::Kobject, L4RE_PROTO_INHIBITOR >::c\(\)](#).

Here is the call graph for this function:



15.294.3.3 release()

```
l4_ret_t L4Re::Inhibitor::release (
    l4_umword_t id)
```

Release a specific inhibitor lock.

Parameters

| | |
|-----------|--|
| <i>id</i> | The ID of the inhibitor lock to release. |
|-----------|--|

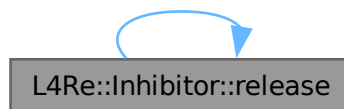
Return values

| | |
|------------|---|
| 0 | Success |
| -L4_ENODEV | Lock with the given <i>id</i> does not exist. |

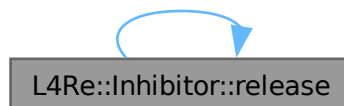
References [release\(\)](#).

Referenced by [release\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

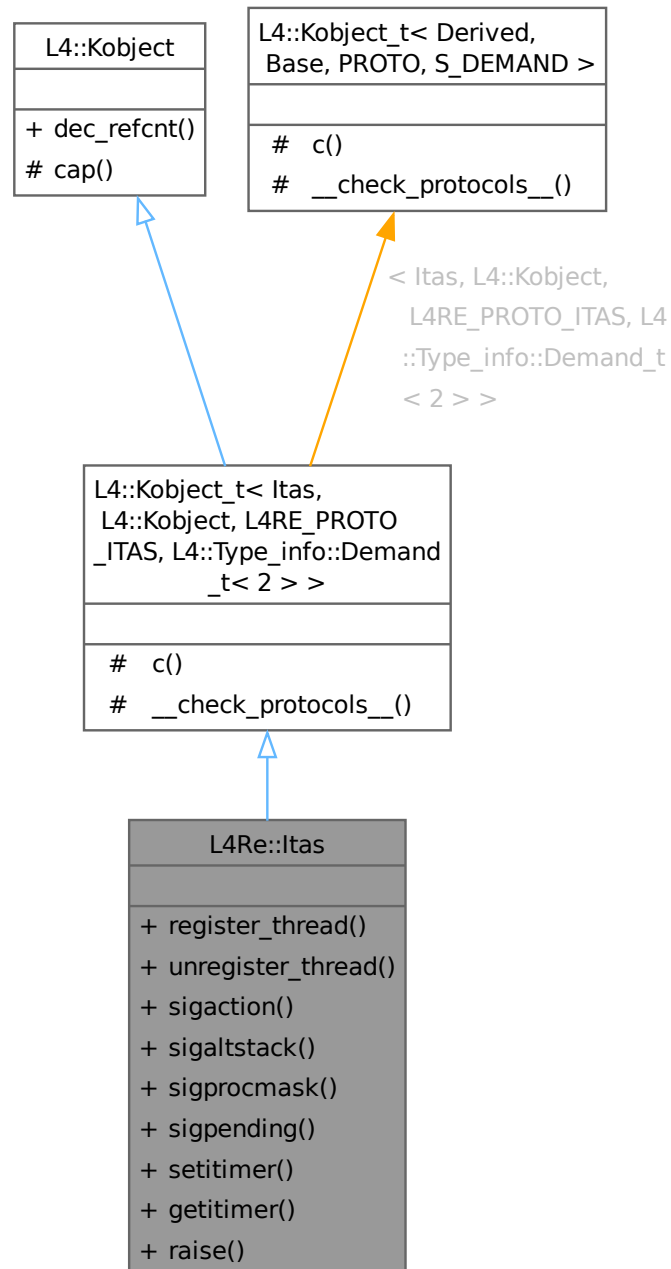
- `l4/re/inhibitor`

15.295 L4Re::Itas Class Reference

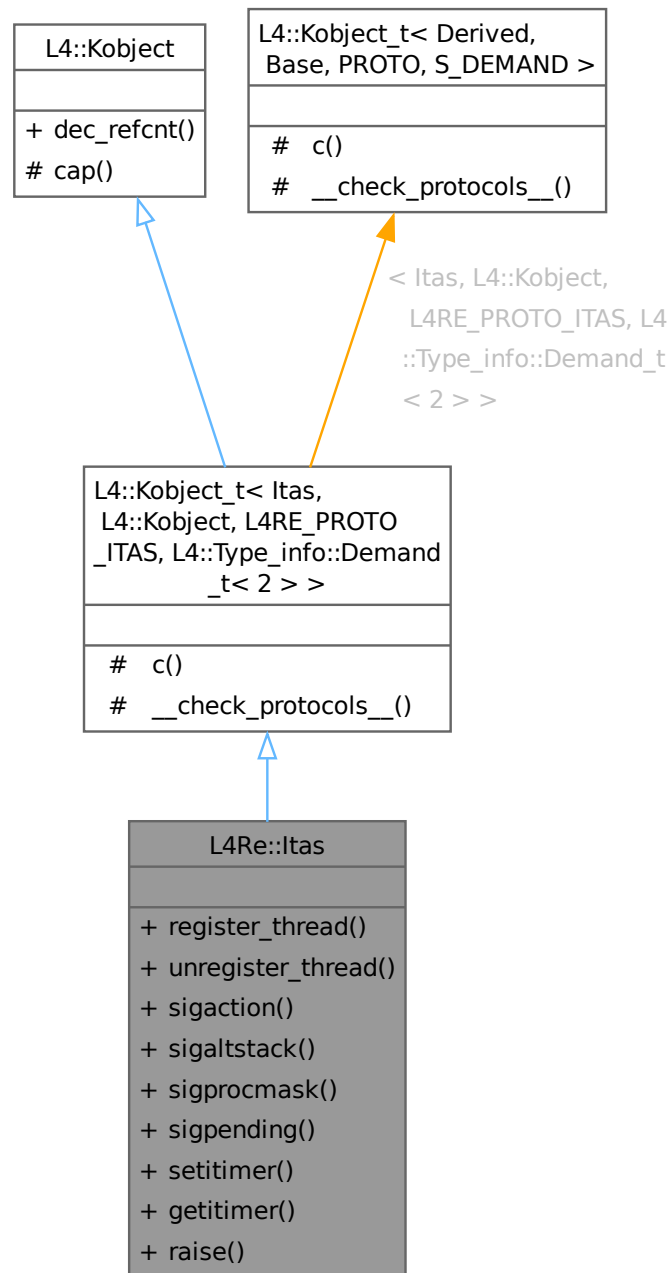
Interface to the ITAS.

```
#include <itas>
```

Inheritance diagram for L4Re::Itas:



Collaboration diagram for L4Re::Itas:



Public Types

- enum : unsigned { `ignore_sigaction` = ~0U }

Public Member Functions

- int `register_thread` (L4::lpc::Cap< L4::Thread > parent, L4::lpc::Cap< L4::Thread > thread_cap, l4_addr_t thread_utcb)

- *Register new thread.*
- `int unregister_thread (L4::lpc::Cap< L4::Thread > thread)`
- *Unregister a thread.*
- `int sigaction (int signum, const struct sigaction *act, struct sigaction *oldact)`
- *Examine and change a POSIX signal action.*
- `int sigaltstack (L4::lpc::Cap< L4::Thread > thread, const struct sigaltstack *ss, struct sigaltstack *oss)`
- *Examine or set alternate POSIX signal stack.*
- `int sigprocmask (L4::lpc::Cap< L4::Thread > thread, int how, sigset_t const *set, sigset_t *oldset)`
- *Examine or set process signal mask.*
- `int sigpending (L4::lpc::Cap< L4::Thread > thread, sigset_t *set)`
- *Query pending signals.*
- `int setitimer (int which, const struct itimerval *new_value, struct itimerval *old_value)`
- *Set process interval timer.*
- `int getitimer (int which, struct itimerval *curr_value)`
- *Get process interval timer.*
- `int raise (L4::lpc::Cap< L4::Thread > thread, int sig)`
- *Send a signal to the calling thread.*

Public Member Functions inherited from L4::Kobject

- `l4_msgtag_t dec_refcnt (l4_mword_t diff, l4_utcb_t *utcb=l4_utcb())`
- *Decrement the in kernel reference counter for the object.*

Additional Inherited Members

Protected Types inherited from

L4::Kobject_t< Itas, L4::Kobject, L4RE_PROTO_ITAS, L4::Type_info::Demand_t< 2 > >

- `typedef Itas Class`
- *The target interface type (inheriting from [Kobject_t](#)).*
- `typedef Typeid::Iface< PROTO, Itas > __Iface`
- *The interface description for the derived class.*
- `typedef Typeid::Merge_list< Typeid::Iface_list< __Iface >, typename L4::Kobject::__Iface_list > __Iface_list`
- *The list of all RPC interfaces provided directly or through inheritance.*

Protected Member Functions inherited from

L4::Kobject_t< Itas, L4::Kobject, L4RE_PROTO_ITAS, L4::Type_info::Demand_t< 2 > >

- `L4::Cap< Class > c () const noexcept`
- *Get the capability to ourselves.*

Protected Member Functions inherited from L4::Kobject

- `l4_cap_idx_t cap () const noexcept`
- *Return capability selector.*

Static Protected Member Functions inherited from**[L4::Kobject_t< Itas, L4::Kobject, L4RE_PROTO_ITAS, L4::Type_info::Demand_t< 2 > >](#)**

- static void **__check_protocols__** () noexcept
Helper to check for protocol conflicts.

15.295.1 Detailed Description

Interface to the ITAS.

This is an internal interface between libc and the l4re_itas. Do not use it. It is subject to change.

Definition at line 30 of file [itas](#).

15.295.2 Member Enumeration Documentation**15.295.2.1 anonymous enum**

```
anonymous enum : unsigned
```

Enumerator

| | |
|------------------|--|
| Ignore_sigaction | Ignore new action of sigaction() call. |
|------------------|--|

Definition at line 63 of file [itas](#).

15.295.3 Member Function Documentation**15.295.3.1 getitimer()**

```
int L4Re::Itas::getitimer (
    int which,
    struct itimerval * curr_value)
```

Get process interval timer.

See IEEE Std 1003.1-2017 [getitimer\(\)](#) for details.

Parameters

| | | |
|-----|-------------------|---------------------------|
| in | <i>which</i> | Timer type (ITIMER_REAL). |
| out | <i>curr_value</i> | Old timer value. |

References [getitimer\(\)](#).

Referenced by [getitimer\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.295.3.2 raise()

```

int L4Re::Itas::raise (
    L4::Ipc::Cap< L4::Thread > thread,
    int sig)
  
```

Send a signal to the calling thread.

Parameters

| | | |
|----|---------------|-----------------------------------|
| in | <i>thread</i> | Thread cap of the calling thread. |
| in | <i>sig</i> | Signal that shall be raised. |

References [raise\(\)](#).

Referenced by [raise\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.295.3.3 register_thread()

```
int L4Re::Itas::register_thread (
    L4::Ipc::Cap< L4::Thread > parent,
    L4::Ipc::Cap< L4::Thread > thread_cap,
    l4_addr_t thread_utcb)
```

Register new thread.

Makes the newly created thread known to ITAS. The ITAS will do the `thread_cap->control()` to bind the thread to the task and attach the gates for pager and exception handler.

Parameters

| | |
|--------------------|---|
| <i>parent</i> | The capability of the thread that created the new thread. |
| <i>thread_cap</i> | The capability of the new thread. |
| <i>thread_utcb</i> | The address of the allocated UTCB of the new thread. |

15.295.3.4 setitimer()

```
int L4Re::Itas::setitimer (
    int which,
    const struct itimerval * new_value,
    struct itimerval * old_value)
```

Set process interval timer.

See IEEE Std 1003.1-2017 [setitimer\(\)](#) for details.

Parameters

| | | |
|-----|------------------|---------------------------|
| in | <i>which</i> | Timer type (ITIMER_REAL). |
| in | <i>new_value</i> | New timer value. |
| out | <i>old_value</i> | Old timer value. |

Referenced by [sigpending\(\)](#).

Here is the caller graph for this function:



15.295.3.5 sigaction()

```

int L4Re::Itas::sigaction (
    int signum,
    const struct sigaction * act,
    struct sigaction * oldact)
  
```

Examine and change a POSIX signal action.

See IEEE Std 1003.1-2024 [sigaction\(\)](#) for the behaviour of the method.

If `act->sa_flags` is [Ignore_sigaction](#), the new action is ignored.

Parameters

| | | |
|-----|---------------|---|
| in | <i>signum</i> | Signal number to be examined and/or modified. |
| in | <i>act</i> | New signal action. |
| out | <i>oldact</i> | Old signal action. |

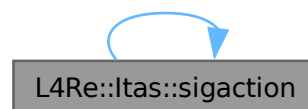
References [sigaction\(\)](#), and [sigaltstack\(\)](#).

Referenced by [sigaction\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.295.3.6 sigaltstack()

```
int L4Re::Itas::sigaltstack (
    L4::Ipc::Cap< L4::Thread > thread,
    const struct sigaltstack * ss,
    struct sigaltstack * oss)
```

Examine or set alternate POSIX signal stack.

See IEEE Std 1003.1-2024 [sigaltstack\(\)](#) for the behaviour of the method.

If `ss->ss_flags` is -1, the new sigaltstack will be ignored.

Parameters

| | | |
|-----|---------------|---|
| in | <i>thread</i> | Thread cap of the thread whose sigaltstack is examined and/or modified. |
| in | <i>ss</i> | The new sigaltstack. |
| out | <i>oss</i> | The old sigaltstack. |

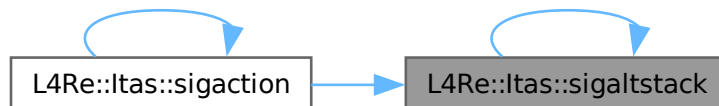
References [sigaltstack\(\)](#), and [sigprocmask\(\)](#).

Referenced by [sigaction\(\)](#), and [sigaltstack\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.295.3.7 sigpending()

```
int L4Re::Itas::sigpending (
    L4::Ipc::Cap< L4::Thread > thread,
    sigset_t * set)
```

Query pending signals.

See IEEE Std 1003.1-2024 [sigpending\(\)](#) for details.

Parameters

| | | |
|-----|---------------|---|
| in | <i>thread</i> | Thread cap of the thread whose pending signal are examined. |
| out | <i>set</i> | Pending signals of thread. |

References [setitimer\(\)](#).

Referenced by [sigprocmask\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.295.3.8 sigprocmask()

```
int L4Re::Itas::sigprocmask (
    L4::Ipc::Cap< L4::Thread > thread,
    int how,
    sigset_t const * set,
    sigset_t * oldset)
```

Examine or set process signal mask.

See IEEE Std 1003.1-2024 [sigprocmask\(\)](#) for the behaviour or the method.

If `how` is -1, the signal mask is left unchanged.

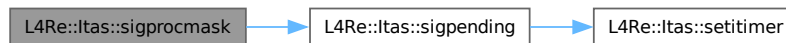
Parameters

| | | |
|-----|---------------|---|
| in | <i>thread</i> | Thread cap of the thread whose signal mask is examined and/or modified. |
| in | <i>how</i> | Operation (SIG_BLOCK, SIG_UNBLOCK, SIG_SETMASK or -1). |
| in | <i>set</i> | The new signal mask. |
| out | <i>oldset</i> | The old signal mask. |

References [sigpending\(\)](#).

Referenced by [sigaltstack\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.295.3.9 unregister_thread()

```
int L4Re::Itas::unregister_thread (
    L4::Ipc::Cap< L4::Thread > thread)
```

Unregister a thread.

The gates for the thread's pager and exception handler will be destroyed. Thus, the thread must be destroyed after the call.

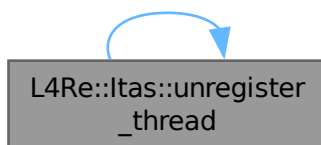
Parameters

| | |
|---------------|-----------------------|
| <i>thread</i> | The destroyed thread. |
|---------------|-----------------------|

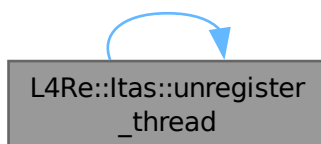
References [unregister_thread\(\)](#).

Referenced by [unregister_thread\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

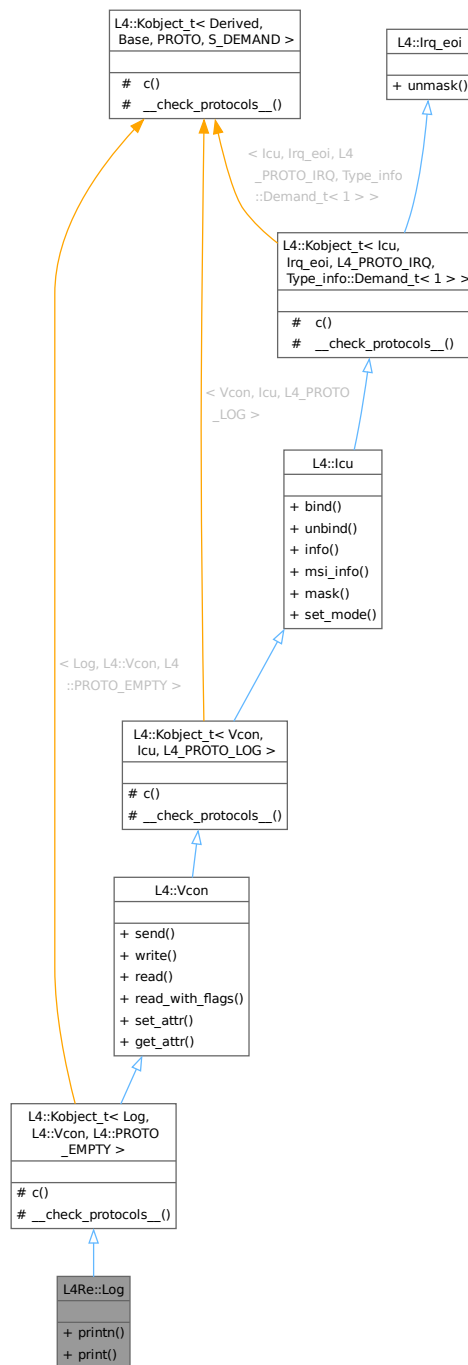
- `I4/re/itas`

15.296 L4Re::Log Class Reference

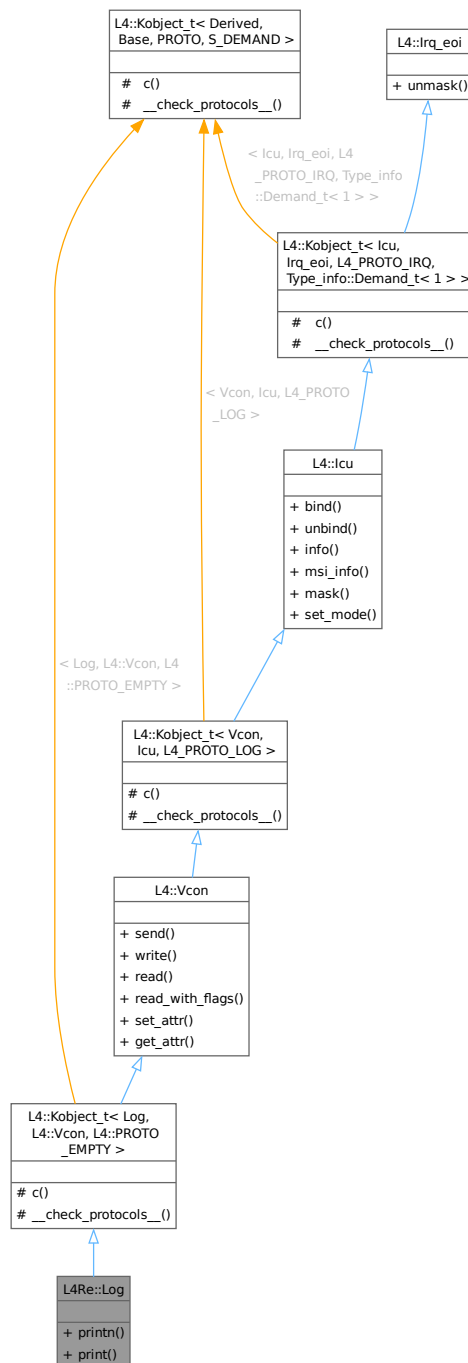
[Log](#) interface class.

```
#include <log>
```

Inheritance diagram for L4Re::Log:



Collaboration diagram for L4Re::Log:



Public Member Functions

- void `printn` (char const *string, int len) const noexcept
Print string with length len, NULL characters don't matter.
- void `print` (char const *string) const noexcept
Print NULL-terminated string.

Public Member Functions inherited from [L4::Vcon](#)

- [l4_msgtag_t send](#) (char const *buf, unsigned size, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) const noexcept
Send data to `this` virtual console.
- long [write](#) (char const *buf, unsigned size, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) const noexcept
Write data to `this` virtual console.
- int [read](#) (char *buf, unsigned size, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) const noexcept
Read data from `this` virtual console.
- int [read_with_flags](#) (char *buf, unsigned size, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) const noexcept
Read data from `this` virtual console which also returns flags.
- [l4_msgtag_t set_attr](#) ([l4_vcon_attr_t](#) const *attr, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) const noexcept
Set the attributes of `this` virtual console.
- [l4_msgtag_t get_attr](#) ([l4_vcon_attr_t](#) *attr, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) const noexcept
Get attributes of `this` virtual console.

Public Member Functions inherited from [L4::Icu](#)

- [l4_msgtag_t bind](#) (unsigned irqnum, [L4::Cap< Triggerable >](#) irq, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) noexcept
Bind an interrupt line of an interrupt controller to an interrupt object.
- [l4_msgtag_t unbind](#) (unsigned irqnum, [L4::Cap< Triggerable >](#) irq, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) noexcept
Remove binding of an interrupt line from the interrupt controller object.
- [l4_msgtag_t info](#) ([l4_icu_info_t](#) *info, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) noexcept
Get information about the ICU features.
- [l4_msgtag_t msi_info](#) ([l4_umword_t](#) irqnum, [l4_uint64_t](#) source, [l4_icu_msi_info_t](#) *msi_info)
Get MSI info about IRQ.
- [l4_msgtag_t mask](#) (unsigned irqnum, [l4_umword_t](#) *label=0, [l4_timeout_t](#) to=[L4_IPC_NEVER](#), [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) noexcept
Mask an IRQ line.
- [l4_msgtag_t set_mode](#) (unsigned irqnum, [l4_umword_t](#) mode, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) noexcept
Set interrupt mode.

Public Member Functions inherited from [L4::Irq_eoi](#)

- [l4_msgtag_t unmask](#) (unsigned irqnum, [l4_umword_t](#) *label=0, [l4_timeout_t](#) to=[L4_IPC_NEVER](#), [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) noexcept
Unmask the given interrupt line.

Additional Inherited Members

Protected Types inherited from [L4::Kobject_t< Log, L4::Vcon, L4::PROTO_EMPTY >](#)

- typedef Log **Class**
The target interface type (inheriting from [Kobject_t](#)).
- typedef Typeid::Iface< PROTO, Log > **__Iface**
The interface description for the derived class.
- typedef Typeid::Merge_list< Typeid::Iface_list< **__Iface** >, typename L4::Vcon::__Iface_list > **__Iface_list**
The list of all RPC interfaces provided directly or through inheritance.

Protected Types inherited from [L4::Kobject_t< Vcon, Icu, L4_PROTO_LOG >](#)

- typedef [Vcon](#) **Class**
The target interface type (inheriting from [Kobject_t](#)).
- typedef Typeid::Iface< PROTO, [Vcon](#) > **__Iface**
The interface description for the derived class.
- typedef Typeid::Merge_list< Typeid::Iface_list< [__Iface](#) >, typename Icu::__Iface_list > **__Iface_list**
The list of all RPC interfaces provided directly or through inheritance.

Protected Types inherited from

[L4::Kobject_t< Icu, Irq_eoi, L4_PROTO_IRQ, Type_info::Demand_t< 1 > >](#)

- typedef [Icu](#) **Class**
The target interface type (inheriting from [Kobject_t](#)).
- typedef Typeid::Iface< PROTO, [Icu](#) > **__Iface**
The interface description for the derived class.
- typedef Typeid::Merge_list< Typeid::Iface_list< [__Iface](#) >, typename Irq_eoi::__Iface_list > **__Iface_list**
The list of all RPC interfaces provided directly or through inheritance.

Protected Member Functions inherited from

[L4::Kobject_t< Log, L4::Vcon, L4::PROTO_EMPTY >](#)

- [L4::Cap< Class > c \(\)](#) const noexcept
Get the capability to ourselves.

Protected Member Functions inherited from

[L4::Kobject_t< Vcon, Icu, L4_PROTO_LOG >](#)

- [L4::Cap< Class > c \(\)](#) const noexcept
Get the capability to ourselves.

Protected Member Functions inherited from

[L4::Kobject_t< Icu, Irq_eoi, L4_PROTO_IRQ, Type_info::Demand_t< 1 > >](#)

- [L4::Cap< Class > c \(\)](#) const noexcept
Get the capability to ourselves.

Static Protected Member Functions inherited from

[L4::Kobject_t< Log, L4::Vcon, L4::PROTO_EMPTY >](#)

- static void [__check_protocols__ \(\)](#) noexcept
Helper to check for protocol conflicts.

Static Protected Member Functions inherited from

[L4::Kobject_t< Vcon, Icu, L4_PROTO_LOG >](#)

- static void [__check_protocols__ \(\)](#) noexcept
Helper to check for protocol conflicts.

Static Protected Member Functions inherited from**[L4::Kobject_t< Icu, Irq_eoi, L4_PROTO_IRQ, Type_info::Demand_t< 1 > >](#)**

- static void **__check_protocols__**() noexcept
Helper to check for protocol conflicts.

15.296.1 Detailed Description

[Log](#) interface class.

Definition at line 33 of file [log](#).

15.296.2 Member Function Documentation**15.296.2.1 print()**

```
void L4Re::Log::print (
    char const * string) const    [noexcept]
```

Print NULL-terminated string.

Parameters

| | |
|---------------|-----------------|
| <i>string</i> | string to print |
|---------------|-----------------|

15.296.2.2 printn()

```
void L4Re::Log::printn (
    char const * string,
    int len) const    [noexcept]
```

Print string with length len, NULL characters don't matter.

Parameters

| | |
|---------------|------------------|
| <i>string</i> | string to print |
| <i>len</i> | length of string |

The documentation for this class was generated from the following file:

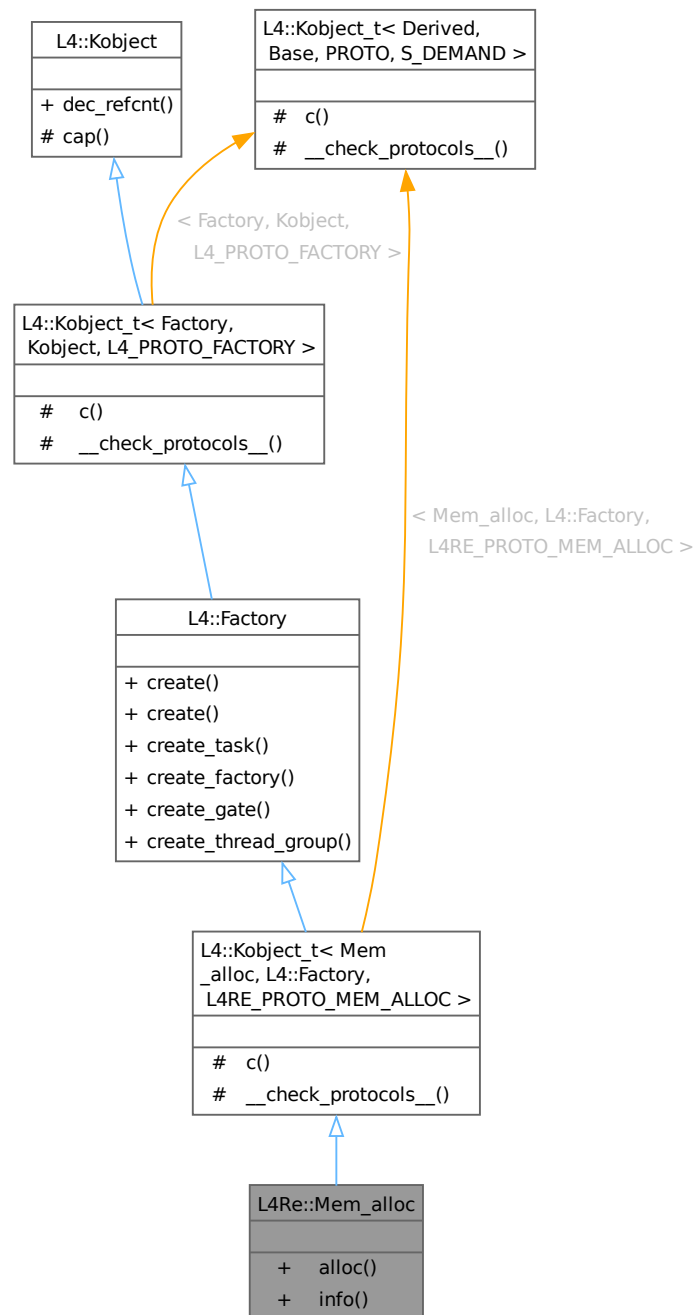
- [l4/re/log](#)

15.297 L4Re::Mem_alloc Class Reference

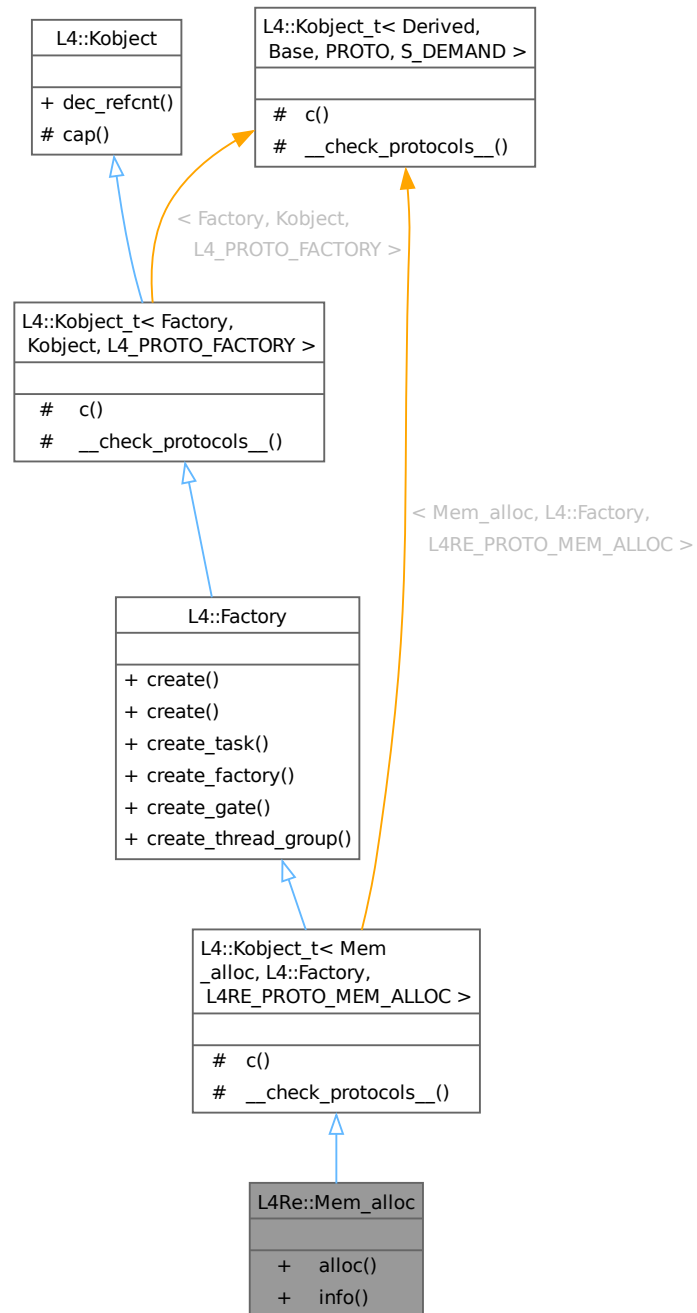
Memory allocation interface.

```
#include <mem_alloc>
```

Inheritance diagram for L4Re::Mem_alloc:



Collaboration diagram for L4Re::Mem_alloc:



Data Structures

- struct [Stats](#)
Statistics about memory-allocator.

Public Types

- enum [Mem_alloc_flags](#) { [Continuous](#) = 0x01 , [Pinned](#) = 0x02 , [Super_pages](#) = 0x04 , [Fixed_paddr](#) = 0x08 }

Flags for the allocator.

Public Member Functions

- [l4_ret_t alloc](#) (long size, [L4::Cap](#)< Dataspace > mem, unsigned long flags=0, unsigned long align=0, [l4_addr_t](#) paddr=0) const noexcept
Allocate anonymous memory.
- long [info](#) ([Stats](#) &stats)
Get allocator information.

Public Member Functions inherited from [L4::Factory](#)

- [S create](#) ([Cap](#)< void > target, long obj, [l4_utcb_t](#) *utcb=[l4_utcb](#)()) noexcept
Generic create call to the factory.
- template<typename OBJ>
[S create](#) ([Cap](#)< OBJ > target, [l4_utcb_t](#) *utcb=[l4_utcb](#)()) noexcept
Create call for typed capabilities.
- [l4_msgtag_t create_task](#) ([Cap](#)< [Task](#) > const &target_cap, [l4_fpage_t](#) *utcb_area, [l4_utcb_t](#) *utcb=[l4_utcb](#)()) noexcept
Create a new task.
- [l4_msgtag_t create_factory](#) ([Cap](#)< [Factory](#) > const &target_cap, unsigned long limit, [l4_utcb_t](#) *utcb=[l4_utcb](#)()) noexcept
Create a new factory.
- [l4_msgtag_t create_gate](#) ([Cap](#)< void > const &target_cap, [Cap](#)< [Snd_destination](#) > const &snd_dst_cap, [l4_umword_t](#) label, [l4_utcb_t](#) *utcb=[l4_utcb](#)()) noexcept
Create a new IPC gate, optionally bound to a send destination (a thread or thread group).
- [l4_msgtag_t create_thread_group](#) ([Cap](#)< [Thread_group](#) > const &target_cap, unsigned policy, [l4_utcb_t](#) *utcb=[l4_utcb](#)()) noexcept
Create a new thread group.

Public Member Functions inherited from [L4::Kobject](#)

- [l4_msgtag_t dec_refcnt](#) ([l4_mword_t](#) diff, [l4_utcb_t](#) *utcb=[l4_utcb](#)())
Decrement the in kernel reference counter for the object.

Additional Inherited Members

Protected Types inherited from

[L4::Kobject_t](#)< [Mem_alloc](#), [L4::Factory](#), [L4RE_PROTO_MEM_ALLOC](#) >

- typedef [Mem_alloc](#) **Class**
The target interface type (inheriting from [Kobject_t](#)).
- typedef [Typeid::Iface](#)< [PROTO](#), [Mem_alloc](#) > **__Iface**
The interface description for the derived class.
- typedef [Typeid::Merge_list](#)< [Typeid::Iface_list](#)< **__Iface** >, typename [L4::Factory::__Iface_list](#) > **__Iface_list**
The list of all RPC interfaces provided directly or through inheritance.

Protected Types inherited from [L4::Kobject_t< Factory, Kobject, L4_PROTO_FACTORY >](#)

- typedef [Factory](#) **Class**
The target interface type (inheriting from [Kobject_t](#)).
- typedef Typeid::Iface< [PROTO](#), [Factory](#) > **__Iface**
The interface description for the derived class.
- typedef Typeid::Merge_list< Typeid::Iface_list< **__Iface** >, typename Kobject::__Iface_list > **__Iface_list**
The list of all RPC interfaces provided directly or through inheritance.

Protected Member Functions inherited from [L4::Kobject_t< Mem_alloc, L4::Factory, L4RE_PROTO_MEM_ALLOC >](#)

- [L4::Cap< Class > c](#) () const noexcept
Get the capability to ourselves.

Protected Member Functions inherited from [L4::Kobject_t< Factory, Kobject, L4_PROTO_FACTORY >](#)

- [L4::Cap< Class > c](#) () const noexcept
Get the capability to ourselves.

Protected Member Functions inherited from [L4::Kobject](#)

- [l4_cap_idx_t cap](#) () const noexcept
Return capability selector.

Static Protected Member Functions inherited from [L4::Kobject_t< Mem_alloc, L4::Factory, L4RE_PROTO_MEM_ALLOC >](#)

- static void [__check_protocols](#) () noexcept
Helper to check for protocol conflicts.

Static Protected Member Functions inherited from [L4::Kobject_t< Factory, Kobject, L4_PROTO_FACTORY >](#)

- static void [__check_protocols](#) () noexcept
Helper to check for protocol conflicts.

15.297.1 Detailed Description

Memory allocation interface.

The memory-allocator API is the basic API to allocate memory from the [L4Re](#) subsystem. The memory is allocated in terms of dataspace (see [L4Re::Dataspace](#)). The provided dataspace have at least the property that data written to such a dataspace is available as long as the dataspace is not freed or the data is not overwritten. In particular, the memory backing a dataspace from an allocator need not be allocated instantly, but may be allocated lazily on demand.

A memory allocator can provide dataspace with additional properties, such as physically contiguous memory, pre-allocated memory, or pinned memory. To request memory with an additional property the [L4Re::Mem_alloc::alloc\(\)](#) method provides a flags parameter. If the concrete implementation of a memory allocator does not support or allow allocation of memory with a certain property, the allocation may be refused.

Definition at line 52 of file [mem_alloc](#).

15.297.2 Member Enumeration Documentation

15.297.2.1 Mem_alloc_flags

```
enum L4Re::Mem_alloc::Mem_alloc_flags
```

Flags for the allocator.

They describe requested properties of the allocated memory. Support of these properties by the dataspace provider is optional.

Enumerator

| | |
|-------------|---|
| Continuous | Allocate physically contiguous memory. |
| Pinned | Deprecated, use L4Re::Dma_space instead. |
| Super_pages | Allocate super pages. |
| Fixed_paddr | Allocate at fixed physical address. Only honored on no-MMU systems. Will fail on MMU systems. |

Definition at line 62 of file [mem_alloc](#).

15.297.3 Member Function Documentation

15.297.3.1 alloc()

```
L4_ret_t L4Re::Mem_alloc::alloc (
    long size,
    L4::Cap< Dataspace > mem,
    unsigned long flags = 0,
    unsigned long align = 0,
    L4_addr_t paddr = 0) const [noexcept]
```

Allocate anonymous memory.

Parameters

| | | |
|-----|--------------|--|
| | <i>size</i> | Size in bytes to be requested. Allocation granularity is (super)pages, however, the allocator will store the byte-granular given size as the size of the dataspace and consecutively will use this byte-granular size for servicing the dataspace. Allocators may optionally also implement a maximum allocation strategy: if <i>size</i> is a negative value and <i>flags</i> set the Mem_alloc_flags::Continuous bit, the allocator tries to allocate as much memory as possible leaving an amount of at least <i>-size</i> bytes within the associated quota. |
| out | <i>mem</i> | Capability slot where the capability to the dataspace is received. |
| | <i>flags</i> | Special dataspace properties, see Mem_alloc_flags |
| | <i>align</i> | Log2 alignment of dataspace if supported by allocator, will be at least L4_PAGESHIFT, with Super_pages flag set at least L4_SUPERPAGESHIFT |
| | <i>paddr</i> | The physical address where the dataspace should be allocated if Mem_alloc_flags::Fixed flag is set. |

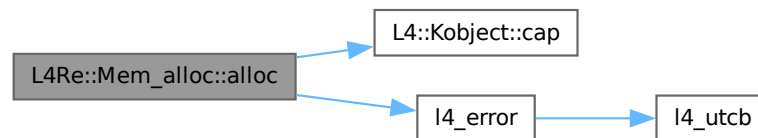
Return values

| | |
|------------|------------------------------|
| 0 | Success |
| -L4_ERANGE | Given size not supported. |
| -L4_ENOMEM | Not enough memory available. |
| <0 | IPC error |

Definition at line 24 of file [mem_alloc_impl.h](#).

References [L4::Kobject::cap\(\)](#), [Fixed_paddr](#), and [l4_error\(\)](#).

Here is the call graph for this function:

**15.297.3.2 info()**

```
long L4Re::Mem_alloc::info (  
    Stats & stats)
```

Get allocator information.

Parameters

| | | |
|-----|-------|-----------------------|
| out | stats | Allocator information |
|-----|-------|-----------------------|

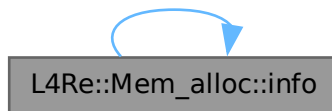
Return values

| | |
|----|-----------|
| 0 | Success |
| <0 | IPC error |

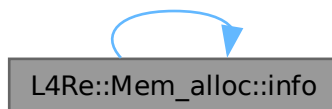
References [info\(\)](#), and [L4_INLINE_RPC](#).

Referenced by [info\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following files:

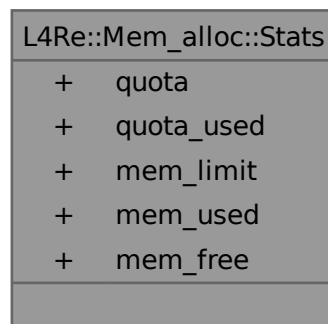
- [l4/re/mem_alloc](#)
- [l4/re/impl/mem_alloc_impl.h](#)

15.298 L4Re::Mem_alloc::Stats Struct Reference

Statistics about memory-allocator.

```
#include <mem_alloc>
```

Collaboration diagram for L4Re::Mem_alloc::Stats:



Data Fields

- [l4_size_t quota](#)
Memory quota of this allocator.
- [l4_size_t quota_used](#)
Amount of currently used quota of this allocator.
- [l4_size_t mem_limit](#)
Maximum amount of memory that can be allocated by this allocator.
- [l4_size_t mem_used](#)
Amount of currently allocated memory.
- [l4_size_t mem_free](#)
Amount of memory that is still available for allocation.

15.298.1 Detailed Description

Statistics about memory-allocator.

Definition at line 74 of file [mem_alloc](#).

15.298.2 Field Documentation

15.298.2.1 mem_free

[l4_size_t](#) L4Re::Mem_alloc::Stats::mem_free

Amount of memory that is still available for allocation.

This field can be lower than [mem_limit](#) - [mem_used](#). In this case the system may be over-committed and there is globally not enough memory left. Also, if the quota is already used up for sub-factories (see [quota_used](#)), there may be not enough quota left.

Definition at line 126 of file [mem_alloc](#).

15.298.2.2 mem_limit

[l4_size_t](#) L4Re::Mem_alloc::Stats::mem_limit

Maximum amount of memory that can be allocated by this allocator.

Will never exceed the [quota](#) but may be smaller if the system has less memory installed.

Definition at line 102 of file [mem_alloc](#).

15.298.2.3 mem_used

`l4_size_t L4Re::Mem_alloc::Stats::mem_used`

Amount of currently allocated memory.

This field represents the amount of memory that is in use by this allocator. It recursively includes the memory used by sub-factories, if any.

Will never exceed `mem_limit` or `quota_used`.

Note

Dataspaces may allocate memory lazily! As such, the field will increase only after pages have been allocated to a dataspace.

Definition at line 116 of file `mem_alloc`.

15.298.2.4 quota

`l4_size_t L4Re::Mem_alloc::Stats::quota`

Memory quota of this allocator.

Strictly limits the amount of memory that can be allocated. This may be larger than there is actual physical memory available. In particular, the root factory has an artificial quota and returns -1 in this field.

Definition at line 83 of file `mem_alloc`.

15.298.2.5 quota_used

`l4_size_t L4Re::Mem_alloc::Stats::quota_used`

Amount of currently used quota of this allocator.

The amount of used quota is not necessarily linked to the current memory usage. See `mem_used` for this information. The quota of a factory is immediately and fully accounted to the parent factory quota.

This value may even exceed `mem_limit` if the system is over-committed.

Definition at line 94 of file `mem_alloc`.

The documentation for this struct was generated from the following file:

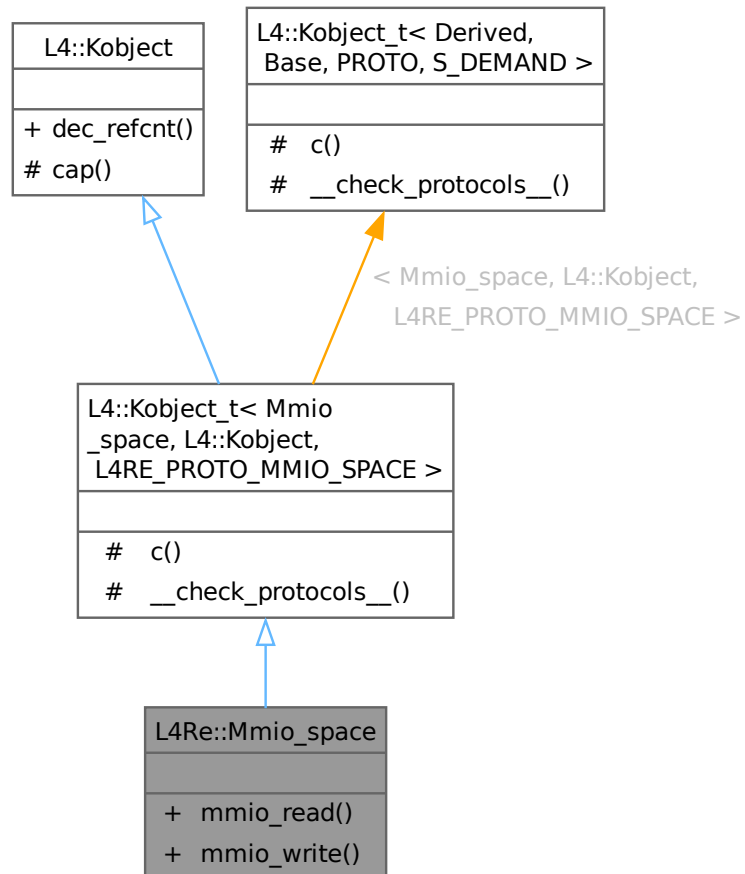
- `l4/re/mem_alloc`

15.299 L4Re::Mmio_space Struct Reference

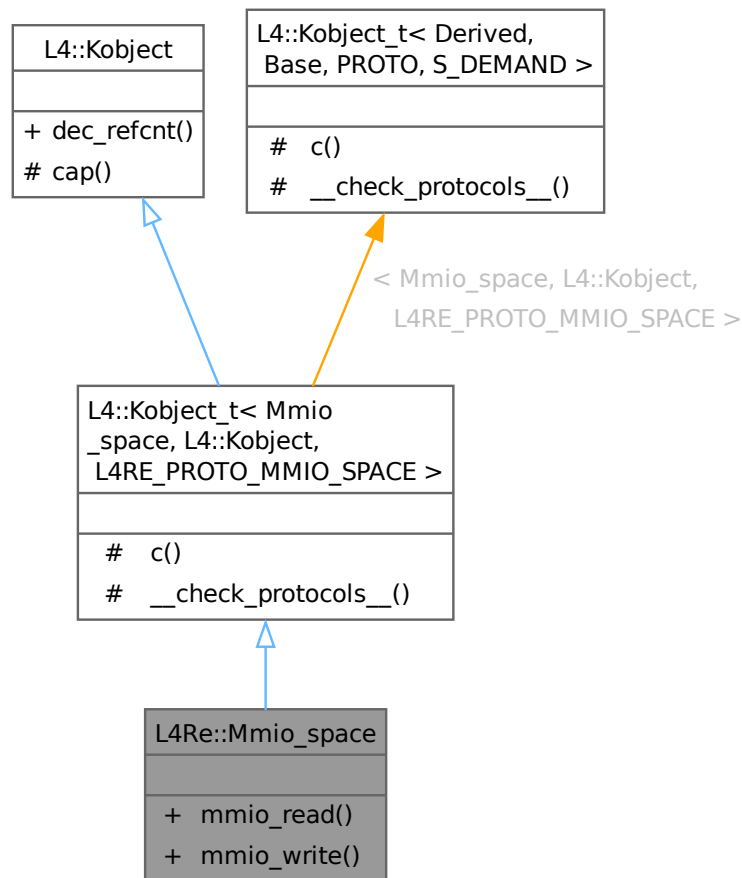
Interface for memory-like address space accessible via IPC.

```
#include <mmio_space>
```

Inheritance diagram for L4Re::Mmio_space:



Collaboration diagram for L4Re::Mmio_space:



Public Types

- enum `Access_width` { `Wd_8bit = 0` , `Wd_16bit = 1` , `Wd_32bit = 2` , `Wd_64bit = 3` }
Actual size of the value to read or write.
- typedef `l4_uint64_t Addr`
Device address.

Public Member Functions

- `l4_ret_t mmio_read (Addr addr, char width, l4_uint64_t *value)`
Read a value from the given address.
- `l4_ret_t mmio_write (Addr addr, char width, l4_uint64_t value)`
Write a value to the given address.

Public Member Functions inherited from L4::Kobject

- `l4_msgtag_t dec_refcnt (l4_mword_t diff, l4_utcb_t *utcb=l4_utcb())`
Decrement the in kernel reference counter for the object.

Additional Inherited Members

Protected Types inherited from

[L4::Kobject_t](#)< [Mmio_space](#), [L4::Kobject](#), [L4RE_PROTO_MMIO_SPACE](#) >

- typedef [Mmio_space](#) **Class**
The target interface type (inheriting from [Kobject_t](#)).
- typedef Typeid::Iface< [PROTO](#), [Mmio_space](#) > **__iface**
The interface description for the derived class.
- typedef Typeid::Merge_list< Typeid::Iface_list< **__iface** >, typename [L4::Kobject::__iface_list](#) > **__iface_list**
The list of all RPC interfaces provided directly or through inheritance.

Protected Member Functions inherited from

[L4::Kobject_t](#)< [Mmio_space](#), [L4::Kobject](#), [L4RE_PROTO_MMIO_SPACE](#) >

- [L4::Cap](#)< [Class](#) > **c** () const noexcept
Get the capability to ourselves.

Protected Member Functions inherited from [L4::Kobject](#)

- [l4_cap_idx_t](#) **cap** () const noexcept
Return capability selector.

Static Protected Member Functions inherited from

[L4::Kobject_t](#)< [Mmio_space](#), [L4::Kobject](#), [L4RE_PROTO_MMIO_SPACE](#) >

- static void **__check_protocols__** () noexcept
Helper to check for protocol conflicts.

15.299.1 Detailed Description

Interface for memory-like address space accessible via IPC.

This interface defines methods for indirect access to MMIO regions.

Memory mapped IO (MMIO) is used by device drivers to control hardware devices. Access to MMIO regions is assigned to user-level device drivers via mappings of memory pages.

However, there are hardware platforms where MMIO regions for different devices share the same memory page. With respect to security and safety, it is often not allowed to map a memory page to multiple device drivers because the driver of one device could then influence operation of another device, which violates security boundaries.

A solution to that problem is to implement a third (trusted) component that gets exclusive access to the shared memory page, and that drivers can access via IPC with the [Mmio_space](#) protocol. This proxy-component can then enforce an access policy.

Include File

```
#include <l4/re/mmio_space>
```

Definition at line 45 of file [mmio_space](#).

15.299.2 Member Enumeration Documentation

15.299.2.1 Access_width

```
enum L4Re::Mmio_space::Access_width
```

Actual size of the value to read or write.

Enumerator

| | |
|----------|-------------------------|
| Wd_8bit | Value is a byte. |
| Wd_16bit | Value is a 2-byte word. |
| Wd_32bit | Value is a 4-byte word. |
| Wd_64bit | Value is a 8-byte word. |

Definition at line 49 of file [mmio_space](#).

15.299.3 Member Function Documentation

15.299.3.1 mmio_read()

```
l4_ret_t L4Re::Mmio_space::mmio_read (
    Addr addr,
    char width,
    l4_uint64_t * value)
```

Read a value from the given address.

Parameters

| | | |
|-----|--------------|--|
| | <i>addr</i> | Device virtual address to read from. The address must be aligned relative to the access width. |
| | <i>width</i> | Access width of value to be read, see Access_width . |
| out | <i>value</i> | Return value. If width is smaller than 64 bit,the upper bits are guaranteed to be 0. |

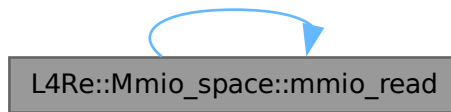
Return values

| | |
|----------------------------|--|
| L4_EOK | Success. |
| -L4_EPERM | Insufficient read rights. |
| -L4_EINVAL | Address does not exist or cannot be accessed with the given width. |

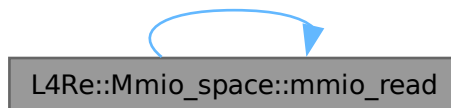
References [mmio_read\(\)](#).

Referenced by [mmio_read\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.299.3.2 mmio_write()

```
l4_ret_t L4Re::Mmio_space::mmio_write (  
    Addr addr,  
    char width,  
    l4_uint64_t value)
```

Write a value to the given address.

Parameters

| | |
|--------------|---|
| <i>addr</i> | Device virtual address to write to. The address must be aligned relative to the access width. |
| <i>width</i> | Access width of value to write, see Access_width . |
| <i>value</i> | Value to write. If width is smaller than 64 bit, the upper bits are ignored. |

Return values

| | |
|-------------------------|--|
| L4_EOK | Success. |
| <code>-L4_EPERM</code> | Insufficient write rights. |
| <code>-L4_EINVAL</code> | Address does not exist or cannot be accessed with the given width. |

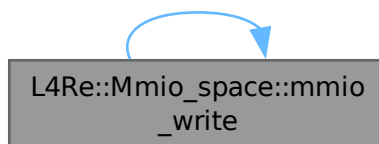
References [mmio_write\(\)](#).

Referenced by [mmio_write\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this struct was generated from the following file:

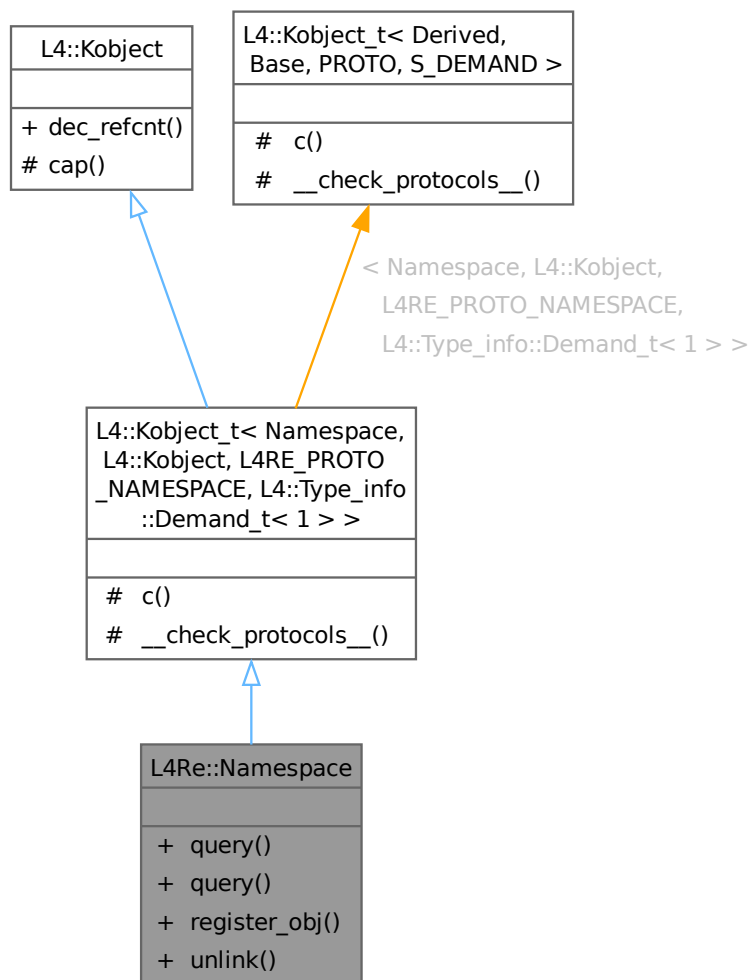
- [l4/re/mmio_space](#)

15.300 L4Re::Namespace Class Reference

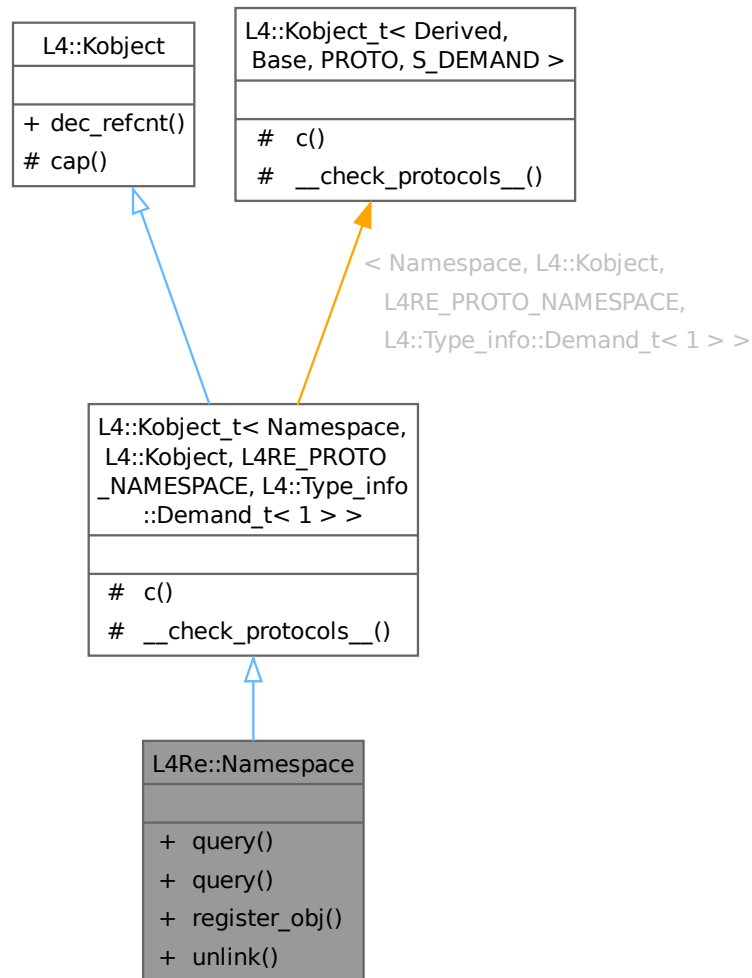
Name-space interface.

```
#include <namespace>
```

Inheritance diagram for L4Re::Namespace:



Collaboration diagram for L4Re::Namespace:



Public Types

- enum **Register_flags** {
Ro = L4_CAP_FPAGE_RO , **Rw** = L4_CAP_FPAGE_RW , **Rs** = L4_CAP_FPAGE_RS , **Rws** = L4_CAP_FPAGE_RWS ,
Strong = L4_CAP_FPAGE_S , **Trusted** = 0x008 , **Cap_flags** = Ro | Rw | Strong | Trusted , **Link** = 0x100 ,
Overwrite = 0x200 }
Flags for registering name spaces.
- enum **Query_result_flags** { **Partly_resolved** = 0x020 }
Flags returned by query IPC, only used internally.
- enum **Query_timeout** { **To_default** = 3600000 , **To_non_blocking** = 0 }
Timeout values for query operation.

Public Member Functions

- l4_ret_t query** (char const *name, L4::Cap< void > const &cap, int timeout=To_default, l4_umword_t *local_id=0, bool iterate=true) const noexcept

Query the name space for a named object.

- [l4_ret_t query](#) (char const *name, unsigned len, [L4::Cap](#)< void > const &cap, int timeout=[To_default](#), [l4_umword_t](#) *local_id=0, bool iterate=true) const noexcept

Query the name space for a named object.

- [l4_ret_t register_obj](#) (char const *name, [L4::lpc::Cap](#)< void > obj, unsigned flags=[Rw](#)) const noexcept

Register an object with a name.

- [l4_ret_t unlink](#) (char const *name)

Remove an entry from the name space.

Public Member Functions inherited from [L4::Kobject](#)

- [l4_msgtag_t dec_refcnt](#) ([l4_mword_t](#) diff, [l4_utcb_t](#) *utcb=[l4_utcb](#)())

Decrement the in kernel reference counter for the object.

Additional Inherited Members

Protected Types inherited from

[L4::Kobject_t](#)< [Namespace](#), [L4::Kobject](#), [L4RE_PROTO_NAMESPACE](#), [L4::Type_info::Demand_t](#)< 1 >

- typedef [Namespace](#) **Class**

The target interface type (inheriting from [Kobject_t](#)).

- typedef [Typeid::Iface](#)< [PROTO](#), [Namespace](#) > **__iface**

The interface description for the derived class.

- typedef [Typeid::Merge_list](#)< [Typeid::Iface_list](#)< **__iface** >, typename [L4::Kobject::__iface_list](#) > **__iface_list**

The list of all RPC interfaces provided directly or through inheritance.

Protected Member Functions inherited from

[L4::Kobject_t](#)< [Namespace](#), [L4::Kobject](#), [L4RE_PROTO_NAMESPACE](#), [L4::Type_info::Demand_t](#)< 1 >

- [L4::Cap](#)< [Class](#) > **c** () const noexcept

Get the capability to ourselves.

Protected Member Functions inherited from [L4::Kobject](#)

- [l4_cap_idx_t cap](#) () const noexcept

Return capability selector.

Static Protected Member Functions inherited from

[L4::Kobject_t](#)< [Namespace](#), [L4::Kobject](#), [L4RE_PROTO_NAMESPACE](#), [L4::Type_info::Demand_t](#)< 1 >

- static void **__check_protocols** () noexcept

Helper to check for protocol conflicts.

15.300.1 Detailed Description

Name-space interface.

All name space objects must provide this interface. However, it is not mandatory that a name space object allows to register new capabilities.

The name lookup is done iteratively, this means the hierarchical names are resolved component wise by the client itself.

Definition at line 49 of file [namespace](#).

15.300.2 Member Enumeration Documentation

15.300.2.1 Query_result_flags

```
enum L4Re::Namespace::Query_result_flags
```

Flags returned by query IPC, only used internally.

Enumerator

| | |
|-----------------|--------------------------------|
| Partly_resolved | Name was only partly resolved. |
|-----------------|--------------------------------|

Definition at line 77 of file [namespace](#).

15.300.2.2 Query_timeout

```
enum L4Re::Namespace::Query_timeout
```

Timeout values for query operation.

Enumerator

| | |
|-----------------|--------------------------------------|
| To_default | Default timeout. |
| To_non_blocking | Expect callee to answer immediately. |

Definition at line 83 of file [namespace](#).

15.300.2.3 Register_flags

```
enum L4Re::Namespace::Register_flags
```

Flags for registering name spaces.

Enumerator

| | |
|-----------|--|
| Ro | Read-only. |
| Rw | Read-write. |
| Rs | Read-only + strong. |
| Rws | Read-write + strong. |
| Strong | Strong. |
| Trusted | Obsolete, do not use. |
| Link | Obsolete, do not use. |
| Overwrite | If entry already exists, overwrite it. |

Definition at line 57 of file [namespace](#).

15.300.3 Member Function Documentation

15.300.3.1 query() [1/2]

```
l4_ret_t L4Re::Namespace::query (
    char const * name,
    L4::Cap< void > const & cap,
    int timeout = To_default,
    l4_umword_t * local_id = 0,
    bool iterate = true) const [noexcept]
```

Query the name space for a named object.

Parameters

| | | |
|-----|-----------------|--|
| in | <i>name</i> | String to query (without any leading slashes). |
| out | <i>cap</i> | Capability slot where the received capability will be put. |
| in | <i>timeout</i> | Timeout of query in milliseconds. The client will only wait if a name has already been registered with the server but no object has yet been attached. |
| out | <i>local_id</i> | If given, L4_RCV_ITEM_LOCAL_ID will be set for the IPC from the name space, so that if the capability that was received is a local item, the capability ID will be returned with this parameter. |
| in | <i>iterate</i> | If true, the client will try to resolve names by iteratively calling the name spaces until the name is fully resolved. |

Return values

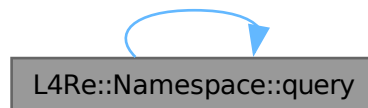
| | |
|-------------------------|---|
| <code>0</code> | Name could be fully resolved. |
| <code>>0</code> | Name could only be partly resolved. The number of remaining characters is returned. |
| <code>-L4_ENOENT</code> | Entry could not be found. |
| <code>-L4_EAGAIN</code> | Entry exists but no object is yet attached. Try again later. |
| <code><0</code> | IPC errors, see l4_error_code_t . |

Definition at line 114 of file [namespace_impl.h](#).

References [query\(\)](#).

Referenced by [query\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.300.3.2 `query()` [2/2]

```

l4_ret_t L4Re::Namespace::query (
    char const * name,
    unsigned len,
    L4::Cap< void > const & cap,
    int timeout = To_default,
    l4_umword_t * local_id = 0,
    bool iterate = true) const [noexcept]
  
```

Query the name space for a named object.

The query string does not necessarily need to be null-terminated.

Parameters

| | | |
|-----|-----------------|--|
| in | <i>len</i> | Length of the string to query without any terminating null characters. |
| in | <i>name</i> | String to query (without any leading slashes). |
| out | <i>cap</i> | Capability slot where the received capability will be put. |
| in | <i>timeout</i> | Timeout of query in milliseconds. The client will only wait if a name has already been registered with the server but no object has yet been attached. |
| out | <i>local_id</i> | If given, L4_RCV_ITEM_LOCAL_ID will be set for the IPC from the name space, so that if the capability that was received is a local item, the capability ID will be returned with this parameter. |
| in | <i>iterate</i> | If true, the client will try to resolve names by iteratively calling the name spaces until the name is fully resolved. |

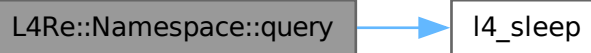
Return values

| | |
|-------------------|---|
| <i>0</i> | Name could be fully resolved. |
| <i>>0</i> | Name could only be partly resolved. The number of remaining characters is returned. |
| <i>-L4_ENOENT</i> | Entry could not be found. |
| <i>-L4_EAGAIN</i> | Entry exists but no object is yet attached. Try again later. |
| <i><0</i> | IPC errors, see l4_error_code_t . |

Definition at line 66 of file [namespace_impl.h](#).

References [L4_EAGAIN](#), [L4_EINVAL](#), [l4_sleep\(\)](#), and [L4_UNLIKELY](#).

Here is the call graph for this function:



15.300.3.3 register_obj()

```

l4_ret_t L4Re::Namespace::register_obj (
    char const * name,
    L4::Ipc::Cap< void > obj,
    unsigned flags = Rw) const [inline], [noexcept]
  
```

Register an object with a name.

Parameters

| | |
|--------------|---|
| <i>name</i> | Name under which the object should be registered. |
| <i>obj</i> | Capability to object to register. An invalid capability may be given to only reserve the name for later use. |
| <i>flags</i> | Flags to assign to the entry, see L4Re::Namespace::Register_flags . Note that the rights that are assigned to a capability are not only determined by the rights given in these flags but also by the rights with which the <code>obj</code> capability was mapped to the name space. |

Return values

| | |
|-------------------|---|
| <i>0</i> | Object was successfully registered with <i>name</i> . |
| <i>-L4_EEXIST</i> | Name already registered. |
| <i>-L4_EPERM</i> | Insufficient permissions; see precondition. |
| <i>-L4_ENOMEM</i> | Server has insufficient resources. |
| <i>-L4_EINVAL</i> | Invalid parameter. |
| <i><0</i> | IPC errors, see l4_error_code_t . |

Precondition

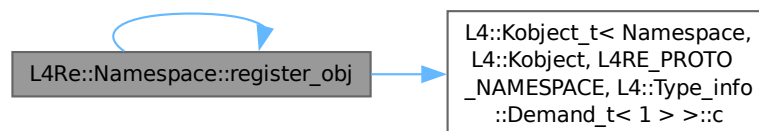
The invoked [Namespace](#) capability must have the permission [L4_CAP_FPAGE_W](#).

Definition at line 167 of file [namespace](#).

References [L4::Kobject_t< Namespace, L4::Kobject, L4RE_PROTO_NAMESPACE, L4::Type_info::Demand_t< 1 > >::c\(\)](#), [register_obj\(\)](#), and [Rw](#).

Referenced by [register_obj\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.300.3.4 unlink()

```
l4_ret_t L4Re::Namespace::unlink (
    char const * name) [inline]
```

Remove an entry from the name space.

Parameters

| | |
|-------------|------------------------------|
| <i>name</i> | Name of the entry to remove. |
|-------------|------------------------------|

Return values

| | |
|-------------|---|
| 0 | Entry successfully removed. |
| -L4_ENOENT | Given name does not exist. |
| -L4_EPERM | Insufficient permissions; see precondition. |
| -L4_EACCESS | Name cannot be removed. |
| <0 | IPC errors, see l4_error_code_t . |

Precondition

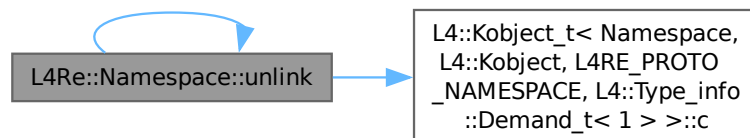
The invoked [Namespace](#) capability must have the permission [L4_CAP_FPAGE_W](#).

Definition at line 193 of file [namespace](#).

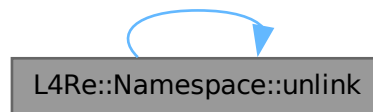
References [L4::Kobject_t< Namespace, L4::Kobject, L4RE_PROTO_NAMESPACE, L4::Type_info::Demand_t< 1 > >::c\(\)](#), and [unlink\(\)](#).

Referenced by [unlink\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following files:

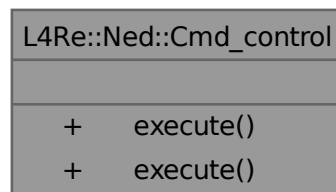
- [l4/re/namespace](#)
- [l4/re/impl/namespace_impl.h](#)

15.301 L4Re::Ned::Cmd_control Class Reference

Direct control interface for Ned.

```
#include <cmd_control>
```

Collaboration diagram for L4Re::Ned::Cmd_control:



Public Member Functions

- long [execute](#) (L4::lpc::String<> cmd) noexcept
Execute the given Lua code.
- long [execute](#) (L4::lpc::String<> cmd, L4::lpc::String< char > *result) noexcept
Execute the given Lua code.

15.301.1 Detailed Description

Direct control interface for Ned.

Definition at line 19 of file [cmd_control](#).

15.301.2 Member Function Documentation

15.301.2.1 execute() [1/2]

```
long L4Re::Ned::Cmd_control::execute (
    L4::Ipc::String<> cmd) [inline], [noexcept]
```

Execute the given Lua code.

Parameters

| | | |
|----|------------|----------------------------------|
| in | <i>cmd</i> | String with Lua code to execute. |
|----|------------|----------------------------------|

Return values

| | |
|-------------------|---------------------------------|
| <i>L4_EOK</i> | Code was successfully executed. |
| <i>-L4_EINVAL</i> | Code could not be parsed. |
| <i>-L4_EIO</i> | Error during code execution. |

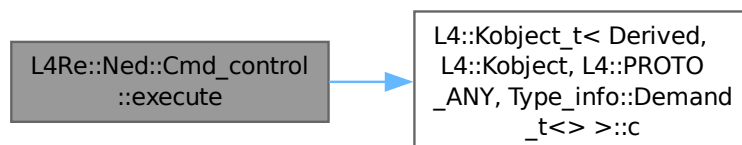
The code is executed using the global Lua state of ned which is retained between successive calls to execute. Thus you may define data in one call to execute and use it in a subsequent call.

This function does not return any results from the execution of the Lua code itself.

Definition at line 42 of file [cmd_control](#).

References [L4::Kobject_t< Derived, L4::Kobject, L4::PROTO_ANY, Type_info::Demand_t<> >::c\(\)](#).

Here is the call graph for this function:



15.301.2.2 execute() [2/2]

```
long L4Re::Ned::Cmd_control::execute (
    L4::Ipc::String<> cmd,
    L4::Ipc::String< char > * result) [inline], [noexcept]
```

Execute the given Lua code.

Parameters

| | | |
|-----|---------------|---|
| in | <i>cmd</i> | String with Lua code to execute. |
| out | <i>result</i> | The first return value of the Lua code block as string. |

Return values

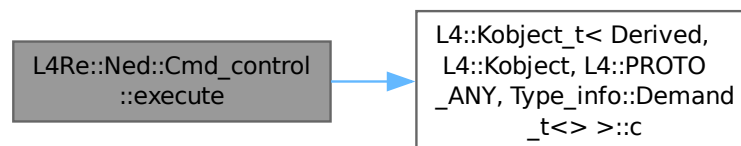
| | |
|-------------------|---------------------------------|
| <i>L4_EOK</i> | Code was successfully executed. |
| <i>-L4_EINVAL</i> | Code could not be parsed. |
| <i>-L4_EIO</i> | Error during code execution. |

The code is executed using the global Lua state of ned which is retained between successive calls to execute. Thus you may define data in one call to execute and use it in a subsequent call.

Definition at line 64 of file [cmd_control](#).

References [L4::Kobject_t< Derived, L4::Kobject, L4::PROTO_ANY, Type_info::Demand_t<> >::c\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

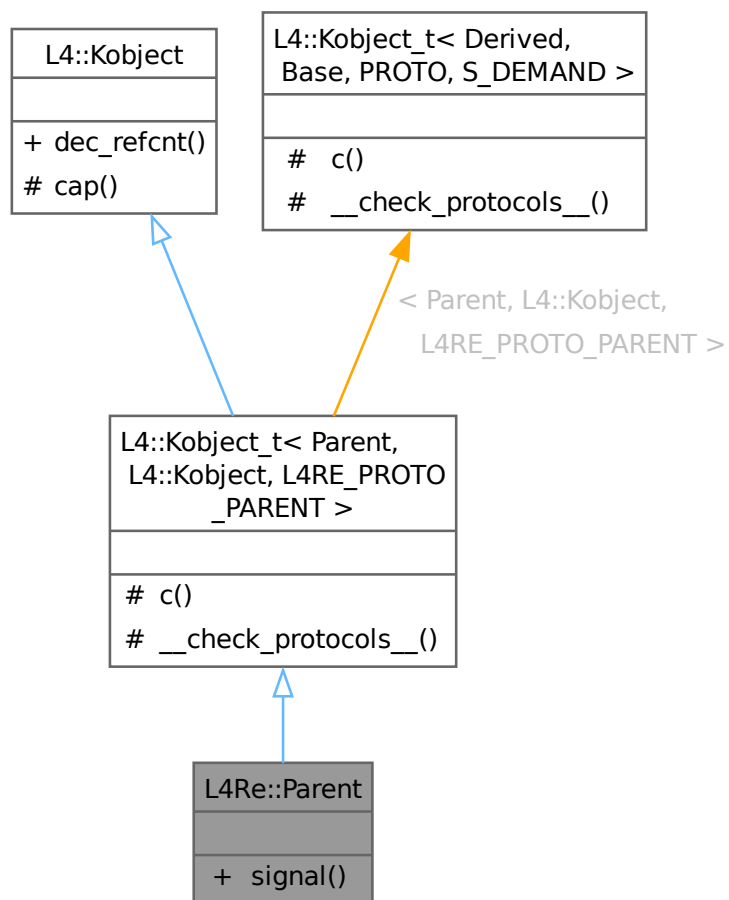
- `pkg/l4re-core/ned/lib/include/cmd_control`

15.302 L4Re::Parent Class Reference

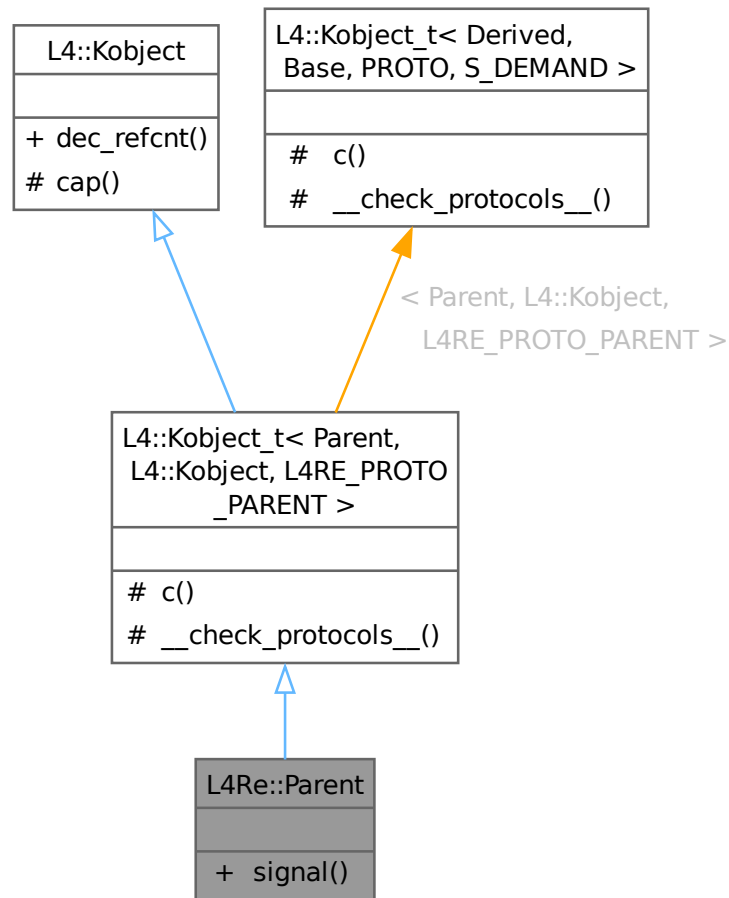
[Parent](#) interface.

```
#include <parent>
```

Inheritance diagram for L4Re::Parent:



Collaboration diagram for L4Re::Parent:



Public Member Functions

- [l4_ret_t signal](#) (unsigned long sig, unsigned long val)
Send a signal to the parent.

Public Member Functions inherited from [L4::Kobject](#)

- [l4_msgtag_t dec_refcnt](#) (l4_mword_t diff, l4_utcb_t *utcb=l4_utcb())
Decrement the in kernel reference counter for the object.

Additional Inherited Members

Protected Types inherited from

[L4::Kobject_t < Parent, L4::Kobject, L4RE_PROTO_PARENT >](#)

- typedef Parent **Class**

The target interface type (inheriting from [Kobject_t](#)).

- typedef Typeid::Iface< PROTO, Parent > **__iface**

The interface description for the derived class.

- typedef Typeid::Merge_list< Typeid::Iface_list< [__iface](#) >, typename L4::Kobject::__iface_list > **__iface_list**

The list of all RPC interfaces provided directly or through inheritance.

Protected Member Functions inherited from [L4::Kobject_t](#)< [Parent](#), [L4::Kobject](#), [L4RE_PROTO_PARENT](#) >

- [L4::Cap](#)< [Class](#) > **c** () const noexcept

Get the capability to ourselves.

Protected Member Functions inherited from [L4::Kobject](#)

- [l4_cap_idx_t](#) **cap** () const noexcept

Return capability selector.

Static Protected Member Functions inherited from [L4::Kobject_t](#)< [Parent](#), [L4::Kobject](#), [L4RE_PROTO_PARENT](#) >

- static void **__check_protocols__** () noexcept

Helper to check for protocol conflicts.

15.302.1 Detailed Description

[Parent](#) interface.

See also

[Parent API](#) for more details about the purpose.

Definition at line 42 of file [parent](#).

15.302.2 Member Function Documentation

15.302.2.1 [signal\(\)](#)

```
l4_ret_t L4Re::Parent::signal (
    unsigned long sig,
    unsigned long val)
```

Send a signal to the parent.

Parameters

| | |
|------------|---------------------|
| <i>sig</i> | Signal to send |
| <i>val</i> | Value of the signal |

Return values

| | |
|----|-----------|
| 0 | Success |
| <0 | IPC error |

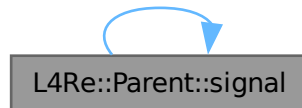
Note

The implementations of this interface in Moe and Ned only recognize the signal 0, in which case they will terminate the application from which the interface was invoked and not return. In this case, `val` is treated as the application's return code. For any other value of `sig`, the method just returns successfully.

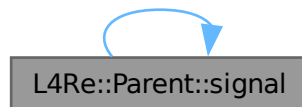
References [signal\(\)](#).

Referenced by [signal\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

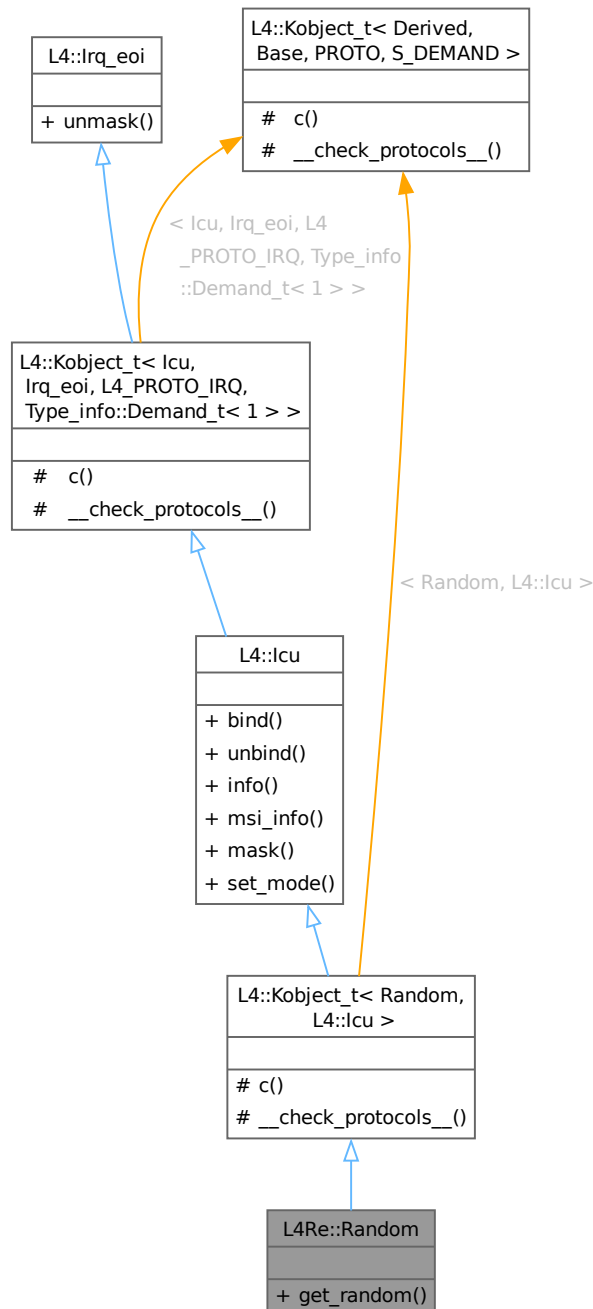
- [l4/re/parent](#)

15.303 L4Re::Random Struct Reference

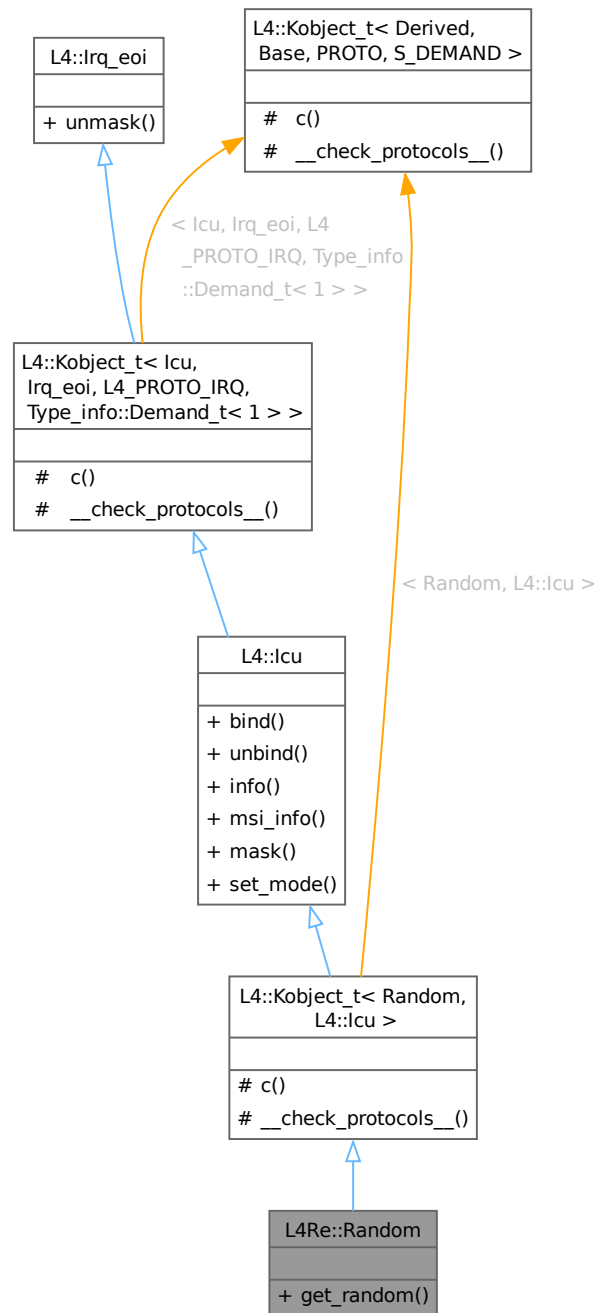
Low-bandwidth interface for random number generators.

```
#include <random>
```

Inheritance diagram for L4Re::Random:



Collaboration diagram for L4Re::Random:



Public Member Functions

- long `get_random` (`l4_size_t` size, `L4::lpc::Array`< char, unsigned long > *buffer)
Get a random number.

Public Member Functions inherited from **L4::lcu**

- `l4_msgtag_t` `bind` (unsigned irqnum, `L4::Cap`< `Triggerable` > irq, `l4_utcb_t` *utcb=`l4_utcb()`) noexcept

- Bind an interrupt line of an interrupt controller to an interrupt object.*

• [l4_msgtag_t unbind](#) (unsigned irqnum, [L4::Cap](#)< [Triggerable](#) > irq, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) noexcept

Remove binding of an interrupt line from the interrupt controller object.
- [l4_msgtag_t info](#) ([l4_icu_info_t](#) *info, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) noexcept

Get information about the ICU features.
- [l4_msgtag_t msi_info](#) ([l4_umword_t](#) irqnum, [l4_uint64_t](#) source, [l4_icu_msi_info_t](#) *msi_info)

Get MSI info about IRQ.
- [l4_msgtag_t mask](#) (unsigned irqnum, [l4_umword_t](#) *label=0, [l4_timeout_t](#) to=[L4_IPC_NEVER](#), [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) noexcept

Mask an IRQ line.
- [l4_msgtag_t set_mode](#) (unsigned irqnum, [l4_umword_t](#) mode, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) noexcept

Set interrupt mode.

Public Member Functions inherited from [L4::lrq_eoi](#)

- [l4_msgtag_t unmask](#) (unsigned irqnum, [l4_umword_t](#) *label=0, [l4_timeout_t](#) to=[L4_IPC_NEVER](#), [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) noexcept

Unmask the given interrupt line.

Additional Inherited Members

Protected Types inherited from [L4::Kobject_t](#)< [Random](#), [L4::lcu](#) >

- typedef [Random](#) **Class**

The target interface type (inheriting from [Kobject_t](#)).
- typedef Typeid::Iface< [PROTO_ANY](#), [Random](#) > **__Iface**

The interface description for the derived class.
- typedef Typeid::Merge_list< Typeid::Iface_list< **__Iface** >, typename [L4::lcu](#)::__Iface_list > **__Iface_list**

The list of all RPC interfaces provided directly or through inheritance.

Protected Types inherited from [L4::Kobject_t](#)< [lcu](#), [lrq_eoi](#), [L4_PROTO_IRQ](#), [Type_info::Demand_t](#)< 1 > >

- typedef [lcu](#) **Class**

The target interface type (inheriting from [Kobject_t](#)).
- typedef Typeid::Iface< [PROTO](#), [lcu](#) > **__Iface**

The interface description for the derived class.
- typedef Typeid::Merge_list< Typeid::Iface_list< **__Iface** >, typename [lrq_eoi](#)::__Iface_list > **__Iface_list**

The list of all RPC interfaces provided directly or through inheritance.

Protected Member Functions inherited from [L4::Kobject_t](#)< [Random](#), [L4::lcu](#) >

- [L4::Cap](#)< [Class](#) > **c** () const noexcept

Get the capability to ourselves.

Protected Member Functions inherited from

[L4::Kobject_t< Icu, Irq_eoi, L4_PROTO_IRQ, Type_info::Demand_t< 1 > >](#)

- [L4::Cap< Class > c \(\)](#) const noexcept

Get the capability to ourselves.

Static Protected Member Functions inherited from [L4::Kobject_t< Random, L4::Icu >](#)

- static void [__check_protocols__ \(\)](#) noexcept

Helper to check for protocol conflicts.

Static Protected Member Functions inherited from

[L4::Kobject_t< Icu, Irq_eoi, L4_PROTO_IRQ, Type_info::Demand_t< 1 > >](#)

- static void [__check_protocols__ \(\)](#) noexcept

Helper to check for protocol conflicts.

15.303.1 Detailed Description

Low-bandwidth interface for random number generators.

The interface offers an ICU interface where a client can register an interrupt to get notified when entropy is available. Support for notifications is optional. If a service does not implement notification, it must return 0 for the number of interrupts in the [info\(\)](#) call. The notification interrupt must have index 0.

Include File

```
#include <l4/re/random>
```

Definition at line 33 of file [random](#).

15.303.2 Member Function Documentation

15.303.2.1 [get_random\(\)](#)

```
long L4Re::Random::get_random (
    l4_size_t size,
    L4::Ipc::Array< char, unsigned long > * buffer)
```

Get a random number.

Parameters

| | | |
|-----|---------------|---|
| | <i>size</i> | Number of bytes of entropy requested. |
| out | <i>buffer</i> | Buffer containing the random number. Each byte in the buffer contains 8 bits of randomness. |

Return values

| | |
|----------------------|--|
| ≥ 0 | Actual size of the returned random number in bytes. This may be less than the requested size. The return value may also be 0 if temporarily no entropy is available. |
| <code>-L4_EIO</code> | Source of randomness permanently unavailable. |
| < 0 | IPC error. |

This function should never block. It should immediately return as much entropy as is available. If the call returns less than the requested bytes and a notification interrupt was installed, then the service triggers an interrupt as soon as the remaining entropy is available. That means that when an interrupt is triggered, the service must guarantee that the next call to [get_random\(\)](#) returns at least the number of missing bytes for the call that initially triggered the notification.

If [get_random\(\)](#) is called while a notification is pending, then the behaviour is implementation-defined.

The documentation for this struct was generated from the following file:

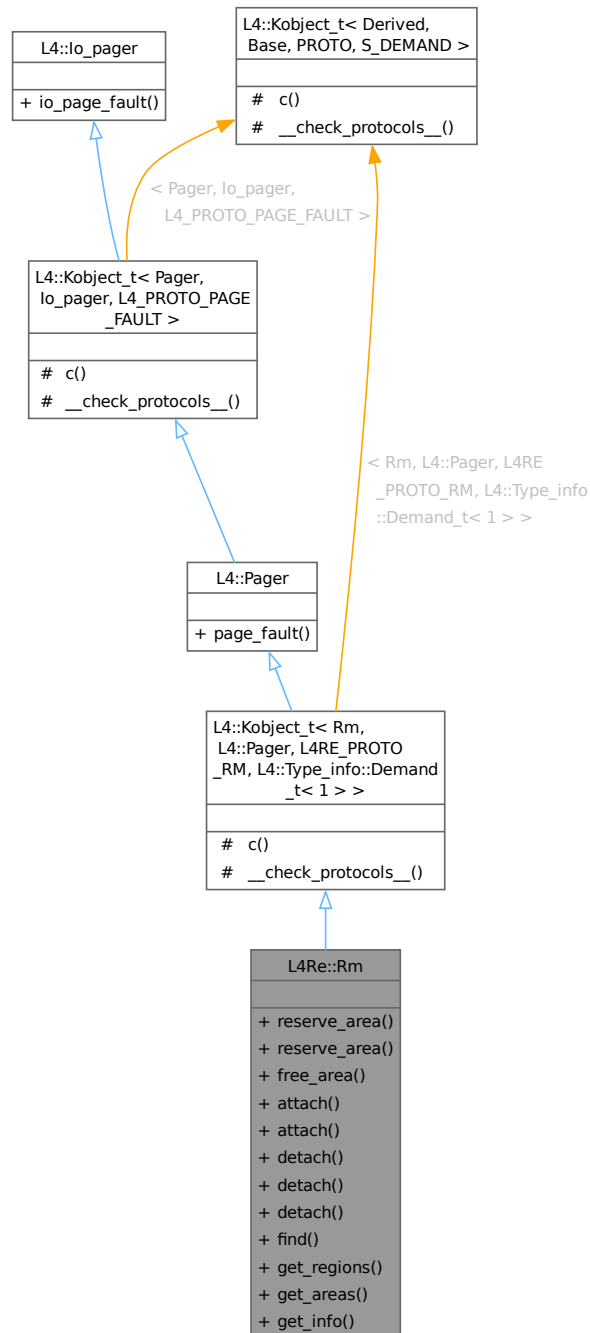
- [l4/re/random](#)

15.304 L4Re::Rm Class Reference

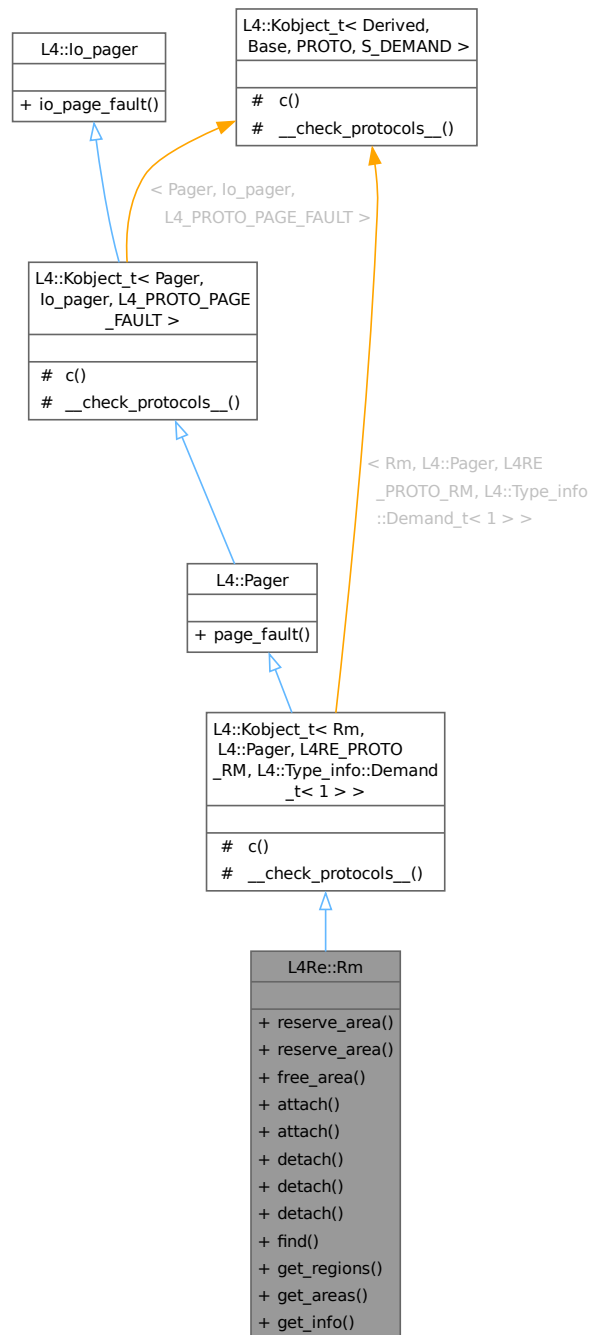
[Region](#) map.

```
#include <l4/re/rm>
```

Inheritance diagram for L4Re::Rm:



Collaboration diagram for L4Re::Rm:



Data Structures

- struct [F](#)
Rm flags definitions.
- class [Unique_region](#)
Unique region.
- struct [Region](#)

A region is a range of virtual addresses which is backed by content.

- struct [Area](#)

An area is a range of virtual addresses which is reserved, see [L4Re::Rm::reserve_area\(\)](#).

Public Types

- enum [Detach_result](#) { [Detached_ds](#) = 0 , [Kept_ds](#) = 1 , [Split_ds](#) = 2 , [Detach_result_mask](#) = 3 , [Detach_again](#) = 4 }
- Result values for detach operation.*
- enum [Region_flag_shifts](#) { [Caching_shift](#) = Dataspace::F::Caching_shift }
- Region flag shifts.*
- enum [Detach_flags](#) { [Detach_exact](#) = 1 , [Detach_overlap](#) = 2 , [Detach_keep](#) = 4 }
- Flags for detach operation.*

Public Member Functions

- [l4_ret_t reserve_area](#) ([l4_addr_t](#) *start, unsigned long size, Flags flags=[Flags](#)(0), unsigned char align=[L4_PAGESHIFT](#)) const noexcept
- Reserve the given area in the region map.*
- template<typename T>
[l4_ret_t reserve_area](#) (T **start, unsigned long size, Flags flags=[Flags](#)(0), unsigned char align=[L4_PAGESHIFT](#)) const noexcept
- Reserve the given area in the region map.*
- [l4_ret_t free_area](#) ([l4_addr_t](#) addr)
- Free an area from the region map.*
- [l4_ret_t attach](#) ([l4_addr_t](#) *start, unsigned long size, Flags flags, [L4::lpc::Cap](#)< Dataspace > mem, Offset offs=0, unsigned char align=[L4_PAGESHIFT](#), [L4::Cap](#)< [L4::Task](#) > const task=[L4::Cap](#)< [L4::Task](#) >::Invalid, char const *name=nullptr, Offset backing_offset=0) const noexcept
- Attach a data space to a region.*
- template<typename T>
[l4_ret_t attach](#) (T **start, unsigned long size, Flags flags, [L4::lpc::Cap](#)< Dataspace > mem, Offset offs=0, unsigned char align=[L4_PAGESHIFT](#), [L4::Cap](#)< [L4::Task](#) > const task=[L4::Cap](#)< [L4::Task](#) >::Invalid, char const *name=nullptr, Offset backing_offset=0) const noexcept
- Attach a data space to a region.*
- [l4_ret_t detach](#) ([l4_addr_t](#) addr, [L4::Cap](#)< Dataspace > *mem, [L4::Cap](#)< [L4::Task](#) > const &task=[This_task](#)) const noexcept
- Detach and unmap a region from the address space.*
- [l4_ret_t detach](#) (void *addr, [L4::Cap](#)< Dataspace > *mem, [L4::Cap](#)< [L4::Task](#) > const &task=[This_task](#)) const noexcept
- Detach and unmap a region from the address space.*
- [l4_ret_t detach](#) ([l4_addr_t](#) start, unsigned long size, [L4::Cap](#)< Dataspace > *mem, [L4::Cap](#)< [L4::Task](#) > const &task) const noexcept
- Detach and unmap all parts of the regions within the specified interval.*
- [l4_ret_t find](#) ([l4_addr_t](#) *addr, unsigned long *size, Offset *offset, [L4Re::Rm::Flags](#) *flags, [L4::Cap](#)< Dataspace > *m) noexcept
- Find a region given an address and size.*
- [l4_ret_t get_regions](#) ([l4_addr_t](#) start, [L4::lpc::Ret_array](#)< [Region](#) > regions)
- Return the list of regions whose starting addresses are higher or equal to *start* in the address space managed by this region map.*
- long [get_areas](#) ([l4_addr_t](#) start, [L4::lpc::Ret_array](#)< [Area](#) > areas)
- Return the list of areas whose starting addresses are higher or equal to *start* in the address space managed by this region map.*
- [l4_ret_t get_info](#) ([l4_addr_t](#) addr, [L4::lpc::String](#)< char > &name, Offset &backing_offset)
- Return auxiliary information of a region.*

Public Member Functions inherited from [L4::Pager](#)

- [l4_msgtag_t page_fault](#) ([l4_umword_t](#) pfa, [l4_umword_t](#) pc, [L4::lpc::Rcv_fpage](#) rwin, [L4::lpc::Opt<L4::lpc::Snd_fpage & > fp](#))

Page-fault protocol message.

Public Member Functions inherited from [L4::lo_pager](#)

- [l4_msgtag_t io_page_fault](#) ([l4_fpage_t](#) io_pfa, [l4_umword_t](#) pc, [L4::lpc::Rcv_fpage](#) rwin, [L4::lpc::Opt<L4::lpc::Snd_fpage & > fp](#))

IO page fault protocol message.

Additional Inherited Members

Protected Types inherited from

[L4::Kobject_t< Rm, L4::Pager, L4RE_PROTO_RM, L4::Type_info::Demand_t< 1 > >](#)

- typedef [Rm](#) **Class**
The target interface type (inheriting from [Kobject_t](#)).
- typedef Typeid::Iface< PROTO, [Rm](#) > **__Iface**
The interface description for the derived class.
- typedef Typeid::Merge_list< Typeid::Iface_list< **__Iface** >, typename [L4::Pager::__Iface_list](#) > **__Iface_list**
The list of all RPC interfaces provided directly or through inheritance.

Protected Types inherited from

[L4::Kobject_t< Pager, lo_pager, L4_PROTO_PAGE_FAULT >](#)

- typedef [Pager](#) **Class**
The target interface type (inheriting from [Kobject_t](#)).
- typedef Typeid::Iface< PROTO, [Pager](#) > **__Iface**
The interface description for the derived class.
- typedef Typeid::Merge_list< Typeid::Iface_list< **__Iface** >, typename [lo_pager::__Iface_list](#) > **__Iface_list**
The list of all RPC interfaces provided directly or through inheritance.

Protected Member Functions inherited from

[L4::Kobject_t< Rm, L4::Pager, L4RE_PROTO_RM, L4::Type_info::Demand_t< 1 > >](#)

- [L4::Cap< Class > c](#) () const noexcept
Get the capability to ourselves.

Protected Member Functions inherited from

[L4::Kobject_t< Pager, lo_pager, L4_PROTO_PAGE_FAULT >](#)

- [L4::Cap< Class > c](#) () const noexcept
Get the capability to ourselves.

Static Protected Member Functions inherited from**[L4::Kobject_t< Rm, L4::Pager, L4RE_PROTO_RM, L4::Type_info::Demand_t< 1 > >](#)**

- static void **`__check_protocols__`** () noexcept
Helper to check for protocol conflicts.

Static Protected Member Functions inherited from**[L4::Kobject_t< Pager, lo_pager, L4_PROTO_PAGE_FAULT >](#)**

- static void **`__check_protocols__`** () noexcept
Helper to check for protocol conflicts.

15.304.1 Detailed Description

[Region](#) map.

See also

[Region map API](#) .

Definition at line [81](#) of file [rm](#).

15.304.2 Member Enumeration Documentation**15.304.2.1 Detach_flags**

enum [L4Re::Rm::Detach_flags](#)

Flags for detach operation.

Enumerator

| | |
|--------------------------------|--|
| Detach_exact | Do an unmap of the exact region given. |
| Detach_overlap | Do an unmap of all overlapping regions. |
| Detach_keep | Do not free the detached data space, ignore the F::Detach_free . |

Definition at line [216](#) of file [rm](#).

15.304.2.2 Detach_result

enum `L4Re::Rm::Detach_result`

Result values for detach operation.

Enumerator

| | |
|--------------|----------------------------------|
| Detached_ds | Detached data space. |
| Kept_ds | Kept data space. |
| Split_ds | Split data space, and done. |
| Detach_again | Detached data space, more to do. |

Definition at line 89 of file `rm`.

15.304.2.3 Region_flag_shifts

enum `L4Re::Rm::Region_flag_shifts`

`Region` flag shifts.

Enumerator

| | |
|---------------|--------------------------------------|
| Caching_shift | Start of <code>Rm</code> cache bits. |
|---------------|--------------------------------------|

Definition at line 100 of file `rm`.

15.304.3 Member Function Documentation

15.304.3.1 attach() [1/2]

```
l4_ret_t L4Re::Rm::attach (
    l4_addr_t * start,
    unsigned long size,
    Rm::Flags flags,
    L4::Ipc::Cap< Dataspace > mem,
    Rm::Offset offs = 0,
    unsigned char align = L4_PAGESHIFT,
    L4::Cap< L4::Task > const task = L4::Cap<L4::Task>::Invalid,
    char const * name = nullptr,
    Rm::Offset backing_offset = 0) const [noexcept]
```

Attach a data space to a region.

Parameters

| | | |
|----------------|-----------------------|---|
| <i>in, out</i> | <i>start</i> | Virtual start address where the region manager shall attach the data space. Will be rounded down to the nearest start of a page. If L4Re::Rm::F::Search_addr is given this value is used as the start address to search for a free virtual memory region and the resulting address is returned here. If L4Re::Rm::F::In_area is given the value is used as a selector for the area (see L4Re::Rm::reserve_area) to attach the data space to. |
| | <i>size</i> | Size of the data space to attach (in bytes). Will be rounded up to the nearest multiple of the page size. |
| | <i>flags</i> | The flags control how and with which rights the dataspace is attached to the region. See L4Re::Rm::F::Attach_flags and L4Re::Rm::F::Region_flags . The caller must specify the desired rights of the attached region explicitly. The default set of rights is empty. If the F : Eager_map flag is set this function may also return L4Re::Dataspace::map error codes if the mapping fails. |
| | <i>mem</i> | Data space. |
| | <i>offs</i> | Offset into the data space to use. |
| | <i>align</i> | Alignment of the virtual region, log2-size, default: a page (L4_PAGESHIFT). This is only meaningful if the L4Re::Rm::F::Search_addr flag is used. |
| | <i>task</i> | Optional destination task of mapping if F : Eager_map flag was passed. If invalid, the mapping is established in the current task. This parameter is only useful if the region manager is for a foreign task. |
| | <i>name</i> | Optional name of the region. |
| | <i>backing_offset</i> | Optional value describing an offset into the backing store of this region. |

Return values

| | |
|-------------------|--|
| 0 | Success |
| -L4_ENOENT | No area could be found (see L4Re::Rm::F::In_area) |
| -L4_EPERM | Operation not allowed. |
| -L4_EINVAL | |
| -L4_EADDRNOTAVAIL | The given address is not available. |
| <0 | IPC errors |

Makes the whole or parts of a data space visible in the virtual memory of the corresponding task. The corresponding region in the virtual address space is backed with the contents of the dataspace.

Note

When searching for a free place in the virtual address space, the space between *start* and the end of the virtual address space is searched.

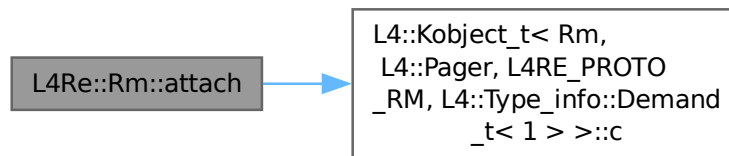
There is no region object created, instead the region is defined by a virtual address within this range (see [L4Re::Rm::find](#)).

Definition at line 35 of file [rm_impl.h](#).

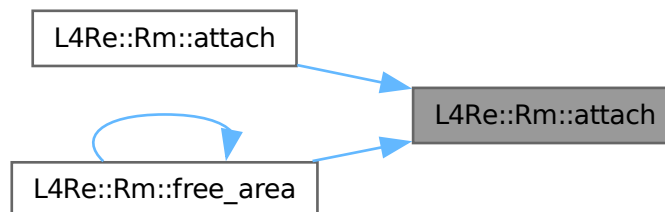
References [L4::Kobject_t< Rm, L4::Pager, L4RE_PROTO_RM, L4::Type_info::Demand_t< 1 > >::c\(\)](#), [L4Re::Rm::F::Eager_map](#), [L4Re::Rm::F::Kernel](#), [L4Re::Rm::F::No_eager_map](#), [L4Re::Rm::F::Reserved](#), and [L4Re::Rm::F::Rights_mask](#).

Referenced by [attach\(\)](#), and [free_area\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.304.3.2 attach() [2/2]

```

template<typename T>
l4_ret_t L4Re::Rm::attach (
    T ** start,
    unsigned long size,
    Flags flags,
    L4::Ipc::Cap< Dataspace > mem,
    Offset offs = 0,
    unsigned char align = L4_PAGESHIFT,
    L4::Cap< L4::Task > const task = L4::Cap<L4::Task>::Invalid,
    char const * name = nullptr,
    Offset backing_offset = 0) const [inline], [noexcept]
  
```

Attach a data space to a region.

Parameters

| | | |
|----------------|-----------------------|---|
| <i>in, out</i> | <i>start</i> | Virtual start address where the region manager shall attach the data space. Will be rounded down to the nearest start of a page. If L4Re::Rm::F::Search_addr is given this value is used as the start address to search for a free virtual memory region and the resulting address is returned here. If L4Re::Rm::F::In_area is given the value is used as a selector for the area (see L4Re::Rm::reserve_area) to attach the data space to. |
| | <i>size</i> | Size of the data space to attach (in bytes). Will be rounded up to the nearest multiple of the page size. |
| | <i>flags</i> | The flags control how and with which rights the dataspace is attached to the region. See L4Re::Rm::F::Attach_flags and L4Re::Rm::F::Region_flags . The caller must specify the desired rights of the attached region explicitly. The default set of rights is empty. If the F::Eager_map flag is set this function may also return L4Re::Dataspace::map error codes if the mapping fails. |
| | <i>mem</i> | Data space. |
| | <i>offs</i> | Offset into the data space to use. |
| | <i>align</i> | Alignment of the virtual region, log2-size, default: a page (L4_PAGESHIFT). This is only meaningful if the L4Re::Rm::F::Search_addr flag is used. |
| | <i>task</i> | Optional destination task of mapping if F::Eager_map flag was passed. If invalid, the mapping is established in the current task. This parameter is only useful if the region manager is for a foreign task. |
| | <i>name</i> | Optional name of the region. |
| | <i>backing_offset</i> | Optional value describing an offset into the backing store of this region. |

Return values

| | |
|-------------------|--|
| 0 | Success |
| -L4_ENOENT | No area could be found (see L4Re::Rm::F::In_area) |
| -L4_EPERM | Operation not allowed. |
| -L4_EINVAL | |
| -L4_EADDRNOTAVAIL | The given address is not available. |
| <0 | IPC errors |

Makes the whole or parts of a data space visible in the virtual memory of the corresponding task. The corresponding region in the virtual address space is backed with the contents of the dataspace.

Note

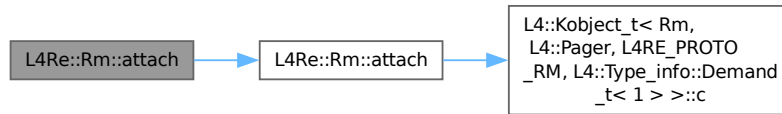
When searching for a free place in the virtual address space, the space between *start* and the end of the virtual address space is searched.

There is no region object created, instead the region is defined by a virtual address within this range (see [L4Re::Rm::find](#)).

Definition at line 409 of file [rm](#).

References [attach\(\)](#), and [L4_PAGESHIFT](#).

Here is the call graph for this function:



15.304.3.3 detach() [1/3]

```

l4_ret_t L4Re::Rm::detach (
    l4_addr_t addr,
    L4::Cap< Dataspace > * mem,
    L4::Cap< L4::Task > const & task = This_task) const [inline], [noexcept]
  
```

Detach and unmap a region from the address space.

Parameters

| | | |
|-----|-------------|--|
| | <i>addr</i> | Virtual address of region, any address within the region is valid. |
| out | <i>mem</i> | Dataspace that is affected. Give 0 if not interested. |
| | <i>task</i> | This argument specifies the task where the pages are unmapped. Provide L4::Cap<L4::Task>::Invalid for none. The default is the current task. |

Return values

| | |
|---|------------------|
| L4Re::Rm::Detach_result | On success. |
| <code>-L4_ENOENT</code> | No region found. |
| <code><0</code> | IPC errors |

Frees a region in the virtual address space given by *addr* (address type). The corresponding part of the address space is now available again.

Definition at line 766 of file [rm](#).

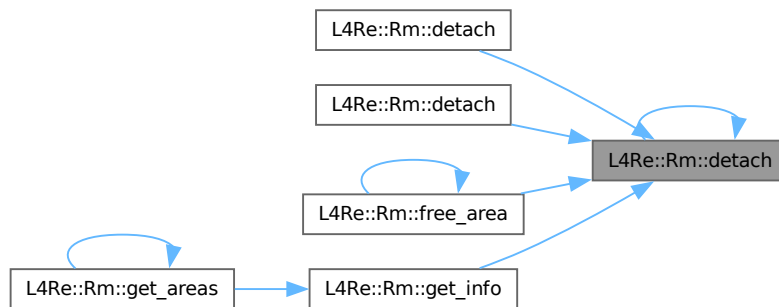
References [detach\(\)](#), and [Detach_overlap](#).

Referenced by [detach\(\)](#), [detach\(\)](#), [detach\(\)](#), [free_area\(\)](#), and [get_info\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.304.3.4 detach() [2/3]

```

l4_ret_t L4Re::Rm::detach (
    l4_addr_t start,
    unsigned long size,
    L4::Cap< Dataspace > * mem,
    L4::Cap< L4::Task > const & task) const [inline], [noexcept]

```

Detach and unmap all parts of the regions within the specified interval.

Parameters

| | | |
|-----|--------------|--|
| | <i>start</i> | Start of area to detach, must be within region. |
| | <i>size</i> | Size of of area to detach (in bytes). |
| out | <i>mem</i> | Dataspace that is affected. Give 0 if not interested. |
| | <i>task</i> | This argument specifies the task where the pages are unmapped. Provide L4::Cap<L4::Task>::Invalid for none. The default is the current task. |

Return values

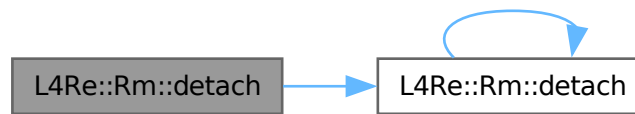
| | |
|---|------------------|
| L4Re::Rm::Detach_result | On success. |
| <code>-L4_ENOENT</code> | No region found. |
| <code><0</code> | IPC errors |

Frees all regions within the interval given by start and size. If a region overlaps the start or the end of the interval this region is only detached partly. If the interval is within one region the original region is split up into two separate regions.

Definition at line 779 of file [rm](#).

References [detach\(\)](#), and [Detach_exact](#).

Here is the call graph for this function:



15.304.3.5 detach() [3/3]

```

l4_ret_t L4Re::Rm::detach (
    void * addr,
    L4::Cap< Dataspace > * mem,
    L4::Cap< L4::Task > const & task = This_task) const [inline], [noexcept]
  
```

Detach and unmap a region from the address space.

Parameters

| | | |
|-----|-------------|--|
| | <i>addr</i> | Virtual address of region, any address within the region is valid. |
| out | <i>mem</i> | Dataspace that is affected. Give 0 if not interested. |
| | <i>task</i> | This argument specifies the task where the pages are unmapped. Provide L4::Cap<L4::Task>::Invalid for none. The default is the current task. |

Return values

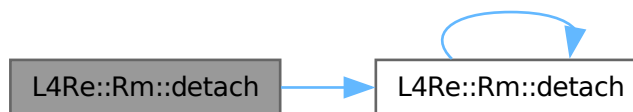
| | |
|---|------------------|
| L4Re::Rm::Detach_result | On success. |
| <code>-L4_ENOENT</code> | No region found. |
| <code><0</code> | IPC errors |

Frees a region in the virtual address space given by *addr* (address type). The corresponding part of the address space is now available again.

Definition at line 771 of file [rm](#).

References [detach\(\)](#), and [Detach_overlap](#).

Here is the call graph for this function:



15.304.3.6 find()

```

l4_ret_t L4Re::Rm::find (
    l4_addr_t * addr,
    unsigned long * size,
    Offset * offset,
    L4Re::Rm::Flags * flags,
    L4::Cap< Dataspace > * m) [inline], [noexcept]
  
```

Find a region given an address and size.

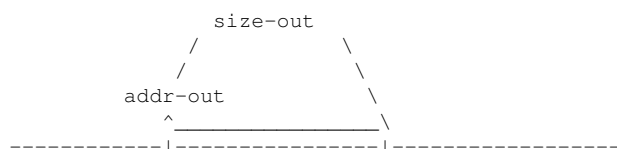
Parameters

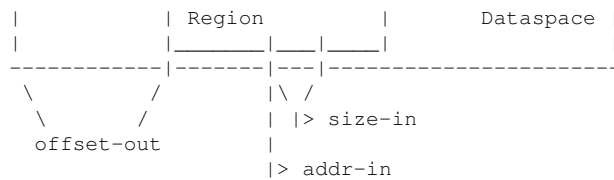
| | | |
|---------|---------------|---|
| in, out | <i>addr</i> | Address to look for. Returns the start address of the found region. |
| in, out | <i>size</i> | Size of the area to look for (in bytes). Returns the size of the found region (in bytes). |
| out | <i>offset</i> | Offset at the beginning of the region within the associated dataspace. |
| out | <i>flags</i> | Region flags, see F::Region_flags (and F::In_area). |
| out | <i>m</i> | Associated dataspace or paging service. |

Return values

| | |
|------------|------------------------|
| 0 | Success |
| -L4_EPERM | Operation not allowed. |
| -L4_ENOENT | No region found. |
| <0 | IPC errors |

This function returns the properties of the region that contains the area described by the *addr* and *size* parameter. If no such region is found but a reserved area, the area is returned and [F::In_area](#) is set in *flags*. Note, in the case of an area the *offset* and *m* return values are invalid.



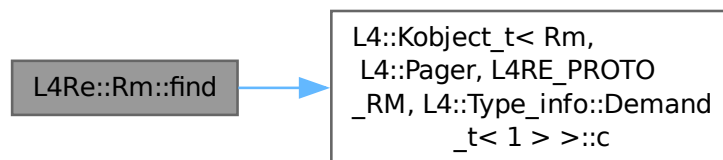
**Note**

The value of the size input parameter should be 1 to assure that a region can be determined unambiguously.

Definition at line 675 of file [rm](#).

References [L4::Kobject_t< Rm, L4::Pager, L4RE_PROTO_RM, L4::Type_info::Demand_t< 1 > >::c\(\)](#).

Here is the call graph for this function:

**15.304.3.7 free_area()**

```
l4_ret_t L4Re::Rm::free_area (
    l4_addr_t addr)
```

Free an area from the region map.

Parameters

| | |
|-------------|-------------------------------------|
| <i>addr</i> | An address within the area to free. |
|-------------|-------------------------------------|

Return values

| | |
|------------|----------------|
| 0 | Success |
| -L4_ENOENT | No area found. |
| <0 | IPC errors |

Note

The data spaces that are attached to that area are not detached by this operation.

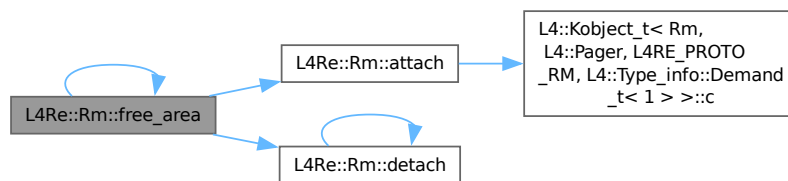
See also

[reserve_area\(\)](#) for more information about areas.

References [attach\(\)](#), [detach\(\)](#), [free_area\(\)](#), [L4::Cap_base::Invalid](#), and [L4_PAGESHIFT](#).

Referenced by [free_area\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**15.304.3.8 get_areas()**

```

long L4Re::Rm::get_areas (
    l4_addr_t start,
    L4::Ipc::Ret_array< Area > areas)
  
```

Return the list of areas whose starting addresses are higher or equal to `start` in the address space managed by this region map.

Parameters

| | | |
|--|--------------|--|
| | <i>start</i> | Virtual address from where to start searching. |
|--|--------------|--|

| | | |
|-----|--------------|---|
| out | <i>areas</i> | List of areas found in this region map. |
|-----|--------------|---|

Return values

| | |
|----------|---|
| ≥ 0 | Number of returned areas in the <i>areas</i> array. |
| < 0 | IPC errors |

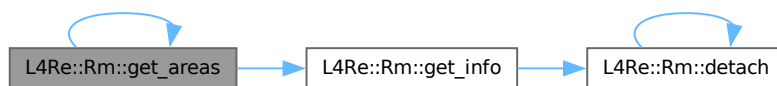
Note

The returned list of areas might not be complete and the caller shall use the function repeatedly with a start address one larger than the end address of the last area from the previous call.

References [get_areas\(\)](#), and [get_info\(\)](#).

Referenced by [get_areas\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**15.304.3.9 get_info()**

```

l4_ret_t L4Re::Rm::get_info (
    l4_addr_t addr,
    L4::Ipc::String< char > & name,
    Offset & backing_offset)
  
```

Return auxiliary information of a region.

This is a debugging feature and might not be available.

Parameters

| | | |
|-----|-----------------------|--------------------------------|
| | <i>addr</i> | Virtual address of the region. |
| out | <i>name</i> | Name of the region. |
| out | <i>backing_offset</i> | Backing offset information. |

Return values

| | |
|-------------------|-----------------------------------|
| <i>0</i> | Success |
| <i>-L4_ENOENT</i> | Region not found. |
| <i>-L4_ENOSYS</i> | Function not available. |
| <i><0</i> | IPC errors |

References [detach\(\)](#).

Referenced by [get_areas\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.304.3.10 get_regions()

```

l4_ret_t L4Re::Rm::get_regions (
    l4_addr_t start,
    L4::Ipc::Ret_array< Region > regions)
  
```


Return the list of regions whose starting addresses are higher or equal to `start` in the address space managed by this region map.

Parameters

| | | |
|------------|----------------|--|
| | <i>start</i> | Virtual address from where to start searching. |
| <i>out</i> | <i>regions</i> | List of regions found in this region map. |

Return values

| | |
|----------|---|
| ≥ 0 | Number of returned regions in the <code>regions</code> array. |
| < 0 | IPC errors |

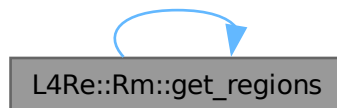
Note

The returned list of regions might not be complete and the caller shall use the function repeatedly with a start address one larger than the end address of the last region from the previous call.

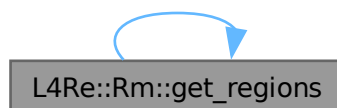
References [get_regions\(\)](#).

Referenced by [get_regions\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.304.3.11 reserve_area() [1/2]

```
l4_ret_t L4Re::Rm::reserve_area (
    l4_addr_t * start,
    unsigned long size,
    Flags flags = Flags(0),
    unsigned char align = L4_PAGESHIFT) const [inline], [noexcept]
```

Reserve the given area in the region map.

Parameters

| | | |
|----------------|--------------|---|
| <i>in, out</i> | <i>start</i> | The virtual start address of the area to reserve. Returns the start address of the area. |
| | <i>size</i> | The size of the area to reserve (in bytes). |
| | <i>flags</i> | Flags for the reserved area (see L4Re::Rm::F::Region_flags and L4Re::Rm::F::Attach_flags). |
| | <i>align</i> | Alignment of area if searched as bits (log2 value). |

Return values

| | |
|-------------------|------------------------------------|
| 0 | Success |
| -L4_EADDRNOTAVAIL | The given area cannot be reserved. |
| <0 | IPC errors |

This function reserves an area within the virtual address space managed by the region map. There are two kinds of areas available:

- Reserved areas (*flags* = [L4Re::Rm::F::Reserved](#)), where no data spaces can be attached
- Special purpose areas (*flags* = 0), where data spaces can be attached to the area via the [L4Re::Rm::F::In_area](#) flag and a start address within the area itself.

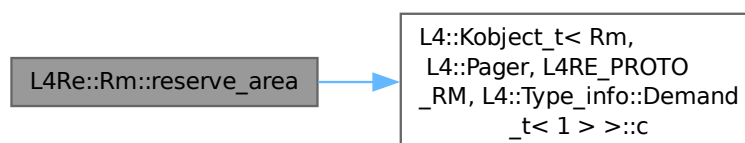
Note

When searching for a free place in the virtual address space (with *flags* = [L4Re::Rm::F::Search_addr](#)), the space between *start* and the end of the virtual address space is searched.

Definition at line 277 of file [rm](#).

References [L4::Kobject_t< Rm, L4::Pager, L4RE_PROTO_RM, L4::Type_info::Demand_t< 1 > >::c\(\)](#), and [L4_PAGESHIFT](#).

Here is the call graph for this function:



15.304.3.12 `reserve_area()` [2/2]

```
template<typename T>
l4_ret_t L4Re::Rm::reserve_area (
    T ** start,
    unsigned long size,
    Flags flags = Flags(0),
    unsigned char align = L4_PAGESHIFT) const [inline], [noexcept]
```

Reserve the given area in the region map.

Parameters

| | | |
|----------------|--------------|---|
| <i>in, out</i> | <i>start</i> | The virtual start address of the area to reserve. Returns the start address of the area. |
| | <i>size</i> | The size of the area to reserve (in bytes). |
| | <i>flags</i> | Flags for the reserved area (see F::Region_flags and F::Attach_flags). |
| | <i>align</i> | Alignment of area if searched as bits (log2 value). |

Return values

| | |
|--------------------------|------------------------------------|
| <i>0</i> | Success |
| <i>-L4_EADDRNOTAVAIL</i> | The given area cannot be reserved. |
| <i><0</i> | IPC errors |

For more information, please refer to the analogous function

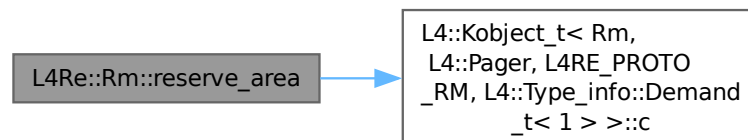
See also

[L4Re::Rm::reserve_area](#).

Definition at line 305 of file [rm](#).

References [L4::Kobject_t< Rm, L4::Pager, L4RE_PROTO_RM, L4::Type_info::Demand_t< 1 > >::c\(\)](#), [L4_PAGESHIFT](#), and

Here is the call graph for this function:



The documentation for this class was generated from the following files:

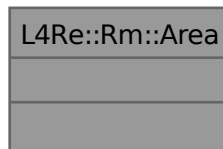
- [l4/re/rm](#)
- [l4/re/impl/rm_impl.h](#)

15.305 L4Re::Rm::Area Struct Reference

An area is a range of virtual addresses which is reserved, see [L4Re::Rm::reserve_area\(\)](#).

```
#include <rm>
```

Collaboration diagram for L4Re::Rm::Area:



15.305.1 Detailed Description

An area is a range of virtual addresses which is reserved, see [L4Re::Rm::reserve_area\(\)](#).

See also

[Region map API](#)

Definition at line [702](#) of file [rm](#).

The documentation for this struct was generated from the following file:

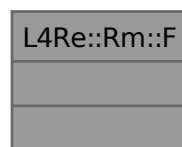
- [l4/re/rm](#)

15.306 L4Re::Rm::F Struct Reference

[Rm](#) flags definitions.

```
#include <rm>
```

Collaboration diagram for L4Re::Rm::F:



Public Types

- enum [Attach_flags](#) : l4_uint32_t {
[Search_addr](#) = 0x20000 , [In_area](#) = 0x40000 , [Eager_map](#) = 0x80000 , [No_eager_map](#) = 0x100000 ,
[Attach_mask](#) = 0x1f0000 }
Flags for attach operation.
- enum [Region_flags](#) : l4_uint16_t {
[Rights_mask](#) = 0x0f , [R](#) = Dataspace::F::R , [W](#) = Dataspace::F::W , [X](#) = Dataspace::F::X ,
[RW](#) = Dataspace::F::RW , [RX](#) = Dataspace::F::RX , [RWX](#) = Dataspace::F::RWX , [Kernel](#) = 0x100 ,
[Detach_free](#) = 0x200 , [Pager](#) = 0x400 , [Reserved](#) = 0x800 , [Caching_mask](#) = Dataspace::F::Caching_mask ,
[Cache_normal](#) = Dataspace::F::Normal , [Cache_buffered](#) = Dataspace::F::Bufferable , [Cache_uncached](#) =
Dataspace::F::Uncacheable , [Ds_map_mask](#) = 0xff ,
[Region_flags_mask](#) = 0xffff }
Region flags (permissions, cacheability, special).

15.306.1 Detailed Description

[Rm](#) flags definitions.

Definition at line [107](#) of file [rm](#).

15.306.2 Member Enumeration Documentation

15.306.2.1 Attach_flags

```
enum L4Re::Rm::F::Attach_flags : l4_uint32_t
```

Flags for attach operation.

Enumerator

| | |
|------------------------------|---|
| Search_addr | Search for a suitable address range. |
| In_area | Search only in area, or map into area. |
| Eager_map | Eagerly map the attached data space in. |
| No_eager_map | Prevent eager mapping of the attached data space. |
| Attach_mask | Mask of all attach flags. |

Definition at line [110](#) of file [rm](#).

15.306.2.2 Region_flags

```
enum L4Re::Rm::F::Region_flags : 14_uint16_t
```

[Region](#) flags (permissions, cacheability, special).

Enumerator

| | |
|-------------------|---|
| Rights_mask | Region rights. |
| R | Readable region. |
| W | Writable region. |
| X | Executable region. |
| RW | Readable and writable region. |
| RX | Readable and executable region. |
| RWX | Readable, writable and executable region. |
| Kernel | Kernel-provided memory (KUMEM). |
| Detach_free | Free the portion of the data space after detach. |
| Pager | Region has a pager. Page faults in this region are not handled by ITAS but rather forwarded to the external region manager / pager. |
| Reserved | Region is reserved (blocked). |
| Caching_mask | Mask of all Rm cache bits. |
| Cache_normal | Cache bits for normal cacheable memory. This is the default if no other cache-related flag was specified. |
| Cache_buffered | Cache bits for buffered (write combining) memory. |
| Cache_uncached | Cache bits for uncached memory. |
| Ds_map_mask | Mask for all bits for cache options and rights. |
| Region_flags_mask | Mask of all region flags. |

Definition at line 128 of file [rm](#).

The documentation for this struct was generated from the following file:

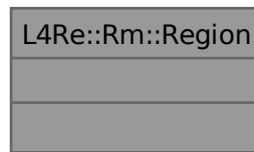
- [l4/re/rm](#)

15.307 L4Re::Rm::Region Struct Reference

A region is a range of virtual addresses which is backed by content.

```
#include <rm>
```

Collaboration diagram for L4Re::Rm::Region:



15.307.1 Detailed Description

A region is a range of virtual addresses which is backed by content.

See also

[Region map API](#)

Definition at line 689 of file [rm](#).

The documentation for this struct was generated from the following file:

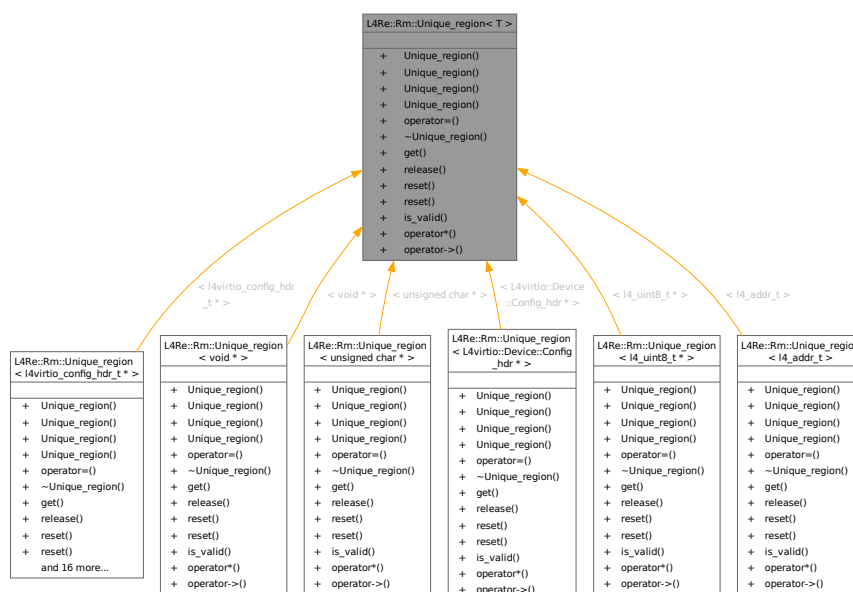
- [l4/re/rm](#)

15.308 L4Re::Rm::Unique_region< T > Class Template Reference

Unique region.

```
#include <rm>
```

Inheritance diagram for L4Re::Rm::Unique_region< T >:



Collaboration diagram for L4Re::Rm::Unique_region< T >:

| L4Re::Rm::Unique_region< T > | |
|------------------------------|------------------|
| | |
| + | Unique_region() |
| + | Unique_region() |
| + | Unique_region() |
| + | Unique_region() |
| + | operator=() |
| + | ~Unique_region() |
| + | get() |
| + | release() |
| + | reset() |
| + | reset() |
| + | is_valid() |
| + | operator*() |
| + | operator->() |

Public Member Functions

- **Unique_region** () noexcept
Construct an invalid [Unique_region](#).
- [Unique_region](#) (T addr) noexcept
Construct a [Unique_region](#) from an address.
- [Unique_region](#) (T addr, [L4::Cap](#)< [Rm](#) > const &rm) noexcept
Construct a valid [Unique_region](#) from an address and a region manager.
- [Unique_region](#) (Unique_region &&o) noexcept
Move-Construct a [Unique_region](#).
- Unique_region & [operator=](#) (Unique_region &&o) noexcept
Move-assign a [Unique_region](#).
- [~Unique_region](#) () noexcept
Destructor.
- T [get](#) () const noexcept
Return the address.
- T [release](#) () noexcept
Return the address and invalidate the [Unique_region](#).
- void [reset](#) (T addr, [L4::Cap](#)< [Rm](#) > const &rm) noexcept
Set new address and region manager.
- void **reset** () noexcept
Make the [Unique_region](#) invalid.

- `bool is_valid ()` const noexcept
Check if the `Unique_region` is valid.
- `T operator* ()` const noexcept
Dereference the address.
- `T operator-> ()` const noexcept
Member access for the address.

15.308.1 Detailed Description

```
template<typename T>
class L4Re::Rm::Unique_region< T >
```

Unique region.

Capture a single region with automatic detach on destruction and unique ownership. Stores the start address and the region-mapper capability internally. A unique region is valid precisely if the internal region-mapper capability is valid. The features for unique ownership and automatic detach are only active for valid unique regions.

Definition at line 435 of file [rm](#).

15.308.2 Constructor & Destructor Documentation

15.308.2.1 Unique_region() [1/3]

```
template<typename T>
L4Re::Rm::Unique_region< T >::Unique_region (
    T addr) [inline], [explicit], [noexcept]
```

Construct a `Unique_region` from an address.

No region manager is set.

Parameters

| | |
|-------------|-----------------|
| <i>addr</i> | The new address |
|-------------|-----------------|

Definition at line 456 of file [rm](#).

15.308.2.2 Unique_region() [2/3]

```
template<typename T>
L4Re::Rm::Unique_region< T >::Unique_region (
    T addr,
    L4::Cap< Rm > const & rm) [inline], [noexcept]
```

Construct a valid [Unique_region](#) from an address and a region manager.

Parameters

| | |
|-------------|--------------------|
| <i>addr</i> | The address |
| <i>rm</i> | The region manager |

Definition at line [465](#) of file [rm](#).

15.308.2.3 Unique_region() [3/3]

```
template<typename T>
L4Re::Rm::Unique_region< T >::Unique_region (
    Unique_region< T > && o) [inline], [noexcept]
```

Move-Construct a [Unique_region](#).

Parameters

| | |
|----------|------------------------------------|
| <i>o</i> | L-value reference to other region. |
|----------|------------------------------------|

Definition at line [473](#) of file [rm](#).

15.308.2.4 ~Unique_region()

```
template<typename T>
L4Re::Rm::Unique_region< T >::~~Unique_region () [inline], [noexcept]
```

Destructor.

If the region is valid, call [detach](#).

Definition at line [498](#) of file [rm](#).

15.308.3 Member Function Documentation

15.308.3.1 get()

```
template<typename T>
T L4Re::Rm::Unique_region< T >::get () const [inline], [noexcept]
```

Return the address.

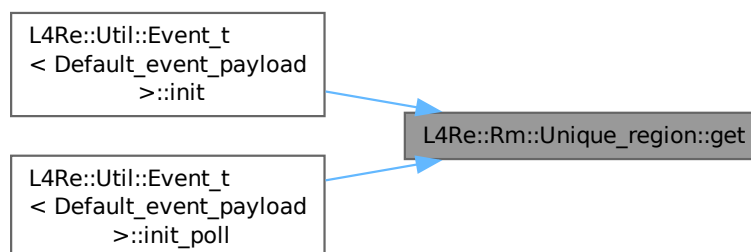
Returns

the address

Definition at line 509 of file [rm](#).

Referenced by [L4Re::Util::Event_t< Default_event_payload >::init\(\)](#), and [L4Re::Util::Event_t< Default_event_payload >::init_poll\(\)](#).

Here is the caller graph for this function:



15.308.3.2 is_valid()

```
template<typename T>
bool L4Re::Rm::Unique_region< T >::is_valid () const [inline], [noexcept]
```

Check if the [Unique_region](#) is valid.

Returns

true iff the [Unique_region](#) is valid

Definition at line 549 of file [rm](#).

15.308.3.3 operator=()

```
template<typename T>
Unique_region & L4Re::Rm::Unique_region< T >::operator= (
    Unique_region< T > && o) [inline], [noexcept]
```

Move-assign a [Unique_region](#).

Parameters

| | |
|----------|--|
| <i>o</i> | L-value reference to region to assign from |
|----------|--|

Definition at line [481](#) of file [rm](#).

15.308.3.4 release()

```
template<typename T>
T L4Re::Rm::Unique_region< T >::release () [inline], [noexcept]
```

Return the address and invalidate the [Unique_region](#).

Returns

the address

Definition at line [517](#) of file [rm](#).

15.308.3.5 reset()

```
template<typename T>
void L4Re::Rm::Unique_region< T >::reset (
    T addr,
    L4::Cap< Rm > const & rm) [inline], [noexcept]
```

Set new address and region manager.

Parameters

| | |
|-------------|------------------------|
| <i>addr</i> | The new address |
| <i>rm</i> | The new region manager |

Definition at line [529](#) of file [rm](#).

The documentation for this class was generated from the following file:

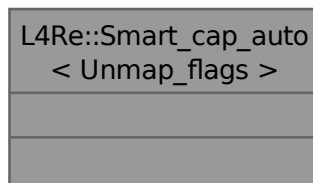
- [l4/re/rm](#)

15.309 L4Re::Smart_cap_auto< Unmap_flags > Class Template Reference

Helper for [Unique_cap](#) and [Unique_del_cap](#).

```
#include <cap_alloc>
```

Collaboration diagram for L4Re::Smart_cap_auto< Unmap_flags >:



15.309.1 Detailed Description

```
template<unsigned long Unmap_flags = L4_FP_ALL_SPACES>
class L4Re::Smart_cap_auto< Unmap_flags >
```

Helper for [Unique_cap](#) and [Unique_del_cap](#).

Definition at line 103 of file [cap_alloc](#).

The documentation for this class was generated from the following file:

- [l4/re/cap_alloc](#)

15.310 L4Re::Smart_count_cap< Unmap_flags > Class Template Reference

Helper for [Ref_cap](#) and [Ref_del_cap](#).

```
#include <cap_alloc>
```

Collaboration diagram for L4Re::Smart_count_cap< Unmap_flags >:

| L4Re::Smart_count_cap < Unmap_flags > | |
|--|--------------|
| | |
| + | free() |
| + | copy() |
| + | invalidate() |

Public Member Functions

- void **free** (L4::Cap_base &c) noexcept
Free operation for L4::Smart_cap (decrement ref count and delete if 0).
- L4::Cap_base **copy** (L4::Cap_base const &src)
Copy operation for L4::Smart_cap (increment ref count).

Static Public Member Functions

- static void **invalidate** (L4::Cap_base &c) noexcept
Invalidate operation for L4::Smart_cap.

15.310.1 Detailed Description

```
template<unsigned long Unmap_flags = L4_FP_ALL_SPACES>
class L4Re::Smart_count_cap< Unmap_flags >
```

Helper for Ref_cap and Ref_del_cap.

Definition at line 132 of file [cap_alloc](#).

The documentation for this class was generated from the following file:

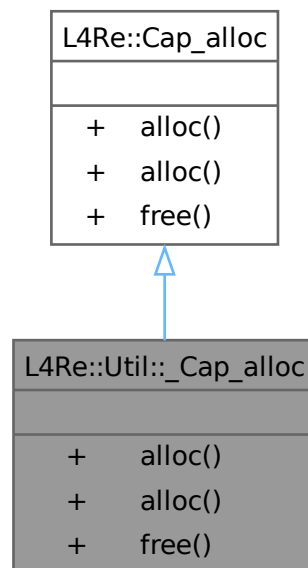
- l4/re/[cap_alloc](#)

15.311 L4Re::Util::_Cap_alloc Class Reference

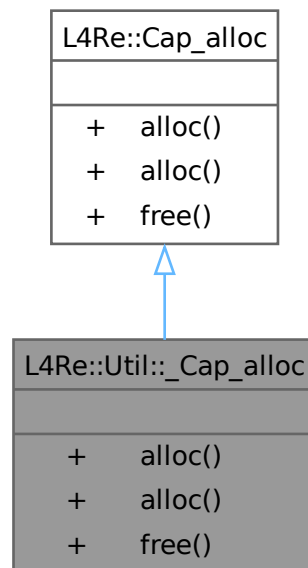
Adapter to expose the cap allocator implementation as [L4Re::Cap_alloc](#) compatible class.

```
#include <cap_alloc_impl.h>
```

Inheritance diagram for L4Re::Util::_Cap_alloc:



Collaboration diagram for L4Re::Util::_Cap_alloc:



Public Member Functions

- `L4::Cap< void > alloc ()` noexcept override
Allocate a capability.
- `template<typename T>`
`L4::Cap< T > alloc ()` noexcept
Allocate a capability.
- `void free (L4::Cap< void > cap, l4_cap_idx_t task=L4_INVALID_CAP, unsigned unmap_flags=L4_FP_ALL_SPACES)` noexcept override
Free a capability.

Public Member Functions inherited from L4Re::Cap_alloc

- `template<typename T>`
`L4::Cap< T > alloc ()` noexcept
Allocate a capability.

15.311.1 Detailed Description

Adapter to expose the cap allocator implementation as `L4Re::Cap_alloc` compatible class.

Not intended to be used in application code.

Definition at line 66 of file `cap_alloc_impl.h`.

15.311.2 Member Function Documentation

15.311.2.1 alloc() [1/2]

```
template<typename T>
L4::Cap< T > L4Re::Util::_Cap_alloc::alloc () [inline], [virtual], [noexcept]
```

Allocate a capability.

Returns

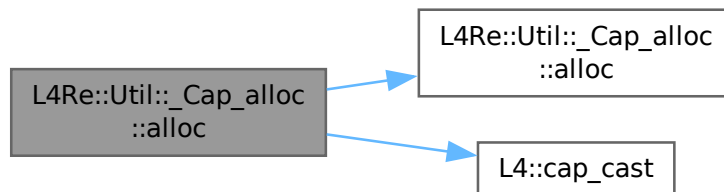
Capability of type void

Implements [L4Re::Cap_alloc](#).

Definition at line 79 of file [cap_alloc_impl.h](#).

References [alloc\(\)](#), and [L4::cap_cast\(\)](#).

Here is the call graph for this function:



15.311.2.2 alloc() [2/2]

```
L4::Cap< void > L4Re::Util::_Cap_alloc::alloc () [inline], [override], [virtual], [noexcept]
```

Allocate a capability.

Returns

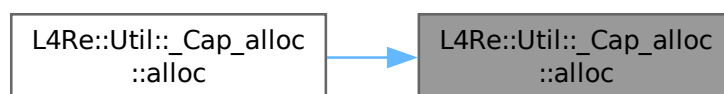
Capability of type void

Implements [L4Re::Cap_alloc](#).

Definition at line 75 of file [cap_alloc_impl.h](#).

Referenced by [alloc\(\)](#).

Here is the caller graph for this function:



15.311.2.3 free()

```
void L4Re::Util::_Cap_alloc::free (
    L4::Cap< void > cap,
    l4_cap_idx_t task = L4_INVALID_CAP,
    unsigned unmap_flags = L4_FP_ALL_SPACES) [inline], [override], [virtual], [noexcept]
```

Free a capability.

Parameters

| | |
|--------------------|---|
| <i>cap</i> | Capability to free. |
| <i>task</i> | If set, task to unmap the capability from. |
| <i>unmap_flags</i> | Flags for unmap, see l4_unmap_flags_t . |

Implements [L4Re::Cap_alloc](#).

Definition at line 85 of file [cap_alloc_impl.h](#).

References [L4Re::Util::L4_FP_ALL_SPACES](#), and [L4_INVALID_CAP](#).

The documentation for this class was generated from the following file:

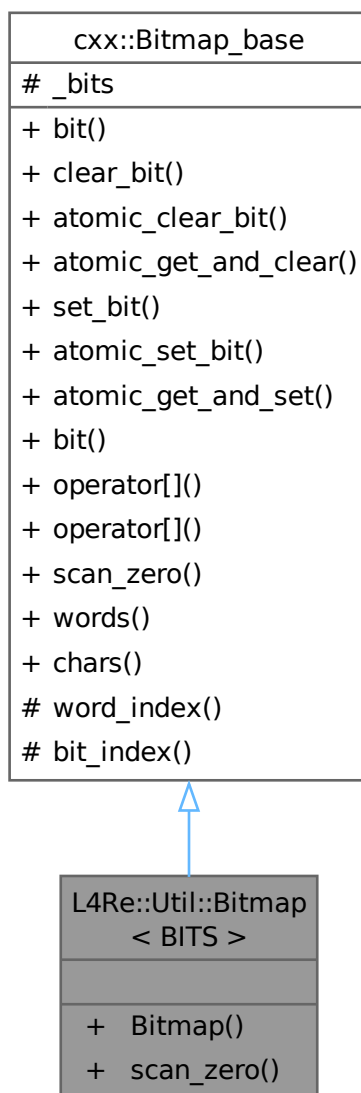
- [l4/re/util/cap_alloc_impl.h](#)

15.312 L4Re::Util::Bitmap< BITS > Class Template Reference

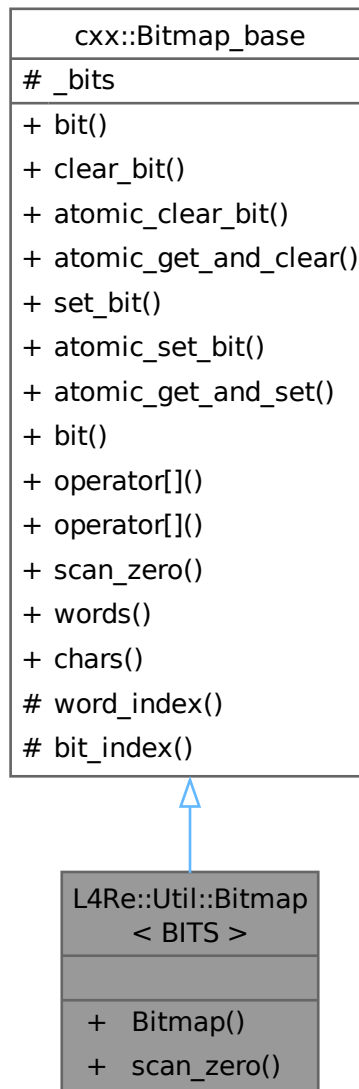
A static bitmap.

```
#include <bitmap>
```

Inheritance diagram for L4Re::Util::Bitmap< BITS >:



Collaboration diagram for L4Re::Util::Bitmap< BITS >:



Public Member Functions

- **Bitmap** () noexcept
Create a bitmap with `BITS` bits.
- long [scan_zero](#) (long start_bit=0) const noexcept
Scan for the first zero bit.

Public Member Functions inherited from [cxx::Bitmap_base](#)

- void [bit](#) (long bit, bool on) noexcept

- Set the value of bit *bit* to on.

 - void `clear_bit` (long *bit*) noexcept

Clear bit *bit*.
- void `atomic_clear_bit` (long *bit*) noexcept

Clear bit *bit* atomically.
- `word_type atomic_get_and_clear` (long *bit*) noexcept

Clear bit *bit* atomically and return old state.
- void `set_bit` (long *bit*) noexcept

Set bit *bit*.
- void `atomic_set_bit` (long *bit*) noexcept

Set bit *bit* atomically.
- `word_type atomic_get_and_set` (long *bit*) noexcept

Set bit *bit* atomically and return old state.
- `word_type bit` (long *bit*) const noexcept

Get the truth value of a bit.
- `word_type operator[]` (long *bit*) const noexcept

Get the bit at index *bit*.
- `Bit operator[]` (long *bit*) noexcept

Get the lvalue for the bit at index *bit*.
- long `scan_zero` (long max_bit, long start_bit=0) const noexcept

Scan for the first zero bit.

Additional Inherited Members

Static Public Member Functions inherited from `cxx::Bitmap_base`

- static long `words` (long bits) noexcept

Get the number of *words* that are used for the bitmap.
- static long `chars` (long bits) noexcept

Get the number of *chars* that are used for the bitmap.

Protected Types inherited from `cxx::Bitmap_base`

- enum { `W_bits` = sizeof(word_type) * 8 , `C_bits` = 8 }
 - typedef unsigned long `word_type`
- Data type for each element of the bit buffer.

Static Protected Member Functions inherited from `cxx::Bitmap_base`

- static unsigned `word_index` (unsigned *bit*)

Get the word index for the given bit.
- static unsigned `bit_index` (unsigned *bit*)

Get the bit index within *word_type* for the given bit.

Protected Attributes inherited from `cxx::Bitmap_base`

- `word_type * _bits`
- Pointer to the buffer storing the bits.

15.312.1 Detailed Description

```
template<int BITS>
class L4Re::Util::Bitmap< BITS >
```

A static bitmap.

Template Parameters

| | |
|-------------|---|
| <i>BITS</i> | The number of bits that shall be in the bitmap. |
|-------------|---|

Definition at line 220 of file [bitmap](#).

15.312.2 Member Function Documentation

15.312.2.1 scan_zero()

```
template<int BITS>
long cxx::Bitmap< BITS >::scan_zero (
    long start_bit = 0) const    [inline], [noexcept]
```

Scan for the first zero bit.

Parameters

| | |
|------------------|--|
| <i>start_bit</i> | Hint at the number of the first bit to look at. Zero bits below <i>start_bit</i> may or may not be taken into account by the implementation. |
|------------------|--|

Return values

| | |
|--------------|---|
| <i>>=</i> | 0 Number of first zero bit found. |
| <i>-1</i> | All bits at <i>start_bit</i> or higher are set. |

Compared to [Bitmap_base::scan_zero\(\)](#), the upper bound is set to BITS.

Definition at line 365 of file [bitmap](#).

The documentation for this class was generated from the following file:

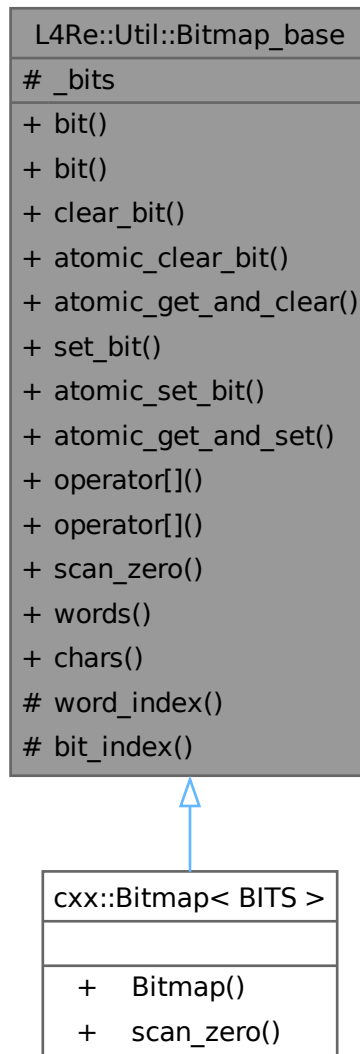
- I4/cxx/bitmap

15.313 L4Re::Util::Bitmap_base Class Reference

Basic bitmap abstraction.

```
#include <bitmap>
```

Inheritance diagram for L4Re::Util::Bitmap_base:



Collaboration diagram for L4Re::Util::Bitmap_base:

| L4Re::Util::Bitmap_base |
|--------------------------|
| # _bits |
| + bit() |
| + bit() |
| + clear_bit() |
| + atomic_clear_bit() |
| + atomic_get_and_clear() |
| + set_bit() |
| + atomic_set_bit() |
| + atomic_get_and_set() |
| + operator[]() |
| + operator[]() |
| + scan_zero() |
| + words() |
| + chars() |
| # word_index() |
| # bit_index() |

Data Structures

- class [Bit](#)
A writable bit in a bitmap.
- class [Word](#)
Helper abstraction for a word contained in the bitmap.
- class [Char](#)
Helper abstraction for a byte contained in the bitmap.

Public Member Functions

- void [bit](#) (long bit, bool on) noexcept
Set the value of bit [bit](#) to on.
- [word_type](#) [bit](#) (long bit) const noexcept
Get the truth value of a bit.
- void [clear_bit](#) (long [bit](#)) noexcept
Clear bit [bit](#).
- void [atomic_clear_bit](#) (long [bit](#)) noexcept
Clear bit [bit](#) atomically.

- [word_type atomic_get_and_clear](#) (long [bit](#)) noexcept
Clear bit [bit](#) atomically and return old state.
- void [set_bit](#) (long [bit](#)) noexcept
Set bit [bit](#).
- void [atomic_set_bit](#) (long [bit](#)) noexcept
Set bit [bit](#) atomically.
- [word_type atomic_get_and_set](#) (long [bit](#)) noexcept
Set bit [bit](#) atomically and return old state.
- [word_type operator\[\]](#) (long [bit](#)) const noexcept
Get the bit at index [bit](#).
- [Bit operator\[\]](#) (long [bit](#)) noexcept
Get the lvalue for the bit at index [bit](#).
- long [scan_zero](#) (long max_bit, long start_bit=0) const noexcept
Scan for the first zero bit.

Static Public Member Functions

- static long **words** (long bits) noexcept
Get the number of words that are used for the bitmap.
- static long **chars** (long bits) noexcept
Get the number of chars that are used for the bitmap.

Protected Types

- enum { [W_bits](#) = sizeof(word_type) * 8 , [C_bits](#) = 8 }
- typedef unsigned long **word_type**
Data type for each element of the bit buffer.

Static Protected Member Functions

- static unsigned [word_index](#) (unsigned [bit](#))
Get the word index for the given bit.
- static unsigned [bit_index](#) (unsigned [bit](#))
Get the bit index within [word_type](#) for the given bit.

Protected Attributes

- [word_type](#) * **_bits**
Pointer to the buffer storing the bits.

15.313.1 Detailed Description

Basic bitmap abstraction.

This abstraction keeps a pointer to a memory area that is used as bitmap.

Definition at line 18 of file [bitmap](#).

15.313.2 Member Enumeration Documentation

15.313.2.1 anonymous enum

anonymous enum [protected]

Enumerator

| | |
|--------|---|
| W_bits | number of bits in word_type |
| C_bits | number of bits in char |

Definition at line [26](#) of file [bitmap](#).

15.313.3 Member Function Documentation

15.313.3.1 atomic_clear_bit()

```
void cxx::Bitmap\_base::atomic\_clear\_bit (  
    long bit) [inline], [noexcept]
```

Clear bit [bit](#) atomically.

Use this function for multi-threaded access to the bitmap.

Parameters

| | |
|------------|---------------------------------|
| <i>bit</i> | The number of the bit to clear. |
|------------|---------------------------------|

Definition at line [269](#) of file [bitmap](#).

15.313.3.2 atomic_get_and_clear()

```
Bitmap\_base::word\_type cxx::Bitmap\_base::atomic\_get\_and\_clear (  
    long bit) [inline], [noexcept]
```

Clear bit [bit](#) atomically and return old state.

Use this function for multi-threaded access to the bitmap.

Parameters

| | |
|------------|---------------------------------|
| <i>bit</i> | The number of the bit to clear. |
|------------|---------------------------------|

Definition at line [279](#) of file [bitmap](#).

15.313.3.3 atomic_get_and_set()

```
Bitmap_base::word_type cxx::Bitmap_base::atomic_get_and_set (
    long bit) [inline], [noexcept]
```

Set bit `bit` atomically and return old state.

Use this function for multi-threaded access to the bitmap.

Parameters

| | |
|------------|-------------------------------|
| <i>bit</i> | The number of the bit to set. |
|------------|-------------------------------|

Definition at line 308 of file `bitmap`.

15.313.3.4 atomic_set_bit()

```
void cxx::Bitmap_base::atomic_set_bit (
    long bit) [inline], [noexcept]
```

Set bit `bit` atomically.

Use this function for multi-threaded access to the bitmap.

Parameters

| | |
|------------|-------------------------------|
| <i>bit</i> | The number of the bit to set. |
|------------|-------------------------------|

Definition at line 298 of file `bitmap`.

15.313.3.5 bit() [1/2]

```
Bitmap_base::word_type cxx::Bitmap_base::bit (
    long bit) const [inline], [noexcept]
```

Get the truth value of a bit.

Parameters

| | |
|------------|--------------------------------|
| <i>bit</i> | The number of the bit to read. |
|------------|--------------------------------|

Return values

| | |
|------|-----------------|
| 0 | Bit is not set. |
| != 0 | Bit is set. |

Definition at line 318 of file `bitmap`.

15.313.3.6 `bit()` [2/2]

```
void cxx::Bitmap_base::bit (
    long bit,
    bool on) [inline], [noexcept]
```

Set the value of bit `bit` to on.

Parameters

| | |
|------------|--|
| <i>bit</i> | The number of the bit. |
| <i>on</i> | The boolean value that shall be assigned to the bit. |

Definition at line 251 of file `bitmap`.

15.313.3.7 `bit_index()`

```
unsigned cxx::Bitmap_base::bit_index (
    unsigned bit) [inline], [static], [protected]
```

Get the bit index within `word_type` for the given bit.

Parameters

| | |
|------------|------------------------------|
| <i>bit</i> | The bit index in the bitmap. |
|------------|------------------------------|

Returns

the bit index within `word_type` (bit % W_bits).

Definition at line 53 of file `bitmap`.

15.313.3.8 `clear_bit()`

```
void cxx::Bitmap_base::clear_bit (
    long bit) [inline], [noexcept]
```

Clear bit `bit`.

Parameters

| | |
|------------|---------------------------------|
| <i>bit</i> | The number of the bit to clear. |
|------------|---------------------------------|

Definition at line 260 of file `bitmap`.

15.313.3.9 operator[]() [1/2]

```
word_type cxx::Bitmap_base::operator[] (
    long bit) const [inline], [noexcept]
```

Get the bit at index `bit`.

Parameters

| | |
|------------|--------------------------------|
| <i>bit</i> | The number of the bit to read. |
|------------|--------------------------------|

Return values

| | |
|------|-----------------|
| 0 | Bit is not set. |
| != 0 | Bit is set. |

Definition at line 181 of file `bitmap`.

15.313.3.10 operator[]() [2/2]

```
Bit cxx::Bitmap_base::operator[] (
    long bit) [inline], [noexcept]
```

Get the lvalue for the bit at index `bit`.

Parameters

| | |
|------------|-------------|
| <i>bit</i> | The number. |
|------------|-------------|

Returns

lvalue for `bit`

Definition at line 191 of file `bitmap`.

15.313.3.11 scan_zero()

```
long cxx::Bitmap_base::scan_zero (
    long max_bit,
    long start_bit = 0) const [inline], [noexcept]
```

Scan for the first zero bit.

Parameters

| | |
|----------------|---|
| <i>max_bit</i> | Upper bound (exclusive) for the scanning operation. |
|----------------|---|

| | |
|------------------|--|
| <i>start_bit</i> | Hint at the number of the first bit to look at. Zero bits below <i>start_bit</i> may or may not be taken into account by the implementation. |
|------------------|--|

Return values

| | |
|--------------|---|
| <i>>=</i> | 0 Number of first zero bit found. |
| <i>-1</i> | All bits between <i>start_bit</i> and <i>max_bit</i> are set. |

Definition at line 339 of file [bitmap](#).

15.313.3.12 `set_bit()`

```
void cxx::Bitmap_base::set_bit (
    long bit) [inline], [noexcept]
```

Set bit *bit*.

Parameters

| | |
|------------|-------------------------------|
| <i>bit</i> | The number of the bit to set. |
|------------|-------------------------------|

Definition at line 289 of file [bitmap](#).

15.313.3.13 `word_index()`

```
unsigned cxx::Bitmap_base::word_index (
    unsigned bit) [inline], [static], [protected]
```

Get the word index for the given bit.

Parameters

| | |
|------------|-----------------------------------|
| <i>bit</i> | The index of the bit in question. |
|------------|-----------------------------------|

Returns

the index in [Bitmap_base::_bits](#) for the given bit (bit / W_bits).

Definition at line 44 of file [bitmap](#).

The documentation for this class was generated from the following file:

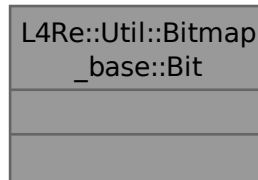
- I4/cxx/bitmap

15.314 L4Re::Util::Bitmap_base::Bit Class Reference

A writable bit in a bitmap.

```
#include <bitmap>
```

Collaboration diagram for L4Re::Util::Bitmap_base::Bit:



15.314.1 Detailed Description

A writable bit in a bitmap.

Definition at line 58 of file [bitmap](#).

The documentation for this class was generated from the following file:

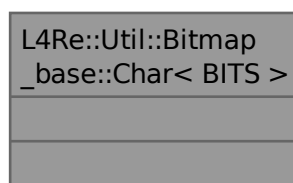
- I4/cxx/bitmap

15.315 L4Re::Util::Bitmap_base::Char< BITS > Class Template Reference

Helper abstraction for a byte contained in the bitmap.

```
#include <bitmap>
```

Collaboration diagram for L4Re::Util::Bitmap_base::Char< BITS >:



15.315.1 Detailed Description

```
template<long BITS>
```

```
class L4Re::Util::Bitmap_base::Char< BITS >
```

Helper abstraction for a byte contained in the bitmap.

Definition at line 95 of file [bitmap](#).

The documentation for this class was generated from the following file:

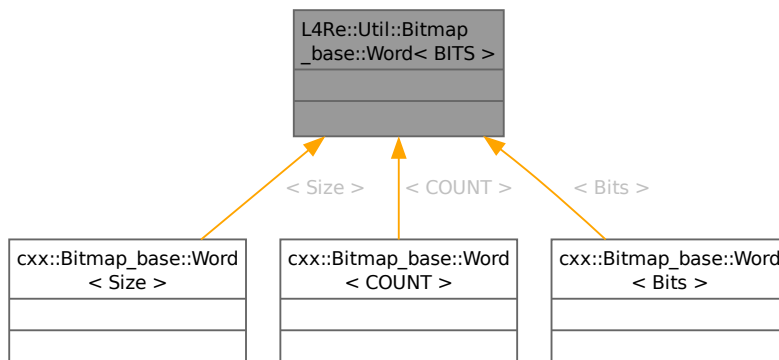
- I4/cxx/bitmap

15.316 L4Re::Util::Bitmap_base::Word< BITS > Class Template Reference

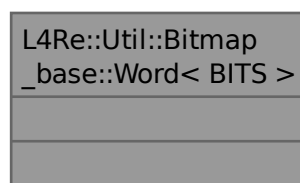
Helper abstraction for a word contained in the bitmap.

```
#include <bitmap>
```

Inheritance diagram for L4Re::Util::Bitmap_base::Word< BITS >:



Collaboration diagram for L4Re::Util::Bitmap_base::Word< BITS >:



15.316.1 Detailed Description

```
template<long BITS>  
class L4Re::Util::Bitmap_base::Word< BITS >
```

Helper abstraction for a word contained in the bitmap.

Definition at line 79 of file [bitmap](#).

The documentation for this class was generated from the following file:

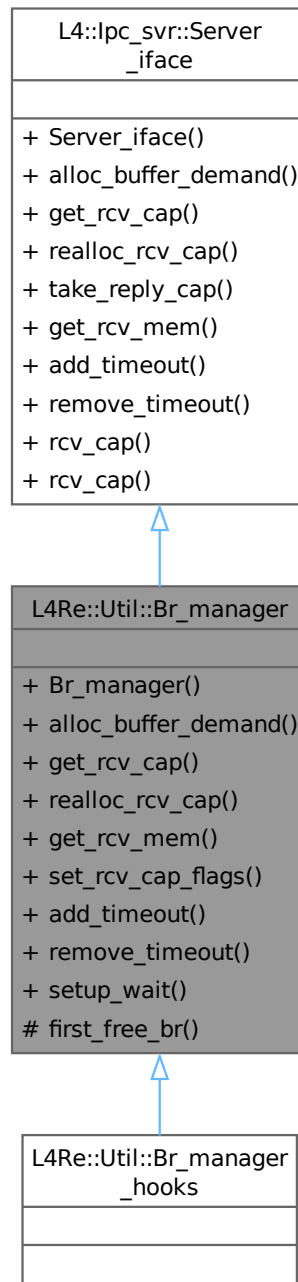
- l4/cxx/bitmap

15.317 L4Re::Util::Br_manager Class Reference

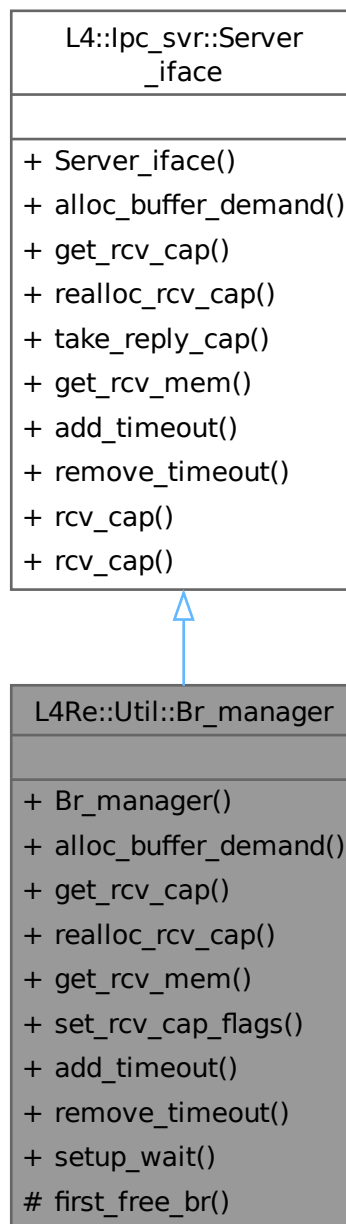
Buffer-register (BR) manager for [L4::Server](#).

```
#include <br_manager>
```

Inheritance diagram for L4Re::Util::Br_manager:



Collaboration diagram for L4Re::Util::Br_manager:



Public Member Functions

- **Br_manager ()**
Make a buffer-register (BR) manager.
- `int alloc_buffer_demand (Demand const &d)` override
Tells the server to allocate buffers for the given demand.
- `L4::Cap< void > get_rcv_cap (int i)` const override

- *Get capability slot allocated to the given receive buffer.*
- `int realloc_rcv_cap (int i)` override
Allocate a new capability for the given receive buffer.
- `cxx::Result< L4::lpc_svr::Server_iface::Mem_window > get_rcv_mem ()` noexcept override
Take the current memory receive window.
- `void set_rcv_cap_flags (unsigned long flags)`
Set the receive flags for the buffers.
- `int add_timeout (L4::lpc_svr::Timeout *, l4_kernel_clock_t)` override
No timeouts handled by us.
- `int remove_timeout (L4::lpc_svr::Timeout *)` override
No timeouts handled by us.
- `void setup_wait (l4_utcb_t *utcb, L4::lpc_svr::Reply_mode)`
setup_wait() used the server loop (L4::Server)

Public Member Functions inherited from `L4::lpc_svr::Server_iface`

- `Server_iface ()`
Make a server interface.
- `virtual cxx::Result< L4::Reply_cap > take_reply_cap ()` noexcept
Take the currently used reply capability.
- `template<typename T> L4::Cap< T > rcv_cap (int index) const`
Get given receive buffer as typed capability.
- `L4::Cap< void > rcv_cap (int index) const`
Get receive cap with the given index as generic (void) type.

Protected Member Functions

- `unsigned first_free_br () const`
Used for assigning BRs for a timeout.

Additional Inherited Members

Public Types inherited from `L4::lpc_svr::Server_iface`

- `using Demand = L4::Type_info::Demand`
Data type expressing server-side demand for receive buffers.

15.317.1 Detailed Description

Buffer-register (BR) manager for `L4::Server`.

Implementation of the `L4::lpc_svr::Server_iface` API for managing the server-side receive buffers needed for a set of server objects running within a server.

Definition at line 27 of file `br_manager`.

15.317.2 Member Function Documentation

15.317.2.1 alloc_buffer_demand()

```
int L4Re::Util::Br_manager::alloc_buffer_demand (
    Demand const & demand) [inline], [override], [virtual]
```

Tells the server to allocate buffers for the given demand.

Parameters

| | |
|---------------|--|
| <i>demand</i> | The total server-side demand of receive buffers needed for a given interface, see Demand . |
|---------------|--|

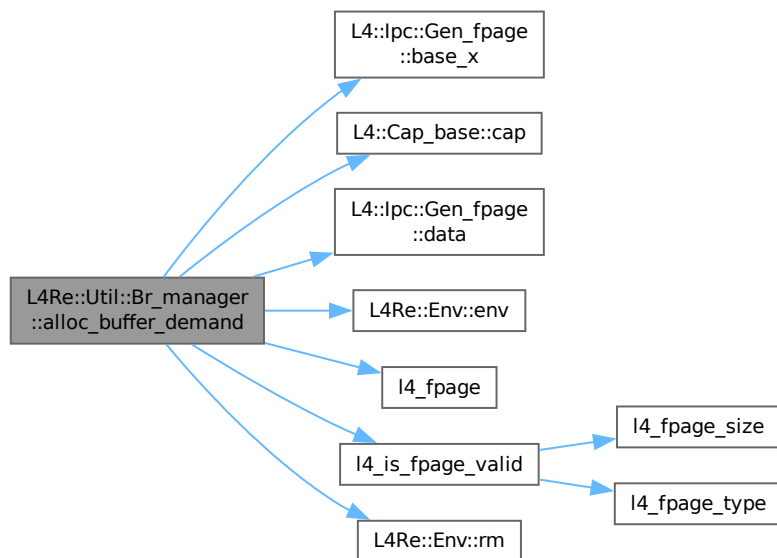
This function is not called by user applications directly. Usually the server implementation or the registry implementation calls this function whenever a new object is registered at the server.

Implements [L4::lpc_svr::Server_iface](#).

Definition at line 72 of file [br_manager](#).

References [L4::lpc::Gen_fpage::base_x\(\)](#), [L4::Cap_base::cap\(\)](#), [L4Re::Util::cap_alloc](#), [L4::Type_info::Demand::caps](#), [L4::lpc::Gen_fpage::data\(\)](#), [L4Re::Env::env\(\)](#), [L4_EINVAL](#), [L4_ENOMEM](#), [L4_EOK](#), [L4_ERANGE](#), [I4_fpage\(\)](#), [L4_FPAGE_RWX](#), [I4_is_fpage_valid\(\)](#), [L4::Type_info::Demand::mem](#), [L4::Type_info::Demand::ports](#), [L4Re::Rm::F::Reserved](#), [L4Re::Env::rm\(\)](#), and [L4Re::Rm::F::Search_addr](#).

Here is the call graph for this function:



15.317.2.2 `get_rcv_cap()`

```
L4::Cap< void > L4Re::Util::Br_manager::get_rcv_cap (
    int index) const [inline], [override], [virtual]
```

Get capability slot allocated to the given receive buffer.

Parameters

| | |
|--------------|--|
| <i>index</i> | The receive buffer index of the expected capability argument ($0 \leq \text{index} < \text{caps}$ registered with <code>alloc_buffer_demand()</code>). |
|--------------|--|

Precondition

$0 \leq \text{index} < \text{caps}$ registered with `alloc_buffer_demand()`

Returns

Capability slot currently allocated to the given receive buffer.

Implements [L4::lpc_svr::Server_iface](#).

Definition at line 139 of file `br_manager`.

References [L4::Cap_base::Invalid](#), and [L4_CAP_MASK](#).

15.317.2.3 `get_rcv_mem()`

```
cxx::Result< L4::Ipc_svr::Server_iface::Mem_window > L4Re::Util::Br_manager::get_rcv_mem ()
[inline], [override], [virtual], [noexcept]
```

Take the current memory receive window.

The caller takes the ownership of the memory receive window. A new receive window will be allocated for the next call.

Attention

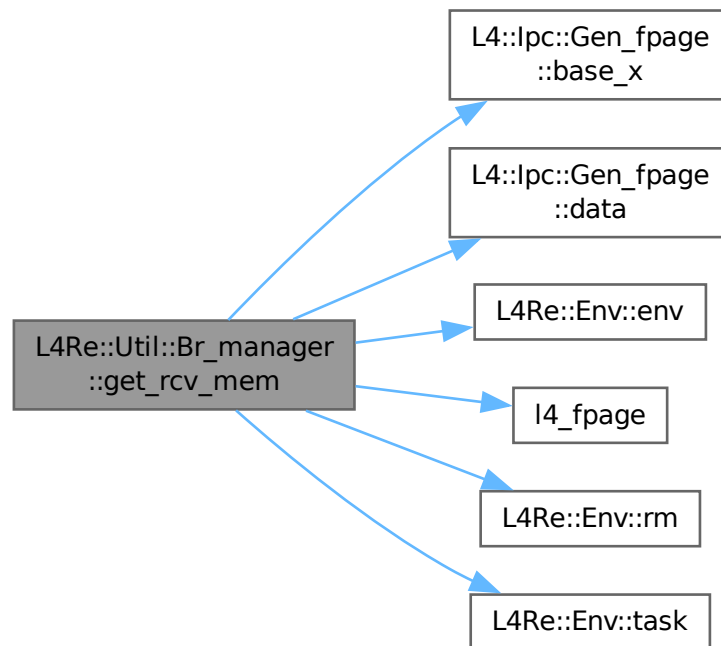
The caller has to ensure that the received IPC call actually mapped some memory flexpage. Still, the IPC client can revoke the pages at any time. Thus, any access into the memory window could create unresolved page faults.

Reimplemented from [L4::lpc_svr::Server_iface](#).

Definition at line 165 of file `br_manager`.

References [L4::lpc::Gen_fpage::base_x\(\)](#), [L4::lpc::Gen_fpage::data\(\)](#), [L4Re::Env::env\(\)](#), [L4_EINVAL](#), [L4_ENOMEM](#), [L4Re::Util::L4_FP_ALL_SPACES](#), [l4_fpage\(\)](#), [L4_FPAGE_RWX](#), [L4Re::Rm::F::Reserved](#), [L4Re::Env::rm\(\)](#), [L4Re::Rm::F::Search_addr](#), and [L4Re::Env::task\(\)](#).

Here is the call graph for this function:



15.317.2.4 realloc_rcv_cap()

```
int L4Re::Util::Br_manager::realloc_rcv_cap (
    int index) [inline], [override], [virtual]
```

Allocate a new capability for the given receive buffer.

Parameters

| | |
|--------------|---|
| <i>index</i> | The receive buffer index of the expected capability argument ($0 \leq \text{index} < \text{caps}$ registered with alloc_buffer_demand()). |
|--------------|---|

Precondition

$0 \leq \text{index} < \text{caps}$ registered with [alloc_buffer_demand\(\)](#)

Returns

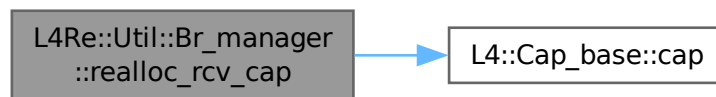
0 on success, < 0 on error.

Implements [L4::lpc_svr::Server_iface](#).

Definition at line 147 of file [br_manager](#).

References [L4::Cap_base::cap\(\)](#), [L4Re::Util::cap_alloc](#), [L4_EINVAL](#), [L4_ENOMEM](#), and [L4_EOK](#).

Here is the call graph for this function:

**15.317.2.5 set_rcv_cap_flags()**

```
void L4Re::Util::Br_manager::set_rcv_cap_flags (
    unsigned long flags) [inline]
```

Set the receive flags for the buffers.

Precondition

Must be called before any handlers are registered.

Parameters

| | |
|--------------|--|
| <i>flags</i> | New receive capability flags, see l4_msg_item_consts_t . |
|--------------|--|

Definition at line 200 of file [br_manager](#).

References [l4_assert](#).

The documentation for this class was generated from the following file:

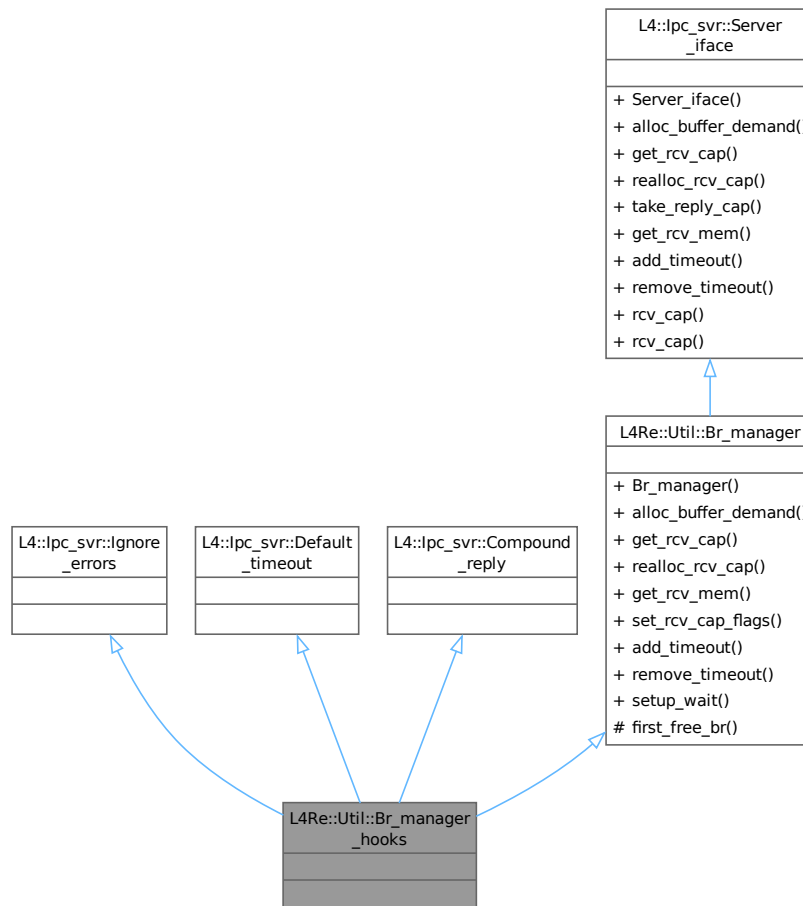
- `l4/re/util/br_manager`

15.318 L4Re::Util::Br_manager_hooks Struct Reference

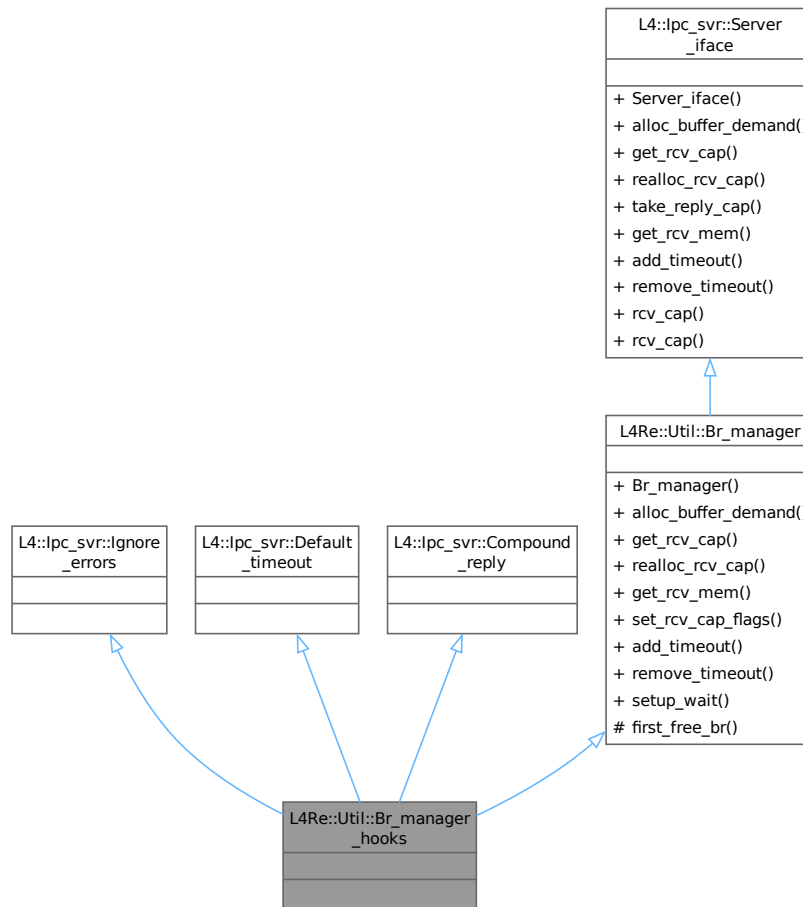
Predefined server-loop hooks for a server loop using the [Br_manager](#).

```
#include <br_manager>
```

Inheritance diagram for L4Re::Util::Br_manager_hooks:



Collaboration diagram for L4Re::Util::Br_manager_hooks:



Additional Inherited Members

Public Types inherited from L4::lpc_svr::Server_iface

- using **Demand** = L4::Type_info::Demand
Data type expressing server-side demand for receive buffers.

Public Member Functions inherited from L4Re::Util::Br_manager

- **Br_manager** ()
Make a buffer-register (BR) manager.
- int **alloc_buffer_demand** (Demand const &d) override
Tells the server to allocate buffers for the given demand.
- L4::Cap< void > **get_rcv_cap** (int i) const override
Get capability slot allocated to the given receive buffer.
- int **realloc_rcv_cap** (int i) override
Allocate a new capability for the given receive buffer.

- `cx::Result< L4::lpc_svr::Server_iface::Mem_window > get_rcv_mem ()` noexcept override
Take the current memory receive window.
- void `set_rcv_cap_flags` (unsigned long flags)
Set the receive flags for the buffers.
- int `add_timeout` (L4::lpc_svr::Timeout *, l4_kernel_clock_t) override
No timeouts handled by us.
- int `remove_timeout` (L4::lpc_svr::Timeout *) override
No timeouts handled by us.
- void `setup_wait` (l4_utcb_t *utcb, L4::lpc_svr::Reply_mode)
setup_wait() used the server loop (L4::Server)

Public Member Functions inherited from L4::lpc_svr::Server_iface

- `Server_iface ()`
Make a server interface.
- virtual `cx::Result< L4::Reply_cap > take_reply_cap ()` noexcept
Take the currently used reply capability.
- template<typename T>
L4::Cap< T > `rcv_cap` (int index) const
Get given receive buffer as typed capability.
- L4::Cap< void > `rcv_cap` (int index) const
Get receive cap with the given index as generic (void) type.

Protected Member Functions inherited from L4Re::Util::Br_manager

- unsigned `first_free_br` () const
Used for assigning BRs for a timeout.

15.318.1 Detailed Description

Predefined server-loop hooks for a server loop using the [Br_manager](#).

This class can be used whenever a server loop including full management of receive buffer resources is needed.

Definition at line 250 of file [br_manager](#).

The documentation for this struct was generated from the following file:

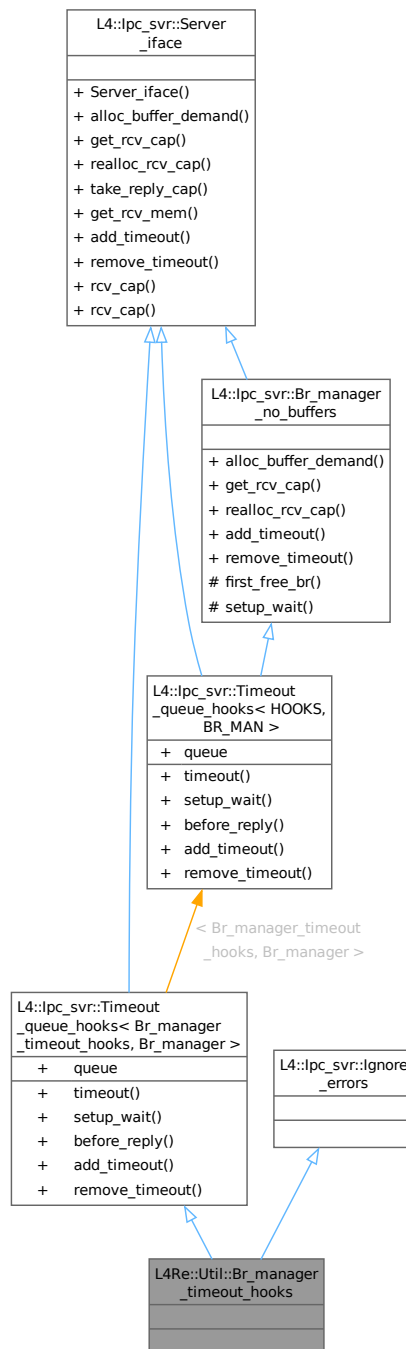
- l4/re/util/br_manager

15.319 L4Re::Util::Br_manager_timeout_hooks Struct Reference

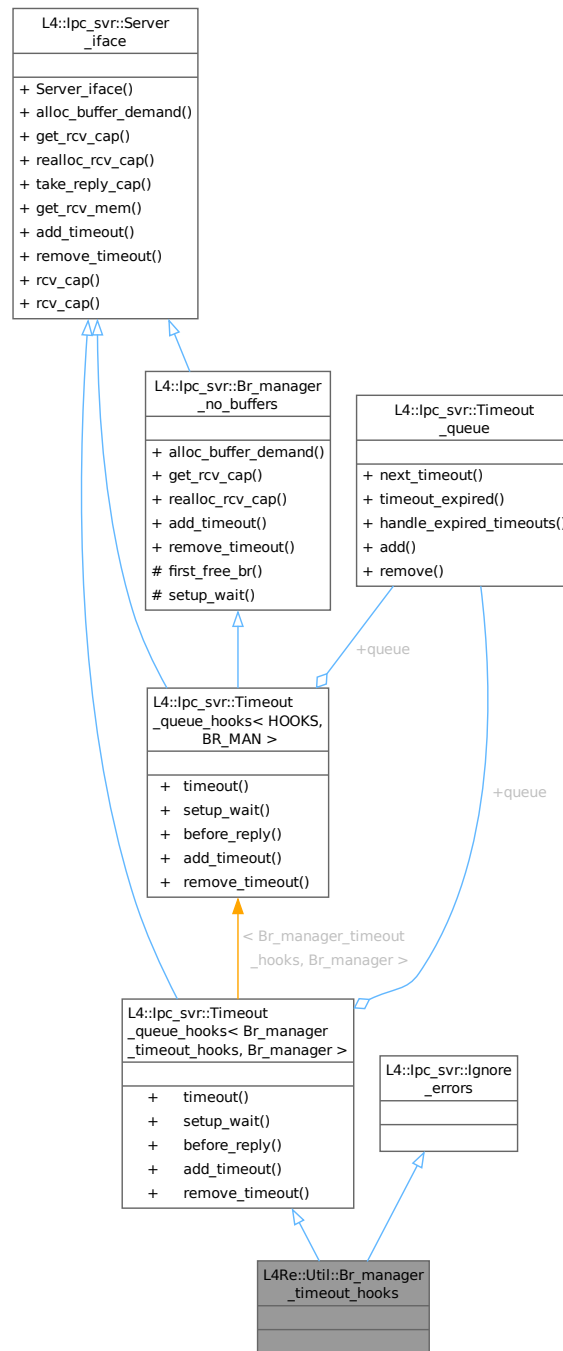
Predefined server-loop hooks for a server with using the [Br_manager](#) and a timeout queue.

```
#include <br_manager>
```

Inheritance diagram for L4Re::Util::Br_manager_timeout_hooks:



Collaboration diagram for L4Re::Util::Br_manager_timeout_hooks:



Additional Inherited Members

Public Types inherited from L4::lpc_svr::Server_iface

- using **Demand** = L4::Type_info::Demand

Data type expressing server-side demand for receive buffers.

Public Member Functions inherited from

[L4::lpc_svr::Timeout_queue_hooks](#)< [Br_manager_timeout_hooks](#), [Br_manager](#) >

- [l4_timeout_t](#) [timeout](#) ()
get the time for the next timeout
- void [setup_wait](#) ([l4_utcb_t](#) *utcb, [L4::lpc_svr::Reply_mode](#) mode)
setup_wait() for the server loop
- [L4::lpc_svr::Reply_mode](#) [before_reply](#) ([l4_msgtag_t](#), [l4_utcb_t](#) *)
server loop hook
- int [add_timeout](#) ([Timeout](#) *timeout, [l4_kernel_clock_t](#) time) override
Add a timeout to the queue for time time.
- int [remove_timeout](#) ([Timeout](#) *timeout) override
Remove timeout from the queue.

Public Member Functions inherited from [L4::lpc_svr::Server_iface](#)

- [Server_iface](#) ()
Make a server interface.
- virtual int [alloc_buffer_demand](#) ([Demand](#) const &demand)=0
Tells the server to allocate buffers for the given demand.
- virtual [L4::Cap](#)< void > [get_rcv_cap](#) (int index) const =0
Get capability slot allocated to the given receive buffer.
- virtual int [realloc_rcv_cap](#) (int index)=0
Allocate a new capability for the given receive buffer.
- virtual [cxx::Result](#)< [L4::Reply_cap](#) > [take_reply_cap](#) () noexcept
Take the currently used reply capability.
- virtual [cxx::Result](#)< [Mem_window](#) > [get_rcv_mem](#) () noexcept
Take the current memory receive window.
- template<typename T>
[L4::Cap](#)< T > [rcv_cap](#) (int index) const
Get given receive buffer as typed capability.
- [L4::Cap](#)< void > [rcv_cap](#) (int index) const
Get receive cap with the given index as generic (void) type.

Data Fields inherited from

[L4::lpc_svr::Timeout_queue_hooks](#)< [Br_manager_timeout_hooks](#), [Br_manager](#) >

- [Timeout_queue](#) [queue](#)
Use this timeout queue.

15.319.1 Detailed Description

Predefined server-loop hooks for a server with using the [Br_manager](#) and a timeout queue.

This class can be used for server loops that need the full package of buffer-register management and a timeout queue.

Definition at line 264 of file [br_manager](#).

The documentation for this struct was generated from the following file:

- [l4/re/util/br_manager](#)

15.320 L4Re::Util::Cap_alloc_base Class Reference

Capability allocator.

```
#include <bitmap_cap_alloc>
```

Collaboration diagram for L4Re::Util::Cap_alloc_base:

| L4Re::Util::Cap_alloc_base | |
|----------------------------|---------|
| | |
| + | alloc() |
| + | free() |

Public Member Functions

- template<typename T>
[L4::Cap](#)< T > **alloc** () noexcept
Allocate a capability slot.
- template<typename T>
void **free** ([L4::Cap](#)< T > const &cap, [l4_cap_idx_t](#) task=[L4_INVALID_CAP](#), [l4_umword_t](#) unmap_↵
flags=[L4_FP_ALL_SPACES](#)) noexcept
Free a capability slot.

15.320.1 Detailed Description

Capability allocator.

Definition at line 28 of file [bitmap_cap_alloc](#).

The documentation for this class was generated from the following file:

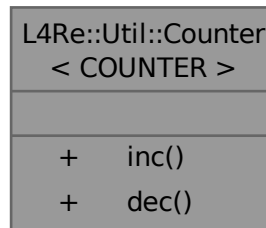
- [l4/re/util/bitmap_cap_alloc](#)

15.321 L4Re::Util::Counter< COUNTER > Struct Template Reference

Counter for [Counting_cap_alloc](#) with variable data width.

```
#include <counting_cap_alloc>
```

Collaboration diagram for L4Re::Util::Counter< COUNTER >:



Public Member Functions

- bool [inc](#) ()
Increment counter if not yet saturated.
- Type [dec](#) ()
Decrement counter if not saturated.

15.321.1 Detailed Description

```
template<typename COUNTER = unsigned char>
struct L4Re::Util::Counter< COUNTER >
```

Counter for [Counting_cap_alloc](#) with variable data width.

This version is not thread safe.

Definition at line 27 of file [counting_cap_alloc](#).

15.321.2 Member Function Documentation

15.321.2.1 dec()

```
template<typename COUNTER = unsigned char>
Type L4Re::Util::Counter< COUNTER >::dec () [inline]
```

Decrement counter if not saturated.

Once the counter reached the saturated state, the counter value isn't changed.

Definition at line 67 of file [counting_cap_alloc](#).

15.321.2.2 inc()

```
template<typename COUNTER = unsigned char>
bool L4Re::Util::Counter< COUNTER >::inc () [inline]
```

Increment counter if not yet saturated.

Once the counter reached the saturated state, the counter value isn't changed.

Return values

| | |
|--------------|---|
| <i>false</i> | The counter just went saturated after it was increased. |
| <i>true</i> | Either the counter was already saturated – in that case the counter value was not changed, or the counter was not saturated – in that case the counter was increased. |

Definition at line 50 of file [counting_cap_alloc](#).

The documentation for this struct was generated from the following file:

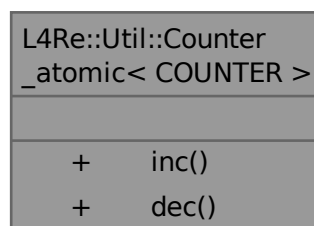
- [l4/re/util/counting_cap_alloc](#)

15.322 L4Re::Util::Counter_atomic< COUNTER > Struct Template Reference

Thread safe version of counter for [Counting_cap_alloc](#).

```
#include <counting_cap_alloc>
```

Collaboration diagram for L4Re::Util::Counter_atomic< COUNTER >:



Public Member Functions

- bool [inc](#) ()
Increment counter if not yet saturated.
- Type [dec](#) ()
Decrement counter if not saturated.

15.322.1 Detailed Description

```
template<typename COUNTER = unsigned char>
struct L4Re::Util::Counter_atomic< COUNTER >
```

Thread safe version of counter for [Counting_cap_alloc](#).

Despite using atomic instructions, this version has to make sure that capability slots are not reused too early. If the last reference is gone, the capability slot has to be unmapped. The slot must only be allocated again when the unmap has completed. This is accomplished by starting with an initial count of 2. The last reference will decrease the counter to 1. Only then, after the slot was unmapped, will the counter be set to 0. This will allow other threads to reallocate the slot.

Definition at line 98 of file [counting_cap_alloc](#).

15.322.2 Member Function Documentation

15.322.2.1 dec()

```
template<typename COUNTER = unsigned char>
Type L4Re::Util::Counter_atomic< COUNTER >::dec () [inline]
```

Decrement counter if not saturated.

Once the counter reached the saturated state, the counter value isn't changed.

Definition at line 142 of file [counting_cap_alloc](#).

15.322.2.2 inc()

```
template<typename COUNTER = unsigned char>
bool L4Re::Util::Counter_atomic< COUNTER >::inc () [inline]
```

Increment counter if not yet saturated.

Once the counter reached the saturated state, the counter value isn't changed.

Return values

| | |
|--------------|---|
| <i>false</i> | The counter just went saturated after it was increased. |
| <i>true</i> | Either the counter was already saturated – in that case the counter value was not changed, or the counter was not saturated – in that case the counter was increased. |

Definition at line 121 of file [counting_cap_alloc](#).

The documentation for this struct was generated from the following file:

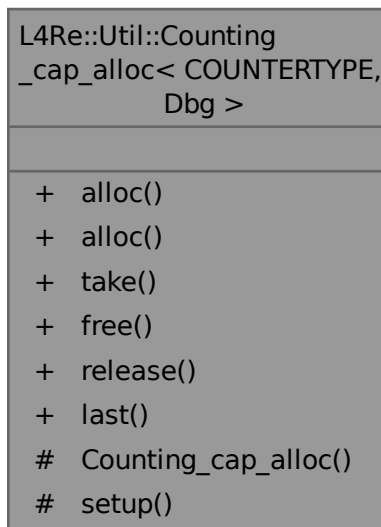
- [l4/re/util/counting_cap_alloc](#)

15.323 L4Re::Util::Counting_cap_alloc< COUNTERTYPE, Dbg > Class Template Reference

Internal reference-counting cap allocator.

```
#include <counting_cap_alloc>
```

Collaboration diagram for L4Re::Util::Counting_cap_alloc< COUNTERTYPE, Dbg >:



Public Member Functions

- `L4::Cap< void > alloc () noexcept`
Allocate a new capability slot.
- `template<typename T> L4::Cap< T > alloc () noexcept`
Allocate a new capability slot.
- `void take (L4::Cap< void > cap) noexcept`
Increase the reference counter for the capability.
- `bool free (L4::Cap< void > cap, i4_cap_idx_t task=L4_INVALID_CAP, unsigned unmap_flags=L4_FP_ALL_SPACES) noexcept`
Free the capability.
- `bool release (L4::Cap< void > cap, i4_cap_idx_t task=L4_INVALID_CAP, unsigned unmap_↔ flags=L4_FP_ALL_SPACES) noexcept`
Decrease the reference counter for a capability.
- `long last () noexcept`
Return highest capability id managed by this allocator.

Protected Member Functions

- [Counting_cap_alloc](#) () noexcept
Create a new, empty allocator.
- void [setup](#) (void *m, long capacity, long bias, Dbg *dbg) noexcept
Set up the backing memory for the allocator and the area of managed capability slots.

15.323.1 Detailed Description

```
template<typename COUNTERTYPE, typename Dbg>
class L4Re::Util::Counting_cap_alloc< COUNTERTYPE, Dbg >
```

Internal reference-counting cap allocator.

This is intended for internal use only. [L4Re](#) applications should use [L4Re::Util::cap_alloc\(\)](#).

Allocator for capability slots that automatically frees the slot and optionally unmaps the capability when the reference count goes down to zero. Reference counting must be done manually via [take\(\)](#) and [release\(\)](#). The backing store for the reference counters must be provided in the [setup\(\)](#) method. The allocator can recognize capability slots that are not managed by itself and does nothing on such slots.

Note

The user must ensure that the backing store is zero-initialized.

The user must ensure that the capability slots managed by this allocator are not used by a different allocator, see [setup\(\)](#).

The operations in this class are not thread-safe.

Definition at line 191 of file [counting_cap_alloc](#).

15.323.2 Constructor & Destructor Documentation

15.323.2.1 Counting_cap_alloc()

```
template<typename COUNTERTYPE, typename Dbg>
L4Re::Util::Counting\_cap\_alloc< COUNTERTYPE, Dbg >::Counting_cap_alloc () [inline], [protected],
[noexcept]
```

Create a new, empty allocator.

Needs to be initialized with [setup\(\)](#) before it can be used.

Definition at line 224 of file [counting_cap_alloc](#).

15.323.3 Member Function Documentation

15.323.3.1 alloc() [1/2]

```
template<typename COUNTERTYPE, typename Dbg>
template<typename T>
L4::Cap< T > L4Re::Util::Counting_cap_alloc< COUNTERTYPE, Dbg >::alloc () [inline], [noexcept]
```

Allocate a new capability slot.

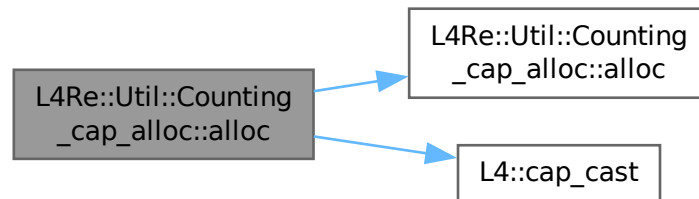
Returns

The newly allocated capability slot, invalid if the allocator was exhausted.

Definition at line 282 of file [counting_cap_alloc](#).

References [alloc\(\)](#), and [L4::cap_cast\(\)](#).

Here is the call graph for this function:



15.323.3.2 alloc() [2/2]

```
template<typename COUNTERTYPE, typename Dbg>
L4::Cap< void > L4Re::Util::Counting_cap_alloc< COUNTERTYPE, Dbg >::alloc () [inline], [noexcept]
```

Allocate a new capability slot.

Returns

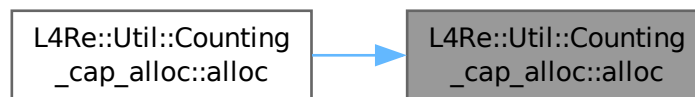
The newly allocated capability slot, invalid if the allocator was exhausted.

Definition at line 257 of file [counting_cap_alloc](#).

References [L4::Cap_base::Invalid](#), and [L4_CAP_SHIFT](#).

Referenced by [alloc\(\)](#).

Here is the caller graph for this function:

**15.323.3.3 free()**

```

template<typename COUNTERTYPE, typename Dbg>
bool L4Re::Util::Counting_cap_alloc< COUNTERTYPE, Dbg >::free (
    L4::Cap< void > cap,
    l4_cap_idx_t task = L4_INVALID_CAP,
    unsigned unmap_flags = L4_FP_ALL_SPACES) [inline], [noexcept]
  
```

Free the capability.

Parameters

| | |
|--------------------|---|
| <i>cap</i> | Capability to free. |
| <i>task</i> | If set, task to unmap the capability from. |
| <i>unmap_flags</i> | Flags for unmap, see l4_unmap_flags_t . |

Precondition

The capability has been allocated. Calling free twice on a capability managed by this allocator results in undefined behaviour.

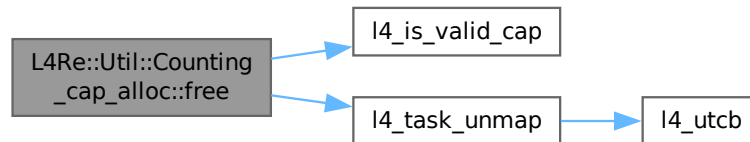
Returns

True, if the capability was managed by this allocator.

Definition at line 321 of file [counting_cap_alloc](#).

References [l4_assert](#), [L4Re::Util::L4_FP_ALL_SPACES](#), [L4_INVALID_CAP](#), [l4_is_valid_cap\(\)](#), and [l4_task_unmap\(\)](#).

Here is the call graph for this function:

**15.323.3.4 release()**

```

template<typename COUNTERTYPE, typename Dbg>
bool L4Re::Util::Counting_cap_alloc< COUNTERTYPE, Dbg >::release (
    L4::Cap< void > cap,
    l4_cap_idx_t task = L4_INVALID_CAP,
    unsigned unmap_flags = L4_FP_ALL_SPACES) [inline], [noexcept]
  
```

Decrease the reference counter for a capability.

Parameters

| | |
|--------------------|---|
| <i>cap</i> | Capability to release. |
| <i>task</i> | If set, task to unmap the capability from. |
| <i>unmap_flags</i> | Flags for unmap, see l4_unmap_flags_t . |

Precondition

The capability has been allocated. Calling `release` on a free capability results in undefined behaviour.

Returns

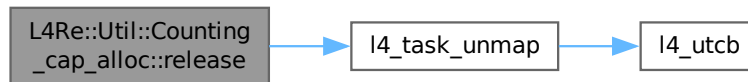
True, if the capability was freed as a result of this operation. If false is returned the capability is either still in use or is not managed by this allocator.

Does nothing apart from returning false if the capability is not managed by this allocator.

Definition at line 359 of file [counting_cap_alloc](#).

References [l4_assert](#), [L4Re::Util::L4_FP_ALL_SPACES](#), [L4_INVALID_CAP](#), and [l4_task_unmap\(\)](#).

Here is the call graph for this function:



15.323.3.5 setup()

```

template<typename COUNTERTYPE, typename Dbg>
void L4Re::Util::Counting_cap_alloc< COUNTERTYPE, Dbg >::setup (
    void * m,
    long capacity,
    long bias,
    Dbg * dbg) [inline], [protected], [noexcept]
  
```

Set up the backing memory for the allocator and the area of managed capability slots.

Parameters

| | |
|-----------------|---|
| <i>m</i> | Pointer to backing memory. |
| <i>capacity</i> | Number of capabilities that can be stored. |
| <i>bias</i> | First capability id to use by this allocator. |
| <i>dbg</i> | Logger for warnings if counter got saturated. |

The allocator will manage the capability slots between `bias` and `bias + capacity - 1` (inclusive). It is the responsibility of the user to ensure that these slots are not used otherwise.

Definition at line 242 of file [counting_cap_alloc](#).

15.323.3.6 take()

```

template<typename COUNTERTYPE, typename Dbg>
void L4Re::Util::Counting_cap_alloc< COUNTERTYPE, Dbg >::take (
    L4::Cap< void > cap) [inline], [noexcept]
  
```

Increase the reference counter for the capability.

Parameters

| | |
|------------|--|
| <i>cap</i> | Capability, whose reference counter should be increased. |
|------------|--|

If the capability was still free, it will be automatically allocated. Silently does nothing if the capability is not managed by this allocator.

Definition at line 296 of file [counting_cap_alloc](#).

References [L4_CAP_SHIFT](#), and [L4_UNLIKELY](#).

The documentation for this class was generated from the following file:

- [l4/re/util/counting_cap_alloc](#)

15.324 L4Re::Util::Dataspace_svr Class Reference

[Dataspace](#) server class.

```
#include <dataspace_svr>
```

Collaboration diagram for L4Re::Util::Dataspace_svr:

| L4Re::Util::Dataspace_svr |
|--|
| <ul style="list-style-type: none"> + map() + map_hook() + take() + release() + copy() + clear() + allocate() + page_shift() + is_static() + map_info() |

Public Member Functions

- [l4_ret_t map](#) (Dataspace::Offset offset, Dataspace::Map_addr local_addr, Dataspace::Flags flags, Dataspace::Map_addr min_addr, Dataspace::Map_addr max_addr, [L4::lpc::Snd_fpage](#) &memory)
Map a region of the dataspace.
- virtual [l4_ret_t map_hook](#) (Dataspace::Offset offs, unsigned order, Dataspace::Flags flags, Dataspace::Map_addr *base, unsigned *send_order)
A hook that is called for acquiring the data to be mapped.
- virtual void [take](#) () noexcept
Take a reference to this dataspace.
- virtual unsigned long [release](#) () noexcept
Release a reference to this dataspace.
- virtual [l4_ret_t copy](#) ([l4_addr_t](#) dst_offs, [l4_umword_t](#) src_id, [l4_addr_t](#) src_offs, unsigned long size) noexcept
Copy from src dataspace to this destination dataspace.
- virtual [l4_ret_t clear](#) (unsigned long offs, unsigned long size) const noexcept
Clear a region in the dataspace.
- virtual [l4_ret_t allocate](#) ([l4_addr_t](#) offset, [l4_size_t](#) size, unsigned access) noexcept
Allocate a region within a dataspace.
- virtual unsigned long [page_shift](#) () const noexcept
Define the size of the flexpage to map.
- virtual bool [is_static](#) () const noexcept
Return whether the dataspace is static.
- virtual [l4_ret_t map_info](#) ([l4_addr_t](#) &start_addr, [l4_addr_t](#) &end_addr) noexcept
Return mapping information for no-MMU systems.

15.324.1 Detailed Description

[Dataspace](#) server class.

The default implementation of the interface provides a continuous dataspace with contiguous pages.

Definition at line 29 of file [dataspace_svr](#).

15.324.2 Member Function Documentation

15.324.2.1 allocate()

```
virtual l4\_ret\_t L4Re::Util::Dataspace_svr::allocate (
    l4\_addr\_t offset,
    l4\_size\_t size,
    unsigned access) [inline], [virtual], [noexcept]
```

Allocate a region within a dataspace.

Parameters

| | |
|---------------|------------------------------------|
| <i>offset</i> | Offset in the dataspace, in bytes. |
| <i>size</i> | Size of the range, in bytes. |

| | |
|---------------|---|
| <i>access</i> | Access mode with which the memory backing the dataspace region should be allocated. |
|---------------|---|

Return values

| | |
|----|---------|
| 0 | Success |
| <0 | Error |

Definition at line 224 of file [dataspace_svr](#).

References [L4_ENODEV](#).

15.324.2.2 clear()

```
virtual l4_ret_t L4Re::Util::Dataspace_svr::clear (  
    unsigned long offs,  
    unsigned long size) const [inline], [virtual], [noexcept]
```

Clear a region in the dataspace.

Parameters

| | |
|-------------|---------------------|
| <i>offs</i> | Start of the region |
| <i>size</i> | Size of the region |

Return values

| | |
|----|---------|
| 0 | Success |
| <0 | Error |

Definition at line 192 of file [dataspace_svr](#).

References [L4_ERANGE](#).

15.324.2.3 copy()

```
virtual l4_ret_t L4Re::Util::Dataspace_svr::copy (  
    l4_addr_t dst_offs,  
    l4_umword_t src_id,  
    l4_addr_t src_offs,  
    unsigned long size) [inline], [virtual], [noexcept]
```

Copy from src dataspace to this destination dataspace.

Parameters

| | |
|-----------------|---------------------------------------|
| <i>dst_offs</i> | Offset into the destination dataspace |
| <i>src_id</i> | Local id of the source dataspace |
| <i>src_offs</i> | Offset into the source dataspace |
| <i>size</i> | Number of bytes to copy |

Return values

| | |
|----------|---|
| ≥ 0 | Number of bytes copied |
| < 0 | An error occurred. The error code may depend on the implementation. |

Definition at line 175 of file [dataspace_svr](#).

References [L4_ENODEV](#).

15.324.2.4 is_static()

```
virtual bool L4Re::Util::Dataspace_svr::is_static () const [inline], [virtual], [noexcept]
```

Return whether the dataspace is static.

Returns

True if dataspace is static

Definition at line 244 of file [dataspace_svr](#).

15.324.2.5 map()

```
l4_ret_t L4Re::Util::Dataspace_svr::map (
    Dataspace::Offset offset,
    Dataspace::Map_addr local_addr,
    Dataspace::Flags flags,
    Dataspace::Map_addr min_addr,
    Dataspace::Map_addr max_addr,
    L4::Ipc::Snd_fpage & memory) [inline]
```

Map a region of the dataspace.

Parameters

| | | |
|--|-------------------|--|
| | <i>offset</i> | Offset to start within data space |
| | <i>local_addr</i> | Local address to map to. |
| | <i>flags</i> | Dataspace flags, see L4Re::Dataspace::F::Flags . |

| | | |
|-----|-----------------|----------------------------------|
| | <i>min_addr</i> | Defines start of receive window. |
| | <i>max_addr</i> | Defines end of receive window. |
| out | <i>memory</i> | Send fpage to map |

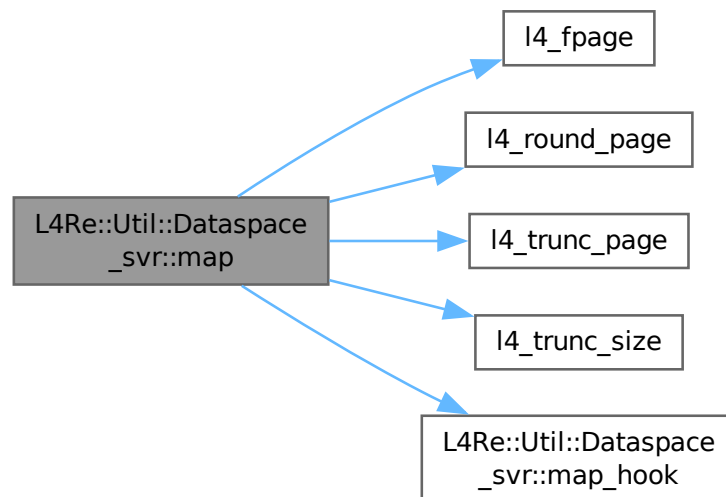
Return values

| | |
|----|---------|
| 0 | Success |
| <0 | Error |

Definition at line 57 of file [dataspace_svr](#).

References [L4_EOK](#), [L4_ERANGE](#), [l4_fpage\(\)](#), [L4_PAGESHIFT](#), [l4_round_page\(\)](#), [l4_trunc_page\(\)](#), [l4_trunc_size\(\)](#), and [map_hook\(\)](#).

Here is the call graph for this function:



15.324.2.6 map_hook()

```

virtual l4\_ret\_t L4Re::Util::Dataspace_svr::map_hook (
    Dataspace::Offset offs,
    unsigned order,
    Dataspace::Flags flags,
    Dataspace::Map_addr * base,
    unsigned * send_order) [inline], [virtual]

```

A hook that is called for acquiring the data to be mapped.

Parameters

| | |
|-------------------|---|
| <i>offs</i> | Offset in dataspace to supply |
| <i>order</i> | Log2-size of data to supply |
| <i>flags</i> | Flags for the mapping |
| <i>base</i> | Start address of the flexpage to be mapped |
| <i>send_order</i> | Order (log2 of size) of the flexpage to be mapped |

Return values

| | |
|----------|--|
| < 0 | Error and the map request will be aborted with that error. |
| ≥ 0 | Success |

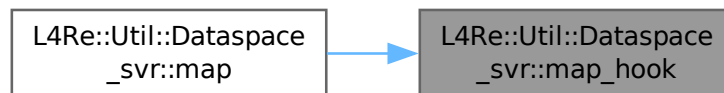
See also

[map](#)

Definition at line 136 of file [dataspace_svr](#).

Referenced by [map\(\)](#).

Here is the caller graph for this function:

**15.324.2.7 map_info()**

```
virtual l4\_ret\_t L4Re::Util::Dataspace_svr::map_info (
    l4\_addr\_t & start_addr,
    l4\_addr\_t & end_addr) [inline], [virtual], [noexcept]
```

Return mapping information for no-MMU systems.

The method is only called on no-MMU systems. It should return the address of the underlying backing buffer so that the caller might map the dataspace.

The default implementation always returns an error because the derived class must provide the required information.

See also

[L4Re::Dataspace::map_info\(\)](#)

Definition at line 259 of file [dataspace_svr](#).

References [L4_EPERM](#).

15.324.2.8 page_shift()

```
virtual unsigned long L4Re::Util::Dataspace_svr::page_shift () const [inline], [virtual],  
[noexcept]
```

Define the size of the flexpage to map.

Returns

flexpage size

Definition at line 236 of file [dataspace_svr](#).

References [L4_LOG2_PAGESIZE](#).

15.324.2.9 release()

```
virtual unsigned long L4Re::Util::Dataspace_svr::release () [inline], [virtual], [noexcept]
```

Release a reference to this dataspace.

Returns

Number of references to the dataspace

Default does nothing and returns always zero.

Definition at line 160 of file [dataspace_svr](#).

15.324.2.10 take()

```
virtual void L4Re::Util::Dataspace_svr::take () [inline], [virtual], [noexcept]
```

Take a reference to this dataspace.

Default does nothing.

Definition at line 150 of file [dataspace_svr](#).

The documentation for this class was generated from the following file:

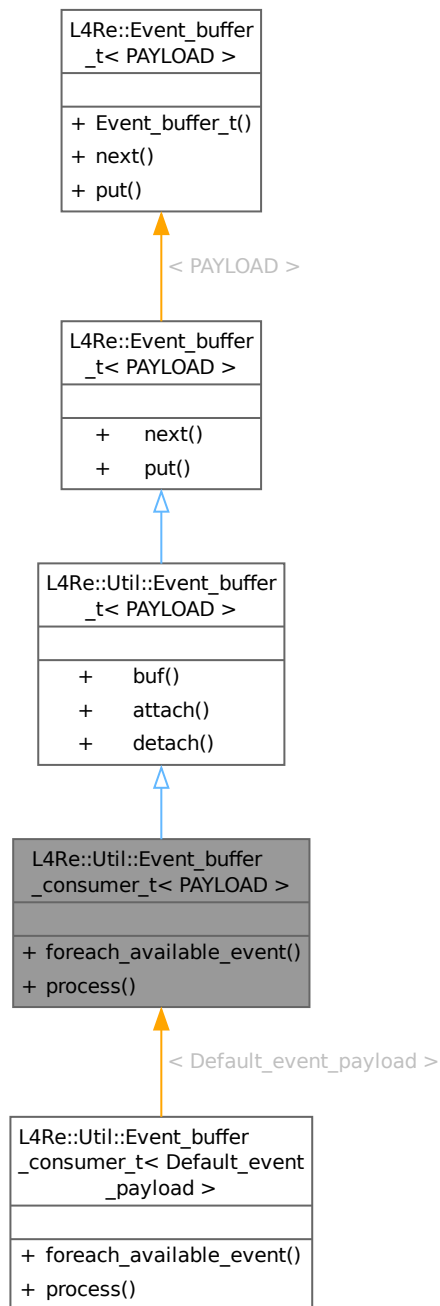
- [l4/re/util/dataspace_svr](#)

15.325 L4Re::Util::Event_buffer_consumer_t< PAYLOAD > Class Template Reference

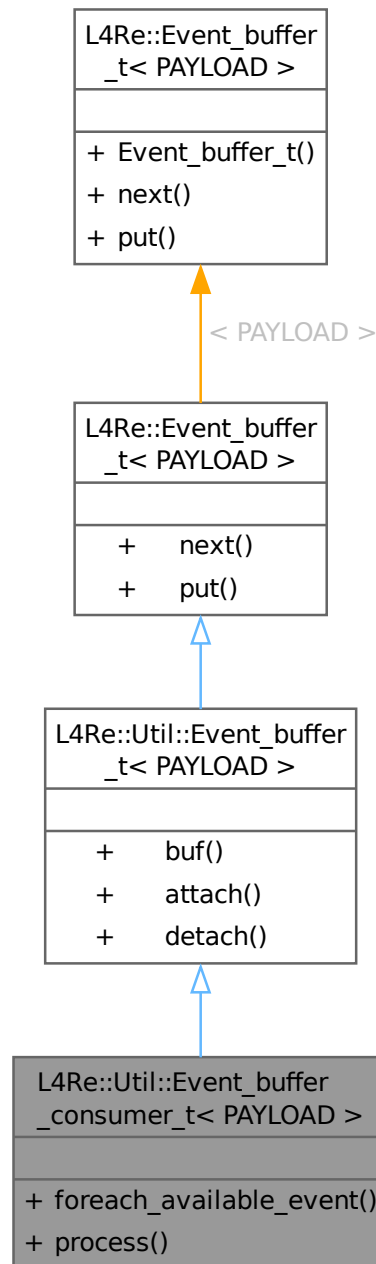
An event buffer consumer.

```
#include <event_buffer>
```

Inheritance diagram for L4Re::Util::Event_buffer_consumer_t< PAYLOAD >:



Collaboration diagram for L4Re::Util::Event_buffer_consumer_t< PAYLOAD >:



Public Member Functions

- `template<typename CB, typename D>`
`void foreach_available_event (CB const &cb, D data=D())`
Call function on every available event.
- `template<typename CB, typename D>`
`void process (L4::Cap< L4::Irq > irq, L4::Cap< L4::Thread > thread, CB const &cb, D data=D())`
Continuously wait for events and process them.

Public Member Functions inherited from [L4Re::Util::Event_buffer_t< PAYLOAD >](#)

- void * [buf](#) () const noexcept
Return the buffer.
- long [attach](#) ([L4::Cap](#)< [L4Re::Dataspace](#) > ds, [L4::Cap](#)< [L4Re::Rm](#) > rm) noexcept
Attach event buffer from address space.
- long [detach](#) ([L4::Cap](#)< [L4Re::Rm](#) > rm) noexcept
Detach event buffer from address space.

Public Member Functions inherited from [L4Re::Event_buffer_t< PAYLOAD >](#)

- Event * [next](#) () noexcept
Next event in buffer.
- bool [put](#) (Event const &ev) noexcept
Put event into buffer at current position.

15.325.1 Detailed Description

template<typename PAYLOAD>

class [L4Re::Util::Event_buffer_consumer_t< PAYLOAD >](#)

An event buffer consumer.

Definition at line 83 of file [event_buffer](#).

15.325.2 Member Function Documentation

15.325.2.1 [foreach_available_event\(\)](#)

template<typename PAYLOAD>

template<typename CB, typename D>

```
void L4Re::Util::Event\_buffer\_consumer\_t< PAYLOAD >::foreach\_available\_event (
    CB const & cb,
    D data = D()) [inline]
```

Call function on every available event.

Parameters

| | |
|-------------|--|
| <i>cb</i> | Function callback. |
| <i>data</i> | Data to pass as an argument to the callback. |

Definition at line 94 of file [event_buffer](#).

Referenced by [L4Re::Util::Event_buffer_consumer_t< Default_event_payload >::process\(\)](#).

Here is the caller graph for this function:



15.325.2.2 process()

```
template<typename PAYLOAD>
template<typename CB, typename D>
void L4Re::Util::Event_buffer_consumer_t< PAYLOAD >::process (
    L4::Cap< L4::Irq > irq,
    L4::Cap< L4::Thread > thread,
    CB const & cb,
    D data = D()) [inline]
```

Continuously wait for events and process them.

Parameters

| | |
|---------------|---|
| <i>irq</i> | Event signal to wait for. |
| <i>thread</i> | Thread capability of the thread calling this function. |
| <i>cb</i> | Callback function that is called for each received event. |
| <i>data</i> | Data to pass as an argument to the processing callback. |

Note

This function never returns.

Definition at line 115 of file [event_buffer](#).

The documentation for this class was generated from the following file:

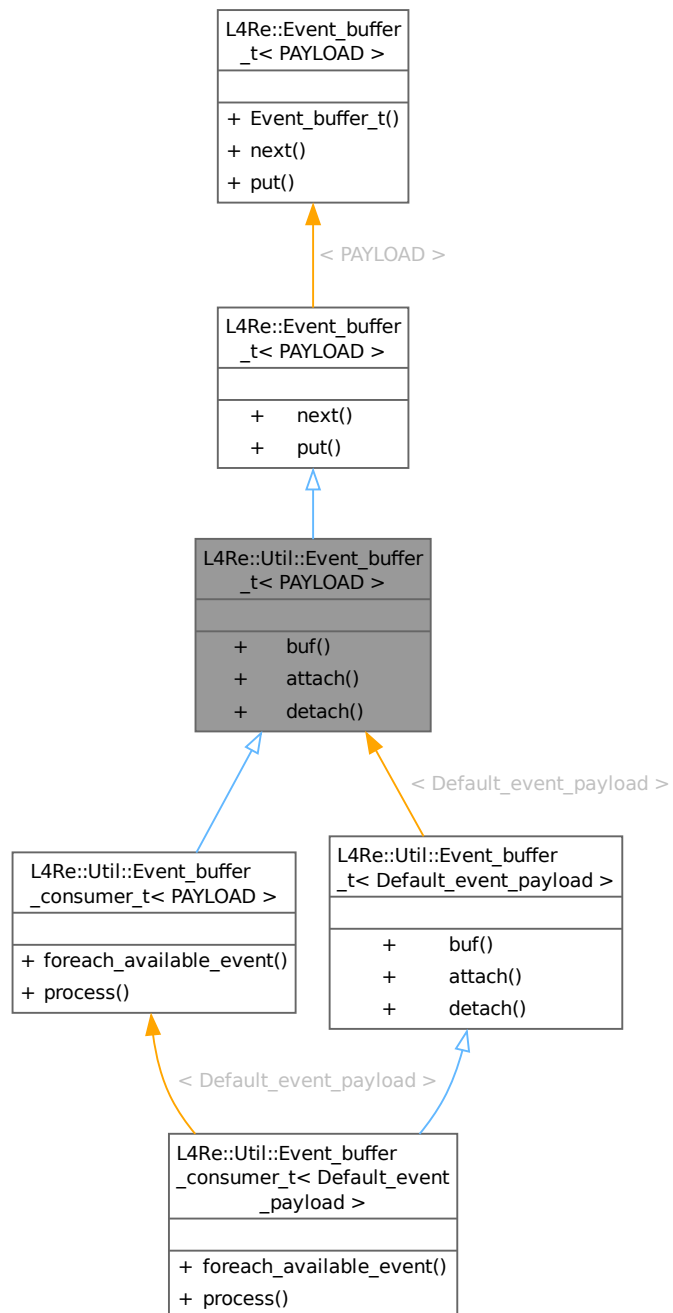
- l4/re/util/event_buffer

15.326 L4Re::Util::Event_buffer_t< PAYLOAD > Class Template Reference

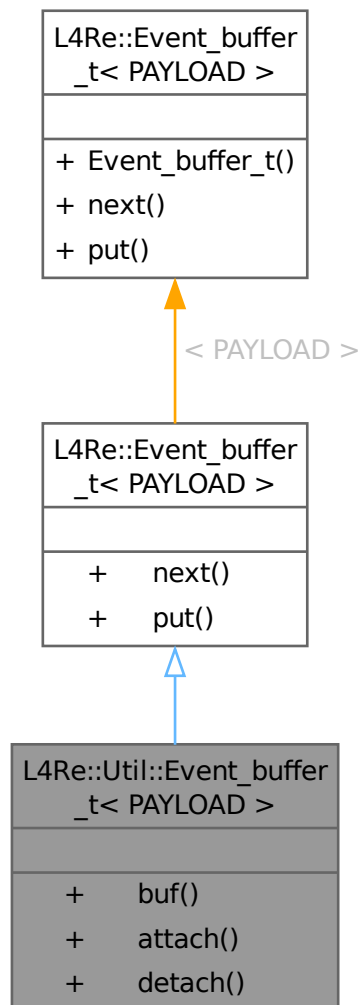
Event_buffer utility class.

```
#include <event_buffer>
```

Inheritance diagram for L4Re::Util::Event_buffer_t< PAYLOAD >:



Collaboration diagram for L4Re::Util::Event_buffer_t< PAYLOAD >:



Public Member Functions

- void * [buf](#) () const noexcept
Return the buffer.
- long [attach](#) (L4::Cap< L4Re::Dataspace > ds, L4::Cap< L4Re::Rm > rm) noexcept
Attach event buffer from address space.
- long [detach](#) (L4::Cap< L4Re::Rm > rm) noexcept
Detach event buffer from address space.

Public Member Functions inherited from [L4Re::Event_buffer_t< PAYLOAD >](#)

- Event * [next](#) () noexcept
Next event in buffer.
- bool [put](#) (Event const &ev) noexcept
Put event into buffer at current position.

15.326.1 Detailed Description

```
template<typename PAYLOAD>
class L4Re::Util::Event_buffer_t< PAYLOAD >
```

Event_buffer utility class.

Definition at line 25 of file [event_buffer](#).

15.326.2 Member Function Documentation

15.326.2.1 attach()

```
template<typename PAYLOAD>
long L4Re::Util::Event_buffer_t< PAYLOAD >::attach (
    L4::Cap< L4Re::Dataspace > ds,
    L4::Cap< L4Re::Rm > rm) [inline], [noexcept]
```

Attach event buffer from address space.

Parameters

| | |
|-----------|--|
| <i>ds</i> | Dataspace of the event buffer. |
| <i>rm</i> | Region manager to attach buffer to. |

Returns

0 on success, negative error code otherwise.

Definition at line 45 of file [event_buffer](#).

15.326.2.2 buf()

```
template<typename PAYLOAD>
void * L4Re::Util::Event_buffer_t< PAYLOAD >::buf () const [inline], [noexcept]
```

Return the buffer.

Returns

Pointer to the event buffer.

Definition at line 35 of file [event_buffer](#).

15.326.2.3 detach()

```
template<typename PAYLOAD>
long L4Re::Util::Event_buffer_t< PAYLOAD >::detach (
    L4::Cap< L4Re::Rm > rm) [inline], [noexcept]
```

Detach event buffer from address space.

Parameters

| | |
|-----------|---------------------------------------|
| <i>rm</i> | Region manager to detach buffer from. |
|-----------|---------------------------------------|

Returns

0 on success, negative error code otherwise.

Definition at line 68 of file [event_buffer](#).

The documentation for this class was generated from the following file:

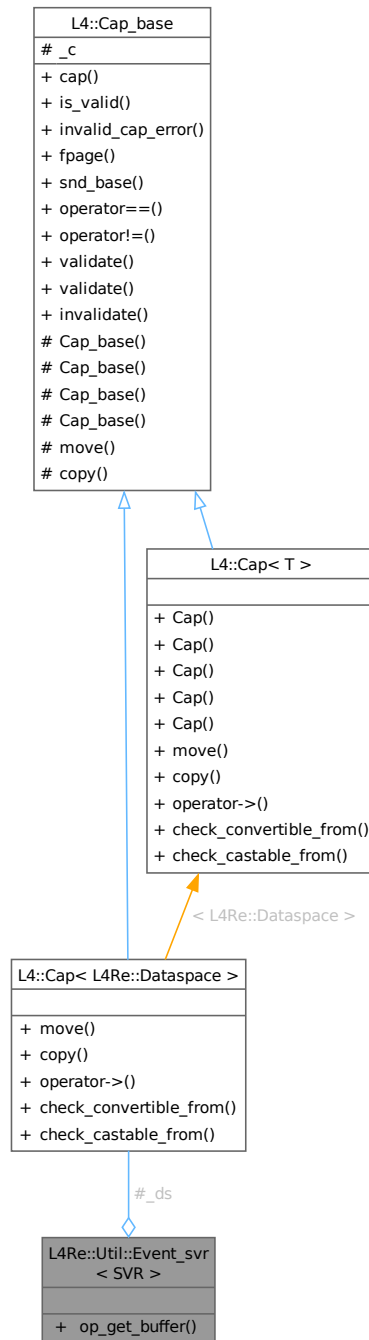
- l4/re/util/event_buffer

15.327 L4Re::Util::Event_svr< SVR > Class Template Reference

Convenience wrapper for implementing an event server.

```
#include <event_svr>
```

Collaboration diagram for L4Re::Util::Event_svr< SVR >:



Public Member Functions

- `l4_ret_t op_get_buffer` (L4Re::Event::Rights, [L4::ipc::Cap< L4Re::Dataspace >](#) &ds)
Handle *L4Re::Event* protocol.

15.327.1 Detailed Description

```
template<typename SVR>
class L4Re::Util::Event_svr< SVR >
```

Convenience wrapper for implementing an event server.

See also

[L4Re::Event](#), [L4Re::Util::Event_t](#)

Definition at line 28 of file [event_svr](#).

The documentation for this class was generated from the following file:

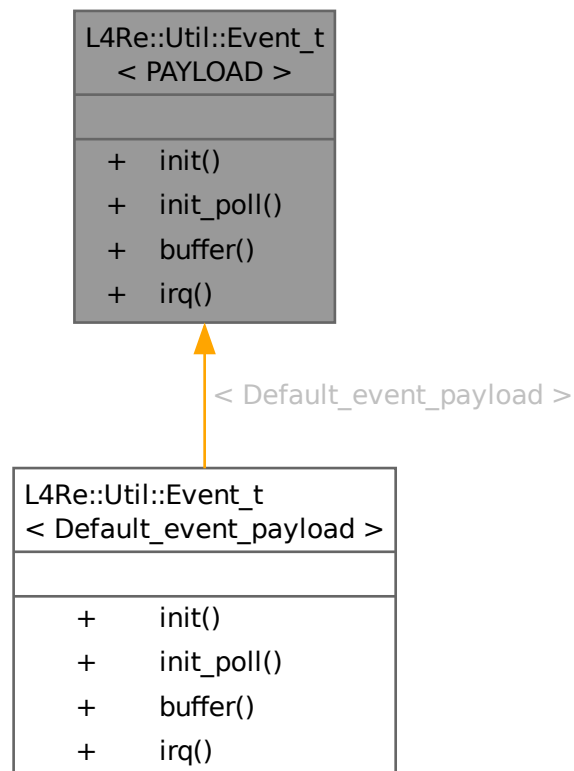
- [l4/re/util/event_svr](#)

15.328 L4Re::Util::Event_t< PAYLOAD > Class Template Reference

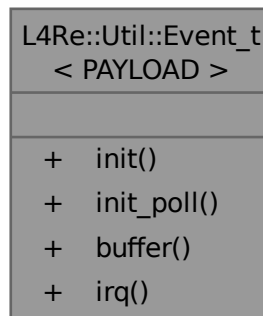
Convenience wrapper for getting access to an event object.

```
#include <event>
```

Inheritance diagram for L4Re::Util::Event_t< PAYLOAD >:



Collaboration diagram for L4Re::Util::Event_t< PAYLOAD >:



Public Types

- enum [Mode](#) { [Mode_irq](#) , [Mode_polling](#) }
- Modes of operation.*

Public Member Functions

- template<typename IRQ_TYPE>
int [init](#) (L4::Cap< [L4Re::Event](#) > event, [L4Re::Env](#) const *env=[L4Re::Env::env\(\)](#), [L4Re::Cap_alloc](#) *ca=&[L4Re::Util::cap_alloc](#))
Initialise an event object.
- int [init_poll](#) (L4::Cap< [L4Re::Event](#) > event, [L4Re::Env](#) const *env=[L4Re::Env::env\(\)](#), [L4Re::Cap_alloc](#) *ca=&[L4Re::Util::cap_alloc](#))
Initialise an event object in polling mode.
- [L4Re::Event_buffer_t](#)< PAYLOAD > & [buffer](#) ()
Get event buffer.
- [L4::Cap](#)< [L4::Triggerable](#) > [irq](#) () const
Get event IRQ.

15.328.1 Detailed Description

```
template<typename PAYLOAD>
class L4Re::Util::Event_t< PAYLOAD >
```

Convenience wrapper for getting access to an event object.

After calling [init\(\)](#) the class supplies the event-buffer and the associated IRQ object.

Definition at line [32](#) of file [event](#).

15.328.2 Member Enumeration Documentation

15.328.2.1 Mode

```
template<typename PAYLOAD>
enum L4Re::Util::Event_t::Mode
```

Modes of operation.

Enumerator

| | |
|--------------|---|
| Mode_irq | Create an IRQ and attach, to get notifications. |
| Mode_polling | Do not use an IRQ. |

Definition at line 38 of file [event](#).

15.328.3 Member Function Documentation

15.328.3.1 buffer()

```
template<typename PAYLOAD>
L4Re::Event_buffer_t< PAYLOAD > & L4Re::Util::Event_t< PAYLOAD >::buffer () [inline]
```

Get event buffer.

Returns

[Event](#) buffer object.

Definition at line 148 of file [event](#).

15.328.3.2 init()

```
template<typename PAYLOAD>
template<typename IRQ_TYPE>
int L4Re::Util::Event_t< PAYLOAD >::init (
    L4::Cap< L4Re::Event > event,
    L4Re::Env const * env = L4Re::Env::env(),
    L4Re::Cap_alloc * ca = &L4Re::Util::cap_alloc) [inline]
```

Initialise an event object.

Template Parameters

| | |
|-----------------|---|
| <i>IRQ_TYPE</i> | Type used for handling notifications from the event provider. This must be derived from L4::Triggerable . |
|-----------------|---|

Parameters

| | |
|--------------|----------------------------------|
| <i>event</i> | Capability to event. |
| <i>env</i> | Pointer to L4Re-Environment |
| <i>ca</i> | Pointer to capability allocator. |

Return values

| | |
|------------|--|
| 0 | Success |
| -L4_ENOMEM | No memory to allocate required capabilities. |
| <0 | Other IPC errors. |

Definition at line 59 of file [event](#).

15.328.3.3 init_poll()

```
template<typename PAYLOAD>
int L4Re::Util::Event_t< PAYLOAD >::init_poll (
    L4::Cap< L4Re::Event > event,
    L4Re::Env const * env = L4Re::Env::env(),
    L4Re::Cap_alloc * ca = &L4Re::Util::cap_alloc) [inline]
```

Initialise an event object in polling mode.

Parameters

| | |
|--------------|----------------------------------|
| <i>event</i> | Capability to event. |
| <i>env</i> | Pointer to L4Re-Environment |
| <i>ca</i> | Pointer to capability allocator. |

Return values

| | |
|------------|--|
| 0 | Success |
| -L4_ENOMEM | No memory to allocate required capabilities. |
| <0 | Other IPC errors. |

Definition at line 112 of file [event](#).

15.328.3.4 irq()

```
template<typename PAYLOAD>
L4::Cap< L4::Triggerable > L4Re::Util::Event_t< PAYLOAD >::irq () const [inline]
```

Get event IRQ.

Returns

[Event](#) IRQ.

Definition at line [155](#) of file [event](#).

The documentation for this class was generated from the following file:

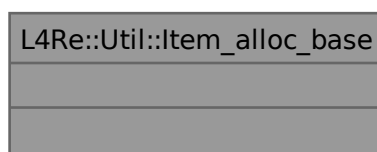
- [I4/re/util/event](#)

15.329 L4Re::Util::Item_alloc_base Class Reference

Item allocator.

```
#include <item_alloc>
```

Collaboration diagram for L4Re::Util::Item_alloc_base:



15.329.1 Detailed Description

Item allocator.

Definition at line [27](#) of file [item_alloc](#).

The documentation for this class was generated from the following file:

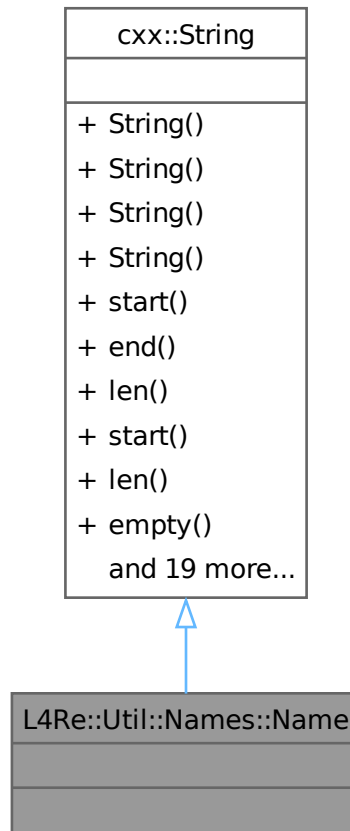
- [I4/re/util/item_alloc](#)

15.330 L4Re::Util::Names::Name Class Reference

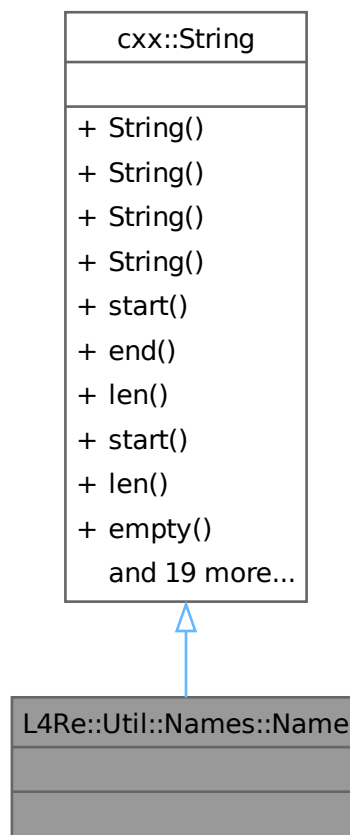
[Name](#) class.

```
#include <name_space_svr>
```

Inheritance diagram for L4Re::Util::Names::Name:



Collaboration diagram for L4Re::Util::Names::Name:



Additional Inherited Members

Public Types inherited from `cxx::String`

- `typedef char const * Index`
Character index type.

Public Member Functions inherited from `cxx::String`

- **String** (char const *s) noexcept
Initialize from a zero-terminated string.
- **String** (char const *s, unsigned long len) noexcept
Initialize from a pointer to first character and a length.
- **String** (char const *s, char const *e) noexcept
Initialize with start and end pointer.
- **String** ()
Zero-initialize. Create an invalid string.

- **Index start** () const
Pointer to first character.
- **Index end** () const
Pointer to first byte behind the string.
- int **len** () const
Length.
- void **start** (char const *s)
Set start.
- void **len** (unsigned long len)
Set length.
- bool **empty** () const
Check if the string has length zero.
- **String head** (**Index end**) const
Return prefix up to index.
- **String head** (unsigned long **end**) const
*Prefix of length **end**.*
- **String substr** (unsigned long idx, unsigned long **len**=~0UL) const
*Substring of length **len** starting at **idx**.*
- **String substr** (char const ***start**, unsigned long **len**=0) const
*Substring of length **len** starting at **start**.*
- template<typename F>
char const * **find_match** (F &&match) const
*Find matching character. **match** should be a function such as **isspace**.*
- char const * **find** (char const *c) const
*Find character. Return **end()** if not found.*
- char const * **find** (int c) const
*Find character. Return **end()** if not found.*
- char const * **rfind** (char const *c) const
*Find right-most character. Return **end()** if not found.*
- **Index starts_with** (cxx::String const &c) const
*Check if **c** is a prefix of string.*
- char const * **find** (int c, char const *s) const
*Find character **c** starting at position **s**. Return **end()** if not found.*
- char const * **find** (char const *c, char const *s) const
*Find character **c** starting at position **s**.*
- char const & **operator[]** (unsigned long idx) const
*Get character at **idx**.*
- char const & **operator[]** (int idx) const
*Get character at **idx**.*
- char const & **operator[]** (**Index** idx) const
*Get character at **idx**.*
- bool **eof** (char const *s) const
*Check if pointer **s** points behind string.*
- template<typename INT>
int **from_dec** (INT *v) const
Convert decimal string to integer.
- template<typename INT>
int **from_hex** (INT *v) const
Convert hex string to integer.
- bool **operator==** (**String** const &o) const
Equality.
- bool **operator!=** (**String** const &o) const
Inequality.

15.330.1 Detailed Description

[Name](#) class.

Definition at line 28 of file [name_space_svr](#).

The documentation for this class was generated from the following file:

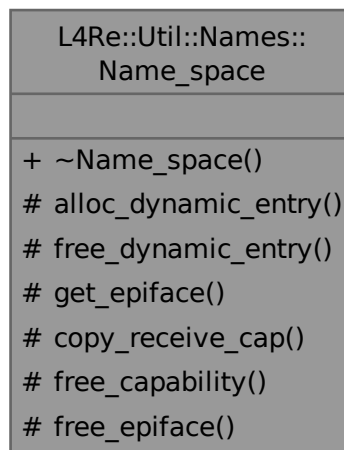
- [l4/re/util/name_space_svr](#)

15.331 L4Re::Util::Names::Name_space Class Reference

Abstract server-side implementation of the L4::Namespace interface.

```
#include <name_space_svr>
```

Collaboration diagram for L4Re::Util::Names::Name_space:



Public Member Functions

- virtual **~Name_space** ()

The destructor of the derived class is responsible for freeing resources.

Protected Member Functions

- virtual `Entry * alloc_dynamic_entry (Name const &n, unsigned flags)=0`
Allocate a new entry for the given name.
- virtual `void free_dynamic_entry (Entry *e)=0`
Free an entry previously allocated with `alloc_dynamic_entry()`.
- virtual `l4_ret_t get_epiface (l4_umword_t data, bool is_local, L4::Epiface **lo)=0`
Return a pointer to the epiface assigned to a given label.
- virtual `l4_ret_t copy_receive_cap (L4::Cap< void > *cap)=0`
Return the receive capability for permanent use.
- virtual `void free_capability (L4::Cap< void > cap)=0`
Free a capability previously acquired with `copy_receive_cap()`.
- virtual `void free_epiface (L4::Epiface *epiface)=0`
Free epiface previously acquired with `get_epiface()`.

15.331.1 Detailed Description

Abstract server-side implementation of the `L4::Namespace` interface.

Note

The derived class is responsible for resource management through the provided interfaces. This includes freeing all resources on destruction!

Definition at line 180 of file `name_space_svr`.

15.331.2 Member Function Documentation

15.331.2.1 `alloc_dynamic_entry()`

```
virtual Entry * L4Re::Util::Names::Name_space::alloc_dynamic_entry (
    Name const & n,
    unsigned flags) [protected], [pure virtual]
```

Allocate a new entry for the given name.

Parameters

| | |
|--------------|---|
| <i>n</i> | <code>Name</code> of the entry, must be copied. |
| <i>flags</i> | Entry flags, see <code>Obj::Flags</code> . |

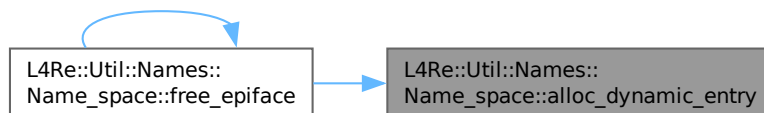
Returns

A pointer to the newly allocated entry or NULL on error.

This method is called when a new entry was received. It must allocate memory, copy `n` out of the receive buffer and wrap everything in an Entry.

Referenced by [free_epiface\(\)](#).

Here is the caller graph for this function:

**15.331.2.2 copy_receive_cap()**

```
virtual l4\_ret\_t L4Re::Util::Names::Name_space::copy_receive_cap (
    L4::Cap< void > * cap) [protected], [pure virtual]
```

Return the receive capability for permanent use.

Parameters

| | | |
|-----|------------|--|
| out | <i>cap</i> | Capability slot with the received capability. Must be permanently available until free_capability() is called. |
|-----|------------|--|

This method is called when a new entry is registered together with a capability mapping. It must decide whether and where to store the capability and return the final capability slot. Typical implementations return the capability slot in the receive window and allocate a new receive window.

15.331.2.3 free_capability()

```
virtual void L4Re::Util::Names::Name_space::free_capability (
    L4::Cap< void > cap) [protected], [pure virtual]
```

Free a capability previously acquired with [copy_receive_cap\(\)](#).

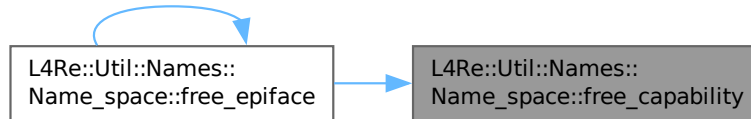
Parameters

| | | |
|----|------------|---------------------|
| in | <i>cap</i> | Capability to free. |
|----|------------|---------------------|

Counterpart of [copy_receive_cap](#). Free the capability slot when the entry is deleted or changed.

Referenced by [free_epiface\(\)](#).

Here is the caller graph for this function:



15.331.2.4 `free_dynamic_entry()`

```
virtual void L4Re::Util::Names::Name_space::free_dynamic_entry (
    Entry * e)    [protected], [pure virtual]
```

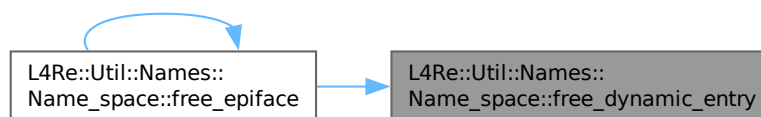
Free an entry previously allocated with [alloc_dynamic_entry\(\)](#).

Parameters

| | |
|----------|----------------|
| <i>e</i> | Entry to free. |
|----------|----------------|

Referenced by [free_epiface\(\)](#).

Here is the caller graph for this function:



15.331.2.5 `free_epiface()`

```
virtual void L4Re::Util::Names::Name_space::free_epiface (
    L4::Epiface * epiface) [protected], [pure virtual]
```

Free epiface previously acquired with [get_epiface\(\)](#).

Parameters

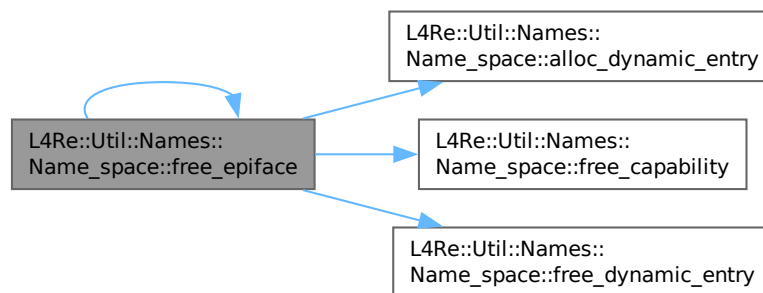
| | | |
|----|----------------|------------------|
| in | <i>epiface</i> | Epiface to free. |
|----|----------------|------------------|

Called when an entry that points to an epiface is deleted allowing implementations that hold resources to free them.

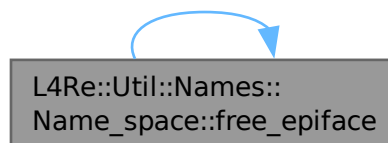
References [alloc_dynamic_entry\(\)](#), [free_capability\(\)](#), [free_dynamic_entry\(\)](#), [free_epiface\(\)](#), [L4_BASE_TASK_CAP](#), [L4_EEXIST](#), [L4_ENOMEM](#), and [L4Re::Namespace::Overwrite](#).

Referenced by [free_epiface\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.331.2.6 get_epiface()

```

virtual l4_ret_t L4Re::Util::Names::Name_space::get_epiface (
    l4_umword_t data,
    bool is_local,
    L4::Epiface ** lo) [protected], [pure virtual]
  
```

Return a pointer to the epiface assigned to a given label.

Parameters

| | | |
|-----|-----------------|--|
| in | <i>data</i> | Label or in the local case the capability slot of the receiving capability. |
| in | <i>is_local</i> | If true, a local capability slot was supplied, if false the label of a locally bound IPC gate. |
| out | <i>lo</i> | Pointer to epiface responsible for the capability. |

Returns

[L4_EOK](#) if a valid interface could be found or an error message otherwise.

This method is called when a new entry is registered and some local ID was received for the capability. In this case, the generic implementation needs to get the epiface in order to get the capability.

The callee must make sure that the epiface remains valid until `free_epiface` is called. In particular, the capability slot must not be reallocated as long as the namespace server holds a reference to the epiface.

The documentation for this class was generated from the following file:

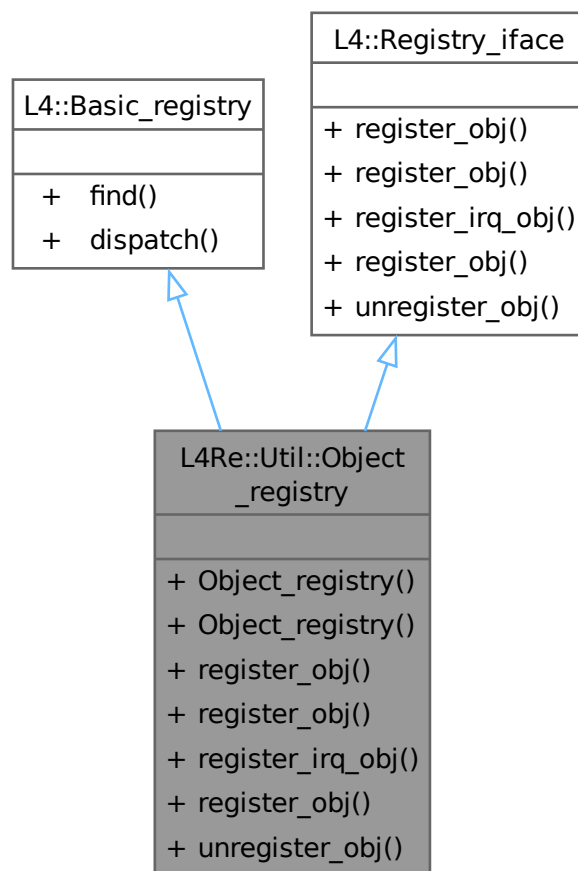
- `l4/re/util/name_space_svr`

15.332 L4Re::Util::Object_registry Class Reference

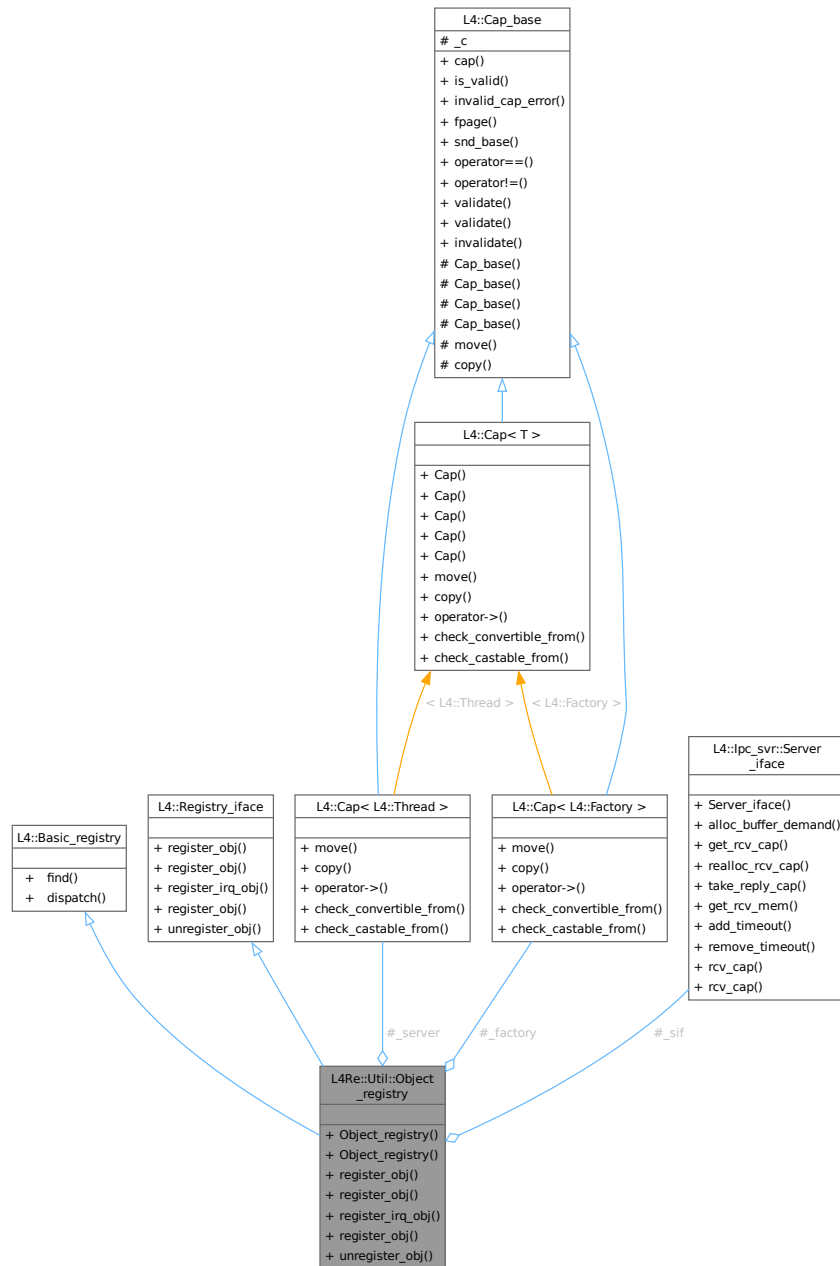
A registry that manages server objects and their attached IPC gates for a single server loop for a specific thread.

```
#include <object_registry>
```

Inheritance diagram for L4Re::Util::Object_registry:



Collaboration diagram for L4Re::Util::Object_registry:



Public Member Functions

- `Object_registry (L4::lpc_svr::Server_iface *sif)`
Create a registry for the main thread of the task using the default factory.
- `Object_registry (L4::lpc_svr::Server_iface *sif, L4::Cap< L4::Thread > server, L4::Cap< L4::Factory > factory)`
Create a registry for arbitrary threads.
- `L4::Cap< void > register_obj (L4::Epiface *o, char const *service)` override
Register a new server object to a pre-allocated receive endpoint.
- `L4::Cap< void > register_obj (L4::Epiface *o)` override

- Register a new server object on a newly allocated capability.*
 - [L4::Cap](#)< [L4::Irq](#) > [register_irq_obj](#) ([L4::Epiface](#) *o) override
Register a handler for an interrupt.
 - [L4::Cap](#)< [L4::Rcv_endpoint](#) > [register_obj](#) ([L4::Epiface](#) *o, [L4::Cap](#)< [L4::Rcv_endpoint](#) > ep) override
Register a handler for an already existing interrupt.
 - void [unregister_obj](#) ([L4::Epiface](#) *o, bool unmap=true) override
Remove a server object from the handler list.

Additional Inherited Members

Static Public Member Functions inherited from [L4::Basic_registry](#)

- static Value * [find](#) ([l4_umword_t](#) label)
Get the server object for an [lpc_gate](#) label.
- static [l4_msgtag_t](#) [dispatch](#) ([l4_msgtag_t](#) tag, [l4_umword_t](#) label, [l4_utcb_t](#) *utcb)
The dispatch function called by the server loop.

15.332.1 Detailed Description

A registry that manages server objects and their attached IPC gates for a single server loop for a specific thread.

This class manages most of the setup of a server object. If necessary, an IPC gate is created, the specified thread is bound to the IPC gate. Incoming IPC is dispatched to the server object based on the label of the IPC gate.

The object registry is also able to manage IRQ endpoints. They require a different method for the object creation. Otherwise they are handled in the same way as IPC gates: a server object is responsible to process the incoming interrupts.

Definition at line 41 of file [object_registry](#).

15.332.2 Constructor & Destructor Documentation

15.332.2.1 [Object_registry\(\)](#) [1/2]

```
L4Re::Util::Object_registry::Object_registry (
    L4::Ipc\_svr::Server\_iface * sif) [inline], [explicit]
```

Create a registry for the main thread of the task using the default factory.

Parameters

| | |
|------------|------------------------|
| <i>sif</i> | Server loop interface. |
|------------|------------------------|

Definition at line 67 of file [object_registry](#).

15.332.2.2 Object_registry() [2/2]

```
L4Re::Util::Object_registry::Object_registry (
    L4::Ipc_svr::Server_iface * sif,
    L4::Cap< L4::Thread > server,
    L4::Cap< L4::Factory > factory) [inline]
```

Create a registry for arbitrary threads.

Parameters

| | |
|----------------|---|
| <i>sif</i> | Server loop interface. |
| <i>server</i> | Capability to the thread that executes the server objects. |
| <i>factory</i> | Capability to a factory object capable of creating new IPC gates. |

Definition at line 81 of file [object_registry](#).

15.332.3 Member Function Documentation

15.332.3.1 register_irq_obj()

```
L4::Cap< L4::Irq > L4Re::Util::Object_registry::register_irq_obj (
    L4::Epiface * o) [inline], [override], [virtual]
```

Register a handler for an interrupt.

Parameters

| | |
|----------|----------------------------------|
| <i>o</i> | Server object that handles IRQs. |
|----------|----------------------------------|

Return values

| | |
|---|--|
| L4::Cap<L4::Irq> | Capability to a new IRQ object on success. |
| L4::Cap<L4::Irq>::Invalid | The allocation of the IRQ has failed. |

The IRQ will be newly allocated using the registry's factory object. The caller must call [unregister_obj\(\)](#) to free all resources.

Implements [L4::Registry_iface](#).

Definition at line 227 of file [object_registry](#).

15.332.3.2 register_obj() [1/3]

```
L4::Cap< void > L4Re::Util::Object_registry::register_obj (
    L4::Epiface * o) [inline], [override], [virtual]
```

Register a new server object on a newly allocated capability.

Parameters

| | |
|----------|--|
| <i>o</i> | Server object that handles IPC requests. |
|----------|--|

Return values

| | |
|---------------------------------------|--|
| <i>L4::Cap< void ></i> | A valid capability to a new IPC gate. |
| <i>L4::Cap< void >::Invalid</i> | The allocation of the IPC gate has failed. |

The IPC gate will be allocated using the registry's factory. The caller must call [unregister_obj\(\)](#) to free all resources.

Implements [L4::Registry_iface](#).

Definition at line 211 of file [object_registry](#).

15.332.3.3 register_obj() [2/3]

```
L4::Cap< void > L4Re::Util::Object_registry::register_obj (
    L4::Epiface * o,
    char const * service) [inline], [override], [virtual]
```

Register a new server object to a pre-allocated receive endpoint.

Parameters

| | |
|----------------|---|
| <i>o</i> | Server object that handles IPC requests. |
| <i>service</i> | Name of a pre-allocated receive endpoint. |

Return values

| | |
|---------------------------------------|--|
| <i>L4::Cap< void ></i> | The capability known as <i>service</i> on success. |
| <i>L4::Cap< void >::Invalid</i> | No capability with the given name found. |

The interface must be freed with [unregister_obj\(\)](#) by the caller to unbind the thread from the capability.

Implements [L4::Registry_iface](#).

Definition at line 194 of file [object_registry](#).

15.332.3.4 register_obj() [3/3]

```
L4::Cap< L4::Rcv_endpoint > L4Re::Util::Object_registry::register_obj (
    L4::Epiface * o,
    L4::Cap< L4::Rcv_endpoint > ep) [inline], [override], [virtual]
```

Register a handler for an already existing interrupt.

Parameters

| | |
|-----------|---|
| <i>o</i> | Server object that handles the IPC. |
| <i>ep</i> | Capability to a receive endpoint, may be a hardware or software interrupt or an IPC gate. |

Return values

| | |
|--|--------------------------------------|
| L4::Cap<L4::Rcv_endpoint> | Capability <i>ep</i> on success. |
| L4::Cap<L4::Rcv_endpoint>::Invalid | The IRQ attach operation has failed. |

The interface must be freed with [unregister_obj\(\)](#) by the caller to unbind the thread from the capability.

Implements [L4::Registry_iface](#).

Definition at line 246 of file [object_registry](#).

15.332.3.5 unregister_obj()

```
void L4Re::Util::Object_registry::unregister_obj (
    L4::Epiface * o,
    bool unmap = true) [inline], [override], [virtual]
```

Remove a server object from the handler list.

Parameters

| | |
|--------------|--|
| <i>o</i> | Server object to unbind. |
| <i>unmap</i> | Specifies if the object capability shall be unmapped (true) or not. The default (true) is to unmap the capability. |

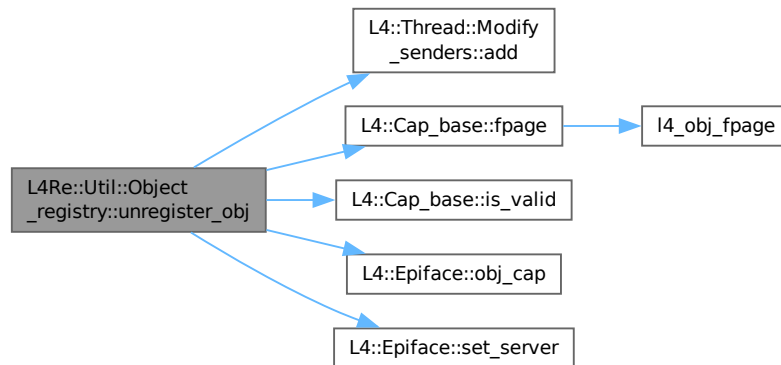
The capability used by the server object will be unmapped if *unmap* is true.

Implements [L4::Registry_iface](#).

Definition at line 262 of file [object_registry](#).

References [L4::Thread::Modify_senders::add\(\)](#), [L4Re::Util::cap_alloc](#), [L4::Cap_base::fpage\(\)](#), [L4::Cap_base::Invalid](#), [L4::Cap_base::is_valid\(\)](#), [L4Re::Util::L4_FP_ALL_SPACES](#), [L4::Epiface::obj_cap\(\)](#), and [L4::Epiface::set_server\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

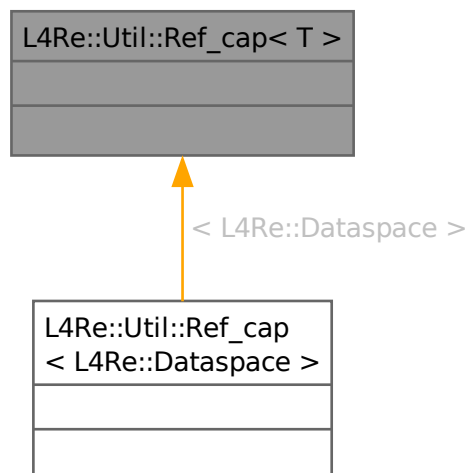
- `l4/re/util/object_registry`

15.333 L4Re::Util::Ref_cap< T > Struct Template Reference

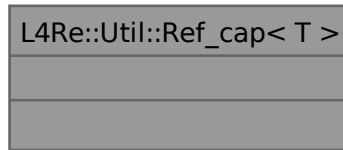
Automatic capability that implements automatic free and unmap of the capability selector.

```
#include <cap_alloc>
```

Inheritance diagram for L4Re::Util::Ref_cap< T >:



Collaboration diagram for L4Re::Util::Ref_cap< T >:



15.333.1 Detailed Description

```
template<typename T>
struct L4Re::Util::Ref_cap< T >
```

Automatic capability that implements automatic free and unmap of the capability selector.

Template Parameters

| | |
|----------|--|
| <i>T</i> | Type of the object that is referred by the capability. |
|----------|--|

This kind of automatic capability implements a counted reference to a capability selector. The capability shall be unmapped and freed when the reference count in the allocator goes to zero.

Usage:

```
L4Re::Util::Ref_cap<L4Re::Dataspace>::Cap global_ds_cap;

{
    L4Re::Util::Ref_cap<L4Re::Dataspace>::Cap
    ds_cap(L4Re::Util::cap_alloc.alloc<L4Re::Dataspace>());
    // reference count for the allocated cap selector is now 1

    // use the dataspace cap
    L4Re::chksys(mem_alloc->alloc(4096, ds_cap.get()));

    global_ds_cap = ds_cap;
    // reference count is now 2
    ...
}
// reference count dropped to 1 (ds_cap is no longer existing).
```

Definition at line 184 of file [cap_alloc](#).

The documentation for this struct was generated from the following file:

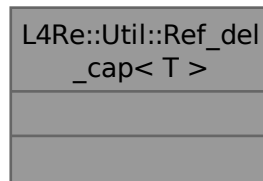
- [l4/re/util/cap_alloc](#)

15.334 L4Re::Util::Ref_del_cap< T > Struct Template Reference

Automatic capability that implements automatic free and unmap+delete of the capability selector.

```
#include <cap_alloc>
```

Collaboration diagram for L4Re::Util::Ref_del_cap< T >:



15.334.1 Detailed Description

```
template<typename T>
struct L4Re::Util::Ref_del_cap< T >
```

Automatic capability that implements automatic free and unmap+delete of the capability selector.

Template Parameters

| | |
|----------|--|
| <i>T</i> | Type of the object that is referred by the capability. |
|----------|--|

This kind of automatic capability implements a counted reference to a capability selector. The capability shall be unmapped and freed when the reference count in the allocator goes to zero. The main difference to [Ref_cap](#) is that the unmap is done with the deletion flag enabled and this leads to the deletion of the object if the current task holds appropriate deletion rights.

Usage:

```

L4Re::Util::Ref_del_cap<L4Re::Dataspace>::Cap global_ds_cap;

{
    L4Re::Util::Ref_del_cap<L4Re::Dataspace>::Cap
        ds_cap(L4Re::Util::cap_alloc.alloc<L4Re::Dataspace>());
    // reference count for the allocated cap selector is now 1

    // use the dataspace cap
    L4Re::chksys(mem_alloc->alloc(4096, ds_cap.get()));

    global_ds_cap = ds_cap;
    // reference count is now 2
    ...
}
// reference count dropped to 1 (ds_cap is no longer existing).
...
global_ds_cap = L4_INVALID_CAP;
// reference count dropped to 0 (data space shall be deleted).
```

Definition at line 225 of file [cap_alloc](#).

The documentation for this struct was generated from the following file:

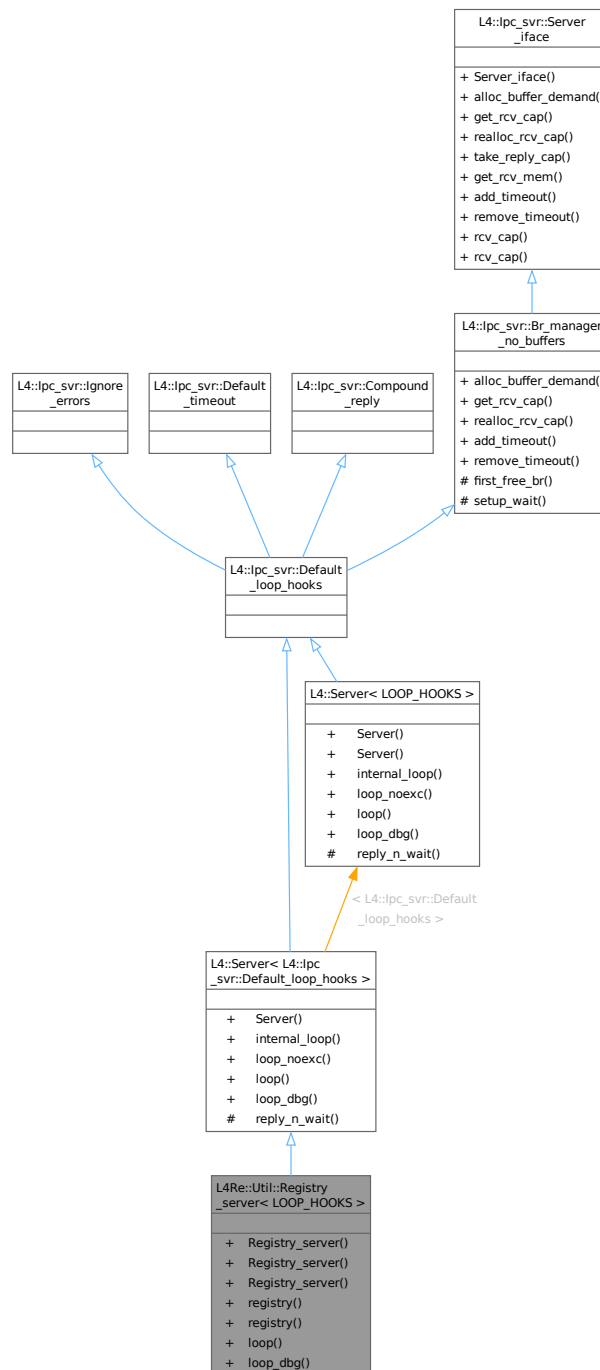
- [l4/re/util/cap_alloc](#)

15.335 L4Re::Util::Registry_server< LOOP_HOOKS > Class Template Reference

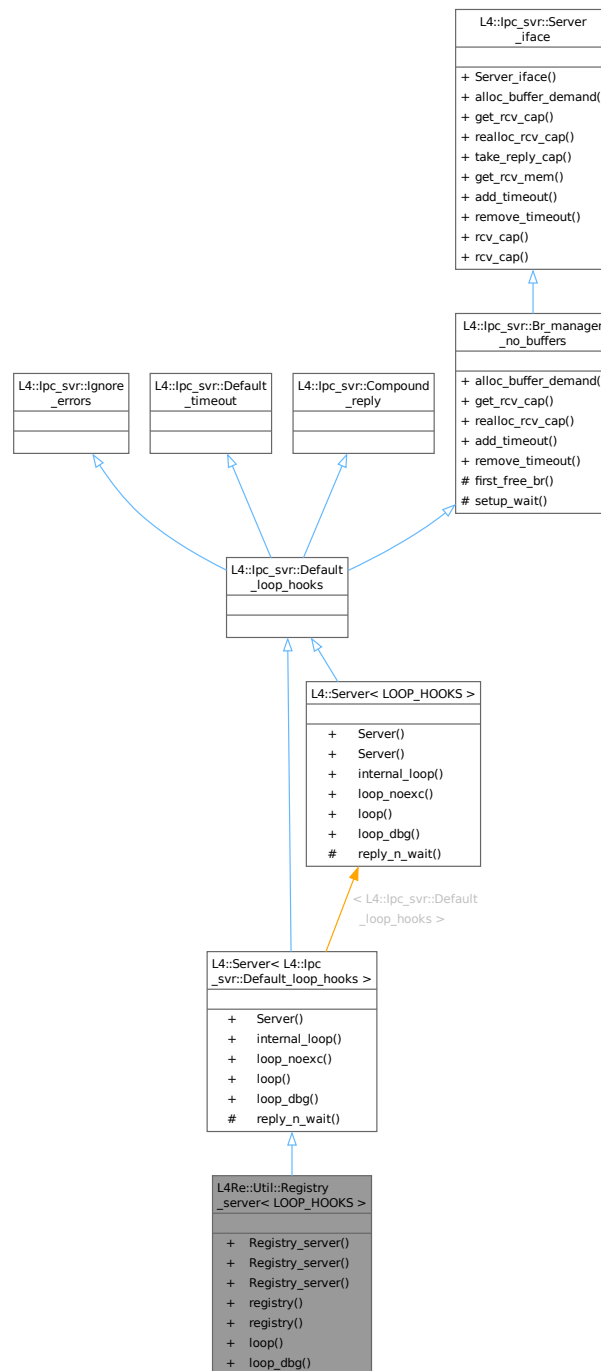
A server loop object which has a [Object_registry](#) included.

```
#include <object_registry>
```

Inheritance diagram for L4Re::Util::Registry_server< LOOP_HOOKS >:



Collaboration diagram for L4Re::Util::Registry_server< LOOP_HOOKS >:



Public Member Functions

- [Registry_server\(\)](#)
Create a new server loop object for the main thread of the task.
- [Registry_server](#) (`L4::utcb_t *`, `L4::Cap< L4::Thread >` server, `L4::Cap< L4::Factory >` factory)
Create a new server loop object for an arbitrary thread and factory.
- [Registry_server](#) (`L4::Cap< L4::Thread >` server, `L4::Cap< L4::Factory >` factory)

- *Create a new server loop object for an arbitrary thread and factory.*
- `Object_registry` const * **registry** () const
Return registry of this server loop.
- `Object_registry` * **registry** ()
Return registry of this server loop.
- void `L4_NORETURN loop` (`l4_utcb_t` *utcb=`l4_utcb`())
Start the server loop.
- template<typename Printer>
void `L4_NORETURN loop_dbg` (Printer printer, `l4_utcb_t` *utcb=`l4_utcb`())
Start the server loop with error printing.

Public Member Functions inherited from `L4::Server< L4::lpc_svr::Default_loop_hooks >`

- `Server` (`l4_utcb_t` *)
Initializes the server loop.
- `L4_NORETURN` void `internal_loop` (DISPATCH dispatch, `l4_utcb_t` *)
The server loop.
- `L4_NORETURN` void `loop_noexc` (R r, `l4_utcb_t` *u=`l4_utcb`())
Server loop without exception handling.
- `L4_NORETURN` void `loop` (R r, `l4_utcb_t` *u=`l4_utcb`())
Server loop with internal exception handling.
- `L4_NORETURN` void `loop_dbg` (R r, Printer p, `l4_utcb_t` *u=`l4_utcb`())
Server loop with internal exception handling including message printing.

Public Member Functions inherited from `L4::lpc_svr::Br_manager_no_buffers`

- int `alloc_buffer_demand` (`Demand` const &demand) override
Tells the server to allocate buffers for the given demand.
- `L4::Cap< void >` **get_rcv_cap** (int) const override
Returns `L4::Cap<void>::Invalid`, we have no buffer management.
- int **realloc_rcv_cap** (int) override
Returns -L4_ENOMEM, we have no buffer management.
- int **add_timeout** (`Timeout` *, `l4_kernel_clock_t`) override
Returns -L4_ENOSYS, we have no timeout queue.
- int **remove_timeout** (`Timeout` *) override
Returns -L4_ENOSYS, we have no timeout queue.

Public Member Functions inherited from `L4::lpc_svr::Server_iface`

- `Server_iface` ()
Make a server interface.
- virtual `cxx::Result< L4::Reply_cap >` `take_reply_cap` () noexcept
Take the currently used reply capability.
- virtual `cxx::Result< Mem_window >` `get_rcv_mem` () noexcept
Take the current memory receive window.
- template<typename T>
`L4::Cap< T >` `rcv_cap` (int index) const
Get given receive buffer as typed capability.
- `L4::Cap< void >` `rcv_cap` (int index) const
Get receive cap with the given index as generic (void) type.

Additional Inherited Members

Public Types inherited from [L4::lpc_svr::Server_iface](#)

- using **Demand** = [L4::Type_info::Demand](#)
Data type expressing server-side demand for receive buffers.

Protected Member Functions inherited from [L4::Server< L4::lpc_svr::Default_loop_hooks >](#)

- [l4_msgtag_t](#) **reply_n_wait** ([l4_msgtag_t](#) reply, [l4_umword_t](#) *p, [l4_utcb_t](#) *)
Internal implementation for reply and wait.

Protected Member Functions inherited from [L4::lpc_svr::Br_manager_no_buffers](#)

- unsigned **first_free_br** () const
Returns 1 as first free buffer.
- void **setup_wait** ([l4_utcb_t](#) *utcb, [L4::lpc_svr::Reply_mode](#))
Setup wait function for the server loop ([Server<>](#)).

15.335.1 Detailed Description

```
template<typename LOOP_HOOKS = L4::lpc_svr::Default_loop_hooks>
class L4Re::Util::Registry_server< LOOP_HOOKS >
```

A server loop object which has a [Object_registry](#) included.

Examples

[examples/clntsrv/src/server.cc](#), [examples/libs/l4re/c++/shared_ds/ds_srv.cc](#), and [examples/libs/l4re/streammap/server.cc](#).

Definition at line 293 of file [object_registry](#).

15.335.2 Constructor & Destructor Documentation

15.335.2.1 Registry_server() [1/3]

```
template<typename LOOP_HOOKS = L4::lpc_svr::Default_loop_hooks>
L4Re::Util::Registry_server< LOOP_HOOKS >::Registry_server () [inline]
```

Create a new server loop object for the main thread of the task.

Precondition

Must be called from the main thread or behaviour is undefined.

Definition at line 305 of file [object_registry](#).

15.335.2.2 Registry_server() [2/3]

```
template<typename LOOP_HOOKS = L4::Ipc_svr::Default_loop_hooks>
L4Re::Util::Registry_server< LOOP_HOOKS >::Registry_server (
    l4_utcb_t * ,
    L4::Cap< L4::Thread > server,
    L4::Cap< L4::Factory > factory) [inline]
```

Create a new server loop object for an arbitrary thread and factory.

Parameters

| | |
|----------------|--|
| <i>server</i> | Capability to thread running the server loop. |
| <i>factory</i> | Capability to factory object used to create new IPC gates. |

Deprecated Note that this variant of the constructor is deprecated, please do not supply the UTCB pointer, it's not used.

Definition at line 317 of file [object_registry](#).

15.335.2.3 Registry_server() [3/3]

```
template<typename LOOP_HOOKS = L4::Ipc_svr::Default_loop_hooks>
L4Re::Util::Registry_server< LOOP_HOOKS >::Registry_server (
    L4::Cap< L4::Thread > server,
    L4::Cap< L4::Factory > factory) [inline]
```

Create a new server loop object for an arbitrary thread and factory.

Parameters

| | |
|----------------|--|
| <i>server</i> | Capability to thread running the server loop. |
| <i>factory</i> | Capability to factory object used to create new IPC gates. |

Definition at line 328 of file [object_registry](#).

15.335.3 Member Function Documentation

15.335.3.1 loop()

```
template<typename LOOP_HOOKS = L4::Ipc_svr::Default_loop_hooks>
void L4_NORETURN L4Re::Util::Registry_server< LOOP_HOOKS >::loop (
    l4_utcb_t * utcb = l4_utcb()) [inline]
```

Start the server loop.

Parameters

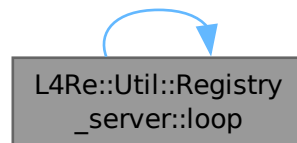
| | |
|-------------|---|
| <i>utcb</i> | The UTCB of the thread running the server loop, defaults to l4_utcb() . |
|-------------|---|

Definition at line 344 of file [object_registry](#).

References [L4_NORETURN](#), and [loop\(\)](#).

Referenced by [loop\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.335.3.2 `loop_dbg()`

```

template<typename LOOP_HOOKS = L4::Ipc_svr::Default_loop_hooks>
template<typename Printer>
void L4\_NORETURN L4Re::Util::Registry_server< LOOP_HOOKS >::loop_dbg (
    Printer printer,
    l4\_utcb\_t * utcb = l4\_utcb\(\)) [inline]
  
```

Start the server loop with error printing.

Template Parameters

| | |
|----------------|-------------------|
| <i>Printer</i> | The printer type. |
|----------------|-------------------|

Parameters

| | |
|----------------|---|
| <i>printer</i> | The printer object on which printf() is called. |
| <i>utcb</i> | The UTCB of the thread running the server loop, defaults to l4_utcb() . |

Definition at line [356](#) of file [object_registry](#).

References [L4_NORETURN](#), and [loop_dbg\(\)](#).

Referenced by [loop_dbg\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

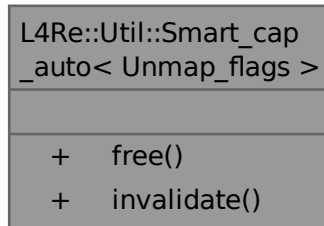
- [l4/re/util/object_registry](#)

15.336 L4Re::Util::Smart_cap_auto< Unmap_flags > Class Template Reference

Helper for [Unique_cap](#) and [Unique_del_cap](#).

```
#include <cap_alloc>
```

Collaboration diagram for L4Re::Util::Smart_cap_auto< Unmap_flags >:



Static Public Member Functions

- static void **free** ([L4::Cap_base](#) &c)
Free the provided capability.
- static void **invalidate** ([L4::Cap_base](#) &c)
Invalidate the provided capability.

15.336.1 Detailed Description

```
template<unsigned long Unmap_flags = L4_FP_ALL_SPACES>
class L4Re::Util::Smart_cap_auto< Unmap_flags >
```

Helper for [Unique_cap](#) and [Unique_del_cap](#).

Definition at line 87 of file [cap_alloc](#).

The documentation for this class was generated from the following file:

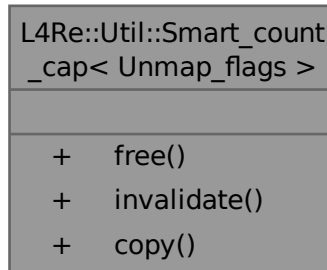
- [l4/re/util/cap_alloc](#)

15.337 L4Re::Util::Smart_count_cap< Unmap_flags > Class Template Reference

Helper for [Ref_cap](#) and [Ref_del_cap](#).

```
#include <cap_alloc>
```

Collaboration diagram for L4Re::Util::Smart_count_cap< Unmap_flags >:



Static Public Member Functions

- static void **free** ([L4::Cap_base](#) &c) noexcept
Free operation for [L4::Smart_cap](#) (decrement ref count and delete if 0).
- static void **invalidate** ([L4::Cap_base](#) &c) noexcept
Invalidate operation for [L4::Smart_cap](#).
- static [L4::Cap_base](#) **copy** ([L4::Cap_base](#) const &src)
Copy operation for [L4::Smart_cap](#) (increment ref count).

15.337.1 Detailed Description

```
template<unsigned long Unmap_flags = L4_FP_ALL_SPACES>
class L4Re::Util::Smart_count_cap< Unmap_flags >
```

Helper for [Ref_cap](#) and [Ref_del_cap](#).

Definition at line 118 of file [cap_alloc](#).

The documentation for this class was generated from the following file:

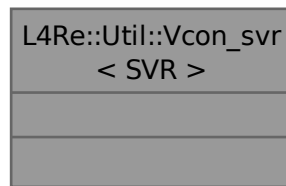
- [l4/re/util/cap_alloc](#)

15.338 L4Re::Util::Vcon_svr< SVR > Class Template Reference

[Console](#) server template class.

```
#include <vcon_svr>
```


Collaboration diagram for L4Re::Util::Vcon_svr< SVR >:



15.338.1 Detailed Description

```
template<typename SVR>
class L4Re::Util::Vcon_svr< SVR >
```

[Console](#) server template class.

This template uses `vcon_write()` and `vcon_read()` to get and deliver data from the implementor.

`vcon_read()` needs to update the status argument with the `L4_vcon_read_stat` flags, especially the `L4_VCON_READ_STAT_DONE` flag to indicate that there's nothing more to read for the other end.

`vcon_write()` gets the live data from the UTCB. Make sure to copy out the data before using the UTCB again.

The size parameter of both functions is given in bytes.

Definition at line 36 of file [vcon_svr](#).

The documentation for this class was generated from the following file:

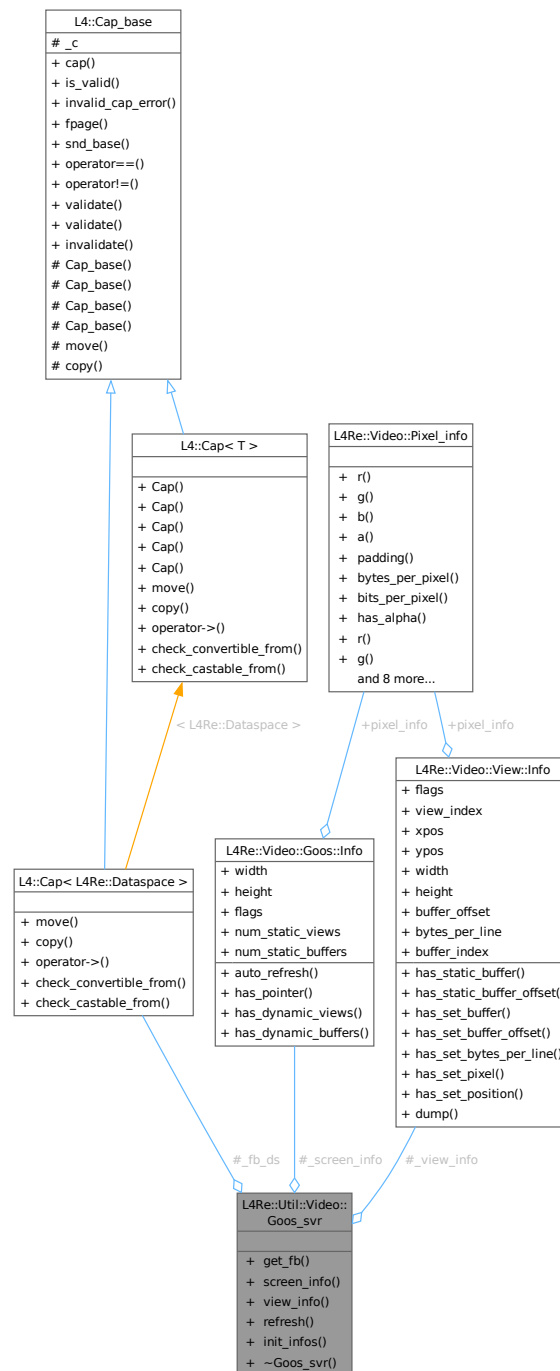
- `l4/re/util/vcon_svr`

15.339 L4Re::Util::Video::Goos_svr Class Reference

Goos server class.

```
#include <goos_svr>
```

Collaboration diagram for L4Re::Util::Video::Goos_svr:



Public Member Functions

- **L4::Cap< L4Re::Dataspace >** `get_fb ()` const
Return framebuffer memory dataspace.
- **L4Re::Video::Goos::Info** const * `screen_info ()` const
Goos information structure.
- **L4Re::Video::View::Info** const * `view_info ()` const

View information structure.

- virtual int [refresh](#) (int x, int y, int w, int h)

Refresh area of the framebuffer.

- void [init_infos](#) ()

Initialize the view information structure of this object.

- virtual \sim [Goos_svr](#) ()

Destructor.

Protected Attributes

- [L4::Cap](#)< [L4Re::Dataspace](#) > [_fb_ds](#)

Goos memory dataspace.

- [L4Re::Video::Goos::Info](#) [_screen_info](#)

Goos information.

- [L4Re::Video::View::Info](#) [_view_info](#)

View information.

15.339.1 Detailed Description

Goos server class.

Definition at line 25 of file [goos_svr](#).

15.339.2 Member Function Documentation

15.339.2.1 [get_fb\(\)](#)

```
L4::Cap< L4Re::Dataspace > L4Re::Util::Video::Goos\_svr::get\_fb () const [inline]
```

Return framebuffer memory dataspace.

Returns

Goos memory dataspace

Definition at line 42 of file [goos_svr](#).

References [_fb_ds](#).

15.339.2.2 [init_infos\(\)](#)

```
void L4Re::Util::Video::Goos\_svr::init\_infos () [inline]
```

Initialize the view information structure of this object.

This function initializes the view info structure of this goos object based on the information in the goos information, i.e. the width, height and pixel_info of the goos information has to contain valid values before calling [init_info\(\)](#).

Definition at line 78 of file [goos_svr](#).

References [_screen_info](#), and [_view_info](#).

15.339.2.3 refresh()

```
virtual int L4Re::Util::Video::Goos_svr::refresh (  
    int x,  
    int y,  
    int w,  
    int h) [inline], [virtual]
```

Refresh area of the framebuffer.

Parameters

| | |
|----------|--------------------------|
| <i>x</i> | X coordinate (pixels) |
| <i>y</i> | Y coordinate (pixels) |
| <i>w</i> | Width of area in pixels |
| <i>h</i> | Height of area in pixels |

Returns

0 on success, negative error code otherwise

Definition at line 66 of file [goos_svr](#).

References [L4_ENOSYS](#).

15.339.2.4 screen_info()

```
L4Re::Video::Goos::Info const * L4Re::Util::Video::Goos_svr::screen_info () const [inline]
```

Goos information structure.

Returns

Return goos information structure.

Definition at line 48 of file [goos_svr](#).

References [_screen_info](#).

15.339.2.5 view_info()

```
L4Re::Video::View::Info const * L4Re::Util::Video::Goos_svr::view_info () const [inline]
```

View information structure.

Returns

Return view information structure.

Definition at line 54 of file [goos_svr](#).

References [_view_info](#).

The documentation for this class was generated from the following file:

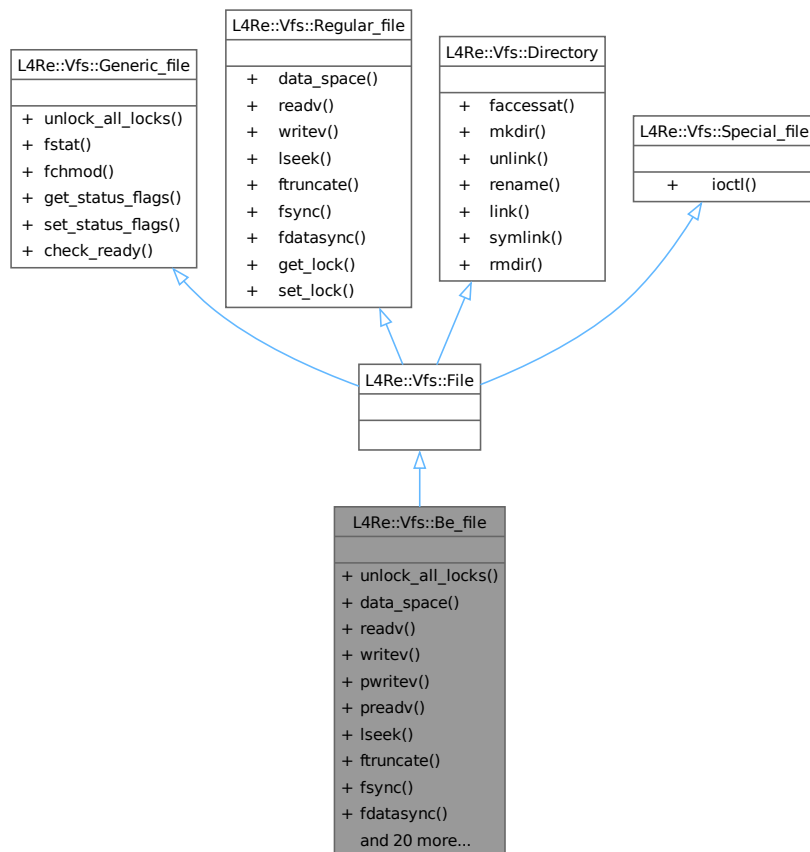
- [l4/re/util/video/goos_svr](#)

15.340 L4Re::Vfs::Be_file Class Reference

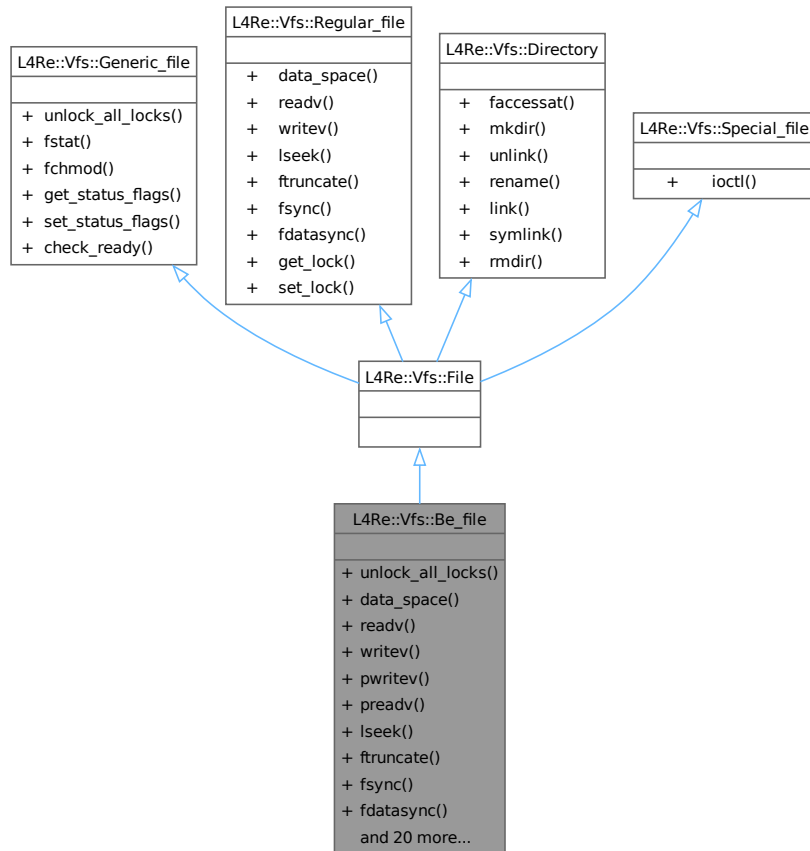
Boiler plate class for implementing an open file for [L4Re::Vfs](#).

```
#include <backend>
```

Inheritance diagram for L4Re::Vfs::Be_file:



Collaboration diagram for L4Re::Vfs::Be_file:



Public Member Functions

- `int unlock_all_locks ()` noexcept override
Unlock all locks on the file.
- `L4::Cap< L4Re::Dataspace > data_space ()` noexcept override
Get an `L4Re::Dataspace` object for the file.
- `ssize_t readv (const struct iovec *, int)` noexcept override
Default backend for POSIX read and readv functions.
- `ssize_t writev (const struct iovec *, int)` noexcept override
Default backend for POSIX write and writev functions.
- `ssize_t pwritev (const struct iovec *, int, off64_t)` noexcept override
Default backend for POSIX pwrite and pwritev functions.
- `ssize_t preadv (const struct iovec *, int, off64_t)` noexcept override
Default backend for POSIX pread and preadv functions.
- `off64_t lseek (off64_t, int)` noexcept override
Default backend for POSIX seek and lseek functions.
- `int ftruncate (off64_t)` noexcept override
Default backend for the POSIX truncate, ftruncate and similar functions.
- `int fsync ()` const noexcept override
Default backend for POSIX fsync.

- int **fdatasync** () const noexcept override
Default backend for POSIX fdatasync.
- int **ioctl** (unsigned long, va_list) noexcept override
Default backend for POSIX ioctl.
- int **fstat** (struct stat64 *) const noexcept override
Get status information for the file.
- int **fchmod** (mode_t) noexcept override
Default backend for POSIX chmod and fchmod.
- int **get_status_flags** () const noexcept override
Default backend for POSIX fcntl subfunctions.
- int **set_status_flags** (long) noexcept override
Default backend for POSIX fcntl subfunctions.
- int **get_lock** (struct flock64 *) noexcept override
Default backend for POSIX fcntl subfunctions.
- int **set_lock** (struct flock64 *, bool) noexcept override
Default backend for POSIX fcntl subfunctions.
- int **faccessat** (const char *, int, int) noexcept override
Default backend for POSIX access and faccessat functions.
- int **fchmodat** (const char *, mode_t, int) noexcept override
Default backend for POSIX fchmodat function.
- int **utime** (const struct utimbuf *) noexcept override
Default backend for POSIX utime.
- int **utimes** (const struct timeval[2]) noexcept override
Default backend for POSIX utimes.
- int **utimensat** (const char *, const struct timespec[2], int) noexcept override
Default backend for POSIX utimensat.
- int **mkdir** (const char *, mode_t) noexcept override
Default backend for POSIX mkdir and mkdirat.
- int **unlink** (const char *) noexcept override
Default backend for POSIX unlink, unlinkat.
- int **rename** (const char *, const char *) noexcept override
Default backend for POSIX rename, renameat.
- int **link** (const char *, const char *) noexcept override
Default backend for POSIX link, linkat.
- int **symlink** (const char *, const char *) noexcept override
Default backend for POSIX symlink, symlinkat.
- int **rmdir** (const char *) noexcept override
Default backend for POSIX rmdir, rmdirat.
- ssize_t **readlink** (char *, size_t) override
Default backend for POSIX readlink, readlinkat.
- bool **check_ready** (Ready_type) noexcept override
Default implementation of a readiness check.

Additional Inherited Members

Public Types inherited from L4Re::Vfs::Generic_file

- enum **Ready_type** : unsigned
Type of I/O operation/condition a file can indicate readiness.

15.340.1 Detailed Description

Boiler plate class for implementing an open file for [L4Re::Vfs](#).

This class may be used as a base class for everything that a POSIX file descriptor may point to. This are things such as regular files, directories, special device files, streams, pipes, and so on.

Examples

[tmpfs/lib/src/fs.cc](#).

Definition at line 30 of file [backend](#).

15.340.2 Member Function Documentation

15.340.2.1 `check_ready()`

```
bool L4Re::Vfs::Be_file::check_ready (
    Ready_type ) [inline], [override], [virtual], [noexcept]
```

Default implementation of a readiness check.

By default, we assume a file is not ready for an I/O operation/condition since the proper semantics of that relies on the backend.

Returns

Always false.

Implements [L4Re::Vfs::Generic_file](#).

Definition at line 217 of file [backend](#).

15.340.2.2 `data_space()`

```
L4::Cap< L4Re::Dataspace > L4Re::Vfs::Be_file::data_space () [inline], [override], [virtual],
[noexcept]
```

Get an [L4Re::Dataspace](#) object for the file.

This is used as a backend for POSIX mmap and mmap2 functions.

Note

mmap is not possible if the function returns an invalid capability.

Returns

A capability to an [L4Re::Dataspace](#) that represents the file contents in an [L4Re](#) way.

Implements [L4Re::Vfs::Regular_file](#).

Definition at line 47 of file [backend](#).

References [L4::Cap_base::Invalid](#).

15.340.2.3 fstat()

```
int L4Re::Vfs::Be_file::fstat (
    struct stat64 * buf) const [inline], [override], [virtual], [noexcept]
```

Get status information for the file.

This is the backend for POSIX fstat, stat, fstat64 and friends.

Parameters

| | | |
|-----|-----|--|
| out | buf | This buffer is filled with the status information. |
|-----|-----|--|

Returns

0 on success, or <0 on error.

Implements [L4Re::Vfs::Generic_file](#).

Definition at line 86 of file [backend](#).

15.340.2.4 unlock_all_locks()

```
int L4Re::Vfs::Be_file::unlock_all_locks () [inline], [override], [virtual], [noexcept]
```

Unlock all locks on the file.

Note

All locks means all locks independent of which file the locks were taken by.

This method is called by the POSIX close implementation to get the POSIX semantics of releasing all locks taken by this application on a close for any fd referencing the real file.

Returns

0 on success, or <0 on error.

Implements [L4Re::Vfs::Generic_file](#).

Definition at line 43 of file [backend](#).

The documentation for this class was generated from the following file:

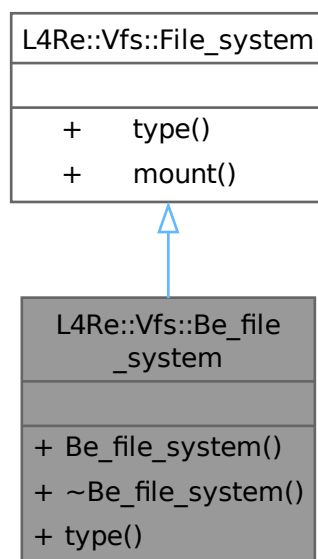
- l4/l4re_vfs/backend

15.341 L4Re::Vfs::Be_file_system Class Reference

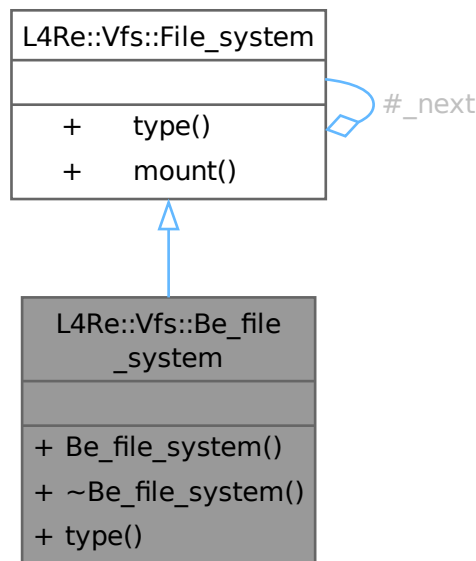
Boilerplate class for implementing a [L4Re::Vfs::File_system](#).

```
#include <backend>
```

Inheritance diagram for L4Re::Vfs::Be_file_system:



Collaboration diagram for L4Re::Vfs::Be_file_system:



Public Member Functions

- [Be_file_system](#) (char const *fstype) noexcept
Create a file-system object for the given fstype.
- [~Be_file_system](#) () noexcept
Destroy a file-system object.
- char const * [type](#) () const noexcept override
Return the file-system type.

Public Member Functions inherited from L4Re::Vfs::File_system

- virtual int [mount](#) (char const *source, unsigned long mountflags, void const *data, [cxx::Ref_ptr](#)< [File](#) > *dir) noexcept=0
Create a directory object dir representing source mounted with this file system.

15.341.1 Detailed Description

Boilerplate class for implementing a [L4Re::Vfs::File_system](#).

This class already takes care of registering and unregistering the file system in the global registry and implements the [type\(\)](#) method.

Examples

[tmpfs/lib/src/fs.cc](#).

Definition at line 310 of file [backend](#).

15.341.2 Constructor & Destructor Documentation

15.341.2.1 `Be_file_system()`

```
L4Re::Vfs::Be_file_system::Be_file_system (
    char const * fstype)    [inline], [explicit], [noexcept]
```

Create a file-system object for the given *fstype*.

Parameters

| | |
|---------------|--|
| <i>fstype</i> | The type that type() shall return. |
|---------------|--|

This constructor takes care of registering the file system in the registry of `L4Re::Vfs::vfs_ops`.

Definition at line 324 of file [backend](#).

15.341.2.2 `~Be_file_system()`

```
L4Re::Vfs::Be_file_system::~~Be_file_system ()    [inline], [noexcept]
```

Destroy a file-system object.

This destructor takes care of removing this file system from the registry of `L4Re::Vfs::vfs_ops`.

Definition at line 336 of file [backend](#).

15.341.3 Member Function Documentation

15.341.3.1 `type()`

```
char const * L4Re::Vfs::Be_file_system::type () const    [inline], [override], [virtual], [noexcept]
```

Return the file-system type.

Returns the file-system type given as *fstype* in the constructor.

Implements [L4Re::Vfs::File_system](#).

Definition at line 346 of file [backend](#).

The documentation for this class was generated from the following file:

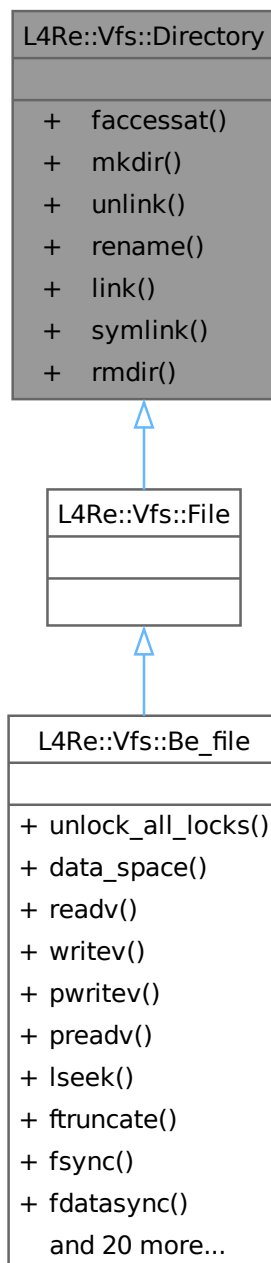
- `l4/l4re_vfs/backend`

15.342 L4Re::Vfs::Directory Class Reference

Interface for a POSIX file that is a directory.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::Directory:



Collaboration diagram for L4Re::Vfs::Directory:

| L4Re::Vfs::Directory | |
|----------------------|-------------|
| | |
| + | faccessat() |
| + | mkdir() |
| + | unlink() |
| + | rename() |
| + | link() |
| + | symlink() |
| + | rmdir() |

Public Member Functions

- virtual int [faccessat](#) (const char *path, int mode, int flags) noexcept=0
Check access permissions on the given file.
- virtual int [mkdir](#) (const char *path, mode_t mode) noexcept=0
Create a new subdirectory.
- virtual int [unlink](#) (const char *path) noexcept=0
Unlink the given file from that directory.
- virtual int [rename](#) (const char *src_path, const char *dst_path) noexcept=0
Rename the given file.
- virtual int [link](#) (const char *src_path, const char *dst_path) noexcept=0
Create a hard link (second name) for the given file.
- virtual int [symlink](#) (const char *src_path, const char *dst_path) noexcept=0
Create a symbolic link for the given file.
- virtual int [rmdir](#) (const char *path) noexcept=0
Delete an empty directory.

15.342.1 Detailed Description

Interface for a POSIX file that is a directory.

This interface provides functionality for directory files in the [L4Re::Vfs](#). However, real objects always use the combined [L4Re::Vfs::File](#) interface.

Definition at line 160 of file [vfs.h](#).

15.342.2 Member Function Documentation

15.342.2.1 faccessat()

```
virtual int L4Re::Vfs::Directory::faccessat (  
    const char * path,  
    int mode,  
    int flags) [pure virtual], [noexcept]
```

Check access permissions on the given file.

Backend function for POSIX access and faccessat functions.

Parameters

| | |
|--------------|--|
| <i>path</i> | The path relative to this directory. Note: <i>path</i> is relative to this directory and may contain subdirectories. |
| <i>mode</i> | The access mode to check. |
| <i>flags</i> | The flags as in POSIX faccessat (AT_EACCESS, AT_SYMLINK_NOFOLLOW). |

Returns

0 on success, or <0 on error.

Implemented in [L4Re::Vfs::Be_file](#).

15.342.2.2 link()

```
virtual int L4Re::Vfs::Directory::link (  
    const char * src_path,  
    const char * dst_path) [pure virtual], [noexcept]
```

Create a hard link (second name) for the given file.

Backend for the POSIX link and linkat functions.

Parameters

| | |
|-----------------|---|
| <i>src_path</i> | The old name of the file. Note: <i>src_path</i> is relative to this directory and may contain subdirectories. |
| <i>dst_path</i> | The new (second) name for the file. Note: <i>dst_path</i> is relative to this directory and may contain subdirectories. |

Returns

0 on success, or <0 on error.

Implemented in [L4Re::Vfs::Be_file](#).

15.342.2.3 mkdir()

```
virtual int L4Re::Vfs::Directory::mkdir (  
    const char * path,  
    mode_t mode)    [pure virtual], [noexcept]
```

Create a new subdirectory.

Backend for POSIX mkdir and mkdirat function calls.

Parameters

| | |
|-------------|---|
| <i>path</i> | The name of the subdirectory to create. Note: <i>path</i> is relative to this directory and may contain subdirectories. |
| <i>mode</i> | The file mode to use for the new directory. |

Returns

0 on success, or <0 on error. -ENOTDIR if this or some component in path is not a directory.

Implemented in [L4Re::Vfs::Be_file](#).

15.342.2.4 rename()

```
virtual int L4Re::Vfs::Directory::rename (  
    const char * src_path,  
    const char * dst_path)    [pure virtual], [noexcept]
```

Rename the given file.

Backend for the POSIX rename, renameat functions.

Parameters

| | |
|-----------------|---|
| <i>src_path</i> | The old name of the file to rename. Note: <i>src_path</i> is relative to this directory and may contain subdirectories. |
| <i>dst_path</i> | The new name for the file. Note: <i>dst_path</i> is relative to this directory and may contain subdirectories. |

Returns

0 on success, or <0 on error.

Implemented in [L4Re::Vfs::Be_file](#).

15.342.2.5 rmdir()

```
virtual int L4Re::Vfs::Directory::rmdir (  
    const char * path) [pure virtual], [noexcept]
```

Delete an empty directory.

Backend for POSIX rmdir, rmdirat functions.

Parameters

| | |
|-------------|--|
| <i>path</i> | The name of the directory to remove. Note: <i>path</i> is relative to this directory and may contain subdirectories. |
|-------------|--|

Returns

0 on success, or <0 on error.

Implemented in [L4Re::Vfs::Be_file](#).

15.342.2.6 symlink()

```
virtual int L4Re::Vfs::Directory::symlink (  
    const char * src_path,  
    const char * dst_path) [pure virtual], [noexcept]
```

Create a symbolic link for the given file.

Backend for the POSIX symlink and symlinkat functions.

Parameters

| | |
|-----------------|---|
| <i>src_path</i> | The old name of the file. Note: <i>src_path</i> shall be an absolute path. |
| <i>dst_path</i> | The name for symlink. Note: <i>dst_path</i> is relative to this directory and may contain subdirectories. |

Returns

0 on success, or <0 on error.

Implemented in [L4Re::Vfs::Be_file](#).

15.342.2.7 unlink()

```
virtual int L4Re::Vfs::Directory::unlink (
    const char * path) [pure virtual], [noexcept]
```

Unlink the given file from that directory.

Backend for the POSIX unlink and unlinkat functions.

Parameters

| | |
|-------------|---|
| <i>path</i> | The name of the file to unlink. Note: <i>path</i> is relative to this directory and may contain subdirectories. |
|-------------|---|

Returns

0 on success, or <0 on error.

Implemented in [L4Re::Vfs::Be_file](#).

The documentation for this class was generated from the following file:

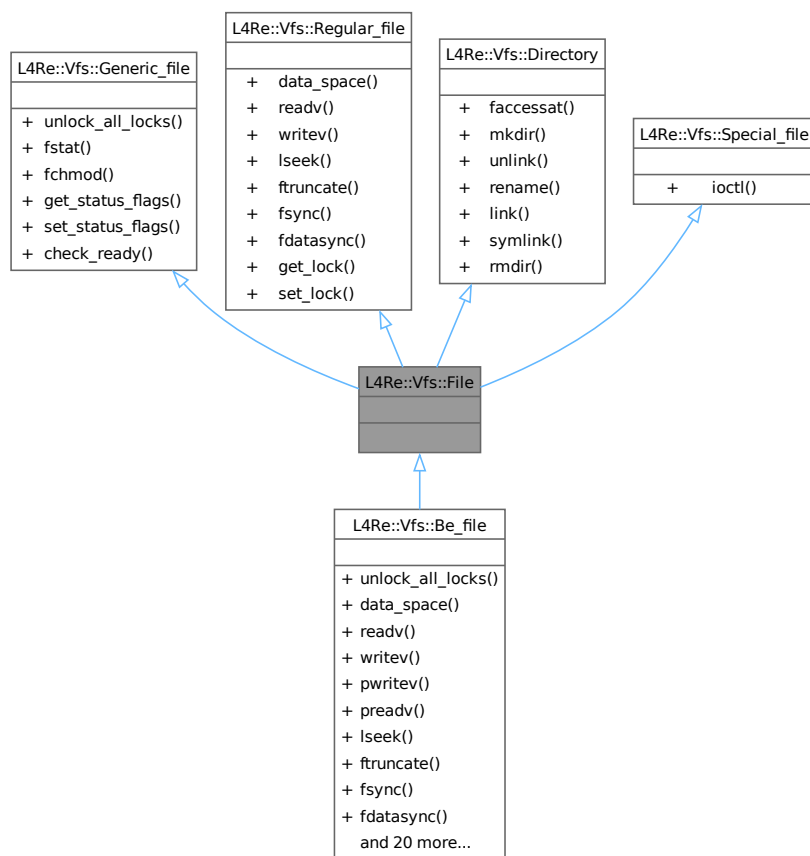
- l4/l4re_vfs/vfs.h

15.343 L4Re::Vfs::File Class Reference

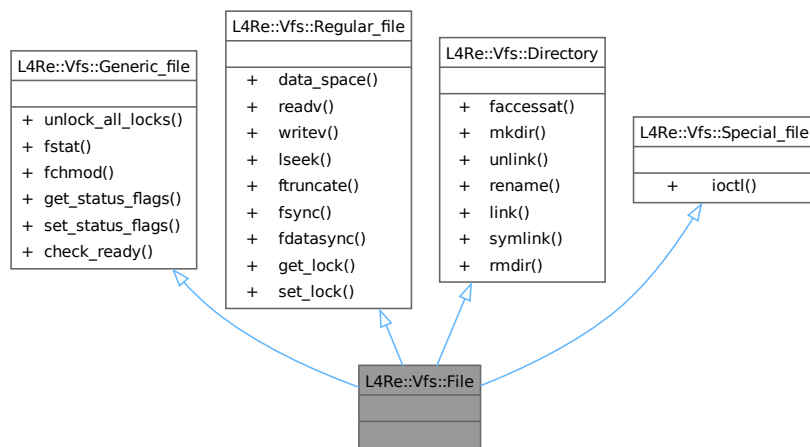
The basic interface for an open POSIX file.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::File:



Collaboration diagram for L4Re::Vfs::File:



Additional Inherited Members

Public Types inherited from [L4Re::Vfs::Generic_file](#)

- enum [Ready_type](#) : unsigned
Type of I/O operation/condition a file can indicate readiness.

Public Member Functions inherited from [L4Re::Vfs::Generic_file](#)

- virtual int [unlock_all_locks](#) () noexcept=0
Unlock all locks on the file.
- virtual int [fstat](#) (struct stat64 *buf) const noexcept=0
Get status information for the file.
- virtual int [fchmod](#) (mode_t) noexcept=0
Change POSIX access rights on the file.
- virtual int [get_status_flags](#) () const noexcept=0
Get file status flags (fcntl F_GETFL).
- virtual int [set_status_flags](#) (long flags) noexcept=0
Set file status flags (fcntl F_SETFL).
- virtual bool [check_ready](#) ([Ready_type](#) rt) noexcept=0
Check whether the file is ready for an I/O operation/condition.

Public Member Functions inherited from [L4Re::Vfs::Regular_file](#)

- virtual [L4::Cap](#)< [L4Re::Dataspace](#) > [data_space](#) () noexcept=0
Get an [L4Re::Dataspace](#) object for the file.
- virtual ssize_t [readv](#) (const struct iovec *, int iovcnt) noexcept=0
Read one or more blocks of data from the file.
- virtual ssize_t [writev](#) (const struct iovec *, int iovcnt) noexcept=0
Write one or more blocks of data to the file.
- virtual off64_t [lseek](#) (off64_t, int) noexcept=0
Change the file pointer.
- virtual int [ftruncate](#) (off64_t pos) noexcept=0
Truncate the file at the given position.
- virtual int [fsync](#) () const noexcept=0
Sync the data and meta data to persistent storage.
- virtual int [fdatasync](#) () const noexcept=0
Sync the data to persistent storage.
- virtual int [get_lock](#) (struct flock64 *lock) noexcept=0
Test if the given lock can be placed in the file.
- virtual int [set_lock](#) (struct flock64 *lock, bool wait) noexcept=0
Acquire or release the given lock on the file.

Public Member Functions inherited from L4Re::Vfs::Directory

- virtual int [faccessat](#) (const char *path, int mode, int flags) noexcept=0
Check access permissions on the given file.
- virtual int [mkdir](#) (const char *path, mode_t mode) noexcept=0
Create a new subdirectory.
- virtual int [unlink](#) (const char *path) noexcept=0
Unlink the given file from that directory.
- virtual int [rename](#) (const char *src_path, const char *dst_path) noexcept=0
Rename the given file.
- virtual int [link](#) (const char *src_path, const char *dst_path) noexcept=0
Create a hard link (second name) for the given file.
- virtual int [symlink](#) (const char *src_path, const char *dst_path) noexcept=0
Create a symbolic link for the given file.
- virtual int [rmdir](#) (const char *path) noexcept=0
Delete an empty directory.

Public Member Functions inherited from L4Re::Vfs::Special_file

- virtual int [ioctl](#) (unsigned long cmd, va_list args) noexcept=0
The famous IO control.

15.343.1 Detailed Description

The basic interface for an open POSIX file.

An open POSIX file can be anything that hides behind a POSIX file descriptor. This means that even directories are files. An open file can be anything from a directory to a special device file so see [Generic_file](#), [Regular_file](#), [Directory](#), and [Special_file](#) for more information.

Note

For implementing a backend for the [L4Re::Vfs](#) [L4Re::Vfs::Be_file](#) may be used as a base class.

Definition at line 455 of file [vfs.h](#).

The documentation for this class was generated from the following file:

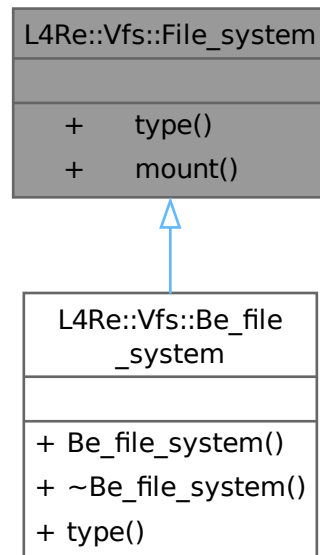
- [l4/l4re_vfs/vfs.h](#)

15.344 L4Re::Vfs::File_system Class Reference

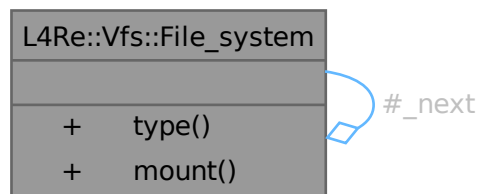
Basic interface for an [L4Re::Vfs](#) file system.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::File_system:



Collaboration diagram for L4Re::Vfs::File_system:



Public Member Functions

- virtual char const * `type()` const noexcept=0
Returns the type of the file system used in mount as fstype argument.
- virtual int `mount` (char const *source, unsigned long mountflags, void const *data, [cxx::Ref_ptr< File >](#) *dir) noexcept=0
Create a directory object dir representing source mounted with this file system.

15.344.1 Detailed Description

Basic interface for an [L4Re::Vfs](#) file system.

Note

For implementing a special file system [L4Re::Vfs::Be_file_system](#) may be used as a base class.

The main purpose of this interface is to have a single object for each supported file-system type (e.g., ext2, vfat) that exists in the application and is registered at the [L4Re::Vfs::Fs](#) singleton available via [L4Re::Vfs::vfs_ops](#). Ultimately, the POSIX mount function calls the [File_system::mount](#) method matching the file-system type given in mount.

Definition at line 857 of file [vfs.h](#).

15.344.2 Member Function Documentation

15.344.2.1 mount()

```
virtual int L4Re::Vfs::File_system::mount (
    char const * source,
    unsigned long mountflags,
    void const * data,
    cxx::Ref_ptr< File > * dir) [pure virtual], [noexcept]
```

Create a directory object *dir* representing *source* mounted with this file system.

Parameters

| | | |
|-----|-------------------|---|
| | <i>source</i> | The path to the source device to mount. This may also be some URL or anything file-system specific. |
| | <i>mountflags</i> | The mount flags as specified in the POSIX mount call. |
| | <i>data</i> | The data as specified in the POSIX mount call. The contents are file-system specific. |
| out | <i>dir</i> | A new directory object representing the file-system root directory. |

Returns

0 on success, and <0 on error (e.g. -EINVAL).

References [mount\(\)](#).

Referenced by [mount\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.344.2.2 `type()`

```
virtual char const * L4Re::Vfs::File_system::type () const [pure virtual], [noexcept]
```

Returns the type of the file system used in mount as fstype argument.

Note

This method is already provided by [Be_file_system](#).

Implemented in [L4Re::Vfs::Be_file_system](#).

The documentation for this class was generated from the following file:

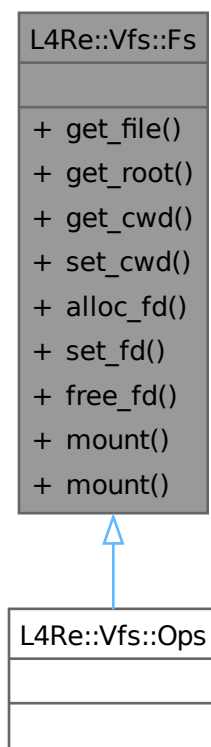
- `l4/l4re_vfs/vfs.h`

15.345 L4Re::Vfs::Fs Class Reference

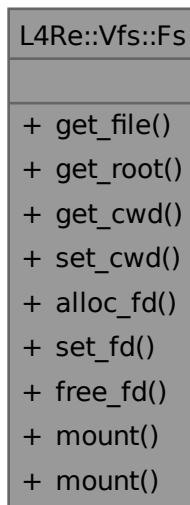
POSIX File-system related functionality.

```
#include <vfs.h>
```


Inheritance diagram for L4Re::Vfs::Fs:



Collaboration diagram for L4Re::Vfs::Fs:



Public Member Functions

- virtual `cxx::Ref_ptr< File > get_file (int fd)` noexcept=0
Get the L4Re::Vfs::File for the file descriptor fd.
- virtual `cxx::Ref_ptr< File > get_root ()` noexcept=0
Get the directory object for the application's root directory.
- virtual `cxx::Ref_ptr< File > get_cwd ()` noexcept
Get the directory object for the application's current working directory.
- virtual void `set_cwd (cxx::Ref_ptr< File > const &)` noexcept
Set the current working directory for the application.
- virtual int `alloc_fd (cxx::Ref_ptr< File > const &f=cxx::Ref_ptr<>::Nil)` noexcept=0
Allocate the next free file descriptor.
- virtual `cxx::Pair< cxx::Ref_ptr< File >, int > set_fd (int fd, cxx::Ref_ptr< File > const &f=cxx::Ref_ptr<>::Nil)` noexcept=0
Set the file object referenced by the file descriptor fd.
- virtual `cxx::Ref_ptr< File > free_fd (int fd)` noexcept=0
Free the file descriptor fd.
- virtual int `mount (char const *path, cxx::Ref_ptr< File > const &dir)` noexcept=0
Mount a given file object at the given global path in the VFS.
- int `mount (char const *source, char const *target, char const *fstype, unsigned long mountflags, void const *data)` noexcept
Backend for the POSIX mount call.

15.345.1 Detailed Description

POSIX File-system related functionality.

Note

This class usually exists as a singleton and as a superclass of [L4Re::Vfs::Ops](#) (

See also

[L4Re::Vfs::vfs_ops](#)).

Definition at line [946](#) of file [vfs.h](#).

15.345.2 Member Function Documentation

15.345.2.1 alloc_fd()

```
virtual int L4Re::Vfs::Fs::alloc_fd (  
    cxx::Ref\_ptr< File > const & f = cxx::Ref\_ptr<>::Nil)  [pure virtual], [noexcept]
```

Allocate the next free file descriptor.

Parameters

| | |
|----------|---|
| <i>f</i> | The file to assign to that file descriptor. |
|----------|---|

Returns

The allocated file descriptor, or -EMFILE on error.

15.345.2.2 free_fd()

```
virtual cxx::Ref\_ptr< File > L4Re::Vfs::Fs::free_fd (  
    int fd)  [pure virtual], [noexcept]
```

Free the file descriptor *fd*.

Parameters

| | |
|-----------|------------------------------|
| <i>fd</i> | The file descriptor to free. |
|-----------|------------------------------|

Returns

A pointer to the file object that was assigned to the fd.

15.345.2.3 `get_file()`

```
virtual cxx::Ref_ptr< File > L4Re::Vfs::Fs::get_file (
    int fd) [pure virtual], [noexcept]
```

Get the [L4Re::Vfs::File](#) for the file descriptor *fd*.

Parameters

| | |
|-----------|-----------------------------------|
| <i>fd</i> | The POSIX file descriptor number. |
|-----------|-----------------------------------|

Returns

A pointer to the [File](#) object, or 0 if *fd* is not open.

15.345.2.4 `mount()`

```
virtual int L4Re::Vfs::Fs::mount (
    char const * path,
    cxx::Ref_ptr< File > const & dir) [pure virtual], [noexcept]
```

Mount a given file object at the given global path in the VFS.

Parameters

| | |
|-------------|---|
| <i>path</i> | The global path to mount <i>dir</i> at. |
| <i>dir</i> | A pointer to the file/directory object that shall be mounted at <i>path</i> . |

Returns

0 on success, or <0 on error.

References [mount\(\)](#).

Referenced by [mount\(\)](#), and [mount\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.345.2.5 set_fd()

```
virtual cxx::Pair< cxx::Ref_ptr< File >, int > L4Re::Vfs::Fs::set_fd (
    int fd,
    cxx::Ref_ptr< File > const & f = cxx::Ref_ptr<>::Nil) [pure virtual], [noexcept]
```

Set the file object referenced by the file descriptor *fd*.

Parameters

| | |
|-----------|--|
| <i>fd</i> | The file descriptor to set to <i>f</i> . |
| <i>f</i> | The file object to assign. |

Returns

A pair of a pointer to the file object that was previously assigned to *fd* (*first*) and a return value (*second*). *second* contains `-#EBADF` if the passed file descriptor is outside the valid range. *first* contains a Nil pointer in that case. On success, *second* contains 0.

The documentation for this class was generated from the following files:

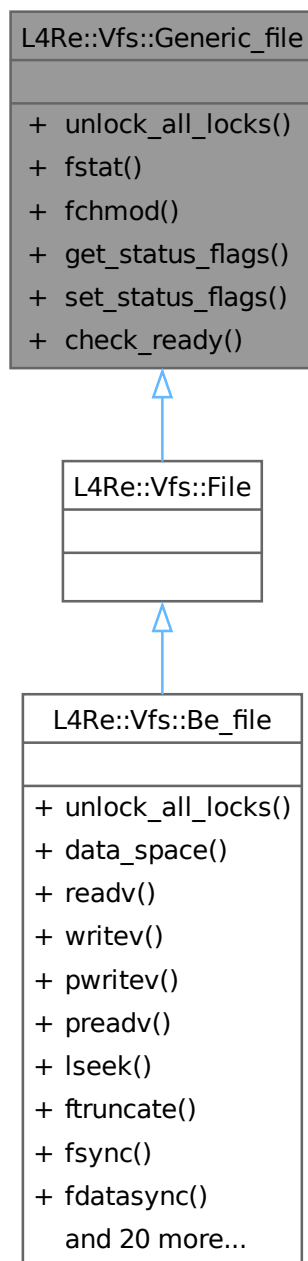
- l4/l4re_vfs/vfs.h
- l4/l4re_vfs/impl/vfs_impl.h

15.346 L4Re::Vfs::Generic_file Class Reference

The common interface for an open POSIX file.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::Generic_file:



Collaboration diagram for L4Re::Vfs::Generic_file:

| L4Re::Vfs::Generic_file |
|--|
| <ul style="list-style-type: none"> + unlock_all_locks() + fstat() + fchmod() + get_status_flags() + set_status_flags() + check_ready() |

Public Types

- enum [Ready_type](#) : unsigned
Type of I/O operation/condition a file can indicate readiness.

Public Member Functions

- virtual int [unlock_all_locks](#) () noexcept=0
Unlock all locks on the file.
- virtual int [fstat](#) (struct stat64 *buf) const noexcept=0
Get status information for the file.
- virtual int [fchmod](#) (mode_t) noexcept=0
Change POSIX access rights on the file.
- virtual int [get_status_flags](#) () const noexcept=0
Get file status flags (fcntl F_GETFL).
- virtual int [set_status_flags](#) (long flags) noexcept=0
Set file status flags (fcntl F_SETFL).
- virtual bool [check_ready](#) ([Ready_type](#) rt) noexcept=0
Check whether the file is ready for an I/O operation/condition.

15.346.1 Detailed Description

The common interface for an open POSIX file.

This interface is common to all kinds of open files, independent of the file type (e.g., directory, regular file etc.). However, in the [L4Re::Vfs](#) the interface [File](#) is used for every real object.

See also

[L4Re::Vfs::File](#) for more information.

Definition at line 52 of file [vfs.h](#).

15.346.2 Member Enumeration Documentation

15.346.2.1 Ready_type

```
enum L4Re::Vfs::Generic_file::Ready_type : unsigned
```

Type of I/O operation/condition a file can indicate readiness.

As defined by select() and similar functions.

Definition at line 60 of file [vfs.h](#).

15.346.3 Member Function Documentation

15.346.3.1 check_ready()

```
virtual bool L4Re::Vfs::Generic_file::check_ready (
    Ready_type rt) [pure virtual], [noexcept]
```

Check whether the file is ready for an I/O operation/condition.

This method is used by the implementation of select() and similar functions.

Parameters

| | |
|-----------|---|
| <i>rt</i> | Type of the I/O operation/condition to be ready, as defined by the select() and similar functions (Read, Write, Exception). |
|-----------|---|

Return values

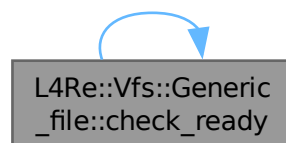
| | |
|--------------|--|
| <i>true</i> | The file is ready for the given type of I/O operation/condition. |
| <i>false</i> | The file is not ready for the given type of I/O operation/condition. |

Implemented in [L4Re::Vfs::Be_file](#).

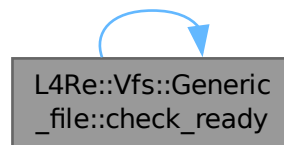
References [check_ready\(\)](#).

Referenced by [check_ready\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.346.3.2 fchmod()

```
virtual int L4Re::Vfs::Generic_file::fchmod (
    mode_t ) [pure virtual], [noexcept]
```

Change POSIX access rights on the file.

Backend for POSIX chmod and fchmod.

Implemented in [L4Re::Vfs::Be_file](#).

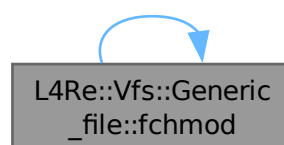
References [fchmod\(\)](#).

Referenced by [fchmod\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.346.3.3 `fstat()`

```
virtual int L4Re::Vfs::Generic_file::fstat (  
    struct stat64 * buf) const [pure virtual], [noexcept]
```

Get status information for the file.

This is the backend for POSIX `fstat`, `stat`, `fstat64` and friends.

Parameters

| | | |
|------------------|------------------|--|
| <code>out</code> | <code>buf</code> | This buffer is filled with the status information. |
|------------------|------------------|--|

Returns

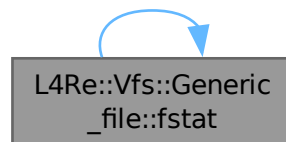
0 on success, or <0 on error.

Implemented in [L4Re::Vfs::Be_file](#).

References [fstat\(\)](#).

Referenced by [fstat\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.346.3.4 get_status_flags()

```
virtual int L4Re::Vfs::Generic_file::get_status_flags () const [pure virtual], [noexcept]
```

Get file status flags (fcntl F_GETFL).

This function is used by the fcntl implementation for the F_GETFL command.

Returns

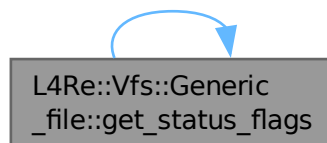
flags such as O_RDONLY, O_WRONLY, O_RDWR, O_DIRECT, O_ASYNC, O_NOATIME, O_NONBLOCK, or <0 on error.

Implemented in [L4Re::Vfs::Be_file](#).

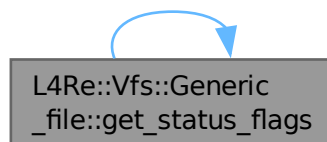
References [get_status_flags\(\)](#).

Referenced by [get_status_flags\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.346.3.5 set_status_flags()

```
virtual int L4Re::Vfs::Generic_file::set_status_flags (  
    long flags) [pure virtual], [noexcept]
```

Set file status flags (fcntl F_SETFL).

This function is used by the fcntl implementation for the F_SETFL command.

Parameters

| | |
|--------------|---|
| <i>flags</i> | The file status flags to set. This must be a combination of O_RDONLY, O_WRONLY, O_RDWR, O_APPEND, O_ASYNC, O_DIRECT, O_NOATIME, O_NONBLOCK. |
|--------------|---|

Note

Creation flags such as O_CREAT, O_EXCL, O_NOCTTY, O_TRUNC are ignored.

Returns

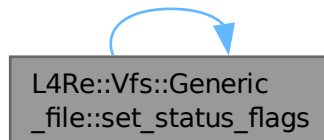
0 on success, or <0 on error.

Implemented in [L4Re::Vfs::Be_file](#).

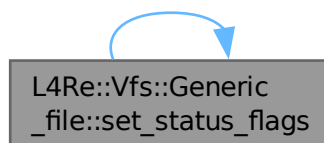
References [set_status_flags\(\)](#).

Referenced by [set_status_flags\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.346.3.6 unlock_all_locks()

```
virtual int L4Re::Vfs::Generic_file::unlock_all_locks () [pure virtual], [noexcept]
```

Unlock all locks on the file.

Note

All locks means all locks independent of which file the locks were taken by.

This method is called by the POSIX close implementation to get the POSIX semantics of releasing all locks taken by this application on a close for any fd referencing the real file.

Returns

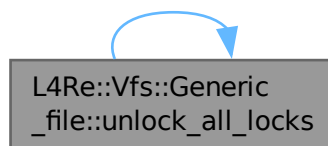
0 on success, or <0 on error.

Implemented in [L4Re::Vfs::Be_file](#).

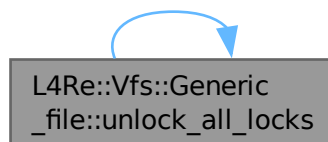
References [unlock_all_locks\(\)](#).

Referenced by [unlock_all_locks\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

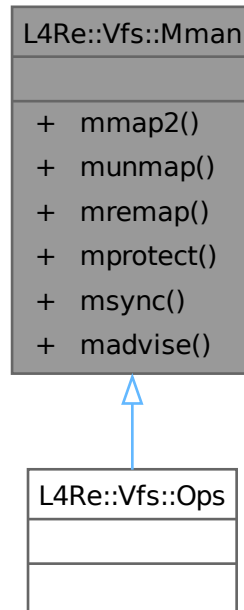
- [l4/l4re_vfs/vfs.h](#)

15.347 L4Re::Vfs::Mman Class Reference

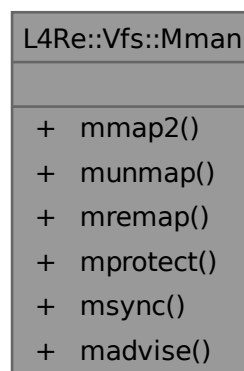
Interface for POSIX memory management.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::Mman:



Collaboration diagram for L4Re::Vfs::Mman:



Public Member Functions

- virtual int **mmap2** (void *start, size_t len, int prot, int flags, int fd, off_t offset, void **ptr) noexcept=0
Backend for the mmap2 system call.
- virtual int **munmap** (void *start, size_t len) noexcept=0
Backend for the munmap system call.
- virtual int **mremap** (void *old, size_t old_sz, size_t new_sz, int flags, void **new_addr) noexcept=0
Backend for the mremap system call.
- virtual int **mprotect** (const void *a, size_t sz, int prot) noexcept=0
Backend for the mprotect system call.
- virtual int **msync** (void *addr, size_t len, int flags) noexcept=0
Backend for the msync system call.
- virtual int **madvise** (void *addr, size_t len, int advice) noexcept=0
Backend for the madvise system call.

15.347.1 Detailed Description

Interface for POSIX memory management.

Note

This interface usually exists as a singleton and as a superclass of [L4Re::Vfs::Ops](#).

An implementation for this interface is in [l4/l4re_vfs/impl/vfs_impl.h](#) and used by the l4re_vfs library or by the VFS implementation in ldso.

Definition at line 773 of file [vfs.h](#).

The documentation for this class was generated from the following file:

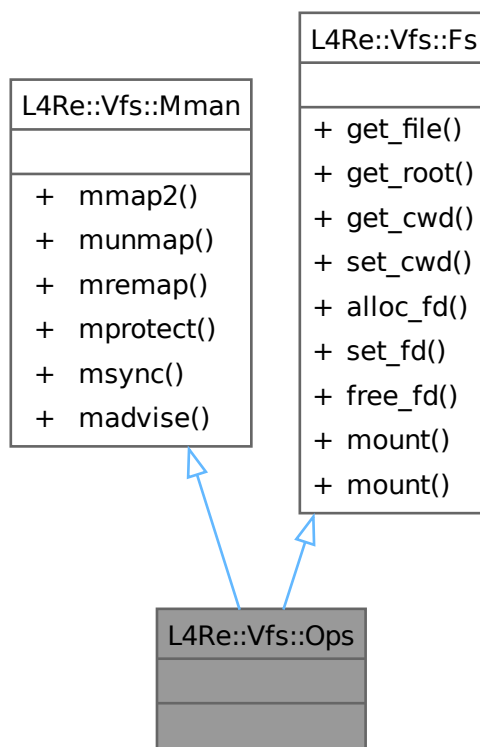
- l4/l4re_vfs/vfs.h

15.348 L4Re::Vfs::Ops Class Reference

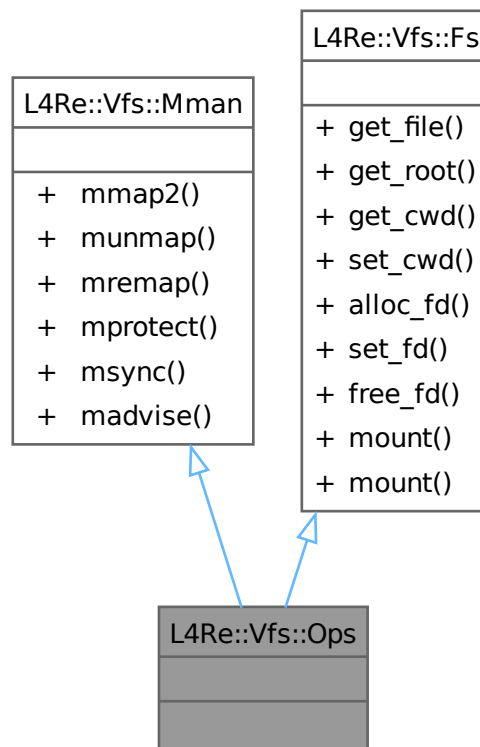
Interface for the POSIX backends of an application.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::Ops:



Collaboration diagram for L4Re::Vfs::Ops:



Additional Inherited Members

Public Member Functions inherited from [L4Re::Vfs::Mman](#)

- virtual int **mmap2** (void *start, size_t len, int prot, int flags, int fd, off_t offset, void **ptr) noexcept=0
Backend for the mmap2 system call.
- virtual int **munmap** (void *start, size_t len) noexcept=0
Backend for the munmap system call.
- virtual int **mremap** (void *old, size_t old_sz, size_t new_sz, int flags, void **new_addr) noexcept=0
Backend for the mremap system call.
- virtual int **mprotect** (const void *a, size_t sz, int prot) noexcept=0
Backend for the mprotect system call.
- virtual int **msync** (void *addr, size_t len, int flags) noexcept=0
Backend for the msync system call.
- virtual int **madvice** (void *addr, size_t len, int advice) noexcept=0
Backend for the madvice system call.

Public Member Functions inherited from `L4Re::Vfs::Fs`

- virtual `cxx::Ref_ptr< File > get_file` (int fd) noexcept=0
Get the `L4Re::Vfs::File` for the file descriptor fd.
- virtual `cxx::Ref_ptr< File > get_root` () noexcept=0
Get the directory object for the application's root directory.
- virtual `cxx::Ref_ptr< File > get_cwd` () noexcept
Get the directory object for the application's current working directory.
- virtual void `set_cwd` (`cxx::Ref_ptr< File > const &`) noexcept
Set the current working directory for the application.
- virtual int `alloc_fd` (`cxx::Ref_ptr< File > const &`=`cxx::Ref_ptr<>::Nil`) noexcept=0
Allocate the next free file descriptor.
- virtual `cxx::Pair< cxx::Ref_ptr< File >, int > set_fd` (int fd, `cxx::Ref_ptr< File > const &`=`cxx::Ref_ptr<>::Nil`) noexcept=0
Set the file object referenced by the file descriptor fd.
- virtual `cxx::Ref_ptr< File > free_fd` (int fd) noexcept=0
Free the file descriptor fd.
- virtual int `mount` (char const *path, `cxx::Ref_ptr< File > const &`dir) noexcept=0
Mount a given file object at the given global path in the VFS.
- int `mount` (char const *source, char const *target, char const *fstype, unsigned long mountflags, void const *data) noexcept
Backend for the POSIX mount call.

15.348.1 Detailed Description

Interface for the POSIX backends of an application.

Note

There usually exists a single instance of this interface available via `L4Re::Vfs::vfs_ops` that is used for all kinds of C-Library functions.

Definition at line 1109 of file `vfs.h`.

The documentation for this class was generated from the following file:

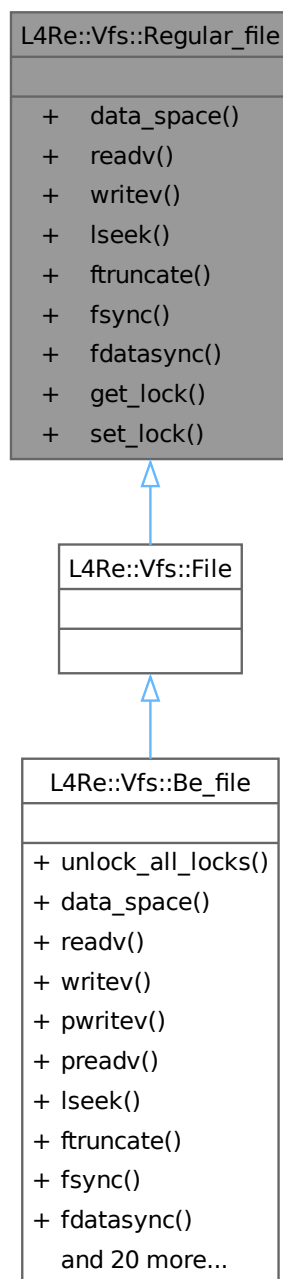
- `l4/l4re_vfs/vfs.h`

15.349 L4Re::Vfs::Regular_file Class Reference

Interface for a POSIX file that provides regular file semantics.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::Regular_file:



Collaboration diagram for `L4Re::Vfs::Regular_file`:

| L4Re::Vfs::Regular_file | |
|-------------------------|---------------------------|
| | |
| + | <code>data_space()</code> |
| + | <code>readv()</code> |
| + | <code>writew()</code> |
| + | <code>lseek()</code> |
| + | <code>ftruncate()</code> |
| + | <code>fsync()</code> |
| + | <code>fdatasync()</code> |
| + | <code>get_lock()</code> |
| + | <code>set_lock()</code> |

Public Member Functions

- virtual `L4::Cap< L4Re::Dataspace > data_space ()` noexcept=0
Get an `L4Re::Dataspace` object for the file.
- virtual `ssize_t readv (const struct iovec *, int iovcnt)` noexcept=0
Read one or more blocks of data from the file.
- virtual `ssize_t writew (const struct iovec *, int iovcnt)` noexcept=0
Write one or more blocks of data to the file.
- virtual `off64_t lseek (off64_t, int)` noexcept=0
Change the file pointer.
- virtual `int ftruncate (off64_t pos)` noexcept=0
Truncate the file at the given position.
- virtual `int fsync ()` const noexcept=0
Sync the data and meta data to persistent storage.
- virtual `int fdatasync ()` const noexcept=0
Sync the data to persistent storage.
- virtual `int get_lock (struct flock64 *lock)` noexcept=0
Test if the given lock can be placed in the file.
- virtual `int set_lock (struct flock64 *lock, bool wait)` noexcept=0
Acquire or release the given lock on the file.

15.349.1 Detailed Description

Interface for a POSIX file that provides regular file semantics.

Real objects always use the combined `L4Re::Vfs::File` interface.

Definition at line 287 of file `vfs.h`.

15.349.2 Member Function Documentation

15.349.2.1 data_space()

```
virtual L4::Cap< L4Re::Dataspace > L4Re::Vfs::Regular_file::data_space () [pure virtual],
[noexcept]
```

Get an [L4Re::Dataspace](#) object for the file.

This is used as a backend for POSIX mmap and mmap2 functions.

Note

mmap is not possible if the function returns an invalid capability.

Returns

A capability to an [L4Re::Dataspace](#) that represents the file contents in an [L4Re](#) way.

Implemented in [L4Re::Vfs::Be_file](#).

15.349.2.2 fdatsync()

```
virtual int L4Re::Vfs::Regular_file::fdatsync () const [pure virtual], [noexcept]
```

Sync the data to persistent storage.

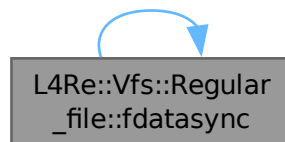
This is the backend for POSIX fdatsync.

Implemented in [L4Re::Vfs::Be_file](#).

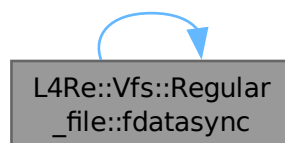
References [fdatsync\(\)](#).

Referenced by [fdatsync\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.349.2.3 fsync()

```
virtual int L4Re::Vfs::Regular_file::fsync () const [pure virtual], [noexcept]
```

Sync the data and meta data to persistent storage.

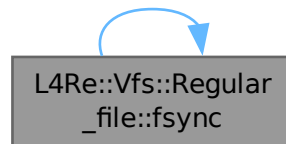
This is the backend for POSIX fsync.

Implemented in [L4Re::Vfs::Be_file](#).

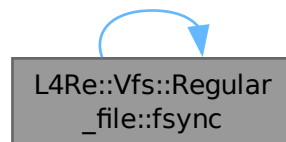
References [fsync\(\)](#).

Referenced by [fsync\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.349.2.4 ftruncate()

```
virtual int L4Re::Vfs::Regular_file::ftruncate (  
    off64_t pos) [pure virtual], [noexcept]
```

Truncate the file at the given position.

This function is the backend for truncate and friends.

Parameters

| | |
|------------|--|
| <i>pos</i> | The offset at which the file shall be truncated. |
|------------|--|

Returns

0 on success, or <0 on error.

Implemented in [L4Re::Vfs::Be_file](#).

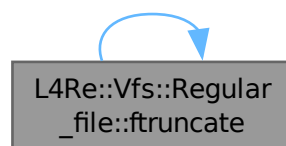
References [ftruncate\(\)](#).

Referenced by [ftruncate\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**15.349.2.5 get_lock()**

```
virtual int L4Re::Vfs::Regular_file::get_lock (
    struct flock64 * lock) [pure virtual], [noexcept]
```

Test if the given lock can be placed in the file.

This function is used as backend for fcntl F_GETLK commands.

Parameters

| | |
|-------------|--|
| <i>lock</i> | The lock that shall be placed on the file. The <i>l_type</i> member will contain <code>F_UNLCK</code> if the lock could be placed. |
|-------------|--|

Returns

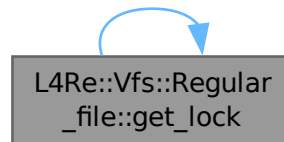
0 on success, <0 on error.

Implemented in [L4Re::Vfs::Be_file](#).

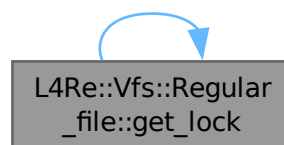
References [get_lock\(\)](#).

Referenced by [get_lock\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**15.349.2.6 lseek()**

```
virtual off64_t L4Re::Vfs::Regular_file::lseek (
    off64_t ,
    int ) [pure virtual], [noexcept]
```

Change the file pointer.

This is the backend for POSIX seek, lseek and friends.

Returns

The new file position, or <0 on error.

Implemented in [L4Re::Vfs::Be_file](#).

References [lseek\(\)](#).

Referenced by [lseek\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.349.2.7 readv()

```
virtual ssize_t L4Re::Vfs::Regular_file::readv (  
    const struct iovec * ,  
    int iovcnt) [pure virtual], [noexcept]
```

Read one or more blocks of data from the file.

This function acts as backend for POSIX read and readv calls and reads data starting from the f_pos pointer of that open file. The file pointer is advanced according to the number of bytes read.

Returns

The number of bytes read from the file, or <0 on error.

Implemented in [L4Re::Vfs::Be_file](#).

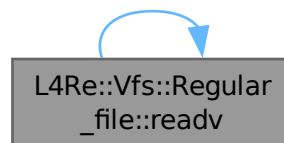
References [readv\(\)](#).

Referenced by [readv\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**15.349.2.8 set_lock()**

```
virtual int L4Re::Vfs::Regular_file::set_lock (
    struct flock64 * lock,
    bool wait) [pure virtual], [noexcept]
```

Acquire or release the given lock on the file.

This function is used as backend for `fcntl F_SETLK` and `F_SETLKW` commands.

Parameters

| | |
|-------------|--|
| <i>lock</i> | The lock that shall be placed on the file. |
|-------------|--|

| | |
|-------------|---|
| <i>wait</i> | If true, then block if there is a conflicting lock on the file. |
|-------------|---|

Returns

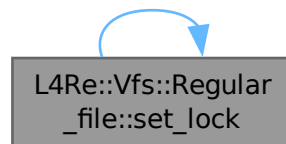
0 on success, <0 on error.

Implemented in [L4Re::Vfs::Be_file](#).

References [set_lock\(\)](#).

Referenced by [set_lock\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**15.349.2.9 writev()**

```

virtual ssize_t L4Re::Vfs::Regular_file::writev (
    const struct iovec * ,
    int iovcnt) [pure virtual], [noexcept]
  
```

Write one or more blocks of data to the file.

This function acts as backend for POSIX write and writev calls. The data is written starting at the current file pointer and the file pointer must be advanced according to the number of written bytes.

Returns

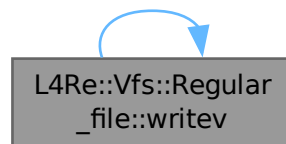
The number of bytes written to the file, or <0 on error.

Implemented in [L4Re::Vfs::Be_file](#).

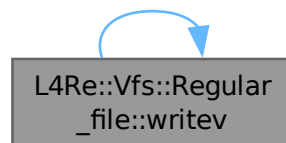
References [writev\(\)](#).

Referenced by [writev\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

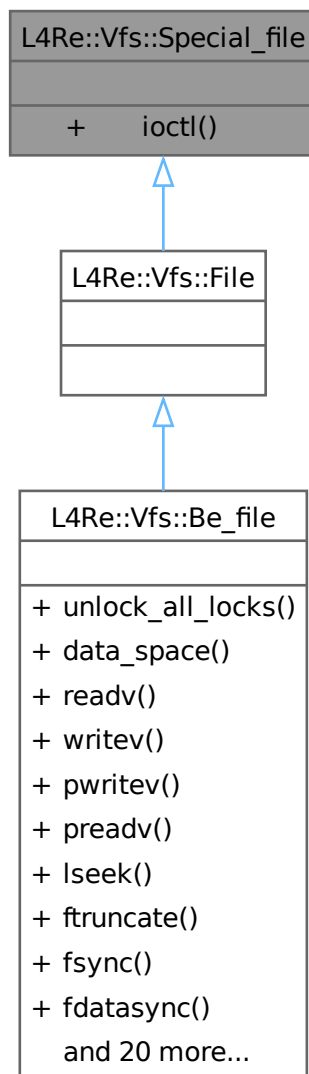
- `l4/l4re_vfs/vfs.h`

15.350 L4Re::Vfs::Special_file Class Reference

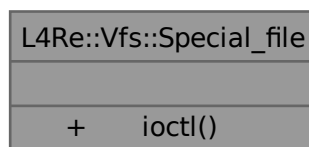
Interface for a POSIX file that provides special file semantics.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::Special_file:



Collaboration diagram for L4Re::Vfs::Special_file:



Public Member Functions

- virtual int [ioctl](#) (unsigned long cmd, va_list args) noexcept=0
The famous IO control.

15.350.1 Detailed Description

Interface for a POSIX file that provides special file semantics.

Real objects always use the combined [L4Re::Vfs::File](#) interface.

Definition at line [420](#) of file [vfs.h](#).

15.350.2 Member Function Documentation

15.350.2.1 ioctl()

```
virtual int L4Re::Vfs::Special_file::ioctl (  
    unsigned long cmd,  
    va_list args) [pure virtual], [noexcept]
```

The famous IO control.

Backend for POSIX generic object invocation ioctl.

Parameters

| | |
|-------------|--|
| <i>cmd</i> | The ioctl command. |
| <i>args</i> | The arguments for the ioctl, usually some kind of pointer. |

Returns

>=0 on success, or <0 on error.

Implemented in [L4Re::Vfs::Be_file](#).

The documentation for this class was generated from the following file:

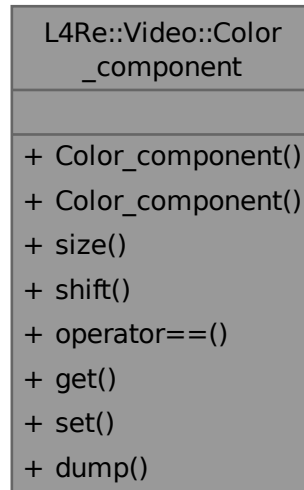
- [l4/l4re_vfs/vfs.h](#)

15.351 L4Re::Video::Color_component Class Reference

A color component.

```
#include <colors>
```

Collaboration diagram for L4Re::Video::Color_component:



Public Member Functions

- **Color_component ()**
Constructor.
- **Color_component** (unsigned char bits, unsigned char shift)
Constructor.
- unsigned char **size** () const
Return the number of bits used by the component.
- unsigned char **shift** () const
Return the position of the component in the pixel.
- bool **operator==** (**Color_component** const &o) const
Compare for equality.
- int **get** (unsigned long v) const
Get component from value (normalized to 16bits).
- long unsigned **set** (int v) const
Transform 16bit normalized value to the component in the color space.
- template<typename OUT>
void **dump** (OUT &s) const
Dump information on the view information to a stream.

15.351.1 Detailed Description

A color component.

Definition at line 21 of file [colors](#).

15.351.2 Constructor & Destructor Documentation

15.351.2.1 Color_component()

```
L4Re::Video::Color_component::Color_component (
    unsigned char bits,
    unsigned char shift) [inline]
```

Constructor.

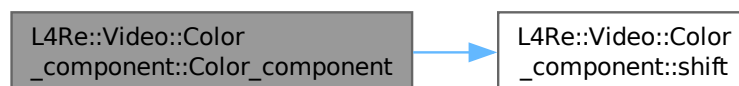
Parameters

| | |
|--------------|--|
| <i>bits</i> | Number of bits used by the component |
| <i>shift</i> | Position in bits of the component in the pixel |

Definition at line 36 of file [colors](#).

References [shift\(\)](#).

Here is the call graph for this function:



15.351.3 Member Function Documentation

15.351.3.1 dump()

```
template<typename OUT>
void L4Re::Video::Color_component::dump (
    OUT & s) const [inline]
```

Dump information on the view information to a stream.

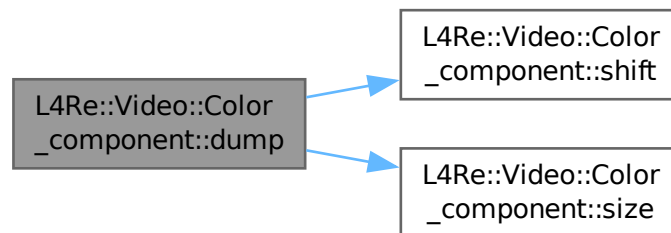
Parameters

| | |
|----------|--------|
| <i>s</i> | Stream |
|----------|--------|

Definition at line 81 of file [colors](#).

References [shift\(\)](#), and [size\(\)](#).

Here is the call graph for this function:

**15.351.3.2 get()**

```
int L4Re::Video::Color_component::get (
    unsigned long v) const [inline]
```

Get component from value (normalized to 16bits).

Parameters

| | |
|----------|-------|
| <i>v</i> | Value |
|----------|-------|

Returns

Converted value

Definition at line 63 of file [colors](#).

15.351.3.3 operator==()

```
bool L4Re::Video::Color_component::operator== (
    Color_component const & o) const [inline]
```

Compare for equality.

Returns

True if the same components are described, false if not.

Definition at line 55 of file [colors](#).

References [Color_component\(\)](#).

Here is the call graph for this function:

**15.351.3.4 set()**

```
long unsigned L4Re::Video::Color_component::set (
    int v) const [inline]
```

Transform 16bit normalized value to the component in the color space.

Parameters

| | |
|---|-------------------------------|
| v | Value return Converted value. |
|---|-------------------------------|

Definition at line 73 of file [colors](#).

15.351.3.5 shift()

```
unsigned char L4Re::Video::Color_component::shift () const [inline]
```

Return the position of the component in the pixel.

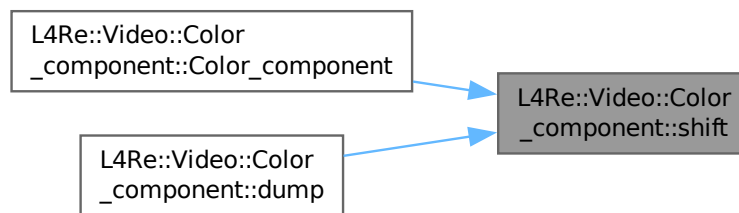
Returns

Position in bits of the component in the pixel

Definition at line 49 of file [colors](#).

Referenced by [Color_component\(\)](#), and [dump\(\)](#).

Here is the caller graph for this function:



15.351.3.6 size()

```
unsigned char L4Re::Video::Color_component::size () const [inline]
```

Return the number of bits used by the component.

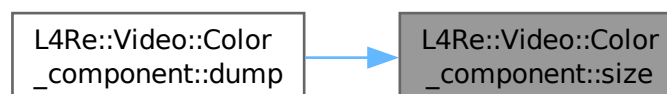
Returns

Number of bits used by the component

Definition at line 43 of file [colors](#).

Referenced by [dump\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

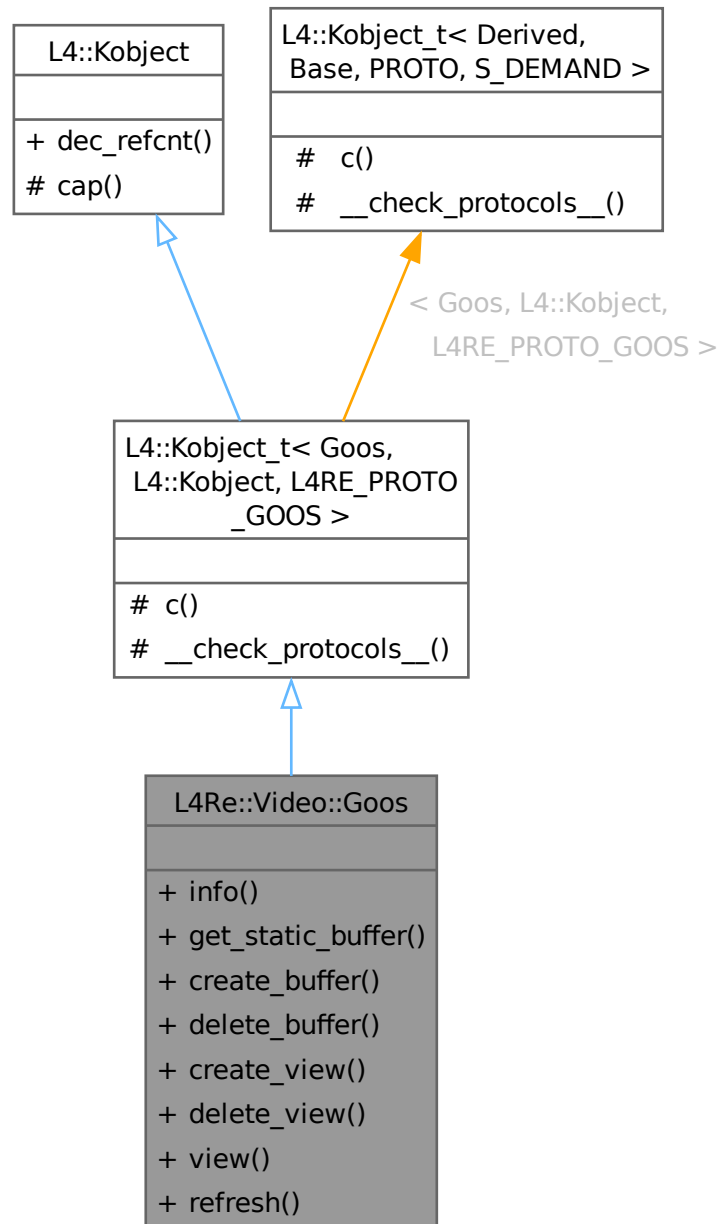
- `l4/re/video/colors`

15.352 L4Re::Video::Goos Class Reference

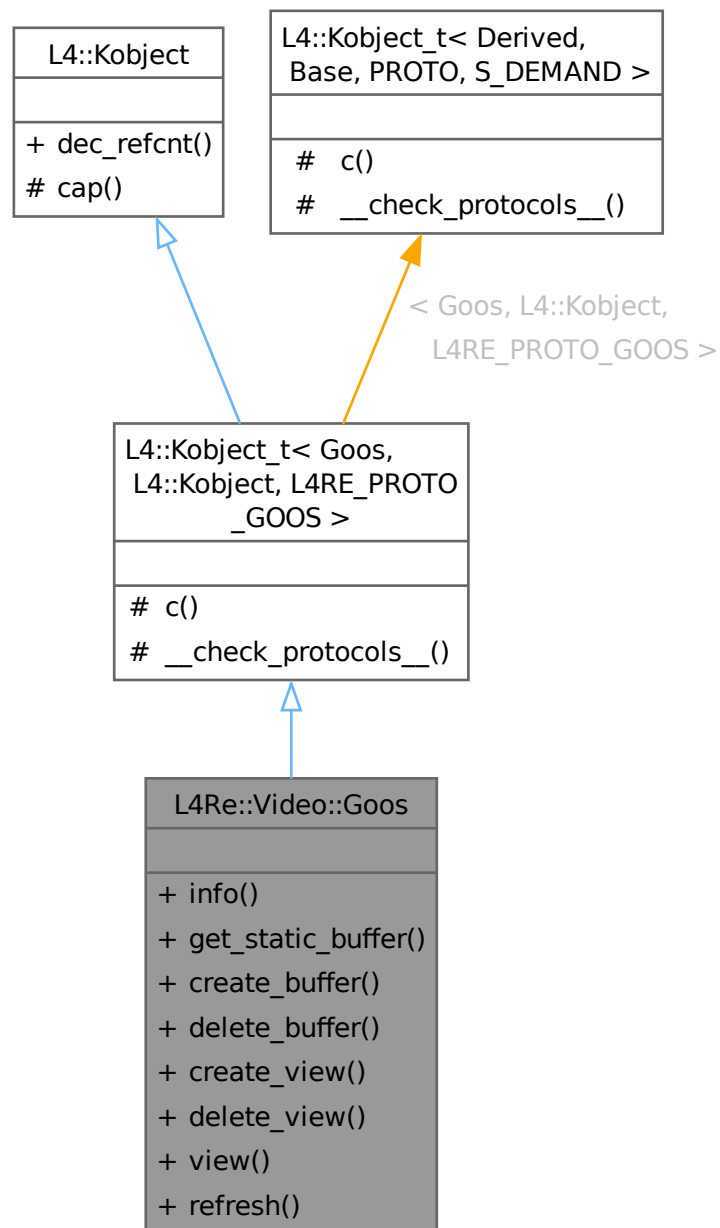
Class that abstracts framebuffers.

```
#include <goos>
```

Inheritance diagram for L4Re::Video::Goos:



Collaboration diagram for L4Re::Video::Goos:



Data Structures

- struct [Info](#)
Information structure of a [Goos](#).

Public Types

- enum [Flags](#) { [F_auto_refresh](#) = 0x01 , [F_pointer](#) = 0x02 , [F_dynamic_views](#) = 0x04 , [F_dynamic_buffers](#) = 0x08 }

Flags for a Goos.

Public Member Functions

- [l4_ret_t info](#) ([Info](#) *info)
Return the [Goos](#) information of the [Goos](#).
- [l4_ret_t get_static_buffer](#) (unsigned idx, [L4::lpc::Out](#)< [L4::Cap](#)< [L4Re::Dataspace](#) > > rbuf)
Return a static buffer of a [Goos](#).
- [l4_ret_t create_buffer](#) (unsigned long size, [L4::lpc::Out](#)< [L4::Cap](#)< [L4Re::Dataspace](#) > > rbuf)
Create a buffer.
- [l4_ret_t delete_buffer](#) (unsigned idx)
Delete a buffer.
- [l4_ret_t create_view](#) ([View](#) *view, [l4_utcb_t](#) *utcb=[l4_utcb](#)()) const noexcept
Create a view.
- [l4_ret_t delete_view](#) ([View](#) const &v, [l4_utcb_t](#) *utcb=[l4_utcb](#)()) const noexcept
Delete a view.
- [View view](#) (unsigned index) const noexcept
Return a view.
- [l4_ret_t refresh](#) (int x, int y, int w, int h)
Trigger refreshing of the given area on the virtual screen.

Public Member Functions inherited from [L4::Kobject](#)

- [l4_msgtag_t dec_refcnt](#) ([l4_mword_t](#) diff, [l4_utcb_t](#) *utcb=[l4_utcb](#)())
Decrement the in kernel reference counter for the object.

Additional Inherited Members

Protected Types inherited from

[L4::Kobject_t](#)< [Goos](#), [L4::Kobject](#), [L4RE_PROTO_GOOS](#) >

- typedef [Goos](#) **Class**
The target interface type (inheriting from [Kobject_t](#)).
- typedef [Typeid::Iface](#)< [PROTO](#), [Goos](#) > **__Iface**
The interface description for the derived class.
- typedef [Typeid::Merge_list](#)< [Typeid::Iface_list](#)< **__Iface** >, typename [L4::Kobject::__Iface_list](#) > **__Iface_list**
The list of all RPC interfaces provided directly or through inheritance.

Protected Member Functions inherited from

[L4::Kobject_t](#)< [Goos](#), [L4::Kobject](#), [L4RE_PROTO_GOOS](#) >

- [L4::Cap](#)< [Class](#) > **c** () const noexcept
Get the capability to ourselves.

Protected Member Functions inherited from [L4::Kobject](#)

- [l4_cap_idx_t cap](#) () const noexcept
Return capability selector.

Static Protected Member Functions inherited from [L4::Kobject_t](#)< [Goos](#), [L4::Kobject](#), [L4RE_PROTO_GOOS](#) >

- static void `__check_protocols__()` noexcept
Helper to check for protocol conflicts.

15.352.1 Detailed Description

[Class](#) that abstracts framebuffer.

A framebuffer is the pixel data that is displayed on a screen and a [Goos](#) object lets the user manipulate that data. A [Goos](#) makes use of two kinds of objects:

- Buffers in the form of [L4Re::Dataspace](#) objects. These hold the bytes for the pixel data.
- [L4Re::Video::View](#) objects.

Both can either be static, that is their number and configuration is fixed and determined by the framebuffer, or they can be dynamic, with the user allocating them.

Definition at line 223 of file [goos](#).

15.352.2 Member Enumeration Documentation

15.352.2.1 Flags

```
enum L4Re::Video::Goos::Flags
```

[Flags](#) for a [Goos](#).

Enumerator

| | |
|--------------------------------|--|
| <code>F_auto_refresh</code> | The graphics display is automatically refreshed. |
| <code>F_pointer</code> | We have a mouse pointer. |
| <code>F_dynamic_views</code> | Supports dynamically allocated views. |
| <code>F_dynamic_buffers</code> | Supports dynamically allocated buffers. |

Definition at line 228 of file [goos](#).

15.352.3 Member Function Documentation

15.352.3.1 create_buffer()

```
l4_ret_t L4Re::Video::Goos::create_buffer (
    unsigned long size,
    L4::Ipc::Out< L4::Cap< L4Re::Dataspace > > rbuf)
```

Create a buffer.

Parameters

| | |
|-------------|---|
| <i>size</i> | Size of buffer in bytes. |
| <i>rbuf</i> | Capability slot to point the buffer dataspace to. |

Return values

| | |
|----------|--|
| ≥ 0 | Success, the value returned is the buffer index. |
| < 0 | Error |

Referenced by [get_static_buffer\(\)](#).

Here is the caller graph for this function:



15.352.3.2 create_view()

```
l4_ret_t L4Re::Video::Goos::create_view (
    View * view,
    l4_utcb_t * utcb = l4_utcb()) const [inline], [noexcept]
```

Create a view.

Parameters

| | | |
|-----|-------------|--|
| out | <i>view</i> | A view object. |
| | <i>utcb</i> | UTCB of the caller. This is a default parameter. |

Return values

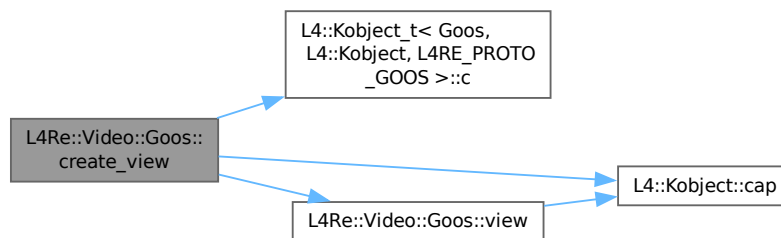
| | |
|----------|--|
| ≥ 0 | Success, the value returned is the view index. |
| < 0 | Error |

Definition at line 312 of file [goos](#).

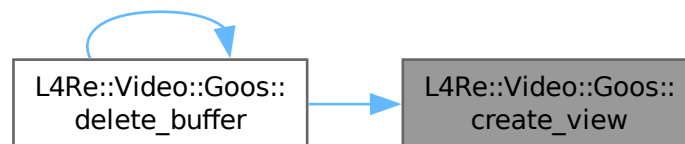
References [L4::Kobject_t< Goos, L4::Kobject, L4RE_PROTO_GOOS >::c\(\)](#), [L4::Kobject::cap\(\)](#), and [view\(\)](#).

Referenced by [delete_buffer\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.352.3.3 delete_buffer()

```
l4_ret_t L4Re::Video::Goos::delete_buffer (
    unsigned idx)
```

Delete a buffer.

Parameters

| | |
|------------|-----------------------------------|
| <i>idx</i> | Buffer to delete. |
|------------|-----------------------------------|

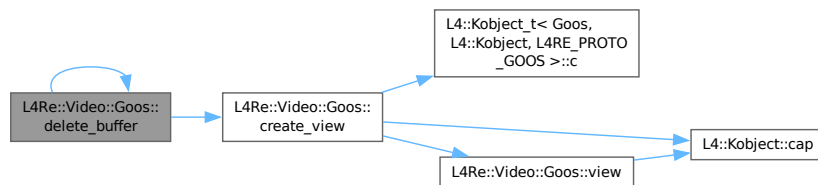
Return values

| | |
|----|---------|
| 0 | Success |
| <0 | Error |

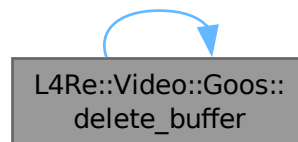
References [create_view\(\)](#), and [delete_buffer\(\)](#).

Referenced by [delete_buffer\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**15.352.3.4 delete_view()**

```

l4_ret_t L4Re::Video::Goos::delete_view (
    View const & v,
    l4_utcb_t * utcb = l4_utcb()) const [inline], [noexcept]

```

Delete a view.

Parameters

| | |
|-------------|--|
| <i>v</i> | The view object to delete. |
| <i>utcb</i> | UTCB of the caller. This is a default parameter. |

Return values

| | |
|----|---------|
| 0 | Success |
| <0 | Error |

Definition at line 332 of file [goos](#).

References [L4::Kobject_t< Goos, L4::Kobject, L4RE_PROTO_GOOS >::c\(\)](#).

Here is the call graph for this function:

**15.352.3.5 get_static_buffer()**

```

l4_ret_t L4Re::Video::Goos::get_static_buffer (
    unsigned idx,
    L4::Ipc::Out< L4::Cap< L4Re::Dataspace > > rbuf)
  
```

Return a static buffer of a [Goos](#).

Parameters

| | |
|-------------|---|
| <i>idx</i> | Index of the static buffer. |
| <i>rbuf</i> | Capability slot to point the buffer dataspace to. |

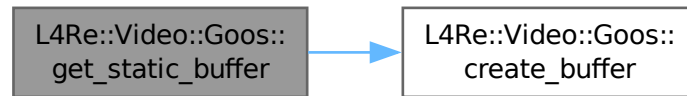
Return values

| | |
|----|---------|
| 0 | Success |
| <0 | Error |

References [create_buffer\(\)](#).

Referenced by [info\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.352.3.6 info()

```
l4_ret_t L4Re::Video::Goos::info (
    Info * info)
```

Return the [Goos](#) information of the [Goos](#).

Parameters

| | | |
|-----|------|---|
| out | info | Goos information structure pointer. |
|-----|------|---|

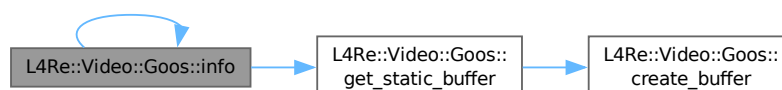
Return values

| | |
|----|---------|
| 0 | Success |
| <0 | Error |

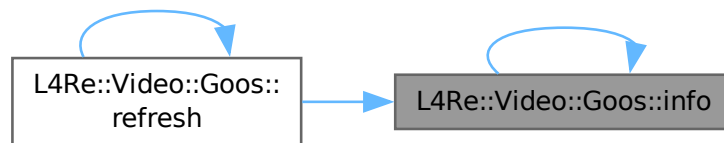
References [get_static_buffer\(\)](#), and [info\(\)](#).

Referenced by [info\(\)](#), and [refresh\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.352.3.7 view()

```
View L4Re::Video::Goos::view (  
    unsigned index) const [inline], [noexcept]
```

Return a view.

Parameters

| | |
|--------------|------------------------------|
| <i>index</i> | Index of the view to return. |
|--------------|------------------------------|

Returns

The view.

Definition at line 363 of file [goos](#).

References [L4::Kobject::cap\(\)](#).

Referenced by [create_view\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

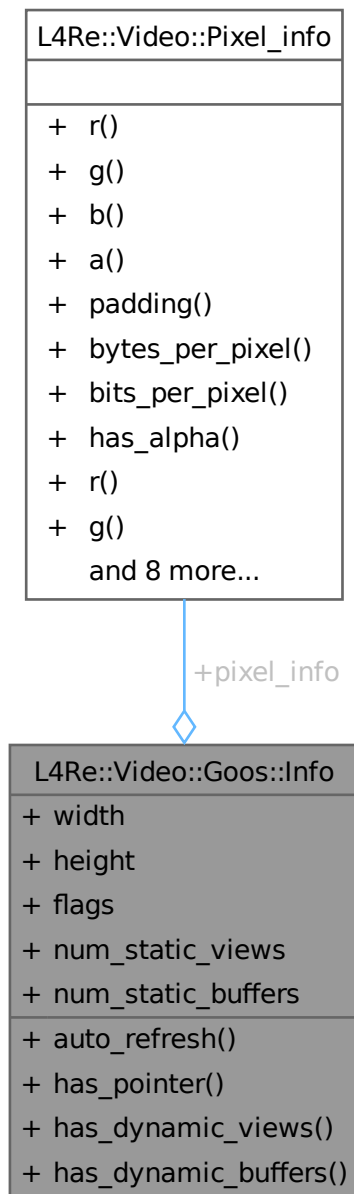
- `I4/re/video/goos`

15.353 L4Re::Video::Goos::Info Struct Reference

Information structure of a [Goos](#).

```
#include <goos>
```

Collaboration diagram for L4Re::Video::Goos::Info:



Public Member Functions

- bool **auto_refresh** () const
Return whether this [Goos](#) does auto refreshing or the view refresh functions must be used to make changes visible.
- bool **has_pointer** () const
Return whether a pointer is used by the provider of the [Goos](#).
- bool **has_dynamic_views** () const
Return whether dynamic view are supported.
- bool **has_dynamic_buffers** () const
Return whether dynamic buffers are supported.

Data Fields

- unsigned long **width**
Width.
- unsigned long **height**
Height.
- unsigned **flags**
Flags, see [Flags](#).
- unsigned **num_static_views**
Number of static view.
- unsigned **num_static_buffers**
Number of static buffers.
- [Pixel_info](#) **pixel_info**
Pixel information.

15.353.1 Detailed Description

Information structure of a [Goos](#).

Definition at line [237](#) of file [goos](#).

The documentation for this struct was generated from the following file:

- [l4/re/video/goos](#)

15.354 L4Re::Video::Pixel_info Class Reference

Pixel information.

```
#include <colors>
```

Collaboration diagram for L4Re::Video::Pixel_info:

| L4Re::Video::Pixel_info |
|--|
| <ul style="list-style-type: none"> + r() + g() + b() + a() + padding() + bytes_per_pixel() + bits_per_pixel() + has_alpha() + r() + g() and 8 more... |

Public Member Functions

- [Color_component](#) const & [r](#) () const
Return the red color component of the pixel.
- [Color_component](#) const & [g](#) () const
Return the green color component of the pixel.
- [Color_component](#) const & [b](#) () const
Return the blue color component of the pixel.
- [Color_component](#) const & [a](#) () const
Return the alpha color component of the pixel.
- [Color_component](#) const [padding](#) () const
Compute the padding pseudo component.
- unsigned char [bytes_per_pixel](#) () const
Query size of pixel in bytes.
- unsigned char [bits_per_pixel](#) () const
Number of bits of the pixel.
- bool [has_alpha](#) () const
Return whether the pixel has an alpha channel.
- void [r](#) ([Color_component](#) const &c)
Set the red color component of the pixel.
- void [g](#) ([Color_component](#) const &c)
Set the green color component of the pixel.
- void [b](#) ([Color_component](#) const &c)
Set the blue color component of the pixel.
- void [a](#) ([Color_component](#) const &c)
Set the alpha color component of the pixel.
- void [bytes_per_pixel](#) (unsigned char bpp)
Set the size of the pixel in bytes.
- **Pixel_info** ()
Constructor.
- [Pixel_info](#) (unsigned char bpp, char [r](#), char [rs](#), char [g](#), char [gs](#), char [b](#), char [bs](#), char [a](#)=0, char [as](#)=0)
Constructor.
- template<typename VBI>
[Pixel_info](#) (VBI const *vbi)
Convenience constructor.
- bool [operator==](#) ([Pixel_info](#) const &o) const
Compare for complete equality of the color space.
- template<typename OUT>
void [dump](#) (OUT &s) const
Dump information on the pixel to a stream.

15.354.1 Detailed Description

Pixel information.

This class wraps the information on a pixel, such as the size and position of each color component in the pixel.

Definition at line 94 of file [colors](#).

15.354.2 Constructor & Destructor Documentation

15.354.2.1 Pixel_info() [1/2]

```
L4Re::Video::Pixel_info::Pixel_info (  
    unsigned char bpp,  
    char r,  
    char rs,  
    char g,  
    char gs,  
    char b,  
    char bs,  
    char a = 0,  
    char as = 0) [inline]
```

Constructor.

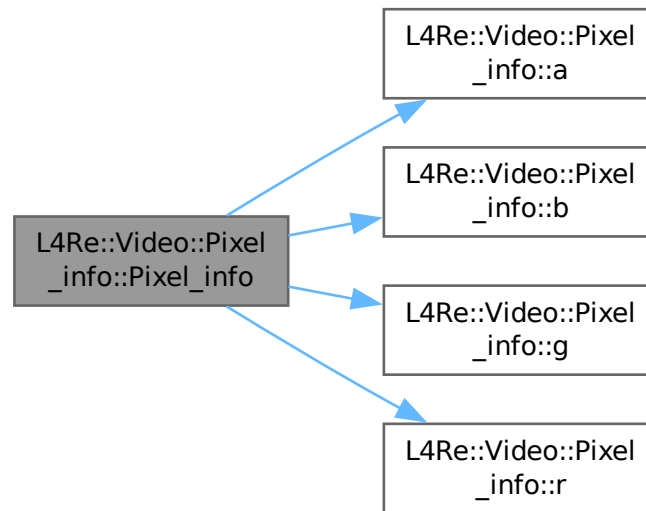
Parameters

| | |
|------------|---------------------------------------|
| <i>bpp</i> | Size of pixel in bytes. |
| <i>r</i> | Red component size. |
| <i>rs</i> | Red component shift. |
| <i>g</i> | Green component size. |
| <i>gs</i> | Green component shift. |
| <i>b</i> | Blue component size. |
| <i>bs</i> | Blue component shift. |
| <i>a</i> | Alpha component size, defaults to 0. |
| <i>as</i> | Alpha component shift, defaults to 0. |

Definition at line 212 of file [colors](#).

References [a\(\)](#), [b\(\)](#), [g\(\)](#), and [r\(\)](#).

Here is the call graph for this function:



15.354.2.2 Pixel_info() [2/2]

```
template<typename VBI>
L4Re::Video::Pixel_info::Pixel_info (
    VBI const * vbi) [inline], [explicit]
```

Convenience constructor.

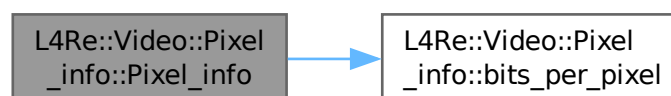
Parameters

| | |
|------------|--|
| <i>vbi</i> | Suitable information structure. Convenience constructor to create the pixel info from a VESA Framebuffer Info. |
|------------|--|

Definition at line 224 of file [colors](#).

References [bits_per_pixel\(\)](#).

Here is the call graph for this function:



15.354.3 Member Function Documentation

15.354.3.1 `a()` [1/2]

```
Color_component const & L4Re::Video::Pixel_info::a () const [inline]
```

Return the alpha color component of the pixel.

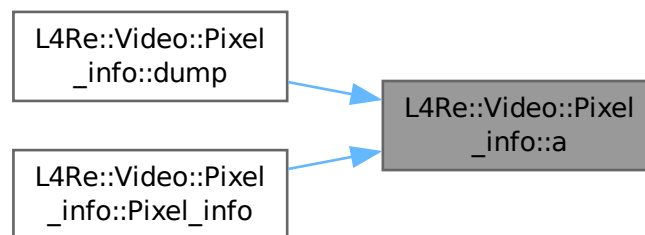
Returns

Alpha color component.

Definition at line 123 of file `colors`.

Referenced by `dump()`, and `Pixel_info()`.

Here is the caller graph for this function:



15.354.3.2 `a()` [2/2]

```
void L4Re::Video::Pixel_info::a (
    Color_component const & c) [inline]
```

Set the alpha color component of the pixel.

Parameters

| | |
|----------------|------------------------|
| <code>c</code> | Alpha color component. |
|----------------|------------------------|

Definition at line 187 of file `colors`.

15.354.3.3 b() [1/2]

```
Color_component const & L4Re::Video::Pixel_info::b () const [inline]
```

Return the blue color component of the pixel.

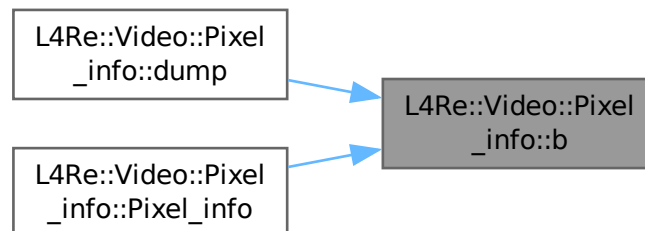
Returns

Blue color component.

Definition at line 117 of file [colors](#).

Referenced by [dump\(\)](#), and [Pixel_info\(\)](#).

Here is the caller graph for this function:

**15.354.3.4 b()** [2/2]

```
void L4Re::Video::Pixel_info::b (
    Color_component const & c) [inline]
```

Set the blue color component of the pixel.

Parameters

| | |
|----------|-----------------------|
| <i>c</i> | Blue color component. |
|----------|-----------------------|

Definition at line 181 of file [colors](#).

15.354.3.5 bits_per_pixel()

```
unsigned char L4Re::Video::Pixel_info::bits_per_pixel () const [inline]
```

Number of bits of the pixel.

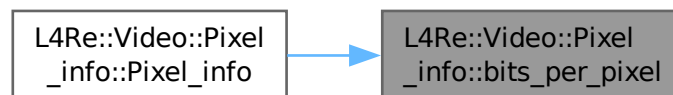
Returns

Number of bits used by the pixel.

Definition at line 156 of file [colors](#).

Referenced by [Pixel_info\(\)](#).

Here is the caller graph for this function:



15.354.3.6 bytes_per_pixel() [1/2]

```
unsigned char L4Re::Video::Pixel_info::bytes_per_pixel () const [inline]
```

Query size of pixel in bytes.

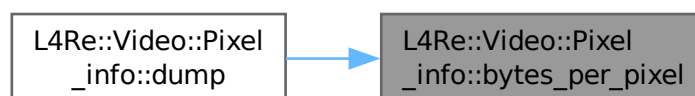
Returns

Size of pixel in bytes.

Definition at line 150 of file [colors](#).

Referenced by [dump\(\)](#).

Here is the caller graph for this function:



15.354.3.7 bytes_per_pixel() [2/2]

```
void L4Re::Video::Pixel_info::bytes_per_pixel (  
    unsigned char bpp) [inline]
```

Set the size of the pixel in bytes.

Parameters

| | |
|------------|-------------------------|
| <i>bpp</i> | Size of pixel in bytes. |
|------------|-------------------------|

Definition at line 193 of file [colors](#).

15.354.3.8 dump()

```
template<typename OUT>  
void L4Re::Video::Pixel_info::dump (  
    OUT & s) const [inline]
```

Dump information on the pixel to a stream.

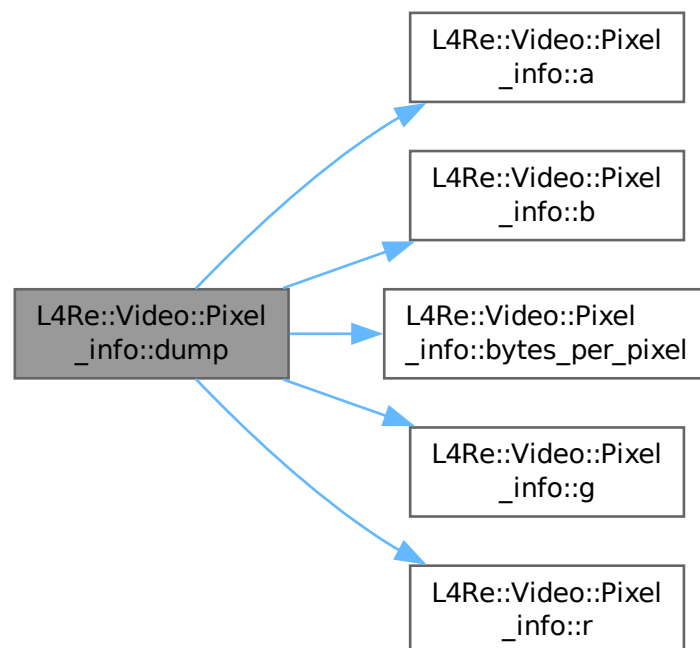
Parameters

| | |
|----------|--------|
| <i>s</i> | Stream |
|----------|--------|

Definition at line 246 of file [colors](#).

References [a\(\)](#), [b\(\)](#), [bytes_per_pixel\(\)](#), [g\(\)](#), and [r\(\)](#).

Here is the call graph for this function:



15.354.3.9 g() [1/2]

```
Color_component const & L4Re::Video::Pixel_info::g () const [inline]
```

Return the green color component of the pixel.

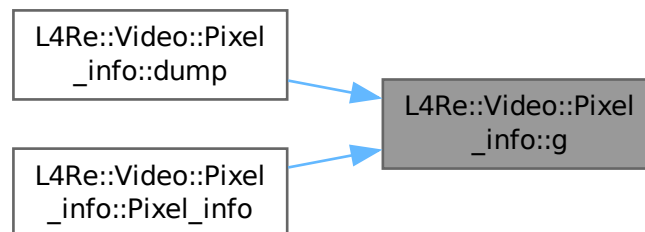
Returns

Green color component.

Definition at line 111 of file [colors](#).

Referenced by [dump\(\)](#), and [Pixel_info\(\)](#).

Here is the caller graph for this function:

**15.354.3.10 g() [2/2]**

```
void L4Re::Video::Pixel_info::g (
    Color_component const & c) [inline]
```

Set the green color component of the pixel.

Parameters

| | |
|----------|------------------------|
| <i>c</i> | Green color component. |
|----------|------------------------|

Definition at line 175 of file [colors](#).

15.354.3.11 has_alpha()

```
bool L4Re::Video::Pixel_info::has_alpha () const [inline]
```

Return whether the pixel has an alpha channel.

Returns

True if the pixel has an alpha channel, false if not.

Definition at line 163 of file [colors](#).

15.354.3.12 operator==()

```
bool L4Re::Video::Pixel_info::operator== (
    Pixel_info const & o) const [inline]
```

Compare for complete equality of the color space.

Parameters

| | |
|----------|---|
| <i>o</i> | A Pixel_info to compare to. |
|----------|---|

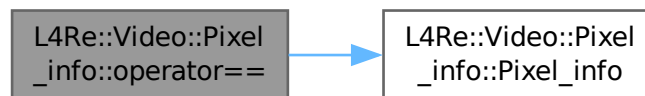
Returns

true if the both [Pixel_info](#)'s are equal, false if not.

Definition at line [236](#) of file [colors](#).

References [Pixel_info\(\)](#).

Here is the call graph for this function:



15.354.3.13 padding()

```
Color_component const L4Re::Video::Pixel_info::padding () const [inline]
```

Compute the padding pseudo component.

The padding pseudo component represents the trailing bits that are reserved in RGB32 and similar pixel formats.

Returns

Padding pseudo component.

Definition at line [131](#) of file [colors](#).

15.354.3.14 `r()` [1/2]

```
Color_component const & L4Re::Video::Pixel_info::r () const [inline]
```

Return the red color component of the pixel.

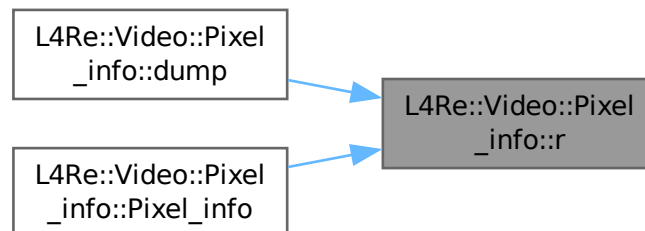
Returns

Red color component.

Definition at line 105 of file [colors](#).

Referenced by [dump\(\)](#), and [Pixel_info\(\)](#).

Here is the caller graph for this function:

**15.354.3.15** `r()` [2/2]

```
void L4Re::Video::Pixel_info::r (
    Color_component const & c) [inline]
```

Set the red color component of the pixel.

Parameters

| | |
|----------------|----------------------|
| <code>c</code> | Red color component. |
|----------------|----------------------|

Definition at line 169 of file [colors](#).

The documentation for this class was generated from the following file:

- `l4/re/video/colors`

15.355 L4Re::Video::View Class Reference

[View](#) of a framebuffer.

```
#include <goos>
```

Collaboration diagram for L4Re::Video::View:

| L4Re::Video::View |
|--|
| <ul style="list-style-type: none"> + info() + set_info() + set_viewport() + stack() + push_top() + push_bottom() + refresh() + valid() |

Data Structures

- struct [Info](#)
Information structure of a view.

Public Types

- enum [Flags](#) {
[F_none](#) = 0x00 , [F_set_buffer](#) = 0x01 , [F_set_buffer_offset](#) = 0x02 , [F_set_bytes_per_line](#) = 0x04 ,
[F_set_pixel](#) = 0x08 , [F_set_position](#) = 0x10 , [F_dyn_allocated](#) = 0x20 , [F_set_background](#) = 0x40 ,
[F_set_flags](#) = 0x80 , [F_fully_dynamic](#) }
Flags on a view.
- enum [V_flags](#) { [F_above](#) = 0x1000 , [F_flags_mask](#) = 0xff000 }
Property flags of a view.

Public Member Functions

- `int info (Info *info) const noexcept`
Return the view information of the view.
- `int set_info (Info const &info) const noexcept`
Set the information structure for this view.
- `int set_viewport (int scr_x, int scr_y, int w, int h, unsigned long buf_offset) const noexcept`
Set the position of the view in the [Goos](#).
- `int stack (View const &pivot, bool behind=true) const noexcept`
Move this view in the view stack.
- `int push_top () const noexcept`
Make this view the top-most view.
- `int push_bottom () const noexcept`
Push this view the back.
- `int refresh (int x, int y, int w, int h) const noexcept`
Refresh/Redraw the view.
- `bool valid () const`
Return whether this view is valid.

15.355.1 Detailed Description

[View](#) of a framebuffer.

A view is a rectangular subset of a framebuffer managed by a [Goos](#) object. The [Goos](#) orders multiple views in a stack which determines which view is on top in case they overlap. The view's pixel data is provided by a backing buffer, which must belong to the [Goos](#). It can be static or dynamically allocated, depending on the framebuffer.

See also

[L4Re::Video::Goos](#)

Definition at line 39 of file [goos](#).

15.355.2 Member Enumeration Documentation

15.355.2.1 Flags

enum [L4Re::Video::View::Flags](#)

[Flags](#) on a view.

Enumerator

| | |
|-----------------------------------|---|
| <code>F_none</code> | everything for this view is static (the VESA-FB case) |
| <code>F_set_buffer</code> | buffer object for this view can be changed |
| <code>F_set_buffer_offset</code> | buffer offset can be set |
| <code>F_set_bytes_per_line</code> | bytes per line can be set |

| | |
|------------------|--|
| F_set_pixel | pixel type can be set |
| F_set_position | position on screen can be set |
| F_dyn_allocated | View is dynamically allocated. |
| F_set_background | Set view as background for session. |
| F_set_flags | Set view flags (. See also V_flags) |
| F_fully_dynamic | Flags for a fully dynamic view. |

Definition at line 59 of file [goos](#).

15.355.2.2 V_flags

```
enum L4Re::Video::View::V_flags
```

Property flags of a view.

Such flags can be set or deleted with the [F_set_flags](#) operation using the [set_info\(\)](#) method.

Enumerator

| | |
|--------------|--|
| F_above | Flag the view as stay on top. |
| F_flags_mask | Mask containing all possible property flags. |

Definition at line 82 of file [goos](#).

15.355.3 Member Function Documentation

15.355.3.1 info()

```
int L4Re::Video::View::info (
    Info * info) const [inline], [noexcept]
```

Return the view information of the view.

Parameters

| | | |
|-----|-------------|--------------------------------|
| out | <i>info</i> | Information structure pointer. |
|-----|-------------|--------------------------------|

Return values

| | |
|---|---------|
| 0 | Success |
|---|---------|

| | |
|----|-------|
| <0 | Error |
|----|-------|

Definition at line 367 of file [goos](#).

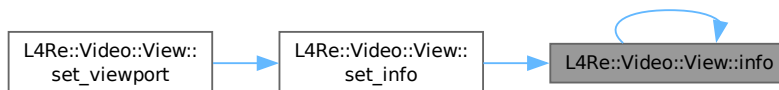
References [info\(\)](#).

Referenced by [info\(\)](#), and [set_info\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.355.3.2 refresh()

```

int L4Re::Video::View::refresh (
    int x,
    int y,
    int w,
    int h) const [inline], [noexcept]
  
```

Refresh/Redraw the view.

Parameters

| | |
|----------|-------------|
| <i>x</i> | X position. |
| <i>y</i> | Y position. |
| <i>w</i> | Width. |
| <i>h</i> | Height. |

Return values

| | |
|----|---------|
| 0 | Success |
| <0 | Error |

Definition at line 379 of file [goos](#).

15.355.3.3 set_info()

```
int L4Re::Video::View::set_info (  
    Info const & info) const [inline], [noexcept]
```

Set the information structure for this view.

Parameters

| | |
|-------------|------------------------|
| <i>info</i> | Information structure. |
|-------------|------------------------|

Return values

| | |
|----|---------|
| 0 | Success |
| <0 | Error |

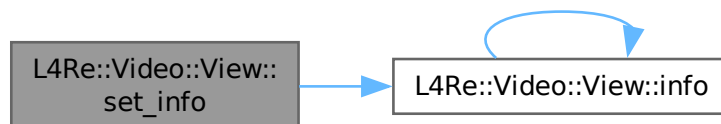
The function will also set the view port according to the values given in the information structure.

Definition at line 371 of file [goos](#).

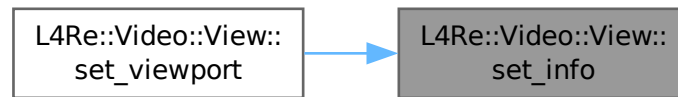
References [info\(\)](#).

Referenced by [set_viewport\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.355.3.4 set_viewport()

```

int L4Re::Video::View::set_viewport (
    int scr_x,
    int scr_y,
    int w,
    int h,
    unsigned long buf_offset) const [inline], [noexcept]
  
```

Set the position of the view in the [Goos](#).

Parameters

| | |
|-------------------|-------------------------------|
| <i>scr_x</i> | X position |
| <i>scr_y</i> | Y position |
| <i>w</i> | Width |
| <i>h</i> | Height |
| <i>buf_offset</i> | Offset in the buffer in bytes |

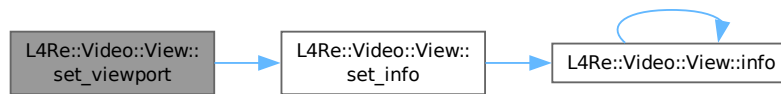
Return values

| | |
|----|---------|
| 0 | Success |
| <0 | Error |

Definition at line [383](#) of file [goos](#).

References [L4Re::Video::View::Info::buffer_index](#), [L4Re::Video::View::Info::buffer_offset](#), [L4Re::Video::View::Info::bytes_per_line](#), [F_set_buffer_offset](#), [F_set_position](#), [L4Re::Video::View::Info::flags](#), [L4Re::Video::View::Info::height](#), [L4Re::Video::View::Info::pixel_info](#), [set_info\(\)](#), [L4Re::Video::View::Info::view_index](#), [L4Re::Video::View::Info::width](#), [L4Re::Video::View::Info::xpos](#), and [L4Re::Video::View::Info::ypos](#).

Here is the call graph for this function:



15.355.3.5 stack()

```
int L4Re::Video::View::stack (
    View const & pivot,
    bool behind = true) const [inline], [noexcept]
```

Move this view in the view stack.

Parameters

| | |
|---------------|--|
| <i>pivot</i> | View to move relative to |
| <i>behind</i> | When true move the view behind the pivot view, if false move the view before the pivot view. |

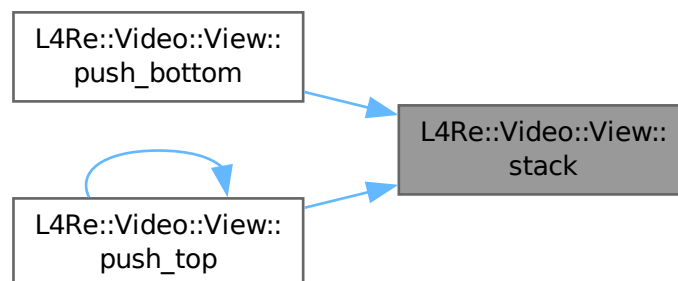
Return values

| | |
|----|---------|
| 0 | Success |
| <0 | Error |

Definition at line 375 of file [goos](#).

Referenced by [push_bottom\(\)](#), and [push_top\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

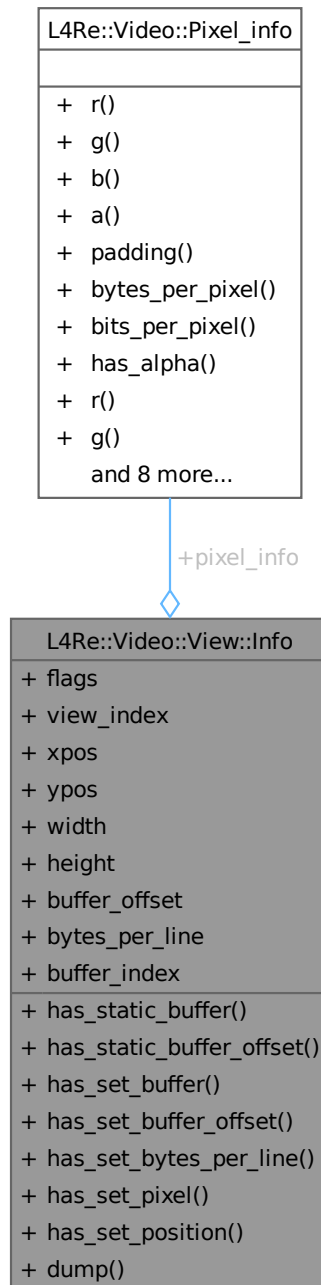
- `I4/re/video/goos`

15.356 L4Re::Video::View::Info Struct Reference

Information structure of a view.

```
#include <goos>
```

Collaboration diagram for L4Re::Video::View::Info:



Public Member Functions

- bool **has_static_buffer** () const
Return whether the view has a static buffer.
- bool **has_static_buffer_offset** () const
Return whether the static buffer offset is available.
- bool **has_set_buffer** () const
Return whether a buffer is set.
- bool **has_set_buffer_offset** () const
Return whether the given buffer offset is valid.
- bool **has_set_bytes_per_line** () const
Return whether the given bytes-per-line value is valid.
- bool **has_set_pixel** () const
Return whether the given pixel information is valid.
- bool **has_set_position** () const
Return whether the position information given is valid.
- template<typename OUT>
void **dump** (OUT &s) const
Dump information on the view information to a stream.

Data Fields

- unsigned **flags** = 0
Flags, see [Flags](#) and [V_flags](#).
- unsigned **view_index** = 0
Index of the view.
- unsigned long **xpos** = 0
X position in pixels of the view in the [Goos](#).
- unsigned long **ypos** = 0
Y position in pixels of the view in the [Goos](#).
- unsigned long **width** = 0
Width of the view in pixels.
- unsigned long **height** = 0
Height of the view in pixels.
- unsigned long **buffer_offset** = 0
Offset in the memory buffer in bytes.
- unsigned long **bytes_per_line** = 0
Bytes per line.
- [Pixel_info](#) **pixel_info**
Pixel information.
- unsigned **buffer_index** = 0
Number of the buffer used for this view.

15.356.1 Detailed Description

Information structure of a view.

Definition at line 91 of file [goos](#).

The documentation for this struct was generated from the following file:

- l4/re/video/goos

15.357 l4re_aux_t Struct Reference

Auxiliary descriptor.

```
#include <l4aux.h>
```

Collaboration diagram for l4re_aux_t:

| l4re_aux_t |
|-------------|
| + binary |
| + kip_ds |
| + dbg_lvl |
| + ldr_flags |
| + ldr_base |
| |

Data Fields

- char const * **binary**
Binary name.
- [l4_cap_idx_t](#) **kip_ds**
Data space of the KIP.
- [l4_umword_t](#) **dbg_lvl**
Debug levels for l4re.
- [l4_umword_t](#) **ldr_flags**
Flags for l4re, see [l4re_aux_ldr_flags_t](#).
- [l4_addr_t](#) **ldr_base**
Load offset of executable.

15.357.1 Detailed Description

Auxiliary descriptor.

Definition at line 40 of file [l4aux.h](#).

The documentation for this struct was generated from the following file:

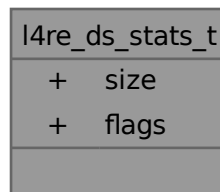
- [l4/re/l4aux.h](#)

15.358 l4re_ds_stats_t Struct Reference

Information about the data space.

```
#include <dataspace.h>
```

Collaboration diagram for l4re_ds_stats_t:



Data Fields

- l4re_ds_size_t **size**
size
- l4re_ds_flags_t **flags**
flags

15.358.1 Detailed Description

Information about the data space.

Definition at line 39 of file [dataspace.h](#).

The documentation for this struct was generated from the following file:

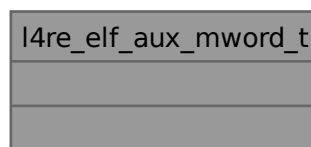
- l4/re/c/[dataspace.h](#)

15.359 l4re_elf_aux_mword_t Struct Reference

Auxiliary vector element for a single unsigned data word.

```
#include <elf_aux.h>
```

Collaboration diagram for l4re_elf_aux_mword_t:



15.359.1 Detailed Description

Auxiliary vector element for a single unsigned data word.

Definition at line 118 of file [elf_aux.h](#).

The documentation for this struct was generated from the following file:

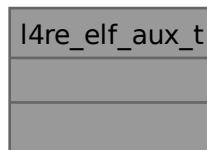
- [l4/re/elf_aux.h](#)

15.360 l4re_elf_aux_t Struct Reference

Generic header for each auxiliary vector element.

```
#include <elf_aux.h>
```

Collaboration diagram for l4re_elf_aux_t:



15.360.1 Detailed Description

Generic header for each auxiliary vector element.

Definition at line 98 of file [elf_aux.h](#).

The documentation for this struct was generated from the following file:

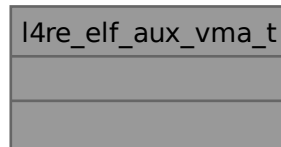
- [l4/re/elf_aux.h](#)

15.361 l4re_elf_aux_vma_t Struct Reference

Auxiliary vector element for a reserved virtual memory area.

```
#include <elf_aux.h>
```

Collaboration diagram for l4re_elf_aux_vma_t:



15.361.1 Detailed Description

Auxiliary vector element for a reserved virtual memory area.

Definition at line 107 of file [elf_aux.h](#).

The documentation for this struct was generated from the following file:

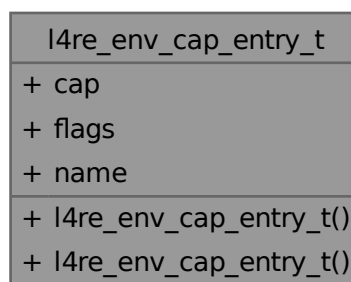
- [l4/re/elf_aux.h](#)

15.362 l4re_env_cap_entry_t Struct Reference

Entry in the [L4Re](#) environment array for the named initial objects.

```
#include <env.h>
```

Collaboration diagram for l4re_env_cap_entry_t:



Public Member Functions

- [l4re_env_cap_entry_t\(\)](#) [L4_NOTHROW](#)
Create an invalid entry.
- [l4re_env_cap_entry_t](#)(char const *n, [l4_cap_idx_t](#) c, [l4_umword_t](#) f=0) [L4_NOTHROW](#)
Create an entry with the name n, capability c, and flags f.

Data Fields

- [l4_cap_idx_t](#) **cap**
The capability selector for the object.
- [l4_umword_t](#) **flags**
Flags for the object.
- char **name** [16]
The name of the object.

15.362.1 Detailed Description

Entry in the [L4Re](#) environment array for the named initial objects.

Definition at line 43 of file [env.h](#).

15.362.2 Constructor & Destructor Documentation

15.362.2.1 l4re_env_cap_entry_t()

```
l4re_env_cap_entry_t::l4re_env_cap_entry_t (
    char const * n,
    l4_cap_idx_t c,
    l4_umword_t f = 0) [inline]
```

Create an entry with the name *n*, capability *c*, and flags *f*.

Parameters

| | |
|----------|--|
| <i>n</i> | is the name of the initial object. |
| <i>c</i> | is the capability selector that refers the initial object. |
| <i>f</i> | are the additional flags for the object. |

Definition at line 74 of file [env.h](#).

References [cap](#), [flags](#), [L4_NOTHROW](#), and [name](#).

15.362.3 Field Documentation

15.362.3.1 flags

`l4_umword_t l4re_env_cap_entry_t::flags`

Flags for the object.

Note

Currently unused, except as an end marker for the entry list.

Definition at line 54 of file [env.h](#).

Referenced by [l4re_env_cap_entry_t\(\)](#), [l4re_env_cap_entry_t\(\)](#), and [l4re_env_get_cap_l\(\)](#).

The documentation for this struct was generated from the following file:

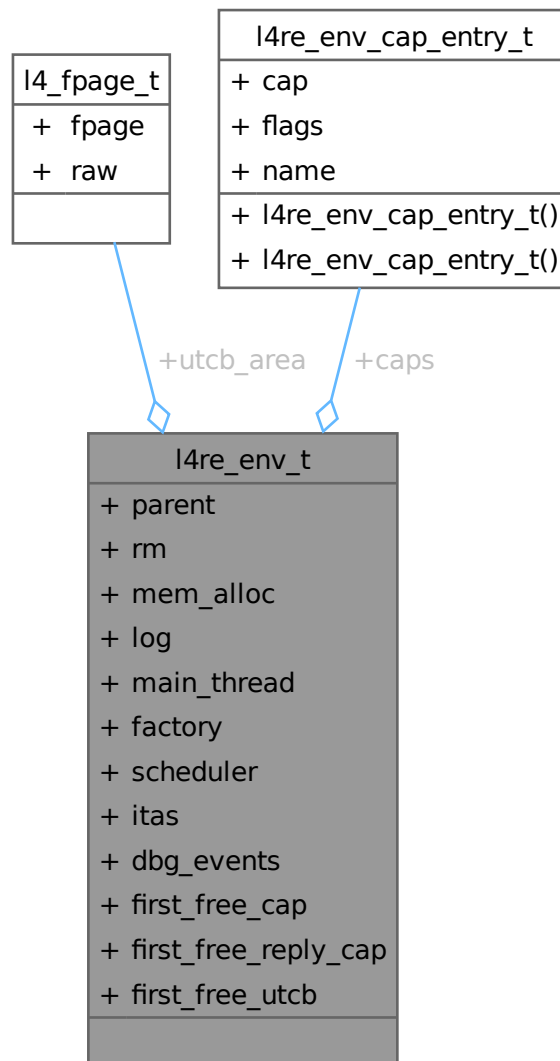
- [l4/re/env.h](#)

15.363 l4re_env_t Struct Reference

Initial environment data structure.

```
#include <env.h>
```

Collaboration diagram for `l4re_env_t`:



Data Fields

- [l4_cap_idx_t](#) **parent**
Parent object-capability.
- [l4_cap_idx_t](#) **rm**
Region map object-capability.
- [l4_cap_idx_t](#) **mem_alloc**
Memory allocator object-capability.
- [l4_cap_idx_t](#) **log**
Logging object-capability.
- [l4_cap_idx_t](#) **main_thread**
Object-capability of the first user thread.

- [l4_cap_idx_t](#) **factory**
Object-capability of the factory available to the task.
- [l4_cap_idx_t](#) **scheduler**
Object capability for the scheduler set to use.
- [l4_cap_idx_t](#) **itas**
ITAS services object-capability.
- [l4_cap_idx_t](#) **dbg_events**
Object-capability of the debug events service.
- [l4_cap_idx_t](#) **first_free_cap**
First capability index available to the application.
- [l4_umword_t](#) **first_free_reply_cap**
First reply capability index available to the application.
- [l4_fpage_t](#) **utcb_area**
UTCB area of the task.
- [l4_addr_t](#) **first_free_utcb**
First UTCB within the UTCB area available to the application.
- [l4re_env_cap_entry_t](#) * **caps**
Pointer to the first entry in the initial objects array which contains [l4re_env_cap_entry_t](#) elements.

15.363.1 Detailed Description

Initial environment data structure.

See also

[Initial environment](#)

Definition at line 111 of file [env.h](#).

15.363.2 Field Documentation

15.363.2.1 caps

```
l4re\_env\_cap\_entry\_t* l4re\_env\_t::caps
```

Pointer to the first entry in the initial objects array which contains [l4re_env_cap_entry_t](#) elements.

The array is terminated by an invalid entry with a `flags` value of `~0u1`.

Definition at line 131 of file [env.h](#).

The documentation for this struct was generated from the following file:

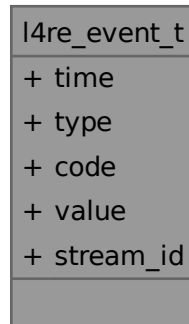
- [l4/re/env.h](#)

15.364 l4re_event_t Struct Reference

Event structure used in buffer.

```
#include <event.h>
```

Collaboration diagram for l4re_event_t:



Data Fields

- long long **time**
Time stamp of the event.
- unsigned short **type**
Type of the event.
- unsigned short **code**
Code of the event.
- int **value**
Value of the event.
- [l4_umword_t](#) **stream_id**
Stream ID.

15.364.1 Detailed Description

Event structure used in buffer.

Definition at line 30 of file [event.h](#).

The documentation for this struct was generated from the following file:

- [l4/re/c/event.h](#)

15.365 l4re_video_color_component_t Struct Reference

Color component structure.

```
#include <colors.h>
```

Collaboration diagram for l4re_video_color_component_t:

| l4re_video_color_component_t | |
|------------------------------|-------|
| + | size |
| + | shift |
| | |

Data Fields

- unsigned char **size**
Size in bits.
- unsigned char **shift**
offset in pixel

15.365.1 Detailed Description

Color component structure.

Definition at line 20 of file [colors.h](#).

The documentation for this struct was generated from the following file:

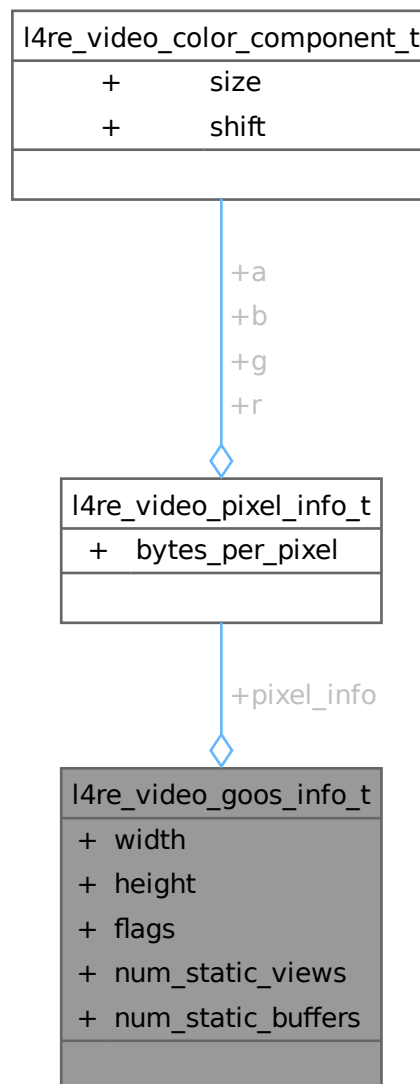
- [l4re/c/video/colors.h](#)

15.366 l4re_video_goos_info_t Struct Reference

Goos information structure.

```
#include <goos.h>
```

Collaboration diagram for `l4re_video_goos_info_t`:



Data Fields

- unsigned long **width**
Width of the goos.
- unsigned long **height**
Height of the goos.
- unsigned **flags**
Flags of the framebuffer, see [l4re_video_goos_info_flags_t](#).
- unsigned **num_static_views**
Number of static views.
- unsigned **num_static_buffers**

Number of static buffers.

- [l4re_video_pixel_info_t pixel_info](#)

Pixel layout of the goos.

15.366.1 Detailed Description

Goos information structure.

Definition at line 41 of file [goos.h](#).

The documentation for this struct was generated from the following file:

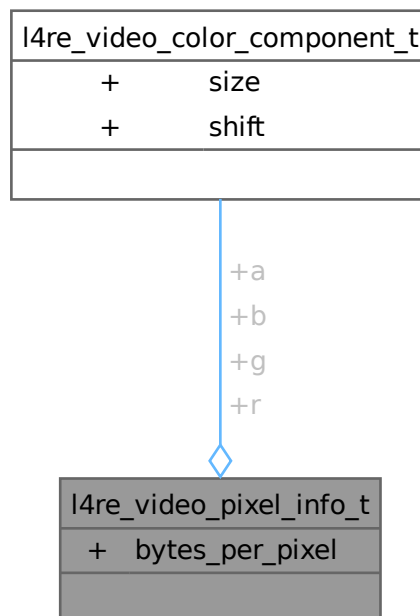
- [l4re/c/video/goos.h](#)

15.367 l4re_video_pixel_info_t Struct Reference

Pixel_info structure.

```
#include <colors.h>
```

Collaboration diagram for l4re_video_pixel_info_t:



Data Fields

- [l4re_video_color_component_t a](#)
Colors.
- unsigned char **bytes_per_pixel**
Bytes per pixel.

15.367.1 Detailed Description

Pixel_info structure.

Definition at line 30 of file [colors.h](#).

The documentation for this struct was generated from the following file:

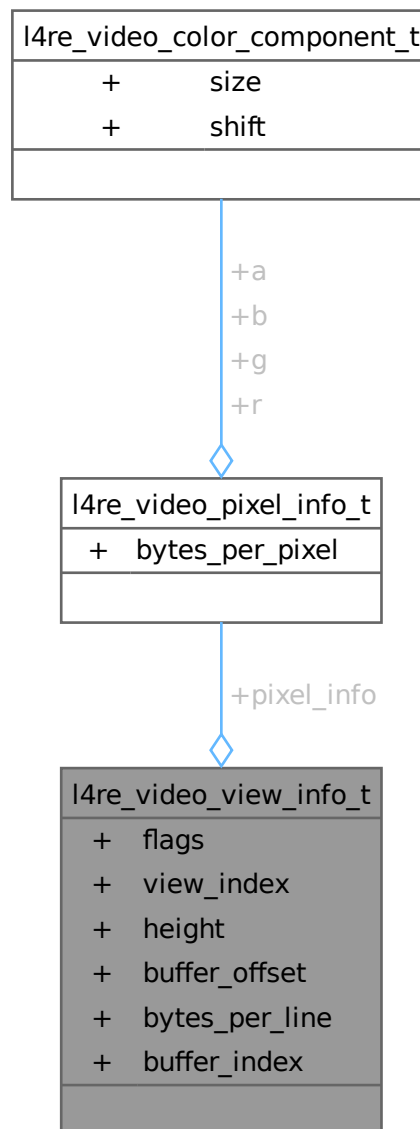
- [l4re/c/video/colors.h](#)

15.368 l4re_video_view_info_t Struct Reference

View information structure.

```
#include <view.h>
```


Collaboration diagram for l4re_video_view_info_t:



Data Fields

- unsigned **flags**
Flags.
- unsigned **view_index**
Number of view in the goos.
- unsigned long **height**
Position in goos and size of view.
- unsigned long **buffer_offset**
Memory offset in goos buffer.

- unsigned long **bytes_per_line**
Size of line in view.
- [l4re_video_pixel_info_t](#) **pixel_info**
Pixel info.
- unsigned **buffer_index**
Number of buffer of goos.

15.368.1 Detailed Description

View information structure.

Definition at line 49 of file [view.h](#).

The documentation for this struct was generated from the following file:

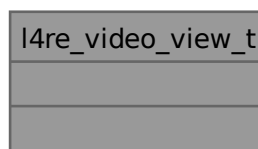
- [l4re/c/video/view.h](#)

15.369 l4re_video_view_t Struct Reference

C representation of a goos view.

```
#include <view.h>
```

Collaboration diagram for `l4re_video_view_t`:



15.369.1 Detailed Description

C representation of a goos view.

A view is a visible rectangle that provides a view to the contents of a buffer (frame buffer) memory object and is placed on a real screen.

Definition at line 68 of file [view.h](#).

The documentation for this struct was generated from the following file:

- [l4re/c/video/view.h](#)

15.370 l4shmc_ringbuf_head_t Struct Reference

Head field of a ring buffer.

```
#include <ringbuf.h>
```

Collaboration diagram for l4shmc_ringbuf_head_t:

| l4shmc_ringbuf_head_t |
|-----------------------|
| + next_read |
| + next_write |
| + bytes_filled |
| + sender_waits |
| + data |

Data Fields

- unsigned **next_read**
offset to next read packet
- unsigned **next_write**
offset to next write packet
- unsigned **bytes_filled**
bytes filled in buffer
- unsigned **sender_waits**
sender waiting?
- char **data** []
tail pointer -> data

15.370.1 Detailed Description

Head field of a ring buffer.

Definition at line 59 of file [ringbuf.h](#).

The documentation for this struct was generated from the following file:

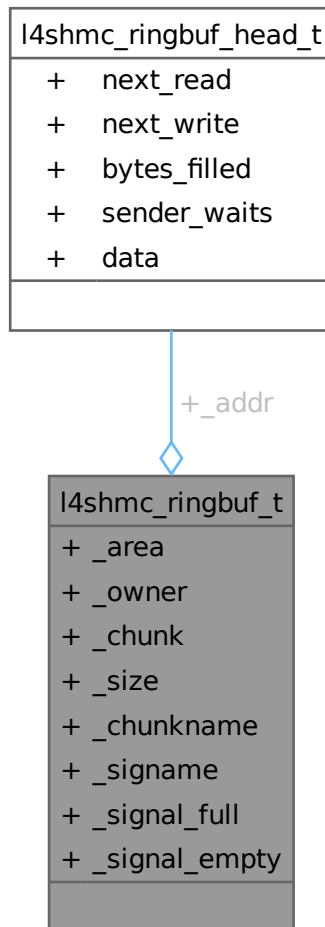
- [l4/shmc/ringbuf.h](#)

15.371 l4shmc_ringbuf_t Struct Reference

Ring buffer.

```
#include <ringbuf.h>
```

Collaboration diagram for l4shmc_ringbuf_t:



Data Fields

- l4shmc_area_t * **_area**
L4SHM area this buffer is located in.
- l4_cap_idx_t **_owner**
owner (attached to send/recv signal)
- l4shmc_chunk_t **_chunk**
chunk descriptor
- unsigned **_size**

- chunk size // XXX do we need this?*
- `char * _chunkname`
name of the ring buffer chunk
- `char * _signame`
base name of the ring buffer signals
- `l4shmc_ringbuf_head_t * _addr`
pointer to ring buffer head
- `l4shmc_signal_t _signal_full`
"full" signal - triggered when data is produced
- `l4shmc_signal_t _signal_empty`
"empty" signal - triggered when data is consumed

15.371.1 Detailed Description

Ring buffer.

Definition at line 85 of file [ringbuf.h](#).

The documentation for this struct was generated from the following file:

- [l4/shmc/ringbuf.h](#)

15.372 l4util_l4mod_info Struct Reference

Base module structure.

```
#include <l4mod.h>
```

Collaboration diagram for l4util_l4mod_info:

| l4util_l4mod_info |
|-------------------|
| + flags |
| + cmdline |
| + mods_addr |
| + mods_count |
| + vbe_ctrl_info |
| + vbe_mode_info |
| |

Data Fields

- [l4_uint64_t](#) **flags**
Flags.
- [l4_uint64_t](#) **cmdline**
Pointer to kernel command line.
- [l4_uint64_t](#) **mods_addr**
Module list.
- [l4_uint32_t](#) **mods_count**
Number of modules.
- [l4_uint64_t](#) **vbe_ctrl_info**
VESA video info, valid if one of vbe_ctrl_info or vbe_mode_info is not zero.
- [l4_uint64_t](#) **vbe_mode_info**
VESA video mode info.

15.372.1 Detailed Description

Base module structure.

Definition at line 36 of file [l4mod.h](#).

15.372.2 Field Documentation

15.372.2.1 vbe_ctrl_info

[l4_uint64_t](#) l4util_l4mod_info::vbe_ctrl_info

VESA video info, valid if one of vbe_ctrl_info or vbe_mode_info is not zero.

VESA video controller info

Definition at line 48 of file [l4mod.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/l4mod.h](#)

15.373 l4util_l4mod_mod Struct Reference

A single module.

```
#include <l4mod.h>
```

Collaboration diagram for l4util_l4mod_mod:

| l4util_l4mod_mod |
|------------------|
| + flags |
| + mod_start |
| + mod_end |
| + cmdline |
| |

Data Fields

- [l4_uint64_t](#) **flags**
Module flags ([l4util_l4mod_mod_info_flag](#)).
- [l4_uint64_t](#) **mod_start**
Starting address of module in memory.
- [l4_uint64_t](#) **mod_end**
End address of module in memory.
- [l4_uint64_t](#) **cmdline**
Module command line.

15.373.1 Detailed Description

A single module.

Definition at line 27 of file [l4mod.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/l4mod.h](#)

15.374 l4util_mb_addr_range_t Struct Reference

INT-15, AX=E820 style "AddressRangeDescriptor" ...with a "size" parameter on the front which is the structure size - 4, pointing to the next one, up until the full buffer length of the memory map has been reached.

```
#include <mb_info.h>
```

Collaboration diagram for l4util_mb_addr_range_t:

| l4util_mb_addr_range_t | |
|------------------------|-------------|
| + | struct_size |
| + | addr |
| + | size |
| + | type |
| | |

Data Fields

- [l4_uint32_t](#) **struct_size**
Size of structure.
- [l4_uint64_t](#) **addr**
Start address.
- [l4_uint64_t](#) **size**
Size of memory range.
- [l4_uint32_t](#) **type**
type of memory range

15.374.1 Detailed Description

INT-15, AX=E820 style "AddressRangeDescriptor" ...with a "size" parameter on the front which is the structure size - 4, pointing to the next one, up until the full buffer length of the memory map has been reached.

Definition at line 48 of file [mb_info.h](#).

The documentation for this struct was generated from the following file:

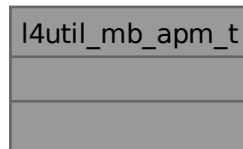
- [l4/util/mb_info.h](#)

15.375 l4util_mb_apm_t Struct Reference

APM BIOS info.

```
#include <mb_info.h>
```

Collaboration diagram for l4util_mb_apm_t:



15.375.1 Detailed Description

APM BIOS info.

Definition at line 90 of file [mb_info.h](#).

The documentation for this struct was generated from the following file:

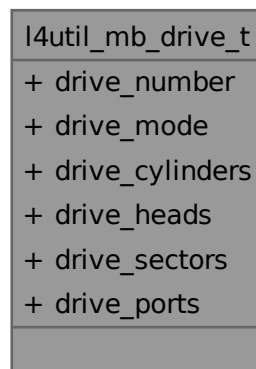
- [l4/util/mb_info.h](#)

15.376 l4util_mb_drive_t Struct Reference

Drive Info structure.

```
#include <mb_info.h>
```

Collaboration diagram for l4util_mb_drive_t:



Data Fields

- [l4_uint8_t](#) **drive_number**
< The size of this structure.
- [l4_uint8_t](#) **drive_mode**
< The BIOS drive number.
- [l4_uint16_t](#) **drive_cylinders**
< The access mode (see below).
- [l4_uint8_t](#) **drive_heads**
< number of cylinders
- [l4_uint8_t](#) **drive_sectors**
< number of heads
- [l4_uint16_t](#) **drive_ports** [0]
< number of sectors per track

15.376.1 Detailed Description

Drive Info structure.

Definition at line 73 of file [mb_info.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/mb_info.h](#)

15.377 l4util_mb_info_t Struct Reference

MultiBoot Info description.

```
#include <mb_info.h>
```

Collaboration diagram for `l4util_mb_info_t`:

| <code>l4util_mb_info_t</code> |
|-------------------------------|
| + flags |
| + mem_lower |
| + mem_upper |
| + boot_device |
| + cmdline |
| + mods_count |
| + mods_addr |
| + tabsize |
| + num |
| + mmap_length |
| and 12 more... |

Data Fields

- [l4_uint32_t flags](#)
MultiBoot info version number.
- [l4_uint32_t mem_lower](#)
available memory below 1MB
- [l4_uint32_t mem_upper](#)
available memory starting from 1MB [KB]
- [l4_uint32_t boot_device](#)
"root" partition
- [l4_uint32_t cmdline](#)
Kernel command line.
- [l4_uint32_t mods_count](#)
number of modules
- [l4_uint32_t mods_addr](#)
module list
- [l4_uint32_t mmap_length](#)
size of memory mapping buffer
- [l4_uint32_t mmap_addr](#)
address of memory mapping buffer
- [l4_uint32_t drives_length](#)
size of drive info buffer
- [l4_uint32_t drives_addr](#)
address of driver info buffer
- [l4_uint32_t config_table](#)
ROM configuration table.
- [l4_uint32_t boot_loader_name](#)
Boot Loader Name.
- [l4_uint32_t apm_table](#)
APM table.
- [l4_uint32_t vbe_ctrl_info](#)
VESA video controller info.
- [l4_uint32_t vbe_mode_info](#)
VESA video mode info.
- [l4_uint16_t vbe_mode](#)
VESA video mode number.
- [l4_uint16_t vbe_interface_seg](#)
VESA segment of prot BIOS interface.
- [l4_uint16_t vbe_interface_off](#)
VESA offset of prot BIOS interface.
- [l4_uint16_t vbe_interface_len](#)
VESA lenght of prot BIOS interface.

15.377.1 Detailed Description

MultiBoot Info description.

This is the struct passed to the boot image. This is done by placing its address in the EAX register.

Definition at line 247 of file [mb_info.h](#).

The documentation for this struct was generated from the following file:

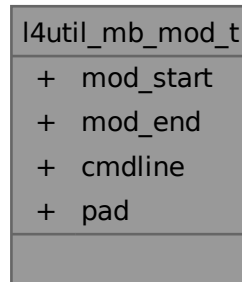
- [l4/util/mb_info.h](#)

15.378 l4util_mb_mod_t Struct Reference

The structure type "mod_list" is used by the [multiboot_info](#) structure.

```
#include <mb_info.h>
```

Collaboration diagram for l4util_mb_mod_t:



Data Fields

- [l4_uint32_t](#) **mod_start**
Starting address of module in memory.
- [l4_uint32_t](#) **mod_end**
End address of module in memory.
- [l4_uint32_t](#) **cmdline**
Module command line.
- [l4_uint32_t](#) **pad**
padding to take it to 16 bytes

15.378.1 Detailed Description

The structure type "mod_list" is used by the [multiboot_info](#) structure.

Definition at line 33 of file [mb_info.h](#).

The documentation for this struct was generated from the following file:

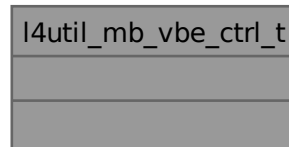
- [l4/util/mb_info.h](#)

15.379 l4util_mb_vbe_ctrl_t Struct Reference

VBE controller information.

```
#include <mb_info.h>
```

Collaboration diagram for l4util_mb_vbe_ctrl_t:



15.379.1 Detailed Description

VBE controller information.

Definition at line 106 of file [mb_info.h](#).

The documentation for this struct was generated from the following file:

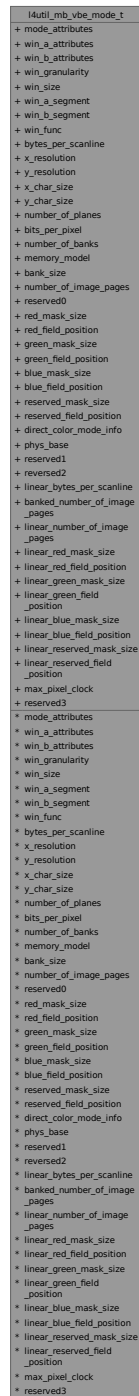
- [l4/util/mb_info.h](#)

15.380 l4util_mb_vbe_mode_t Struct Reference

VBE mode information.

```
#include <mb_info.h>
```

Collaboration diagram for `l4util_mb_vbe_mode_t`:



Data Fields

all VESA versions

- [l4_uint16_t mode_attributes](#)
Mode attributes.
- [l4_uint8_t win_a_attributes](#)
Window A attributes.

- [l4_uint8_t win_b_attributes](#)
Window B attributes.
- [l4_uint16_t win_granularity](#)
Window granularity.
- [l4_uint16_t win_size](#)
Window size.
- [l4_uint16_t win_a_segment](#)
Window A start segment.
- [l4_uint16_t win_b_segment](#)
Window B start segment.
- [l4_uint32_t win_func](#)
Real mode pointer to window function.
- [l4_uint16_t bytes_per_scanline](#)
Bytes per scan line.

>= VESA version 1.2

- [l4_uint16_t x_resolution](#)
Horizontal resolution in pixels or characters.
- [l4_uint16_t y_resolution](#)
Vertical resolution in pixels or characters.
- [l4_uint8_t x_char_size](#)
Character cell width in pixels.
- [l4_uint8_t y_char_size](#)
Character cell height in pixels.
- [l4_uint8_t number_of_planes](#)
Number of memory planes.
- [l4_uint8_t bits_per_pixel](#)
Bits per pixel.
- [l4_uint8_t number_of_banks](#)
Number of banks.
- [l4_uint8_t memory_model](#)
Memory model type.
- [l4_uint8_t bank_size](#)
Bank size in KiB.
- [l4_uint8_t number_of_image_pages](#)
Number of images.
- [l4_uint8_t reserved0](#)
Reserved for page function.

direct color

- [l4_uint8_t red_mask_size](#)
Size of direct color red mask in bits.
- [l4_uint8_t red_field_position](#)
Bit position of LSB of red mask.
- [l4_uint8_t green_mask_size](#)
Size of direct color green mask in bits.
- [l4_uint8_t green_field_position](#)
Bit position of LSB of green mask.
- [l4_uint8_t blue_mask_size](#)
Size of direct color blue mask in bits.
- [l4_uint8_t blue_field_position](#)
Bit position of LSB of blue mask.
- [l4_uint8_t reserved_mask_size](#)
Size of direct color reserved mask in bits.
- [l4_uint8_t reserved_field_position](#)
Bit position of LSB of reserved mask.

- [l4_uint8_t direct_color_mode_info](#)

Direct color mode attributes.

>= VESA version 2.0

- [l4_uint32_t phys_base](#)
Physical address for flat memory memory frame buffer.
- [l4_uint32_t reserved1](#)
Reserved – always set to 0.
- [l4_uint16_t reversed2](#)
Reserved – always set to 0.

>= VESA version 3.0

- [l4_uint16_t linear_bytes_per_scanline](#)
Bytes per scan line for linear modes.
- [l4_uint8_t banked_number_of_image_pages](#)
Number of images for banked modes.
- [l4_uint8_t linear_number_of_image_pages](#)
Number of images for linear modes.
- [l4_uint8_t linear_red_mask_size](#)
Size of direct color red mask (linear modes).
- [l4_uint8_t linear_red_field_position](#)
Bit position of LSB of red mask (linear modes).
- [l4_uint8_t linear_green_mask_size](#)
Size of direct color green mask (linear modes).
- [l4_uint8_t linear_green_field_position](#)
Bit position of LSB of green mask (linear modes).
- [l4_uint8_t linear_blue_mask_size](#)
Size of direct color blue mask (linear modes).
- [l4_uint8_t linear_blue_field_position](#)
Bit position of LSB of blue mask (linear modes).
- [l4_uint8_t linear_reserved_mask_size](#)
Size of direct color reserved mask (linear modes).
- [l4_uint8_t linear_reserved_field_position](#)
Bit position of LSB of reserved mask (linear modes).
- [l4_uint32_t max_pixel_clock](#)
Maximum pixel clock (in Hz) for graphics mode.
- [l4_uint8_t reserved3](#) [190]
Reserved (padding).

15.380.1 Detailed Description

VBE mode information.

Definition at line 125 of file [mb_info.h](#).

The documentation for this struct was generated from the following file:

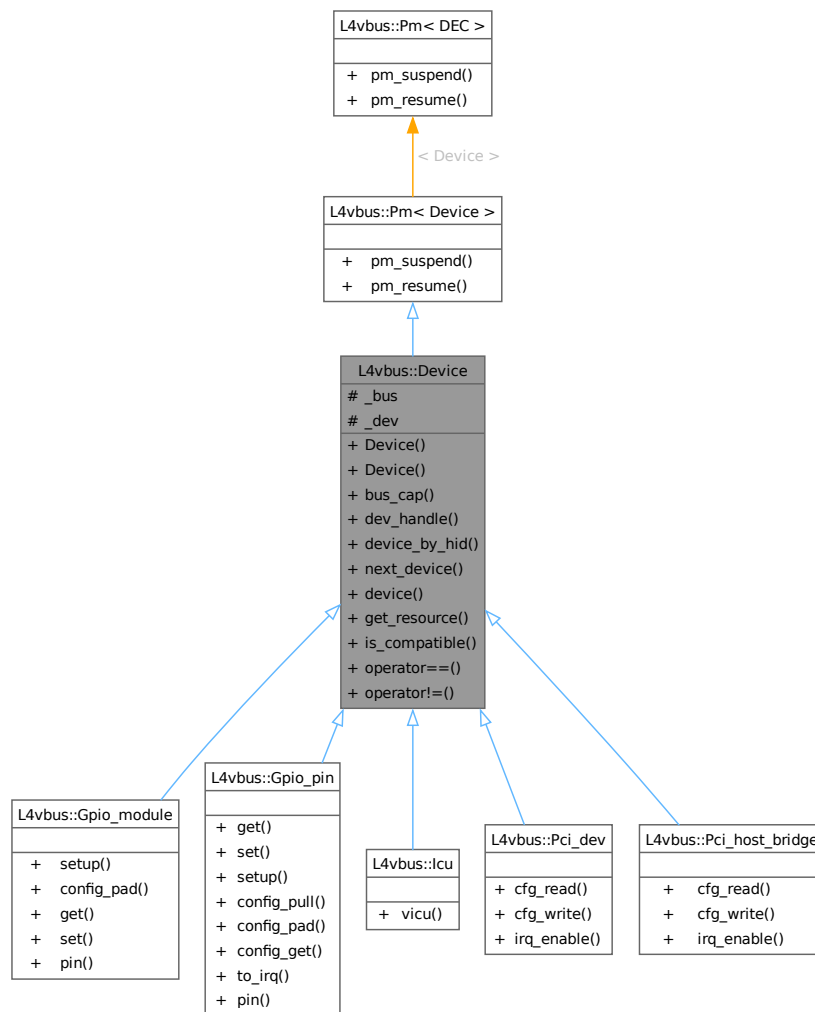
- [l4/util/mb_info.h](#)

15.381 L4vbus::Device Class Reference

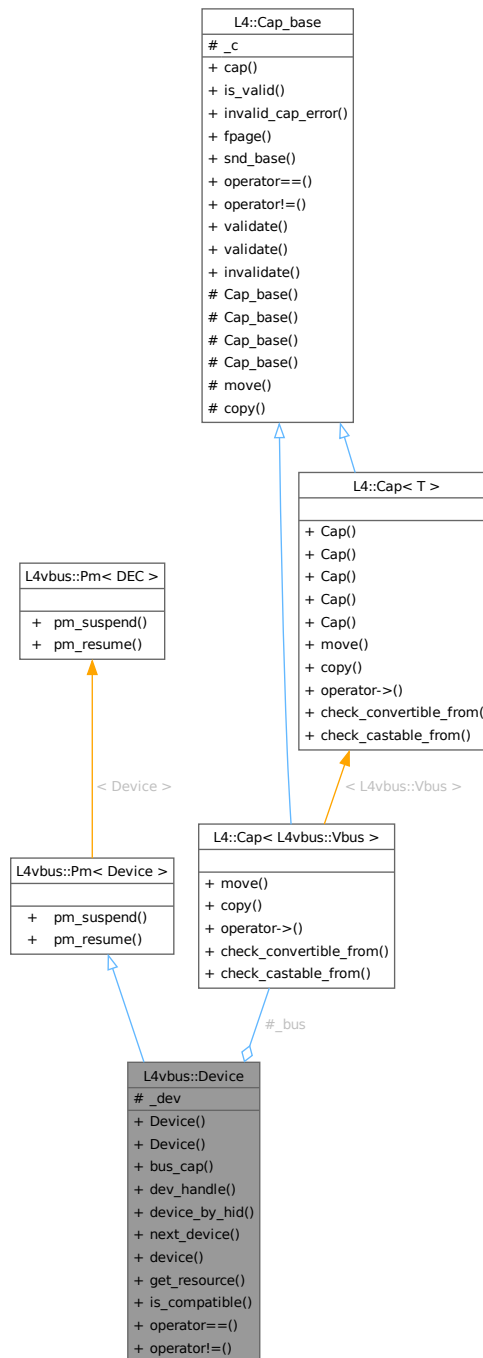
Device on a [L4vbus::Vbus](#).

```
#include <vbus>
```

Inheritance diagram for L4vbus::Device:



Collaboration diagram for L4vbus::Device:



Public Member Functions

- **Device ()**
Construct a new vbus device using the NULL device **L4VBUS_NULL**.
- **Device (L4::Cap< Vbus > bus, l4vbus_device_handle_t dev)**
Construct a new vbus device using a device handle.
- **L4::Cap< Vbus > bus_cap () const**

- Access the *Vbus* capability of the underlying virtual bus.

 - `l4vbus_device_handle_t dev_handle () const`

Access the device handle of this device.
- `int device_by_hid (Device *child, char const *hid, int depth=L4VBUS_MAX_DEPTH, l4vbus_device_t *devinfo=0) const`

Find a device by the hardware interface identifier (HID).
- `int next_device (Device *child, int depth=L4VBUS_MAX_DEPTH, l4vbus_device_t *devinfo=0) const`

Find next child following *child*.
- `int device (l4vbus_device_t *devinfo) const`

Obtain detailed information about a *Vbus* device.
- `int get_resource (unsigned res_idx, l4vbus_resource_t *res) const`

Obtain the resource description of an individual device resource.
- `int is_compatible (char const *cid) const`

Check if the given device has a compatibility ID (CID) or HID that matches *cid*.
- `bool operator== (Device const &o) const`

Test if two devices are the same *Vbus* device.
- `bool operator!= (Device const &o) const`

Test if two *Vbus* devices are not the same.

Public Member Functions inherited from L4vbus::Pm< Device >

- `int pm_suspend () const`

Suspend the device.
- `int pm_resume () const`

Resume the device.

Protected Attributes

- `L4::Cap< Vbus > _bus`
- The *Vbus* capability where this device is located on.
- `l4vbus_device_handle_t _dev`
- The device handle for this device.

15.381.1 Detailed Description

Device on a `L4vbus::Vbus`.

Definition at line 83 of file `vbus`.

15.381.2 Constructor & Destructor Documentation

15.381.2.1 Device()

```
L4vbus::Device::Device (
    L4::Cap< Vbus > bus,
    l4vbus_device_handle_t dev) [inline]
```

Construct a new vbus device using a device handle.

Specifying the special root bus device handle [L4VBUS_ROOT_BUS](#) forms the root device of the corresponding vbus, see [Vbus::root](#).

Parameters

| | |
|------------|--|
| <i>bus</i> | The vbus capability where this device is assigned. |
| <i>dev</i> | The device handle of the device. |

Definition at line [100](#) of file [vbus](#).

References [_bus](#), and [_dev](#).

15.381.3 Member Function Documentation

15.381.3.1 bus_cap()

```
L4::Cap< Vbus > L4vbus::Device::bus_cap () const [inline]
```

Access the [Vbus](#) capability of the underlying virtual bus.

Returns

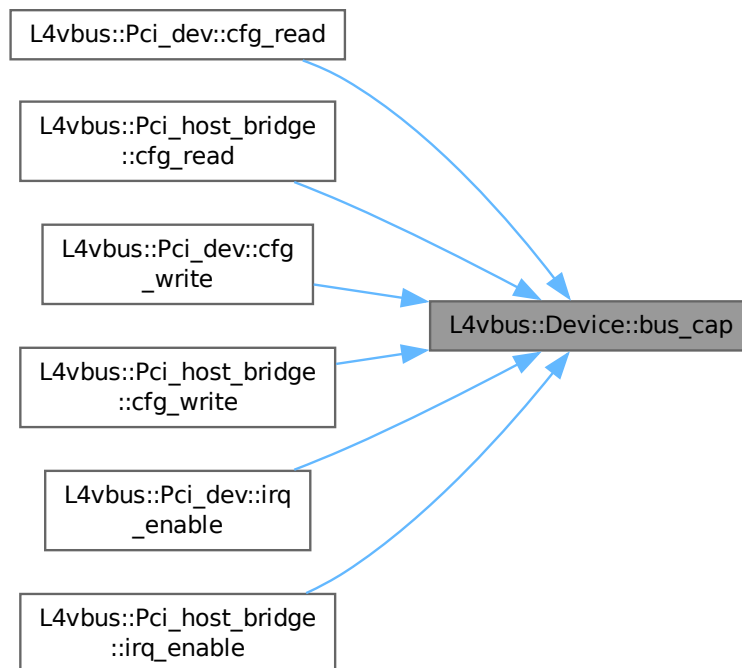
the capability to the underlying [Vbus](#).

Definition at line [107](#) of file [vbus](#).

References [_bus](#).

Referenced by [L4vbus::Pci_dev::cfg_read\(\)](#), [L4vbus::Pci_host_bridge::cfg_read\(\)](#), [L4vbus::Pci_dev::cfg_write\(\)](#), [L4vbus::Pci_host_bridge::cfg_write\(\)](#), [L4vbus::Pci_dev::irq_enable\(\)](#), and [L4vbus::Pci_host_bridge::irq_enable\(\)](#).

Here is the caller graph for this function:



15.381.3.2 dev_handle()

```
l4vbus_device_handle_t L4vbus::Device::dev_handle () const [inline]
```

Access the device handle of this device.

Returns

the device handle for this device.

The device handle is used to directly address the device on its virtual bus.

Definition at line 116 of file `vbus`.

References `_dev`.

15.381.3.3 device()

```
int L4vbus::Device::device (
    l4vbus_device_t * devinfo) const [inline]
```

Obtain detailed information about a [Vbus](#) device.

Parameters

| | | |
|-----|----------------|---|
| out | <i>devinfo</i> | Information structure which contains details about the device. The pointer might be NULL. |
|-----|----------------|---|

Return values

| | |
|------------|---|
| 0 | Success. |
| -L4_ENODEV | No device with the given device handle <i>dev</i> could be found. |

Definition at line [189](#) of file [vbus](#).

References [_bus](#), [_dev](#), and [l4vbus_get_device\(\)](#).

Here is the call graph for this function:



15.381.3.4 device_by_hid()

```
int L4vbus::Device::device_by_hid (
    Device * child,
    char const * hid,
    int depth = L4VBUS_MAX_DEPTH,
    l4vbus_device_t * devinfo = 0) const [inline]
```

Find a device by the hardware interface identifier (HID).

This function searches the vbus for a device with the given HID and returns a handle to the first matching device. The HID usually conforms to an ACPI HID or a Linux device tree compatible identifier.

It is possible to have multiple devices with the same HID on a vbus. In order to find all matching devices this function has to be called repeatedly with *child* pointing to the device found in the previous iteration. The iteration starts at *child* that might be any device node in the tree.

Parameters

| | | |
|---------|----------------|---|
| in, out | <i>child</i> | Handle of the device from where in the device tree the search should start. To start searching from the beginning <i>child</i> must be initialized using the default (L4VBUS_NULL). If a matching device is found, its handle is returned through this parameter. |
| | <i>hid</i> | HID of the device |
| | <i>depth</i> | Maximum depth for the recursive lookup |
| out | <i>devinfo</i> | Device information structure (might be NULL) |

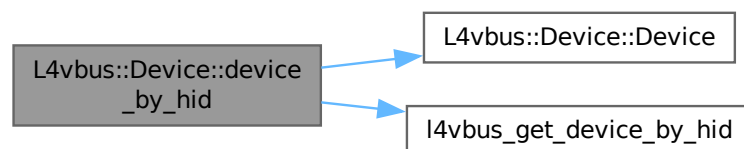
Return values

| | |
|-------------------------|--|
| ≥ 0 | A device with the given HID was found. |
| <code>-L4_ENOENT</code> | No device with the given HID could be found on the vbus. |
| <code>-L4_EINVAL</code> | Invalid or no HID provided. |
| <code>-L4_ENODEV</code> | Function called on a non-existing device. |

Definition at line [148](#) of file [vbus](#).

References [_bus](#), [_dev](#), [Device\(\)](#), and [l4vbus_get_device_by_hid\(\)](#).

Here is the call graph for this function:

**15.381.3.5 get_resource()**

```
int L4vbus::Device::get_resource (
    unsigned res_idx,
    l4vbus_resource_t * res) const [inline]
```

Obtain the resource description of an individual device resource.

Parameters

| | | |
|--|----------------|---|
| | <i>res_idx</i> | Index of the resource for which the resource description should be returned. The total number of resources for a device is available in the l4vbus_device_t structure that is returned by L4vbus::Device::device_by_hid() and L4vbus::Device::next_device() . |
|--|----------------|---|

| | | |
|-----|-----|-----------------------------|
| out | res | Descriptor of the resource. |
|-----|-----|-----------------------------|

This function returns the resource descriptor of an individual device resource selected by the `res_idx` parameter.

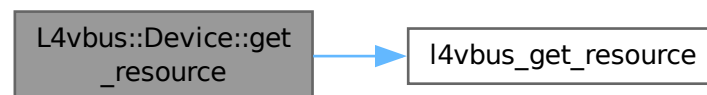
Return values

| | |
|------------|---|
| 0 | Success. |
| -L4_ENOENT | Invalid resource index <code>res_idx</code> . |

Definition at line 209 of file [vbus](#).

References [_bus](#), [_dev](#), and [l4vbus_get_resource\(\)](#).

Here is the call graph for this function:



15.381.3.6 is_compatible()

```
int L4vbus::Device::is_compatible (
    char const * cid) const [inline]
```

Check if the given device has a compatibility ID (CID) or HID that matches *cid*.

Parameters

| | |
|------------|------------------------------|
| <i>cid</i> | the compatibility ID to test |
|------------|------------------------------|

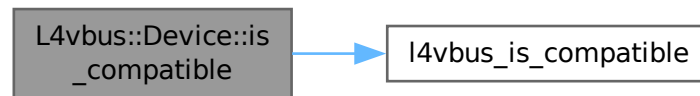
Returns

1 when the given ID (*cid*) matches this device, 0 when the given ID does not match, <0 on error.

Definition at line 223 of file [vbus](#).

References [_bus](#), [_dev](#), and [l4vbus_is_compatible\(\)](#).

Here is the call graph for this function:



15.381.3.7 next_device()

```

int L4vbus::Device::next_device (
    Device * child,
    int depth = L4VBUS_MAX_DEPTH,
    l4vbus_device_t * devinfo = 0) const [inline]
  
```

Find next child following `child`.

Parameters

| | | |
|---------|----------------|---|
| in, out | <i>child</i> | Handle of the device that precedes the device that shall be returned. To start from the beginning, <i>child</i> must be initialized with L4VBUS_NULL (Device::Device). If a device is found, its handle is returned through this parameter. |
| | <i>depth</i> | Depth to look for |
| out | <i>devinfo</i> | Device information (might be NULL) |

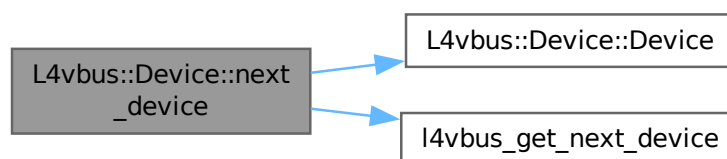
Returns

0 on success, else failure

Definition at line [171](#) of file [vbus](#).

References [_bus](#), [_dev](#), [Device\(\)](#), and [l4vbus_get_next_device\(\)](#).

Here is the call graph for this function:



15.381.3.8 operator!=(())

```
bool L4vbus::Device::operator!= (
    Device const & o) const [inline]
```

Test if two [Vbus](#) devices are not the same.

Returns

true if the two devices are different, false else.

Definition at line [239](#) of file [vbus](#).

References [_bus](#), [_dev](#), and [Device\(\)](#).

Here is the call graph for this function:



15.381.3.9 operator==(())

```
bool L4vbus::Device::operator== (
    Device const & o) const [inline]
```

Test if two devices are the same [Vbus](#) device.

Returns

true if the two devices are the same, false else.

Definition at line [230](#) of file [vbus](#).

References [_bus](#), [_dev](#), and [Device\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

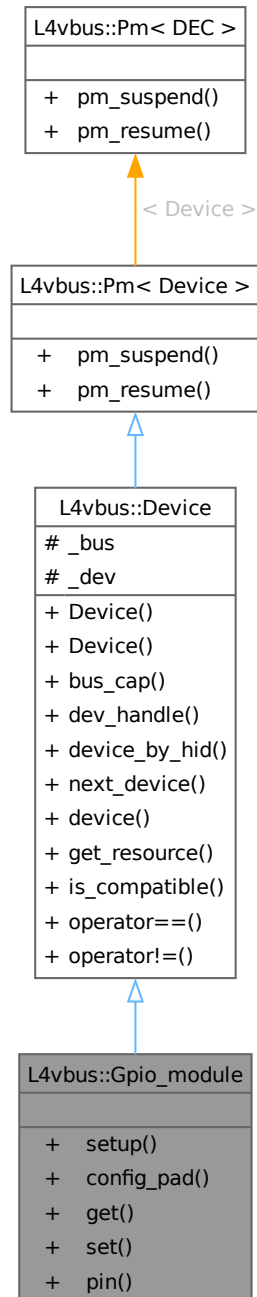
- [I4/vbus/vbus](#)

15.382 L4vbus::Gpio_module Class Reference

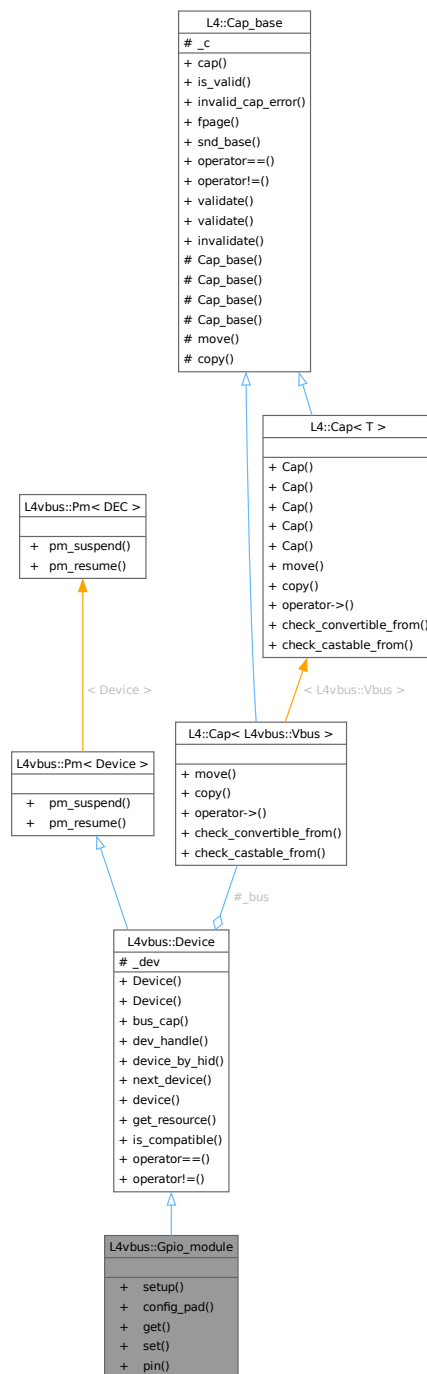
A [Gpio_module](#) groups multiple GPIO pins together.

```
#include <vbus_gpio>
```

Inheritance diagram for L4vbus::Gpio_module:



Collaboration diagram for L4vbus::Gpio_module:



Data Structures

- struct [Pin_slice](#)

A slice of the pins provided by this module.

Public Member Functions

- int [setup](#) ([Pin_slice](#) const &mask, unsigned mode, unsigned value) const

- *Configure function of multiple GPIO pins at once.*
int [config_pad](#) ([Pin_slice](#) const &mask, unsigned func, unsigned value) const
- *Hardware specific configuration function for multiple GPIO pins.*
int [get](#) (unsigned offset, unsigned *data) const
- *Read values of multiple GPIO pins at once.*
int [set](#) ([Pin_slice](#) const &mask, unsigned data)
- *Set multiple GPIO output pins at once.*
[Gpio_pin](#) [pin](#) (unsigned pin) const
- *Get [Gpio_pin](#) for a specific pin of this [Gpio_module](#).*

Public Member Functions inherited from [L4vbus::Device](#)

- [Device](#) ()
Construct a new vbus device using the NULL device [L4VBUS_NULL](#).
- [Device](#) ([L4::Cap](#)< [Vbus](#) > bus, [l4vbus_device_handle_t](#) dev)
Construct a new vbus device using a device handle.
- [L4::Cap](#)< [Vbus](#) > [bus_cap](#) () const
Access the [Vbus](#) capability of the underlying virtual bus.
- [l4vbus_device_handle_t](#) [dev_handle](#) () const
Access the device handle of this device.
- int [device_by_hid](#) ([Device](#) *child, char const *hid, int depth=[L4VBUS_MAX_DEPTH](#), [l4vbus_device_t](#) *devinfo=0) const
Find a device by the hardware interface identifier (HID).
- int [next_device](#) ([Device](#) *child, int depth=[L4VBUS_MAX_DEPTH](#), [l4vbus_device_t](#) *devinfo=0) const
Find next child following [child](#).
- int [device](#) ([l4vbus_device_t](#) *devinfo) const
Obtain detailed information about a [Vbus](#) device.
- int [get_resource](#) (unsigned res_idx, [l4vbus_resource_t](#) *res) const
Obtain the resource description of an individual device resource.
- int [is_compatible](#) (char const *cid) const
Check if the given device has a compatibility ID (CID) or HID that matches [cid](#).
- bool [operator==](#) ([Device](#) const &o) const
Test if two devices are the same [Vbus](#) device.
- bool [operator!=](#) ([Device](#) const &o) const
Test if two [Vbus](#) devices are not the same.

Public Member Functions inherited from [L4vbus::Pm](#)< [Device](#) >

- int [pm_suspend](#) () const
Suspend the device.
- int [pm_resume](#) () const
Resume the device.

Additional Inherited Members

Protected Attributes inherited from [L4vbus::Device](#)

- [L4::Cap](#)< [Vbus](#) > [_bus](#)
The [Vbus](#) capability where this device is located on.
- [l4vbus_device_handle_t](#) [_dev](#)
The device handle for this device.

15.382.1 Detailed Description

A [Gpio_module](#) groups multiple GPIO pins together.

Definition at line 133 of file [vbus_gpio](#).

15.382.2 Member Function Documentation

15.382.2.1 `config_pad()`

```
int L4vbus::Gpio_module::config_pad (  
    Pin_slice const & mask,  
    unsigned func,  
    unsigned value) const [inline]
```

Hardware specific configuration function for multiple GPIO pins.

Parameters

| | |
|--------------|--|
| <i>mask</i> | Mask of GPIO pins to configure. A bit set to 1 configures this pin. A maximum of 32 pins can be configured at once. The real number depends on the hardware and the driver implementation. |
| <i>func</i> | Hardware specific configuration register, usually offset to the GPIO chip's base address. |
| <i>value</i> | Value which is written into the hardware specific configuration register for the specified pins |

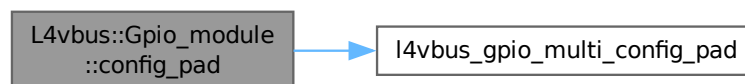
Returns

0 if OK, error code otherwise

Definition at line 185 of file [vbus_gpio](#).

References [L4vbus::Device::_bus](#), [L4vbus::Device::_dev](#), and [l4vbus_gpio_multi_config_pad\(\)](#).

Here is the call graph for this function:



15.382.2.2 get()

```
int L4vbus::Gpio_module::get (
    unsigned offset,
    unsigned * data) const [inline]
```

Read values of multiple GPIO pins at once.

Parameters

| | | |
|-----|---------------|---|
| | <i>offset</i> | Pin corresponding to the LSB in <i>data</i> . Note: allowed may be hardware specific. |
| out | <i>data</i> | Each bit returns the value (0 or 1) for the corresponding GPIO pin. The value of pins that are not accessible is undefined. |

Returns

0 if OK, error code otherwise

Definition at line 201 of file [vbus_gpio](#).

References [L4vbus::Device::_bus](#), [L4vbus::Device::_dev](#), and [l4vbus_gpio_multi_get\(\)](#).

Here is the call graph for this function:



15.382.2.3 pin()

```
Gpio_pin L4vbus::Gpio_module::pin (
    unsigned pin) const [inline]
```

Get [Gpio_pin](#) for a specific pin of this [Gpio_module](#).

Parameters

| | |
|------------|--|
| <i>pin</i> | GPIO pin number to get Gpio_pin for. |
|------------|--|

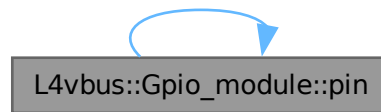
Returns[Gpio_pin](#)

Definition at line [229](#) of file [vbus_gpio](#).

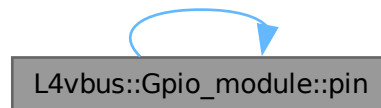
References [pin\(\)](#).

Referenced by [pin\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**15.382.2.4 set()**

```
int L4vbus::Gpio_module::set (
    Pin\_slice const & mask,
    unsigned data) [inline]
```

Set multiple GPIO output pins at once.

Parameters

| | |
|-------------|--|
| <i>mask</i> | Mask of GPIO pins to set. A bit set to 1 selects this pin. A maximum of 32 pins can be set at once. The real number depends on the hardware and the driver implementation. |
| <i>data</i> | Each bit corresponds to the GPIO pin in <i>mask</i> . The value of each bit is written to the GPIO pin if its bit in <i>mask</i> is set. |

Returns

0 if OK, error code otherwise

Definition at line 217 of file [vbus_gpio](#).

References [L4vbus::Device::_bus](#), [L4vbus::Device::_dev](#), and [l4vbus_gpio_multi_set\(\)](#).

Here is the call graph for this function:

**15.382.2.5 setup()**

```
int L4vbus::Gpio_module::setup (
    Pin_slice const & mask,
    unsigned mode,
    unsigned value) const [inline]
```

Configure function of multiple GPIO pins at once.

Parameters

| | |
|--------------|--|
| <i>mask</i> | Mask of GPIO pins to configure. A bit set to 1 configures this pin. A maximum of 32 pins can be configured at once. The real number depends on the hardware and the driver implementation. |
| <i>mode</i> | GPIO function, see L4vbus_gpio_generic_func for generic functions. Hardware specific functions must be provided in the lower 8 bits. |
| <i>value</i> | Optional value to set the GPIO pins to if they are configured as output pins |

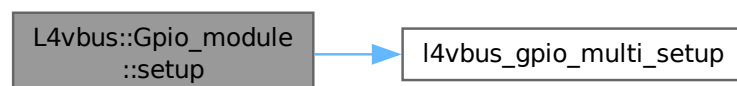
Returns

0 if OK, error code otherwise

Definition at line 166 of file [vbus_gpio](#).

References [L4vbus::Device::_bus](#), [L4vbus::Device::_dev](#), and [l4vbus_gpio_multi_setup\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

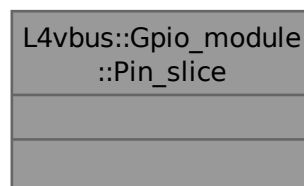
- l4/vbus/vbus_gpio

15.383 L4vbus::Gpio_module::Pin_slice Struct Reference

A slice of the pins provided by this module.

```
#include <vbus_gpio>
```

Collaboration diagram for L4vbus::Gpio_module::Pin_slice:



15.383.1 Detailed Description

A slice of the pins provided by this module.

Data type to specify a selection of pins for the 'multi' methods.

Definition at line [146](#) of file [vbus_gpio](#).

The documentation for this struct was generated from the following file:

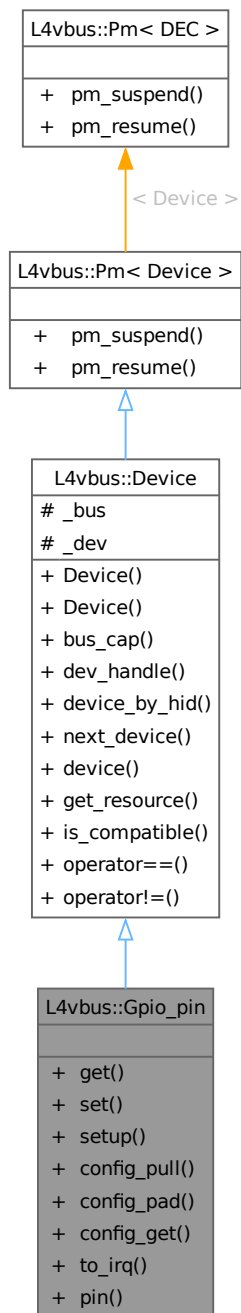
- l4/vbus/vbus_gpio

15.384 L4vbus::Gpio_pin Class Reference

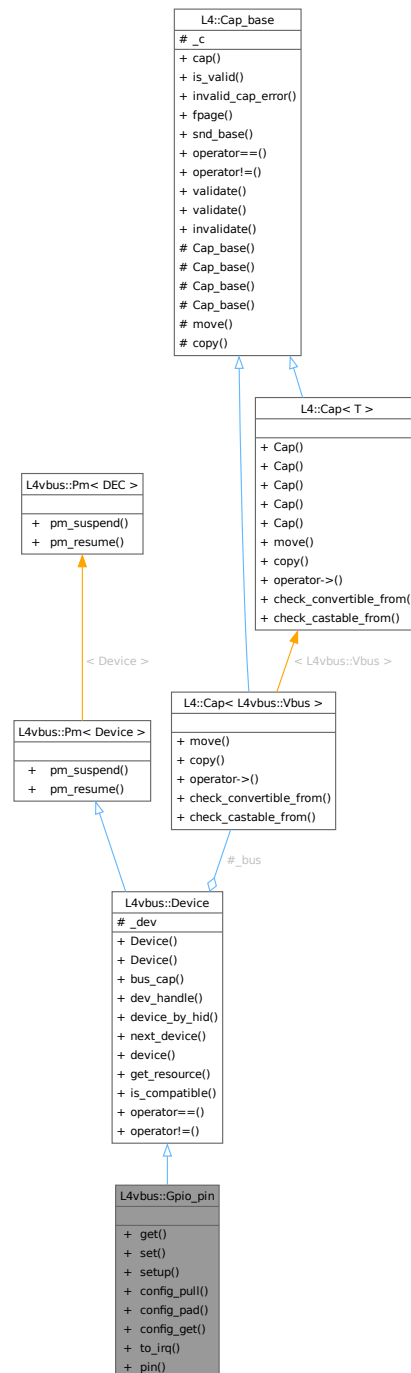
A GPIO pin.

```
#include <vbus_gpio>
```

Inheritance diagram for L4vbus::Gpio_pin:



Collaboration diagram for L4vbus::Gpio_pin:



Public Member Functions

- `int get () const`
Read value of GPIO input pin.
- `int set (int value) const`
Set GPIO output pin.
- `int setup (unsigned mode, unsigned value) const`

- *Configure the function of a GPIO pin.*
- int [config_pull](#) (unsigned mode) const
Generic function to set pull up/down mode.
- int [config_pad](#) (unsigned func, unsigned value) const
Hardware specific configuration function.
- int [config_get](#) (unsigned func, unsigned *value) const
Read hardware specific configuration.
- int [to_irq](#) () const
Create IRQ for GPIO pin.
- unsigned [pin](#) () const
Get pin number.

Public Member Functions inherited from [L4vbus::Device](#)

- [Device](#) ()
Construct a new vbus device using the NULL device [L4VBUS_NULL](#).
- [Device](#) ([L4::Cap](#)< [Vbus](#) > bus, [l4vbus_device_handle_t](#) dev)
Construct a new vbus device using a device handle.
- [L4::Cap](#)< [Vbus](#) > [bus_cap](#) () const
Access the [Vbus](#) capability of the underlying virtual bus.
- [l4vbus_device_handle_t](#) [dev_handle](#) () const
Access the device handle of this device.
- int [device_by_hid](#) ([Device](#) *child, char const *hid, int depth=L4VBUS_MAX_DEPTH, [l4vbus_device_t](#) *devinfo=0) const
Find a device by the hardware interface identifier (HID).
- int [next_device](#) ([Device](#) *child, int depth=L4VBUS_MAX_DEPTH, [l4vbus_device_t](#) *devinfo=0) const
*Find next child following *child*.*
- int [device](#) ([l4vbus_device_t](#) *devinfo) const
Obtain detailed information about a [Vbus](#) device.
- int [get_resource](#) (unsigned res_idx, [l4vbus_resource_t](#) *res) const
Obtain the resource description of an individual device resource.
- int [is_compatible](#) (char const *cid) const
*Check if the given device has a compatibility ID (CID) or HID that matches *cid*.*
- bool [operator==](#) ([Device](#) const &o) const
Test if two devices are the same [Vbus](#) device.
- bool [operator!=](#) ([Device](#) const &o) const
Test if two [Vbus](#) devices are not the same.

Public Member Functions inherited from [L4vbus::Pm](#)< [Device](#) >

- int [pm_suspend](#) () const
Suspend the device.
- int [pm_resume](#) () const
Resume the device.

Additional Inherited Members

Protected Attributes inherited from [L4vbus::Device](#)

- [L4::Cap< Vbus > _bus](#)
The [Vbus](#) capability where this device is located on.
- [l4vbus_device_handle_t _dev](#)
The device handle for this device.

15.384.1 Detailed Description

A GPIO pin.

Definition at line 26 of file [vbus_gpio](#).

15.384.2 Member Function Documentation

15.384.2.1 `config_get()`

```
int L4vbus::Gpio_pin::config_get (
    unsigned func,
    unsigned * value) const [inline]
```

Read hardware specific configuration.

Parameters

| | | |
|-----|--------------|---|
| | <i>func</i> | Hardware specific configuration register to read from. Usually this is an offset to the GPIO chip's base address. |
| out | <i>value</i> | The configuration value. |

Returns

0 if OK, error code otherwise

Definition at line 102 of file [vbus_gpio](#).

References [L4vbus::Device::_bus](#), [L4vbus::Device::_dev](#), and [l4vbus_gpio_config_get\(\)](#).

Here is the call graph for this function:



15.384.2.2 config_pad()

```
int L4vbus::Gpio_pin::config_pad (
    unsigned func,
    unsigned value) const [inline]
```

Hardware specific configuration function.

Parameters

| | |
|--------------|--|
| <i>func</i> | Hardware specific configuration register, usually offset to the GPIO chip's base address |
| <i>value</i> | Value which is written into the hardware specific configuration register for the specified pin |

Returns

0 if OK, error code otherwise

Definition at line 89 of file [vbus_gpio](#).

References [L4vbus::Device::_bus](#), [L4vbus::Device::_dev](#), and [l4vbus_gpio_config_pad\(\)](#).

Here is the call graph for this function:



15.384.2.3 config_pull()

```
int L4vbus::Gpio_pin::config_pull (
    unsigned mode) const [inline]
```

Generic function to set pull up/down mode.

Parameters

| | |
|-------------|---|
| <i>mode</i> | mode for pull up/down resistors, see L4vbus_gpio_pull_modes |
|-------------|---|

Returns

0 if OK, error code otherwise

Definition at line 75 of file [vbus_gpio](#).

References [L4vbus::Device::_bus](#), [L4vbus::Device::_dev](#), and [l4vbus_gpio_config_pull\(\)](#).

Here is the call graph for this function:

**15.384.2.4 get()**

```
int L4vbus::Gpio_pin::get () const [inline]
```

Read value of GPIO input pin.

Returns

Value of GPIO pin (usually 0 or 1), negative error code otherwise.

Definition at line 38 of file [vbus_gpio](#).

References [L4vbus::Device::_bus](#), [L4vbus::Device::_dev](#), and [l4vbus_gpio_get\(\)](#).

Here is the call graph for this function:

**15.384.2.5 pin()**

```
unsigned L4vbus::Gpio_pin::pin () const [inline]
```

Get pin number.

Returns

GPIO pin number

Definition at line 122 of file [vbus_gpio](#).

15.384.2.6 set()

```
int L4vbus::Gpio_pin::set (  
    int value) const [inline]
```

Set GPIO output pin.

Parameters

| | |
|--------------|---|
| <i>value</i> | Value to write to the GPIO pin (usually 0 or 1) |
|--------------|---|

Returns

0 if OK, error code otherwise

Definition at line 49 of file [vbus_gpio](#).

References [L4vbus::Device::_bus](#), [L4vbus::Device::_dev](#), and [l4vbus_gpio_set\(\)](#).

Here is the call graph for this function:



15.384.2.7 setup()

```
int L4vbus::Gpio_pin::setup (  
    unsigned mode,  
    unsigned value) const [inline]
```

Configure the function of a GPIO pin.

Parameters

| | |
|--------------|--|
| <i>mode</i> | GPIO function, see L4vbus_gpio_generic_func for generic functions. Hardware specific functions must be provided in the lower 8 bits. |
| <i>value</i> | Optional value to set the GPIO pin to if it is configured as an output pin |

Returns

0 if OK, error code otherwise

Definition at line 64 of file [vbus_gpio](#).

References [L4vbus::Device::_bus](#), [L4vbus::Device::_dev](#), and [l4vbus_gpio_setup\(\)](#).

Here is the call graph for this function:

**15.384.2.8 to_irq()**

```
int L4vbus::Gpio_pin::to_irq () const [inline]
```

Create IRQ for GPIO pin.

Returns

IRQ number if OK, negative error code otherwise

Definition at line 112 of file [vbus_gpio](#).

References [L4vbus::Device::_bus](#), [L4vbus::Device::_dev](#), and [l4vbus_gpio_to_irq\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

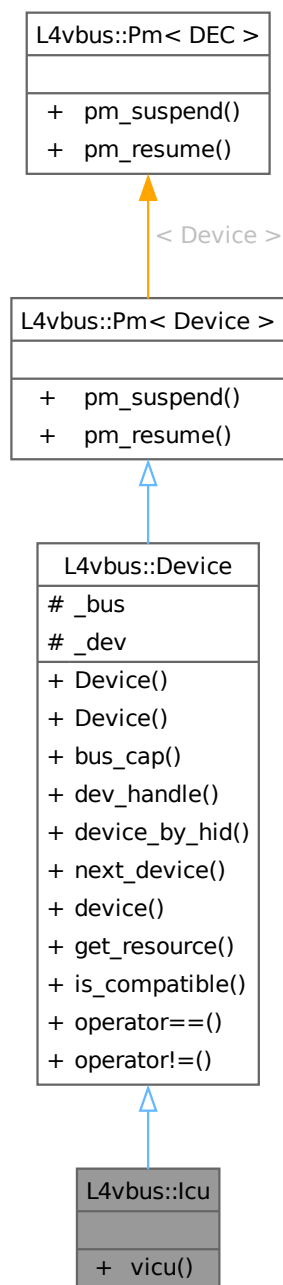
- `I4/vbus/vbus_gpio`

15.385 L4vbus::lcu Class Reference

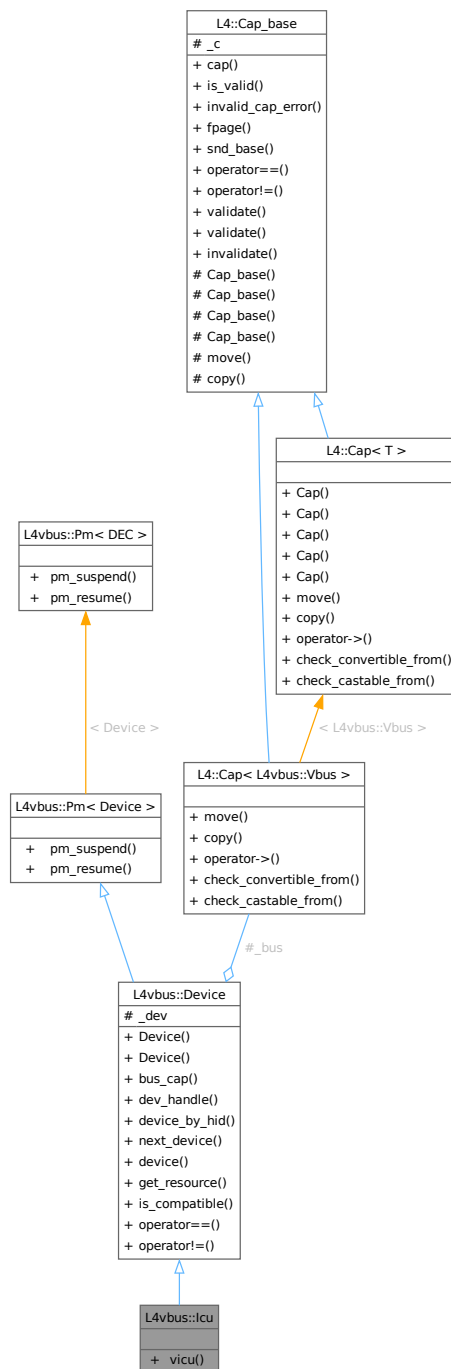
Vbus Interrupt controller API.

```
#include <vbus>
```

Inheritance diagram for L4vbus::lcu:



Collaboration diagram for L4vbus::Icu:



Public Types

- enum **Src_types** { **Src_dev_handle** = L4VBUS_ICU_SRC_DEV_HANDLE }
Flags that can be used with the ICU on a vbus device.

Public Member Functions

- int **vicu** (L4::Cap< L4::Icu > icu) const

Request an [L4::lcu](#) capability for this [Vbus](#)'s virtual ICU.

Public Member Functions inherited from [L4vbus::Device](#)

- [Device](#) ()
Construct a new vbus device using the NULL device [L4VBUS_NULL](#).
- [Device](#) ([L4::Cap](#)< [Vbus](#) > bus, [l4vbus_device_handle_t](#) dev)
Construct a new vbus device using a device handle.
- [L4::Cap](#)< [Vbus](#) > [bus_cap](#) () const
Access the [Vbus](#) capability of the underlying virtual bus.
- [l4vbus_device_handle_t](#) [dev_handle](#) () const
Access the device handle of this device.
- int [device_by_hid](#) ([Device](#) *child, char const *hid, int depth=L4VBUS_MAX_DEPTH, [l4vbus_device_t](#) *devinfo=0) const
Find a device by the hardware interface identifier (HID).
- int [next_device](#) ([Device](#) *child, int depth=L4VBUS_MAX_DEPTH, [l4vbus_device_t](#) *devinfo=0) const
Find next child following *child*.
- int [device](#) ([l4vbus_device_t](#) *devinfo) const
Obtain detailed information about a [Vbus](#) device.
- int [get_resource](#) (unsigned res_idx, [l4vbus_resource_t](#) *res) const
Obtain the resource description of an individual device resource.
- int [is_compatible](#) (char const *cid) const
Check if the given device has a compatibility ID (CID) or HID that matches *cid*.
- bool [operator==](#) ([Device](#) const &o) const
Test if two devices are the same [Vbus](#) device.
- bool [operator!=](#) ([Device](#) const &o) const
Test if two [Vbus](#) devices are not the same.

Public Member Functions inherited from [L4vbus::Pm](#)< [Device](#) >

- int [pm_suspend](#) () const
Suspend the device.
- int [pm_resume](#) () const
Resume the device.

Additional Inherited Members

Protected Attributes inherited from [L4vbus::Device](#)

- [L4::Cap](#)< [Vbus](#) > [_bus](#)
The [Vbus](#) capability where this device is located on.
- [l4vbus_device_handle_t](#) [_dev](#)
The device handle for this device.

15.385.1 Detailed Description

[Vbus](#) Interrupt controller API.

Every [Vbus](#) contains a virtual interrupt control unit that manages the IRQs of the devices on the bus. This class provides access to a capability to an [L4::lcu](#) object which can then be used to interface with the IRQs.

See also

[L4::lcu](#)

Definition at line 260 of file [vbus](#).

15.385.2 Member Enumeration Documentation

15.385.2.1 Src_types

```
enum L4vbus::Icu::Src_types
```

Flags that can be used with the ICU on a vbus device.

Enumerator

| | |
|----------------|---|
| Src_dev_handle | Flag to denote that the value should be interpreted as a device handle. This flag may be used in the <code>source</code> parameter in L4::lcu::msi_info() to denote that the ICU should interpret the source ID as a device handle. |
|----------------|---|

Definition at line 264 of file [vbus](#).

15.385.3 Member Function Documentation

15.385.3.1 vicu()

```
int L4vbus::Icu::vicu (
    L4::Cap< L4::Icu > icu) const [inline]
```

Request an [L4::lcu](#) capability for this [Vbus](#)'s virtual ICU.

Parameters

| | | |
|-----|-----|---|
| out | icu | Capability slot where the L4::lcu capability shall be stored. |
|-----|-----|---|

Return values

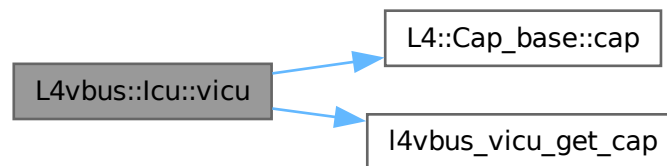
| | |
|---|----------|
| 0 | Success. |
|---|----------|

| | |
|------------------|------------|
| <i>otherwise</i> | IPC error. |
|------------------|------------|

Definition at line 285 of file [vbus](#).

References [L4vbus::Device::_bus](#), [L4vbus::Device::_dev](#), [L4::Cap_base::cap\(\)](#), and [l4vbus_vicu_get_cap\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

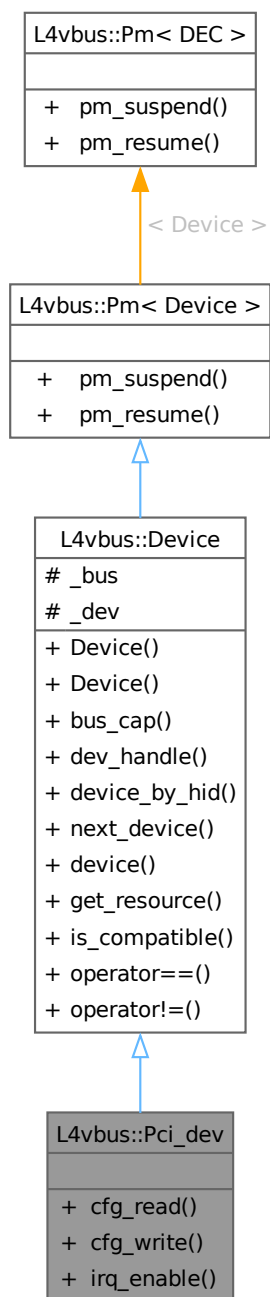
- `I4/vbus/vbus`

15.386 L4vbus::Pci_dev Class Reference

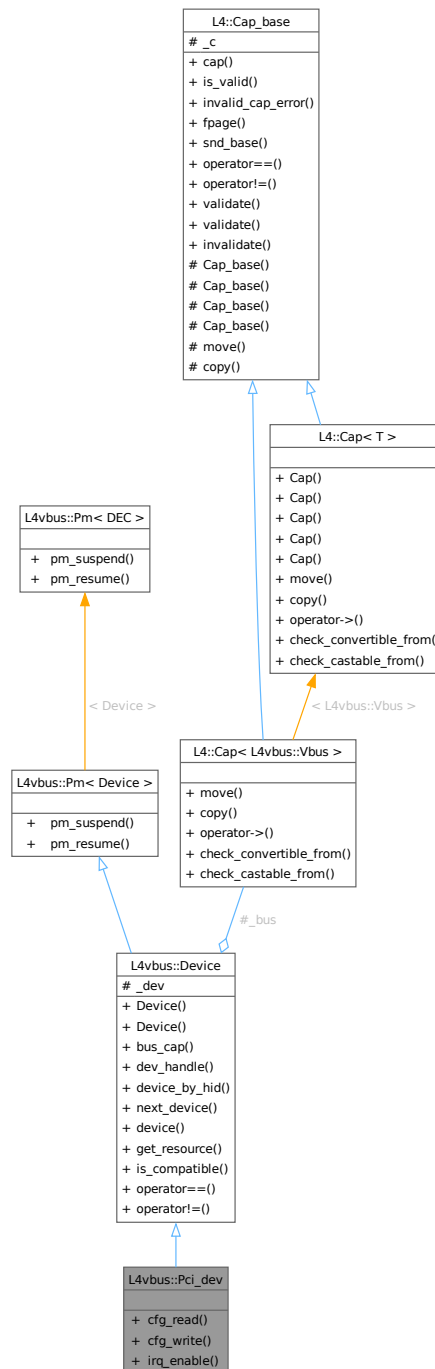
A PCI device.

```
#include <vbus_pci>
```

Inheritance diagram for L4vbus::Pci_dev:



Collaboration diagram for L4vbus::Pci_dev:



Public Member Functions

- `int cfg_read (l4_uint32_t reg, l4_uint32_t *value, l4_uint32_t width) const`
Read from the device's vPCI configuration space.
- `int cfg_write (l4_uint32_t reg, l4_uint32_t value, l4_uint32_t width) const`
Write to the device's vPCI configuration space.
- `int irq_enable (unsigned char *trigger, unsigned char *polarity) const`
Enable the device's PCI interrupt.

Public Member Functions inherited from [L4vbus::Device](#)

- **Device** ()
Construct a new vbus device using the NULL device [L4VBUS_NULL](#).
- **Device** ([L4::Cap](#)< [Vbus](#) > bus, [l4vbus_device_handle_t](#) dev)
Construct a new vbus device using a device handle.
- **L4::Cap**< [Vbus](#) > **bus_cap** () const
Access the [Vbus](#) capability of the underlying virtual bus.
- **l4vbus_device_handle_t** **dev_handle** () const
Access the device handle of this device.
- int **device_by_hid** ([Device](#) *child, char const *hid, int depth=L4VBUS_MAX_DEPTH, [l4vbus_device_t](#) *devinfo=0) const
Find a device by the hardware interface identifier (HID).
- int **next_device** ([Device](#) *child, int depth=L4VBUS_MAX_DEPTH, [l4vbus_device_t](#) *devinfo=0) const
*Find next child following *child*.*
- int **device** ([l4vbus_device_t](#) *devinfo) const
Obtain detailed information about a [Vbus](#) device.
- int **get_resource** (unsigned res_idx, [l4vbus_resource_t](#) *res) const
Obtain the resource description of an individual device resource.
- int **is_compatible** (char const *cid) const
*Check if the given device has a compatibility ID (CID) or HID that matches *cid*.*
- bool **operator==** ([Device](#) const &o) const
Test if two devices are the same [Vbus](#) device.
- bool **operator!=** ([Device](#) const &o) const
Test if two [Vbus](#) devices are not the same.

Public Member Functions inherited from [L4vbus::Pm](#)< [Device](#) >

- int **pm_suspend** () const
Suspend the device.
- int **pm_resume** () const
Resume the device.

Additional Inherited Members

Protected Attributes inherited from [L4vbus::Device](#)

- **L4::Cap**< [Vbus](#) > **_bus**
The [Vbus](#) capability where this device is located on.
- **l4vbus_device_handle_t** **_dev**
The device handle for this device.

15.386.1 Detailed Description

A PCI device.

Definition at line 93 of file [vbus_pci](#).

15.386.2 Member Function Documentation

15.386.2.1 cfg_read()

```
int L4vbus::Pci_dev::cfg_read (
    14_uint32_t reg,
    14_uint32_t * value,
    14_uint32_t width) const [inline]
```

Read from the device's vPCI configuration space.

Parameters

| | | |
|-----|--------------|---|
| | <i>reg</i> | Register in configuration space to read |
| out | <i>value</i> | Value that has been read |
| | <i>width</i> | Width to read in bits (e.g. 8, 16, 32) |

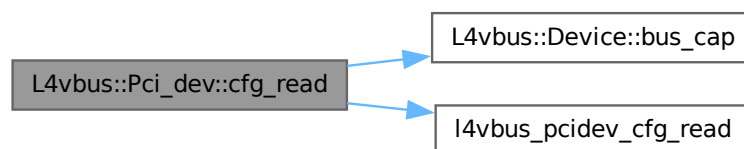
Returns

0 on success, else failure

Definition at line 105 of file [vbus_pci](#).

References [L4vbus::Device::_dev](#), [L4vbus::Device::bus_cap\(\)](#), and [l4vbus_pcidev_cfg_read\(\)](#).

Here is the call graph for this function:



15.386.2.2 cfg_write()

```
int L4vbus::Pci_dev::cfg_write (
    14_uint32_t reg,
    14_uint32_t value,
    14_uint32_t width) const [inline]
```

Write to the device's vPCI configuration space.

Parameters

| | |
|--------------|--|
| <i>reg</i> | Register in configuration space to write |
| <i>value</i> | Value to write |
| <i>width</i> | Width to write in bits (e.g. 8, 16, 32) |

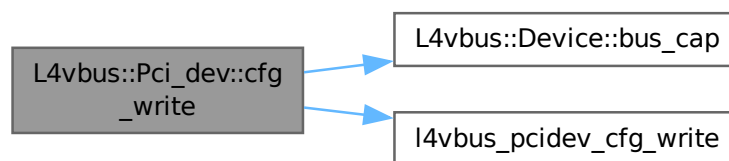
Returns

0 on success, else failure

Definition at line 121 of file [vbus_pci](#).

References [L4vbus::Device::_dev](#), [L4vbus::Device::bus_cap\(\)](#), and [l4vbus_pcidev_cfg_write\(\)](#).

Here is the call graph for this function:



15.386.2.3 `irq_enable()`

```
int L4vbus::Pci_dev::irq_enable (
    unsigned char * trigger,
    unsigned char * polarity) const [inline]
```

Enable the device's PCI interrupt.

Parameters

| | | |
|-----|-----------------|---------------------------------------|
| out | <i>trigger</i> | False if interrupt is level-triggered |
| out | <i>polarity</i> | True if interrupt is of low polarity |

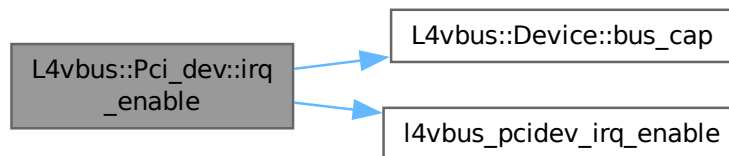
Returns

On success: Interrupt line to be used, else failure

Definition at line 137 of file [vbus_pci](#).

References [L4vbus::Device::_dev](#), [L4vbus::Device::bus_cap\(\)](#), and [l4vbus_pcidev_irq_enable\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

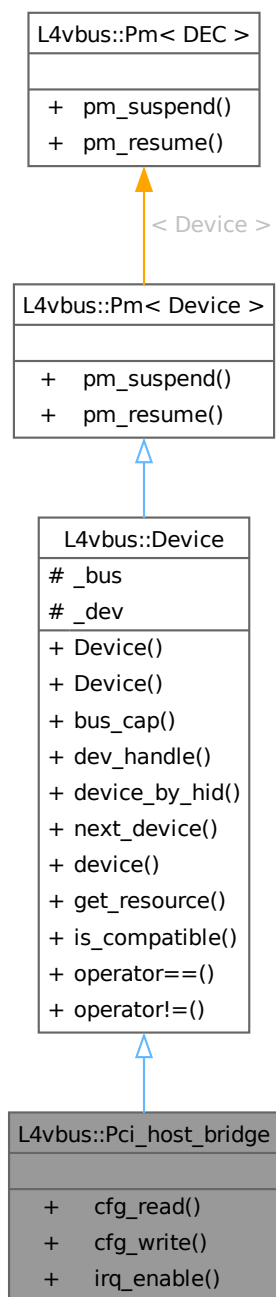
- `l4/vbus/vbus_pci`

15.387 L4vbus::Pci_host_bridge Class Reference

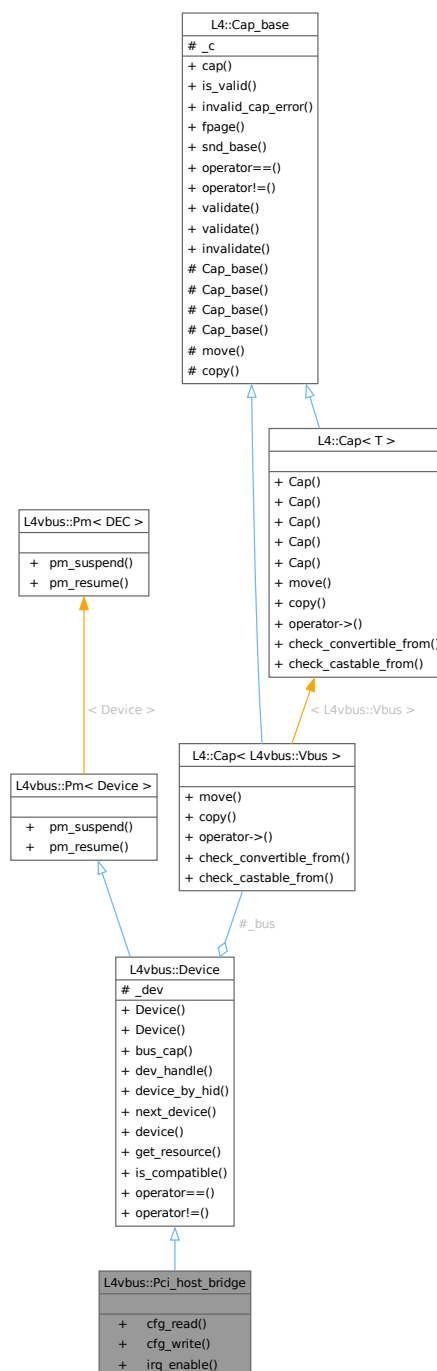
A Pci host bridge.

```
#include <vbus_pci>
```

Inheritance diagram for L4vbus::Pci_host_bridge:



Collaboration diagram for L4vbus::Pci_host_bridge:



Public Member Functions

- `int cfg_read (l4_uint32_t bus, l4_uint32_t devfn, l4_uint32_t reg, l4_uint32_t *value, l4_uint32_t width) const`
Read from the vPCI configuration space using the PCI root bridge.
- `int cfg_write (l4_uint32_t bus, l4_uint32_t devfn, l4_uint32_t reg, l4_uint32_t value, l4_uint32_t width) const`
Write to the vPCI configuration space using the PCI root bridge.

- int `irq_enable` (`l4_uint32_t` bus, `l4_uint32_t` devfn, int pin, unsigned char *trigger, unsigned char *polarity) const

Enable PCI interrupt for a specific device using the PCI root bridge.

Public Member Functions inherited from `L4vbus::Device`

- `Device` ()
Construct a new vbus device using the NULL device `L4VBUS_NULL`.
- `Device` (`L4::Cap`< `Vbus` > bus, `l4vbus_device_handle_t` dev)
Construct a new vbus device using a device handle.
- `L4::Cap`< `Vbus` > `bus_cap` () const
Access the `Vbus` capability of the underlying virtual bus.
- `l4vbus_device_handle_t` `dev_handle` () const
Access the device handle of this device.
- int `device_by_hid` (`Device` *child, char const *hid, int depth=`L4VBUS_MAX_DEPTH`, `l4vbus_device_t` *devinfo=0) const
Find a device by the hardware interface identifier (HID).
- int `next_device` (`Device` *child, int depth=`L4VBUS_MAX_DEPTH`, `l4vbus_device_t` *devinfo=0) const
Find next child following `child`.
- int `device` (`l4vbus_device_t` *devinfo) const
Obtain detailed information about a `Vbus` device.
- int `get_resource` (unsigned res_idx, `l4vbus_resource_t` *res) const
Obtain the resource description of an individual device resource.
- int `is_compatible` (char const *cid) const
Check if the given device has a compatibility ID (CID) or HID that matches cid.
- bool `operator==` (`Device` const &o) const
Test if two devices are the same `Vbus` device.
- bool `operator!=` (`Device` const &o) const
Test if two `Vbus` devices are not the same.

Public Member Functions inherited from `L4vbus::Pm`< `Device` >

- int `pm_suspend` () const
Suspend the device.
- int `pm_resume` () const
Resume the device.

Additional Inherited Members

Protected Attributes inherited from `L4vbus::Device`

- `L4::Cap`< `Vbus` > `_bus`
The `Vbus` capability where this device is located on.
- `l4vbus_device_handle_t` `_dev`
The device handle for this device.

15.387.1 Detailed Description

A Pci host bridge.

Definition at line 25 of file [vbus_pci](#).

15.387.2 Member Function Documentation

15.387.2.1 `cfg_read()`

```
int L4vbus::Pci_host_bridge::cfg_read (
    14_uint32_t bus,
    14_uint32_t devfn,
    14_uint32_t reg,
    14_uint32_t * value,
    14_uint32_t width) const [inline]
```

Read from the vPCI configuration space using the PCI root bridge.

Parameters

| | | |
|-----|--------------|--|
| | <i>bus</i> | Bus number |
| | <i>devfn</i> | Device id (upper 16bit) and function (lower 16bit) |
| | <i>reg</i> | Register in configuration space to read |
| out | <i>value</i> | Value that has been read |
| | <i>width</i> | Width to read in bits (e.g. 8, 16, 32) |

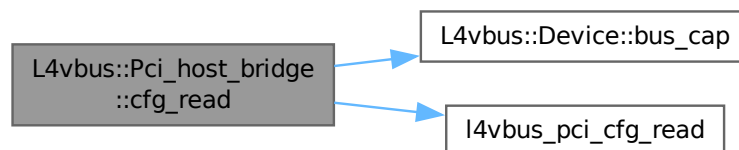
Returns

0 on success, else failure

Definition at line 39 of file [vbus_pci](#).

References [L4vbus::Device::_dev](#), [L4vbus::Device::bus_cap\(\)](#), and [l4vbus_pci_cfg_read\(\)](#).

Here is the call graph for this function:



15.387.2.2 `cfg_write()`

```
int L4vbus::Pci_host_bridge::cfg_write (
    14_uint32_t bus,
    14_uint32_t devfn,
    14_uint32_t reg,
    14_uint32_t value,
    14_uint32_t width) const [inline]
```

Write to the vPCI configuration space using the PCI root bridge.

Parameters

| | |
|--------------|--|
| <i>bus</i> | Bus number |
| <i>devfn</i> | Device id (upper 16bit) and function (lower 16bit) |
| <i>reg</i> | Register in configuration space to write |
| <i>value</i> | Value to write |
| <i>width</i> | Width to write in bits (e.g. 8, 16, 32) |

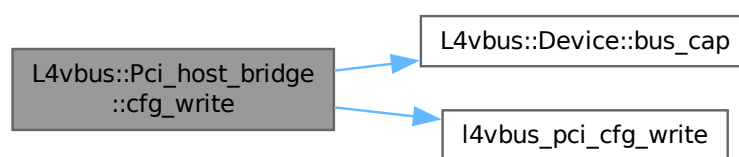
Returns

0 on success, else failure

Definition at line 58 of file [vbus_pci](#).

References [L4vbus::Device::_dev](#), [L4vbus::Device::bus_cap\(\)](#), and [l4vbus_pci_cfg_write\(\)](#).

Here is the call graph for this function:



15.387.2.3 `irq_enable()`

```
int L4vbus::Pci_host_bridge::irq_enable (
    14_uint32_t bus,
    14_uint32_t devfn,
    int pin,
    unsigned char * trigger,
    unsigned char * polarity) const [inline]
```

Enable PCI interrupt for a specific device using the PCI root bridge.

Parameters

| | | |
|-----|-----------------|---|
| | <i>bus</i> | Bus number |
| | <i>devfn</i> | Device id (upper 16bit) and function (lower 16bit) |
| | <i>pin</i> | Interrupt pin (normally as reported in configuration register INTR) |
| out | <i>trigger</i> | False if interrupt is level-triggered |
| out | <i>polarity</i> | True if interrupt is of low polarity |

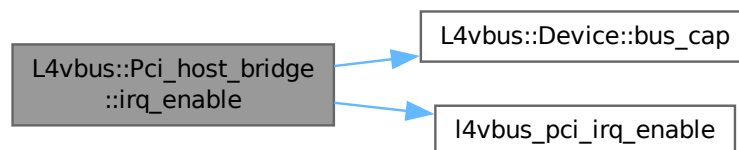
Returns

On success: Interrupt line to be used, else failure

Definition at line 79 of file [vbus_pci](#).

References [L4vbus::Device::_dev](#), [L4vbus::Device::bus_cap\(\)](#), and [l4vbus_pci_irq_enable\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

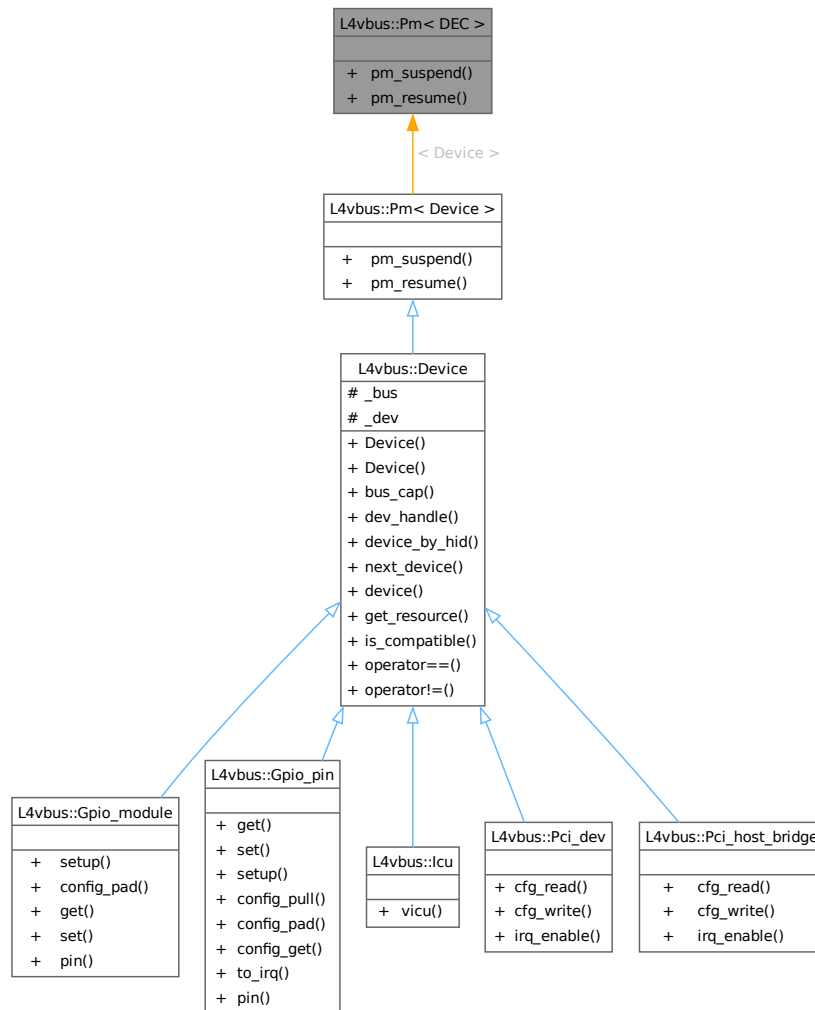
- `l4/vbus/vbus_pci`

15.388 L4vbus::Pm< DEC > Class Template Reference

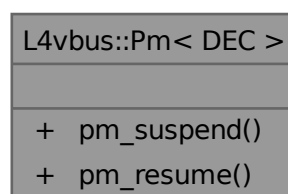
Power-management API mixin.

```
#include <vbus>
```

Inheritance diagram for L4vbus::Pm< DEC >:



Collaboration diagram for L4vbus::Pm< DEC >:



Public Member Functions

- `int pm_suspend () const`
Suspend the device.
- `int pm_resume () const`
Resume the device.

15.388.1 Detailed Description

```
template<typename DEC>
class L4vbus::Pm< DEC >
```

Power-management API mixin.

Devices that inherit from this mixin provide an API to be suspended and resumed.

Definition at line 50 of file [vbus](#).

15.388.2 Member Function Documentation

15.388.2.1 pm_resume()

```
template<typename DEC>
int L4vbus::Pm< DEC >::pm_resume () const [inline]
```

Resume the device.

Switches the device from low-power mode to normal operation and restores the saved state.

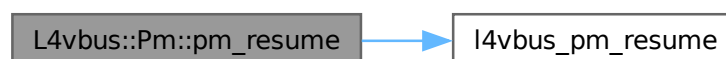
Return values

| | |
|---|----------|
| 0 | Success. |
|---|----------|

Definition at line 74 of file [vbus](#).

References [l4vbus_pm_resume\(\)](#).

Here is the call graph for this function:



15.388.2.2 pm_suspend()

```
template<typename DEC>
int L4vbus::Pm< DEC >::pm_suspend () const [inline]
```

Suspend the device.

Saves the state of the device and puts it into a low-power mode.

Return values

| | |
|---|----------|
| 0 | Success. |
|---|----------|

Definition at line 63 of file [vbus](#).

References [l4vbus_pm_suspend\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

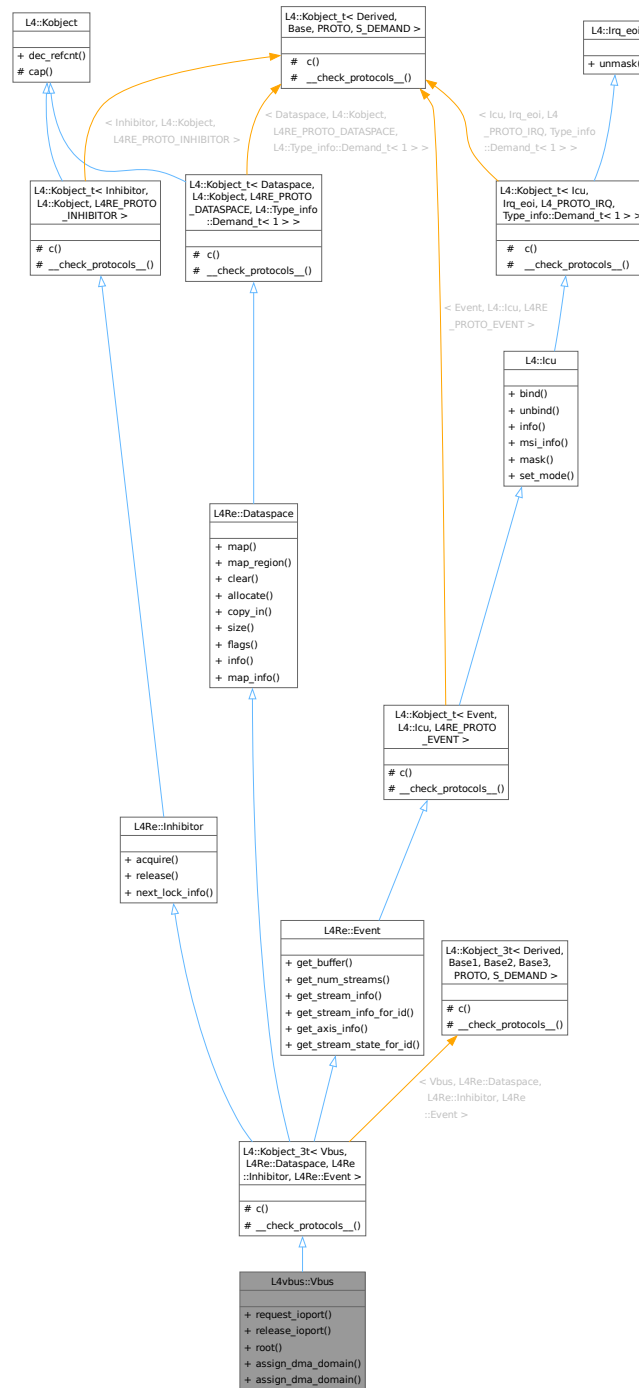
- l4/vbus/vbus

15.389 L4vbus::Vbus Class Reference

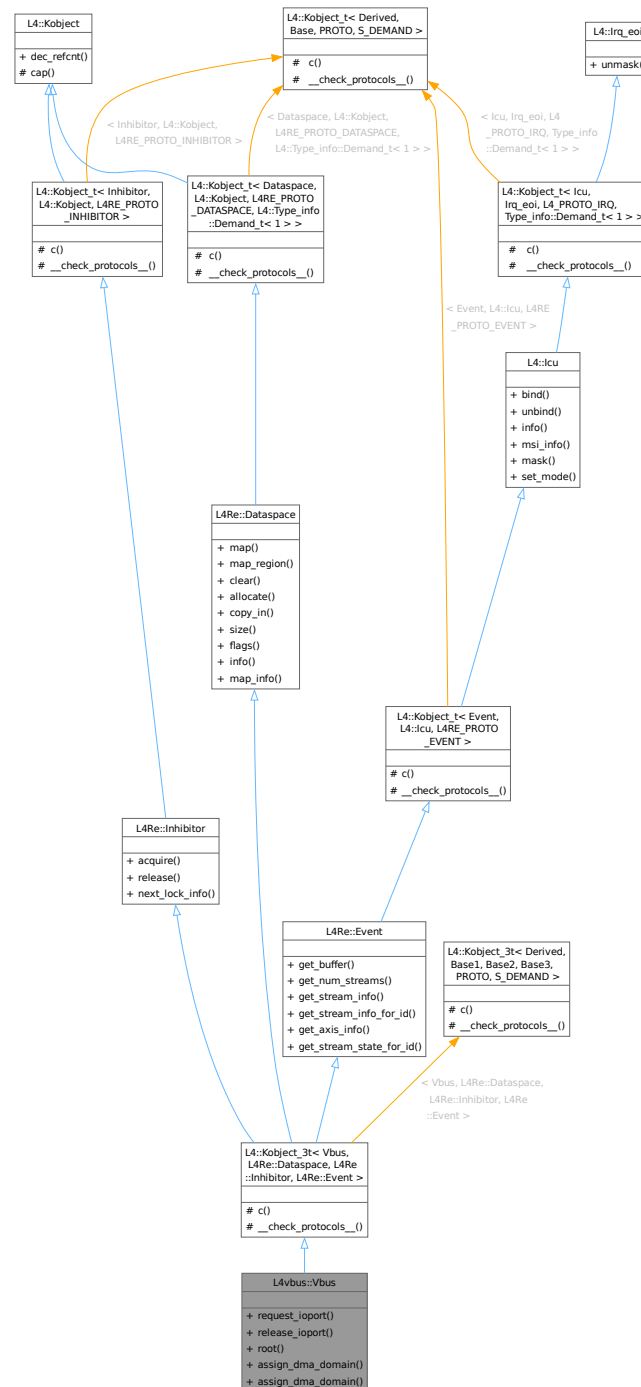
The virtual bus ([Vbus](#)) interface.

```
#include <vbus>
```

Inheritance diagram for L4vbus::Vbus:



Collaboration diagram for L4vbus::Vbus:



Public Member Functions

- `int request_ioport (l4vbus_resource_t *res) const`
Request the given IO port resource from the bus.
- `int release_ioport (l4vbus_resource_t *res) const`
Release the given IO port resource from the bus.
- `Device root () const`

Get the root device of the device tree of this bus.

- int [assign_dma_domain](#) (unsigned domain_id, unsigned flags, [L4::Cap](#)< [L4Re::Dma_space](#) > dma_space) const

Bind or unbind an [L4Re::Dma_space](#) to a DMA domain.

- int [assign_dma_domain](#) (unsigned domain_id, unsigned flags, [L4::Cap](#)< [L4::Task](#) > dma_space) const

Bind or unbind a kernel [DMA space](#) to a DMA domain.

Public Member Functions inherited from [L4Re::Dataspace](#)

- [l4_ret_t map](#) (Offset offset, Flags flags, Map_addr local_addr, Map_addr min_addr, Map_addr max_addr, [L4::Cap](#)< [L4::Task](#) > dst=[L4::Cap](#)< [L4::Task](#) >::Invalid) const noexcept

Request a flexpage mapping from the dataspace.

- [l4_ret_t map_region](#) (Offset offset, Flags flags, Map_addr min_addr, Map_addr max_addr, [L4::Cap](#)< [L4::Task](#) > dst=[L4::Cap](#)< [L4::Task](#) >::Invalid) const noexcept

Map a part of a dataspace into a local memory area.

- [l4_ret_t clear](#) (Offset offset, Size size)

Clear parts of a dataspace.

- [l4_ret_t allocate](#) (Offset offset, Size size)

Allocate a range in the dataspace.

- [l4_ret_t copy_in](#) (Offset dst_offs, [L4::lpc::Cap](#)< Dataspace > src, Offset src_offs, Size size)

Copy contents from another dataspace.

- Size [size](#) () const noexcept

Get size of a dataspace.

- Flags [flags](#) () const noexcept

Get flags of the dataspace.

- [l4_ret_t info](#) (Stats *stats)

Get information on the dataspace.

- long [map_info](#) ([l4_addr_t](#) *start_addr, [l4_addr_t](#) *end_addr)

Get mapping range of dataspace.

Public Member Functions inherited from [L4::Kobject](#)

- [l4_msgtag_t dec_refcnt](#) ([l4_mword_t](#) diff, [l4_utcb_t](#) *utcb=[l4_utcb](#)())

Decrement the in kernel reference counter for the object.

Public Member Functions inherited from [L4Re::Inhibitor](#)

- [l4_ret_t acquire](#) ([l4_umword_t](#) id, [L4::lpc::String](#)<> reason)

Acquire a specific inhibitor lock.

- [l4_ret_t release](#) ([l4_umword_t](#) id)

Release a specific inhibitor lock.

- [l4_ret_t next_lock_info](#) (char *name, unsigned len, [l4_mword_t](#) current_id=-1, [l4_utcb_t](#) *utcb=[l4_utcb](#)())

Get information for the next available inhibitor lock.

Public Member Functions inherited from [L4Re::Event](#)

- [l4_ret_t get_buffer](#) ([L4::lpc::Out](#)< [L4::Cap](#)< Dataspace > > ds)
Get event signal buffer.
- [l4_ret_t get_num_streams](#) ()
Get number of event streams.
- [l4_ret_t get_stream_info](#) (int idx, [Event_stream_info](#) *info)
Get event stream infos.
- [l4_ret_t get_stream_info_for_id](#) ([l4_umword_t](#) stream_id, [Event_stream_info](#) *info)
Get event stream infos.
- [l4_ret_t get_axis_info](#) ([l4_umword_t](#) stream_id, unsigned naxes, unsigned const *axis, [Event_absinfo](#) *info) const noexcept
Get event stream axis infos.
- [l4_ret_t get_stream_state_for_id](#) ([l4_umword_t](#) stream_id, [Event_stream_state](#) *state)
Get event stream state.

Public Member Functions inherited from [L4::Icu](#)

- [l4_msgtag_t bind](#) (unsigned irqnum, [L4::Cap](#)< [Triggerable](#) > irq, [l4_utcb_t](#) *utcb=[l4_utcb](#)()) noexcept
Bind an interrupt line of an interrupt controller to an interrupt object.
- [l4_msgtag_t unbind](#) (unsigned irqnum, [L4::Cap](#)< [Triggerable](#) > irq, [l4_utcb_t](#) *utcb=[l4_utcb](#)()) noexcept
Remove binding of an interrupt line from the interrupt controller object.
- [l4_msgtag_t info](#) ([l4_icu_info_t](#) *info, [l4_utcb_t](#) *utcb=[l4_utcb](#)()) noexcept
Get information about the ICU features.
- [l4_msgtag_t msi_info](#) ([l4_umword_t](#) irqnum, [l4_uint64_t](#) source, [l4_icu_msi_info_t](#) *msi_info)
Get MSI info about IRQ.
- [l4_msgtag_t mask](#) (unsigned irqnum, [l4_umword_t](#) *label=0, [l4_timeout_t](#) to=[L4_IPC_NEVER](#), [l4_utcb_t](#) *utcb=[l4_utcb](#)()) noexcept
Mask an IRQ line.
- [l4_msgtag_t set_mode](#) (unsigned irqnum, [l4_umword_t](#) mode, [l4_utcb_t](#) *utcb=[l4_utcb](#)()) noexcept
Set interrupt mode.

Public Member Functions inherited from [L4::Irq_eoi](#)

- [l4_msgtag_t unmask](#) (unsigned irqnum, [l4_umword_t](#) *label=0, [l4_timeout_t](#) to=[L4_IPC_NEVER](#), [l4_utcb_t](#) *utcb=[l4_utcb](#)()) noexcept
Unmask the given interrupt line.

Additional Inherited Members

Public Types inherited from [L4Re::Inhibitor](#)

- enum { [Name_max](#) = 20 }

Protected Types inherited from**L4::Kobject_3t< Vbus, L4Re::Dataspace, L4Re::Inhibitor, L4Re::Event >**

- typedef Vbus **Class**
The target interface type (inheriting from [Kobject_t](#)).
- typedef Typeid::Iface< [PROTO_ANY](#), Vbus > **__Iface**
The interface description for the derived class.
- typedef Typeid::Merge_list< Typeid::Iface_list< [__Iface](#) >, Typeid::Merge_list< typename L4Re::↵
 Dataspace::__Iface_list, Typeid::Merge_list< typename L4Re::Inhibitor::__Iface_list, typename L4Re::↵
 Event::__Iface_list > > > **__Iface_list**
The list of all RPC interfaces provided directly or through inheritance.

Protected Types inherited from**L4::Kobject_t< Dataspace, L4::Kobject, L4RE_PROTO_DATASPACE, L4::Type_info::Demand_t< 1 > >**

- typedef Dataspace **Class**
The target interface type (inheriting from [Kobject_t](#)).
- typedef Typeid::Iface< [PROTO](#), Dataspace > **__Iface**
The interface description for the derived class.
- typedef Typeid::Merge_list< Typeid::Iface_list< [__Iface](#) >, typename L4::Kobject::__Iface_list > **__Iface_list**
The list of all RPC interfaces provided directly or through inheritance.

Protected Types inherited from**L4::Kobject_t< Inhibitor, L4::Kobject, L4RE_PROTO_INHIBITOR >**

- typedef Inhibitor **Class**
The target interface type (inheriting from [Kobject_t](#)).
- typedef Typeid::Iface< [PROTO](#), Inhibitor > **__Iface**
The interface description for the derived class.
- typedef Typeid::Merge_list< Typeid::Iface_list< [__Iface](#) >, typename L4::Kobject::__Iface_list > **__Iface_list**
The list of all RPC interfaces provided directly or through inheritance.

Protected Types inherited from [L4::Kobject_t< Event, L4::lcu, L4RE_PROTO_EVENT >](#)

- typedef Event **Class**
The target interface type (inheriting from [Kobject_t](#)).
- typedef Typeid::Iface< [PROTO](#), Event > **__Iface**
The interface description for the derived class.
- typedef Typeid::Merge_list< Typeid::Iface_list< [__Iface](#) >, typename L4::lcu::__Iface_list > **__Iface_list**
The list of all RPC interfaces provided directly or through inheritance.

Protected Types inherited from**L4::Kobject_t< lcu, Irq_eoi, L4_PROTO_IRQ, Type_info::Demand_t< 1 > >**

- typedef lcu **Class**
The target interface type (inheriting from [Kobject_t](#)).
- typedef Typeid::Iface< [PROTO](#), lcu > **__Iface**
The interface description for the derived class.
- typedef Typeid::Merge_list< Typeid::Iface_list< [__Iface](#) >, typename Irq_eoi::__Iface_list > **__Iface_list**
The list of all RPC interfaces provided directly or through inheritance.

Protected Member Functions inherited from**[L4::Kobject_3t< Vbus, L4Re::Dataspace, L4Re::Inhibitor, L4Re::Event >](#)**

- [L4::Cap< Class > c \(\)](#) const noexcept

*Get the capability to ourselves.***Protected Member Functions inherited from****[L4::Kobject_t< Dataspace, L4::Kobject, L4RE_PROTO_DATASPACE, L4::Type_info::Demand_t< 1 > >](#)**

- [L4::Cap< Class > c \(\)](#) const noexcept

*Get the capability to ourselves.***Protected Member Functions inherited from [L4::Kobject](#)**

- [l4_cap_idx_t cap \(\)](#) const noexcept

*Return capability selector.***Protected Member Functions inherited from****[L4::Kobject_t< Inhibitor, L4::Kobject, L4RE_PROTO_INHIBITOR >](#)**

- [L4::Cap< Class > c \(\)](#) const noexcept

*Get the capability to ourselves.***Protected Member Functions inherited from****[L4::Kobject_t< Event, L4::lcu, L4RE_PROTO_EVENT >](#)**

- [L4::Cap< Class > c \(\)](#) const noexcept

*Get the capability to ourselves.***Protected Member Functions inherited from****[L4::Kobject_t< lcu, Irq_eoi, L4_PROTO_IRQ, Type_info::Demand_t< 1 > >](#)**

- [L4::Cap< Class > c \(\)](#) const noexcept

*Get the capability to ourselves.***Static Protected Member Functions inherited from****[L4::Kobject_3t< Vbus, L4Re::Dataspace, L4Re::Inhibitor, L4Re::Event >](#)**

- static void [__check_protocols__ \(\)](#) noexcept

*Helper to check for protocol conflicts.***Static Protected Member Functions inherited from****[L4::Kobject_t< Dataspace, L4::Kobject, L4RE_PROTO_DATASPACE, L4::Type_info::Demand_t< 1 > >](#)**

- static void [__check_protocols__ \(\)](#) noexcept

Helper to check for protocol conflicts.

Static Protected Member Functions inherited from**[L4::Kobject_t< Inhibitor, L4::Kobject, L4RE_PROTO_INHIBITOR >](#)**

- static void **`__check_protocols__`** () noexcept
Helper to check for protocol conflicts.

Static Protected Member Functions inherited from**[L4::Kobject_t< Event, L4::lcu, L4RE_PROTO_EVENT >](#)**

- static void **`__check_protocols__`** () noexcept
Helper to check for protocol conflicts.

Static Protected Member Functions inherited from**[L4::Kobject_t< lcu, Irq_eoi, L4_PROTO_IRQ, Type_info::Demand_t< 1 > >](#)**

- static void **`__check_protocols__`** () noexcept
Helper to check for protocol conflicts.

15.389.1 Detailed Description

The virtual bus ([Vbus](#)) interface.

See also

[L4Re Vbus API](#)

Definition at line [298](#) of file [vbus](#).

15.389.2 Member Function Documentation**15.389.2.1 `assign_dma_domain()` [1/2]**

```
int L4vbus::Vbus::assign_dma_domain (
    unsigned domain_id,
    unsigned flags,
    L4::Cap< L4::Task > dma_space) const [inline]
```

Bind or unbind a kernel [DMA space](#) to a DMA domain.

Parameters

| | |
|------------------|---|
| <i>domain_id</i> | DMA domain ID (resource address of DMA domain found on the vBUS). If the value is ~0U the DMA space of the whole vBUS is used. |
| <i>flags</i> | A combination of L4vbus_dma_domain_assign_flags . |
| <i>dma_space</i> | The DMA space capability to bind or unbind, this must be a kernel DMA space (L4::Task created with L4_PROTO_DMA_SPACE) |

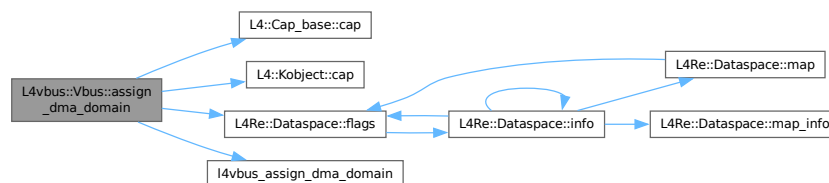
Return values

| | |
|-------------------|---|
| <i>0</i> | Operation completed successfully. |
| <i>-L4_ENOENT</i> | The vbus does not support a global DMA domain or no DMA domain could be found. |
| <i>-L4_EINVAL</i> | Invalid argument used. |
| <i>-L4_EBUSY</i> | DMA domain is already active, this means another DMA space is already assigned. |

Definition at line 382 of file [vbus](#).

References [L4::Cap_base::cap\(\)](#), [L4::Kobject::cap\(\)](#), [L4Re::Dataspace::flags\(\)](#), [l4vbus_assign_dma_domain\(\)](#), [L4VBUS_DMAD_KERNEL_DMA_SPACE](#), and [L4VBUS_DMAD_L4RE_DMA_SPACE](#).

Here is the call graph for this function:

**15.389.2.2 assign_dma_domain() [2/2]**

```

int L4vbus::Vbus::assign_dma_domain (
    unsigned domain_id,
    unsigned flags,
    L4::Cap< L4Re::Dma_space > dma_space) const [inline]

```

Bind or unbind an [L4Re::Dma_space](#) to a DMA domain.

Parameters

| | |
|------------------|--|
| <i>domain_id</i> | DMA domain ID (resource address of DMA domain found on the vBUS). If the value is ~0U the DMA space of the whole vBUS is used. |
| <i>flags</i> | A combination of L4vbus_dma_domain_assign_flags . |
| <i>dma_space</i> | The DMA space capability to bind or unbind, this must be an L4Re::Dma_space |

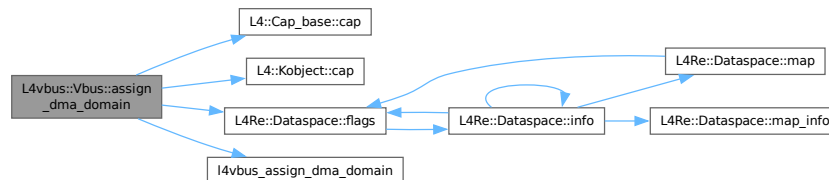
Return values

| | |
|-------------------|---|
| <i>0</i> | Operation completed successfully. |
| <i>-L4_ENOENT</i> | The vbus does not support a global DMA domain or no DMA domain could be found. |
| <i>-L4_EINVAL</i> | Invalid argument used. |
| <i>-L4_EBUSY</i> | DMA domain is already active, this means another DMA space is already assigned. |

Definition at line 357 of file [vbus](#).

References [L4::Cap_base::cap\(\)](#), [L4::Kobject::cap\(\)](#), [L4Re::Dataspace::flags\(\)](#), [l4vbus_assign_dma_domain\(\)](#), [L4VBUS_DMAD_KERNEL_DMA_SPACE](#), and [L4VBUS_DMAD_L4RE_DMA_SPACE](#).

Here is the call graph for this function:



15.389.2.3 release_ioport()

```
int L4vbus::Vbus::release_ioport (
    l4vbus_resource_t * res) const [inline]
```

Release the given IO port resource from the bus.

Parameters

| | | |
|----|-----|---|
| in | res | The IO port resource to be released from the bus. |
|----|-----|---|

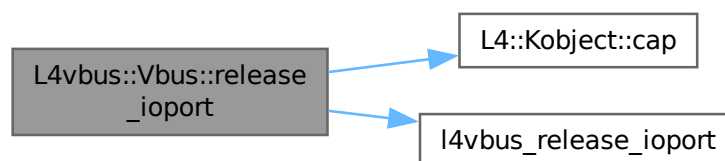
Returns

>=0 on success, <0 on error.

Definition at line 323 of file [vbus](#).

References [L4::Kobject::cap\(\)](#), and [l4vbus_release_ioport\(\)](#).

Here is the call graph for this function:



15.389.2.4 request_ioport()

```
int L4vbus::Vbus::request_ioport (
    l4vbus_resource_t * res) const [inline]
```

Request the given IO port resource from the bus.

Parameters

| | | |
|----|------------|--|
| in | <i>res</i> | The IO port resource to be requested from the bus. |
|----|------------|--|

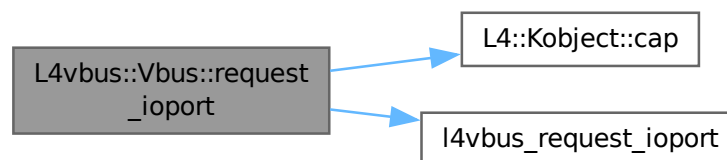
Return values

| | |
|------------|--------------------------------------|
| 0 | Success. |
| -L4_EINVAL | Resource is not an IO port resource. |
| -L4_ENOENT | No matching IO port resource found. |

Definition at line 311 of file [vbus](#).

References [L4::Kobject::cap\(\)](#), and [l4vbus_request_ioport\(\)](#).

Here is the call graph for this function:



15.389.2.5 root()

```
Device L4vbus::Vbus::root () const [inline]
```

Get the root device of the device tree of this bus.

The root device is usually the starting point for iterating the bus, see [Device::next_device](#).

Returns

A [Vbus](#) device representing the root of the device tree.

Definition at line 336 of file [vbus](#).

References [L4::Kobject::cap\(\)](#), and [L4VBUS_ROOT_BUS](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

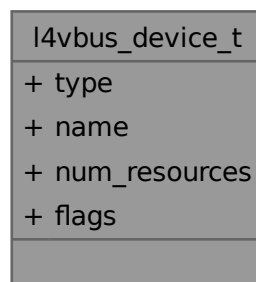
- `l4/vbus/vbus`

15.390 l4vbus_device_t Struct Reference

Detailed information about a vbus device.

```
#include <vbus_types.h>
```

Collaboration diagram for `l4vbus_device_t`:



Data Fields

- [l4_uint32_t](#) **type**
Bitfield of supported sub-interfaces, see [l4vbus_iface_type_t](#).
- char **name** [L4VBUS_DEV_NAME_LEN]
Name.
- unsigned **num_resources**
Number of resources for this device.
- unsigned **flags**
Flags, see [l4vbus_device_flags_t](#).

15.390.1 Detailed Description

Detailed information about a vbus device.

Definition at line 80 of file [vbus_types.h](#).

The documentation for this struct was generated from the following file:

- [l4/vbus/vbus_types.h](#)

15.391 l4vbus_resource_t Struct Reference

Description of a single vbus resource.

```
#include <vbus_types.h>
```

Collaboration diagram for `l4vbus_resource_t`:

| l4vbus_resource_t |
|-------------------|
| + type |
| + flags |
| + start |
| + end |
| + provider |
| + id |
| |

Data Fields

- [l4_uint16_t type](#)
Resource type, see [l4vbus_resource_type_t](#).
- [l4_uint16_t flags](#)
Flags.
- [l4vbus_paddr_t start](#)
Start of resource range.
- [l4vbus_paddr_t end](#)
End of resource range (inclusive).
- [l4vbus_device_handle_t provider](#)
Device handle of the provider of the resource.
- [l4_uint32_t id](#)
Resource ID (4 bytes), usually a 4 letter ASCII name is used.

15.391.1 Detailed Description

Description of a single vbus resource.

Definition at line 23 of file [vbus_types.h](#).

The documentation for this struct was generated from the following file:

- [l4/vbus/vbus_types.h](#)

15.392 L4vcpu::State Class Reference

C++ implementation of state word in the vCPU area.

```
#include <vcpu>
```

Collaboration diagram for L4vcpu::State:



Public Member Functions

- [State](#) (unsigned v)
Initialize state.
- void [add](#) (unsigned bits) throw ()
Add flags.
- void [clear](#) (unsigned bits) throw ()
Clear flags.
- void [set](#) (unsigned v) throw ()
Set flags.

15.392.1 Detailed Description

C++ implementation of state word in the vCPU area.

Definition at line 24 of file [vcpu](#).

15.392.2 Constructor & Destructor Documentation

15.392.2.1 State()

```
L4vcpu::State::State (
    unsigned v) [inline], [explicit]
```

Initialize state.

Parameters

| | |
|----------|----------------|
| <i>v</i> | Initial state. |
|----------|----------------|

Definition at line 34 of file [vcpu](#).

15.392.3 Member Function Documentation

15.392.3.1 add()

```
void L4vcpu::State::add (
    unsigned bits) throw ( ) [inline]
```

Add flags.

Parameters

| | |
|-------------|--------------------------|
| <i>bits</i> | Bits to add to the word. |
|-------------|--------------------------|

Definition at line 41 of file [vcpu](#).

15.392.3.2 clear()

```
void L4vcpu::State::clear (
    unsigned bits) throw ( )    [inline]
```

Clear flags.

Parameters

| | |
|-------------|----------------------------|
| <i>bits</i> | Bits to clear in the word. |
|-------------|----------------------------|

Definition at line 48 of file [vcpu](#).

15.392.3.3 set()

```
void L4vcpu::State::set (
    unsigned v) throw ( )    [inline]
```

Set flags.

Parameters

| | |
|----------|---|
| <i>v</i> | Set the word to the value of <i>v</i> . |
|----------|---|

Definition at line 55 of file [vcpu](#).

The documentation for this class was generated from the following file:

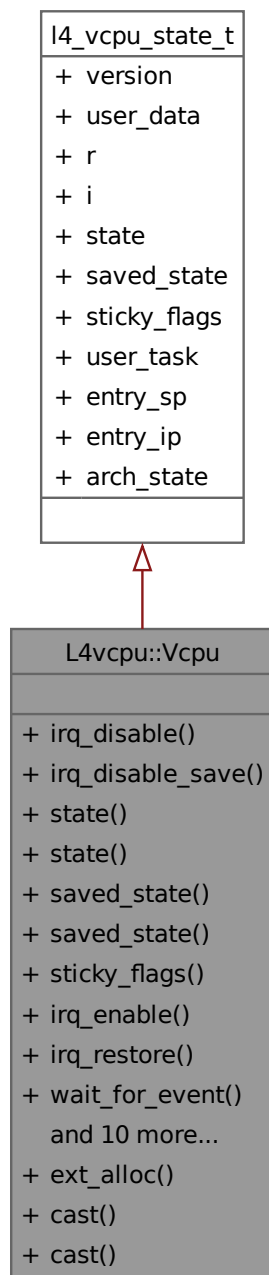
- [l4/vcpu/vcpu](#)

15.393 L4vcpu::Vcpu Class Reference

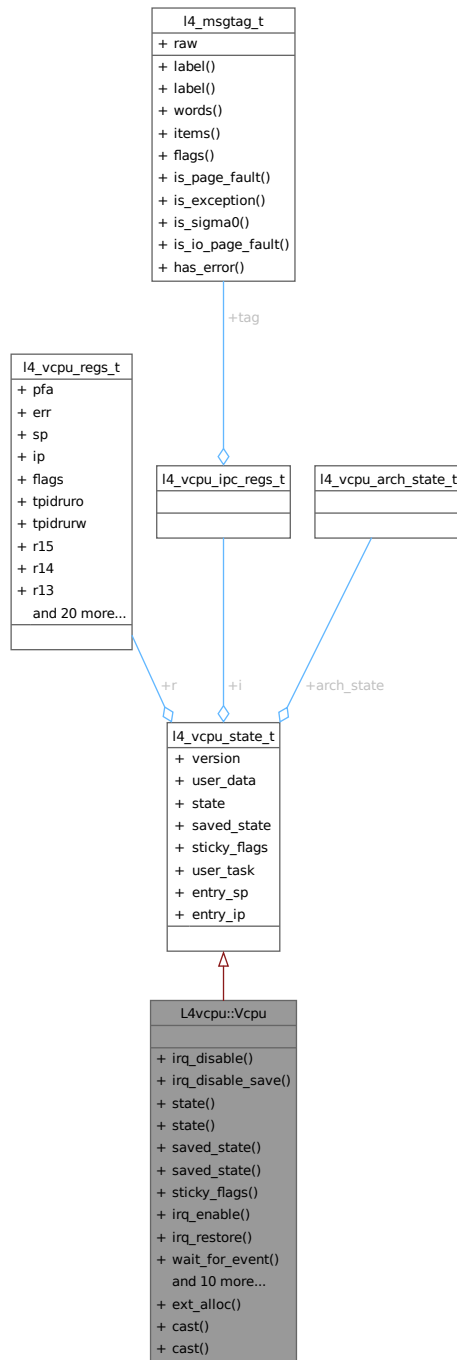
C++ implementation of the vCPU save state area.

```
#include <vcpu>
```

Inheritance diagram for L4vcpu::Vcpu:



Collaboration diagram for L4vcpu::Vcpu:



Public Member Functions

- void **irq_disable** () throw ()
Disable the vCPU for event delivery.
- unsigned **irq_disable_save** () throw ()
Disable the vCPU for event delivery and return previous state.
- **State** * **state** () throw ()

- Get state word.*

 - `State state () const throw ()`
- Get state word.*

 - `State * saved_state () throw ()`
- Get saved_state word.*

 - `State saved_state () const throw ()`
- Get saved_state word.*

 - `l4_uint16_t sticky_flags () const throw ()`
- Get sticky flags.*

 - `void irq_enable (l4_utcb_t *utcb, l4vcpu_event_hndl_t do_event_work_cb, l4vcpu_setup_ipc_t setup_ipc) throw ()`
- Enable the vCPU for event delivery.*

 - `void irq_restore (unsigned s, l4_utcb_t *utcb, l4vcpu_event_hndl_t do_event_work_cb, l4vcpu_setup_ipc_t setup_ipc) throw ()`
- Restore a previously saved IRQ/event state.*

 - `void wait_for_event (l4_utcb_t *utcb, l4vcpu_event_hndl_t do_event_work_cb, l4vcpu_setup_ipc_t setup_ipc) throw ()`
- Wait for event.*

 - `void task (L4::Cap< L4::Task > const task=L4::Cap< L4::Task >::Invalid) throw ()`
- Set the task of the vCPU.*

 - `int is_page_fault_entry () const`
- Return whether the entry reason was a page fault.*

 - `int is_irq_entry () const`
- Return whether the entry reason was an IRQ/IPC message.*

 - `l4_vcpu_regs_t * r () throw ()`
- Return pointer to register state.*

 - `l4_vcpu_regs_t const * r () const throw ()`
- Return pointer to register state.*

 - `l4_vcpu_ipc_regs_t * i () throw ()`
- Return pointer to IPC state.*

 - `l4_vcpu_ipc_regs_t const * i () const throw ()`
- Return pointer to IPC state.*

 - `void entry_sp (l4_umword_t sp)`
- Set vCPU entry stack pointer.*

 - `void entry_ip (l4_umword_t ip)`
- Set vCPU entry instruction pointer.*

 - `void print_state (const char *prefix="") const throw ()`
- Print the state of the vCPU.*

Static Public Member Functions

- `static int ext_alloc (Vcpu **vcpu, l4_addr_t *ext_state, L4::Cap< L4::Task > task=L4Re::Env::env() ->task(), L4::Cap< L4Re::Rm > rm=L4Re::Env::env() ->rm()) throw ()`

Allocate state area for an extended vCPU.
- `static Vcpu * cast (void *x) throw ()`

Cast a void pointer to a class pointer.
- `static Vcpu * cast (l4_addr_t x) throw ()`

Cast an address to a class pointer.

15.393.1 Detailed Description

C++ implementation of the vCPU save state area.

Definition at line 65 of file [vcpu](#).

15.393.2 Member Function Documentation

15.393.2.1 `cast()` [1/2]

```
Vcpu * L4vcpu::Vcpu::cast (
    l4\_addr\_t x) throw ( )    [inline], [static]
```

Cast an address to a class pointer.

Parameters

| | |
|----------|----------|
| <i>x</i> | Pointer. |
|----------|----------|

Returns

Pointer to [Vcpu](#) class.

Definition at line 269 of file [vcpu](#).

15.393.2.2 `cast()` [2/2]

```
Vcpu * L4vcpu::Vcpu::cast (
    void * x) throw ( )    [inline], [static]
```

Cast a void pointer to a class pointer.

Parameters

| | |
|----------|----------|
| <i>x</i> | Pointer. |
|----------|----------|

Returns

Pointer to [Vcpu](#) class.

Definition at line [259](#) of file [vcpu](#).

References [cast\(\)](#).

Referenced by [cast\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**15.393.2.3 entry_ip()**

```
void L4vcpu::Vcpu::entry_ip (  
    14_umword_t ip) [inline]
```

Set vCPU entry instruction pointer.

Parameters

| | |
|-----------|-------------------------------------|
| <i>ip</i> | Instruction pointer address to set. |
|-----------|-------------------------------------|

Definition at line [232](#) of file [vcpu](#).

References [l4_vcpu_state_t::entry_ip](#).

15.393.2.4 entry_sp()

```
void L4vcpu::Vcpu::entry_sp (
    l4_umword_t sp) [inline]
```

Set vCPU entry stack pointer.

Parameters

| | |
|-----------|-------------------------------|
| <i>sp</i> | Stack pointer address to set. |
|-----------|-------------------------------|

Note

The value is only used when entering from a user-task.

Definition at line 225 of file `vcpu`.

References `l4_vcpu_state_t::entry_sp`.

15.393.2.5 ext_alloc()

```
int L4vcpu::Vcpu::ext_alloc (
    Vcpu ** vcpu,
    l4_addr_t * ext_state,
    L4::Cap< L4::Task > task = L4Re::Env::env() ->task(),
    L4::Cap< L4Re::Rm > rm = L4Re::Env::env() ->rm() throw ( ) [static]
```

Allocate state area for an extended vCPU.

Parameters

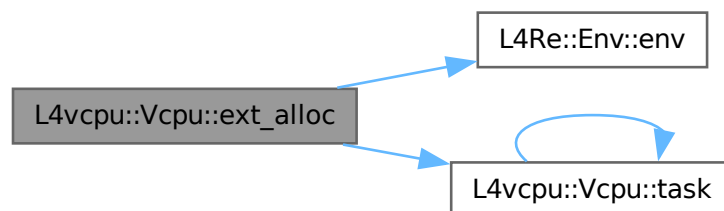
| | | |
|-----|------------------|---|
| out | <i>vcpu</i> | Allocated vcpu-state area. |
| out | <i>ext_state</i> | Allocated extended vcpu-state area. |
| | <i>task</i> | Task to use for allocation, defaults to own task. |
| | <i>rm</i> | Region manager to use for allocation defaults to standard region manager. |

Returns

0 for success, error code otherwise

References `L4Re::Env::env()`, and `task()`.

Here is the call graph for this function:



15.393.2.6 i() [1/2]

```
l4_vcpu_ipc_regs_t * L4vcpu::Vcpu::i () throw ( ) [inline]
```

Return pointer to IPC state.

Returns

Pointer to IPC state.

Definition at line 209 of file `vcpu`.

References `l4_vcpu_state_t::i`.

15.393.2.7 i() [2/2]

```
l4_vcpu_ipc_regs_t const * L4vcpu::Vcpu::i () const throw ( ) [inline]
```

Return pointer to IPC state.

Returns

Pointer to IPC state.

Definition at line 216 of file `vcpu`.

References `l4_vcpu_state_t::i`.

15.393.2.8 irq_disable_save()

```
unsigned L4vcpu::Vcpu::irq_disable_save () throw ( ) [inline]
```

Disable the vCPU for event delivery and return previous state.

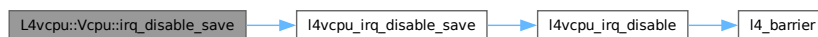
Returns

IRQ state before disabling IRQs.

Definition at line 78 of file `vcpu`.

References `l4vcpu_irq_disable_save()`.

Here is the call graph for this function:



15.393.2.9 irq_enable()

```
void L4vcpu::Vcpu::irq_enable (
    l4_utcb_t * utcb,
    l4vcpu_event_hndl_t do_event_work_cb,
    l4vcpu_setup_ipc_t setup_ipc) throw ( )    [inline]
```

Enable the vCPU for event delivery.

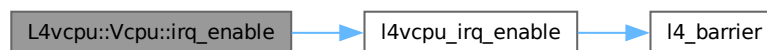
Parameters

| | |
|-------------------------|--|
| <i>utcb</i> | The UTCB to use. |
| <i>do_event_work_cb</i> | Call-back function that is called in case an event (such as an interrupt) is pending. |
| <i>setup_ipc</i> | Call-back function that is called before an IPC operation is called, and before event delivery is enabled. |

Definition at line 135 of file [vcpu](#).

References [l4vcpu_irq_enable\(\)](#).

Here is the call graph for this function:

**15.393.2.10 irq_restore()**

```
void L4vcpu::Vcpu::irq_restore (
    unsigned s,
    l4_utcb_t * utcb,
    l4vcpu_event_hndl_t do_event_work_cb,
    l4vcpu_setup_ipc_t setup_ipc) throw ( )    [inline]
```

Restore a previously saved IRQ/event state.

Parameters

| | |
|-------------------------|--|
| <i>s</i> | IRQ state to be restored. |
| <i>utcb</i> | The UTCB to use. |
| <i>do_event_work_cb</i> | Call-back function that is called in case an event (such as an interrupt) is pending. |
| <i>setup_ipc</i> | Call-back function that is called before an IPC operation is called, and before event delivery is enabled. |

Definition at line 150 of file [vcpu](#).

References [l4vcpu_irq_restore\(\)](#).

Here is the call graph for this function:



15.393.2.11 is_irq_entry()

```
int L4vcpu::Vcpu::is_irq_entry () const [inline]
```

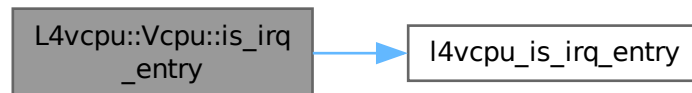
Return whether the entry reason was an IRQ/IPC message.

return 0 if not, !=0 otherwise.

Definition at line 188 of file [vcpu](#).

References [l4vcpu_is_irq_entry\(\)](#).

Here is the call graph for this function:



15.393.2.12 is_page_fault_entry()

```
int L4vcpu::Vcpu::is_page_fault_entry () const [inline]
```

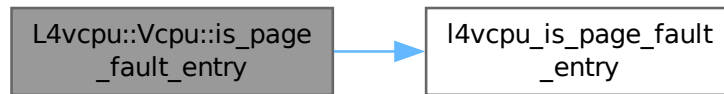
Return whether the entry reason was a page fault.

return 0 if not, !=0 otherwise.

Definition at line 181 of file [vcpu](#).

References [l4vcpu_is_page_fault_entry\(\)](#).

Here is the call graph for this function:



15.393.2.13 r() [1/2]

```
l4_vcpu_regs_t * L4vcpu::Vcpu::r () throw ( ) [inline]
```

Return pointer to register state.

Returns

Pointer to register state.

Definition at line 195 of file `vcpu`.

References `l4_vcpu_state_t::r`.

15.393.2.14 r() [2/2]

```
l4_vcpu_regs_t const * L4vcpu::Vcpu::r () const throw ( ) [inline]
```

Return pointer to register state.

Returns

Pointer to register state.

Definition at line 202 of file `vcpu`.

References `l4_vcpu_state_t::r`.

15.393.2.15 saved_state() [1/2]

```
State * L4vcpu::Vcpu::saved_state () throw ( ) [inline]
```

Get saved_state word.

Returns

Pointer to saved_state word in the vCPU

Definition at line 106 of file `vcpu`.

References `l4_vcpu_state_t::saved_state`.

15.393.2.16 saved_state() [2/2]

```
State L4vcpu::Vcpu::saved_state () const throw ( )    [inline]
```

Get saved_state word.

Returns

Pointer to saved_state word in the vCPU

Definition at line 116 of file [vcpu](#).

References [l4_vcpu_state_t::saved_state](#).

15.393.2.17 state() [1/2]

```
State * L4vcpu::Vcpu::state () throw ( )    [inline]
```

Get state word.

Returns

Pointer to state word in the vCPU

Definition at line 88 of file [vcpu](#).

References [l4_vcpu_state_t::state](#).

15.393.2.18 state() [2/2]

```
State L4vcpu::Vcpu::state () const throw ( )    [inline]
```

Get state word.

Returns

Pointer to state word in the vCPU

Definition at line 99 of file [vcpu](#).

References [l4_vcpu_state_t::state](#).

15.393.2.19 task()

```
void L4vcpu::Vcpu::task (
    L4::Cap< L4::Task > const task = L4::Cap<L4::Task>::Invalid) throw ( )    [inline]
```

Set the task of the vCPU.

Parameters

| | |
|-------------|--|
| <i>task</i> | Task to set, defaults to invalid task. |
|-------------|--|

Definition at line 174 of file [vcpu](#).

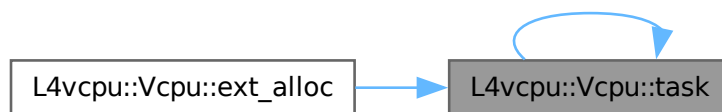
References [L4::Cap_base::Invalid](#), [task\(\)](#), and [l4_vcpu_state_t::user_task](#).

Referenced by [ext_alloc\(\)](#), and [task\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**15.393.2.20 wait_for_event()**

```
void L4vcpu::Vcpu::wait_for_event (
    l4_utcb_t * utcb,
    l4vcpu_event_hndl_t do_event_work_cb,
    l4vcpu_setup_ipc_t setup_ipc) throw ( )    [inline]
```

Wait for event.

Parameters

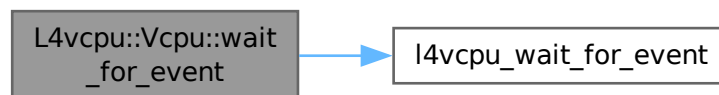
| | |
|-------------------------|---|
| <i>utcb</i> | The UTCB to use. |
| <i>do_event_work_cb</i> | Call-back function that is called in case an event (such as an interrupt) is pending. |
| <i>setup_ipc</i> | Call-back function that is called before an IPC operation is called. |

Note that event delivery remains disabled after this function returns.

Definition at line 166 of file [vcpu](#).

References [l4vcpu_wait_for_event\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

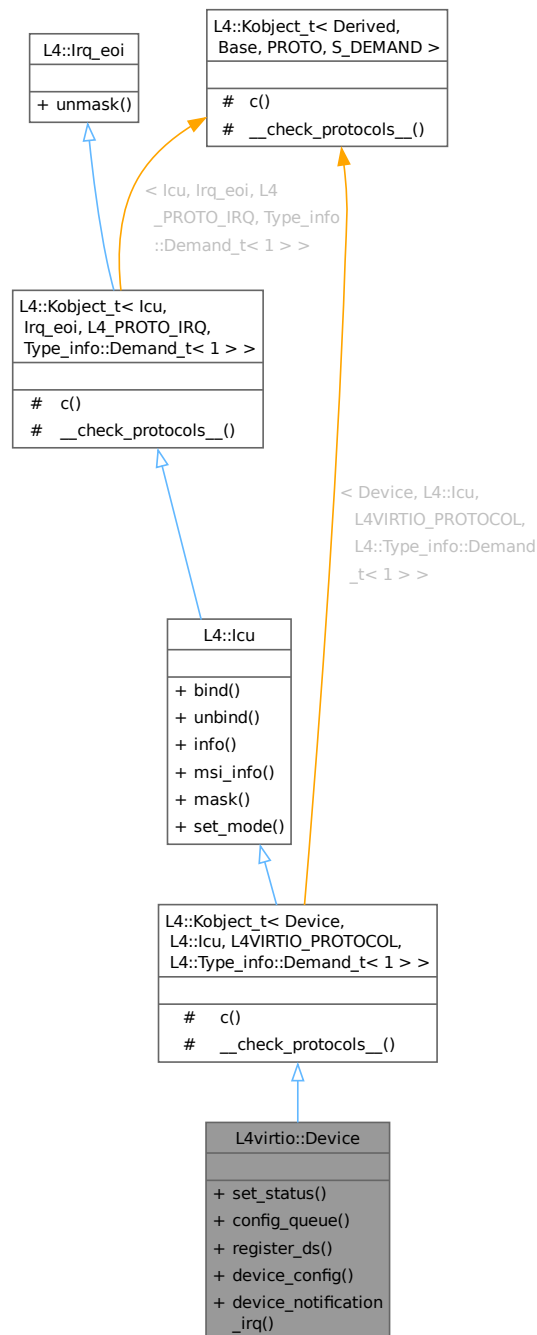
- [l4/vcpu/vcpu](#)

15.394 L4virtio::Device Class Reference

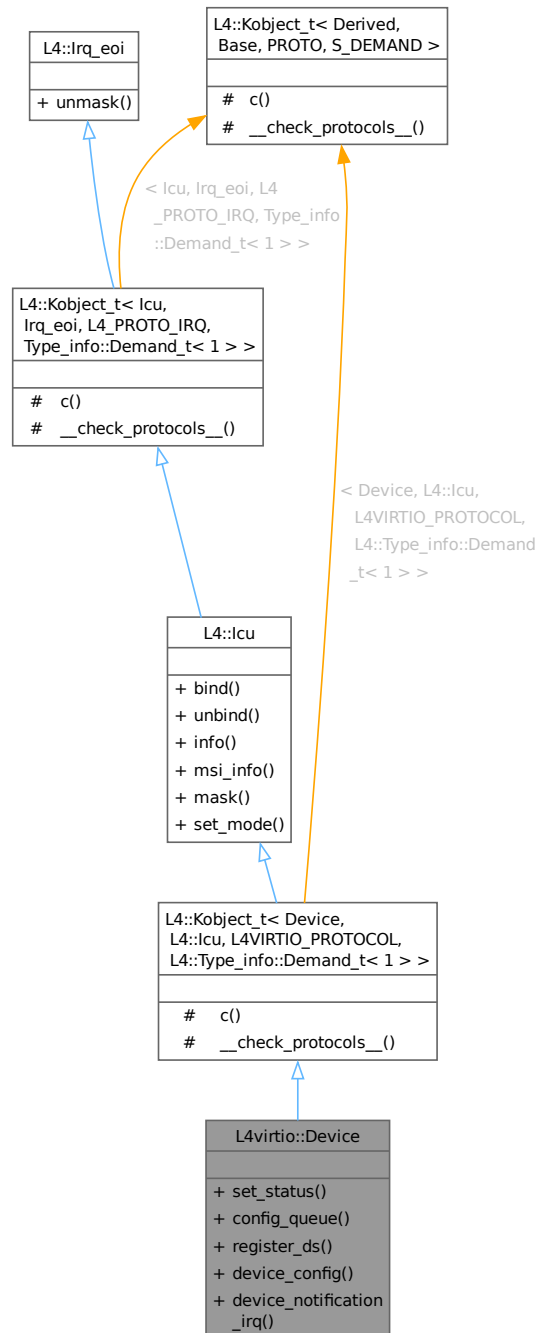
IPC interface for virtio over [L4](#) IPC.

```
#include <l4virtio>
```

Inheritance diagram for L4virtio::Device:



Collaboration diagram for L4virtio::Device:



Public Member Functions

- long [set_status](#) (unsigned status)
Write the VIRTIO status register.
- long [config_queue](#) (unsigned queue)
Trigger queue configuration of the given queue.

- long `register_ds` (`L4::lpc::Cap< L4Re::Dataspace >` ds_cap, `l4_uint64_t` base, `l4_umword_t` offset, `l4_umword_t` size)
Register a shared data space with VIRTIO host.
- long `device_config` (`L4::lpc::Out< L4::Cap< L4Re::Dataspace > >` config_ds, `l4_addr_t` *ds_offset)
Get the dataspace with the L4virtio configuration page.
- long `device_notification_irq` (unsigned index, `L4::lpc::Out< L4::Cap< L4::Triggerable > >` irq)
Get the notification interrupt corresponding to the given index.

Public Member Functions inherited from L4::lcu

- `l4_msgtag_t` `bind` (unsigned irqnum, `L4::Cap< Triggerable >` irq, `l4_utcb_t` *utcb=`l4_utcb()`) noexcept
Bind an interrupt line of an interrupt controller to an interrupt object.
- `l4_msgtag_t` `unbind` (unsigned irqnum, `L4::Cap< Triggerable >` irq, `l4_utcb_t` *utcb=`l4_utcb()`) noexcept
Remove binding of an interrupt line from the interrupt controller object.
- `l4_msgtag_t` `info` (`l4_icu_info_t` *info, `l4_utcb_t` *utcb=`l4_utcb()`) noexcept
Get information about the ICU features.
- `l4_msgtag_t` `msi_info` (`l4_umword_t` irqnum, `l4_uint64_t` source, `l4_icu_msi_info_t` *msi_info)
Get MSI info about IRQ.
- `l4_msgtag_t` `mask` (unsigned irqnum, `l4_umword_t` *label=0, `l4_timeout_t` to=`L4_IPC_NEVER`, `l4_utcb_t` *utcb=`l4_utcb()`) noexcept
Mask an IRQ line.
- `l4_msgtag_t` `set_mode` (unsigned irqnum, `l4_umword_t` mode, `l4_utcb_t` *utcb=`l4_utcb()`) noexcept
Set interrupt mode.

Public Member Functions inherited from L4::lrq_eoi

- `l4_msgtag_t` `unmask` (unsigned irqnum, `l4_umword_t` *label=0, `l4_timeout_t` to=`L4_IPC_NEVER`, `l4_utcb_t` *utcb=`l4_utcb()`) noexcept
Unmask the given interrupt line.

Additional Inherited Members

Protected Types inherited from

`L4::Kobject_t< Device, L4::lcu, L4VIRTIO_PROTOCOL, L4::Type_info::Demand_t< 1 > >`

- typedef Device **Class**
The target interface type (inheriting from Kobject_t).
- typedef Typeid::Iface< PROTO, Device > **__Iface**
The interface description for the derived class.
- typedef Typeid::Merge_list< Typeid::Iface_list< **__Iface** >, typename L4::lcu::__Iface_list > **__Iface_list**
The list of all RPC interfaces provided directly or through inheritance.

Protected Types inherited from

`L4::Kobject_t< lcu, lrq_eoi, L4_PROTO_IRQ, Type_info::Demand_t< 1 > >`

- typedef lcu **Class**
The target interface type (inheriting from Kobject_t).
- typedef Typeid::Iface< PROTO, lcu > **__Iface**
The interface description for the derived class.
- typedef Typeid::Merge_list< Typeid::Iface_list< **__Iface** >, typename lrq_eoi::__Iface_list > **__Iface_list**
The list of all RPC interfaces provided directly or through inheritance.

Protected Member Functions inherited from**L4::Kobject_t< Device, L4::lcu, L4VIRTIO_PROTOCOL, L4::Type_info::Demand_t< 1 > >**

- **L4::Cap< Class > c ()** const noexcept

*Get the capability to ourselves.***Protected Member Functions inherited from****L4::Kobject_t< lcu, lrcu_eoi, L4_PROTO_IRQ, Type_info::Demand_t< 1 > >**

- **L4::Cap< Class > c ()** const noexcept

*Get the capability to ourselves.***Static Protected Member Functions inherited from****L4::Kobject_t< Device, L4::lcu, L4VIRTIO_PROTOCOL, L4::Type_info::Demand_t< 1 > >**

- static void **__check_protocols__ ()** noexcept

*Helper to check for protocol conflicts.***Static Protected Member Functions inherited from****L4::Kobject_t< lcu, lrcu_eoi, L4_PROTO_IRQ, Type_info::Demand_t< 1 > >**

- static void **__check_protocols__ ()** noexcept

*Helper to check for protocol conflicts.***15.394.1 Detailed Description**

IPC interface for virtio over [L4](#) IPC.

The [L4virtio](#) protocol is an adaption of the mmio virtio transport 1.0(4). This interface allows to exchange the necessary resources: device configuration page, notification interrupts and dataspace for payload.

Notification interrupts can be configured independently for changes to the configuration space and each queue through special L4virtio-specific `notify_index` fields in the config page and queue configuration. The interface distinguishes between device-to-driver and driver-to-device notification interrupts.

Device-to-driver interrupts are configured via the ICU interface. The device announces the maximum number of supported interrupts via `lcu::info()`. The driver can then bind interrupts using `lcu::bind()`.

Driver-to-device interrupts must be requested from the device through [device_notification_irq\(\)](#).

Definition at line 39 of file [l4virtio](#).

15.394.2 Member Function Documentation

15.394.2.1 config_queue()

```
long L4virtio::Device::config_queue (
    unsigned queue)
```

Trigger queue configuration of the given queue.

Usually all queues are configured when the status is written to running. However, in some cases queues shall be disabled or enabled dynamically, in this case this function triggers a reconfiguration from the shared memory register of the queue config.

Parameters

| | |
|--------------|---|
| <i>queue</i> | Queue index for the queue to be configured. |
|--------------|---|

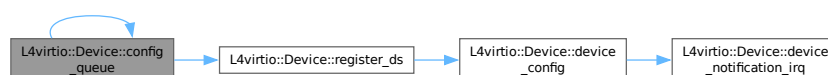
Return values

| | |
|------------|---|
| 0 | on success. |
| -L4_EIO | The queue's status is invalid. |
| -L4_ERANGE | The queue index exceeds the number of queues. |
| -L4_EINVAL | Otherwise. |

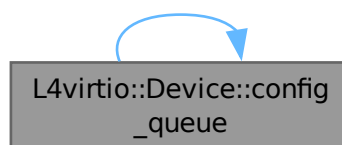
References [config_queue\(\)](#), [L4VIRTIO_OP_REGISTER_DS](#), and [register_ds\(\)](#).

Referenced by [config_queue\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.394.2.2 device_config()

```
long L4virtio::Device::device_config (
    L4::Ipc::Out< L4::Cap< L4Re::Dataspace > > config_ds,
    l4_addr_t * ds_offset)
```

Get the dataspace with the [L4virtio](#) configuration page.

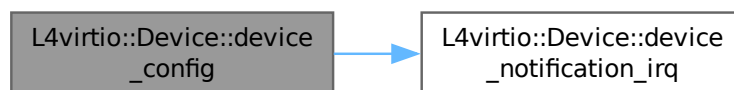
Parameters

| | |
|------------------|---|
| <i>config_ds</i> | Capability for receiving the dataspace capability for the shared L4-VIRTIO config data space. |
| <i>ds_offset</i> | Offset into the dataspace where the device configuration structure starts. |

References [device_notification_irq\(\)](#), and [L4VIRTIO_OP_GET_DEVICE_IRQ](#).

Referenced by [register_ds\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.394.2.3 device_notification_irq()

```
long L4virtio::Device::device_notification_irq (
    unsigned index,
    L4::Ipc::Out< L4::Cap< L4::Triggerable > > irq)
```

Get the notification interrupt corresponding to the given index.

Parameters

| | | |
|-----|--------------|----------------------------------|
| | <i>index</i> | Index of the interrupt. |
| out | <i>irq</i> | Triggerable for the given index. |

Return values

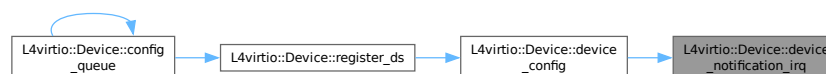
| | |
|------------------|---|
| <i>L4_EOK</i> | Success. |
| <i>L4_ENOSYS</i> | IRQ notification not supported by device. |
| <i><0</i> | Other error. |

An index is only guaranteed to return an IRQ object when the index is set in one of the device notify index fields. The device must return the same interrupt for a given index as long as the index is in use. If an index disappears as a result of a configuration change and then is reused later, the interrupt is not guaranteed to be the same.

Interrupts must always be rerequested after a device reset.

Referenced by [device_config\(\)](#).

Here is the caller graph for this function:



15.394.2.4 register_ds()

```

long L4virtio::Device::register_ds (
    L4::Ipc::Cap< L4Re::Dataspace > ds_cap,
    l4_uint64_t base,
    l4_umword_t offset,
    l4_umword_t size)

```

Register a shared data space with VIRTIO host.

Parameters

| | |
|---------------|--|
| <i>ds_cap</i> | Dataspace capability to register. The lower 8 bits determine the rights mask with which the guest's rights are masked during the registration of the dataspace at the VIRTIO host. |
| <i>base</i> | VIRTIO guest physical start address of shared memory region |
| <i>offset</i> | Offset within the data space that is attached to the given <i>base</i> in the guest physical memory. |
| <i>size</i> | Size of the memory region in the guest |

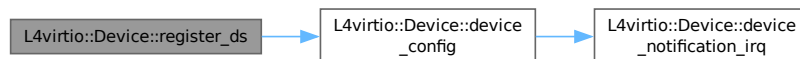
Return values

| | |
|-------------------------|--|
| <code>L4_EOK</code> | Operation successful. |
| <code>-L4_EINVAL</code> | The <code>ds_cap</code> capability is invalid, does not refer to a valid dataspace, is not a trusted dataspace if trusted dataspace validation is enabled, or <code>size</code> and <code>offset</code> specify an invalid region. |
| <code>-L4_ENOMEM</code> | The limit of dataspaces that can be registered has been reached or no capability slot could be allocated. |
| <code>-L4_ERANGE</code> | <code>offset</code> is larger than the size of the dataspace. |
| <code><0</code> | Any error returned by the dataspace when queried for information during setup or any error returned by the region manager from attaching the dataspace. |

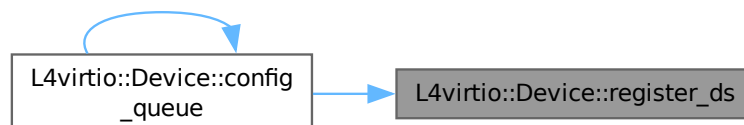
References [device_config\(\)](#), and [L4VIRTIO_OP_DEVICE_CONFIG](#).

Referenced by [config_queue\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.394.2.5 set_status()

```
long L4virtio::Device::set_status (
    unsigned status)
```

Write the VIRTIO status register.

Parameters

| | |
|---------------------|--|
| <code>status</code> | Status word to write to the VIRTIO status. |
|---------------------|--|

Return values

| | |
|---|-------------|
| 0 | on success. |
|---|-------------|

Note

All other registers are accessed via shared memory.

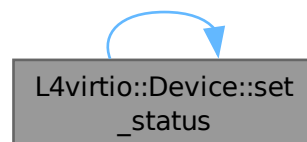
References [L4VIRTIO_OP_CONFIG_QUEUE](#), and [set_status\(\)](#).

Referenced by [set_status\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

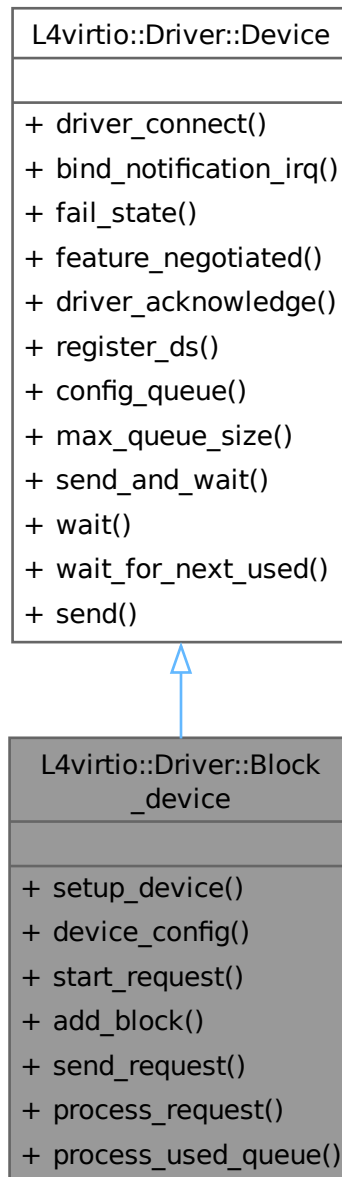
- `I4/I4virtio/I4virtio`

15.395 L4virtio::Driver::Block_device Class Reference

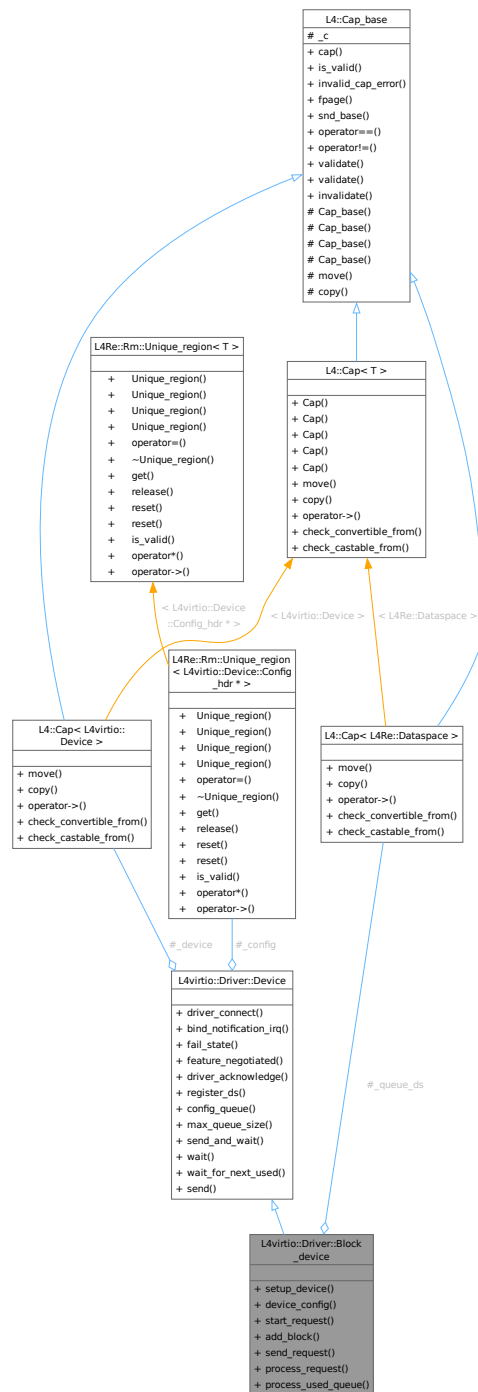
Simple class for accessing a virtio block device synchronously.

```
#include <virtio-block>
```

Inheritance diagram for L4virtio::Driver::Block_device:



Collaboration diagram for L4virtio::Driver::Block_device:



Data Structures

- class [Handle](#)

Handle to an ongoing request.

Public Member Functions

- void `setup_device` (`L4::Cap`< `L4virtio::Device` > `srvcap`, `l4_size_t` `usermem`, void `**userdata`, `Ptr`< void > `&user_devaddr`, `L4::Cap`< `L4Re::Dataspace` > `qds=L4::Cap`< `L4Re::Dataspace` >(), `l4_uint32_t` `fmask0=-1U`, `l4_uint32_t` `fmask1=-1U`)
Establish a connection to the device and set up shared memory.
- `l4virtio_block_config_t` const & `device_config` () const
Return a reference to the device configuration.
- `Handle` `start_request` (`l4_uint64_t` `sector`, `l4_uint32_t` `type`, `Callback` `callback`)
Start the setup of a new request.
- int `add_block` (`Handle` `handle`, `Ptr`< void > `addr`, `l4_uint32_t` `size`)
Add a data block to a request that has already been set up.
- int `send_request` (`Handle` `handle`)
Process request asynchronously.
- int `process_request` (`Handle` `handle`)
Process request synchronously.
- void `process_used_queue` ()
Process and free all items in the used queue.

Public Member Functions inherited from `L4virtio::Driver::Device`

- void `driver_connect` (`L4::Cap`< `L4virtio::Device` > `srvcap`, bool `manage_notify=true`)
Contacts the device and starts the initial handshake.
- int `bind_notification_irq` (unsigned index, `L4::Cap`< `L4::Triggerable` > `irq`) const
Register a triggerable to receive notifications from the device.
- bool `fail_state` () const
Return true if the device is in a fail state.
- bool `feature_negotiated` (unsigned int `feat`) const
Check if a particular feature bit was negotiated with the device.
- int `driver_acknowledge` ()
Finalize handshake with the device.
- int `register_ds` (`L4::Cap`< `L4Re::Dataspace` > `ds`, `l4_umword_t` `offset`, `l4_umword_t` `size`, `l4_uint64_t` `*devaddr`)
Share a dataspace with the device.
- int `config_queue` (int `num`, unsigned `size`, `l4_uint64_t` `desc_addr`, `l4_uint64_t` `avail_addr`, `l4_uint64_t` `used_↔addr`)
Send the virtqueue configuration to the device.
- int `max_queue_size` (int `num`) const
Maximum queue size allowed by the device.
- int `send_and_wait` (`Virtqueue` &`queue`, `l4_uint16_t` `descno`)
Send a request to the device and wait for it to be processed.
- int `wait` (int index) const
Wait for a notification from the device.
- int `wait_for_next_used` (`Virtqueue` &`queue`, `l4_uint32_t` `*len=nullptr`) const
Wait for the next item to arrive in the used queue and return it.
- void `send` (`Virtqueue` &`queue`, `l4_uint16_t` `descno`)
Send a request to the device.

15.395.1 Detailed Description

Simple class for accessing a virtio block device synchronously.

Definition at line 36 of file [virtio-block](#).

15.395.2 Member Function Documentation

15.395.2.1 add_block()

```
int L4virtio::Driver::Block_device::add_block (
    Handle handle,
    Ptr< void > addr,
    l4_uint32_t size) [inline]
```

Add a data block to a request that has already been set up.

Parameters

| | |
|---------------|--|
| <i>handle</i> | Handle to request previously set up with start_request() . |
| <i>addr</i> | Address of data block in device address space. |
| <i>size</i> | Size of data block. |

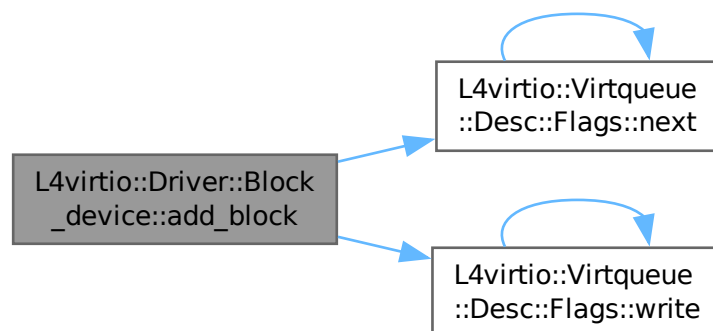
Return values

| | |
|-------------------|--|
| <i>L4_OK</i> | Block was successfully added. |
| <i>-L4_EAGAIN</i> | No descriptors available. Try again later. |

Definition at line 229 of file [virtio-block](#).

References [L4virtio::Virtqueue::Desc::addr](#), [L4virtio::Virtqueue::Desc::flags](#), [L4_EAGAIN](#), [L4_EOK](#), [L4virtio::Virtqueue::Desc::len](#), [L4virtio::Virtqueue::Desc::Flags::next\(\)](#), [L4virtio::Virtqueue::Desc::next](#), [L4virtio::Virtqueue::Desc::Flags::raw](#), and [L4virtio::Virtqueue::Desc::Flags::write\(\)](#).

Here is the call graph for this function:



15.395.2.2 process_request()

```
int L4virtio::Driver::Block_device::process_request (
    Handle handle) [inline]
```

Process request synchronously.

Parameters

| | |
|---------------|-------------------------------|
| <i>handle</i> | Handle to request to process. |
|---------------|-------------------------------|

Return values

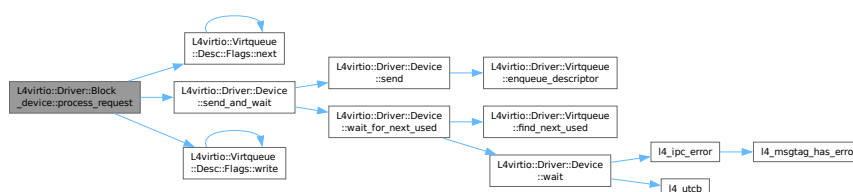
| | |
|-------------------|--|
| <i>L4_EOK</i> | Request processed successfully. |
| <i>-L4_EAGAIN</i> | No descriptors available. Try again later. |
| <i>-L4_EIO</i> | IO error during request processing. |
| <i>-L4_ENOSYS</i> | Unsupported request. |
| <i><0</i> | Another unspecified error occurred. |

Sends a request to the driver that was previously set up with [start_request\(\)](#) and [add_block\(\)](#) and wait for it to be executed.

Definition at line 306 of file [virtio-block](#).

References [L4virtio::Virtqueue::Desc::addr](#), [L4virtio::Virtqueue::Desc::flags](#), [L4_EAGAIN](#), [L4_EINVAL](#), [L4_EIO](#), [L4_ENOSYS](#), [L4_EOK](#), [L4VIRTIO_BLOCK_S_IOERR](#), [L4VIRTIO_BLOCK_S_OK](#), [L4VIRTIO_BLOCK_S_UNSUPP](#), [L4virtio::Virtqueue::Desc::len](#), [L4virtio::Virtqueue::Desc::Flags::next\(\)](#), [L4virtio::Virtqueue::Desc::next](#), [L4virtio::Virtqueue::Desc::Flags::write\(\)](#), [L4virtio::Driver::Device::send_and_wait\(\)](#), and [L4virtio::Virtqueue::Desc::Flags::write\(\)](#).

Here is the call graph for this function:



15.395.2.3 process_used_queue()

```
void L4virtio::Driver::Block_device::process_used_queue () [inline]
```

Process and free all items in the used queue.

If the request has a callback registered it is called after the item has been removed from the queue.

Definition at line 357 of file [virtio-block](#).

References [L4Re::chksys\(\)](#), and [L4_ENOSYS](#).

Here is the call graph for this function:



15.395.2.4 send_request()

```
int L4virtio::Driver::Block_device::send_request (
    Handle handle) [inline]
```

Process request asynchronously.

Parameters

| | |
|---------------|---|
| <i>handle</i> | Handle to request to send to the device |
|---------------|---|

Return values

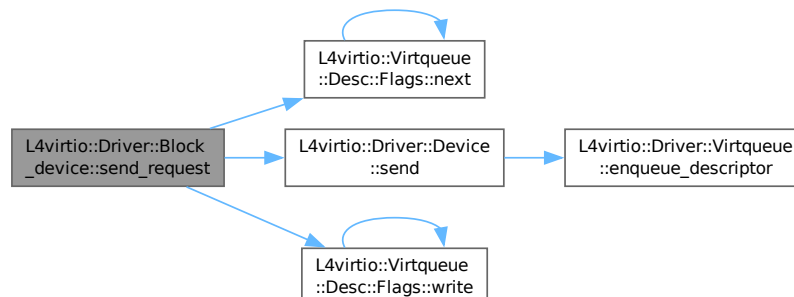
| | |
|-------------------|--|
| <i>L4_OK</i> | Request was successfully scheduled. |
| <i>-L4_EAGAIN</i> | No descriptors available. Try again later. |

Sends a request to the driver that was previously set up with [start_request\(\)](#) and [add_block\(\)](#) and wait for it to be executed.

Definition at line 265 of file [virtio-block](#).

References [L4virtio::Virtqueue::Desc::addr](#), [L4virtio::Virtqueue::Desc::flags](#), [L4_EAGAIN](#), [L4_EOK](#), [L4virtio::Virtqueue::Desc::len](#), [L4virtio::Virtqueue::Desc::Flags::next\(\)](#), [L4virtio::Virtqueue::Desc::next](#), [L4virtio::Virtqueue::Desc::Flags::raw](#), [L4virtio::Driver::Device::send\(\)](#), and [L4virtio::Virtqueue::Desc::Flags::write\(\)](#).

Here is the call graph for this function:



15.395.2.5 `setup_device()`

```
void L4virtio::Driver::Block_device::setup_device (
    L4::Cap< L4virtio::Device > srvcap,
    l4_size_t usermem,
    void ** userdata,
    Ptr< void > & user_devaddr,
    L4::Cap< L4Re::Dataspace > qds = L4::Cap<L4Re::Dataspace>(),
    l4_uint32_t fmask0 = -1U,
    l4_uint32_t fmask1 = -1U) [inline]
```

Establish a connection to the device and set up shared memory.

Parameters

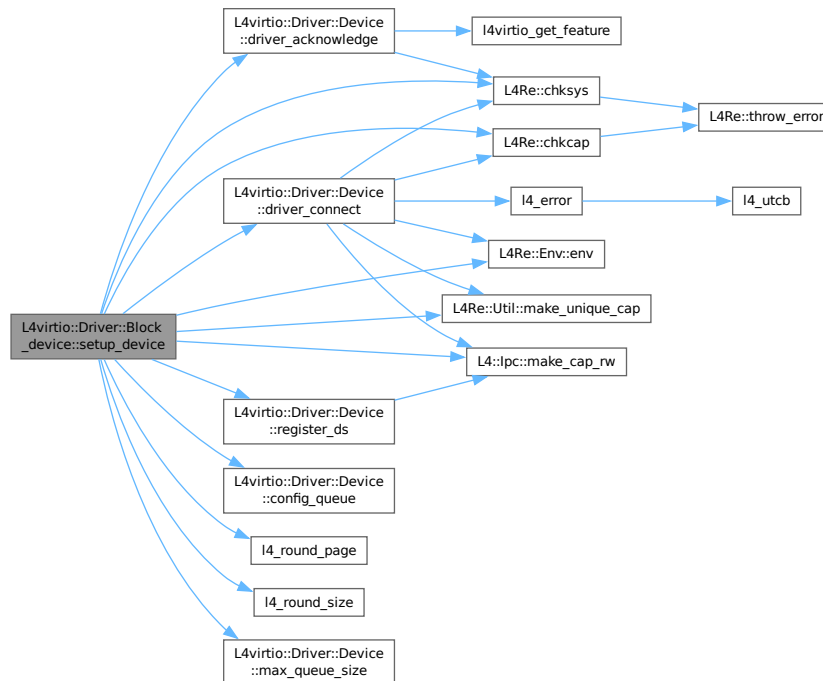
| | | |
|-----|---------------------|---|
| | <i>srvcap</i> | IPC capability of the channel to the server. |
| | <i>usermem</i> | Size of additional memory to share with device. |
| out | <i>userdata</i> | Pointer to the region of user-usable memory. |
| out | <i>user_devaddr</i> | Address of user-usable memory in device address space. |
| | <i>qds</i> | External queue dataspace. If this capability is invalid, the function will attempt to allocate a dataspace on its own. Note that the external queue dataspace must be large enough. |
| | <i>fmask0</i> | Feature bits 0..31 that the driver supports. |
| | <i>fmask1</i> | Feature bits 32..63 that the driver supports. |

This function starts a handshake with the device and sets up the virtqueues for communication and the additional data structures for the block device. It will also allocate and share additional memory that the caller then can use freely, i.e. normally this memory would be used as a reception buffer. The caller may also decide to not make use of this convenience function and request 0 bytes in *usermem*. Then it has to allocate the block buffers for sending/receiving payload manually and share them using [register_ds\(\)](#).

Definition at line 92 of file [virtio-block](#).

References [L4Re::chkcap\(\)](#), [L4Re::chksys\(\)](#), [L4virtio::Driver::Device::config_queue\(\)](#), [L4Re::Mem_alloc::Continuous](#), [L4virtio::Driver::Device::driver_acknowledge\(\)](#), [L4virtio::Driver::Device::driver_connect\(\)](#), [L4Re::Env::env\(\)](#), [L4_EINVAL](#), [L4_ENODEV](#), [L4_PAGESHIFT](#), [l4_round_page\(\)](#), [l4_round_size\(\)](#), [L4VIRTIO_ID_BLOCK](#), [L4::lpc::make_cap_rw\(\)](#), [L4Re::Util::make_unique_cap\(\)](#), [L4virtio::Driver::Device::max_queue_size\(\)](#), [L4Re::Mem_alloc::Pinned](#), [L4virtio::Driver::Device::register_ds\(\)](#), [L4Re::Rm::F::RW](#), and [L4Re::Rm::F::Search_addr](#).

Here is the call graph for this function:



15.395.2.6 start_request()

```

Handle L4virtio::Driver::Block_device::start_request (
    l4_uint64_t sector,
    l4_uint32_t type,
    Callback callback) [inline]

```

Start the setup of a new request.

Parameters

| | |
|-----------------|--|
| <i>sector</i> | First sector to write to/read from. |
| <i>type</i> | Request type. |
| <i>callback</i> | Function to call, when the request is finished. May be 0 for synchronous requests. |

Definition at line 191 of file [virtio-block](#).

References [L4virtio::Virtqueue::Desc::addr](#), [L4virtio::Virtqueue::Desc::flags](#), [l4virtio_block_header_t::ioprio](#), [L4virtio::Virtqueue::Desc::len](#), [L4virtio::Virtqueue::Desc::Flags::raw](#), [l4virtio_block_header_t::sector](#), and [l4virtio_block_header_t::type](#).

The documentation for this class was generated from the following file:

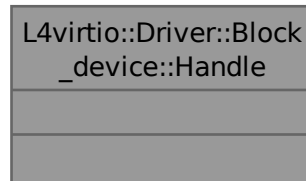
- [l4/l4virtio/client/virtio-block](#)

15.396 L4virtio::Driver::Block_device::Handle Class Reference

[Handle](#) to an ongoing request.

```
#include <virtio-block>
```

Collaboration diagram for L4virtio::Driver::Block_device::Handle:



15.396.1 Detailed Description

[Handle](#) to an ongoing request.

Definition at line 56 of file [virtio-block](#).

The documentation for this class was generated from the following file:

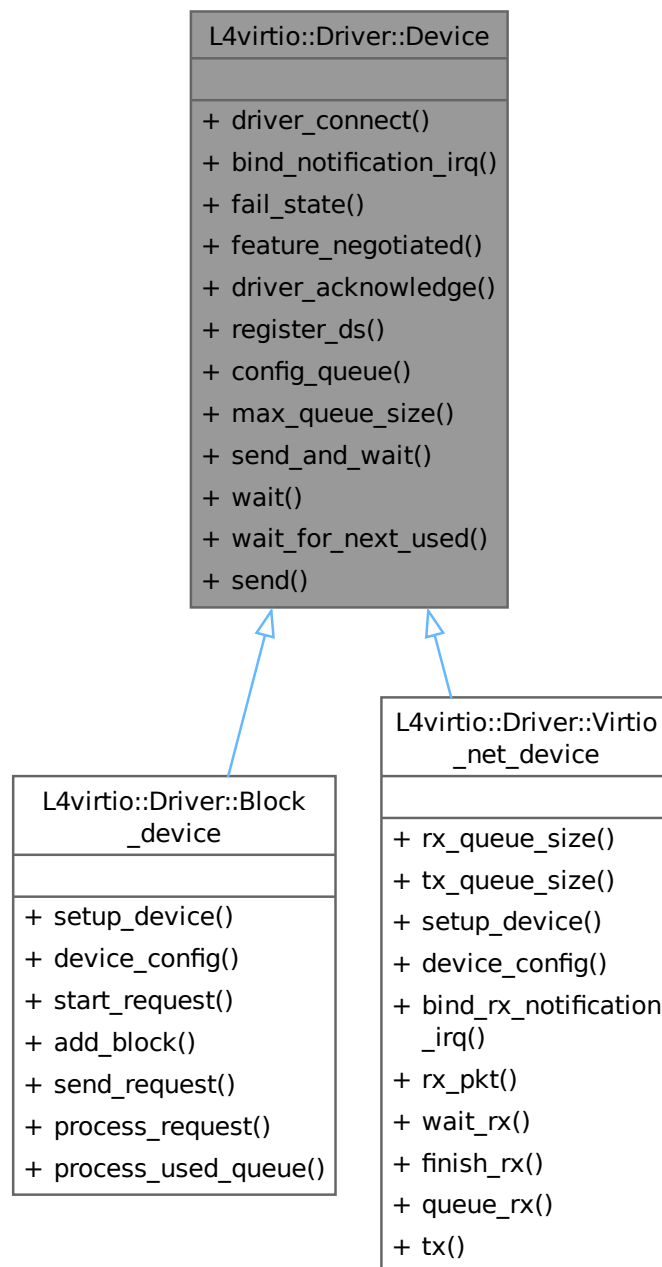
- l4/l4virtio/client/virtio-block

15.397 L4virtio::Driver::Device Class Reference

Client-side implementation for a general virtio device.

```
#include <l4virtio>
```

Inheritance diagram for L4virtio::Driver::Device:





- Generated for L4Re by Doxygen

- Return true if the device is in a fail state.*

 - bool [feature_negotiated](#) (unsigned int feat) const

Check if a particular feature bit was negotiated with the device.
- int [driver_acknowledge](#) ()

Finalize handshake with the device.
- int [register_ds](#) (L4::Cap< L4Re::Dataspace > ds, l4_umword_t offset, l4_umword_t size, l4_uint64_t *devaddr)

Share a dataspace with the device.
- int [config_queue](#) (int num, unsigned size, l4_uint64_t desc_addr, l4_uint64_t avail_addr, l4_uint64_t used_addr)

Send the virtqueue configuration to the device.
- int [max_queue_size](#) (int num) const

Maximum queue size allowed by the device.
- int [send_and_wait](#) (Virtqueue &queue, l4_uint16_t descno)

Send a request to the device and wait for it to be processed.
- int [wait](#) (int index) const

Wait for a notification from the device.
- int [wait_for_next_used](#) (Virtqueue &queue, l4_uint32_t *len=NULLPTR) const

Wait for the next item to arrive in the used queue and return it.
- void [send](#) (Virtqueue &queue, l4_uint16_t descno)

Send a request to the device.

15.397.1 Detailed Description

Client-side implementation for a general virtio device.

Definition at line 31 of file [l4virtio](#).

15.397.2 Member Function Documentation

15.397.2.1 bind_notification_irq()

```
int L4virtio::Driver::Device::bind_notification_irq (
    unsigned index,
    L4::Cap< L4::Triggerable > irq) const [inline]
```

Register a triggerable to receive notifications from the device.

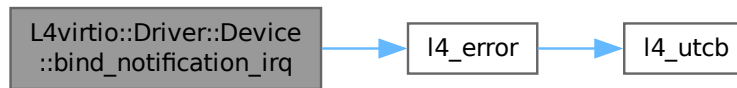
Parameters

| | | |
|-----|--------------|--|
| | <i>index</i> | Index of the interrupt. |
| out | <i>irq</i> | Triggerable to register for notifications. |

Definition at line 129 of file [l4virtio](#).

References [l4_error\(\)](#).

Here is the call graph for this function:



15.397.2.2 config_queue()

```

int L4virtio::Driver::Device::config_queue (
    int num,
    unsigned size,
    l4_uint64_t desc_addr,
    l4_uint64_t avail_addr,
    l4_uint64_t used_addr) [inline]
  
```

Send the virtqueue configuration to the device.

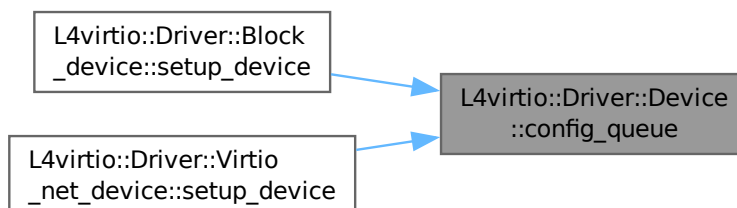
Parameters

| | |
|-------------------|---|
| <i>num</i> | Number of queue to configure. |
| <i>size</i> | Size of rings in the queue, must be a power of 2) |
| <i>desc_addr</i> | Address of descriptor table (device address) |
| <i>avail_addr</i> | Address of available ring (device address) |
| <i>used_addr</i> | Address of used ring (device address) |

Definition at line 212 of file [l4virtio](#).

Referenced by [L4virtio::Driver::Block_device::setup_device\(\)](#), and [L4virtio::Driver::Virtio_net_device::setup_device\(\)](#).

Here is the caller graph for this function:



15.397.2.3 driver_acknowledge()

```
int L4virtio::Driver::Device::driver_acknowledge () [inline]
```

Finalize handshake with the device.

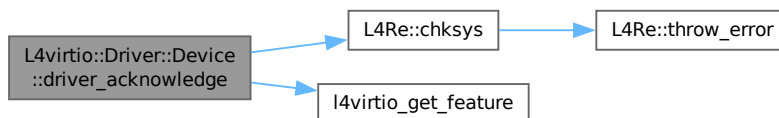
Must be called after all queues have been set up and before the first request is sent. It is still possible to add more shared dataspace after the handshake has been finished.

Definition at line 156 of file [l4virtio](#).

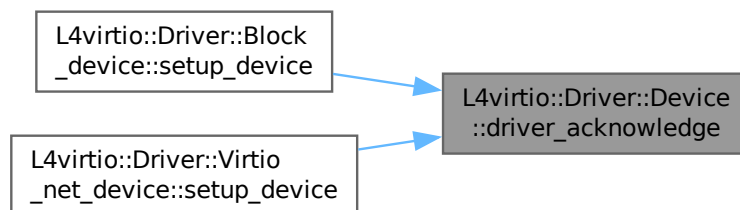
References [L4Re::chksys\(\)](#), [L4_EINVAL](#), [L4_EIO](#), [L4_ENODEV](#), [L4_EOK](#), [L4VIRTIO_FEATURE_VERSION_1](#), [l4virtio_get_feature\(\)](#), [L4VIRTIO_STATUS_DRIVER_OK](#), and [L4VIRTIO_STATUS_FEATURES_OK](#).

Referenced by [L4virtio::Driver::Block_device::setup_device\(\)](#), and [L4virtio::Driver::Virtio_net_device::setup_device\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.397.2.4 driver_connect()

```
void L4virtio::Driver::Device::driver_connect (
    L4::Cap< L4virtio::Device > srvcap,
    bool manage_notify = true) [inline]
```

Contacts the device and starts the initial handshake.

Parameters

| | |
|----------------------|--|
| <i>svrcap</i> | Capability for device communication. |
| <i>manage_notify</i> | Set up a semaphore for notifications from the device. See below. |

Exceptions

| | |
|-----------------------------------|-----------------------------|
| L4::Runtime_error | if the initialisation fails |
|-----------------------------------|-----------------------------|

This function contacts the server, sets up the notification channels and the configuration dataspace. After this is done, the caller can set up any dataspaces it needs. The initialisation then needs to be finished by calling [driver_acknowledge\(\)](#).

Per default this function creates and registers a semaphore for receiving notification from the device. This semaphore is used in the blocking functions [send_and_wait\(\)](#), [wait\(\)](#) and [next_used\(\)](#).

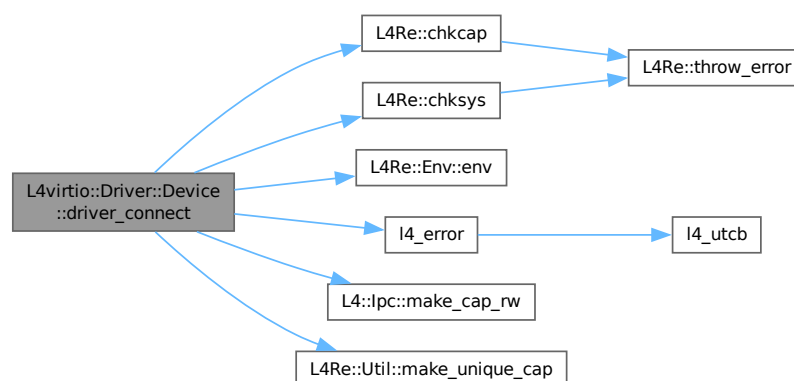
When `manage_notify` is false, then the caller may manually register and handle notification interrupts from the device. This is for example useful, when the client runs in an application with a server loop.

Definition at line 56 of file [l4virtio](#).

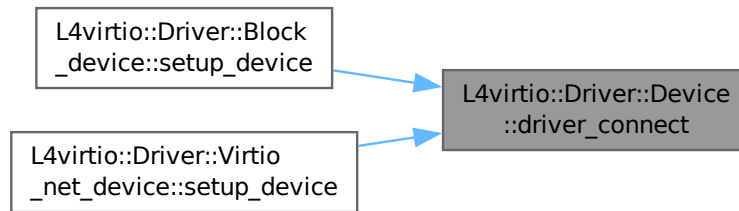
References [L4Re::chkcap\(\)](#), [L4Re::chksys\(\)](#), [L4Re::Env::env\(\)](#), [L4_EINVAL](#), [L4_EIO](#), [L4_ENODEV](#), [l4_error\(\)](#), [L4_PAGEMASK](#), [L4_PAGESHIFT](#), [L4_PAGESIZE](#), [L4_SUPERPAGESIZE](#), [L4VIRTIO_STATUS_ACKNOWLEDGE](#), [L4VIRTIO_STATUS_DRIVER](#), [L4::lpc::make_cap_rw\(\)](#), [L4Re::Util::make_unique_cap\(\)](#), [L4Re::Rm::F::RW](#), and [L4Re::Rm::F::Search_addr](#).

Referenced by [L4virtio::Driver::Block_device::setup_device\(\)](#), and [L4virtio::Driver::Virtio_net_device::setup_device\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.397.2.5 feature_negotiated()

```
bool L4virtio::Driver::Device::feature_negotiated (
    unsigned int feat) const [inline]
```

Check if a particular feature bit was negotiated with the device.

The result is only valid after [driver_acknowledge\(\)](#) was called (when the handshake with the device was completed).

Parameters

| | |
|-------------|------------------|
| <i>feat</i> | The feature bit. |
|-------------|------------------|

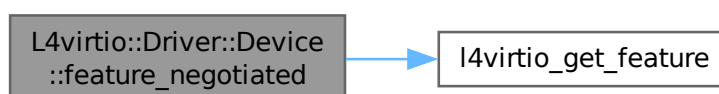
Return values

| | |
|--------------|---|
| <i>true</i> | The feature is supported by both driver and device. |
| <i>false</i> | The feature is not supported by the driver and/or device. |

Definition at line [145](#) of file [l4virtio](#).

References [l4virtio_get_feature\(\)](#).

Here is the call graph for this function:



15.397.2.6 max_queue_size()

```
int L4virtio::Driver::Device::max_queue_size (
    int num) const [inline]
```

Maximum queue size allowed by the device.

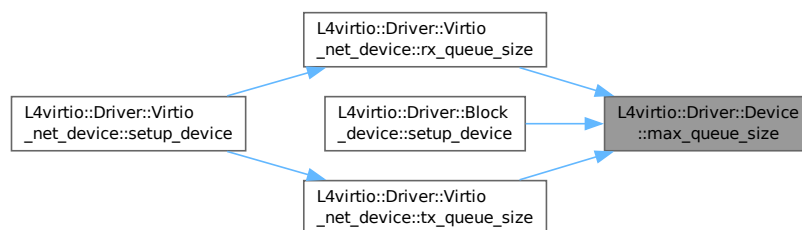
Parameters

| | |
|------------|--|
| <i>num</i> | Number of queue for which to determine the maximum size. |
|------------|--|

Definition at line 230 of file [l4virtio](#).

Referenced by [L4virtio::Driver::Virtio_net_device::rx_queue_size\(\)](#), [L4virtio::Driver::Block_device::setup_device\(\)](#), and [L4virtio::Driver::Virtio_net_device::tx_queue_size\(\)](#).

Here is the caller graph for this function:



15.397.2.7 register_ds()

```
int L4virtio::Driver::Device::register_ds (
    L4::Cap< L4Re::Dataspace > ds,
    l4_umword_t offset,
    l4_umword_t size,
    l4_uint64_t * devaddr) [inline]
```

Share a dataspace with the device.

Parameters

| | |
|----------------|--|
| <i>ds</i> | Dataspace to share with the device. |
| <i>offset</i> | Offset in dataspace where the shared part starts. |
| <i>size</i> | Total size in bytes of the shared space. |
| <i>devaddr</i> | Start of shared space in the device address space. |

Although this function allows to share only a part of the given dataspace for convenience, the granularity of sharing is always the dataspace level. Thus, the remainder of the dataspace is not protected from the device.

When communicating with the device, addresses must be given with respect to the device address space. This is not the same as the virtual address space of the client in order to not leak information about the address space layout.

Definition at line 196 of file [l4virtio](#).

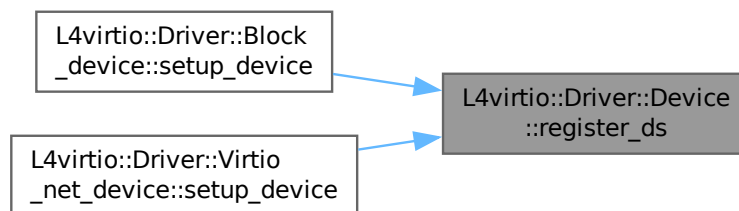
References [L4::lpc::make_cap_rw\(\)](#).

Referenced by [L4virtio::Driver::Block_device::setup_device\(\)](#), and [L4virtio::Driver::Virtio_net_device::setup_device\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.397.2.8 send()

```
void L4virtio::Driver::Device::send (
    Virtqueue & queue,
    l4_uint16_t descno) [inline]
```

Send a request to the device.

Parameters

| | |
|---------------|---|
| <i>queue</i> | Queue that contains the request in its descriptor table |
| <i>descno</i> | Index of first entry in descriptor table where |

Definition at line 312 of file [l4virtio](#).

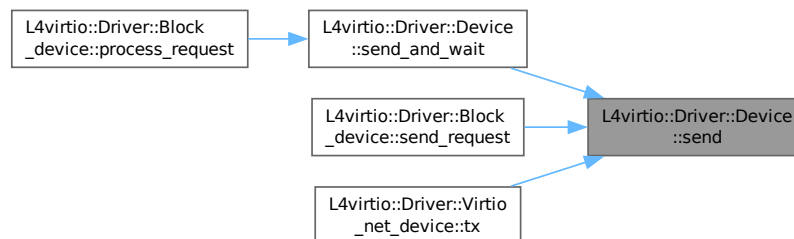
References [L4virtio::Driver::Virtqueue::enqueue_descriptor\(\)](#).

Referenced by [send_and_wait\(\)](#), [L4virtio::Driver::Block_device::send_request\(\)](#), and [L4virtio::Driver::Virtio_net_device::tx\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.397.2.9 send_and_wait()

```
int L4virtio::Driver::Device::send_and_wait (
    Virtqueue & queue,
    l4_uint16_t descno) [inline]
```

Send a request to the device and wait for it to be processed.

Parameters

| | |
|---------------|---|
| <i>queue</i> | Queue that contains the request in its descriptor table |
| <i>descno</i> | Index of first entry in descriptor table where |

This function provides a simple mechanism to send requests synchronously. It must not be used with other requests at the same time as it directly waits for a notification on the device irq cap.

Precondition

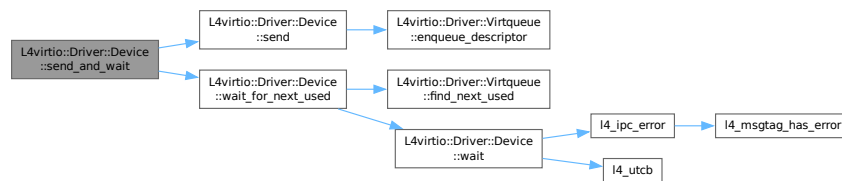
[driver_connect\(\)](#) was called with `manage_notify`.

Definition at line 247 of file [l4virtio](#).

References [L4_EINVAL](#), [L4_EOK](#), [send\(\)](#), and [wait_for_next_used\(\)](#).

Referenced by [L4virtio::Driver::Block_device::process_request\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**15.397.2.10 wait()**

```
int L4virtio::Driver::Device::wait (
    int index) const [inline]
```

Wait for a notification from the device.

Parameters

| | |
|--------------|--------------------------------|
| <i>index</i> | Notification slot to wait for. |
|--------------|--------------------------------|

Precondition

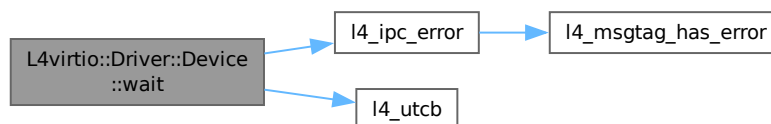
[driver_connect\(\)](#) was called with `manage_notify`.

Definition at line 268 of file [l4virtio](#).

References [L4_EEXIST](#), [l4_ipc_error\(\)](#), and [l4_utcb\(\)](#).

Referenced by [wait_for_next_used\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**15.397.2.11 wait_for_next_used()**

```
int L4virtio::Driver::Device::wait_for_next_used (
    Virtqueue & queue,
    l4_uint32_t * len = nullptr) const [inline]
```

Wait for the next item to arrive in the used queue and return it.

Parameters

| | | |
|-----|--------------|--|
| | <i>queue</i> | A queue. |
| out | <i>len</i> | (optional) Size of valid data in finished block. Note that this is the value reported by the device, which may set it to a value that is larger than the original buffer size. |

Return values

| | |
|----------|--|
| ≥ 0 | Descriptor number of item removed from used queue. |
| < 0 | IPC error while waiting for notification. |

The call blocks until the next item is available in the used queue.

Precondition

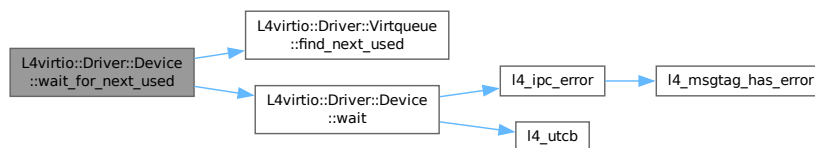
`driver_connect()` was called with `manage_notify`.

Definition at line 291 of file `l4virtio`.

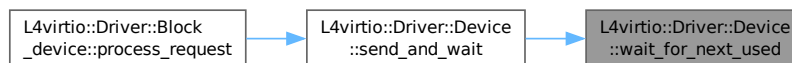
References `L4virtio::Driver::Virtqueue::find_next_used()`, and `wait()`.

Referenced by `send_and_wait()`.

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

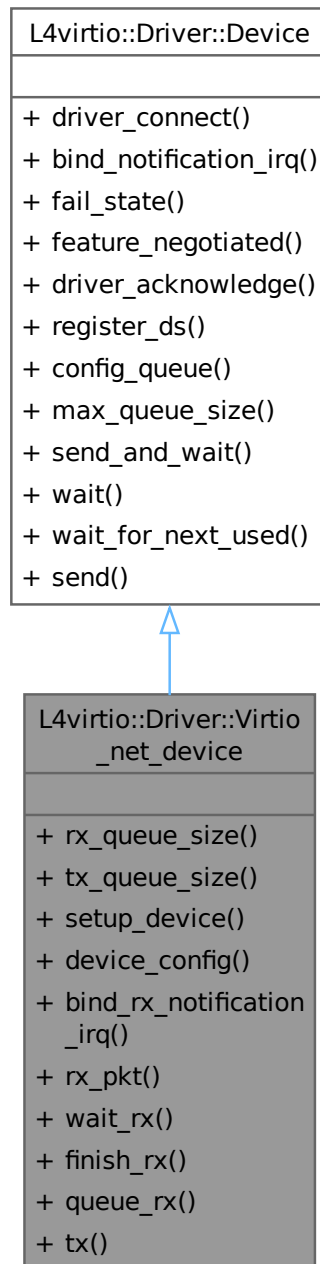
- `I4/I4virtio/client/I4virtio`

15.398 L4virtio::Driver::Virtio_net_device Class Reference

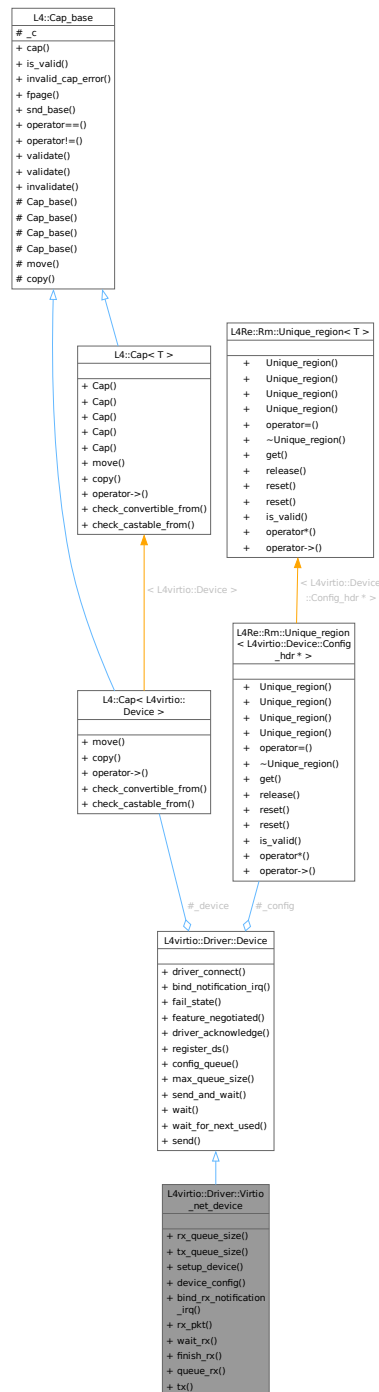
Simple class for accessing a virtio net device.

```
#include <virtio-net>
```

Inheritance diagram for L4virtio::Driver::Virtio_net_device:



Collaboration diagram for L4virtio::Driver::Virtio_net_device:



Data Structures

- struct [Packet](#)

Structure for a network packet (header including data) with maximum size, assuming that no extra features have been negotiated.

Public Member Functions

- int [rx_queue_size](#) () const
Return the maximum receive queue size allowed by the device.
- int [tx_queue_size](#) () const
Return the maximum transmit queue size allowed by the device.
- void [setup_device](#) (L4::Cap< L4virtio::Device > srvcap, L4::Cap< L4Re::Dataspace > queue_ds=L4::Cap< L4Re::Dataspace >::Invalid)
Establish a connection to the device and set up shared memory.
- [l4virtio_net_config_t](#) const & [device_config](#) () const
Return a reference to the device configuration.
- int [bind_rx_notification_irq](#) (L4::Cap< L4::Thread > thread, [l4_umword_t](#) label)
Bind the rx notification IRQ to the specified thread.
- [Packet](#) & [rx_pkt](#) ([l4_uint16_t](#) descno)
Return a reference to the RX packet buffer of the specified descriptor, e.g.
- [l4_uint16_t](#) [wait_rx](#) ([l4_uint32_t](#) *len=nullptr)
Block until a network packet has been received from the device and return the descriptor number.
- void [finish_rx](#) ([l4_uint16_t](#) descno)
Free an RX descriptor number to make it available for the RX queue again.
- void [queue_rx](#) ()
Queue new available descriptors in the RX queue.
- bool [tx](#) (std::function< [l4_uint32_t](#)([Packet](#) &)> prepare)
Attempt to allocate a descriptor in the TX queue and transmit the packet, after calling the prepare callback.

Public Member Functions inherited from [L4virtio::Driver::Device](#)

- void [driver_connect](#) (L4::Cap< L4virtio::Device > srvcap, bool manage_notify=true)
Contacts the device and starts the initial handshake.
- int [bind_notification_irq](#) (unsigned index, L4::Cap< L4::Triggerable > irq) const
Register a triggerable to receive notifications from the device.
- bool [fail_state](#) () const
Return true if the device is in a fail state.
- bool [feature_negotiated](#) (unsigned int feat) const
Check if a particular feature bit was negotiated with the device.
- int [driver_acknowledge](#) ()
Finalize handshake with the device.
- int [register_ds](#) (L4::Cap< L4Re::Dataspace > ds, [l4_umword_t](#) offset, [l4_umword_t](#) size, [l4_uint64_t](#) *devaddr)
Share a dataspace with the device.
- int [config_queue](#) (int num, unsigned size, [l4_uint64_t](#) desc_addr, [l4_uint64_t](#) avail_addr, [l4_uint64_t](#) used_↵ addr)
Send the virtqueue configuration to the device.
- int [max_queue_size](#) (int num) const
Maximum queue size allowed by the device.
- int [send_and_wait](#) (Virtqueue &queue, [l4_uint16_t](#) descno)
Send a request to the device and wait for it to be processed.
- int [wait](#) (int index) const
Wait for a notification from the device.
- int [wait_for_next_used](#) (Virtqueue &queue, [l4_uint32_t](#) *len=nullptr) const
Wait for the next item to arrive in the used queue and return it.
- void [send](#) (Virtqueue &queue, [l4_uint16_t](#) descno)
Send a request to the device.

15.398.1 Detailed Description

Simple class for accessing a virtio net device.

Definition at line 30 of file [virtio-net](#).

15.398.2 Member Function Documentation

15.398.2.1 bind_rx_notification_irq()

```
int L4virtio::Driver::Virtio_net_device::bind_rx_notification_irq (  
    L4::Cap< L4::Thread > thread,  
    l4_umword_t label) [inline]
```

Bind the rx notification IRQ to the specified thread.

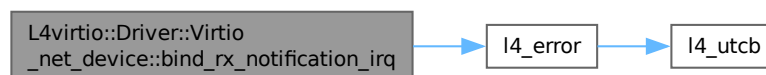
Parameters

| | |
|---------------|---|
| <i>thread</i> | Thread to bind the notification IRQ to. |
| <i>label</i> | Label to assign to the IRQ. |

Definition at line 181 of file [virtio-net](#).

References [l4_error\(\)](#).

Here is the call graph for this function:



15.398.2.2 finish_rx()

```
void L4virtio::Driver::Virtio_net_device::finish_rx (  
    l4_uint16_t descno) [inline]
```

Free an RX descriptor number to make it available for the RX queue again.

Parameters

| | |
|---------------|--|
| <i>descno</i> | Descriptor number in the virtio queue. |
|---------------|--|

Usually [queue_rx\(\)](#) should be called afterwards to queue the freed descriptor(s).

Definition at line 242 of file [virtio-net](#).

15.398.2.3 rx_pkt()

```
Packet & L4virtio::Driver::Virtio_net_device::rx_pkt (
    l4_uint16_t descno) [inline]
```

Return a reference to the RX packet buffer of the specified descriptor, e.g.

from [wait_rx\(\)](#).

Parameters

| | |
|---------------|--|
| <i>descno</i> | Descriptor number in the virtio queue. |
|---------------|--|

Definition at line 192 of file [virtio-net](#).

15.398.2.4 rx_queue_size()

```
int L4virtio::Driver::Virtio_net_device::rx_queue_size () const [inline]
```

Return the maximum receive queue size allowed by the device.

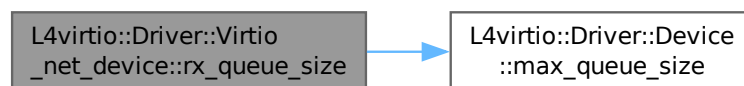
[wait_rx\(\)](#) will return a descriptor number that is smaller than this size.

Definition at line 47 of file [virtio-net](#).

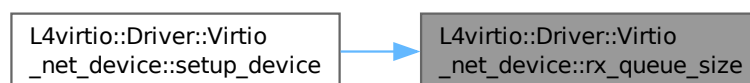
References [L4virtio::Driver::Device::max_queue_size\(\)](#).

Referenced by [setup_device\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.398.2.5 setup_device()

```
void L4virtio::Driver::Virtio_net_device::setup_device (
    L4::Cap< L4virtio::Device > srvcap,
    L4::Cap< L4Re::Dataspace > queue_ds = L4::Cap<L4Re::Dataspace>::Invalid) [inline]
```

Establish a connection to the device and set up shared memory.

Parameters

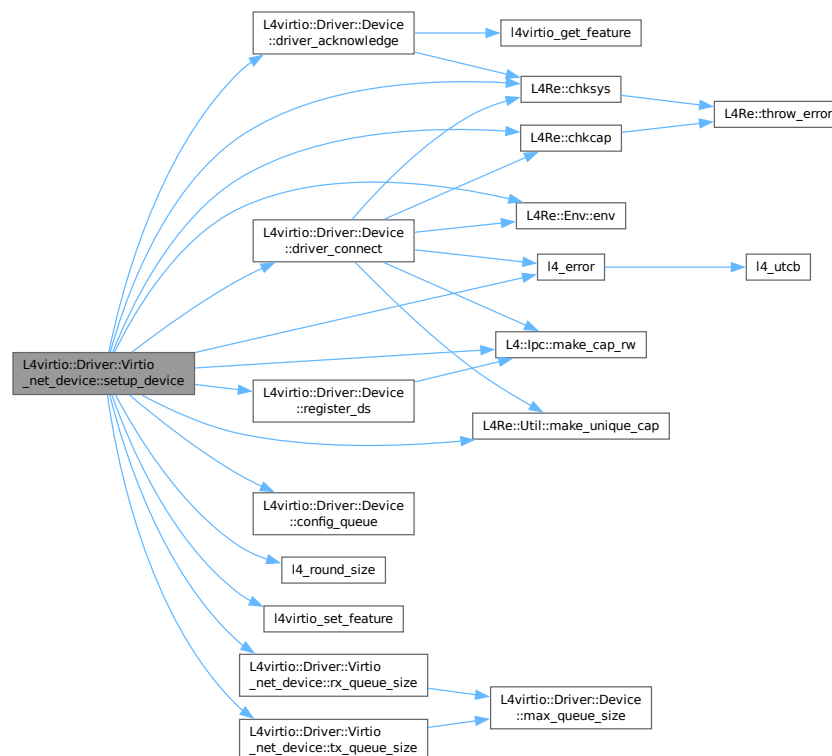
| | |
|-----------------|--|
| <i>srvcap</i> | IPC capability of the channel to the server. |
| <i>queue_ds</i> | (optional) External queue dataspace. If this capability is invalid, the function will attempt to allocate a dataspace on its own. Note that the external queue dataspace must be large enough. |

This function starts a handshake with the device and sets up the virtqueues for communication and the additional data structures for the network device.

Definition at line 70 of file [virtio-net](#).

References [L4Re::chkcap\(\)](#), [L4Re::chksys\(\)](#), [L4virtio::Driver::Device::config_queue\(\)](#), [L4Re::Mem_alloc::Continuous](#), [L4virtio::Driver::Device::driver_acknowledge\(\)](#), [L4virtio::Driver::Device::driver_connect\(\)](#), [L4Re::Env::env\(\)](#), [L4::Cap_base::Invalid](#), [L4_EINVAL](#), [L4_ENODEV](#), [I4_error\(\)](#), [L4_PAGESHIFT](#), [I4_round_size\(\)](#), [L4VIRTIO_FEATURE_VERSION_1](#), [L4VIRTIO_ID_NET](#), [I4virtio_set_feature\(\)](#), [L4::lpc::make_cap_rw\(\)](#), [L4Re::Util::make_unique_cap\(\)](#), [L4Re::Mem_alloc::Pinned](#), [L4virtio::Driver::Device::register_ds\(\)](#), [L4Re::Rm::F::RW](#), [rx_queue_size\(\)](#), [L4Re::Rm::F::Search_addr](#), and [tx_queue_size\(\)](#).

Here is the call graph for this function:



15.398.2.6 tx()

```
bool L4virtio::Driver::Virtio_net_device::tx (
    std::function< 14_uint32_t(Packet &)> prepare) [inline]
```

Attempt to allocate a descriptor in the TX queue and transmit the packet, after calling the prepare callback.

Parameters

| | |
|----------------|--|
| <i>prepare</i> | Function that fills the packet with data, should return the length of the data copied to the packet. |
|----------------|--|

Return values

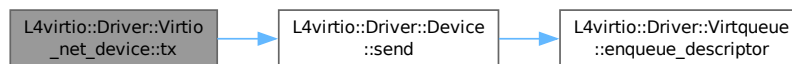
| | |
|--------------|------------------------|
| <i>true</i> | The packet was queued. |
| <i>false</i> | TX queue is full. |

The prepare callback should fill the packet with data and return the length of the packet data (without the size of the virtio-net packet header).

Definition at line 272 of file [virtio-net](#).

References [L4virtio::Driver::Device::send\(\)](#).

Here is the call graph for this function:

**15.398.2.7 tx_queue_size()**

```
int L4virtio::Driver::Virtio_net_device::tx_queue_size () const [inline]
```

Return the maximum transmit queue size allowed by the device.

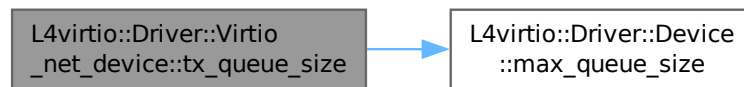
[tx\(\)](#) will fail if the amount of queued packets exceeds this size.

Definition at line 54 of file [virtio-net](#).

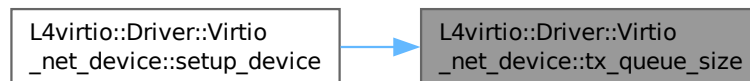
References [L4virtio::Driver::Device::max_queue_size\(\)](#).

Referenced by [setup_device\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.398.2.8 wait_rx()

```
l4_uint16_t L4virtio::Driver::Virtio_net_device::wait_rx (
    l4_uint32_t * len = nullptr) [inline]
```

Block until a network packet has been received from the device and return the descriptor number.

Precondition

The calling thread must be bound to the rx notification IRQ via `bind_rx_notification_irq()`.

Parameters

| | | |
|-----|-----|---|
| out | len | (optional) Length of valid data in RX packet. |
|-----|-----|---|

Returns

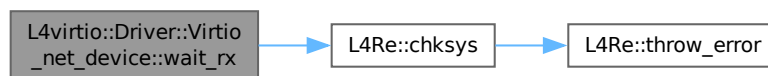
Descriptor number of received packet.

The packet data can be obtained with `rx_pkt()`. `finish_rx()` should be called after the packet buffer can be returned to the RX queue.

Definition at line 214 of file `virtio-net`.

References `L4Re::chksys()`.

Here is the call graph for this function:



The documentation for this class was generated from the following file:

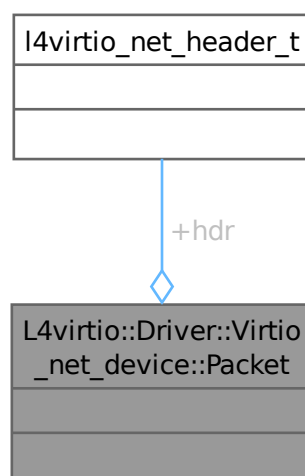
- I4/I4virtio/client/virtio-net

15.399 L4virtio::Driver::Virtio_net_device::Packet Struct Reference

Structure for a network packet (header including data) with maximum size, assuming that no extra features have been negotiated.

```
#include <virtio-net>
```

Collaboration diagram for L4virtio::Driver::Virtio_net_device::Packet:



15.399.1 Detailed Description

Structure for a network packet (header including data) with maximum size, assuming that no extra features have been negotiated.

Definition at line 37 of file [virtio-net](#).

The documentation for this struct was generated from the following file:

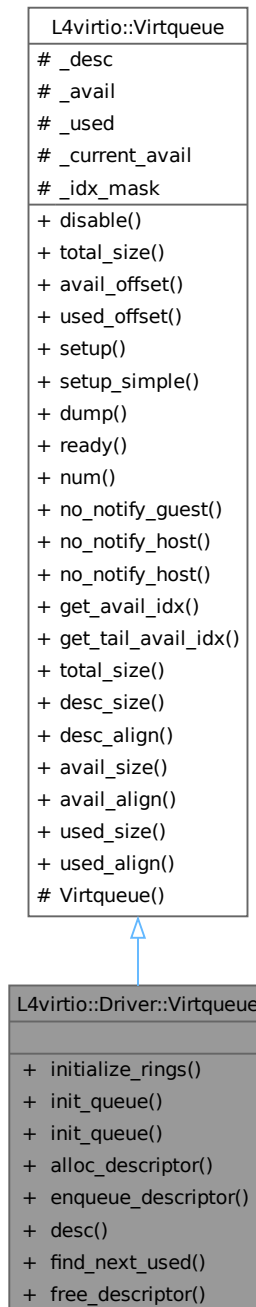
- I4/I4virtio/client/virtio-net

15.400 L4virtio::Driver::Virtqueue Class Reference

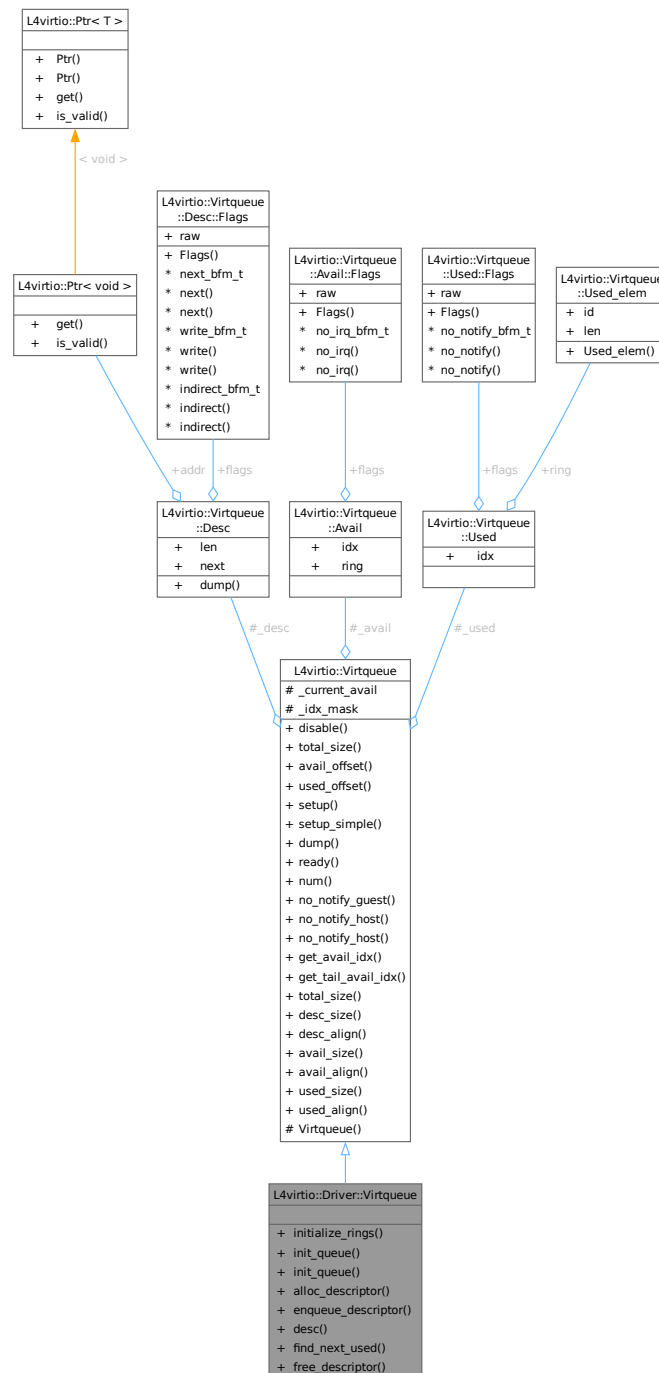
Driver-side implementation of a [Virtqueue](#).

```
#include <virtqueue>
```

Inheritance diagram for L4virtio::Driver::Virtqueue:



Collaboration diagram for L4virtio::Driver::Virtqueue:



Public Member Functions

- void [initialize_rings](#) (unsigned [num](#))
Initialize the descriptor table and the index structures of this queue.
- void [init_queue](#) (unsigned [num](#), void *[desc](#), void *[avail](#), void *[used](#))
Initialize this virtqueue.
- void [init_queue](#) (unsigned [num](#), void *[base](#))

- Initialize this virtqueue.*
- [l4_uint16_t alloc_descriptor](#) ()
Allocate and return an unused descriptor from the descriptor table.
- void [enqueue_descriptor](#) ([l4_uint16_t](#) descno)
Enqueue a descriptor in the available ring.
- [Desc & desc](#) ([l4_uint16_t](#) descno)
Return a reference to a descriptor in the descriptor table.
- [l4_uint16_t find_next_used](#) ([l4_uint32_t](#) *len=nullptr)
Return the next finished block.
- void [free_descriptor](#) ([l4_uint16_t](#) head, [l4_uint16_t](#) tail)
Free a chained list of descriptors in the descriptor queue.

Public Member Functions inherited from [L4virtio::Virtqueue](#)

- void [disable](#) ()
Completely disable the queue.
- unsigned long [total_size](#) () const
Calculate the total size of this virtqueue.
- unsigned long [avail_offset](#) () const
Get the offset of the available ring from the descriptor table.
- unsigned long [used_offset](#) () const
Get the offset of the used ring from the descriptor table.
- void [setup](#) (unsigned [num](#), void *desc, void *avail, void *used)
Enable this queue.
- void [setup_simple](#) (unsigned [num](#), void *ring)
Enable this queue.
- void [dump](#) ([Desc](#) const *d) const
Dump descriptors for this queue.
- bool [ready](#) () const
Test if this queue is in working state.
- unsigned [num](#) () const
- bool [no_notify_guest](#) () const
Get the no IRQ flag of this queue.
- bool [no_notify_host](#) () const
Get the no notify flag of this queue.
- void [no_notify_host](#) (bool value)
Set the no-notify flag for this queue.
- [l4_uint16_t](#) [get_avail_idx](#) () const
Get available index from available ring (for debugging).
- [l4_uint16_t](#) [get_tail_avail_idx](#) () const
Get tail-available index stored in local state (for debugging).

Additional Inherited Members

Public Types inherited from [L4virtio::Virtqueue](#)

- enum
Fixed alignment values for different parts of a virtqueue.

Static Public Member Functions inherited from [L4virtio::Virtqueue](#)

- static unsigned long [total_size](#) (unsigned [num](#))
Calculate the total size for a virtqueue of the given dimensions.
- static unsigned long [desc_size](#) (unsigned [num](#))
Calculate the size of the descriptor table for [num](#) entries.
- static unsigned long [desc_align](#) ()
Get the alignment in zero LSBs needed for the descriptor table.
- static unsigned long [avail_size](#) (unsigned [num](#))
Calculate the size of the available ring for [num](#) entries.
- static unsigned long [avail_align](#) ()
Get the alignment in zero LSBs needed for the available ring.
- static unsigned long [used_size](#) (unsigned [num](#))
Calculate the size of the used ring for [num](#) entries.
- static unsigned long [used_align](#) ()
Get the alignment in zero LSBs needed for the used ring.

Protected Member Functions inherited from [L4virtio::Virtqueue](#)

- [Virtqueue](#) ()=default
Create a disabled virtqueue.

Protected Attributes inherited from [L4virtio::Virtqueue](#)

- [Desc](#) * [_desc](#) = nullptr
pointer to descriptor table, NULL if queue is off.
- [Avail](#) * [_avail](#) = nullptr
pointer to available ring.
- [Used](#) * [_used](#) = nullptr
pointer to used ring.
- [l4_uint16_t](#) [_current_avail](#) = 0
The life counter for the queue.
- [l4_uint16_t](#) [_idx_mask](#) = 0
mask used for indexing into the descriptor table and the rings.

15.400.1 Detailed Description

Driver-side implementation of a [Virtqueue](#).

Adds function for managing the descriptor list, enqueueing new and dequeueing finished requests.

Note

The [Virtqueue](#) implementation is not thread-safe.

Definition at line 477 of file [virtqueue](#).

15.400.2 Member Function Documentation

15.400.2.1 alloc_descriptor()

```
l4_uint16_t L4virtio::Driver::Virtqueue::alloc_descriptor () [inline]
```

Allocate and return an unused descriptor from the descriptor table.

The descriptor will be removed from the free list, the content should be considered undefined. After use, it needs to be freed using [free_descriptor\(\)](#).

Returns

The index of the reserved descriptor or Virtqueue::Eoq if no free descriptor is available.

Note: the implementation uses $(2^{16} - 1)$ as the end of queue marker. That means that the final entry in the queue can not be allocated iff the queue size is 2^{16} .

Definition at line 565 of file [virtqueue](#).

References [L4virtio::Virtqueue::_desc](#).

15.400.2.2 desc()

```
Desc & L4virtio::Driver::Virtqueue::desc (
    l4_uint16_t descno) [inline]
```

Return a reference to a descriptor in the descriptor table.

Parameters

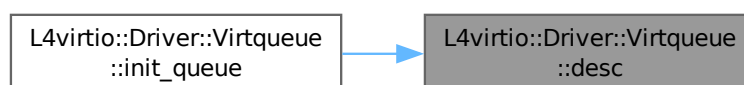
| | |
|---------------|---|
| <i>descno</i> | Index of the descriptor, expected to be in correct range. |
|---------------|---|

Definition at line 597 of file [virtqueue](#).

References [L4virtio::Virtqueue::_desc](#), and [L4virtio::Virtqueue::_idx_mask](#).

Referenced by [init_queue\(\)](#).

Here is the caller graph for this function:



15.400.2.3 enqueue_descriptor()

```
void L4virtio::Driver::Virtqueue::enqueue_descriptor (
    14_uint16_t descno) [inline]
```

Enqueue a descriptor in the available ring.

Parameters

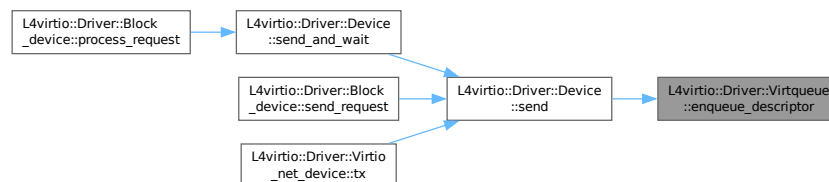
| | |
|---------------|--|
| <i>descno</i> | Index of the head descriptor to enqueue. |
|---------------|--|

Definition at line 581 of file [virtqueue](#).

References [L4virtio::Virtqueue::_avail](#), and [L4virtio::Virtqueue::_idx_mask](#).

Referenced by [L4virtio::Driver::Device::send\(\)](#).

Here is the caller graph for this function:



15.400.2.4 find_next_used()

```
14_uint16_t L4virtio::Driver::Virtqueue::find_next_used (
    14_uint32_t * len = nullptr) [inline]
```

Return the next finished block.

Parameters

| | | |
|------------|------------|--|
| <i>out</i> | <i>len</i> | (optional) Size of valid data in finished block. Note that this is the value reported by the device, which may set it to a value that is larger than the original buffer size. |
|------------|------------|--|

Returns

Index of the head or `Virtqueue::Eq` if no used element is currently available.

Definition at line 616 of file [virtqueue](#).

References [L4virtio::Virtqueue::_current_avail](#), [L4virtio::Virtqueue::_idx_mask](#), and [L4virtio::Virtqueue::_used](#).

Referenced by [L4virtio::Driver::Device::wait_for_next_used\(\)](#).

Here is the caller graph for this function:



15.400.2.5 free_descriptor()

```
void L4virtio::Driver::Virtqueue::free_descriptor (
    l4_uint16_t head,
    l4_uint16_t tail) [inline]
```

Free a chained list of descriptors in the descriptor queue.

Parameters

| | |
|-------------|---|
| <i>head</i> | Index of the first element in the descriptor chain. |
| <i>tail</i> | Index of the last element in the descriptor chain. |

Simply takes the descriptor chain and prepends it to the beginning of the free list. Assumes that the list has been correctly chained.

Definition at line 638 of file [virtqueue](#).

References [L4virtio::Virtqueue::_desc](#), and [L4virtio::Virtqueue::_idx_mask](#).

15.400.2.6 init_queue() [1/2]

```
void L4virtio::Driver::Virtqueue::init_queue (
    unsigned num,
    void * base) [inline]
```

Initialize this virtqueue.

Parameters

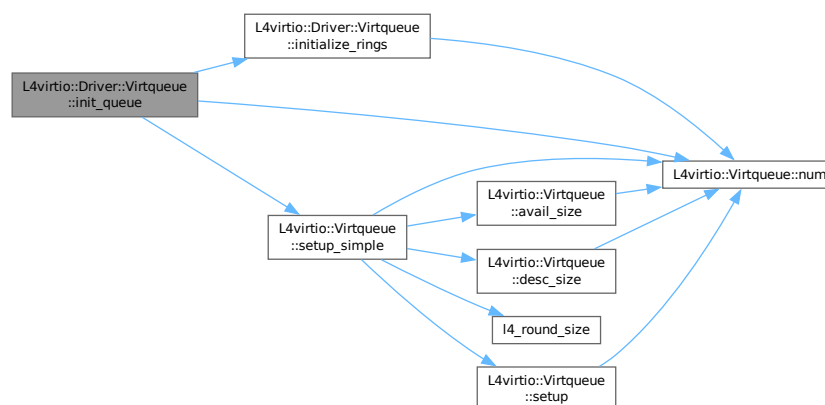
| | |
|-------------|--|
| <i>num</i> | The number of entries in the descriptor table, the available ring, and the used ring (must be a power of 2). |
| <i>base</i> | The base address for the queue data structure. |

This function sets up the memory and initializes the freelist.

Definition at line 544 of file [virtqueue](#).

References [initialize_rings\(\)](#), [L4virtio::Virtqueue::num\(\)](#), and [L4virtio::Virtqueue::setup_simple\(\)](#).

Here is the call graph for this function:



15.400.2.7 `init_queue()` [2/2]

```
void L4virtio::Driver::Virtqueue::init_queue (
    unsigned num,
    void * desc,
    void * avail,
    void * used) [inline]
```

Initialize this virtqueue.

Parameters

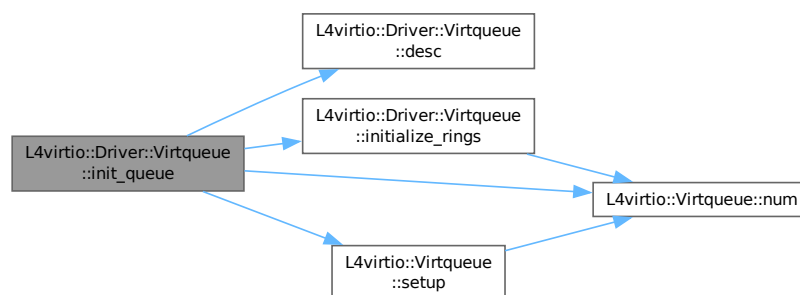
| | |
|--------------|--|
| <i>num</i> | The number of entries in the descriptor table, the available ring, and the used ring (must be a power of 2). |
| <i>desc</i> | The address of the descriptor table. (Must be <code>Desc_align</code> aligned and at least <code>desc_size(num)</code> bytes in size.) |
| <i>avail</i> | The address of the available ring. (Must be <code>Avail_align</code> aligned and at least <code>avail_size(num)</code> bytes in size.) |
| <i>used</i> | The address of the used ring. (Must be <code>Used_align</code> aligned and at least <code>used_size(num)</code> bytes in size.) |

This function sets up the memory and initializes the freelist.

Definition at line 529 of file [virtqueue](#).

References [desc\(\)](#), [initialize_rings\(\)](#), [L4virtio::Virtqueue::num\(\)](#), and [L4virtio::Virtqueue::setup\(\)](#).

Here is the call graph for this function:



15.400.2.8 `initialize_rings()`

```
void L4virtio::Driver::Virtqueue::initialize_rings (
    unsigned num) [inline]
```

Initialize the descriptor table and the index structures of this queue.

Parameters

| | |
|------------|--|
| <i>num</i> | The number of entries in the descriptor table, the available ring, and the used ring (must be a power of 2). |
|------------|--|

Precondition

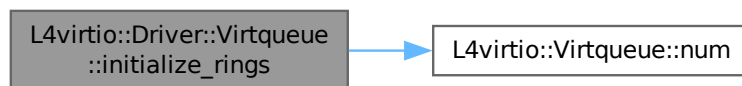
The queue must be set up correctly with [setup\(\)](#) or [setup_simple\(\)](#).

Definition at line 501 of file [virtqueue](#).

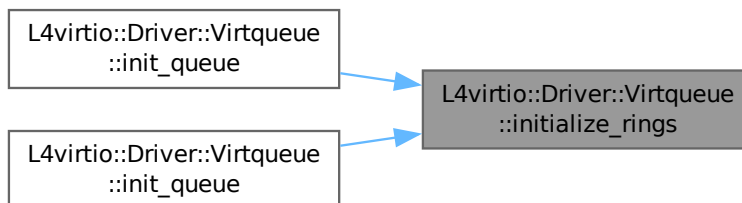
References [L4virtio::Virtqueue::_avail](#), [L4virtio::Virtqueue::_desc](#), [L4virtio::Virtqueue::_used](#), and [L4virtio::Virtqueue::num\(\)](#).

Referenced by [init_queue\(\)](#), and [init_queue\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

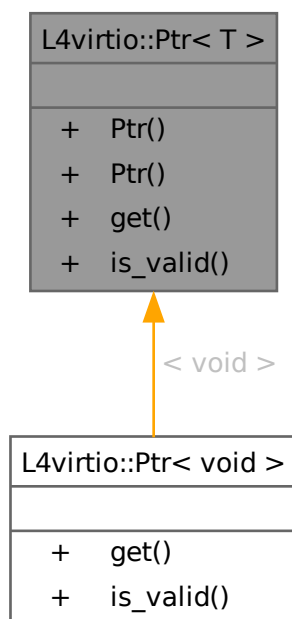
- `I4/I4virtio/virtqueue`

15.401 L4virtio::Ptr< T > Class Template Reference

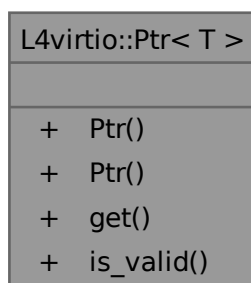
Pointer used in virtio descriptors.

```
#include <virtqueue>
```

Inheritance diagram for L4virtio::Ptr< T >:



Collaboration diagram for L4virtio::Ptr< T >:



Public Types

- enum [Invalid_type](#) { [Invalid](#) }
Type for making an invalid (NULL) [Ptr](#).

Public Member Functions

- [Ptr](#) ([Invalid_type](#))
Make and invalid [Ptr](#).
- [Ptr](#) ([l4_uint64_t](#) vm_addr)
Make a [Ptr](#) from a raw 64bit address.
- [l4_uint64_t](#) [get](#) () const
- bool [is_valid](#) () const

15.401.1 Detailed Description

```
template<typename T>
class L4virtio::Ptr< T >
```

Pointer used in virtio descriptors.

As the descriptor contain guest addresses these pointers cannot be dereferenced directly.

Definition at line 53 of file [virtqueue](#).

15.401.2 Member Enumeration Documentation

15.401.2.1 Invalid_type

```
template<typename T>
enum L4virtio::Ptr::Invalid_type
```

Type for making an invalid (NULL) [Ptr](#).

Enumerator

| | |
|---------|---|
| Invalid | Use to set a Ptr to invalid (NULL). |
|---------|---|

Definition at line 57 of file [virtqueue](#).

15.401.3 Member Function Documentation

15.401.3.1 get()

```
template<typename T>
l4_uint64_t L4virtio::Ptr< T >::get () const [inline]
```

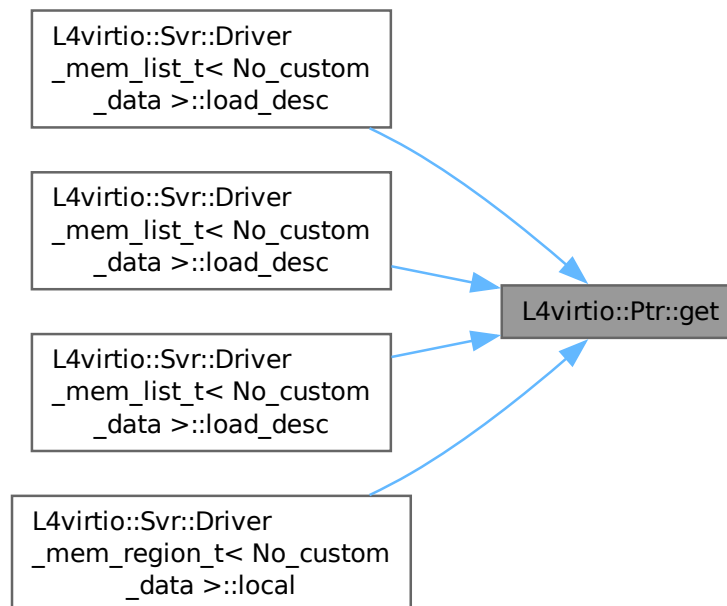
Returns

The raw 64bit address of the stored pointer.

Definition at line 68 of file [virtqueue](#).

Referenced by [L4virtio::Svr::Driver_mem_list_t< No_custom_data >::load_desc\(\)](#), [L4virtio::Svr::Driver_mem_list_t< No_custom_data >::load_desc\(\)](#), [L4virtio::Svr::Driver_mem_list_t< No_custom_data >::load_desc\(\)](#), and [L4virtio::Svr::Driver_mem_region_t< No_custom_data >::local](#).

Here is the caller graph for this function:



15.401.3.2 is_valid()

```
template<typename T>
bool L4virtio::Ptr< T >::is_valid () const [inline]
```

Returns

true if the stored pointer is valid (not NULL).

Definition at line 71 of file [virtqueue](#).

The documentation for this class was generated from the following file:

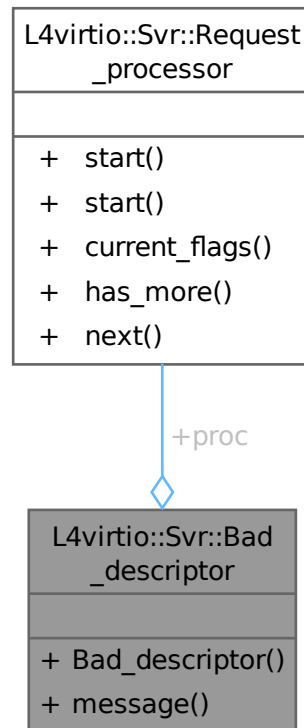
- `l4/l4virtio/virtqueue`

15.402 L4virtio::Svr::Bad_descriptor Struct Reference

Exception used by Queue to indicate descriptor errors.

```
#include <virtio>
```

Collaboration diagram for L4virtio::Svr::Bad_descriptor:



Public Types

- enum [Error](#) {
[Bad_address](#) , [Bad_rights](#) , [Bad_flags](#) , [Bad_next](#) ,
[Bad_size](#) }

The error code.

Public Member Functions

- [Bad_descriptor](#) ([Request_processor](#) const *[proc](#), [Error](#) e)
Make a bad descriptor exception.
- char const * [message](#) () const
Get a human readable description of the error code.

Data Fields

- [Request_processor](#) const * **proc**
The processor that triggered the exception.

15.402.1 Detailed Description

Exception used by Queue to indicate descriptor errors.

Definition at line [397](#) of file [virtio](#).

15.402.2 Member Enumeration Documentation

15.402.2.1 Error

```
enum L4virtio::Svr::Bad_descriptor::Error
```

The error code.

Enumerator

| | |
|-------------|--|
| Bad_address | Address cannot be translated. |
| Bad_rights | Missing access rights on memory. |
| Bad_flags | Invalid combination of descriptor flags. |
| Bad_next | Invalid next index. |
| Bad_size | Invalid size of memory block. |

Definition at line [400](#) of file [virtio](#).

15.402.3 Constructor & Destructor Documentation

15.402.3.1 Bad_descriptor()

```
L4virtio::Svr::Bad_descriptor::Bad_descriptor (  
    Request\_processor const * proc,  
    Error e) [inline]
```

Make a bad descriptor exception.

Parameters

| | |
|-------------|---|
| <i>proc</i> | The request processor causing the exception |
| <i>e</i> | The error code. |

Definition at line [421](#) of file [virtio](#).

References [proc](#).

15.402.4 Member Function Documentation

15.402.4.1 message()

```
char const * L4virtio::Svr::Bad_descriptor::message () const [inline]
```

Get a human readable description of the error code.

Returns

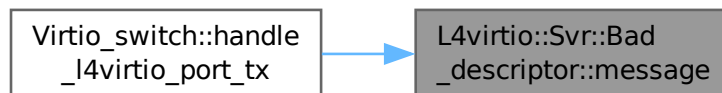
Message describing the error.

Definition at line 430 of file [virtio](#).

References [Bad_address](#), [Bad_flags](#), [Bad_next](#), [Bad_rights](#), and [Bad_size](#).

Referenced by [Virtio_switch::handle_l4virtio_port_tx\(\)](#).

Here is the caller graph for this function:



The documentation for this struct was generated from the following file:

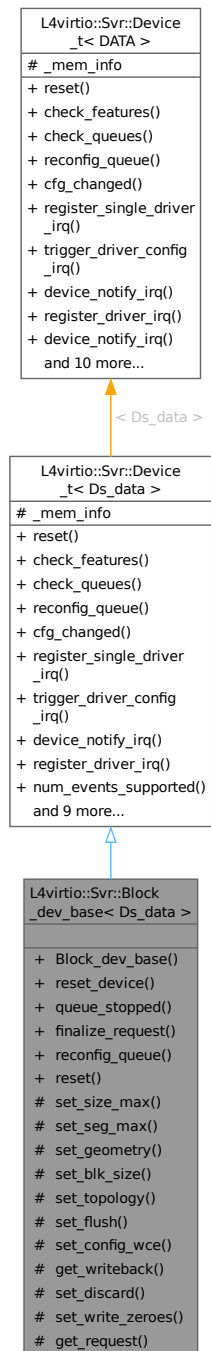
- I4/I4virtio/server/virtio

15.403 L4virtio::Svr::Block_dev_base< Ds_data > Class Template Reference

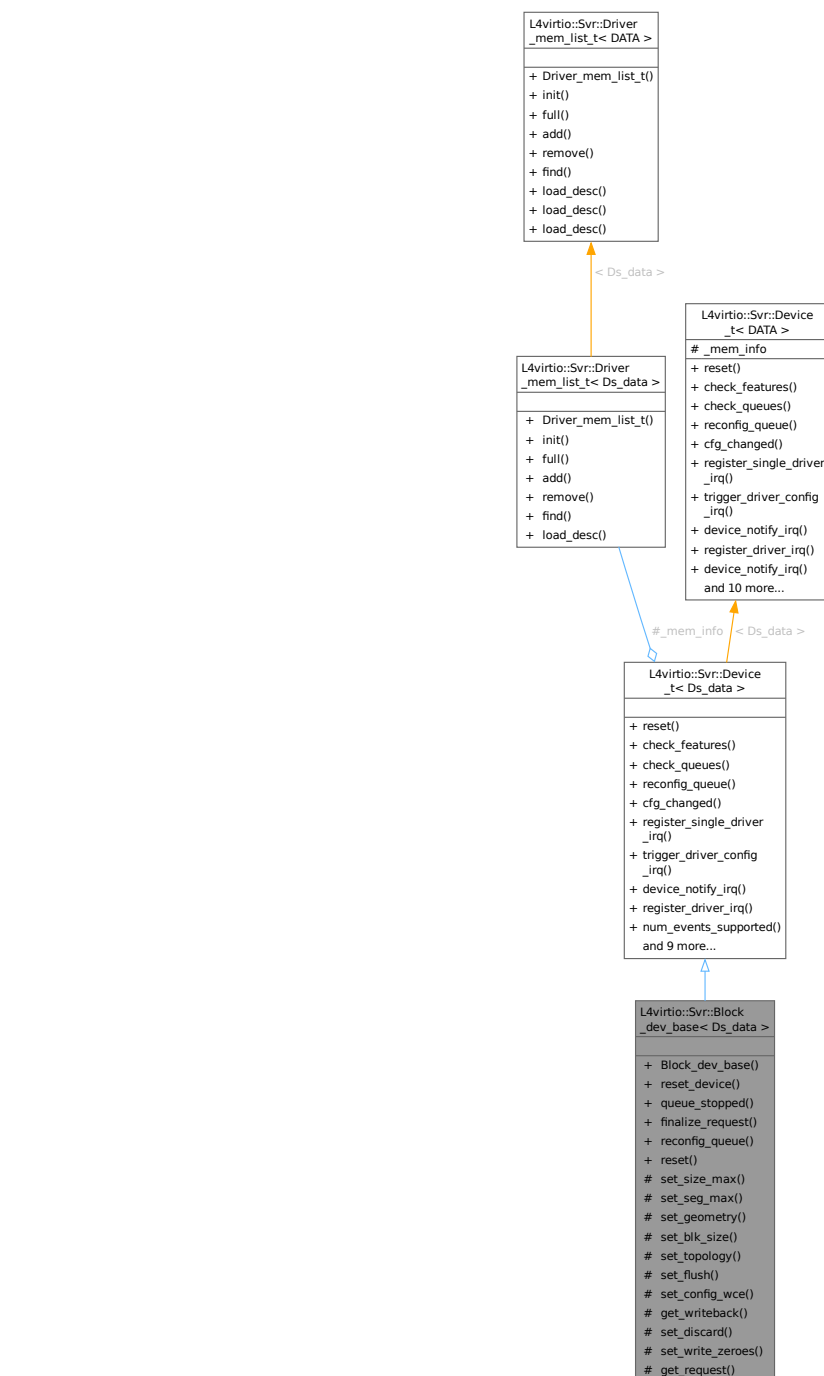
Base class for virtio block devices.

```
#include <virtio-block>
```

Inheritance diagram for L4virtio::Svr::Block_dev_base< Ds_data >:



Collaboration diagram for L4virtio::Svr::Block_dev_base< Ds_data >:



Public Member Functions

- **Block_dev_base** (`l4_uint32_t` vendor, unsigned queue_size, `l4_uint64_t` capacity, bool read_only)
Create a new virtio block device.
- virtual void **reset_device** ()=0
Reset the actual hardware device.
- virtual bool **queue_stopped** ()=0

- *Return true, if the queues should not be processed further.*
- void **finalize_request** (cxx::unique_ptr< Request > req, unsigned sz, l4_uint8_t status=L4VIRTIO_BLOCK_S_OK)
Releases resources related to a request and notifies the client.
- int **reconfig_queue** (unsigned idx) override
callback for client queue-config request
- void **reset** () override
reset callback, called for doing a device reset

Public Member Functions inherited from L4virtio::Svr::Device_t< Ds_data >

- virtual bool **check_features** ()
callback for checking the subset of accepted features
- virtual void **cfg_changed** (unsigned)
callback for client device configuration changes
- virtual L4::Cap< L4::Irq > **device_notify_irq** () const
callback to gather the device notification IRQ (old-style)
- virtual void **register_driver_irq** (unsigned idx)
Callback for registering an notification IRQ (multi IRQ).
- virtual unsigned **num_events_supported** () const
Return the highest notification index supported.
- **Device_t** (Dev_config *dev_config)
Make a device for the given config.
- Mem_list const * **mem_info** () const
Get the memory region list used for this device.
- void **reset_queue_config** (unsigned idx, unsigned num_max, bool inc_generation=false, unsigned device_↔ notify_index=0)
Trigger reset for the configuration space for queue idx.
- void **init_mem_info** (unsigned num)
Initialize the memory region list to the given maximum.
- void **device_error** ()
Transition device into DEVICE_NEEDS_RESET state.
- bool **setup_queue** (Virtqueue *q, unsigned qn, unsigned num_max)
Enable/disable the specified queue.
- bool **handle_mem_cmd_write** ()
Check for a value in the cmd register and handle a write.
- void **enable_trusted_ds_validation** ()
Enable trusted dataspace validation.
- void **add_trusted_dataspaces** (std::shared_ptr< Ds_vector const > ds)
Provide a list of trusted dataspace that can be used for validation.

Protected Member Functions

- void **set_size_max** (l4_uint32_t sz)
Sets the maximum size of any single segment reported to client.
- void **set_seg_max** (l4_uint32_t sz)
Sets the maximum number of segments in a request that is reported to client.
- void **set_geometry** (l4_uint16_t cylinders, l4_uint8_t heads, l4_uint8_t sectors)
Set disk geometry that is reported to the client.
- void **set_blk_size** (l4_uint32_t sz)

- Sets block disk size to be reported to the client.*
- void [set_topology](#) ([l4_uint8_t](#) physical_block_exp, [l4_uint8_t](#) alignment_offset, [l4_uint32_t](#) min_io_size, [l4_uint32_t](#) opt_io_size)
- Sets the I/O alignment information reported back to the client.*
- void **set_flush** ()
- Enables the flush command.*
- void [set_config_wce](#) ([l4_uint8_t](#) writeback)
- Sets cache mode and enables the writeback toggle.*
- [l4_uint8_t](#) [get_writeback](#) ()
- Get the writeback field from the configuration space.*
- void [set_discard](#) ([l4_uint32_t](#) max_discard_sectors, [l4_uint32_t](#) max_discard_seg, [l4_uint32_t](#) discard_↵ sector_alignment)
- Sets constraints for and enables the discard command.*
- void [set_write_zeroes](#) ([l4_uint32_t](#) max_write_zeroes_sectors, [l4_uint32_t](#) max_write_zeroes_seg, [l4_uint8_t](#) write_zeroes_may_unmap)
- Sets constraints for and enables the write zeroes command.*
- cxx::unique_ptr< Request > **get_request** ()
- Return one request if available.*

Additional Inherited Members

Protected Attributes inherited from [L4virtio::Svr::Device_t< Ds_data >](#)

- Mem_list **_mem_info**
- Memory region list.*

15.403.1 Detailed Description

```
template<typename Ds_data>
class L4virtio::Svr::Block_dev_base< Ds_data >
```

Base class for virtio block devices.

Use this class as a base to implement your own specific block device.

Definition at line 259 of file [virtio-block](#).

15.403.2 Constructor & Destructor Documentation

15.403.2.1 Block_dev_base()

```
template<typename Ds_data>
L4virtio::Svr::Block_dev_base< Ds_data >::Block_dev_base (
    l4\_uint32\_t vendor,
    unsigned queue\_size,
    l4\_uint64\_t capacity,
    bool read\_only) [inline]
```

Create a new virtio block device.

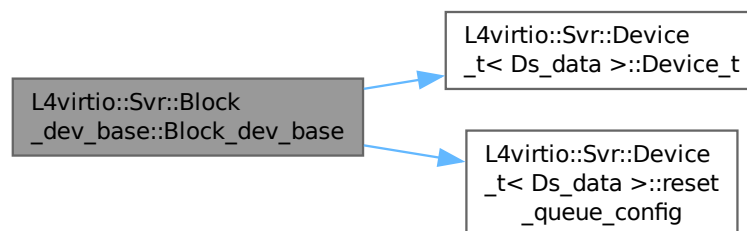
Parameters

| | |
|-------------------|---|
| <i>vendor</i> | Vendor ID |
| <i>queue_size</i> | Number of entries to provide in avail and used queue. |
| <i>capacity</i> | Size of the device in 512-byte sectors. |
| <i>read_only</i> | True, if the device should not be writable. |

Definition at line 444 of file [virtio-block](#).

References [L4virtio::Svr::Device_t<Ds_data>::Device_t\(\)](#), [L4VIRTIO_FEATURE_VERSION_1](#), [L4VIRTIO_ID_BLOCK](#), and [L4virtio::Svr::Device_t<Ds_data>::reset_queue_config\(\)](#).

Here is the call graph for this function:

**15.403.3 Member Function Documentation****15.403.3.1 finalize_request()**

```

template<typename Ds_data>
void L4virtio::Svr::Block_dev_base<Ds_data>::finalize_request (
    cxx::unique_ptr< Request > req,
    unsigned sz,
    l4_uint8_t status = L4VIRTIO_BLOCK_S_OK) [inline]
  
```

Releases resources related to a request and notifies the client.

Parameters

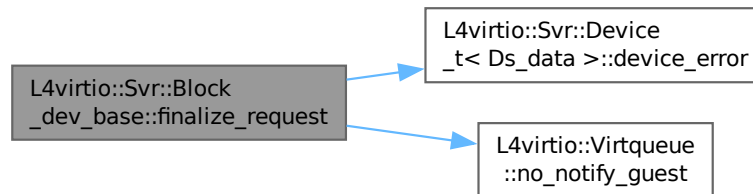
| | |
|---------------|---|
| <i>req</i> | Pointer to request that has finished. |
| <i>sz</i> | Number of bytes consumed. |
| <i>status</i> | Status of request (see L4virtio_block_status). |

This function must be called when an asynchronous request finishes, either successfully or with an error. The status byte in the request must have been set prior to calling it.

Definition at line 483 of file [virtio-block](#).

References [L4virtio::Svr::Device_t< Ds_data >::device_error\(\)](#), [L4VIRTIO_BLOCK_S_OK](#), [L4VIRTIO_IRQ_STATUS_VRING](#), and [L4virtio::Virtqueue::no_notify_guest\(\)](#).

Here is the call graph for this function:



15.403.3.2 `get_writeback()`

```
template<typename Ds_data>
l4_uint8_t L4virtio::Svr::Block_dev_base< Ds_data >::get_writeback () [inline], [protected]
```

Get the writeback field from the configuration space.

Returns

Value of the writeback field.

Definition at line 388 of file [virtio-block](#).

15.403.3.3 `set_blk_size()`

```
template<typename Ds_data>
void L4virtio::Svr::Block_dev_base< Ds_data >::set_blk_size (
    l4_uint32_t sz) [inline], [protected]
```

Sets block disk size to be reported to the client.

Setting this does not change the logical sector size used for addressing the device.

Definition at line 332 of file [virtio-block](#).

15.403.3.4 set_config_wce()

```
template<typename Ds_data>
void L4virtio::Svr::Block_dev_base< Ds_data >::set_config_wce (
    14_uint8_t writeback) [inline], [protected]
```

Sets cache mode and enables the writeback toggle.

Parameters

| | |
|------------------|--|
| <i>writeback</i> | Mode of the cache (0 for writethrough, 1 for writeback). |
|------------------|--|

Definition at line 375 of file [virtio-block](#).

15.403.3.5 set_discard()

```
template<typename Ds_data>
void L4virtio::Svr::Block_dev_base< Ds_data >::set_discard (
    14_uint32_t max_discard_sectors,
    14_uint32_t max_discard_seg,
    14_uint32_t discard_sector_alignment) [inline], [protected]
```

Sets constraints for and enables the discard command.

Parameters

| | |
|---------------------------------|--|
| <i>max_discard_sectors</i> | Maximum discard sectors size. |
| <i>max_discard_seg</i> | Maximum discard segment number. |
| <i>discard_sector_alignment</i> | Can be used by the driver when splitting a request based on alignment. |

Definition at line 402 of file [virtio-block](#).

15.403.3.6 set_size_max()

```
template<typename Ds_data>
void L4virtio::Svr::Block_dev_base< Ds_data >::set_size_max (
    14_uint32_t sz) [inline], [protected]
```

Sets the maximum size of any single segment reported to client.

The limit is also applied to any incoming requests. Requests with larger segments result in an IO error being reported to the client. That means that `process_request()` can safely make the assumption that all segments in the received request are smaller.

Definition at line 290 of file [virtio-block](#).

15.403.3.7 set_topology()

```
template<typename Ds_data>
void L4virtio::Svr::Block_dev_base< Ds_data >::set_topology (
    l4_uint8_t physical_block_exp,
    l4_uint8_t alignment_offset,
    l4_uint32_t min_io_size,
    l4_uint32_t opt_io_size) [inline], [protected]
```

Sets the I/O alignment information reported back to the client.

Parameters

| | |
|---------------------------|---|
| <i>physical_block_exp</i> | Number of logical blocks per physical block(log2) |
| <i>alignment_offset</i> | Offset of the first aligned logical block |
| <i>min_io_size</i> | Suggested minimum I/O size in blocks |
| <i>opt_io_size</i> | Optimal I/O size in blocks |

Definition at line 348 of file [virtio-block](#).

15.403.3.8 set_write_zeroes()

```
template<typename Ds_data>
void L4virtio::Svr::Block_dev_base< Ds_data >::set_write_zeroes (
    l4_uint32_t max_write_zeroes_sectors,
    l4_uint32_t max_write_zeroes_seg,
    l4_uint8_t write_zeroes_may_unmap) [inline], [protected]
```

Sets constraints for and enables the write zeroes command.

Parameters

| | |
|---------------------------------|---|
| <i>max_write_zeroes_sectors</i> | Maximum write zeroes sectors size. |
| <i>max_write_zeroes_seg</i> | maximum write zeroes segment number. |
| <i>write_zeroes_may_unmap</i> | Set if a write zeroes request can result in deallocating one or more sectors. |

Definition at line 422 of file [virtio-block](#).

The documentation for this class was generated from the following file:

- I4/I4virtio/server/virtio-block

15.404 L4virtio::Svr::Block_request< Ds_data > Class Template Reference

A request to read or write data.

```
#include <virtio-block>
```

Collaboration diagram for L4virtio::Svr::Block_request< Ds_data >:

| L4virtio::Svr::Block_request< Ds_data > |
|---|
| <ul style="list-style-type: none"> + data_size() + has_more() + next_block() + header() |

Public Member Functions

- unsigned [data_size](#) () const
Compute the total size of the data in the request.
- bool **has_more** ()
Check if the request contains more data blocks.
- Data_block [next_block](#) ()
Return next block in scatter-gather list.
- [l4virtio_block_header_t](#) const & **header** () const
Return the block request header.

15.404.1 Detailed Description

```
template<typename Ds_data>
class L4virtio::Svr::Block_request< Ds_data >
```

A request to read or write data.

Definition at line 28 of file [virtio-block](#).

15.404.2 Member Function Documentation

15.404.2.1 data_size()

```
template<typename Ds_data>
unsigned L4virtio::Svr::Block_request< Ds_data >::data_size () const [inline]
```

Compute the total size of the data in the request.

Return values

| | |
|------|--------------------------------------|
| Size | in bytes or 0 if there was an error. |
|------|--------------------------------------|

Exceptions

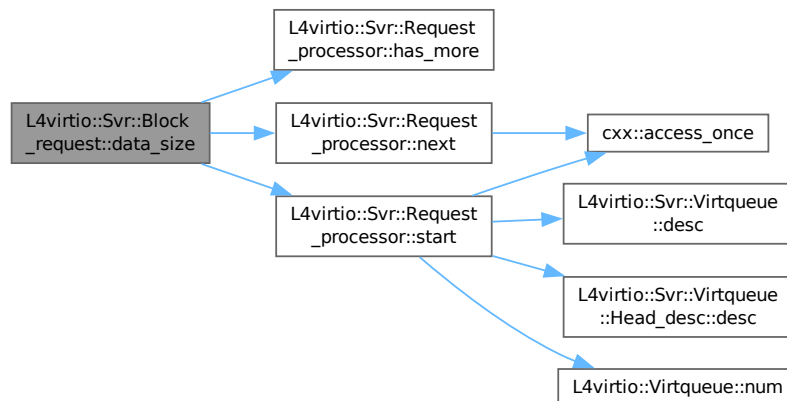
| | |
|--|---------------------------|
| L4::Runtime_error(-L4_EIO) | Request has a bad format. |
|--|---------------------------|

Note that this operation is relatively expensive as it has to iterate over the complete list of blocks.

Definition at line 63 of file [virtio-block](#).

References [L4virtio::Svr::Request_processor::has_more\(\)](#), [L4_EIO](#), [L4virtio::Svr::Request_processor::next\(\)](#), and [L4virtio::Svr::Request_processor::start\(\)](#).

Here is the call graph for this function:



15.404.2.2 next_block()

```
template<typename Ds_data>
Data_block L4virtio::Svr::Block_request< Ds_data >::next_block () [inline]
```

Return next block in scatter-gather list.

Returns

Information about the next data block.

Exceptions

| | |
|-----------------------------------|----------------------------------|
| L4::Runtime_error | No more data block is available. |
| Bad_descriptor | Virtio request is corrupted. |

Definition at line 113 of file [virtio-block](#).

References [L4virtio::Svr::Bad_descriptor::Bad_size](#), and [L4_EEXIST](#).

The documentation for this class was generated from the following file:

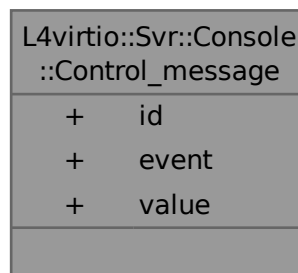
- [l4/l4virtio/server/virtio-block](#)

15.405 L4virtio::Svr::Console::Control_message Struct Reference

Virtio console control message.

```
#include <virtio-console>
```

Collaboration diagram for L4virtio::Svr::Console::Control_message:

**Public Types**

- enum [Events](#) {
[Device_ready](#) = 0 , [Device_add](#) = 1 , [Device_remove](#) = 2 , [Port_ready](#) = 3 ,
[Console_port](#) = 4 , [Resize](#) = 5 , [Port_open](#) = 6 , [Port_name](#) = 7 }

Possible control events.

Data Fields

- [l4_uint32_t id](#)
Port number.
- [l4_uint16_t event](#)
Control event, see [Events](#).
- [l4_uint16_t value](#)
Extra information.

15.405.1 Detailed Description

Virtio console control message.

Definition at line 31 of file [virtio-console](#).

15.405.2 Member Enumeration Documentation

15.405.2.1 Events

enum [L4virtio::Svr::Console::Control_message::Events](#)

Possible control events.

Enumerator

| | |
|---------------|---|
| Device_ready | Sent by driver at initialization. |
| Device_add | Sent by device to create new ports. |
| Device_remove | Sent by device to remove added ports. |
| Port_ready | Sent by driver as response to Device_add . |
| Console_port | Sent by device to nominate port as console port. |
| Resize | Sent by device to indicate a console size change. |
| Port_open | Sent by device and driver to indicate whether a port is open. |
| Port_name | Sent by device to tag a port. |

Definition at line 34 of file [virtio-console](#).

The documentation for this struct was generated from the following file:

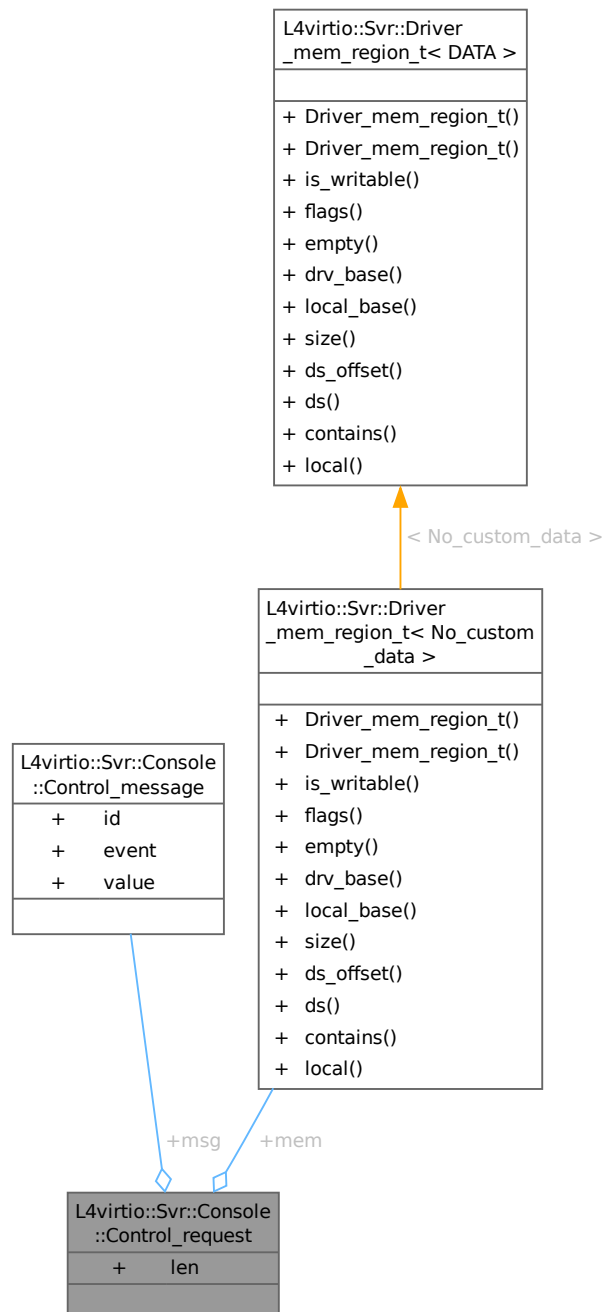
- [l4/l4virtio/server/virtio-console](#)

15.406 L4virtio::Svr::Console::Control_request Struct Reference

Specialised `Virtqueue::Request` providing access to control message payload.

```
#include <virtio-console>
```

Collaboration diagram for `L4virtio::Svr::Console::Control_request`:



Data Fields

- [Control_message](#) * **msg**
Virtual address of the data block (in device space).
- [l4_uint32_t](#) **len**
Length of datablock in bytes.
- [Driver_mem_region](#) * **mem**
Pointer to driver memory region.

15.406.1 Detailed Description

Specialised `Virtqueue::Request` providing access to control message payload.

Definition at line 65 of file [virtio-console](#).

The documentation for this struct was generated from the following file:

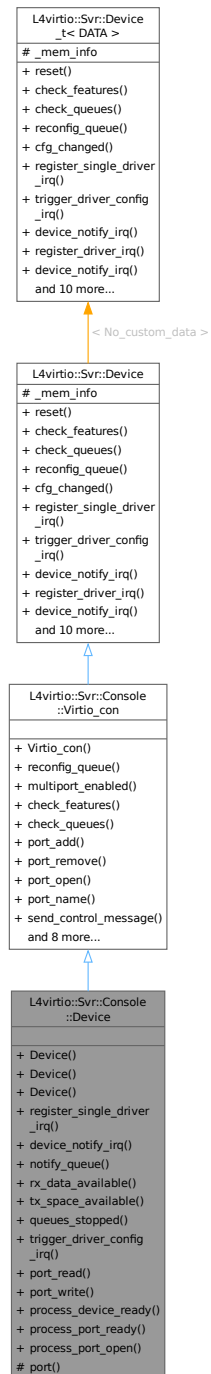
- `l4/l4virtio/server/virtio-console`

15.407 L4virtio::Svr::Console::Device Class Reference

Base class implementing a virtio console device with L4Re-based notification handling.

```
#include <virtio-console-device>
```

Inheritance diagram for L4virtio::Svr::Console::Device:





- Generated for L4Re by Doxygen

- Create a new console [Device](#).
- void **register_single_driver_irq** () override
callback for registering a single guest IRQ for all queues (old-style)
- [L4::Cap](#)< [L4::Irq](#) > **device_notify_irq** () const override
callback to gather the device notification IRQ (old-style)
- void **notify_queue** (Virtqueue *queue) override
Notify queue of available data.
- virtual void **rx_data_available** (unsigned [port](#))=0
Callback to notify that new data is available to be read from port.
- virtual void **tx_space_available** (unsigned [port](#))=0
Callback to notify that data can be written to port.
- virtual bool **queues_stopped** ()
Return true, if the queues should not be processed further.
- void **trigger_driver_config_irq** () override
callback for triggering configuration change notification IRQ
- unsigned [port_read](#) (char *buf, unsigned len, unsigned [port](#)=0)
Read data from port.
- unsigned [port_write](#) (char const *buf, unsigned len, unsigned [port](#)=0)
Write data to port.
- void [process_device_ready](#) ([l4_uint16_t](#) value) override
Callback called on DEVICE_READY event.
- void [process_port_ready](#) ([l4_uint32_t](#) id, [l4_uint16_t](#) value) override
Callback called on PORT_READY event.
- void [process_port_open](#) ([l4_uint32_t](#) id, [l4_uint16_t](#) value) override
Callback called on PORT_OPEN event.

Public Member Functions inherited from [L4virtio::Svr::Console::Virtio_con](#)

- [Virtio_con](#) (unsigned max_ports, bool enable_multiport)
Create a new multiport console device.
- int **reconfig_queue** (unsigned index) override
callback for client queue-config request
- bool **multiport_enabled** () const
Return true if the multiport feature is enabled and control queues are available.
- bool **check_features** (void) override
callback for checking the subset of accepted features
- bool **check_queues** () override
callback for checking if the queues at DRIVER_OK transition
- int [port_add](#) (unsigned idx)
Send a DEVICE_ADD message and update the internal state.
- int [port_remove](#) (unsigned idx)
Send a DEVICE_REMOVE message and update the internal state.
- int [port_open](#) (unsigned idx, bool open)
Send a PORT_OPEN message and update the internal state.
- int [port_name](#) (unsigned idx, char const *name)
Send a PORT_NAME message to announce the port name.
- int [send_control_message](#) ([l4_uint32_t](#) idx, [l4_uint16_t](#) event, [l4_uint16_t](#) value=0, const char *name=0)
Send control message to driver.
- int [handle_control_message](#) ()
Handle control message received from the driver.
- void **reset** () override
reset callback, called for doing a device reset
- virtual void [reset_device](#) ()
Reset the state of the actual console device.

Public Member Functions inherited from L4virtio::Svr::Device_t< No_custom_data >

- virtual void **cfg_changed** (unsigned)
callback for client device configuration changes
- virtual void **register_driver_irq** (unsigned idx)
Callback for registering an notification IRQ (multi IRQ).
- virtual L4::Cap< L4::Irq > **device_notify_irq** (unsigned idx)
Callback to gather the device notification IRQ (multi IRQ).
- virtual unsigned **num_events_supported** () const
Return the highest notification index supported.
- **Device_t** (Dev_config *dev_config)
Make a device for the given config.
- Mem_list const * **mem_info** () const
Get the memory region list used for this device.
- void **reset_queue_config** (unsigned idx, unsigned num_max, bool inc_generation=false, unsigned device_notify_index=0)
Trigger reset for the configuration space for queue idx.
- void **init_mem_info** (unsigned num)
Initialize the memory region list to the given maximum.
- void **device_error** ()
Transition device into DEVICE_NEEDS_RESET state.
- bool **setup_queue** (Virtqueue *q, unsigned qn, unsigned num_max)
Enable/disable the specified queue.
- bool **handle_mem_cmd_write** ()
Check for a value in the cmd register and handle a write.
- void **enable_trusted_ds_validation** ()
Enable trusted dataspace validation.
- void **add_trusted_dataspaces** (std::shared_ptr< Ds_vector const > ds)
Provide a list of trusted dataspaces that can be used for validation.

Protected Member Functions

- Port * **port** (unsigned idx) override
Return the specified port.

Additional Inherited Members

Protected Attributes inherited from L4virtio::Svr::Device_t< No_custom_data >

- Mem_list **_mem_info**
Memory region list.

15.407.1 Detailed Description

Base class implementing a virtio console device with L4Re-based notification handling.

This console device is derived from [Virtio_con](#) and already includes functionality to handle interrupts and notify drivers. If an interrupt is received, all the necessary interaction with the virtqueues is performed and only the actual data processing has to be done by the derived class. By default all available ports are added and an "open"-request of a port by the driver is automatically acknowledged. The derived class can optionally change this behaviour by overriding [process_device_ready\(\)](#), [process_port_ready\(\)](#) and [process_port_open\(\)](#).

This class provides a stream-based interface to access the port data with edge-triggered notification callbacks. If a port receives data from the driver the derived class is notified with the [rx_data_available\(\)](#) callback. The actual data can be retrieved by [port_read\(\)](#). If there was not enough data to be read, the call will return the available partial data. Only then will the [rx_data_available\(\)](#) callback be triggered again.

Data on a port may be transmitted by [port_write\(\)](#). If there were not enough buffers available, only a part of the data will be transmitted. Once there are new buffers available, the [tx_space_available\(\)](#) callback will be invoked. This callback will be called again only after a previous [port_write\(\)](#) was not able to send all requested data.

Use this class as a base to provide your own high-level console device. You must derive from this class as well as [L4::Epiface_t<..., L4virtio::Device>](#). For a working device the [irq_iface\(\)](#) must be registered too. A typical implementation might look like the following:

```
class My_console
: public L4virtio::Svr::Console::Device,
  public L4::Epiface_t<My_console, L4virtio::Device>
{
public:
    My_console(L4Re::Util::Object_registry *r)
    : L4virtio::Svr::Console::Device(0x100)
    {
        init_mem_info(4);
        L4Re::chkcap(r->register_irq_obj(irq_iface()), "virtio notification IRQ");
    }

    void rx_data_available(unsigned port) override
    {
        // call port_read() to fetch available data
    }

    void tx_space_available(unsigned port) override
    {
        // can call port_write() to send (pending) data
    }
};

My_console console(registry);
registry->register_obj(&console, ...);
```

The maximum number of memory regions ([init_mem_info\(\)](#)) should correlate with the number of supported ports.

Definition at line 118 of file [virtio-console-device](#).

15.407.2 Constructor & Destructor Documentation

15.407.2.1 Device() [1/3]

```
L4virtio::Svr::Console::Device::Device (
    unsigned vq_max) [inline], [explicit]
```

Create a new console device.

Parameters

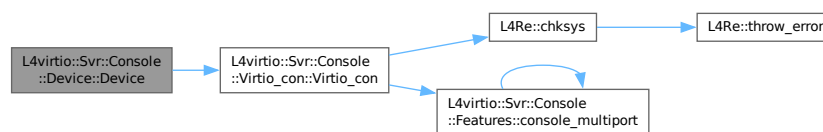
| | |
|---------------|---|
| <i>vq_max</i> | Maximum number of buffers in data queues. |
|---------------|---|

Create a console device with no multiport support, i.e. control queues are disabled.

Definition at line 145 of file [virtio-console-device](#).

References [L4virtio::Svr::Console::Virtio_con::Virtio_con\(\)](#).

Here is the call graph for this function:

**15.407.2.2 Device() [2/3]**

```

L4virtio::Svr::Console::Device::Device (
    unsigned vq_max,
    unsigned ports) [inline], [explicit]

```

Create a new console device.

Parameters

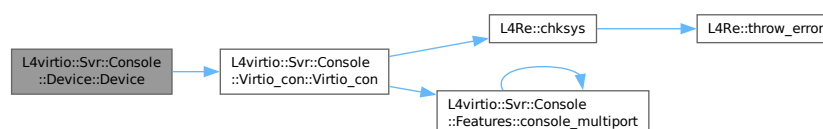
| | |
|---------------|---|
| <i>vq_max</i> | Maximum number of buffers in data queues. |
| <i>ports</i> | Number of ports (maximum 32). |

Create a console device with multiport support, i.e. control queues are enabled.

Definition at line 163 of file [virtio-console-device](#).

References [L4virtio::Svr::Console::Virtio_con::Virtio_con\(\)](#).

Here is the call graph for this function:



15.407.2.3 Device() [3/3]

```
L4virtio::Svr::Console::Device::Device (
    cxx::static_vector< unsigned > const & vq_max_nums) [inline], [explicit]
```

Create a new console [Device](#).

Parameters

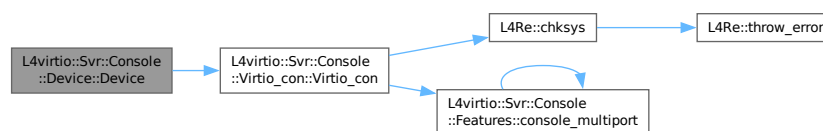
| | |
|--------------------|--|
| <i>vq_max_nums</i> | Maximum number of buffers in data queues, given as a cxx::static_vector with one entry per port. |
|--------------------|--|

Create a console device with multiport support, i.e. control queues are enabled.

Definition at line 182 of file [virtio-console-device](#).

References [L4virtio::Svr::Console::Virtio_con::Virtio_con\(\)](#).

Here is the call graph for this function:



15.407.3 Member Function Documentation

15.407.3.1 notify_queue()

```
void L4virtio::Svr::Console::Device::notify_queue (
    Virtqueue * queue) [inline], [override], [virtual]
```

Notify queue of available data.

Parameters

| | |
|--------------|--------------------------------------|
| <i>queue</i> | Virtqueue to notify. |
|--------------|--------------------------------------|

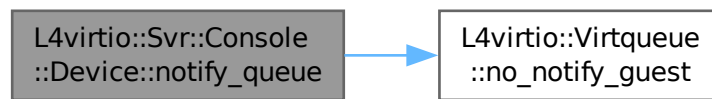
This callback is called whenever data is sent to `queue`. It is the responsibility of the derived class to perform all necessary notification actions, e.g. triggering guest interrupts.

Implements [L4virtio::Svr::Console::Virtio_con](#).

Definition at line 202 of file [virtio-console-device](#).

References [L4VIRTIO_IRQ_STATUS_VRING](#), and [L4virtio::Virtqueue::no_notify_guest\(\)](#).

Here is the call graph for this function:



15.407.3.2 port()

```
Port * L4virtio::Svr::Console::Device::port (
    unsigned port) [inline], [override], [protected], [virtual]
```

Return the specified port.

Parameters

| | |
|-------------|--------------|
| <i>port</i> | Port number. |
|-------------|--------------|

Precondition

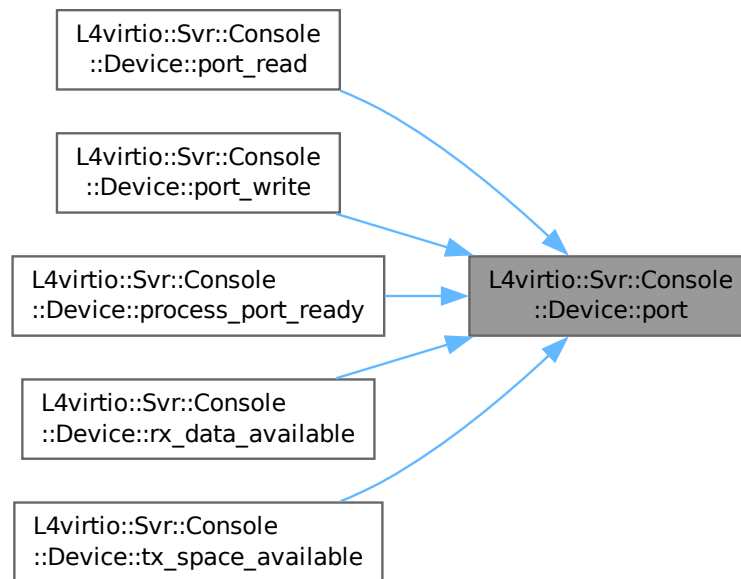
Port number must be lower than the configured maximum number of ports.

Implements [L4virtio::Svr::Console::Virtio_con](#).

Definition at line 450 of file [virtio-console-device](#).

Referenced by [port_read\(\)](#), [port_write\(\)](#), [process_port_ready\(\)](#), [rx_data_available\(\)](#), and [tx_space_available\(\)](#).

Here is the caller graph for this function:



15.407.3.3 port_read()

```

unsigned L4virtio::Svr::Console::Device::port_read (
    char * buf,
    unsigned len,
    unsigned port = 0) [inline]
  
```

Read data from port.

Will read up to *len* bytes from *port* into *buf*. Returns the number of bytes read, which may be less if not enough data was available. If all data was read, the [rx_data_available\(\)](#) callback will be invoked the next time the driver queues new data for the port. The callback won't be called again until all data was consumed again.

Parameters

| | |
|-------------|--|
| <i>buf</i> | The destination buffer |
| <i>len</i> | Size of the buffer |
| <i>port</i> | Port index to read data from |

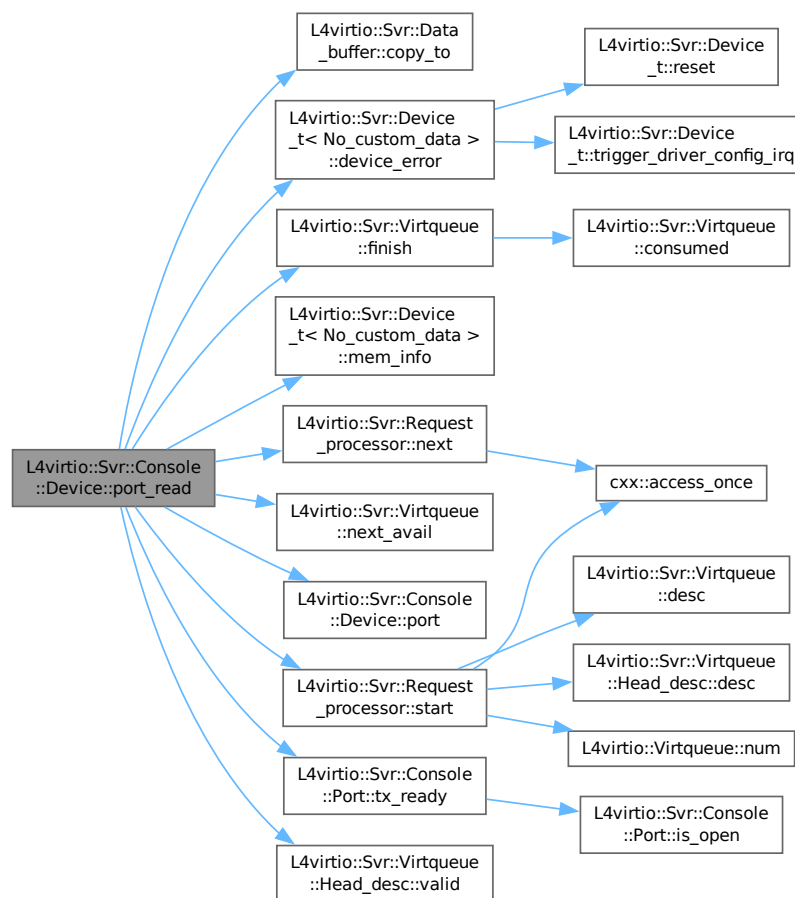
Returns

Number of bytes read

Definition at line 272 of file [virtio-console-device](#).

References [L4virtio::Svr::Data_buffer::copy_to\(\)](#), [L4virtio::Svr::Device_t< No_custom_data >::device_error\(\)](#), [L4virtio::Svr::Virtqueue::finish\(\)](#), [L4virtio::Svr::Data_buffer::left](#), [L4virtio::Svr::Device_t< No_custom_data >::mem_info\(\)](#), [L4virtio::Svr::Request_processor::next\(\)](#), [L4virtio::Svr::Virtqueue::next_avail\(\)](#), [port\(\)](#), [L4virtio::Svr::Data_buffer::pos](#), [L4virtio::Svr::Console::Device_port::request](#), [L4virtio::Svr::Console::Device_port::rp](#), [L4virtio::Svr::Console::Device_port::src](#), [L4virtio::Svr::Request_processor::start\(\)](#), [L4virtio::Svr::Console::Port::tx](#), [L4virtio::Svr::Console::Port::tx_ready\(\)](#), and [L4virtio::Svr::Virtqueue::Head_desc::valid\(\)](#).

Here is the call graph for this function:



15.407.3.4 port_write()

```

unsigned L4virtio::Svr::Console::Device::port_write (
    char const * buf,
    unsigned len,
    unsigned port = 0) [inline]

```

Write data to port.

Will write up to *len* bytes to *port* from *buf*. Returns the number of bytes written, which may be less if not enough virtio buffers were available. If not all data could be written, the [tx_space_available\(\)](#) callback will be invoked the next time the driver queues new receive buffers for the port. The callback won't be called again until all receive buffers were filled again.

Parameters

| | |
|-------------|---|
| <i>buf</i> | The souce buffer |
| <i>len</i> | Size of the buffer |
| <i>port</i> | Port index to write data to |

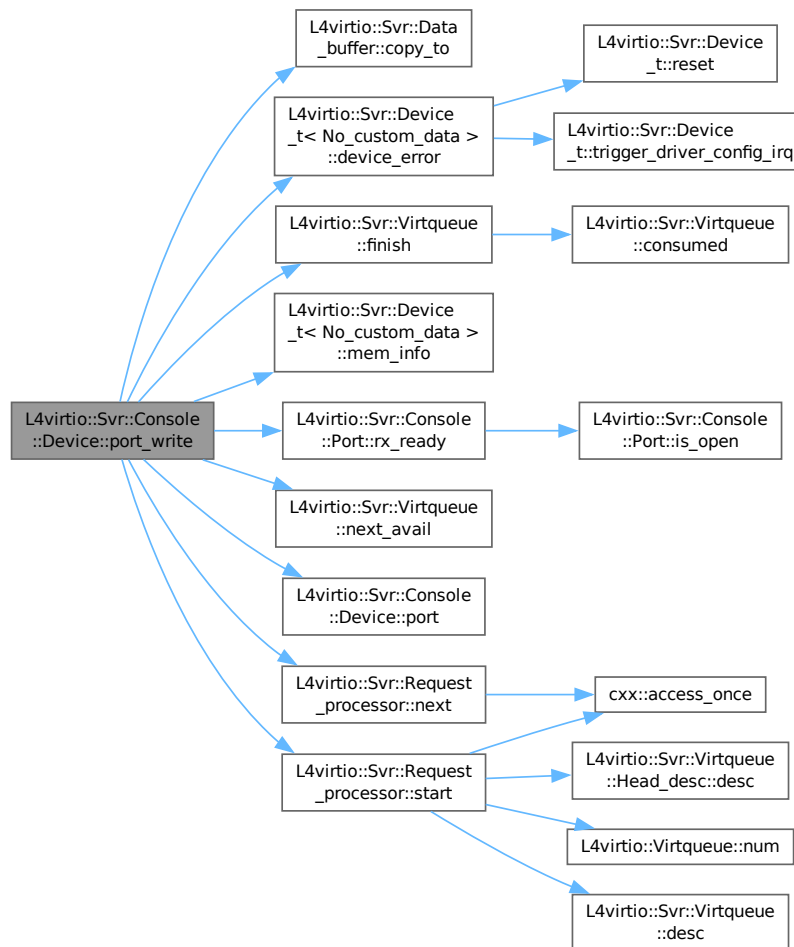
Returns

Number of bytes written

Definition at line [341](#) of file [virtio-console-device](#).

References [L4virtio::Svr::Data_buffer::copy_to\(\)](#), [L4virtio::Svr::Device_t< No_custom_data >::device_error\(\)](#), [L4virtio::Svr::Virtqueue::finish\(\)](#), [L4virtio::Svr::Data_buffer::left](#), [L4virtio::Svr::Device_t< No_custom_data >::mem_info\(\)](#), [L4virtio::Svr::Request_processor::next\(\)](#), [L4virtio::Svr::Virtqueue::next_avail\(\)](#), [port\(\)](#), [L4virtio::Svr::Data_buffer::pos](#), [L4virtio::Svr::Console::Port::rx](#), [L4virtio::Svr::Console::Port::rx_ready\(\)](#), and [L4virtio::Svr::Request_processor::start\(\)](#).

Here is the call graph for this function:



15.407.3.5 process_device_ready()

```
void L4virtio::Svr::Console::Device::process_device_ready (
    l4_uint16_t value) [inline], [override], [virtual]
```

Callback called on DEVICE_READY event.

Parameters

| | |
|--------------|--|
| <i>value</i> | The value field of the control message, indicating if the initialization was successful. |
|--------------|--|

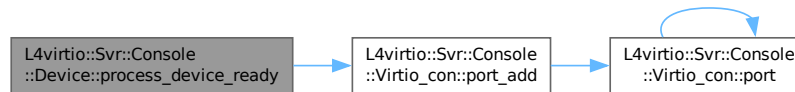
By default, this function adds all ports if the driver indicates successful initialization. Override this function to perform custom actions for a DEVICE_READY event. It is then your responsibility to inform the driver about usable ports, see [port_add\(\)](#).

Implements [L4virtio::Svr::Console::Virtio_con](#).

Definition at line 399 of file [virtio-console-device](#).

References [L4virtio::Svr::Console::Virtio_con::port_add\(\)](#).

Here is the call graph for this function:



15.407.3.6 process_port_open()

```
void L4virtio::Svr::Console::Device::process_port_open (
    l4_uint32_t id,
    l4_uint16_t value) [inline], [override], [virtual]
```

Callback called on PORT_OPEN event.

Parameters

| | |
|--------------|--|
| <i>id</i> | The id field of the control message, i.e. the port number. |
| <i>value</i> | The value field of the control message, indicating if the port was opened or closed. |

The default implementation does nothing. Override it to implement some custom logic to respond to open/close events of the driver.

Implements [L4virtio::Svr::Console::Virtio_con](#).

Definition at line 443 of file [virtio-console-device](#).

15.407.3.7 process_port_ready()

```
void L4virtio::Svr::Console::Device::process_port_ready (
    l4_uint32_t id,
    l4_uint16_t value) [inline], [override], [virtual]
```

Callback called on PORT_READY event.

Parameters

| | |
|--------------|--|
| <i>id</i> | The id field of the control message, i.e. the port number. |
| <i>value</i> | The value field of the control message, indicating if the initialization was successful. |

By default, this function opens the port if the driver is ready. Otherwise, the port is removed if the driver failed to set it up correctly. Override this function to perform custom actions for a PORT_READY event, *after* the generic

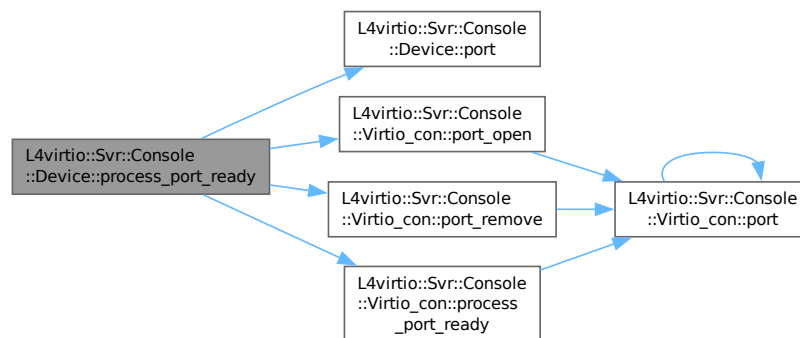
management of the base class. It is then your responsibility to inform the driver about connected or unusable ports. See `port_open()` and `port_remove()`.

Reimplemented from `L4virtio::Svr::Console::Virtio_con`.

Definition at line 422 of file `virtio-console-device`.

References `port()`, `L4virtio::Svr::Console::Port::Port_failed`, `L4virtio::Svr::Console::Virtio_con::port_open()`, `L4virtio::Svr::Console::Port::Port_ready`, `L4virtio::Svr::Console::Virtio_con::port_remove()`, `L4virtio::Svr::Console::Virtio_con::process_port_ready` and `L4virtio::Svr::Console::Port::status`.

Here is the call graph for this function:



The documentation for this class was generated from the following file:

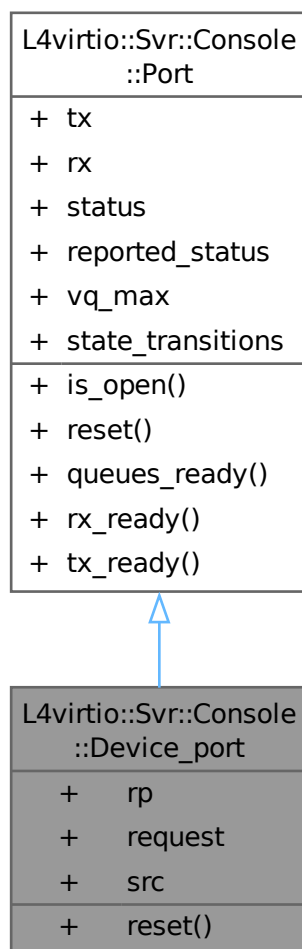
- `l4/l4virtio/server/virtio-console-device`

15.408 L4virtio::Svr::Console::Device_port Struct Reference

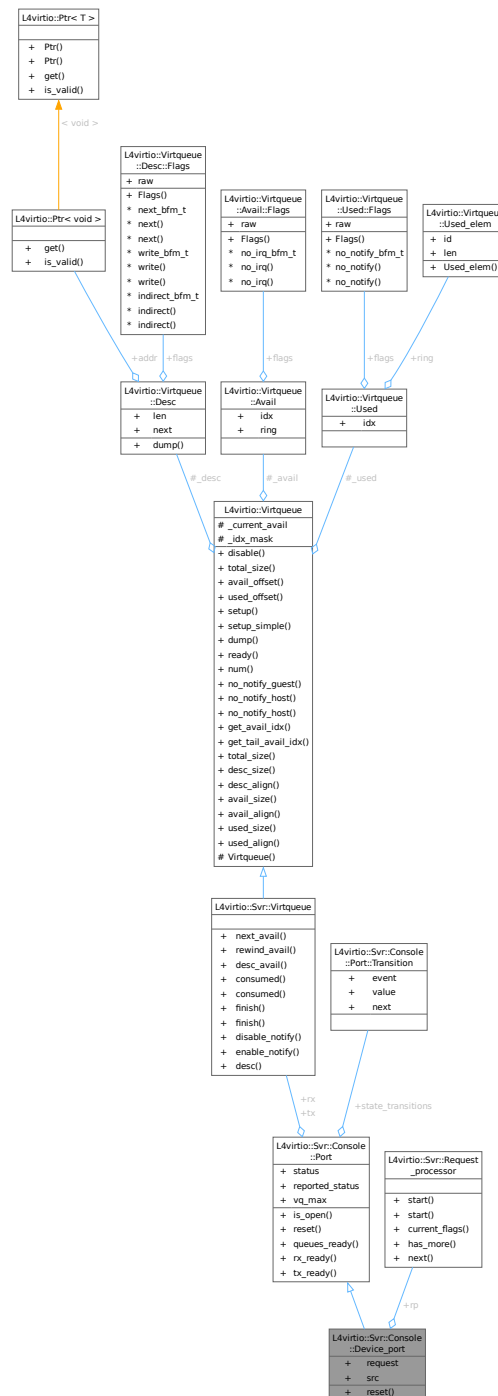
A console port with associated read/write state.

```
#include <virtio-console-device>
```

Inheritance diagram for L4virtio::Svr::Console::Device_port:



Collaboration diagram for L4virtio::Svr::Console::Device_port:



Public Member Functions

- void **reset** () override
Reset the port to the initial state and disable its virtqueues.

Public Member Functions inherited from L4virtio::Svr::Console::Port

- bool **is_open** () const

- *Check that the port is open.*
- bool **queues_ready** () const
Check that both virtqueues are set up correctly.
- bool **rx_ready** () const
Check that device implementation may write to receive queues.
- bool **tx_ready** () const
Check that device implementation may read from transmit queues.

Data Fields

- [Request_processor](#) **rp**
Request processor associated with current request.
- Virtqueue::Request **request**
Current virtio tx queue request.
- [Buffer](#) **src**
Source data block to process.

Data Fields inherited from [L4virtio::Svr::Console::Port](#)

- Virtqueue **tx**
Receiveq of the port.
- Virtqueue **rx**
Transmitq of the port.
- [Port_status](#) **status**
State the port is in.
- [Port_status](#) **reported_status**
State the port was last reported.
- unsigned **vq_max**
Maximum queue sizes for this port.

Additional Inherited Members

Public Types inherited from [L4virtio::Svr::Console::Port](#)

- enum [Port_status](#) {
 [Port_disabled](#) = 0 , [Port_added](#) , [Port_ready](#) , [Port_open](#) ,
 [Port_failed](#) , [Port_num_states](#) }
Possible states of a virtio console port.
- enum
Size of control queues, also used as default size.

Static Public Attributes inherited from [L4virtio::Svr::Console::Port](#)

- static constexpr [Transition state_transitions](#) [[Port_num_states](#)][[Port_num_states](#)]
State transition table from last report state to current state.

15.408.1 Detailed Description

A console port with associated read/write state.

Tracks the notification of the device implementation and holds the state when receiving data from the driver.

Definition at line 26 of file [virtio-console-device](#).

The documentation for this struct was generated from the following file:

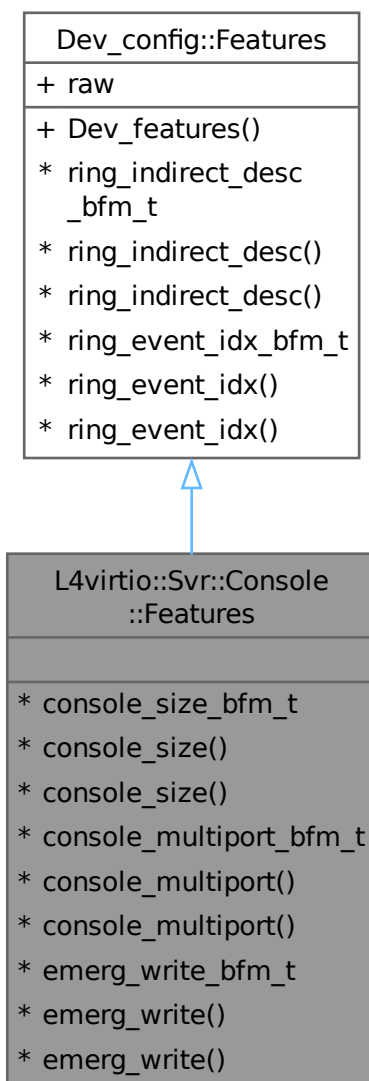
- l4/l4virtio/server/virtio-console-device

15.409 L4virtio::Svr::Console::Features Struct Reference

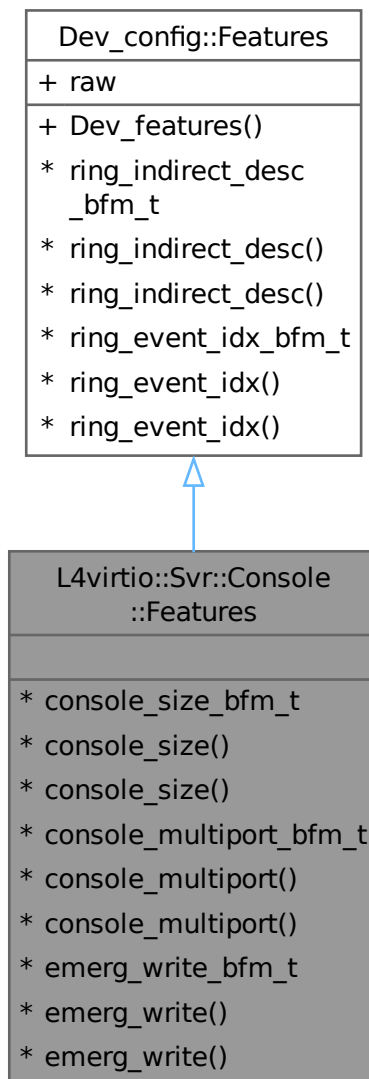
Virtio console specific feature bits.

```
#include <virtio-console>
```

Inheritance diagram for L4virtio::Svr::Console::Features:



Collaboration diagram for L4virtio::Svr::Console::Features:



- typedef `cxx::Bitfield< decltype(raw), 0, 0 >` `console_size_bfm_t`
Configuration `cols` and `rows` are valid.
- constexpr `console_size_bfm_t::Val console_size ()` const
Get the `console_size` bits (0 to 0) of `raw`.
- constexpr `console_size_bfm_t::Ref console_size ()`
Get a reference to the `console_size` bits (0 to 0) of `raw`.
- typedef `cxx::Bitfield< decltype(raw), 1, 1 >` `console_multiport_bfm_t`
`Device` has support for multiple ports.
- constexpr `console_multiport_bfm_t::Val console_multiport ()` const
Get the `console_multiport` bits (1 to 1) of `raw`.

- constexpr [console_multiport_bfm_t::Ref](#) **console_multiport** ()
Get a reference to the [console_multiport](#) bits (1 to 1) of [raw](#).
- typedef [cxx::Bitfield](#)< decltype([raw](#)), 2, 2 > **emerg_write_bfm_t**
[Device](#) has support for emergency write.
- constexpr [emerg_write_bfm_t::Val](#) **emerg_write** () const
Get the [emerg_write](#) bits (2 to 2) of [raw](#).
- constexpr [emerg_write_bfm_t::Ref](#) **emerg_write** ()
Get a reference to the [emerg_write](#) bits (2 to 2) of [raw](#).

Additional Inherited Members

- typedef [cxx::Bitfield](#)< decltype([raw](#)), 28, 28 > **ring_indirect_desc_bfm_t**
Type to access the [ring_indirect_desc](#) bits (28 to 28) of [raw](#).
- typedef [cxx::Bitfield](#)< decltype([raw](#)), 29, 29 > **ring_event_idx_bfm_t**
Type to access the [ring_event_idx](#) bits (29 to 29) of [raw](#).

Public Member Functions inherited from [L4virtio::Svr::Dev_features](#)

- **Dev_features** ([l4_uint32_t](#) v)
Make Features from a raw bitmap.
- constexpr [ring_indirect_desc_bfm_t::Val](#) **ring_indirect_desc** () const
Get the [ring_indirect_desc](#) bits (28 to 28) of [raw](#).
- constexpr [ring_indirect_desc_bfm_t::Ref](#) **ring_indirect_desc** ()
Get a reference to the [ring_indirect_desc](#) bits (28 to 28) of [raw](#).
- constexpr [ring_event_idx_bfm_t::Val](#) **ring_event_idx** () const
Get the [ring_event_idx](#) bits (29 to 29) of [raw](#).
- constexpr [ring_event_idx_bfm_t::Ref](#) **ring_event_idx** ()
Get a reference to the [ring_event_idx](#) bits (29 to 29) of [raw](#).

Data Fields inherited from [L4virtio::Svr::Dev_features](#)

- [l4_uint32_t](#) **raw**
The raw value of the features bitmap.

15.409.1 Detailed Description

Virtio console specific feature bits.

Definition at line 18 of file [virtio-console](#).

15.409.2 Member Typedef Documentation

15.409.2.1 console_multiport_bfm_t

```
typedef cxx::Bitfield<decltype(raw), 1, 1> L4virtio::Svr::Console::Features::console_multiport_bfm_t
```

Device has support for multiple ports.

Type to access the `console_multiport` bits (1 to 1) of `raw`.

Definition at line 25 of file `virtio-console`.

15.409.2.2 console_size_bfm_t

```
typedef cxx::Bitfield<decltype(raw), 0, 0> L4virtio::Svr::Console::Features::console_size_bfm_t
```

Configuration `cols` and `rows` are valid.

Type to access the `console_size` bits (0 to 0) of `raw`.

Definition at line 23 of file `virtio-console`.

15.409.2.3 emerg_write_bfm_t

```
typedef cxx::Bitfield<decltype(raw), 2, 2> L4virtio::Svr::Console::Features::emerg_write_bfm_t
```

Device has support for emergency write.

Type to access the `emerg_write` bits (2 to 2) of `raw`.

Definition at line 27 of file `virtio-console`.

The documentation for this struct was generated from the following file:

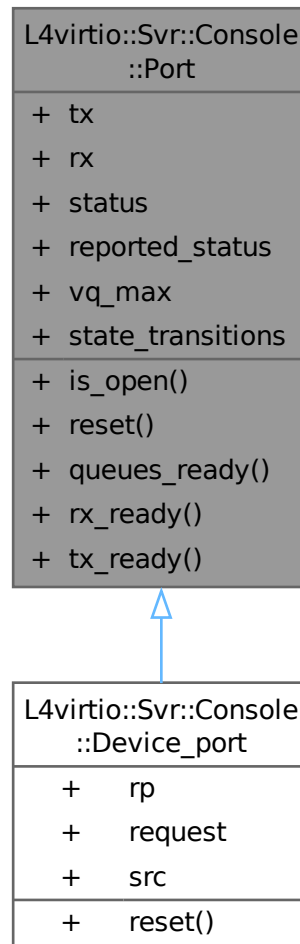
- `l4/l4virtio/server/virtio-console`

15.410 L4virtio::Svr::Console::Port Struct Reference

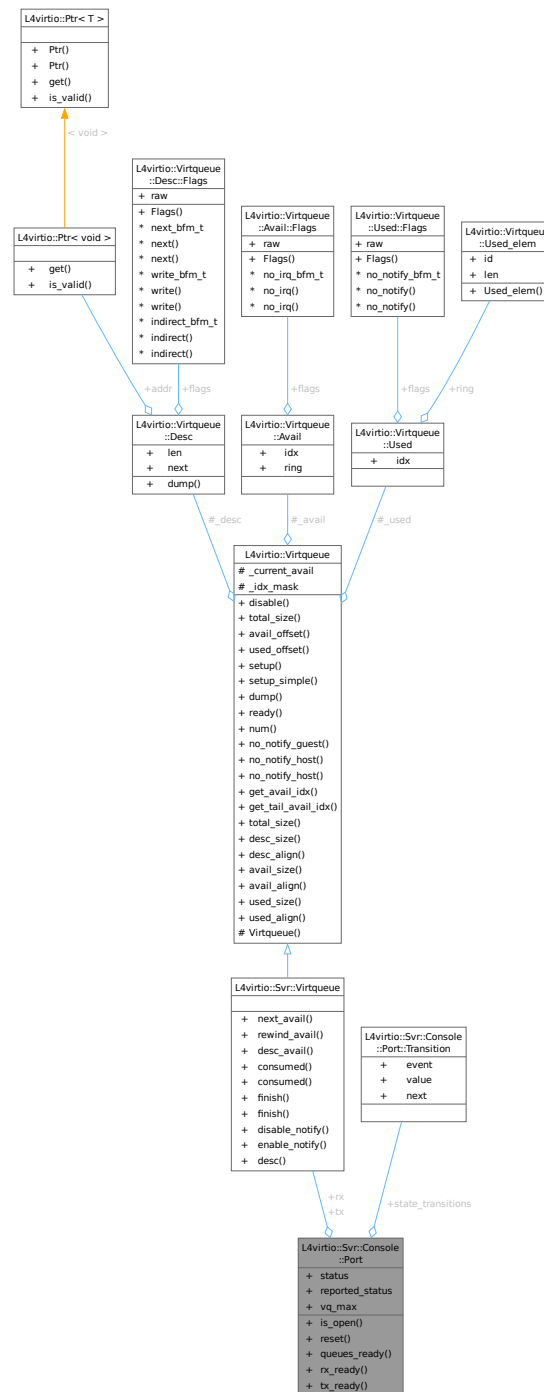
Representation of a Virtio console port.

```
#include <virtio-console>
```

Inheritance diagram for L4virtio::Svr::Console::Port:



Collaboration diagram for L4virtio::Svr::Console::Port:



Data Structures

- struct [Transition](#)

State transition from last report state to current state.

Public Types

- enum [Port_status](#) {
 [Port_disabled](#) = 0 , [Port_added](#) , [Port_ready](#) , [Port_open](#) ,
 [Port_failed](#) , [Port_num_states](#) }

Possible states of a virtio console port.

- enum
Size of control queues, also used as default size.

Public Member Functions

- bool **is_open** () const
Check that the port is open.
- virtual void **reset** ()
Reset the port to the initial state and disable its virtqueues.
- bool **queues_ready** () const
Check that both virtqueues are set up correctly.
- bool **rx_ready** () const
Check that device implementation may write to receive queues.
- bool **tx_ready** () const
Check that device implementation may read from transmit queues.

Data Fields

- Virtqueue **tx**
Receiveq of the port.
- Virtqueue **rx**
Transmitq of the port.
- [Port_status](#) **status**
State the port is in.
- [Port_status](#) **reported_status**
State the port was last reported.
- unsigned **vq_max**
Maximum queue sizes for this port.

Static Public Attributes

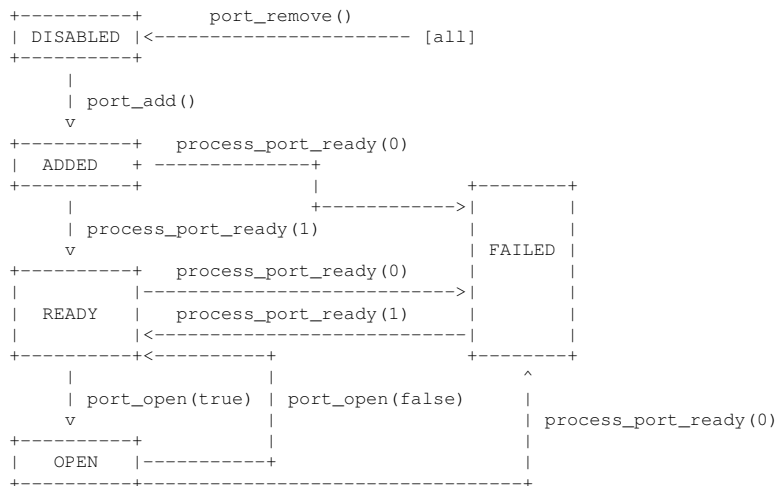
- static constexpr [Transition](#) **state_transitions** [[Port_num_states](#)][[Port_num_states](#)]
State transition table from last report state to current state.

15.410.1 Detailed Description

Representation of a Virtio console port.

Each port consists of a pair of queues for sending and receiving.

A port may be added and removed at runtime when the multi-port feature is enabled. The states are as follows:



Definition at line 109 of file virtio-console.

15.410.2 Member Enumeration Documentation

15.410.2.1 Port_status

```
enum L4virtio::Svr::Console::Port::Port_status
```

Possible states of a virtio console port.

Enumerator

| | |
|-----------------|--|
| Port_disabled | Reset state, waiting for port to be added. |
| Port_added | Port has been added by device, waiting for ready message. |
| Port_ready | Port is ready but still closed. |
| Port_open | Port is in a working state. |
| Port_failed | Device failure, port unusable. |
| Port_num_states | Number of port states. Must be last. |

Definition at line 114 of file virtio-console.

15.410.3 Field Documentation

15.410.3.1 state_transitions

Transition L4virtio::Svr::Console::Port::state_transitions[Port_num_states][Port_num_states]
[static], [constexpr]

State transition table from last report state to current state.

Not all transitions can be made directly. For example, if the last reported state was `Port_disabled` and the current state is `Port_open`, the device has to send two messages: `Control_message::Device_add` and `Control_message::Port_open`. This is expressed by going through an intermediate state (`Port_ready`) on the reporting side.

For the purpose of the driver there are only three coarse states:

1. The port does not exist (`Port_disabled`).
2. The port exists but is closed on the device side (`Port_added`, `Port_ready`, `Port_failed`).
3. The port exists and is open on the device side (`Port_open`).

The state transition table with `Port_added`, `Port_ready` and `Port_failed` as current state are thus identical.

Definition at line 195 of file [virtio-console](#).

The documentation for this struct was generated from the following file:

- `I4/I4virtio/server/virtio-console`

15.411 L4virtio::Svr::Console::Port::Transition Struct Reference

State transition from last report state to current state.

```
#include <virtio-console>
```

Collaboration diagram for L4virtio::Svr::Console::Port::Transition:

| L4virtio::Svr::Console ::Port::Transition | |
|--|-------|
| + | event |
| + | value |
| + | next |
| | |

Data Fields

- [l4_int16_t](#) **event**
[Control_message::Events](#) or <0 if no event is sent.
- [l4_uint16_t](#) **value**
Extra information.
- [Port_status](#) **next**
Next [Port_status](#) state.

15.411.1 Detailed Description

State transition from last report state to current state.

Definition at line 169 of file [virtio-console](#).

The documentation for this struct was generated from the following file:

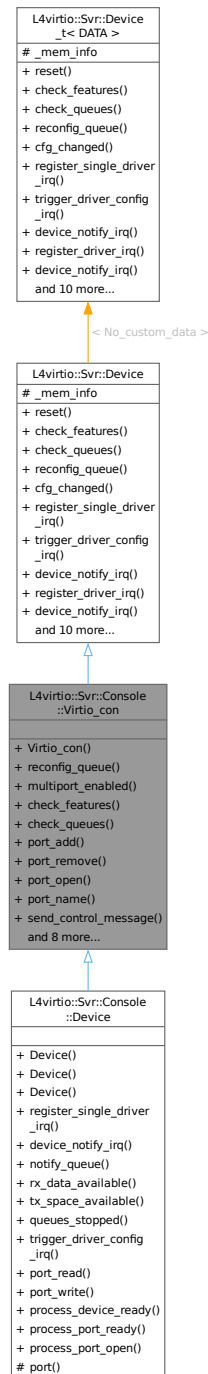
- l4/l4virtio/server/virtio-console

15.412 L4virtio::Svr::Console::Virtio_con Class Reference

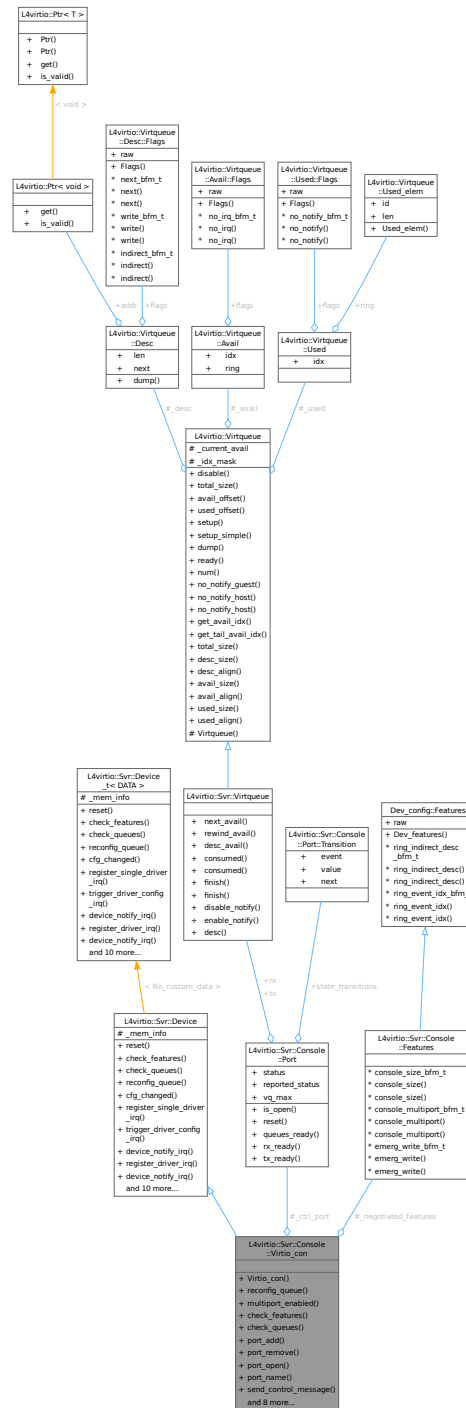
Base class implementing a virtio console functionality.

```
#include <virtio-console>
```

Inheritance diagram for L4virtio::Svr::Console::Virtio_con:



Collaboration diagram for L4virtio::Svr::Console::Virtio_con:



Public Member Functions

- **Virtio_con** (unsigned max_ports, bool enable_multipoint)
Create a new multipoint console device.
- int **reconfig_queue** (unsigned index) override
callback for client queue-config request
- bool **multipoint_enabled** () const

- Return true if the multiport feature is enabled and control queues are available.*

 - bool **check_features** (void) override
callback for checking the subset of accepted features
 - bool **check_queues** () override
callback for checking if the queues at DRIVER_OK transition
 - int **port_add** (unsigned idx)
Send a DEVICE_ADD message and update the internal state.
 - int **port_remove** (unsigned idx)
Send a DEVICE_REMOVE message and update the internal state.
 - int **port_open** (unsigned idx, bool open)
Send a PORT_OPEN message and update the internal state.
 - int **port_name** (unsigned idx, char const *name)
Send a PORT_NAME message to announce the port name.
 - int **send_control_message** (l4_uint32_t idx, l4_uint16_t event, l4_uint16_t value=0, const char *name=0)
Send control message to driver.
 - int **handle_control_message** ()
Handle control message received from the driver.
 - void **reset** () override
reset callback, called for doing a device reset
 - virtual void **reset_device** ()
Reset the state of the actual console device.
 - virtual void **notify_queue** (Virtqueue *queue)=0
Notify queue of available data.
 - virtual Port * **port** (unsigned port)=0
Return the specified port.
 - virtual void **process_device_ready** (l4_uint16_t value)=0
Callback called on DEVICE_READY event.
 - virtual void **process_port_ready** (l4_uint32_t id, l4_uint16_t value)
Callback called on PORT_READY event.
 - virtual void **process_port_open** (l4_uint32_t id, l4_uint16_t value)=0
Callback called on PORT_OPEN event.

Public Member Functions inherited from [L4virtio::Svr::Device_t< No_custom_data >](#)

- virtual void **cfg_changed** (unsigned)
callback for client device configuration changes
- virtual void **register_single_driver_irq** ()
callback for registering a single guest IRQ for all queues (old-style)
- virtual void **trigger_driver_config_irq** ()=0
callback for triggering configuration change notification IRQ
- virtual [L4::Cap< L4::Irq >](#) **device_notify_irq** () const
callback to gather the device notification IRQ (old-style)
- virtual void **register_driver_irq** (unsigned idx)
Callback for registering an notification IRQ (multi IRQ).
- virtual [L4::Cap< L4::Irq >](#) **device_notify_irq** (unsigned idx)
Callback to gather the device notification IRQ (multi IRQ).
- virtual unsigned **num_events_supported** () const
Return the highest notification index supported.
- [Device_t](#) ([Dev_config](#) *dev_config)
Make a device for the given config.

- `Mem_list const * mem_info () const`
Get the memory region list used for this device.
- `void reset_queue_config (unsigned idx, unsigned num_max, bool inc_generation=false, unsigned device_↵ notify_index=0)`
Trigger reset for the configuration space for queue idx.
- `void init_mem_info (unsigned num)`
Initialize the memory region list to the given maximum.
- `void device_error ()`
Transition device into DEVICE_NEEDS_RESET state.
- `bool setup_queue (Virtqueue *q, unsigned qn, unsigned num_max)`
Enable/disable the specified queue.
- `bool handle_mem_cmd_write ()`
Check for a value in the cmd register and handle a write.
- `void enable_trusted_ds_validation ()`
Enable trusted dataspace validation.
- `void add_trusted_dataspaces (std::shared_ptr< Ds_vector const > ds)`
Provide a list of trusted dataspaces that can be used for validation.

Additional Inherited Members

Protected Attributes inherited from `L4virtio::Svr::Device_t< No_custom_data >`

- `Mem_list _mem_info`
Memory region list.

15.412.1 Detailed Description

Base class implementing a virtio console functionality.

It is possible to activate the MULTIPORT feature, in which case incoming control messages need to be dispatched by calling `handle_control_message()`. The derived class must additionally override `process_device_ready()`, `process_port_ready()` and `process_port_open()` to implement the actual behaviour. The derived class has the following responsibilities:

- inform the driver about usable ports once the device is ready as signaled in `process_device_ready()`, see the wrapper `port_add()`.
- inform the driver about unusable ports, see the wrapper `port_remove()`.
- react to open/close events, see the wrapper `port_open()`.

This implementation provides no means to handle interrupts or notify guests, therefore derived classes have to provide this functionality, see `notify_queue()` and `handle_control_message()`. Similarly, all interaction with data queues has to be implemented. Memory for port structures must be managed by the implementor as well.

Use this class as a base to implement your own specific console device.

Definition at line 267 of file `virtio-console`.

15.412.2 Constructor & Destructor Documentation

15.412.2.1 Virtio_con()

```
L4virtio::Svr::Console::Virtio_con::Virtio_con (
    unsigned max_ports,
    bool enable_multiport) [inline], [explicit]
```

Create a new multiport console device.

Parameters

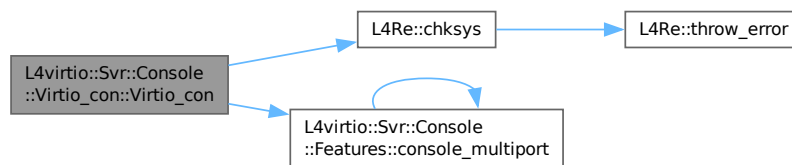
| | |
|-------------------------|--|
| <i>max_ports</i> | Maximum number of ports the device should be able to handle (ignored when <i>enable_multiport</i> is false). |
| <i>enable_multiport</i> | Enable the control queue for dynamic handling of ports. |

Definition at line 293 of file [virtio-console](#).

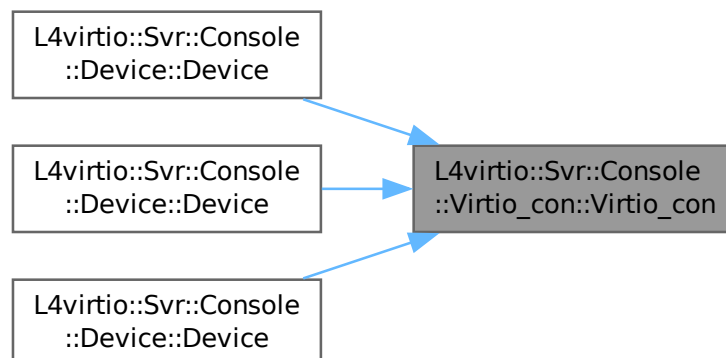
References [L4Re::chksys\(\)](#), [L4virtio::Svr::Console::Features::console_multiport\(\)](#), [L4_EINVAL](#), [L4VIRTIO_ID_CONSOLE](#), and [L4virtio::Svr::Dev_features::raw](#).

Referenced by [L4virtio::Svr::Console::Device::Device\(\)](#), [L4virtio::Svr::Console::Device::Device\(\)](#), and [L4virtio::Svr::Console::Device::Device\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.412.3 Member Function Documentation

15.412.3.1 handle_control_message()

```
int L4virtio::Svr::Console::Virtio_con::handle_control_message () [inline]
```

Handle control message received from the driver.

Return values

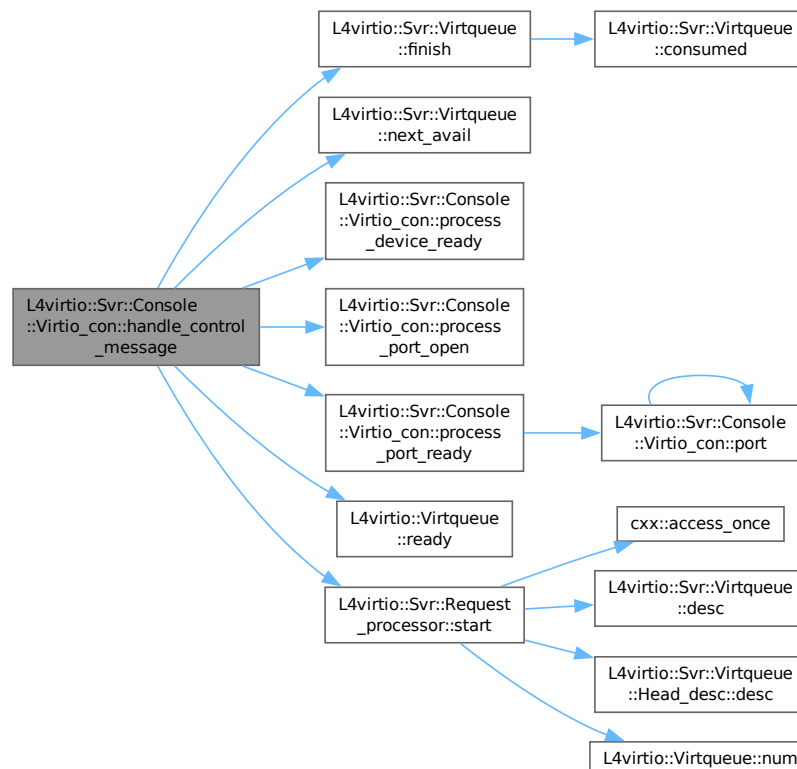
| | |
|-------------------|---------------------------------------|
| <i>L4_EOK</i> | Message has been handled. |
| <i>-L4_ENODEV</i> | Control queue is not ready. |
| <i>-L4_EINVAL</i> | Received an unexpected control event. |

This function performs the basic handling of control messages from the driver. It does all necessary work with the control queues and performs some sanity checks. All other work is deferred to the derived class, see [process_device_ready\(\)](#), [process_port_ready\(\)](#) and [process_port_open\(\)](#).

Definition at line 536 of file [virtio-console](#).

References [L4virtio::Svr::Console::Control_message::Device_ready](#), [L4virtio::Svr::Console::Control_message::event](#), [L4virtio::Svr::Virtqueue::finish\(\)](#), [L4virtio::Svr::Console::Control_message::id](#), [L4_EINVAL](#), [L4_ENODEV](#), [L4_EOK](#), [L4virtio::Svr::Console::Control_request::len](#), [L4virtio::Svr::Console::Control_request::msg](#), [L4virtio::Svr::Virtqueue::next_avail\(\)](#), [L4virtio::Svr::Console::Port::Port_disabled](#), [L4virtio::Svr::Console::Control_message::Port_open](#), [L4virtio::Svr::Console::Port::Port_op](#), [L4virtio::Svr::Console::Control_message::Port_ready](#), [process_device_ready\(\)](#), [process_port_open\(\)](#), [process_port_ready\(\)](#), [L4virtio::Virtqueue::ready\(\)](#), [L4virtio::Svr::Request_processor::start\(\)](#), and [L4virtio::Svr::Console::Control_message::value](#).

Here is the call graph for this function:



15.412.3.2 notify_queue()

```
virtual void L4virtio::Svr::Console::Virtio_con::notify_queue (
    Virtqueue * queue) [pure virtual]
```

Notify queue of available data.

Parameters

| | |
|--------------|--------------------------------------|
| <i>queue</i> | Virtqueue to notify. |
|--------------|--------------------------------------|

This callback is called whenever data is sent to `queue`. It is the responsibility of the derived class to perform all necessary notification actions, e.g. triggering guest interrupts.

Implemented in [L4virtio::Svr::Console::Device](#).

15.412.3.3 port()

```
virtual Port * L4virtio::Svr::Console::Virtio_con::port (
    unsigned port) [pure virtual]
```

Return the specified port.

Parameters

| | |
|-------------|------------------------------|
| <i>port</i> | Port number. |
|-------------|------------------------------|

Precondition

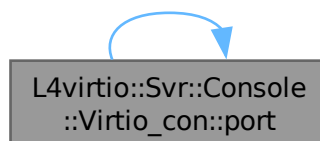
[Port](#) number must be lower than the configured maximum number of ports.

Implemented in [L4virtio::Svr::Console::Device](#).

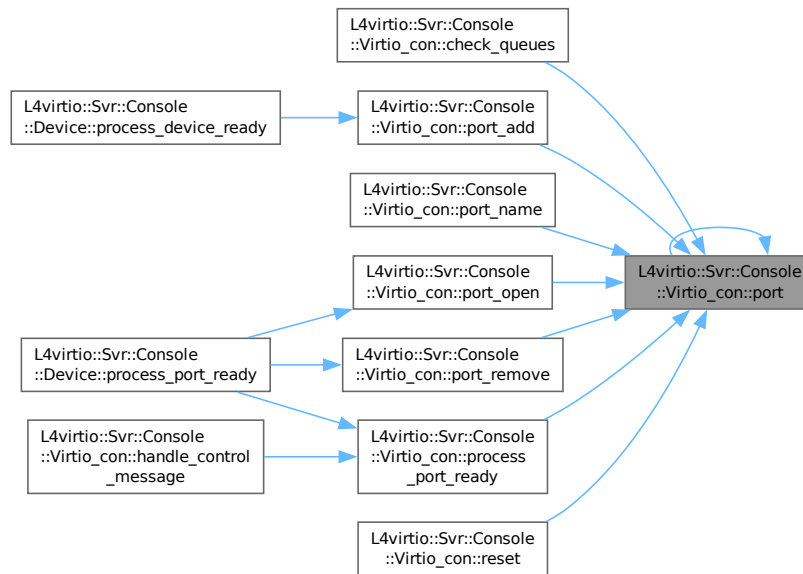
References [port\(\)](#).

Referenced by [check_queues\(\)](#), [port\(\)](#), [port_add\(\)](#), [port_name\(\)](#), [port_open\(\)](#), [port_remove\(\)](#), [process_port_ready\(\)](#), and [reset\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.412.3.4 port_add()

```
int L4virtio::Svr::Console::Virtio_con::port_add (
    unsigned idx) [inline]
```

Send a DEVICE_ADD message and update the internal state.

Parameters

| | |
|------------|----------------------------|
| <i>idx</i> | Port that should be added. |
|------------|----------------------------|

Return values

| | |
|------------------|---------------------------|
| <i>L4_EOK</i> | Message has been sent. |
| <i>-L4_EPERM</i> | Invalid state transition. |

Precondition

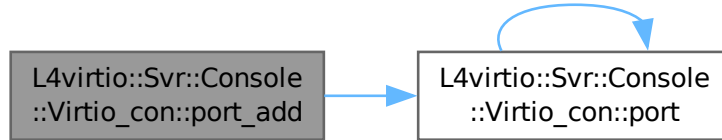
idx must be smaller than the configured number of ports.
 Port must not already exist.

Definition at line 379 of file [virtio-console](#).

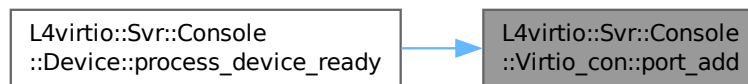
References [L4_EOK](#), [L4_EPERM](#), [port\(\)](#), [L4virtio::Svr::Console::Port::Port_added](#), [L4virtio::Svr::Console::Port::Port_disabled](#), and [L4virtio::Svr::Console::Port::status](#).

Referenced by [L4virtio::Svr::Console::Device::process_device_ready\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.412.3.5 port_name()

```
int L4virtio::Svr::Console::Virtio_con::port_name (
    unsigned idx,
    char const * name) [inline]
```

Send a PORT_NAME message to announce the port name.

Parameters

| | |
|-------------|---|
| <i>idx</i> | Port that should be opened or closed. |
| <i>name</i> | The port name |

Return values

| | |
|------------------|--|
| <i>L4_EOK</i> | Message has been sent. |
| <i>-L4_EPERM</i> | Control message is not allowed in the current state. |

Returns

Errors from [send_control_message\(\)](#)

Precondition

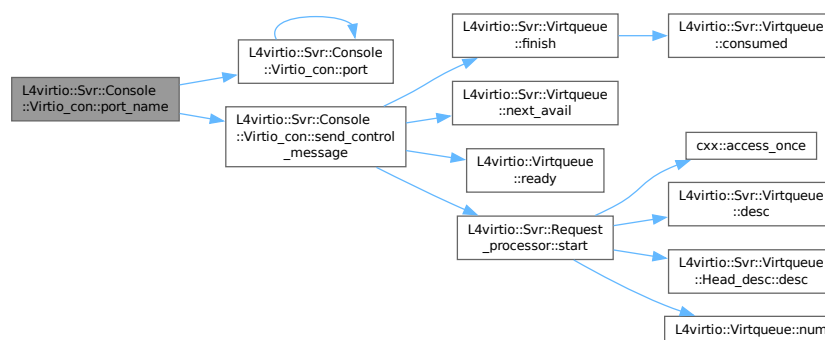
`idx` must be smaller than the configured number of ports.

[Port](#) must already exist.

Definition at line 455 of file [virtio-console](#).

References [L4_EPERM](#), [port\(\)](#), [L4virtio::Svr::Console::Port::Port_disabled](#), [L4virtio::Svr::Console::Control_message::Port_name](#), [send_control_message\(\)](#), and [L4virtio::Svr::Console::Port::status](#).

Here is the call graph for this function:

**15.412.3.6 port_open()**

```
int L4virtio::Svr::Console::Virtio_con::port_open (
    unsigned idx,
    bool open) [inline]
```

Send a PORT_OPEN message and update the internal state.

Parameters

| | |
|-------------|---|
| <i>idx</i> | Port that should be opened or closed. |
| <i>open</i> | Open or close port. |

Return values

| | |
|------------------|---------------------------|
| <i>L4_EOK</i> | Message has been sent. |
| <i>-L4_EPERM</i> | Invalid state transition. |

Precondition

`idx` must be smaller than the configured number of ports.

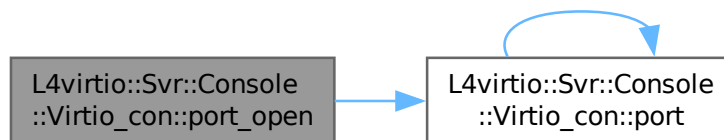
[Port](#) must be ready when opening or open when closing.

Definition at line 428 of file [virtio-console](#).

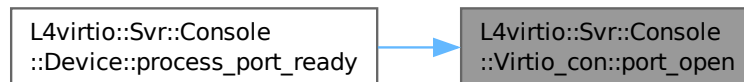
References [L4_EOK](#), [L4_EPERM](#), [port\(\)](#), [L4virtio::Svr::Console::Port::Port_open](#), [L4virtio::Svr::Console::Port::Port_ready](#), and [L4virtio::Svr::Console::Port::status](#).

Referenced by [L4virtio::Svr::Console::Device::process_port_ready\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**15.412.3.7 port_remove()**

```
int L4virtio::Svr::Console::Virtio_con::port_remove (
    unsigned idx) [inline]
```

Send a `DEVICE_REMOVE` message and update the internal state.

Parameters

| | |
|------------------|--|
| <code>idx</code> | Port that should be removed. |
|------------------|--|

Return values

| | |
|------------------------|---------------------------|
| <code>L4_EOK</code> | Message has been sent. |
| <code>-L4_EPERM</code> | Invalid state transition. |

Precondition

`idx` must be smaller than the configured number of ports.

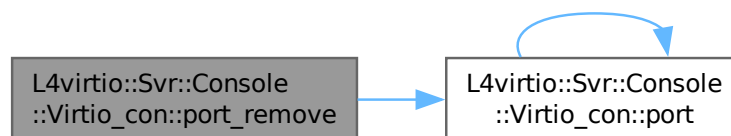
`Port` must already exist.

Definition at line 403 of file [virtio-console](#).

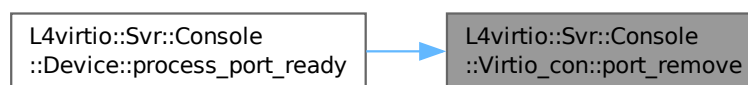
References [L4_EOK](#), [L4_EPERM](#), [port\(\)](#), [L4virtio::Svr::Console::Port::Port_disabled](#), and [L4virtio::Svr::Console::Port::status](#).

Referenced by [L4virtio::Svr::Console::Device::process_port_ready\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**15.412.3.8 process_device_ready()**

```
virtual void L4virtio::Svr::Console::Virtio_con::process_device_ready (
    14_uint16_t value) [pure virtual]
```

Callback called on DEVICE_READY event.

Parameters

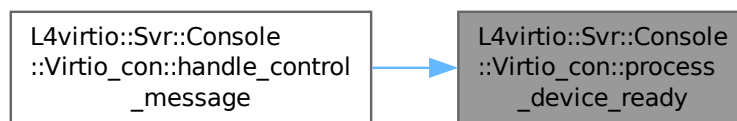
| | |
|--------------|--|
| <i>value</i> | The value field of the control message, indicating if the initialization was successful. |
|--------------|--|

Needs to be overridden by the derived class if the MULTIPOINT feature is enabled. Control messages may be sent only after the driver has successfully initialized the device.

Implemented in [L4virtio::Svr::Console::Device](#).

Referenced by [handle_control_message\(\)](#).

Here is the caller graph for this function:

**15.412.3.9 process_port_open()**

```
virtual void L4virtio::Svr::Console::Virtio_con::process_port_open (
    l4_uint32_t id,
    l4_uint16_t value) [pure virtual]
```

Callback called on PORT_OPEN event.

Parameters

| | |
|--------------|--|
| <i>id</i> | The id field of the control message, i.e. the port number. |
| <i>value</i> | The value field of the control message, indicating if the port was opened or closed. |

Signal that an application has opened the port. Can to be overridden by the derived class if the MULTIPOINT feature is enabled.

Implemented in [L4virtio::Svr::Console::Device](#).

Referenced by [handle_control_message\(\)](#).

Here is the caller graph for this function:



15.412.3.10 process_port_ready()

```
virtual void L4virtio::Svr::Console::Virtio_con::process_port_ready (
    l4_uint32_t id,
    l4_uint16_t value) [inline], [virtual]
```

Callback called on PORT_READY event.

Parameters

| | |
|--------------|--|
| <i>id</i> | The id field of the control message, i.e. the port number. |
| <i>value</i> | The value field of the control message, indicating if the initialization was successful. |

May be overridden by the derived class if the MULTIPORT feature is enabled. This default implementation just sets the status of the port according to the driver message.

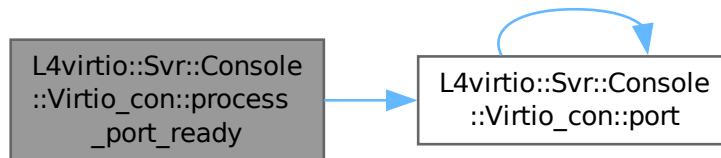
Reimplemented in [L4virtio::Svr::Console::Device](#).

Definition at line 698 of file [virtio-console](#).

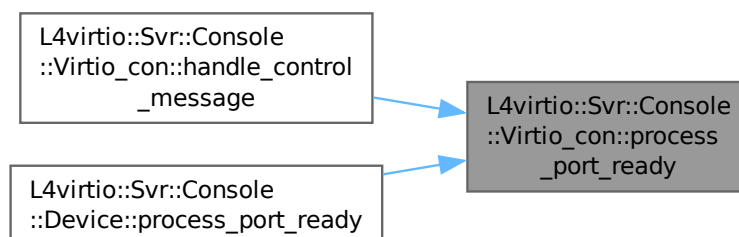
References [port\(\)](#), [L4virtio::Svr::Console::Port::Port_added](#), [L4virtio::Svr::Console::Port::Port_failed](#), [L4virtio::Svr::Console::Port::Port_ready](#), and [L4virtio::Svr::Console::Port::status](#).

Referenced by [handle_control_message\(\)](#), and [L4virtio::Svr::Console::Device::process_port_ready\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.412.3.11 reset_device()

```
virtual void L4virtio::Svr::Console::Virtio_con::reset_device () [inline], [virtual]
```

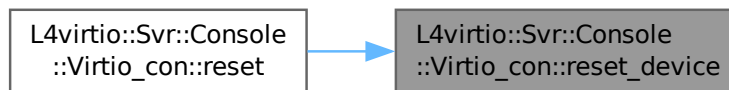
Reset the state of the actual console device.

This callback is called at the end of `reset()`, allowing the derived class to reset internal state.

Definition at line 652 of file `virtio-console`.

Referenced by `reset()`.

Here is the caller graph for this function:



15.412.3.12 send_control_message()

```
int L4virtio::Svr::Console::Virtio_con::send_control_message (
    l4_uint32_t idx,
    l4_uint16_t event,
    l4_uint16_t value = 0,
    const char * name = 0) [inline]
```

Send control message to driver.

Parameters

| | |
|--------------|---------------------------------------|
| <i>idx</i> | Port number. |
| <i>event</i> | Kind of control event. |
| <i>value</i> | Extra information for the event. |
| <i>name</i> | Name to be used for Port_name message |

Return values

| | |
|-------------------|---|
| <i>L4_EOK</i> | Message has been sent. |
| <i>-L4_ENODEV</i> | Control queue is not ready. |
| <i>-L4_EBUSY</i> | Currently no descriptor available in the control queue. |
| <i>-L4_ENOMEM</i> | Client-issued descriptor too small. Device will be set to failed state. |

Precondition

`port` must be smaller than the configured number of ports.

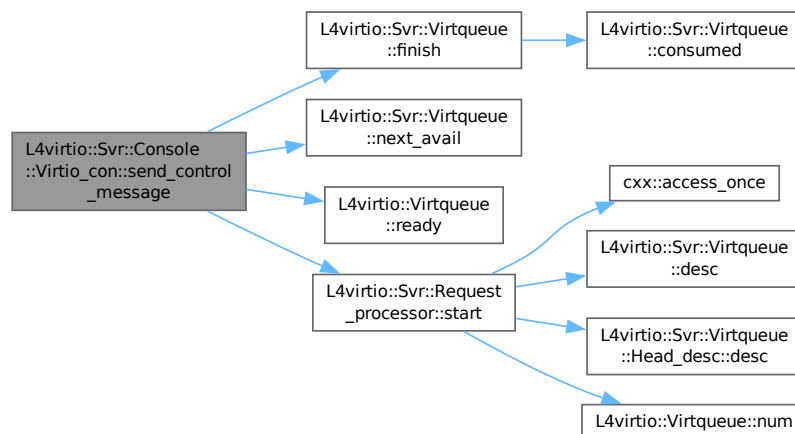
The convenience functions `port_add()`, `port_remove()` and `port_open()` should cover the most use cases and are the preferred way of communication with the driver. If you use this function directly, it is your responsibility to guarantee no invalid control messages are sent to the driver.

Definition at line 487 of file `virtio-console`.

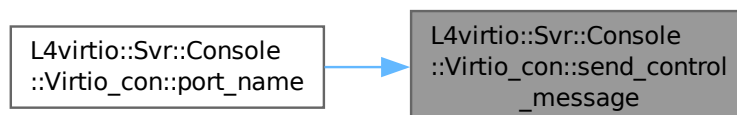
References `L4virtio::Svr::Virtqueue::finish()`, `L4_EBUSY`, `L4_ENODEV`, `L4_ENOMEM`, `L4_EOK`, `L4virtio::Svr::Console::Control_request::msg`, `L4virtio::Svr::Virtqueue::next_avail()`, `L4virtio::Svr::Console::Control_message::Port_name`, `L4virtio::Virtqueue::ready()`, and `L4virtio::Svr::Request_processor::start()`.

Referenced by `port_name()`.

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

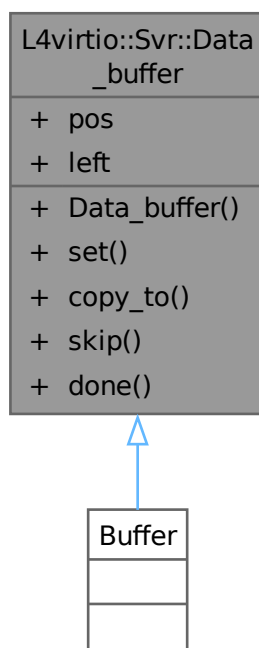
- `l4/l4virtio/server/virtio-console`

15.413 L4virtio::Svr::Data_buffer Struct Reference

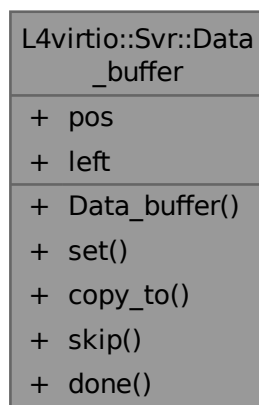
Abstract data buffer.

```
#include <virtio>
```

Inheritance diagram for L4virtio::Svr::Data_buffer:



Collaboration diagram for L4virtio::Svr::Data_buffer:



Public Member Functions

- `template<typename T>`
[Data_buffer](#) (T *p)
Create buffer for object p.
- `template<typename T>`
 void [set](#) (T *p)
Set buffer for object p.
- `l4_uint32_t copy_to` (Data_buffer *dst, `l4_uint32_t` max=UINT_MAX)
Copy contents from this buffer to the destination buffer.
- `l4_uint32_t skip` (`l4_uint32_t` bytes)
Skip given number of bytes in this buffer.
- bool [done](#) () const
Check if there are no more bytes left in the buffer.

Data Fields

- char * **pos**
Current buffer position.
- `l4_uint32_t` **left**
Bytes left in buffer.

15.413.1 Detailed Description

Abstract data buffer.

Definition at line 306 of file [virtio](#).

15.413.2 Constructor & Destructor Documentation

15.413.2.1 Data_buffer()

```
template<typename T>
L4virtio::Svr::Data_buffer::Data_buffer (
    T * p) [inline], [explicit]
```

Create buffer for object p.

Template Parameters

| | |
|----------|---------------------------|
| <i>T</i> | Type of object (implicit) |
|----------|---------------------------|

Parameters

| | |
|----------|--------------------|
| <i>p</i> | Pointer to object. |
|----------|--------------------|

The buffer shall point to the start of the object p and the size left is sizeof(T).

Definition at line 323 of file [virtio](#).

References [left](#), and [pos](#).

15.413.3 Member Function Documentation

15.413.3.1 `copy_to()`

```
14_uint32_t L4virtio::Svr::Data_buffer::copy_to (
    Data_buffer * dst,
    14_uint32_t max = UINT_MAX) [inline]
```

Copy contents from this buffer to the destination buffer.

Parameters

| | |
|------------|---|
| <i>dst</i> | Destination buffer. |
| <i>max</i> | (optional) Maximum number of bytes to copy. |

Returns

the number of bytes copied.

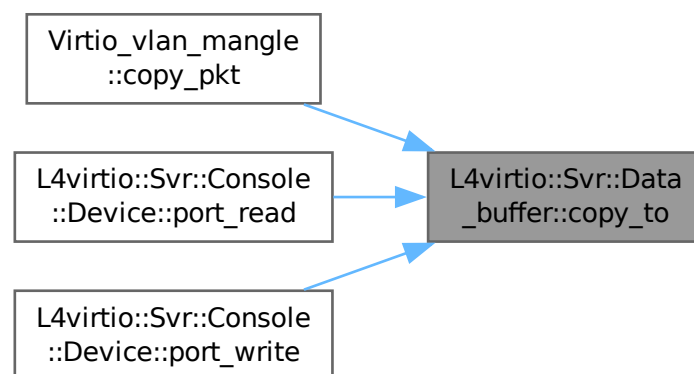
This function copies at most `max` bytes from this to `dst`. If `max` is omitted, copies the maximum number of bytes available that fit `dst`.

Definition at line 354 of file [virtio](#).

References [left](#), and [pos](#).

Referenced by [Virtio_vlan_mangle::copy_pkt\(\)](#), [L4virtio::Svr::Console::Device::port_read\(\)](#), and [L4virtio::Svr::Console::Device::port_w](#)

Here is the caller graph for this function:



15.413.3.2 done()

```
bool L4virtio::Svr::Data_buffer::done () const [inline]
```

Check if there are no more bytes left in the buffer.

Returns

true if there are no more bytes left in the buffer.

Definition at line 388 of file [virtio](#).

References [left](#).

15.413.3.3 set()

```
template<typename T>
void L4virtio::Svr::Data_buffer::set (
    T * p) [inline]
```

Set buffer for object p.

Template Parameters

| | |
|----------|---------------------------|
| <i>T</i> | Type of object (implicit) |
|----------|---------------------------|

Parameters

| | |
|----------|--------------------|
| <i>p</i> | Pointer to object. |
|----------|--------------------|

The buffer shall point to the start of the object p and the size left is sizeof(T).

Definition at line 337 of file [virtio](#).

References [left](#), and [pos](#).

15.413.3.4 skip()

```
l4_uint32_t L4virtio::Svr::Data_buffer::skip (
    l4_uint32_t bytes) [inline]
```

Skip given number of bytes in this buffer.

Parameters

| | |
|--------------|--|
| <i>bytes</i> | Number of bytes that shall be skipped. |
|--------------|--|

Returns

The number of bytes skipped.

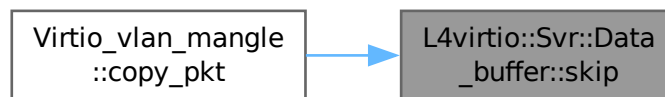
Try to skip the given number of bytes in this buffer, if there are less bytes left in the buffer that given then at most left bytes are skipped and the amount is returned.

Definition at line 375 of file [virtio](#).

References [left](#), and [pos](#).

Referenced by [Virtio_vlan_mangle::copy_pkt\(\)](#).

Here is the caller graph for this function:



The documentation for this struct was generated from the following file:

- [l4/l4virtio/server/virtio](#)

15.414 L4virtio::Svr::Dev_config Class Reference

Abstraction for L4-Virtio device config memory.

```
#include <l4virtio>
```


Collaboration diagram for L4virtio::Svr::Dev_config:



Public Member Functions

- [Dev_config](#) (l4_uint32_t vendor, l4_uint32_t device, unsigned cfg_size, l4_uint32_t num_queues=0)
Create a L4-Virtio config data space.
- [Dev_config](#) (Cfg_cap const &cfg, l4_addr_t cfg_offset, l4_uint32_t vendor, l4_uint32_t device, unsigned cfg_size, l4_uint32_t num_queues=0)
Setup an L4-Virtio config space in an existing data space.
- void [set_device_notify_index](#) (unsigned idx)
Set index of interrupt that driver should trigger for config notifications.
- l4_uint32_t [num_queues](#) () const
Return the number of queues currently usable.
- l4_uint32_t [guest_features](#) (unsigned idx) const
Return a specific set of guest features.
- l4_uint32_t [negotiated_features](#) (unsigned idx) const
Compute a specific set of negotiated features.
- Status [status](#) () const
Get current device status (trusted).
- l4_uint32_t [get_cmd](#) () const
Get the value from the cmd register.
- void [reset_cmd](#) ()
Reset the cmd register after execution of a command.
- void [set_status](#) (Status status)
Set device status register.
- void [add_irq_status](#) (l4_uint32_t status)

- Adds irq status bit.*
- void [set_device_needs_reset](#) ()
Set DEVICE_NEEDS_RESET bit in device status register.
- bool [change_queue_config](#) (l4_uint32_t num_queues)
Setup new queue configuration.
- [l4virtio_config_queue_t](#) volatile const * [qconfig](#) (unsigned index) const
Get queue read-only config data for queue with the given index.
- void [reset_hdr](#) (bool inc_generation=false) const
Reset the config header to the initial contents.
- bool [reset_queue](#) (unsigned index, unsigned num_max, bool inc_generation=false, unsigned device_notify←_index=0) const
Reset queue config for the given queue.
- [l4virtio_config_hdr_t](#) const volatile * [hdr](#) () const
Get a read-only pointer to the config header.
- [L4::Cap](#)< [L4Re::Dataspace](#) > [ds](#) () const
Get data-space capability for the shared config data space.
- [l4_addr_t](#) [ds_offset](#) () const
Return the offset into the config dataspace where the device configuration starts.

15.414.1 Detailed Description

Abstraction for L4-Virtio device config memory.

Virtio defines a device configuration mechanism, L4-Virtio implements this mechanism based on shared memory a [set_status\(\)](#) and a [config_queue\(\)](#) call. This class provides an abstraction for L4-Virtio host implementations to establish such a shared memory data space and providing the necessary contents and access functions.

Definition at line 52 of file [l4virtio](#).

15.414.2 Constructor & Destructor Documentation

15.414.2.1 Dev_config() [1/2]

```
L4virtio::Svr::Dev_config::Dev_config (
    l4_uint32_t vendor,
    l4_uint32_t device,
    unsigned cfg_size,
    l4_uint32_t num_queues = 0) [inline]
```

Create a L4-Virtio config data space.

Parameters

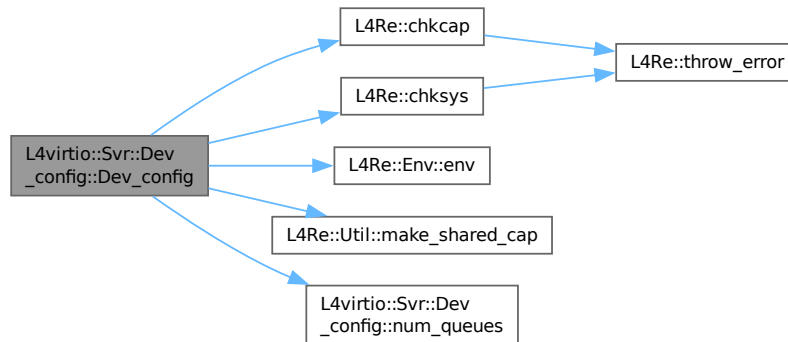
| | |
|-------------------|---|
| <i>vendor</i> | The vendor ID to store in config header. |
| <i>device</i> | The device ID to store in config header. |
| <i>cfg_size</i> | The size of the device-specific config data in bytes. |
| <i>num_queues</i> | The number of queues provided by the device. |

This constructor allocates a data space used for L4-virtio config attaches the data space to the local address space and writes the initial contents to the config header.

Definition at line 112 of file [l4virtio](#).

References [L4Re::chkcap\(\)](#), [L4Re::chksys\(\)](#), [L4Re::Env::env\(\)](#), [L4_PAGESIZE](#), [L4Re::Util::make_shared_cap\(\)](#), and [num_queues\(\)](#).

Here is the call graph for this function:



15.414.2.2 Dev_config() [2/2]

```

L4virtio::Svr::Dev_config::Dev_config (
    Cfg_cap const & cfg,
    l4_addr_t cfg_offset,
    l4_uint32_t vendor,
    l4_uint32_t device,
    unsigned cfg_size,
    l4_uint32_t num_queues = 0) [inline]

```

Setup an L4-Virtio config space in an existing data space.

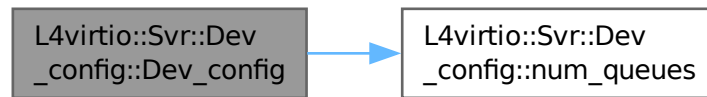
Parameters

| | |
|-------------------|---|
| <i>cfg</i> | Dataspace that should hold the L4-Virtio configuration. |
| <i>cfg_offset</i> | Offset into the dataspace where the configuration starts. |
| <i>vendor</i> | The vendor ID to store in config header. |
| <i>device</i> | The device ID to store in config header. |
| <i>cfg_size</i> | The size of the device-specific config data in bytes. |
| <i>num_queues</i> | The number of queues provided by the device. |

Definition at line 146 of file [l4virtio](#).

References [L4_PAGESIZE](#), and [num_queues\(\)](#).

Here is the call graph for this function:



15.414.3 Member Function Documentation

15.414.3.1 add_irq_status()

```
void L4virtio::Svr::Dev_config::add_irq_status (  
    14_uint32_t status) [inline]
```

Adds irq status bit.

Parameters

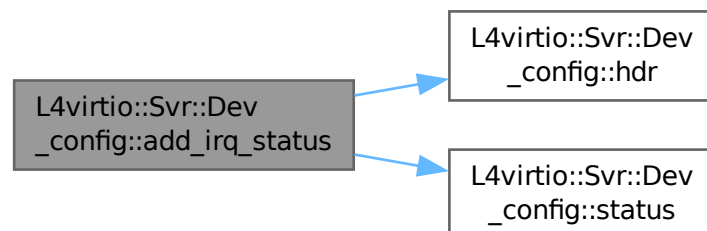
| | |
|---------------|--|
| <i>status</i> | The value to add to the irq status register. |
|---------------|--|

This function adds the status bit to the irq status register.

Definition at line 274 of file [l4virtio](#).

References [hdr\(\)](#), and [status\(\)](#).

Here is the call graph for this function:



15.414.3.2 change_queue_config()

```
bool L4virtio::Svr::Dev_config::change_queue_config (
    14_uint32_t num_queues) [inline]
```

Setup new queue configuration.

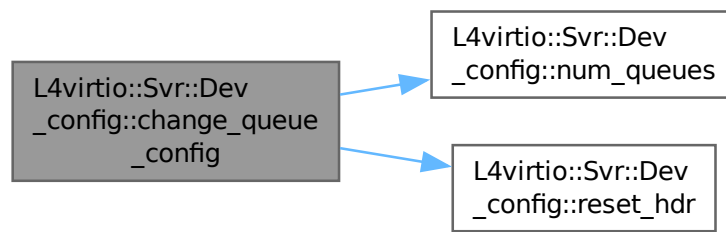
Parameters

| | |
|-------------------|--|
| <i>num_queues</i> | The number of queues provided by the device. |
|-------------------|--|

Definition at line 295 of file [l4virtio](#).

References [L4_PAGESIZE](#), [num_queues\(\)](#), and [reset_hdr\(\)](#).

Here is the call graph for this function:



15.414.3.3 ds()

```
L4::Cap< L4Re::Dataspace > L4virtio::Svr::Dev_config::ds () const [inline]
```

Get data-space capability for the shared config data space.

Returns

Capability for the shared config data space.

Definition at line 388 of file [l4virtio](#).

15.414.3.4 `get_cmd()`

```
l4_uint32_t L4virtio::Svr::Dev_config::get_cmd () const [inline]
```

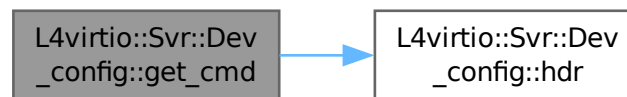
Get the value from the `cmd` register.

Note, the most significant eight bits are the command (0 is nothing to do). The upper eight bit are reset to zero after the command was handled.

Definition at line 239 of file [l4virtio](#).

References [hdr\(\)](#).

Here is the call graph for this function:



15.414.3.5 `guest_features()`

```
l4_uint32_t L4virtio::Svr::Dev_config::guest_features (
    unsigned idx) const [inline]
```

Return a specific set of guest features.

Parameters

| | |
|------------|--------------------------------------|
| <i>idx</i> | Index into the guest features array. |
|------------|--------------------------------------|

Return values

| | |
|------------|---------------------------------|
| <i>The</i> | selected set of guest features. |
|------------|---------------------------------|

This function returns a specific 32bit set of features enabled by the guest/driver. `idx` is the index in the guest features array, resp. the 32 bit set to return.

Definition at line 207 of file [l4virtio](#).

15.414.3.6 hdr()

```
l4virtio_config_hdr_t const volatile * L4virtio::Svr::Dev_config::hdr () const [inline]
```

Get a read-only pointer to the config header.

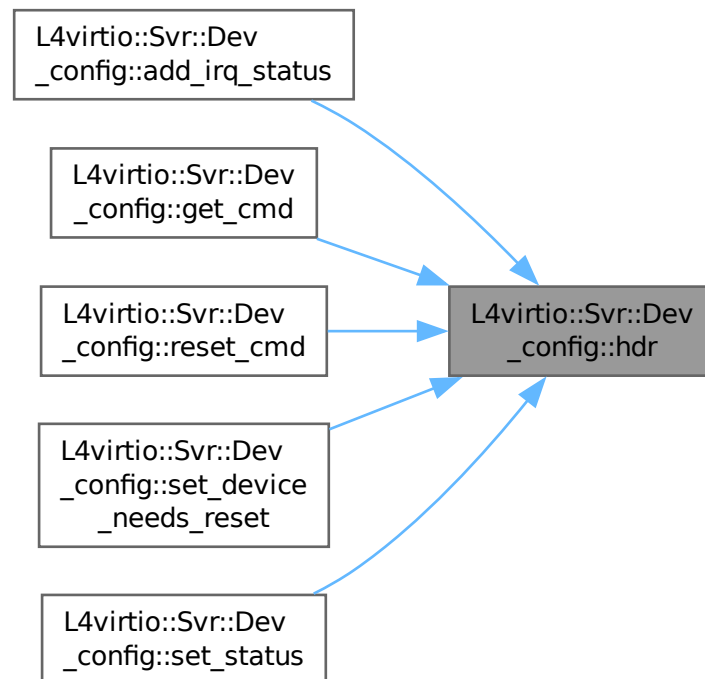
Returns

Read-only pointer to the shared config header.

Definition at line 381 of file [l4virtio](#).

Referenced by [add_irq_status\(\)](#), [get_cmd\(\)](#), [reset_cmd\(\)](#), [set_device_needs_reset\(\)](#), and [set_status\(\)](#).

Here is the caller graph for this function:



15.414.3.7 negotiated_features()

```
l4_uint32_t L4virtio::Svr::Dev_config::negotiated_features (
    unsigned idx) const [inline]
```

Compute a specific set of negotiated features.

Parameters

| | |
|------------|---|
| <i>idx</i> | Index into the guest/host features array. |
|------------|---|

Return values

| | |
|------------|--------------------------------------|
| <i>The</i> | selected set of negotiated features. |
|------------|--------------------------------------|

This function returns a specific 32-bit set of features negotiated by the guest/driver and host/device. *idx* is the index in the guest/host features array, resp. the 32-bit set to return.

Definition at line 221 of file [l4virtio](#).

15.414.3.8 qconfig()

```
l4virtio_config_queue_t volatile const * L4virtio::Svr::Dev_config::qconfig (
    unsigned index) const [inline]
```

Get queue read-only config data for queue with the given *index*.

Parameters

| | |
|--------------|-------------------------|
| <i>index</i> | The index of the queue. |
|--------------|-------------------------|

Returns

Read-only pointer to the config of the queue with the given *index*, or NULL if *index* is out of range.

Definition at line 312 of file [l4virtio](#).

References [L4_UNLIKELY](#).

Referenced by [reset_queue\(\)](#).

Here is the caller graph for this function:



15.414.3.9 reset_cmd()

```
void L4virtio::Svr::Dev_config::reset_cmd () [inline]
```

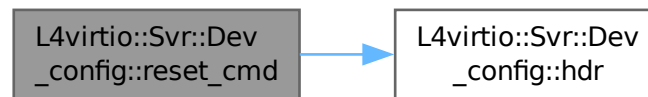
Reset the `cmd` register after execution of a command.

This function resets the `cmd` register in order for the client to detect that the command was executed by the device.

Definition at line 250 of file [l4virtio](#).

References [hdr\(\)](#).

Here is the call graph for this function:

**15.414.3.10 reset_queue()**

```
bool L4virtio::Svr::Dev_config::reset_queue (
    unsigned index,
    unsigned num_max,
    bool inc_generation = false,
    unsigned device_notify_index = 0) const [inline]
```

Reset queue config for the given queue.

Parameters

| | |
|----------------------------|---|
| <i>index</i> | The index of the queue to reset. |
| <i>num_max</i> | The maximum number of descriptors supported by this queue. |
| <i>inc_generation</i> | The config generation will be incremented when this is true. |
| <i>device_notify_index</i> | Device notification Irq index for this queue. See Device_t::device_notify_irq(unsigned) . |

Returns

true on success, or false when *index* is out of range.

Definition at line 356 of file [l4virtio](#).

References [l4virtio_config_queue_t::device_notify_index](#), [L4_UNLIKELY](#), [l4virtio_config_queue_t::num](#), [l4virtio_config_queue_t::num_qconfig\(\)](#), and [l4virtio_config_queue_t::ready](#).

Here is the call graph for this function:



15.414.3.11 `set_device_needs_reset()`

```
void L4virtio::Svr::Dev_config::set_device_needs_reset () [inline]
```

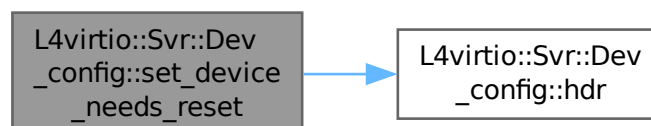
Set `DEVICE_NEEDS_RESET` bit in device status register.

This function sets the internal status register and also the status register in the shared memory to `DEVICE_NEEDS_RESET`.

Definition at line 285 of file [l4virtio](#).

References [hdr\(\)](#).

Here is the call graph for this function:



15.414.3.12 set_device_notify_index()

```
void L4virtio::Svr::Dev_config::set_device_notify_index (
    unsigned idx) [inline]
```

Set index of interrupt that driver should trigger for config notifications.

See [Device_t::device_notify_irq\(unsigned\)](#).

Definition at line 187 of file [l4virtio](#).

15.414.3.13 set_status()

```
void L4virtio::Svr::Dev_config::set_status (
    Status status) [inline]
```

Set device status register.

Parameters

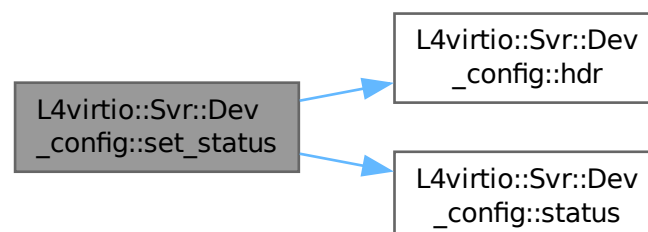
| | |
|---------------|---|
| <i>status</i> | The new value for the device status register. |
|---------------|---|

This function sets the internal status register and also the status register in the shared memory to *status*.

Definition at line 262 of file [l4virtio](#).

References [hdr\(\)](#), and [status\(\)](#).

Here is the call graph for this function:



15.414.3.14 status()

```
Status L4virtio::Svr::Dev_config::status () const [inline]
```

Get current device status (trusted).

Returns

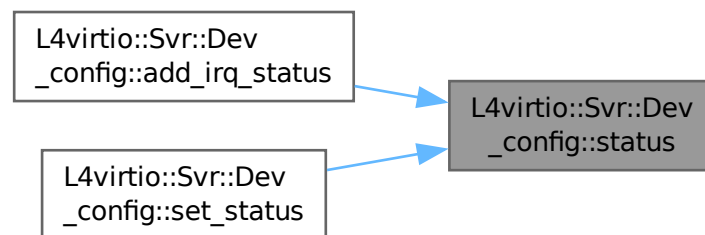
Current device status register (trusted).

The status returned by this function is value stored internally and cannot be written by the guest (i.e., the value can be taken as trusted.)

Definition at line 231 of file [l4virtio](#).

Referenced by [add_irq_status\(\)](#), and [set_status\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

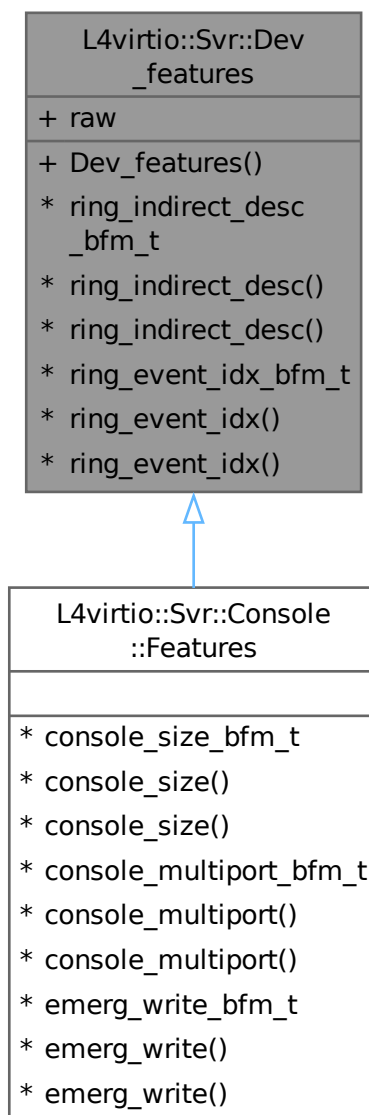
- `l4/l4virtio/server/l4virtio`

15.415 L4virtio::Svr::Dev_features Struct Reference

Type for device feature bitmap.

```
#include <virtio>
```

Inheritance diagram for L4virtio::Svr::Dev_features:



Collaboration diagram for L4virtio::Svr::Dev_features:



Public Member Functions

- **Dev_features** ([l4_uint32_t](#) v)
Make Features from a raw bitmap.

Data Fields

- [l4_uint32_t](#) raw
The raw value of the features bitmap.
- typedef [cxx::Bitfield](#)< decltype(raw), 28, 28 > **ring_indirect_desc_bfm_t**
Type to access the [ring_indirect_desc](#) bits (28 to 28) of raw.
- constexpr [ring_indirect_desc_bfm_t::Val](#) **ring_indirect_desc** () const
Get the [ring_indirect_desc](#) bits (28 to 28) of raw.
- constexpr [ring_indirect_desc_bfm_t::Ref](#) **ring_indirect_desc** ()
Get a reference to the [ring_indirect_desc](#) bits (28 to 28) of raw.
- typedef [cxx::Bitfield](#)< decltype(raw), 29, 29 > **ring_event_idx_bfm_t**
Type to access the [ring_event_idx](#) bits (29 to 29) of raw.
- constexpr [ring_event_idx_bfm_t::Val](#) **ring_event_idx** () const
Get the [ring_event_idx](#) bits (29 to 29) of raw.
- constexpr [ring_event_idx_bfm_t::Ref](#) **ring_event_idx** ()
Get a reference to the [ring_event_idx](#) bits (29 to 29) of raw.

15.415.1 Detailed Description

Type for device feature bitmap.

Definition at line 66 of file [virtio](#).

The documentation for this struct was generated from the following file:

- l4/l4virtio/server/virtio

15.416 L4virtio::Svr::Dev_status Struct Reference

Type of the device status register.

```
#include <virtio>
```

Collaboration diagram for L4virtio::Svr::Dev_status:



Public Member Functions

- **Dev_status** ([l4_uint32_t](#) v)
Make Status from raw value.
- bool [running](#) () const
Check if the device is in running state.

Data Fields

- unsigned char **raw**
Raw value of the VIRTIO device status register.
- typedef `cxx::Bitfield< decltype(raw), 0, 0 >` **acked_bfm_t**
Type to access the `acked` bits (0 to 0) of `raw`.
- constexpr `acked_bfm_t::Val` **acked** () const
Get the `acked` bits (0 to 0) of `raw`.
- constexpr `acked_bfm_t::Ref` **acked** ()
Get a reference to the `acked` bits (0 to 0) of `raw`.
- typedef `cxx::Bitfield< decltype(raw), 1, 1 >` **driver_bfm_t**
Type to access the `driver` bits (1 to 1) of `raw`.
- constexpr `driver_bfm_t::Val` **driver** () const
Get the `driver` bits (1 to 1) of `raw`.
- constexpr `driver_bfm_t::Ref` **driver** ()
Get a reference to the `driver` bits (1 to 1) of `raw`.
- typedef `cxx::Bitfield< decltype(raw), 2, 2 >` **driver_ok_bfm_t**
Type to access the `driver_ok` bits (2 to 2) of `raw`.
- constexpr `driver_ok_bfm_t::Val` **driver_ok** () const
Get the `driver_ok` bits (2 to 2) of `raw`.
- constexpr `driver_ok_bfm_t::Ref` **driver_ok** ()
Get a reference to the `driver_ok` bits (2 to 2) of `raw`.
- typedef `cxx::Bitfield< decltype(raw), 3, 3 >` **features_ok_bfm_t**
Type to access the `features_ok` bits (3 to 3) of `raw`.
- constexpr `features_ok_bfm_t::Val` **features_ok** () const
Get the `features_ok` bits (3 to 3) of `raw`.
- constexpr `features_ok_bfm_t::Ref` **features_ok** ()
Get a reference to the `features_ok` bits (3 to 3) of `raw`.
- typedef `cxx::Bitfield< decltype(raw), 6, 7 >` **fail_state_bfm_t**
Type to access the `fail_state` bits (6 to 7) of `raw`.
- constexpr `fail_state_bfm_t::Val` **fail_state** () const
Get the `fail_state` bits (6 to 7) of `raw`.
- constexpr `fail_state_bfm_t::Ref` **fail_state** ()
Get a reference to the `fail_state` bits (6 to 7) of `raw`.
- typedef `cxx::Bitfield< decltype(raw), 6, 6 >` **device_needs_reset_bfm_t**
Type to access the `device_needs_reset` bits (6 to 6) of `raw`.
- constexpr `device_needs_reset_bfm_t::Val` **device_needs_reset** () const
Get the `device_needs_reset` bits (6 to 6) of `raw`.
- constexpr `device_needs_reset_bfm_t::Ref` **device_needs_reset** ()
Get a reference to the `device_needs_reset` bits (6 to 6) of `raw`.
- typedef `cxx::Bitfield< decltype(raw), 7, 7 >` **failed_bfm_t**
Type to access the `failed` bits (7 to 7) of `raw`.
- constexpr `failed_bfm_t::Val` **failed** () const
Get the `failed` bits (7 to 7) of `raw`.
- constexpr `failed_bfm_t::Ref` **failed** ()
Get a reference to the `failed` bits (7 to 7) of `raw`.

15.416.1 Detailed Description

Type of the device status register.

Definition at line 32 of file [virtio](#).

15.416.2 Member Function Documentation

15.416.2.1 `running()`

```
bool L4virtio::Svr::Dev_status::running () const [inline]
```

Check if the device is in running state.

Returns

true if the device is in running state.

The device is in running state when [acked\(\)](#), [driver\(\)](#), [features_ok\(\)](#), and [driver_ok\(\)](#) return true, and [device_needs_reset\(\)](#) and [failed\(\)](#) return false.

Definition at line 57 of file [virtio](#).

References [raw](#).

The documentation for this struct was generated from the following file:

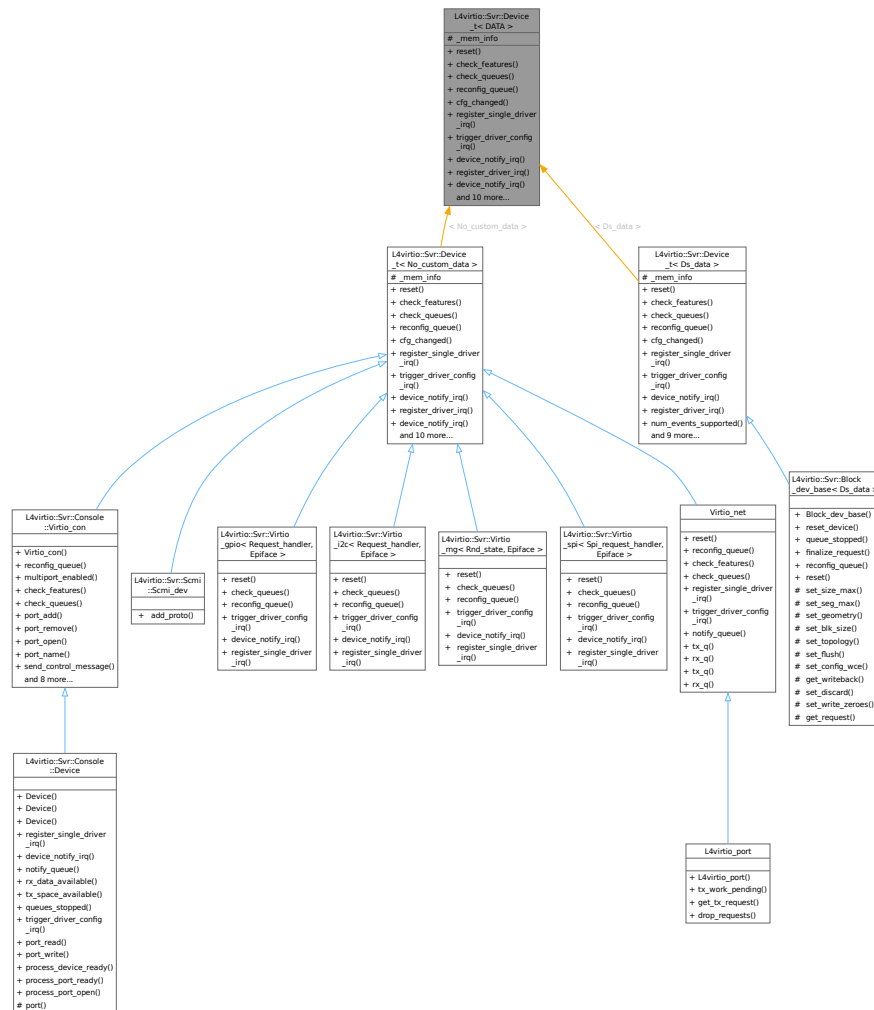
- `l4/l4virtio/server/virtio`

15.417 `L4virtio::Svr::Device_t< DATA >` Class Template Reference

Server-side L4-VIRTIO device stub.

```
#include <l4virtio>
```

Inheritance diagram for L4virtio::Svr::Device_t< DATA >:



Collaboration diagram for L4virtio::Svr::Device_t< DATA >:

| L4virtio::Svr::Device _t< DATA > |
|--|
| # _mem_info |
| + reset() + check_features() + check_queues() + reconfig_queue() + cfg_changed() + register_single_driver_irq() + trigger_driver_config_irq() + device_notify_irq() + register_driver_irq() + device_notify_irq() and 10 more... |

Public Member Functions

- virtual void **reset** ()=0
reset callback, called for doing a device reset
- virtual bool **check_features** ()
callback for checking the subset of accepted features
- virtual bool **check_queues** ()=0
callback for checking if the queues at DRIVER_OK transition
- virtual int **reconfig_queue** (unsigned idx)=0
callback for client queue-config request
- virtual void **cfg_changed** (unsigned)
callback for client device configuration changes
- virtual void **register_single_driver_irq** ()
callback for registering a single guest IRQ for all queues (old-style)
- virtual void **trigger_driver_config_irq** ()=0
callback for triggering configuration change notification IRQ
- virtual L4::Cap< L4::Irq > **device_notify_irq** () const
callback to gather the device notification IRQ (old-style)
- virtual void **register_driver_irq** (unsigned idx)
Callback for registering an notification IRQ (multi IRQ).
- virtual L4::Cap< L4::Irq > **device_notify_irq** (unsigned idx)
Callback to gather the device notification IRQ (multi IRQ).

- virtual unsigned **num_events_supported** () const
Return the highest notification index supported.
- **Device_t** (**Dev_config** *dev_config)
Make a device for the given config.
- **Mem_list** const * **mem_info** () const
Get the memory region list used for this device.
- void **reset_queue_config** (unsigned idx, unsigned num_max, bool inc_generation=false, unsigned device_notify_index=0)
Trigger reset for the configuration space for queue idx.
- void **init_mem_info** (unsigned num)
Initialize the memory region list to the given maximum.
- void **device_error** ()
Transition device into DEVICE_NEEDS_RESET state.
- bool **setup_queue** (Virtqueue *q, unsigned qn, unsigned num_max)
Enable/disable the specified queue.
- bool **handle_mem_cmd_write** ()
Check for a value in the cmd register and handle a write.
- void **enable_trusted_ds_validation** ()
Enable trusted dataspace validation.
- void **add_trusted_dataspaces** (std::shared_ptr< Ds_vector const > ds)
Provide a list of trusted dataspaces that can be used for validation.

Protected Attributes

- **Mem_list _mem_info**
Memory region list.

15.417.1 Detailed Description

```
template<typename DATA>
class L4virtio::Svr::Device_t< DATA >
```

Server-side L4-VIRTIO device stub.

This stub supports new-style multi-event registration (using `get_device_config()`, `bind()` and `get_device_notification_irq()`).

Definition at line 814 of file [l4virtio](#).

15.417.2 Member Function Documentation

15.417.2.1 add_trusted_dataspaces()

```
template<typename DATA>
void L4virtio::Svr::Device_t< DATA >::add_trusted_dataspaces (
    std::shared_ptr< Ds_vector const > ds) [inline]
```

Provide a list of trusted dataspaces that can be used for validation.

Parameters

| | |
|-----------|-----------------------------|
| <i>ds</i> | list of trusted dataspaces. |
|-----------|-----------------------------|

Definition at line 1217 of file [l4virtio](#).

15.417.2.2 device_error()

```
template<typename DATA>
void L4virtio::Svr::Device_t< DATA >::device_error () [inline]
```

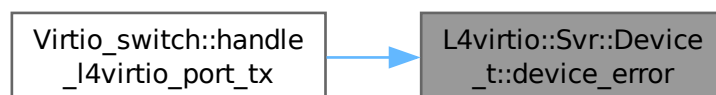
Transition device into DEVICE_NEEDS_RESET state.

This function does a full reset, sets the DEVICE_NEEDS_RESET bit in the device status register, triggering a guest config IRQ if necessary. The driver still needs to perform its own reset and initialization sequence.

Definition at line 1043 of file [l4virtio](#).

Referenced by [Virtio_switch::handle_l4virtio_port_tx\(\)](#).

Here is the caller graph for this function:



15.417.2.3 device_notify_irq()

```
template<typename DATA>
virtual L4::Cap< L4::Irq > L4virtio::Svr::Device_t< DATA >::device_notify_irq (
    unsigned idx) [inline], [virtual]
```

Callback to gather the device notification IRQ (multi IRQ).

The default implementation maps to the implementation for single IRQ notification points.

Definition at line 887 of file [l4virtio](#).

15.417.2.4 handle_mem_cmd_write()

```
template<typename DATA>
bool L4virtio::Svr::Device_t< DATA >::handle_mem_cmd_write () [inline]
```

Check for a value in the `cmd` register and handle a write.

This function checks for a value in the `cmd` register and executes the command if there is any, or returns false if there was no command.

Execution of the command is signaled by a zero in the `cmd` register.

Definition at line 1173 of file [l4virtio](#).

15.417.2.5 init_mem_info()

```
template<typename DATA>
void L4virtio::Svr::Device_t< DATA >::init_mem_info (
    unsigned num) [inline]
```

Initialize the memory region list to the given maximum.

Parameters

| | |
|------------|---|
| <i>num</i> | Maximum number of memory regions that can be managed. |
|------------|---|

Definition at line 1031 of file [l4virtio](#).

15.417.2.6 register_driver_irq()

```
template<typename DATA>
virtual void L4virtio::Svr::Device_t< DATA >::register_driver_irq (
    unsigned idx) [inline], [virtual]
```

Callback for registering an notification IRQ (multi IRQ).

The default implementation maps to the implementation for single IRQ notification points.

Definition at line 873 of file [l4virtio](#).

15.417.2.7 reset_queue_config()

```
template<typename DATA>
void L4virtio::Svr::Device_t< DATA >::reset_queue_config (
    unsigned idx,
    unsigned num_max,
    bool inc_generation = false,
    unsigned device_notify_index = 0) [inline]
```

Trigger reset for the configuration space for queue *idx*.

Parameters

| | |
|----------------------------|---|
| <i>idx</i> | The queue index to reset. |
| <i>num_max</i> | Maximum number of entries in this queue. |
| <i>inc_generation</i> | The config generation will be incremented when this is true. |
| <i>device_notify_index</i> | Device notification Irq index for this queue. See device_notify_irq(unsigned) . |

This function resets the driver-readable configuration space for the queue with the given index. The queue configuration is reset to all 0, and the maximum number of entries in the queue is set to *num_max*.

Definition at line [1019](#) of file [l4virtio](#).

15.417.2.8 setup_queue()

```
template<typename DATA>
bool L4virtio::Svr::Device_t< DATA >::setup_queue (
    Virtqueue * q,
    unsigned qn,
    unsigned num_max) [inline]
```

Enable/disable the specified queue.

Parameters

| | |
|----------------|---|
| <i>q</i> | Pointer to the ring that represents the virtqueue internally. |
| <i>qn</i> | Index of the queue. |
| <i>num_max</i> | Maximum number of supported entries in this queue. |

Returns

true for success.

- This function calculates the parameters of the virtqueue from the clients configuration space values, checks the accessibility of the queue data structures and initializes *q* to ready state when all checks succeeded.

Definition at line [1066](#) of file [l4virtio](#).

The documentation for this class was generated from the following file:

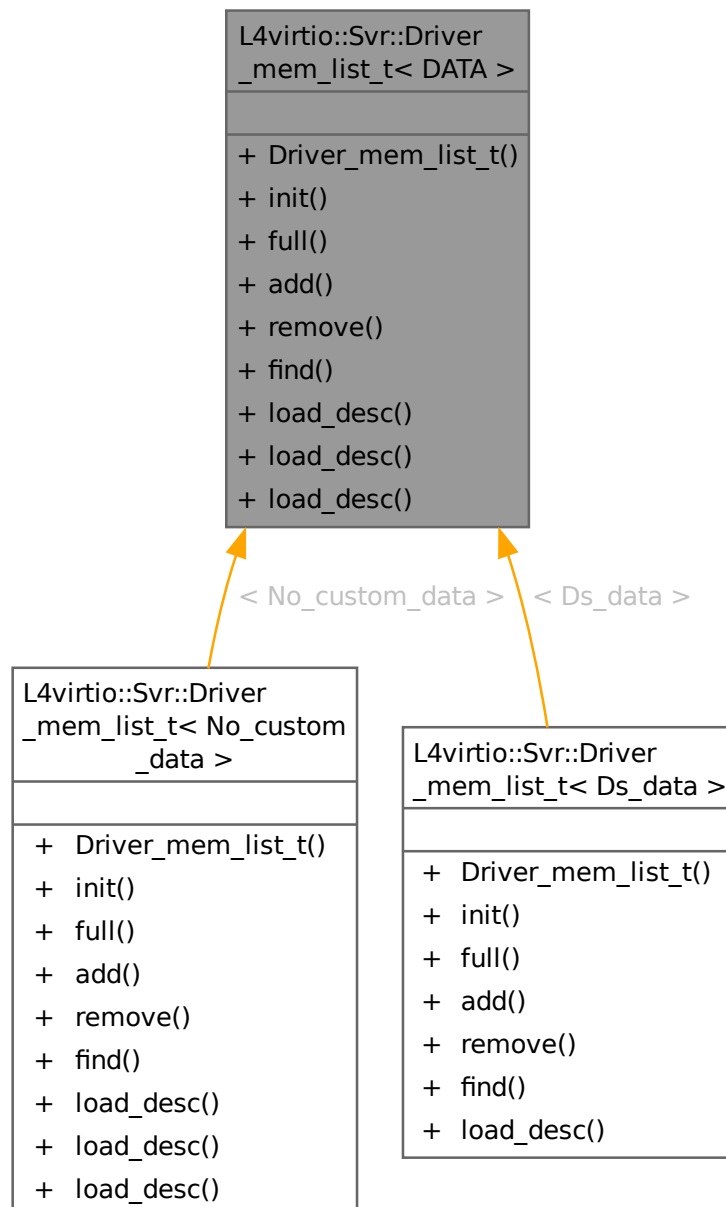
- [l4/l4virtio/server/l4virtio](#)

15.418 L4virtio::Svr::Driver_mem_list_t< DATA > Class Template Reference

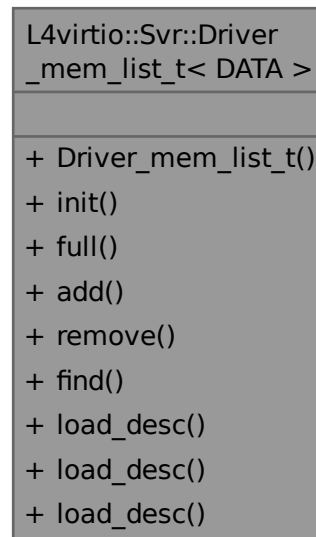
List of driver memory regions assigned to a single L4-VIRTIO transport instance.

```
#include <l4virtio>
```

Inheritance diagram for L4virtio::Svr::Driver_mem_list_t< DATA >:



Collaboration diagram for L4virtio::Svr::Driver_mem_list_t< DATA >:



Public Types

- typedef [L4Re::Util::Unique_cap](#)< [L4Re::Dataspace](#) > **Ds_cap**
type for storing a data-space capability internally

Public Member Functions

- **Driver_mem_list_t ()**
Make an empty, zero capacity list.
- void **init** (unsigned max)
Make a fresh list with capacity max.
- bool **full** () const
- Mem_region const * **add** (l4_uint64_t drv_base, l4_umword_t size, l4_addr_t offset, [Ds_cap](#) &&ds)
Add a new region to the list.
- void **remove** (Mem_region const *r)
Remove the given region from the list.
- Mem_region * **find** (l4_uint64_t base, l4_umword_t size) const
Find memory region containing the given driver address region.
- void **load_desc** ([Virtqueue::Desc](#) const &desc, [Request_processor](#) const *p, [Virtqueue::Desc](#) const **table) const
Default implementation for loading an indirect descriptor.
- void **load_desc** ([Virtqueue::Desc](#) const &desc, [Request_processor](#) const *p, Mem_region const **data) const
Default implementation returning the Driver_mem_region.
- template<typename ARG>
void **load_desc** ([Virtqueue::Desc](#) const &desc, [Request_processor](#) const *p, ARG *data) const
Default implementation returning generic information.

15.418.1 Detailed Description

```
template<typename DATA>
class L4virtio::Svr::Driver_mem_list_t< DATA >
```

List of driver memory regions assigned to a single L4-VIRTIO transport instance.

Note

The regions added to this list *must* never overlap.

Definition at line 642 of file [l4virtio](#).

15.418.2 Member Function Documentation

15.418.2.1 add()

```
template<typename DATA>
Mem_region const * L4virtio::Svr::Driver_mem_list_t< DATA >::add (
    l4_uint64_t drv_base,
    l4_umword_t size,
    l4_addr_t offset,
    Ds_cap && ds) [inline]
```

Add a new region to the list.

Parameters

| | |
|-----------------|--|
| <i>drv_base</i> | Driver base address of the region. |
| <i>size</i> | Size of the region in bytes. |
| <i>offset</i> | Offset within the data space attached to <i>drv_base</i> . |
| <i>ds</i> | Data space backing the driver memory. |

Returns

A pointer to the new region.

Definition at line 682 of file [l4virtio](#).

15.418.2.2 find()

```
template<typename DATA>
Mem_region * L4virtio::Svr::Driver_mem_list_t< DATA >::find (
    l4_uint64_t base,
    l4_umword_t size) const [inline]
```

Find memory region containing the given driver address region.

Parameters

| | |
|-------------|----------------------|
| <i>base</i> | Driver base address. |
| <i>size</i> | Size of the region. |

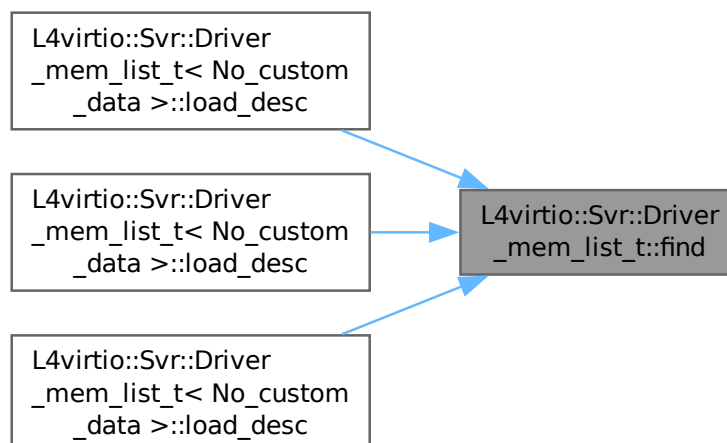
Returns

Pointer to the region containing the given region, NULL if none is found.

Definition at line 716 of file [l4virtio](#).

Referenced by [L4virtio::Svr::Driver_mem_list_t< No_custom_data >::load_desc\(\)](#), [L4virtio::Svr::Driver_mem_list_t< No_custom_data >::load_desc\(\)](#) and [L4virtio::Svr::Driver_mem_list_t< No_custom_data >::load_desc\(\)](#).

Here is the caller graph for this function:



15.418.2.3 full()

```
template<typename DATA>
bool L4virtio::Svr::Driver_mem_list_t< DATA >::full () const [inline]
```

Returns

True if the remaining capacity is 0.

Definition at line 671 of file [l4virtio](#).

Referenced by [L4virtio::Svr::Driver_mem_list_t< No_custom_data >::add\(\)](#).

Here is the caller graph for this function:

**15.418.2.4 init()**

```
template<typename DATA>
void L4virtio::Svr::Driver_mem_list_t< DATA >::init (
    unsigned max) [inline]
```

Make a fresh list with capacity *max*.

Parameters

| | |
|------------|------------------------------|
| <i>max</i> | The capacity of this vector. |
|------------|------------------------------|

Definition at line 663 of file [l4virtio](#).

15.418.2.5 load_desc() [1/3]

```
template<typename DATA>
template<typename ARG>
void L4virtio::Svr::Driver_mem_list_t< DATA >::load_desc (
    Virtqueue::Desc const & desc,
    Request_processor const * p,
    ARG * data) const [inline]
```

Default implementation returning generic information.

Template Parameters

| | |
|------------|--|
| <i>ARG</i> | Abstract argument type used with Request_processor::start() and Request_processor::next() to deliver the result of loading a descriptor. This type must provide a constructor taking three arguments: (1) pointer to a <code>Driver_mem_region</code> , (2) the Virtqueue::Desc descriptor, and (3) a pointer to the calling Request_processor . |
|------------|--|

Parameters

| | | |
|-----|-------------|---|
| | <i>desc</i> | The descriptor to load |
| | <i>p</i> | The request processor calling us |
| out | <i>data</i> | Shall be assigned to <code>ARG(mem, desc, p)</code> |

Exceptions

| | |
|--------------------------------|---|
| Bad_descriptor | The descriptor address could not be translated. |
|--------------------------------|---|

Definition at line 777 of file [l4virtio](#).

15.418.2.6 load_desc() [2/3]

```
template<typename DATA>
void L4virtio::Svr::Driver_mem_list_t< DATA >::load_desc (
    Virtqueue::Desc const & desc,
    Request_processor const * p,
    Mem_region const ** data) const [inline]
```

Default implementation returning the `Driver_mem_region`.

Parameters

| | | |
|-----|-------------|---|
| | <i>desc</i> | The descriptor to load |
| | <i>p</i> | The request processor calling us |
| out | <i>data</i> | Shall be set to a pointer to the <code>Driver_mem_region</code> that covers the descriptor. |

Exceptions

| | |
|--------------------------------|---|
| Bad_descriptor | The descriptor address could not be translated. |
|--------------------------------|---|

Definition at line 750 of file [l4virtio](#).

15.418.2.7 load_desc() [3/3]

```
template<typename DATA>
void L4virtio::Svr::Driver_mem_list_t< DATA >::load_desc (
    Virtqueue::Desc const & desc,
```

```
Request_processor const * p,
Virtqueue::Desc const ** table) const [inline]
```

Default implementation for loading an indirect descriptor.

Parameters

| | | |
|-----|--------------|---|
| | <i>desc</i> | The descriptor to load |
| | <i>p</i> | The request processor calling us |
| out | <i>table</i> | Shall be set to the loaded descriptor table |

Exceptions

| | |
|---------------------------------------|---|
| <i>Bad_descriptor</i> | The descriptor address could not be translated. |
|---------------------------------------|---|

Definition at line 730 of file [l4virtio](#).

15.418.2.8 remove()

```
template<typename DATA>
void L4virtio::Svr::Driver_mem_list_t< DATA >::remove (
    Mem_region const * r) [inline]
```

Remove the given region from the list.

Parameters

| | |
|----------|--|
| <i>r</i> | The region to remove (result from add() , or find()). |
|----------|--|

Definition at line 696 of file [l4virtio](#).

The documentation for this class was generated from the following file:

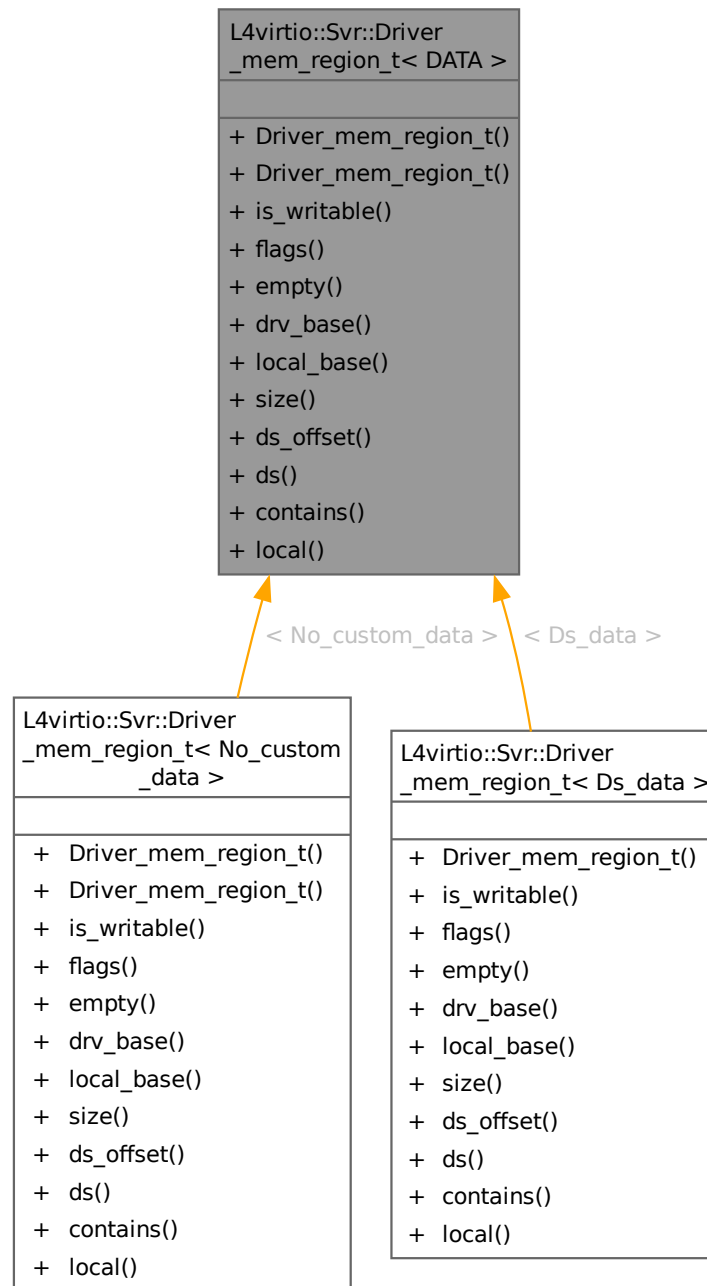
- [l4/l4virtio/server/l4virtio](#)

15.419 L4virtio::Svr::Driver_mem_region_t< DATA > Class Template Reference

Region of driver memory, that shall be managed locally.

```
#include <l4virtio>
```

Inheritance diagram for L4virtio::Svr::Driver_mem_region_t< DATA >:



Collaboration diagram for L4virtio::Svr::Driver_mem_region_t< DATA >:

| L4virtio::Svr::Driver_mem_region_t< DATA > |
|---|
| <ul style="list-style-type: none"> + Driver_mem_region_t() + Driver_mem_region_t() + is_writable() + flags() + empty() + drv_base() + local_base() + size() + ds_offset() + ds() + contains() + local() |

Public Member Functions

- **Driver_mem_region_t ()**
Make default empty memory region.
- **Driver_mem_region_t (l4_uint64_t drv_base, l4_umword_t size, l4_addr_t offset, Ds_cap &&ds)**
Make a local memory region for the given driver values.
- bool **is_writable ()** const
- Flags **flags ()** const
- bool **empty ()** const
- **l4_uint64_t drv_base ()** const
- **l4_addr_t local_base ()** const
- **l4_umword_t size ()** const
- **l4_addr_t ds_offset ()** const
- **L4::Cap< L4Re::Dataspace > ds ()** const
- bool **contains (l4_uint64_t base, l4_umword_t size)** const
Test if the given driver address range is within this region.
- template<typename T>
T * **local (Ptr< T > p)** const
Get the local address for driver address p.

15.419.1 Detailed Description

```
template<typename DATA>
class L4virtio::Svr::Driver_mem_region_t< DATA >
```

Region of driver memory, that shall be managed locally.

Template Parameters

| | |
|-------------|---------------------------------------|
| <i>DATA</i> | Class defining additional information |
|-------------|---------------------------------------|

Definition at line 463 of file [l4virtio](#).

15.419.2 Constructor & Destructor Documentation

15.419.2.1 Driver_mem_region_t()

```
template<typename DATA>
L4virtio::Svr::Driver_mem_region_t< DATA >::Driver_mem_region_t (
    l4_uint64_t drv_base,
    l4_umword_t size,
    l4_addr_t offset,
    Ds_cap && ds) [inline]
```

Make a local memory region for the given driver values.

Parameters

| | |
|-----------------|---|
| <i>drv_base</i> | Base address of the memory region used by the driver. |
| <i>size</i> | Size of the memory region. |
| <i>offset</i> | Offset within the data space that is mapped to <i>drv_base</i> within the driver. |
| <i>ds</i> | Data space capability backing the memory. |

This constructor attaches the region of given data space to the local address space and stores the corresponding data for later reference.

Definition at line 511 of file [l4virtio](#).

15.419.3 Member Function Documentation

15.419.3.1 contains()

```
template<typename DATA>
bool L4virtio::Svr::Driver_mem_region_t< DATA >::contains (
    l4_uint64_t base,
    l4_umword_t size) const [inline]
```

Test if the given driver address range is within this region.

Parameters

| | |
|-------------|-----------------------------------|
| <i>base</i> | The driver base address. |
| <i>size</i> | The size of the region to lookup. |

Returns

true if the given driver address region is contained in this region, false else.

Definition at line 605 of file [l4virtio](#).

15.419.3.2 drv_base()

```
template<typename DATA>
l4_uint64_t L4virtio::Svr::Driver_mem_region_t< DATA >::drv_base () const [inline]
```

Returns

The base address used by the driver.

Definition at line 584 of file [l4virtio](#).

15.419.3.3 ds()

```
template<typename DATA>
L4::Cap< L4Re::Dataspace > L4virtio::Svr::Driver_mem_region_t< DATA >::ds () const [inline]
```

Returns

The data space capability for this region.

Definition at line 596 of file [l4virtio](#).

15.419.3.4 ds_offset()

```
template<typename DATA>
l4_addr_t L4virtio::Svr::Driver_mem_region_t< DATA >::ds_offset () const [inline]
```

Returns

The offset within the data space.

Definition at line 593 of file [l4virtio](#).

15.419.3.5 empty()

```
template<typename DATA>
bool L4virtio::Svr::Driver_mem_region_t< DATA >::empty () const [inline]
```

Returns

True if the region is empty (size == 0), false otherwise.

Definition at line 580 of file [l4virtio](#).

15.419.3.6 flags()

```
template<typename DATA>
Flags L4virtio::Svr::Driver_mem_region_t< DATA >::flags () const [inline]
```

Returns

The flags for this region.

Definition at line 577 of file [l4virtio](#).

15.419.3.7 is_writable()

```
template<typename DATA>
bool L4virtio::Svr::Driver_mem_region_t< DATA >::is_writable () const [inline]
```

Returns

True if the region is writable, false otherwise.

Definition at line 574 of file [l4virtio](#).

15.419.3.8 local()

```
template<typename DATA>
template<typename T>
T * L4virtio::Svr::Driver_mem_region_t< DATA >::local (
    Ptr< T > p) const [inline]
```

Get the local address for driver address *p*.

Parameters

| | |
|----------|------------------------------|
| <i>p</i> | Driver address to translate. |
|----------|------------------------------|

Precondition

p must be contained in this region.

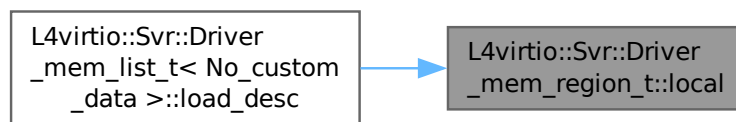
Returns

Local address for the given driver address *p*.

Definition at line 629 of file [l4virtio](#).

Referenced by [L4virtio::Svr::Driver_mem_list_t< No_custom_data >::load_desc\(\)](#).

Here is the caller graph for this function:

**15.419.3.9 local_base()**

```
template<typename DATA>
l4_addr_t L4virtio::Svr::Driver_mem_region_t< DATA >::local_base () const [inline]
```

Returns

The local base address.

Definition at line 587 of file [l4virtio](#).

15.419.3.10 size()

```
template<typename DATA>
l4_umword_t L4virtio::Svr::Driver_mem_region_t< DATA >::size () const [inline]
```

Returns

The size of the region in bytes.

Definition at line 590 of file [l4virtio](#).

The documentation for this class was generated from the following file:

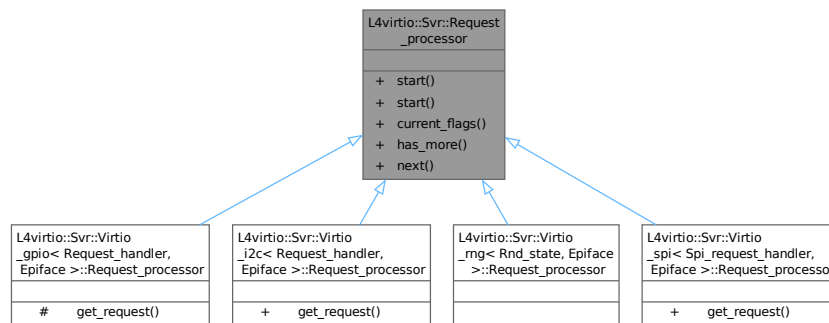
- [l4/l4virtio/server/l4virtio](#)

15.420 L4virtio::Svr::Request_processor Class Reference

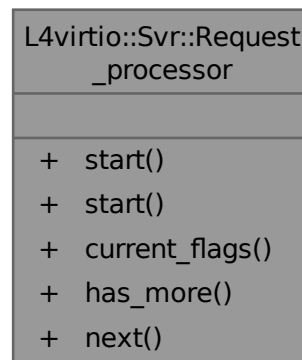
Encapsulate the state for processing a VIRTIO request.

```
#include <virtio>
```

Inheritance diagram for L4virtio::Svr::Request_processor:



Collaboration diagram for L4virtio::Svr::Request_processor:



Public Member Functions

- `template<typename DESC_MAN, typename ... ARGS>`
`void start (DESC_MAN *dm, Virtqueue *ring, Virtqueue::Head_desc const &request, ARGS... args)`
Start processing a new request.
- `template<typename DESC_MAN, typename ... ARGS>`
`Virtqueue::Request const & start (DESC_MAN *dm, Virtqueue::Request const &request, ARGS... args)`
Start processing a new request.
- `Virtqueue::Desc::Flags current_flags () const`
Get the flags of the currently processed descriptor.

- bool [has_more](#) () const
Are there more chained descriptors?
- template<typename DESC_MAN, typename ... ARGS>
bool [next](#) (DESC_MAN *dm, ARGS... args)
Switch to the next descriptor in a descriptor chain.

15.420.1 Detailed Description

Encapsulate the state for processing a VIRTIO request.

A VIRTIO request is a possibly chained list of descriptors retrieved from the available ring of a virtqueue, using [Virtqueue::next_avail\(\)](#).

The descriptor processing depends on helper (DESC_MAN) for interpreting the descriptors in the context of the device implementation.

DESC_MAN has to provide the functionality to safely dereference a descriptor from a descriptor list.

The following methods must be provided by DESC_MAN:

- DESC_MAN::load_desc (Virtqueue::Desc const &desc,
Request_processor const *proc,
Virtqueue::Desc const **table)

This function is used to dereference `desc` as an indirect descriptor table, and must return a pointer to an indirect descriptor table.

- DESC_MAN::load_desc (Virtqueue::Desc const &desc,
Request_processor const *proc, ...)

This function is used to dereference a descriptor as a normal data buffer, and '...' are the arguments that are passed to [start\(\)](#) and [next\(\)](#).

Definition at line [472](#) of file [virtio](#).

15.420.2 Member Function Documentation

15.420.2.1 current_flags()

```
Virtqueue::Desc::Flags L4virtio::Svr::Request_processor::current_flags () const [inline]
```

Get the flags of the currently processed descriptor.

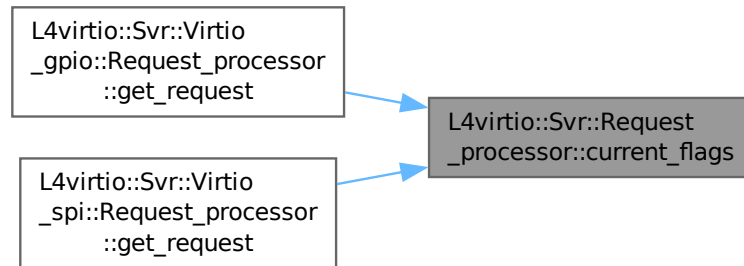
Returns

The flags of the currently processed descriptor.

Definition at line 545 of file [virtio](#).

Referenced by [L4virtio::Svr::Virtio_gpio< Request_handler, Epiface >::Request_processor::get_request\(\)](#), and [L4virtio::Svr::Virtio_spi< Spi_request_handler, Epiface >::Request_processor::get_request\(\)](#).

Here is the caller graph for this function:

**15.420.2.2 has_more()**

```
bool L4virtio::Svr::Request_processor::has_more () const [inline]
```

Are there more chained descriptors?

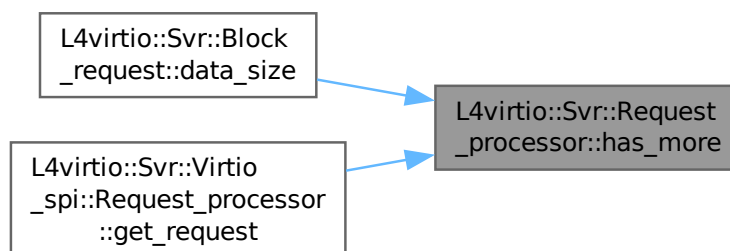
Returns

true if there are more chained descriptors in the current request.

Definition at line 553 of file [virtio](#).

Referenced by [L4virtio::Svr::Block_request< Ds_data >::data_size\(\)](#), and [L4virtio::Svr::Virtio_spi< Spi_request_handler, Epiface >::Request_processor::get_request\(\)](#).

Here is the caller graph for this function:



15.420.2.3 next()

```
template<typename DESC_MAN, typename ... ARGS>
bool L4virtio::Svr::Request_processor::next (
    DESC_MAN * dm,
    ARGS... args) [inline]
```

Switch to the next descriptor in a descriptor chain.

Template Parameters

| | |
|-----------------|--|
| <i>DESC_MAN</i> | Type of descriptor manager (implicit). |
|-----------------|--|

Parameters

| | |
|-------------|---|
| <i>dm</i> | Descriptor manager that is used to translate VIRTIO descriptor addresses. |
| <i>args</i> | Extra arguments passed to dm->load_desc() |

Return values

| | |
|--------------|---------------------------------|
| <i>true</i> | A next descriptor is available. |
| <i>false</i> | No descriptor available. |

Exceptions

| | |
|---------------------------------------|---|
| <i>Bad_descriptor</i> | The next index of this descriptor is invalid. |
|---------------------------------------|---|

Definition at line 570 of file [virtio](#).

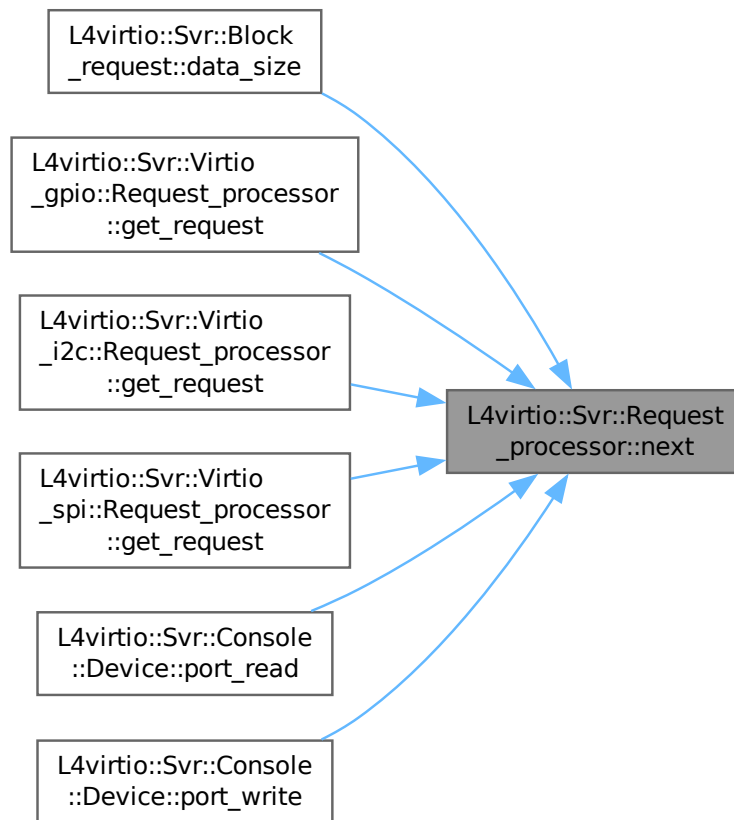
References [cxx::access_once\(\)](#), [L4virtio::Svr::Bad_descriptor::Bad_flags](#), [L4virtio::Svr::Bad_descriptor::Bad_next](#), and [L4_UNLIKELY](#).

Referenced by [L4virtio::Svr::Block_request<Ds_data>::data_size\(\)](#), [L4virtio::Svr::Virtio_gpio<Request_handler, Epiface>::Request_handler::get_request\(\)](#), [L4virtio::Svr::Virtio_i2c<Request_handler, Epiface>::Request_processor::get_request\(\)](#), [L4virtio::Svr::Virtio_spi<Spi_request_handler, Epiface>::Request_processor::get_request\(\)](#), [L4virtio::Svr::Console::Device::port_read\(\)](#), and [L4virtio::Svr::Console::Device::port_write\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.420.2.4 start() [1/2]

```

template<typename DESC_MAN, typename ... ARGS>
void L4virtio::Svr::Request_processor::start (
    DESC_MAN * dm,
    Virtqueue * ring,
    Virtqueue::Head_desc const & request,
    ARGS... args) [inline]
  
```

Start processing a new request.

Template Parameters

| | |
|-----------------------|--|
| <code>DESC_MAN</code> | Type of descriptor manager (implicit). |
|-----------------------|--|

Parameters

| | |
|----------------|---|
| <i>dm</i> | Descriptor manager that is used to translate VIRTIO descriptor addresses. |
| <i>ring</i> | VIRTIO ring of the request. |
| <i>request</i> | VIRTIO request from Virtqueue::next_avail() |
| <i>args</i> | Extra arguments passed to dm->load_desc() |

Precondition

The given request must be valid.

Exceptions

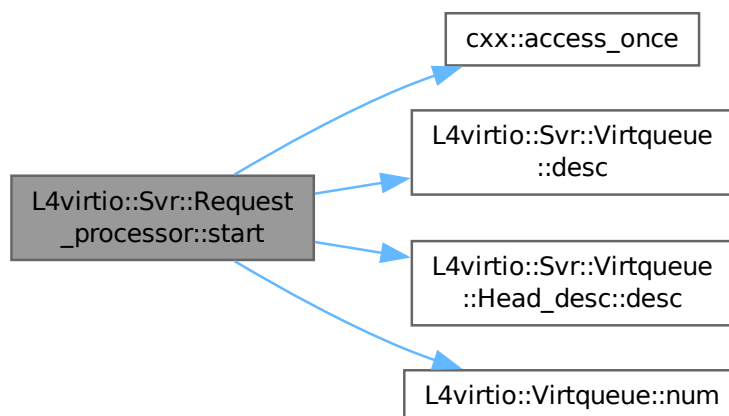
| | |
|--------------------------------|--|
| Bad_descriptor | The descriptor has an invalid size or load_desc() has thrown an exception by itself. |
|--------------------------------|--|

Definition at line 501 of file [virtio](#).

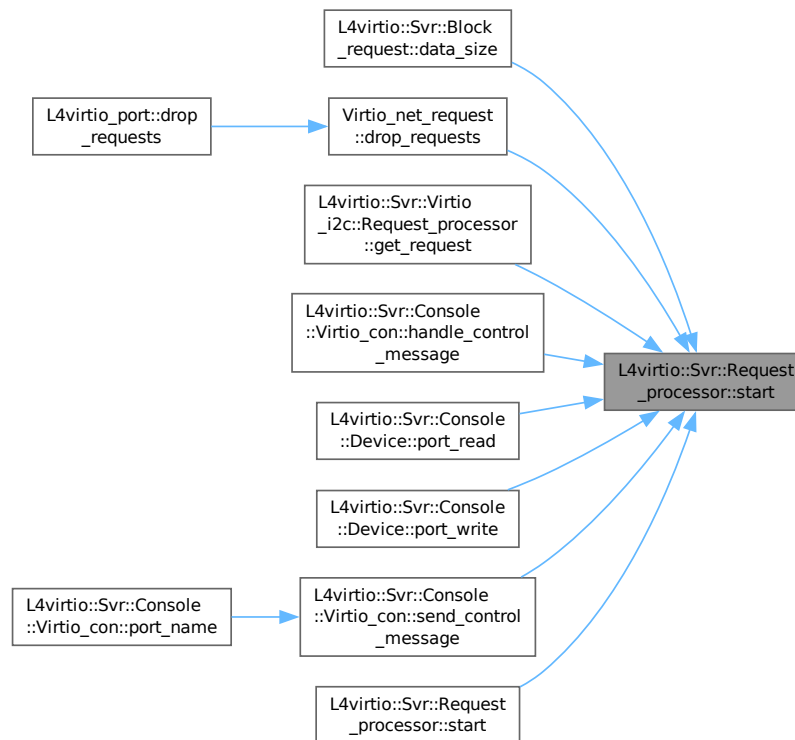
References [cxx::access_once\(\)](#), [L4virtio::Svr::Bad_descriptor::Bad_size](#), [L4virtio::Svr::Virtqueue::desc\(\)](#), [L4virtio::Svr::Virtqueue::Head_desc::desc\(\)](#), [L4_UNLIKELY](#), and [L4virtio::Virtqueue::num\(\)](#).

Referenced by [L4virtio::Svr::Block_request<Ds_data>::data_size\(\)](#), [Virtio_net_request::drop_requests\(\)](#), [L4virtio::Svr::Virtio_i2c<Request_handler, Epiface>::Request_processor::get_request\(\)](#), [L4virtio::Svr::Console::Virtio_con::handle_request\(\)](#), [L4virtio::Svr::Console::Device::port_read\(\)](#), [L4virtio::Svr::Console::Device::port_write\(\)](#), [L4virtio::Svr::Console::Virtio_con::send_control_data\(\)](#), and [start\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.420.2.5 start() [2/2]

```

template<typename DESC_MAN, typename ... ARGS>
Virtqueue::Request const & L4virtio::Svr::Request_processor::start (
    DESC_MAN * dm,
    Virtqueue::Request const & request,
    ARGS... args) [inline]

```

Start processing a new request.

Template Parameters

| | |
|-----------------------|--|
| <code>DESC_MAN</code> | Type of descriptor manager (implicit). |
|-----------------------|--|

Parameters

| | |
|----------------------|---|
| <code>dm</code> | Descriptor manager that is used to translate VIRTIO descriptor addresses. |
| <code>request</code> | VIRTIO request from Virtqueue::next_avail() |
| <code>args</code> | Extra arguments passed to <code>dm->load_desc()</code> |

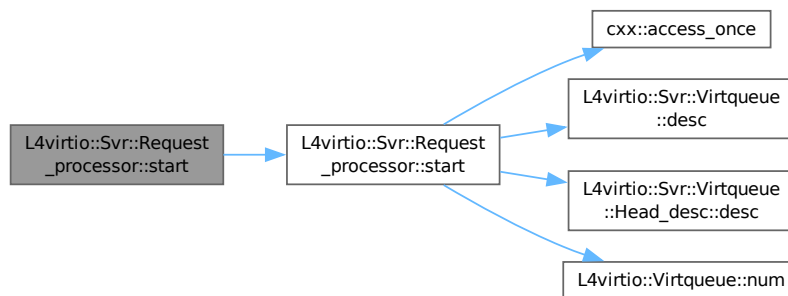
Precondition

The given request must be valid.

Definition at line 534 of file [virtio](#).

References [start\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

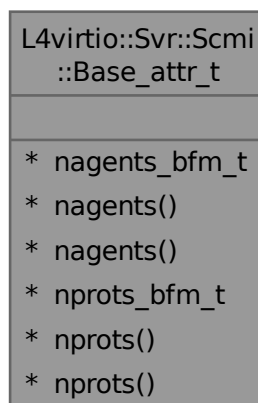
- `l4/l4virtio/server/virtio`

15.421 L4virtio::Svr::Scmi::Base_attr_t Struct Reference

SCMI base protocol attributes.

```
#include <virtio-scmi-device>
```

Collaboration diagram for L4virtio::Svr::Scmi::Base_attr_t:



- typedef [cxx::Bitfield](#)< decltype(attr_raw), 8, 15 > **nagents_bfm_t**
Type to access the *nagents* bits (8 to 15) of *attr_raw*.
 - constexpr [nagents_bfm_t::Val](#) **nagents** () const
Get the *nagents* bits (8 to 15) of *attr_raw*.
 - constexpr [nagents_bfm_t::Ref](#) **nagents** ()
Get a reference to the *nagents* bits (8 to 15) of *attr_raw*.
-
- typedef [cxx::Bitfield](#)< decltype(attr_raw), 0, 7 > **nprots_bfm_t**
Type to access the *nprots* bits (0 to 7) of *attr_raw*.
 - constexpr [nprots_bfm_t::Val](#) **nprots** () const
Get the *nprots* bits (0 to 7) of *attr_raw*.
 - constexpr [nprots_bfm_t::Ref](#) **nprots** ()
Get a reference to the *nprots* bits (0 to 7) of *attr_raw*.

15.421.1 Detailed Description

SCMI base protocol attributes.

Definition at line 90 of file [virtio-scmi-device](#).

The documentation for this struct was generated from the following file:

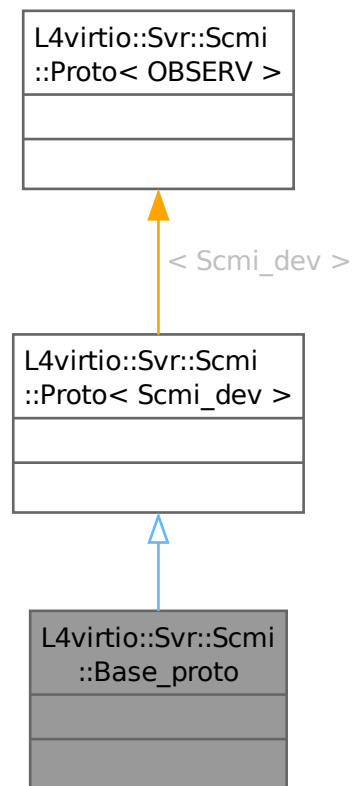
- I4/I4virtio/server/virtio-scmi-device

15.422 L4virtio::Svr::Scmi::Base_proto Class Reference

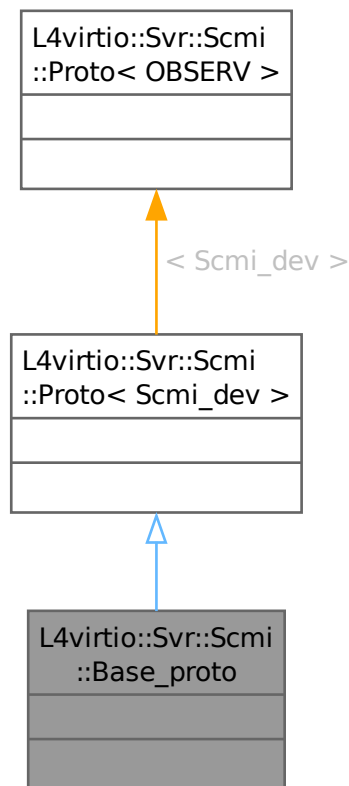
Base class for the SCMI base protocol.

```
#include <virtio-scmi-device>
```

Inheritance diagram for L4virtio::Svr::Scmi::Base_proto:



Collaboration diagram for L4virtio::Svr::Scmi::Base_proto:



15.422.1 Detailed Description

Base class for the SCMI base protocol.

Use this class as a base to implement the base protocol.

Definition at line 458 of file [virtio-scmi-device](#).

The documentation for this class was generated from the following file:

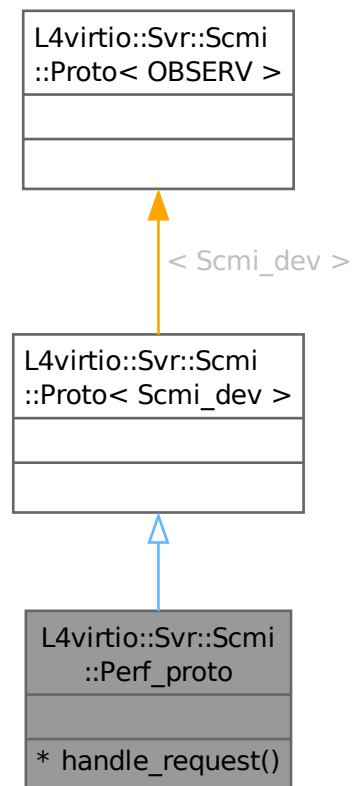
- `l4/l4virtio/server/virtio-scmi-device`

15.423 L4virtio::Svr::Scmi::Perf_proto Class Reference

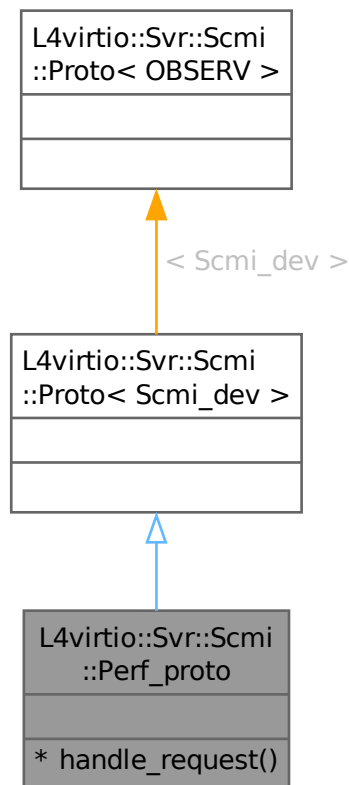
Base class for the SCMI performance protocol.

```
#include <virtio-scmi-device>
```


Inheritance diagram for L4virtio::Svr::Scmi::Perf_proto:



Collaboration diagram for L4virtio::Svr::Scmi::Perf_proto:



15.423.1 Detailed Description

Base class for the SCMI performance protocol.

Use this class as a base to implement the performance protocol.

If you want to use this from a Uvmm Linux guest, the device tree needs to look something like this:

```

firmware {
    scmi {
        compatible = "arm,scmi-virtio";

        #address-cells = <1>;
        #size-cells = <0>;

        cpufreq: protocol@13 {
            reg = <0x13>;
            #clock-cells = <1>;
        };
    };
};
....

cpu@0 {
    device_type = "cpu";
    reg = <0x0>;
    clocks = <&cpufreq 0>; // domain_id
};
  
```

Definition at line 662 of file [virtio-scmi-device](#).

The documentation for this class was generated from the following file:

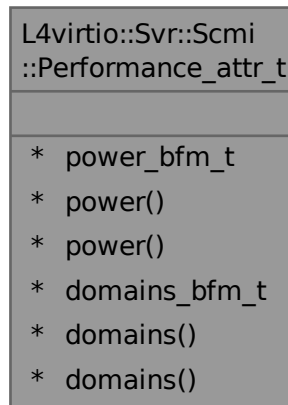
- l4/l4virtio/server/virtio-scmi-device

15.424 L4virtio::Svr::Scmi::Performance_attr_t Struct Reference

SCMI performance protocol attributes.

```
#include <virtio-scmi-device>
```

Collaboration diagram for L4virtio::Svr::Scmi::Performance_attr_t:



- typedef [cxx::Bitfield](#)< decltype(attr_raw), 16, 16 > **power_bfm_t**
Type to access the [power](#) bits (16 to 16) of *attr_raw*.
- constexpr [power_bfm_t::Val](#) **power** () const
Get the [power](#) bits (16 to 16) of *attr_raw*.
- constexpr [power_bfm_t::Ref](#) **power** ()
Get a reference to the [power](#) bits (16 to 16) of *attr_raw*.
- typedef [cxx::Bitfield](#)< decltype(attr_raw), 0, 15 > **domains_bfm_t**
Type to access the [domains](#) bits (0 to 15) of *attr_raw*.
- constexpr [domains_bfm_t::Val](#) **domains** () const
Get the [domains](#) bits (0 to 15) of *attr_raw*.
- constexpr [domains_bfm_t::Ref](#) **domains** ()
Get a reference to the [domains](#) bits (0 to 15) of *attr_raw*.

15.424.1 Detailed Description

SCMI performance protocol attributes.

Definition at line 112 of file [virtio-scmi-device](#).

The documentation for this struct was generated from the following file:

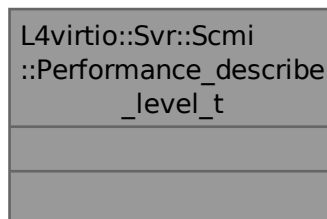
- l4/l4virtio/server/virtio-scmi-device

15.425 L4virtio::Svr::Scmi::Performance_describe_level_t Struct Reference

SCMI performance describe level.

```
#include <virtio-scmi-device>
```

Collaboration diagram for L4virtio::Svr::Scmi::Performance_describe_level_t:



15.425.1 Detailed Description

SCMI performance describe level.

Definition at line 150 of file [virtio-scmi-device](#).

The documentation for this struct was generated from the following file:

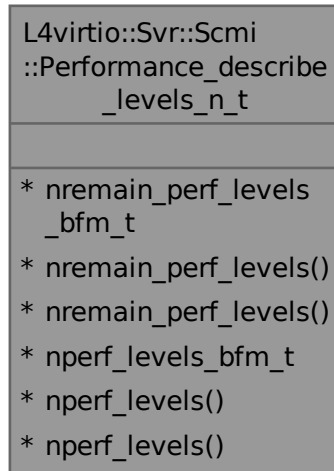
- l4/l4virtio/server/virtio-scmi-device

15.426 L4virtio::Svr::Scmi::Performance_describe_levels_n_t Struct Reference

SCMI performance describe levels numbers.

```
#include <virtio-scmi-device>
```

Collaboration diagram for L4virtio::Svr::Scmi::Performance_describe_levels_n_t:



- typedef [cxx::Bitfield](#)< decltype(num_levels_raw), 16, 31 > **nremain_perf_levels_bfm_t**
Type to access the [nremain_perf_levels](#) bits (16 to 31) of [num_levels_raw](#).
- constexpr [nremain_perf_levels_bfm_t::Val](#) **nremain_perf_levels** () const
Get the [nremain_perf_levels](#) bits (16 to 31) of [num_levels_raw](#).
- constexpr [nremain_perf_levels_bfm_t::Ref](#) **nremain_perf_levels** ()
Get a reference to the [nremain_perf_levels](#) bits (16 to 31) of [num_levels_raw](#).
- typedef [cxx::Bitfield](#)< decltype(num_levels_raw), 0, 11 > **nperf_levels_bfm_t**
Type to access the [nperf_levels](#) bits (0 to 11) of [num_levels_raw](#).
- constexpr [nperf_levels_bfm_t::Val](#) **nperf_levels** () const
Get the [nperf_levels](#) bits (0 to 11) of [num_levels_raw](#).
- constexpr [nperf_levels_bfm_t::Ref](#) **nperf_levels** ()
Get a reference to the [nperf_levels](#) bits (0 to 11) of [num_levels_raw](#).

15.426.1 Detailed Description

SCMI performance describe levels numbers.

Definition at line 142 of file [virtio-scmi-device](#).

The documentation for this struct was generated from the following file:

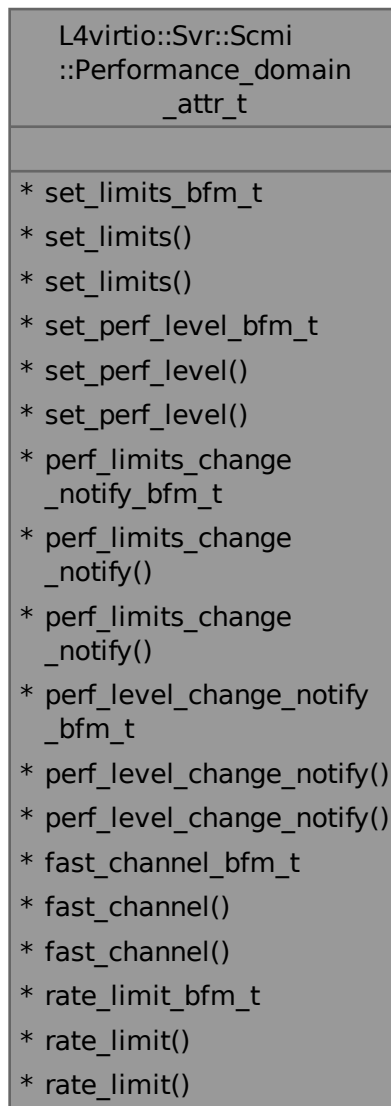
- l4/l4virtio/server/virtio-scmi-device

15.427 L4virtio::Svr::Scmi::Performance_domain_attr_t Struct Reference

SCMI performance domain protocol attributes.

```
#include <virtio-scmi-device>
```

Collaboration diagram for L4virtio::Svr::Scmi::Performance_domain_attr_t:



- typedef [cxx::Bitfield](#)< decltype(attr_raw), 31, 31 > **set_limits_bfm_t**
Type to access the [set_limits](#) bits (31 to 31) of *attr_raw*.
- constexpr [set_limits_bfm_t::Val](#) **set_limits** () const

- Get the [set_limits](#) bits (31 to 31) of *attr_raw*.

 - `constexpr set_limits_bfm_t::Ref set_limits ()`
 Get a reference to the [set_limits](#) bits (31 to 31) of *attr_raw*.
- `typedef cxx::Bitfield< decltype(attr_raw), 30, 30 > set_perf_level_bfm_t`
 Type to access the [set_perf_level](#) bits (30 to 30) of *attr_raw*.
 - `constexpr set_perf_level_bfm_t::Val set_perf_level () const`
 Get the [set_perf_level](#) bits (30 to 30) of *attr_raw*.
 - `constexpr set_perf_level_bfm_t::Ref set_perf_level ()`
 Get a reference to the [set_perf_level](#) bits (30 to 30) of *attr_raw*.
- `typedef cxx::Bitfield< decltype(attr_raw), 29, 29 > perf_limits_change_notify_bfm_t`
 Type to access the [perf_limits_change_notify](#) bits (29 to 29) of *attr_raw*.
 - `constexpr perf_limits_change_notify_bfm_t::Val perf_limits_change_notify () const`
 Get the [perf_limits_change_notify](#) bits (29 to 29) of *attr_raw*.
 - `constexpr perf_limits_change_notify_bfm_t::Ref perf_limits_change_notify ()`
 Get a reference to the [perf_limits_change_notify](#) bits (29 to 29) of *attr_raw*.
- `typedef cxx::Bitfield< decltype(attr_raw), 28, 28 > perf_level_change_notify_bfm_t`
 Type to access the [perf_level_change_notify](#) bits (28 to 28) of *attr_raw*.
 - `constexpr perf_level_change_notify_bfm_t::Val perf_level_change_notify () const`
 Get the [perf_level_change_notify](#) bits (28 to 28) of *attr_raw*.
 - `constexpr perf_level_change_notify_bfm_t::Ref perf_level_change_notify ()`
 Get a reference to the [perf_level_change_notify](#) bits (28 to 28) of *attr_raw*.
- `typedef cxx::Bitfield< decltype(attr_raw), 27, 27 > fast_channel_bfm_t`
 Type to access the [fast_channel](#) bits (27 to 27) of *attr_raw*.
 - `constexpr fast_channel_bfm_t::Val fast_channel () const`
 Get the [fast_channel](#) bits (27 to 27) of *attr_raw*.
 - `constexpr fast_channel_bfm_t::Ref fast_channel ()`
 Get a reference to the [fast_channel](#) bits (27 to 27) of *attr_raw*.
- `typedef cxx::Bitfield< decltype(rate_limit_raw), 0, 19 > rate_limit_bfm_t`
 Type to access the [rate_limit](#) bits (0 to 19) of *rate_limit_raw*.
 - `constexpr rate_limit_bfm_t::Val rate_limit () const`
 Get the [rate_limit](#) bits (0 to 19) of *rate_limit_raw*.
 - `constexpr rate_limit_bfm_t::Ref rate_limit ()`
 Get a reference to the [rate_limit](#) bits (0 to 19) of *rate_limit_raw*.

15.427.1 Detailed Description

SCMI performance domain protocol attributes.

Definition at line 124 of file [virtio-scmi-device](#).

The documentation for this struct was generated from the following file:

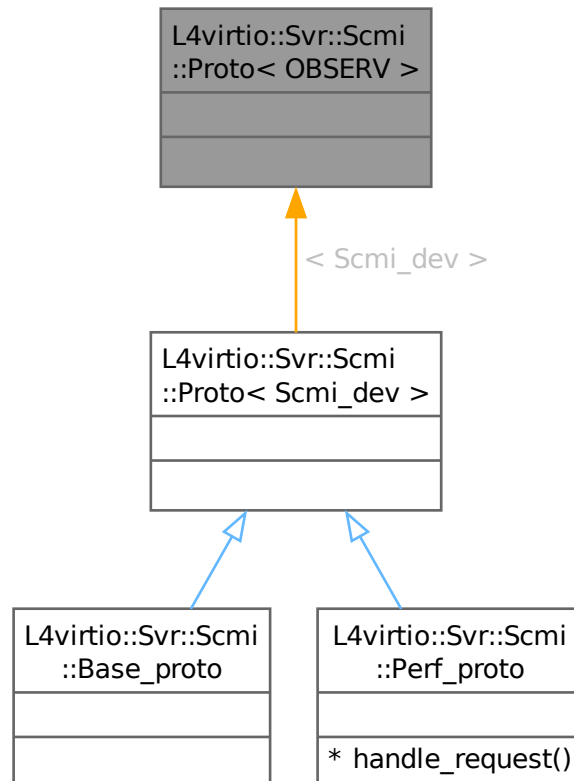
- `I4/I4virtio/server/virtio-scmi-device`

15.428 L4virtio::Svr::Scmi::Proto< OBSERV > Struct Template Reference

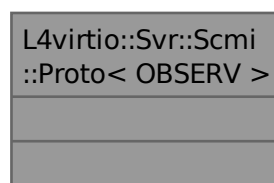
Base class for all protocols.

```
#include <virtio-scmi-device>
```

Inheritance diagram for L4virtio::Svr::Scmi::Proto< OBSERV >:



Collaboration diagram for L4virtio::Svr::Scmi::Proto< OBSERV >:



15.428.1 Detailed Description

```
template<typename OBSERV>
struct L4virtio::Svr::Scmi::Proto< OBSERV >
```

Base class for all protocols.

Defines an interface for processing the virtio buffers for the implemented protocol.

Definition at line 299 of file [virtio-scmi-device](#).

The documentation for this struct was generated from the following file:

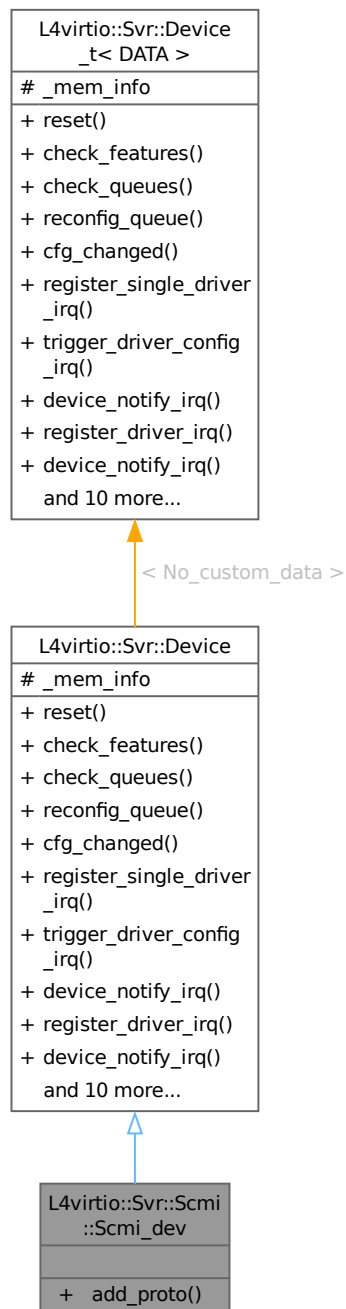
- I4/I4virtio/server/virtio-scmi-device

15.429 L4virtio::Svr::Scmi::Scmi_dev Class Reference

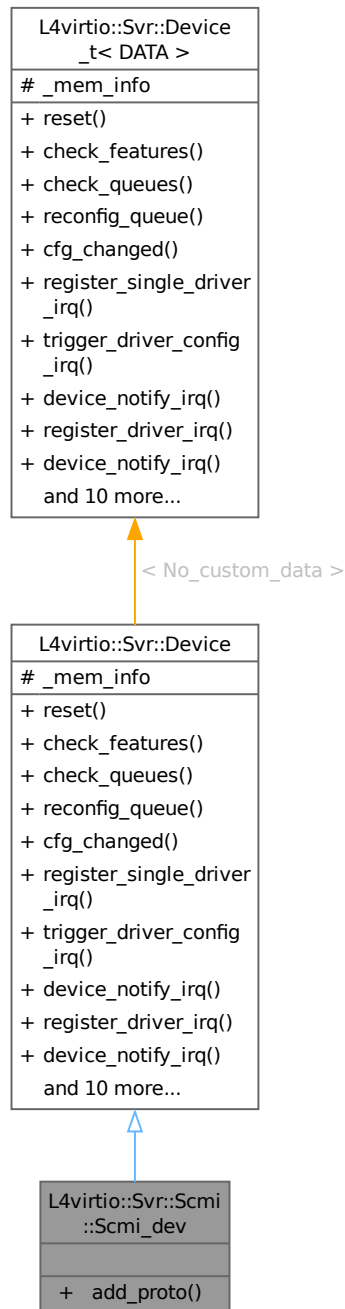
A server implementation of the virtio-scmi protocol.

```
#include <virtio-scmi-device>
```

Inheritance diagram for L4virtio::Svr::Scmi::Scmi_dev:



Collaboration diagram for L4virtio::Svr::Scmi::Scmi_dev:



Public Member Functions

- void **add_proto** (l4_uint32_t id, Proto< Scmi_dev > *proto)

Add an actual protocol implementation with the given id to the server.

Public Member Functions inherited from L4virtio::Svr::Device_t< No_custom_data >

- virtual bool **check_features** ()

- callback for checking the subset of accepted features*
- virtual void **cfg_changed** (unsigned)
 - callback for client device configuration changes*
- virtual void **register_driver_irq** (unsigned idx)
 - Callback for registering an notification IRQ (multi IRQ).*
- virtual **L4::Cap**< **L4::Irq** > **device_notify_irq** (unsigned idx)
 - Callback to gather the device notification IRQ (multi IRQ).*
- virtual unsigned **num_events_supported** () const
 - Return the highest notification index supported.*
- **Device_t** (**Dev_config** *dev_config)
 - Make a device for the given config.*
- **Mem_list** const * **mem_info** () const
 - Get the memory region list used for this device.*
- void **reset_queue_config** (unsigned idx, unsigned num_max, bool inc_generation=false, unsigned device_notify_index=0)
 - Trigger reset for the configuration space for queue idx.*
- void **init_mem_info** (unsigned num)
 - Initialize the memory region list to the given maximum.*
- void **device_error** ()
 - Transition device into DEVICE_NEEDS_RESET state.*
- bool **setup_queue** (Virtqueue *q, unsigned qn, unsigned num_max)
 - Enable/disable the specified queue.*
- bool **handle_mem_cmd_write** ()
 - Check for a value in the cmd register and handle a write.*
- void **enable_trusted_ds_validation** ()
 - Enable trusted dataspace validation.*
- void **add_trusted_dataspaces** (std::shared_ptr< Ds_vector const > ds)
 - Provide a list of trusted dataspaces that can be used for validation.*

Additional Inherited Members

Protected Attributes inherited from **L4virtio::Svr::Device_t**< **No_custom_data** >

- **Mem_list** **_mem_info**
 - Memory region list.*

15.429.1 Detailed Description

A server implementation of the virtio-scmi protocol.

Use this class as a base to implement your own specific SCMI device.

SCMI defines multiple protocols which can be optionally handled. This server implementation is flexible enough to handle any combination of them. The user of this server has to deviate from the provided **Proto** classes (for the protocols he want to handle) and needs to implement the required callbacks.

Right now, support for the base and the performance protocol is provided.

The base protocol is mandatory.

If you want to use this from a Uvmm Linux guest, the device tree needs to look something like this:

```
firmware {
    scmi {
        compatible = "arm,scmi-virtio";

        #address-cells = <1>;
        #size-cells = <0>;

        // ... supported protocols ...
    };
};
```

Definition at line 336 of file [virtio-scmi-device](#).

The documentation for this class was generated from the following file:

- l4/l4virtio/server/virtio-scmi-device

15.430 L4virtio::Svr::Scmi::Scmi_hdr_t Struct Reference

SCMI header.

```
#include <virtio-scmi-device>
```

Collaboration diagram for L4virtio::Svr::Scmi::Scmi_hdr_t:



- typedef [cxx::Bitfield](#)< decltype(hdr_raw), 18, 27 > **token_bfm_t**
Type to access the *token* bits (18 to 27) of *hdr_raw*.

- constexpr [token_bfm_t::Val](#) **token** () const
Get the *token* bits (18 to 27) of *hdr_raw*.
- constexpr [token_bfm_t::Ref](#) **token** ()
Get a reference to the *token* bits (18 to 27) of *hdr_raw*.
- typedef [cxx::Bitfield](#)< decltype(hdr_raw), 10, 17 > **protocol_id_bfm_t**
Type to access the *protocol_id* bits (10 to 17) of *hdr_raw*.
- constexpr [protocol_id_bfm_t::Val](#) **protocol_id** () const
Get the *protocol_id* bits (10 to 17) of *hdr_raw*.
- constexpr [protocol_id_bfm_t::Ref](#) **protocol_id** ()
Get a reference to the *protocol_id* bits (10 to 17) of *hdr_raw*.
- typedef [cxx::Bitfield](#)< decltype(hdr_raw), 8, 9 > **message_type_bfm_t**
Type to access the *message_type* bits (8 to 9) of *hdr_raw*.
- constexpr [message_type_bfm_t::Val](#) **message_type** () const
Get the *message_type* bits (8 to 9) of *hdr_raw*.
- constexpr [message_type_bfm_t::Ref](#) **message_type** ()
Get a reference to the *message_type* bits (8 to 9) of *hdr_raw*.
- typedef [cxx::Bitfield](#)< decltype(hdr_raw), 0, 7 > **message_id_bfm_t**
Type to access the *message_id* bits (0 to 7) of *hdr_raw*.
- constexpr [message_id_bfm_t::Val](#) **message_id** () const
Get the *message_id* bits (0 to 7) of *hdr_raw*.
- constexpr [message_id_bfm_t::Ref](#) **message_id** ()
Get a reference to the *message_id* bits (0 to 7) of *hdr_raw*.

15.430.1 Detailed Description

SCMI header.

Definition at line 45 of file [virtio-scmi-device](#).

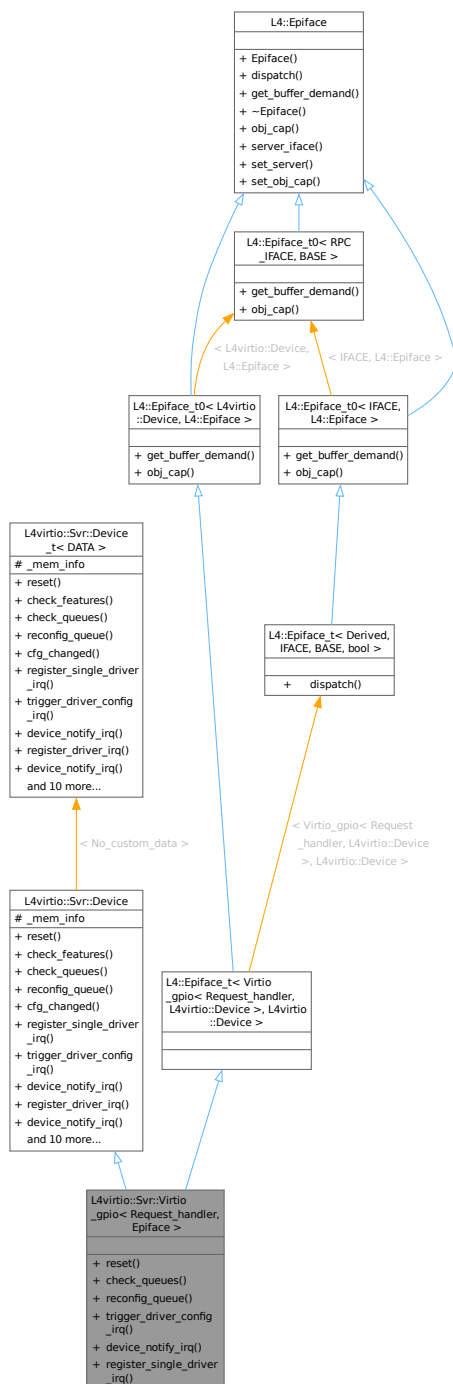
The documentation for this struct was generated from the following file:

- I4/I4virtio/server/virtio-scmi-device

A server implementation of the virtio-gpio protocol.

```
#include <virtio-gpio-device>
```

Inheritance diagram for L4virtio::Svr::Virtio_gpio< Request_handler, Epiface >:



Public Member Functions

- void **reset** () override
reset callback, called for doing a device reset
- bool **check_queues** () override
callback for checking if the queues at DRIVER_OK transition
- int **reconfig_queue** (unsigned idx) override
callback for client queue-config request
- void **trigger_driver_config_irq** () override
callback for triggering configuration change notification IRQ
- L4::Cap< L4::Irq > **device_notify_irq** () const override
callback to gather the device notification IRQ (old-style)
- void **register_single_driver_irq** () override
callback for registering a single guest IRQ for all queues (old-style)

Public Member Functions inherited from L4virtio::Svr::Device_t< No_custom_data >

- virtual bool **check_features** ()
callback for checking the subset of accepted features
- virtual void **cfg_changed** (unsigned)
callback for client device configuration changes
- virtual void **register_driver_irq** (unsigned idx)
Callback for registering an notification IRQ (multi IRQ).
- virtual L4::Cap< L4::Irq > **device_notify_irq** (unsigned idx)
Callback to gather the device notification IRQ (multi IRQ).
- virtual unsigned **num_events_supported** () const
Return the highest notification index supported.
- **Device_t** (Dev_config *dev_config)
Make a device for the given config.
- Mem_list const * **mem_info** () const
Get the memory region list used for this device.
- void **reset_queue_config** (unsigned idx, unsigned num_max, bool inc_generation=false, unsigned device_notify_index=0)
Trigger reset for the configuration space for queue idx.
- void **init_mem_info** (unsigned num)
Initialize the memory region list to the given maximum.
- void **device_error** ()
Transition device into DEVICE_NEEDS_RESET state.
- bool **setup_queue** (Virtqueue *q, unsigned qn, unsigned num_max)
Enable/disable the specified queue.
- bool **handle_mem_cmd_write** ()
Check for a value in the cmd register and handle a write.
- void **enable_trusted_ds_validation** ()
Enable trusted dataspace validation.
- void **add_trusted_dataspaces** (std::shared_ptr< Ds_vector const > ds)
Provide a list of trusted dataspaces that can be used for validation.

Public Member Functions inherited from [L4::Epiface_t0< L4virtio::Device, L4::Epiface >](#)

- [Type_info::Demand](#) **get_buffer_demand** () const
Get the server-side buffer demand based in IFACE.
- [Cap< L4virtio::Device >](#) **obj_cap** () const
Get the (typed) capability to this object.

Public Member Functions inherited from [L4::Epiface](#)

- **Epiface** ()
Make a server object.
- virtual **~Epiface** ()=0
Destroy the object.
- Stored_cap [obj_cap](#) () const
Get the capability to the kernel object belonging to this object.
- [Server_iface](#) * **server_iface** () const
Get pointer to server interface at which the object is currently registered.
- int **set_server** ([Server_iface](#) *srv, [Cap< void >](#) cap, bool managed=false)
Set server registration info for the object.
- void **set_obj_cap** ([Cap< void >](#) const &cap)
Deprecated server registration function.

Additional Inherited Members

Public Types inherited from [L4::Epiface_t0< L4virtio::Device, L4::Epiface >](#)

- using **Interface**
Data type of the IPC interface definition.

Public Types inherited from [L4::Epiface](#)

- using **Server_iface** = [lpc_svr::Server_iface](#)
Type for abstract server interface.
- using **Demand** = [lpc_svr::Server_iface::Demand](#)
Type for server-side receive buffer demand.

Protected Attributes inherited from [L4virtio::Svr::Device_t< No_custom_data >](#)

- **Mem_list_mem_info**
Memory region list.

15.431.1 Detailed Description

```
template<typename Request_handler, typename Epiface = L4virtio::Device>
class L4virtio::Svr::Virtio_gpio< Request_handler, Epiface >
```

A server implementation of the virtio-gpio protocol.

Template Parameters

| | |
|------------------------|---|
| <i>Request_handler</i> | <p>The type that is used to handle incoming requests. Needs to have:</p> <ul style="list-style-type: none"> • <code>bool get_direction(l4_uint16_t gpio, l4_uint8_t *dir)</code> • <code>bool set_direction(l4_uint16_t gpio, l4_uint8_t dir)</code> • <code>bool get_value(l4_uint16_t gpio, l4_uint8_t *val)</code> • <code>bool set_value(l4_uint16_t gpio, l4_uint8_t val)</code> • <code>bool set_irq_type(l4_uint16_t gpio, l4_uint8_t mode)</code> • <code>bool enable_irq(l4_uint16_t gpio, std::shared_ptr<Virtio_gpio::Irq_handler> const &hdl)</code> functions. |
| <i>Epiface</i> | The Epiface to derive from. Defaults to L4virtio::Device . |

Definition at line 142 of file [virtio-gpio-device](#).

The documentation for this class was generated from the following file:

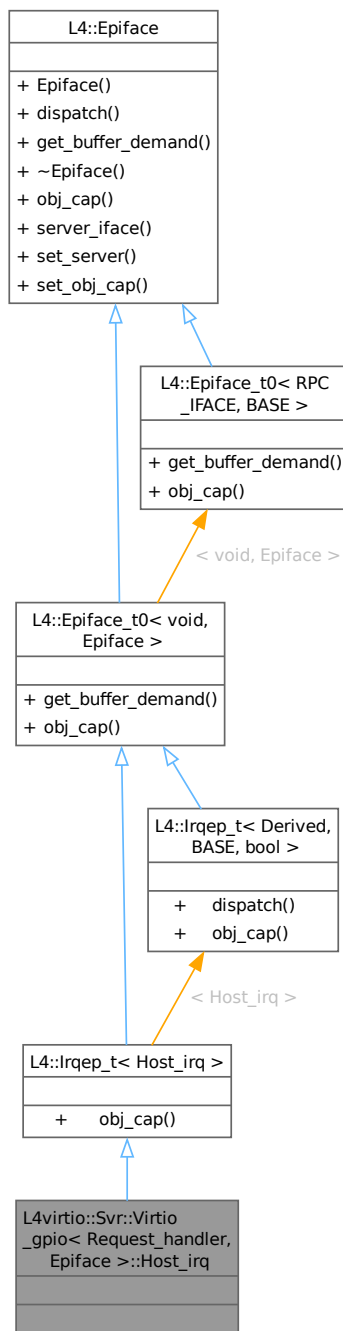
- `l4/l4virtio/server/virtio-gpio-device`

15.432 L4virtio::Svr::Virtio_gpio< Request_handler, Epiface >::Host_irq Struct Reference

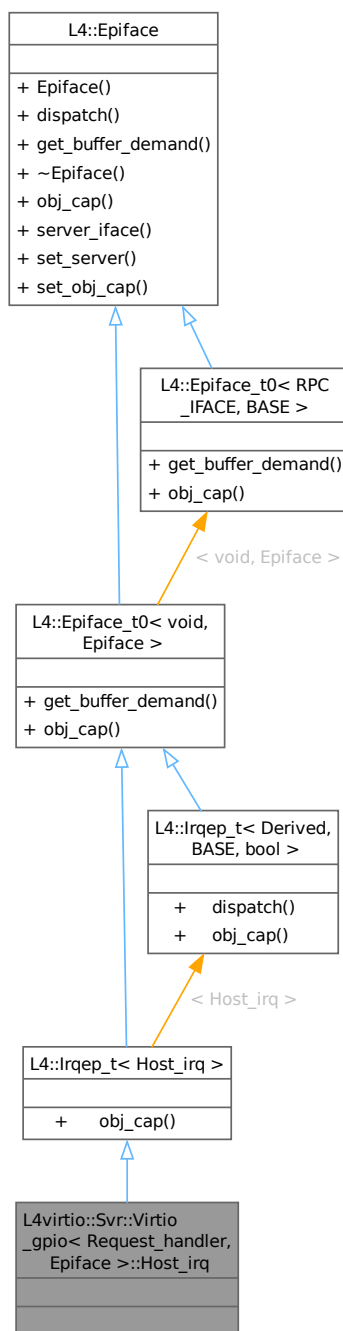
Handler for the host irq.

```
#include <virtio-gpio-device>
```

Inheritance diagram for L4virtio::Svr::Virtio_gpio< Request_handler, Epiface >::Host_irq:



Collaboration diagram for L4virtio::Svr::Virtio_gpio< Request_handler, Epiface >::Host_irq:



Additional Inherited Members

Public Types inherited from **L4::Epiface_t0< void, Epiface >**

- using **Interface**

Data type of the IPC interface definition.

Public Types inherited from [L4::Epiface](#)

- using **Server_iface** = [lpc_svr::Server_iface](#)
Type for abstract server interface.
- using **Demand** = [lpc_svr::Server_iface::Demand](#)
Type for server-side receive buffer demand.

Public Member Functions inherited from [L4::lrqep_t< Host_irq >](#)

- [Cap< L4::lrq > obj_cap](#) () const
Get the (typed) capability to this object.

Public Member Functions inherited from [L4::Epiface_t0< void, Epiface >](#)

- [Type_info::Demand get_buffer_demand](#) () const
Get the server-side buffer demand based in IFACE.
- [Cap< void > obj_cap](#) () const
Get the (typed) capability to this object.

Public Member Functions inherited from [L4::Epiface](#)

- **Epiface** ()
Make a server object.
- virtual **~Epiface** ()=0
Destroy the object.
- Stored_cap [obj_cap](#) () const
Get the capability to the kernel object belonging to this object.
- [Server_iface * server_iface](#) () const
Get pointer to server interface at which the object is currently registered.
- int [set_server](#) ([Server_iface *srv](#), [Cap< void > cap](#), bool managed=false)
Set server registration info for the object.
- void [set_obj_cap](#) ([Cap< void > const &cap](#))
Deprecated server registration function.

15.432.1 Detailed Description

```
template<typename Request_handler, typename Epiface = L4virtio::Device>
struct L4virtio::Svr::Virtio_gpio< Request_handler, Epiface >::Host_irq
```

Handler for the host irq.

An [L4::lrqep_t](#) to handle irq's send to the server.

Definition at line 198 of file [virtio-gpio-device](#).

The documentation for this struct was generated from the following file:

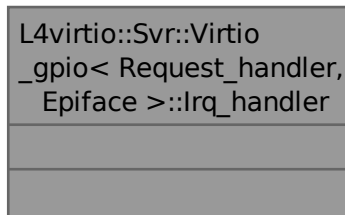
- l4/l4virtio/server/virtio-gpio-device

15.433 L4virtio::Svr::Virtio_gpio< Request_handler, Epiface >::Irq_handler Struct Reference

Handler for an gpio pin irq.

```
#include <virtio-gpio-device>
```

Collaboration diagram for L4virtio::Svr::Virtio_gpio< Request_handler, Epiface >::Irq_handler:



15.433.1 Detailed Description

```
template<typename Request_handler, typename Epiface = L4virtio::Device>
struct L4virtio::Svr::Virtio_gpio< Request_handler, Epiface >::Irq_handler
```

Handler for an gpio pin irq.

This notifies the virtio client that an gpio pin irq happened or the operation was canceled.

This needs to be called by any server implementation of the [Virtio_gpio](#) class, after an irq was enabled with the enable method of the [Gpio_request_handler](#).

Definition at line [166](#) of file [virtio-gpio-device](#).

The documentation for this struct was generated from the following file:

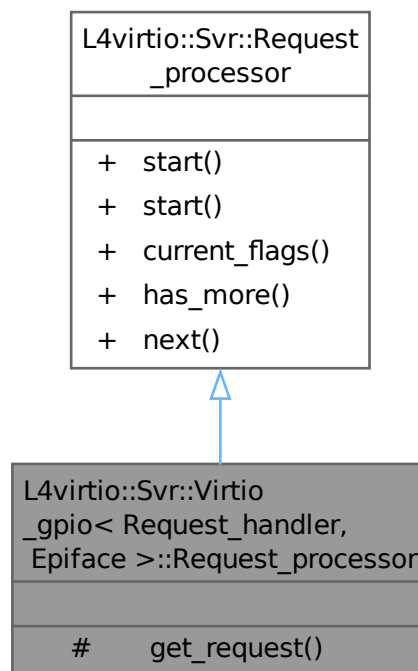
- [I4/I4virtio/server/virtio-gpio-device](#)

15.434 L4virtio::Svr::Virtio_gpio< Request_handler, Epiface >::Request_processor Struct Reference

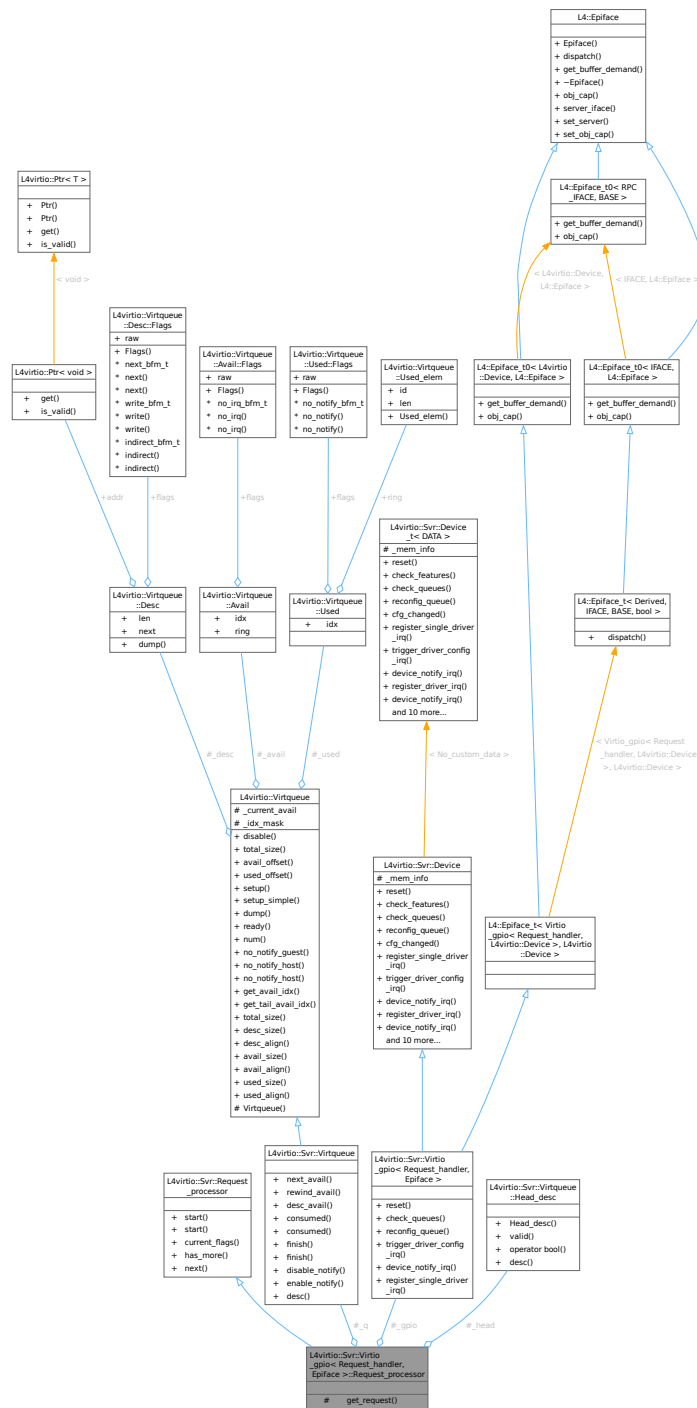
Generic handler for the Virtio requests.

```
#include <virtio-gpio-device>
```

Inheritance diagram for L4virtio::Svr::Virtio_gpio< Request_handler, Epiface >::Request_processor:



Collaboration diagram for L4virtio::Svr::Virtio_gpio< Request_handler, Epiface >::Request_processor:



Protected Member Functions

- `template<typename T>`
`T get_request ()`

The driver prepares the GPIO request in two data parts: 1st: in_hdr 2nd: out_hdr.

Additional Inherited Members

Public Member Functions inherited from [L4virtio::Svr::Request_processor](#)

- `template<typename DESC_MAN, typename ... ARGS>`
`void start (DESC_MAN *dm, Virtqueue *ring, Virtqueue::Head_desc const &request, ARGS... args)`
Start processing a new request.
- `template<typename DESC_MAN, typename ... ARGS>`
`Virtqueue::Request const & start (DESC_MAN *dm, Virtqueue::Request const &request, ARGS... args)`
Start processing a new request.
- `Virtqueue::Desc::Flags current_flags () const`
Get the flags of the currently processed descriptor.
- `bool has_more () const`
Are there more chained descriptors?
- `template<typename DESC_MAN, typename ... ARGS>`
`bool next (DESC_MAN *dm, ARGS... args)`
Switch to the next descriptor in a descriptor chain.

15.434.1 Detailed Description

`template<typename Request_handler, typename Epiface = L4virtio::Device>`
`struct L4virtio::Svr::Virtio_gpio< Request_handler, Epiface >::Request_processor`

Generic handler for the Virtio requests.

Definition at line 213 of file [virtio-gpio-device](#).

15.434.2 Member Function Documentation

15.434.2.1 `get_request()`

```
template<typename Request_handler, typename Epiface = L4virtio::Device>
template<typename T>
T L4virtio::Svr::Virtio\_gpio< Request\_handler, Epiface >::Request\_processor::get\_request ()
[inline], [protected]
```

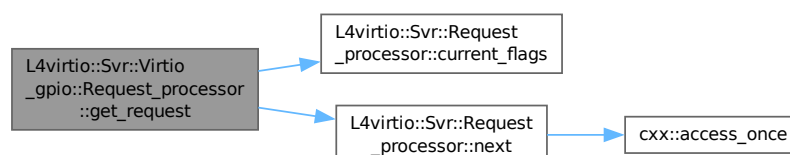
The driver prepares the GPIO request in two data parts: 1st: in_hdr 2rd: out_hdr.

This parses the two Data_buffers and create the Gpio_* structure.

Definition at line 241 of file [virtio-gpio-device](#).

References [L4virtio::Svr::Request_processor::current_flags\(\)](#), and [L4virtio::Svr::Request_processor::next\(\)](#).

Here is the call graph for this function:



The documentation for this struct was generated from the following file:

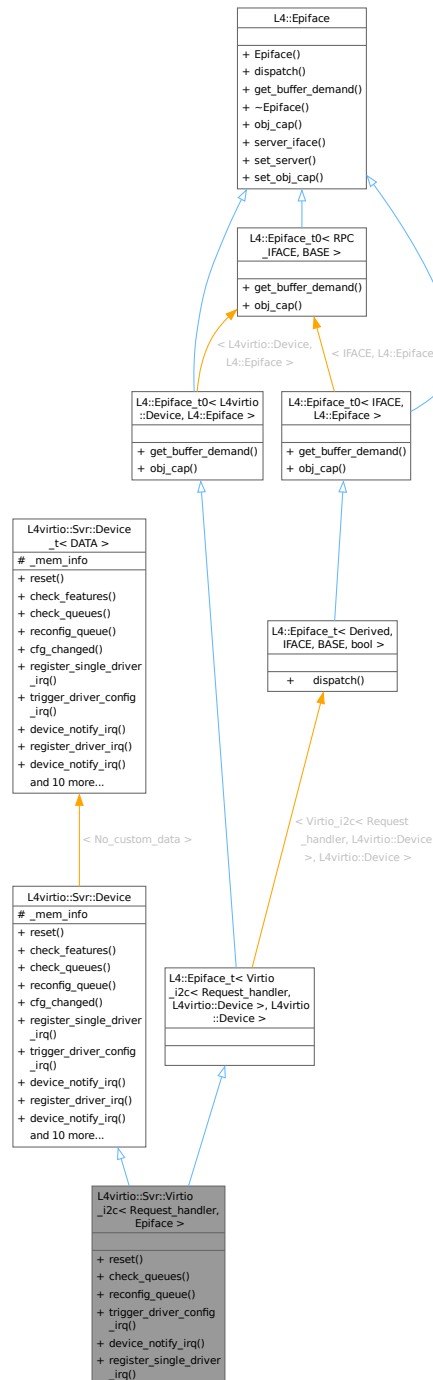
- `I4/I4virtio/server/virtio-gpio-device`

15.435 L4virtio::Svr::Virtio_i2c< Request_handler, Epiface > Class Template Reference

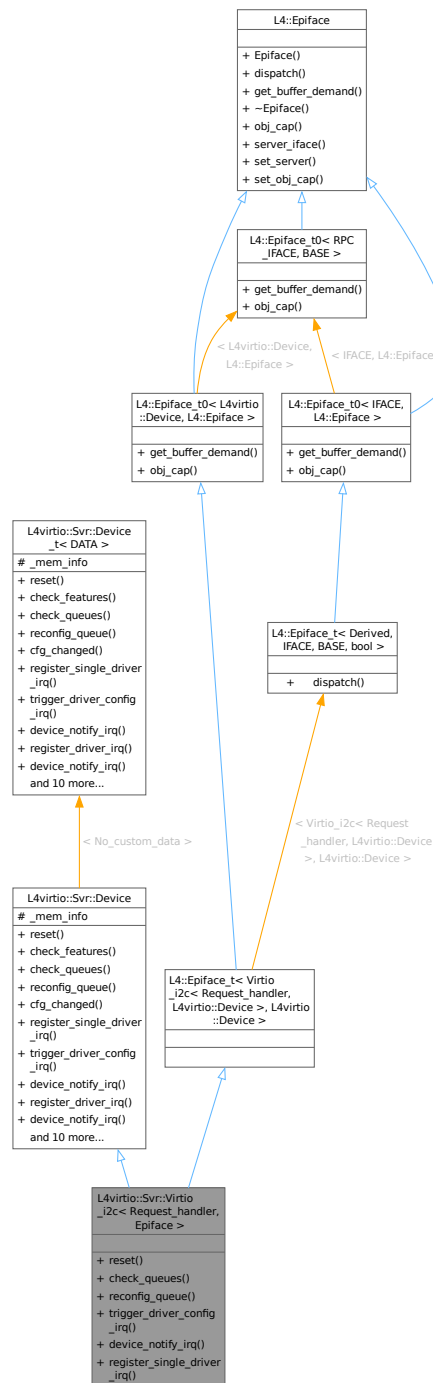
A server implementation of the virtio-i2c protocol.

```
#include <virtio-i2c-device>
```

Inheritance diagram for L4virtio::Svr::Virtio_i2c< Request_handler, Epiface >:



Collaboration diagram for L4virtio::Svr::Virtio_i2c< Request_handler, Epiface >:



Data Structures

- class [Host_irq](#)
Handler for the host irq.
- class [Request_processor](#)
Handler for the Virtio requests.

Public Member Functions

- void **reset** () override
reset callback, called for doing a device reset
- bool **check_queues** () override
callback for checking if the queues at DRIVER_OK transition
- int **reconfig_queue** (unsigned idx) override
callback for client queue-config request
- void **trigger_driver_config_irq** () override
callback for triggering configuration change notification IRQ
- L4::Cap< L4::Irq > **device_notify_irq** () const override
callback to gather the device notification IRQ (old-style)
- void **register_single_driver_irq** () override
callback for registering a single guest IRQ for all queues (old-style)

Public Member Functions inherited from L4virtio::Svr::Device_t< No_custom_data >

- virtual bool **check_features** ()
callback for checking the subset of accepted features
- virtual void **cfg_changed** (unsigned)
callback for client device configuration changes
- virtual void **register_driver_irq** (unsigned idx)
Callback for registering an notification IRQ (multi IRQ).
- virtual L4::Cap< L4::Irq > **device_notify_irq** (unsigned idx)
Callback to gather the device notification IRQ (multi IRQ).
- virtual unsigned **num_events_supported** () const
Return the highest notification index supported.
- **Device_t** (**Dev_config** *dev_config)
Make a device for the given config.
- Mem_list const * **mem_info** () const
Get the memory region list used for this device.
- void **reset_queue_config** (unsigned idx, unsigned num_max, bool inc_generation=false, unsigned device_notify_index=0)
Trigger reset for the configuration space for queue idx.
- void **init_mem_info** (unsigned num)
Initialize the memory region list to the given maximum.
- void **device_error** ()
Transition device into DEVICE_NEEDS_RESET state.
- bool **setup_queue** (Virtqueue *q, unsigned qn, unsigned num_max)
Enable/disable the specified queue.
- bool **handle_mem_cmd_write** ()
Check for a value in the cmd register and handle a write.
- void **enable_trusted_ds_validation** ()
Enable trusted dataspace validation.
- void **add_trusted_dataspaces** (std::shared_ptr< Ds_vector const > ds)
Provide a list of trusted dataspaces that can be used for validation.

Public Member Functions inherited from [L4::Epiface_t0< L4virtio::Device, L4::Epiface >](#)

- [Type_info::Demand](#) **get_buffer_demand** () const
Get the server-side buffer demand based in IFACE.
- [Cap< L4virtio::Device >](#) **obj_cap** () const
Get the (typed) capability to this object.

Public Member Functions inherited from [L4::Epiface](#)

- **Epiface** ()
Make a server object.
- virtual **~Epiface** ()=0
Destroy the object.
- Stored_cap [obj_cap](#) () const
Get the capability to the kernel object belonging to this object.
- [Server_iface](#) * **server_iface** () const
Get pointer to server interface at which the object is currently registered.
- int **set_server** ([Server_iface](#) *srv, [Cap< void >](#) cap, bool managed=false)
Set server registration info for the object.
- void **set_obj_cap** ([Cap< void >](#) const &cap)
Deprecated server registration function.

Additional Inherited Members

Public Types inherited from [L4::Epiface_t0< L4virtio::Device, L4::Epiface >](#)

- using **Interface**
Data type of the IPC interface definition.

Public Types inherited from [L4::Epiface](#)

- using **Server_iface** = [lpc_svr::Server_iface](#)
Type for abstract server interface.
- using **Demand** = [lpc_svr::Server_iface::Demand](#)
Type for server-side receive buffer demand.

Protected Attributes inherited from [L4virtio::Svr::Device_t< No_custom_data >](#)

- **Mem_list_mem_info**
Memory region list.

15.435.1 Detailed Description

```
template<typename Request_handler, typename Epiface = L4virtio::Device>
class L4virtio::Svr::Virtio_i2c< Request_handler, Epiface >
```

A server implementation of the virtio-i2c protocol.

Template Parameters

| | |
|------------------------|---|
| <i>Request_handler</i> | The type that is used to handle incoming requests. Needs to have <code>T handle_read(l4_uint8_t *, unsigned)</code> and <code>T handle_write(l4_uint8_t const *, unsigned)</code> functions, with <code>T</code> assignable to <code>std::optional<bool></code> . |
| <i>Epiface</i> | The Epiface to derive from. Defaults to L4virtio::Device . |

Definition at line 89 of file [virtio-i2c-device](#).

The documentation for this class was generated from the following file:

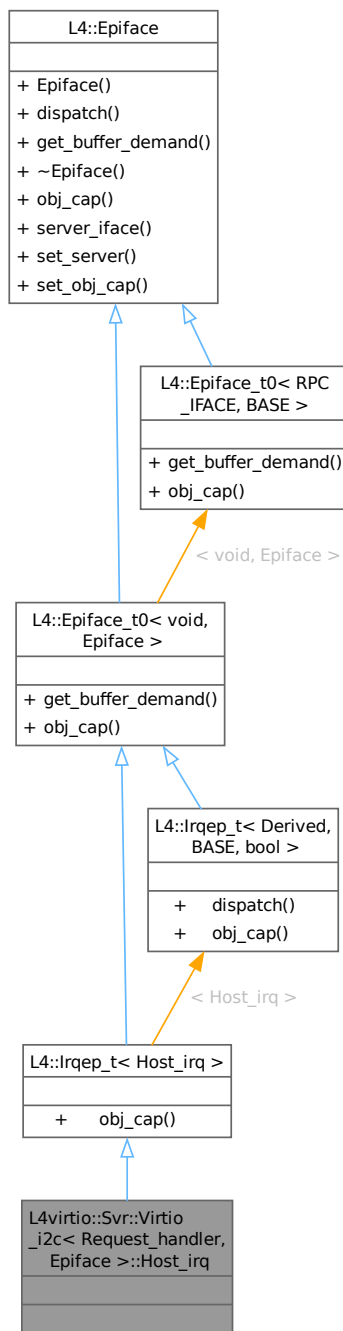
- l4/l4virtio/server/virtio-i2c-device

15.436 L4virtio::Svr::Virtio_i2c< Request_handler, Epiface >::Host_irq Class Reference

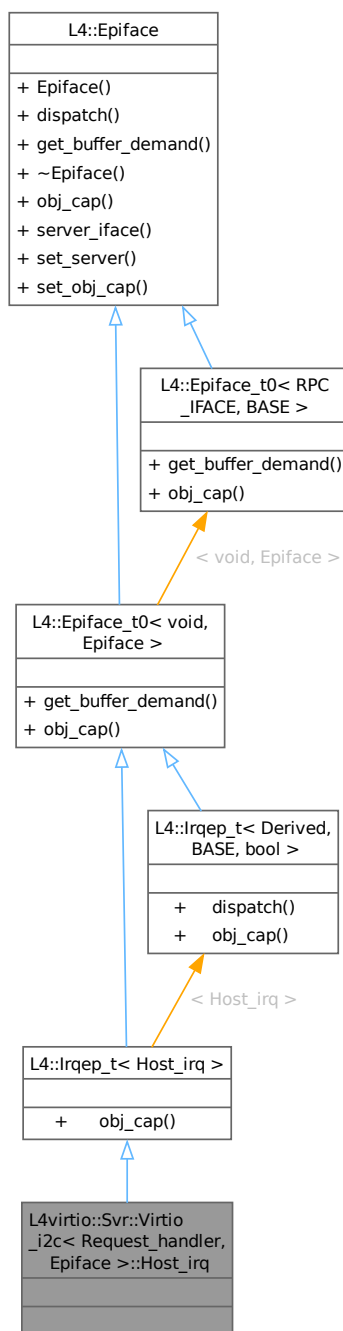
Handler for the host irq.

```
#include <virtio-i2c-device>
```

Inheritance diagram for L4virtio::Svr::Virtio_i2c< Request_handler, Epiface >::Host_irq:



Collaboration diagram for L4virtio::Svr::Virtio_i2c< Request_handler, Epiface >::Host_irq:



Additional Inherited Members

Public Types inherited from **L4::Epiface_t0< void, Epiface >**

- using **Interface**

Data type of the IPC interface definition.

Public Types inherited from [L4::Epiface](#)

- using **Server_iface** = [lpc_svr::Server_iface](#)
Type for abstract server interface.
- using **Demand** = [lpc_svr::Server_iface::Demand](#)
Type for server-side receive buffer demand.

Public Member Functions inherited from [L4::lrqep_t< Host_irq >](#)

- [Cap< L4::lrq > obj_cap](#) () const
Get the (typed) capability to this object.

Public Member Functions inherited from [L4::Epiface_t0< void, Epiface >](#)

- [Type_info::Demand get_buffer_demand](#) () const
Get the server-side buffer demand based in IFACE.
- [Cap< void > obj_cap](#) () const
Get the (typed) capability to this object.

Public Member Functions inherited from [L4::Epiface](#)

- **Epiface** ()
Make a server object.
- virtual **~Epiface** ()=0
Destroy the object.
- Stored_cap [obj_cap](#) () const
Get the capability to the kernel object belonging to this object.
- [Server_iface * server_iface](#) () const
Get pointer to server interface at which the object is currently registered.
- int [set_server](#) ([Server_iface *srv](#), [Cap< void > cap](#), bool managed=false)
Set server registration info for the object.
- void [set_obj_cap](#) ([Cap< void > const &cap](#))
Deprecated server registration function.

15.436.1 Detailed Description

```
template<typename Request_handler, typename Epiface = L4virtio::Device>
class L4virtio::Svr::Virtio_i2c< Request_handler, Epiface >::Host_irq
```

Handler for the host irq.

An [L4::lrqep_t](#) to handle irq's send to the server.

Definition at line 109 of file [virtio-i2c-device](#).

The documentation for this class was generated from the following file:

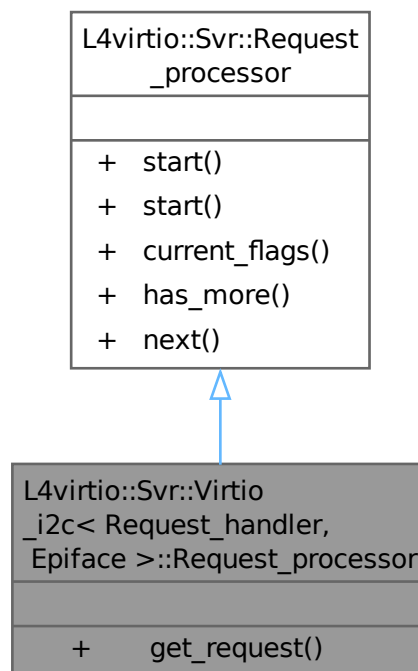
- [l4/l4virtio/server/virtio-i2c-device](#)

15.437 L4virtio::Svr::Virtio_i2c< Request_handler, Epiface >::Request_processor Class Reference

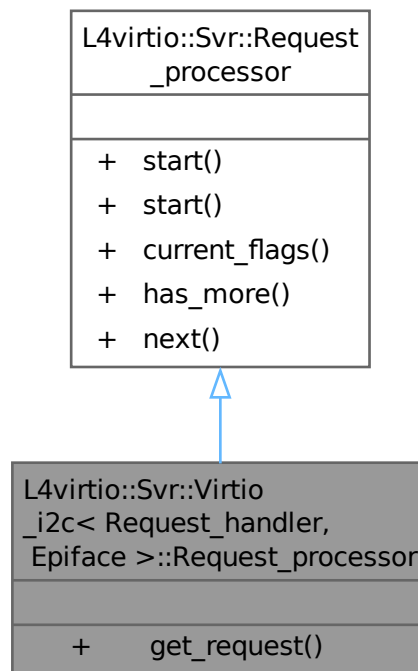
Handler for the Virtio requests.

```
#include <virtio-i2c-device>
```

Inheritance diagram for L4virtio::Svr::Virtio_i2c< Request_handler, Epiface >::Request_processor:



Collaboration diagram for L4virtio::Svr::Virtio_i2c< Request_handler, Epiface >::Request_processor:



Public Member Functions

- I2c_req [get_request](#) ()

Linux prepares the I2C request in three data parts: 1st: out_hdr 2nd: buffer (optional) 3rd: in_hdr.

Public Member Functions inherited from [L4virtio::Svr::Request_processor](#)

- template<typename DESC_MAN, typename ... ARGS>
void [start](#) (DESC_MAN *dm, Virtqueue *ring, [Virtqueue::Head_desc](#) const &request, ARGS... args)
Start processing a new request.
- template<typename DESC_MAN, typename ... ARGS>
Virtqueue::Request const & [start](#) (DESC_MAN *dm, Virtqueue::Request const &request, ARGS... args)
Start processing a new request.
- [Virtqueue::Desc::Flags](#) [current_flags](#) () const
Get the flags of the currently processed descriptor.
- bool [has_more](#) () const
Are there more chained descriptors?
- template<typename DESC_MAN, typename ... ARGS>
bool [next](#) (DESC_MAN *dm, ARGS... args)
Switch to the next descriptor in a descriptor chain.

15.437.1 Detailed Description

```
template<typename Request_handler, typename Epiface = L4virtio::Device>
class L4virtio::Svr::Virtio_i2c< Request_handler, Epiface >::Request_processor
```

Handler for the Virtio requests.

Definition at line 126 of file [virtio-i2c-device](#).

15.437.2 Member Function Documentation

15.437.2.1 get_request()

```
template<typename Request_handler, typename Epiface = L4virtio::Device>
I2c_req L4virtio::Svr::Virtio_i2c< Request_handler, Epiface >::Request_processor::get_request
() [inline]
```

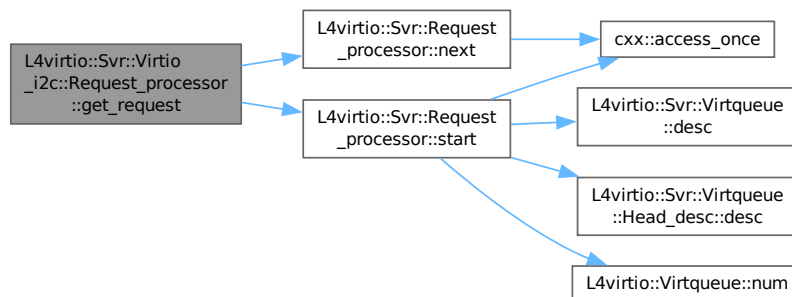
Linux prepares the I2C request in three data parts: 1st: out_hdr 2nd: buffer (optional) 3rd: in_hdr.

This parses the three Data_buffers and recreate the virtio_i2c_req structure.

Definition at line 162 of file [virtio-i2c-device](#).

References [L4_UNLIKELY](#), [L4virtio::Svr::Data_buffer::left](#), [L4virtio::Svr::Request_processor::next\(\)](#), [L4virtio::Svr::Data_buffer::pos](#), and [L4virtio::Svr::Request_processor::start\(\)](#).

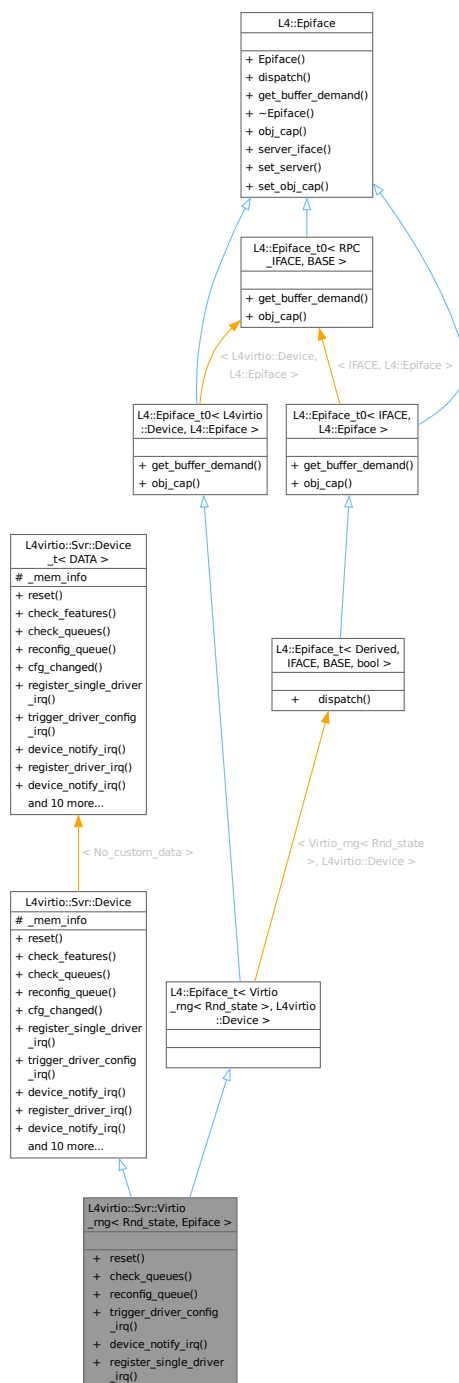
Here is the call graph for this function:



The documentation for this class was generated from the following file:

- `I4/I4virtio/server/virtio-i2c-device`

Collaboration diagram for L4virtio::Svr::Virtio_rng< Rnd_state, Epiface >:



Data Structures

- class `Host_irq`
Handler for the host irq.
- class `Request_processor`
Handler for the Virtio requests.

Public Member Functions

- void **reset** () override
reset callback, called for doing a device reset
- bool **check_queues** () override
callback for checking if the queues at DRIVER_OK transition
- int **reconfig_queue** (unsigned idx) override
callback for client queue-config request
- void **trigger_driver_config_irq** () override
callback for triggering configuration change notification IRQ
- [L4::Cap](#) < [L4::Irq](#) > **device_notify_irq** () const override
callback to gather the device notification IRQ (old-style)
- void **register_single_driver_irq** () override
callback for registering a single guest IRQ for all queues (old-style)

Public Member Functions inherited from [L4virtio::Svr::Device_t](#) < [No_custom_data](#) >

- virtual bool **check_features** ()
callback for checking the subset of accepted features
- virtual void **cfg_changed** (unsigned)
callback for client device configuration changes
- virtual void **register_driver_irq** (unsigned idx)
Callback for registering an notification IRQ (multi IRQ).
- virtual [L4::Cap](#) < [L4::Irq](#) > **device_notify_irq** (unsigned idx)
Callback to gather the device notification IRQ (multi IRQ).
- virtual unsigned **num_events_supported** () const
Return the highest notification index supported.
- **Device_t** ([Dev_config](#) *dev_config)
Make a device for the given config.
- Mem_list const * **mem_info** () const
Get the memory region list used for this device.
- void **reset_queue_config** (unsigned idx, unsigned num_max, bool inc_generation=false, unsigned device_notify_index=0)
Trigger reset for the configuration space for queue idx.
- void **init_mem_info** (unsigned num)
Initialize the memory region list to the given maximum.
- void **device_error** ()
Transition device into DEVICE_NEEDS_RESET state.
- bool **setup_queue** (Virtqueue *q, unsigned qn, unsigned num_max)
Enable/disable the specified queue.
- bool **handle_mem_cmd_write** ()
Check for a value in the cmd register and handle a write.
- void **enable_trusted_ds_validation** ()
Enable trusted dataspace validation.
- void **add_trusted_dataspaces** (std::shared_ptr < [Ds_vector](#) const > ds)
Provide a list of trusted dataspaces that can be used for validation.

Public Member Functions inherited from L4::Epiface_t0< L4virtio::Device, L4::Epiface >

- [Type_info::Demand](#) **get_buffer_demand** () const
Get the server-side buffer demand based in IFACE.
- [Cap< L4virtio::Device >](#) **obj_cap** () const
Get the (typed) capability to this object.

Public Member Functions inherited from L4::Epiface

- **Epiface** ()
Make a server object.
- virtual **~Epiface** ()=0
Destroy the object.
- Stored_cap [obj_cap](#) () const
Get the capability to the kernel object belonging to this object.
- [Server_iface](#) * **server_iface** () const
Get pointer to server interface at which the object is currently registered.
- int **set_server** ([Server_iface](#) *srv, [Cap< void >](#) cap, bool managed=false)
Set server registration info for the object.
- void **set_obj_cap** ([Cap< void >](#) const &cap)
Deprecated server registration function.

Additional Inherited Members**Public Types inherited from L4::Epiface_t0< L4virtio::Device, L4::Epiface >**

- using **Interface**
Data type of the IPC interface definition.

Public Types inherited from L4::Epiface

- using **Server_iface** = [lpc_svr::Server_iface](#)
Type for abstract server interface.
- using **Demand** = [lpc_svr::Server_iface::Demand](#)
Type for server-side receive buffer demand.

Protected Attributes inherited from L4virtio::Svr::Device_t< No_custom_data >

- **Mem_list_mem_info**
Memory region list.

15.438.1 Detailed Description

```
template<typename Rnd_state, typename Epiface = L4virtio::Device>
class L4virtio::Svr::Virtio_rng< Rnd_state, Epiface >
```

A server implementation of the virtio-rng protocol.

Template Parameters

| | |
|------------------|---|
| <i>Rnd_state</i> | The type that implements the random data generation. <code>Rnd_state::get_random(int len, unsigned char *buf)</code> is called to get len random bytes written into buf TODO: virtio-rng supports providing less random bytes then requested. This API currently does not support that, as I do not have a test case. |
| <i>Epiface</i> | The Epiface to derive from. Defaults to <code>L4virtio::Device</code> . |

Definition at line 33 of file [virtio-rng-device](#).

The documentation for this class was generated from the following file:

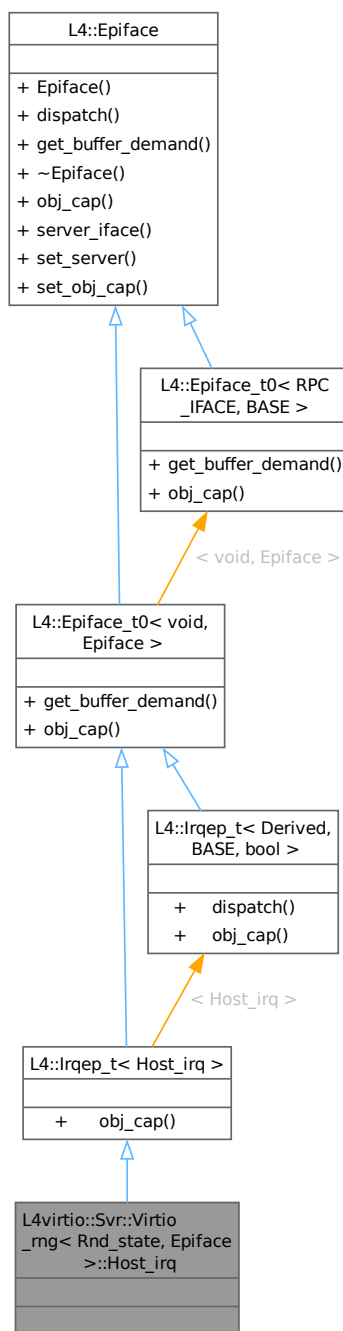
- `l4/l4virtio/server/virtio-rng-device`

15.439 L4virtio::Svr::Virtio_rng< Rnd_state, Epiface >::Host_irq Class Reference

Handler for the host irq.

```
#include <virtio-rng-device>
```

Inheritance diagram for L4virtio::Svr::Virtio_rng< Rnd_state, Epiface >::Host_irq:



Public Types inherited from L4::Epiface

- using **Server_iface** = [lpc_svr::Server_iface](#)
Type for abstract server interface.
- using **Demand** = [lpc_svr::Server_iface::Demand](#)
Type for server-side receive buffer demand.

Public Member Functions inherited from L4::lrqep_t< Host_irq >

- [Cap< L4::lrq > obj_cap](#) () const
Get the (typed) capability to this object.

Public Member Functions inherited from L4::Epiface_t0< void, Epiface >

- [Type_info::Demand get_buffer_demand](#) () const
Get the server-side buffer demand based in IFACE.
- [Cap< void > obj_cap](#) () const
Get the (typed) capability to this object.

Public Member Functions inherited from L4::Epiface

- **Epiface** ()
Make a server object.
- virtual **~Epiface** ()=0
Destroy the object.
- Stored_cap [obj_cap](#) () const
Get the capability to the kernel object belonging to this object.
- [Server_iface * server_iface](#) () const
Get pointer to server interface at which the object is currently registered.
- int [set_server](#) ([Server_iface *srv](#), [Cap< void > cap](#), bool managed=false)
Set server registration info for the object.
- void [set_obj_cap](#) ([Cap< void > const &cap](#))
Deprecated server registration function.

15.439.1 Detailed Description

```
template<typename Rnd_state, typename Epiface = L4virtio::Device>
class L4virtio::Svr::Virtio_rng< Rnd_state, Epiface >::Host_irq
```

Handler for the host irq.

An [L4::lrqep_t](#) to handle irq's send to the server.

Definition at line 51 of file [virtio-rng-device](#).

The documentation for this class was generated from the following file:

- [l4/l4virtio/server/virtio-rng-device](#)

15.440 L4virtio::Svr::Virtio_rng< Rnd_state, Epiface >::Request_processor Class Reference

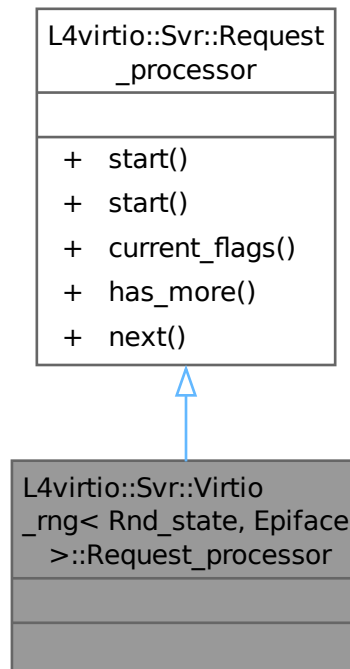
Handler for the Virtio requests.

```
#include <virtio-rng-device>
```

Inheritance diagram for L4virtio::Svr::Virtio_rng< Rnd_state, Epiface >::Request_processor:



Collaboration diagram for L4virtio::Svr::Virtio_rng< Rnd_state, Epiface >::Request_processor:



Additional Inherited Members

Public Member Functions inherited from [L4virtio::Svr::Request_processor](#)

- `template<typename DESC_MAN, typename ... ARGS>`
`void start (DESC_MAN *dm, Virtqueue *ring, Virtqueue::Head_desc const &request, ARGS... args)`
Start processing a new request.
- `template<typename DESC_MAN, typename ... ARGS>`
`Virtqueue::Request const & start (DESC_MAN *dm, Virtqueue::Request const &request, ARGS... args)`
Start processing a new request.
- `Virtqueue::Desc::Flags current_flags () const`
Get the flags of the currently processed descriptor.
- `bool has_more () const`
Are there more chained descriptors?
- `template<typename DESC_MAN, typename ... ARGS>`
`bool next (DESC_MAN *dm, ARGS... args)`
Switch to the next descriptor in a descriptor chain.

15.440.1 Detailed Description

```
template<typename Rnd_state, typename Epiface = L4virtio::Device>  
class L4virtio::Svr::Virtio_rng< Rnd_state, Epiface >::Request_processor
```

Handler for the Virtio requests.

Definition at line 68 of file [virtio-rng-device](#).

The documentation for this class was generated from the following file:

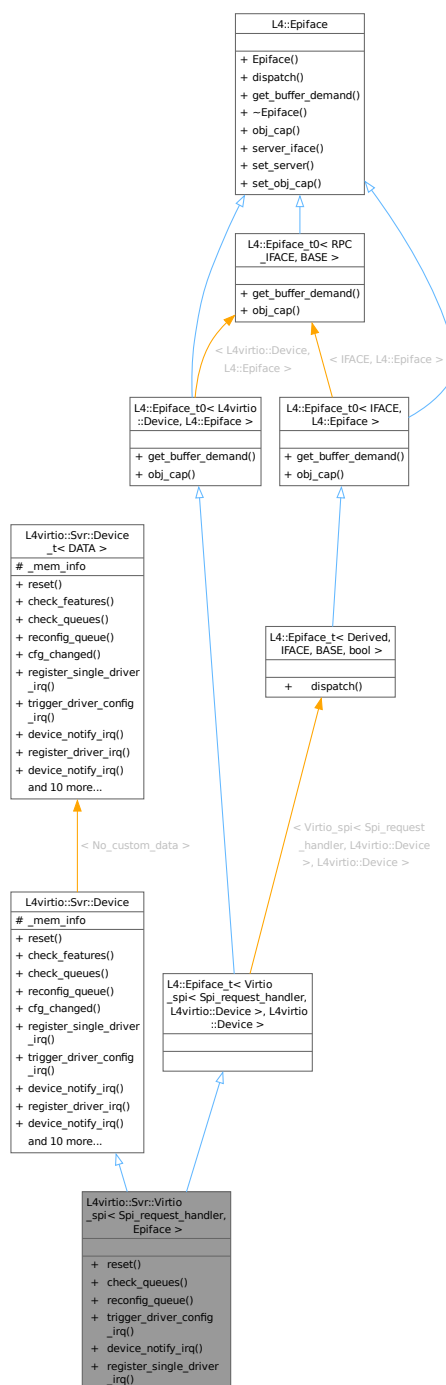
- I4/I4virtio/server/virtio-rng-device

15.441 L4virtio::Svr::Virtio_spi< Spi_request_handler, Epiface > Class Template Reference

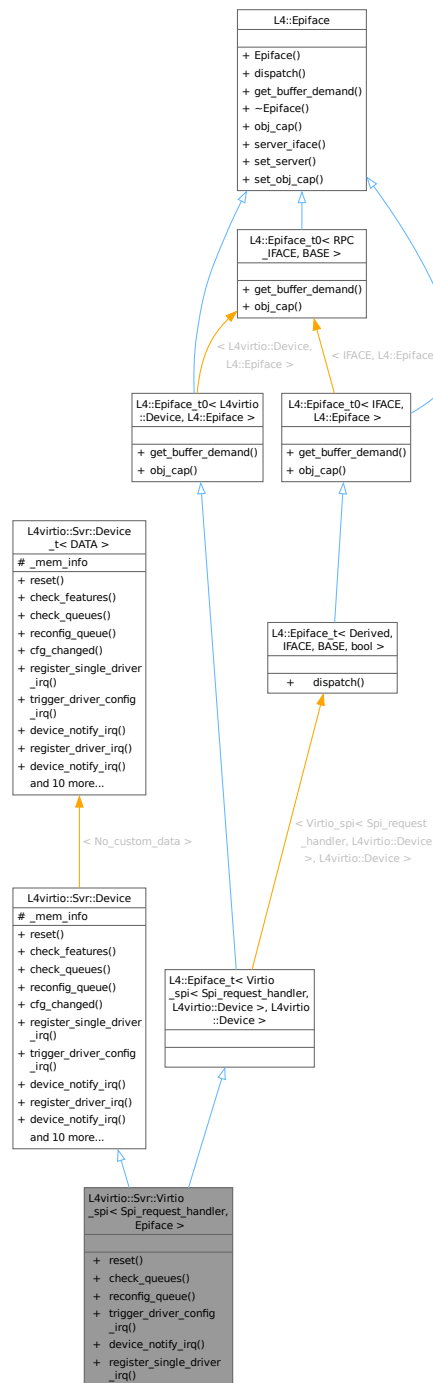
A server implementation of the virtio-spi protocol.

```
#include <virtio-spi-device>
```


Inheritance diagram for L4virtio::Svr::Virtio_spi< Spi_request_handler, Epiface >:



Collaboration diagram for L4virtio::Svr::Virtio_spi< Spi_request_handler, Epiface >:



Data Structures

- class [Host_irq](#)
Handler for the host IRQ.
- class [Request_processor](#)
Handler for the Virtio requests.

Public Member Functions

- void **reset** () override
reset callback, called for doing a device reset
- bool **check_queues** () override
callback for checking if the queues at DRIVER_OK transition
- int **reconfig_queue** (unsigned idx) override
callback for client queue-config request
- void **trigger_driver_config_irq** () override
callback for triggering configuration change notification IRQ
- L4::Cap< L4::Irq > **device_notify_irq** () const override
callback to gather the device notification IRQ (old-style)
- void **register_single_driver_irq** () override
callback for registering a single guest IRQ for all queues (old-style)

Public Member Functions inherited from L4virtio::Svr::Device_t< No_custom_data >

- virtual bool **check_features** ()
callback for checking the subset of accepted features
- virtual void **cfg_changed** (unsigned)
callback for client device configuration changes
- virtual void **register_driver_irq** (unsigned idx)
Callback for registering an notification IRQ (multi IRQ).
- virtual L4::Cap< L4::Irq > **device_notify_irq** (unsigned idx)
Callback to gather the device notification IRQ (multi IRQ).
- virtual unsigned **num_events_supported** () const
Return the highest notification index supported.
- **Device_t** (Dev_config *dev_config)
Make a device for the given config.
- Mem_list const * **mem_info** () const
Get the memory region list used for this device.
- void **reset_queue_config** (unsigned idx, unsigned num_max, bool inc_generation=false, unsigned device_notify_index=0)
Trigger reset for the configuration space for queue idx.
- void **init_mem_info** (unsigned num)
Initialize the memory region list to the given maximum.
- void **device_error** ()
Transition device into DEVICE_NEEDS_RESET state.
- bool **setup_queue** (Virtqueue *q, unsigned qn, unsigned num_max)
Enable/disable the specified queue.
- bool **handle_mem_cmd_write** ()
Check for a value in the cmd register and handle a write.
- void **enable_trusted_ds_validation** ()
Enable trusted dataspace validation.
- void **add_trusted_dataspaces** (std::shared_ptr< Ds_vector const > ds)
Provide a list of trusted dataspaces that can be used for validation.

Public Member Functions inherited from [L4::Epiface_t0< L4virtio::Device, L4::Epiface >](#)

- [Type_info::Demand](#) **get_buffer_demand** () const
Get the server-side buffer demand based in IFACE.
- [Cap< L4virtio::Device >](#) **obj_cap** () const
Get the (typed) capability to this object.

Public Member Functions inherited from [L4::Epiface](#)

- **Epiface** ()
Make a server object.
- virtual **~Epiface** ()=0
Destroy the object.
- Stored_cap [obj_cap](#) () const
Get the capability to the kernel object belonging to this object.
- [Server_iface](#) * **server_iface** () const
Get pointer to server interface at which the object is currently registered.
- int **set_server** ([Server_iface](#) *srv, [Cap< void >](#) cap, bool managed=false)
Set server registration info for the object.
- void **set_obj_cap** ([Cap< void >](#) const &cap)
Deprecated server registration function.

Additional Inherited Members

Public Types inherited from [L4::Epiface_t0< L4virtio::Device, L4::Epiface >](#)

- using **Interface**
Data type of the IPC interface definition.

Public Types inherited from [L4::Epiface](#)

- using **Server_iface** = [lpc_svr::Server_iface](#)
Type for abstract server interface.
- using **Demand** = [lpc_svr::Server_iface::Demand](#)
Type for server-side receive buffer demand.

Protected Attributes inherited from [L4virtio::Svr::Device_t< No_custom_data >](#)

- **Mem_list_mem_info**
Memory region list.

15.441.1 Detailed Description

```
template<typename Spi_request_handler, typename Epiface = L4virtio::Device>
class L4virtio::Svr::Virtio_spi< Spi_request_handler, Epiface >
```

A server implementation of the virtio-spi protocol.

Template Parameters

| | |
|----------------------------|---|
| <i>Spi_request_handler</i> | The type that is used to handle incoming requests. Needs to have: <ul style="list-style-type: none">• <code>handle_transfer(Spi_transfer_head const &, l4_uint8_t const *, l4_uint8_t *, unsigned) function.</code> |
| <i>Epiface</i> | The Epiface to derive from. Defaults to L4virtio::Device . |

Definition at line [101](#) of file [virtio-spi-device](#).

The documentation for this class was generated from the following file:

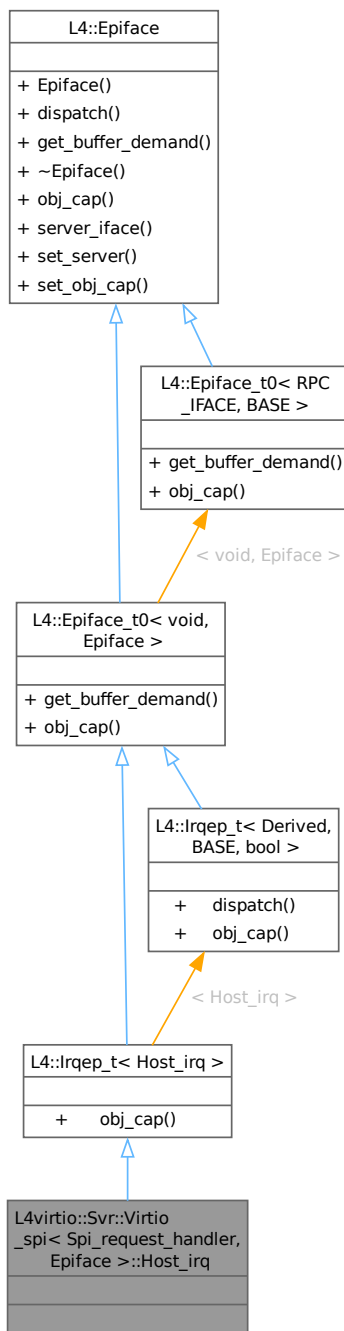
- `l4/l4virtio/server/virtio-spi-device`

15.442 L4virtio::Svr::Virtio_spi< Spi_request_handler, Epiface >::Host_irq Class Reference

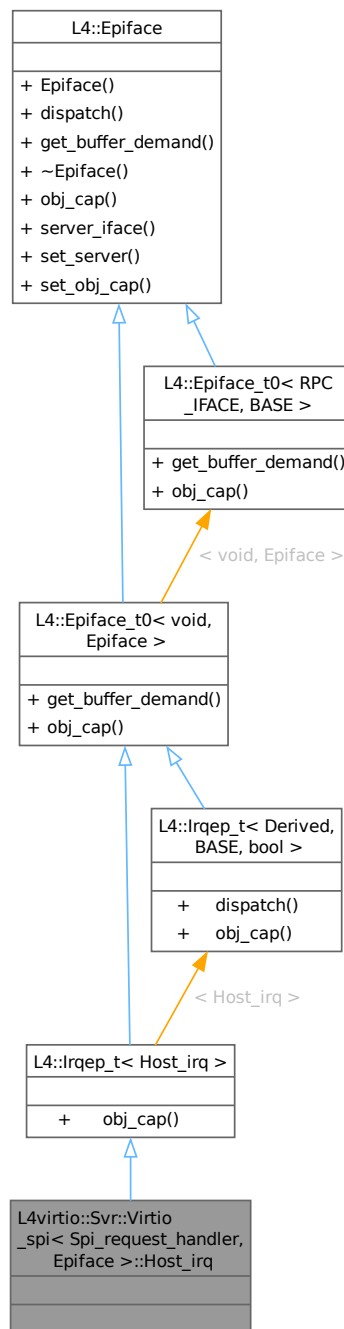
Handler for the host IRQ.

```
#include <virtio-spi-device>
```

Inheritance diagram for L4virtio::Svr::Virtio_spi< Spi_request_handler, Epiface >::Host_irq:



Collaboration diagram for L4virtio::Svr::Virtio_spi< Spi_request_handler, Epiface >::Host_irq:



Additional Inherited Members

Public Types inherited from L4::Epiface_t0< void, Epiface >

- using **Interface**

Data type of the IPC interface definition.

Public Types inherited from [L4::Epiface](#)

- using **Server_iface** = [lpc_svr::Server_iface](#)
Type for abstract server interface.
- using **Demand** = [lpc_svr::Server_iface::Demand](#)
Type for server-side receive buffer demand.

Public Member Functions inherited from [L4::lrqep_t< Host_irq >](#)

- [Cap< L4::lrq > obj_cap](#) () const
Get the (typed) capability to this object.

Public Member Functions inherited from [L4::Epiface_t0< void, Epiface >](#)

- [Type_info::Demand get_buffer_demand](#) () const
Get the server-side buffer demand based in IFACE.
- [Cap< void > obj_cap](#) () const
Get the (typed) capability to this object.

Public Member Functions inherited from [L4::Epiface](#)

- **Epiface** ()
Make a server object.
- virtual **~Epiface** ()=0
Destroy the object.
- Stored_cap [obj_cap](#) () const
Get the capability to the kernel object belonging to this object.
- [Server_iface * server_iface](#) () const
Get pointer to server interface at which the object is currently registered.
- int [set_server](#) ([Server_iface *srv](#), [Cap< void > cap](#), bool managed=false)
Set server registration info for the object.
- void [set_obj_cap](#) ([Cap< void > const &cap](#))
Deprecated server registration function.

15.442.1 Detailed Description

```
template<typename Spi_request_handler, typename Epiface = L4virtio::Device>
class L4virtio::Svr::Virtio_spi< Spi_request_handler, Epiface >::Host_irq
```

Handler for the host IRQ.

An [L4::lrqep_t](#) to handle irq's send to the server.

Definition at line 118 of file [virtio-spi-device](#).

The documentation for this class was generated from the following file:

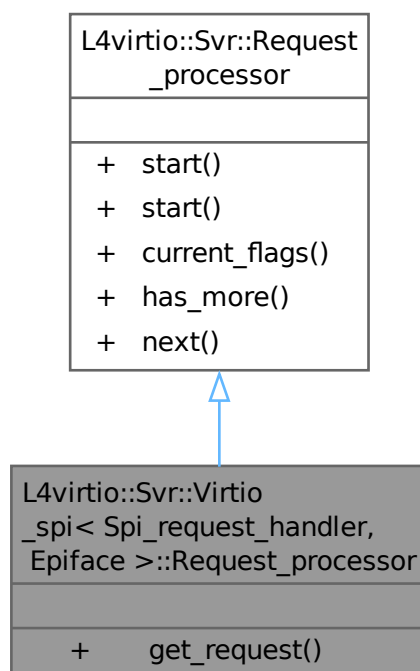
- I4/I4virtio/server/virtio-spi-device

15.443 L4virtio::Svr::Virtio_spi< Spi_request_handler, Epiface >::Request_processor Class Reference

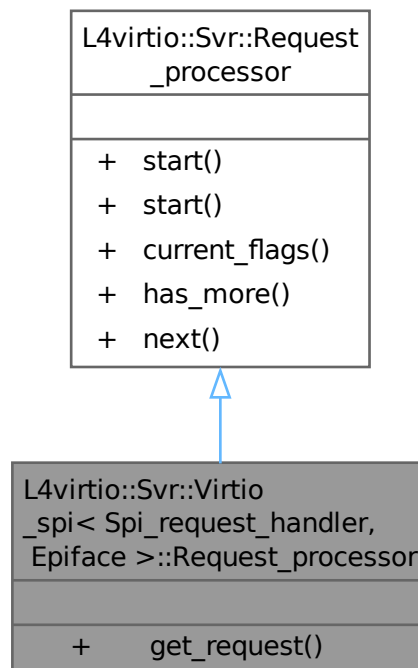
Handler for the Virtio requests.

```
#include <virtio-spi-device>
```

Inheritance diagram for L4virtio::Svr::Virtio_spi< Spi_request_handler, Epiface >::Request_processor:



Collaboration diagram for L4virtio::Svr::Virtio_spi< Spi_request_handler, Epiface >::Request_processor:



Public Member Functions

- Spi_transfer_req [get_request](#) ()

Linux prepares the SPI request in three or four data parts: 1st: transfer_head 2nd: TX buffer (not present for RX half-duplex transfers) 3rd: RX buffer (not present for TX half-duplex transfers) 4th: transfer result.

Public Member Functions inherited from [L4virtio::Svr::Request_processor](#)

- template<typename DESC_MAN, typename ... ARGS>
void [start](#) (DESC_MAN *dm, Virtqueue *ring, [Virtqueue::Head_desc](#) const &request, ARGS... args)
Start processing a new request.
- template<typename DESC_MAN, typename ... ARGS>
Virtqueue::Request const & [start](#) (DESC_MAN *dm, Virtqueue::Request const &request, ARGS... args)
Start processing a new request.
- [Virtqueue::Desc::Flags](#) [current_flags](#) () const
Get the flags of the currently processed descriptor.
- bool [has_more](#) () const
Are there more chained descriptors?
- template<typename DESC_MAN, typename ... ARGS>
bool [next](#) (DESC_MAN *dm, ARGS... args)
Switch to the next descriptor in a descriptor chain.

15.443.1 Detailed Description

```
template<typename Spi_request_handler, typename Epiface = L4virtio::Device>
class L4virtio::Svr::Virtio_spi< Spi_request_handler, Epiface >::Request_processor
```

Handler for the Virtio requests.

Definition at line 135 of file [virtio-spi-device](#).

15.443.2 Member Function Documentation

15.443.2.1 get_request()

```
template<typename Spi_request_handler, typename Epiface = L4virtio::Device>
Spi_transfer_req L4virtio::Svr::Virtio_spi< Spi_request_handler, Epiface >::Request_processor↔
::get_request () [inline]
```

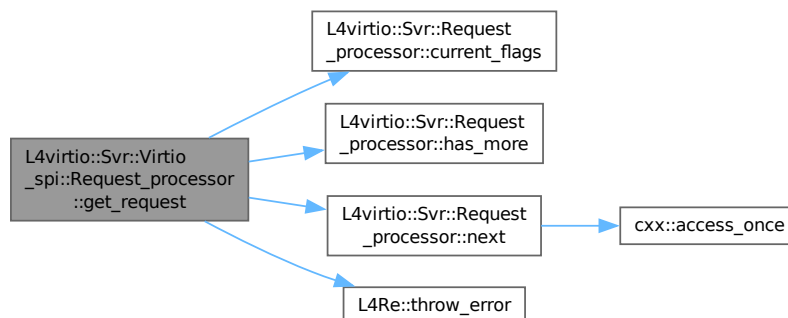
Linux prepares the SPI request in three or four data parts: 1st: transfer_head 2nd: TX buffer (not present for RX half-duplex transfers) 3rd: RX buffer (not present for TX half-duplex transfers) 4th: transfer result.

This parses the three/four Data_buffers and recreates the virtio_spi_transfer_req structure.

Definition at line 185 of file [virtio-spi-device](#).

References [L4virtio::Svr::Request_processor::current_flags\(\)](#), [L4virtio::Svr::Request_processor::has_more\(\)](#), [L4_EIO](#), [L4virtio::Svr::Request_processor::next\(\)](#), and [L4Re::throw_error\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

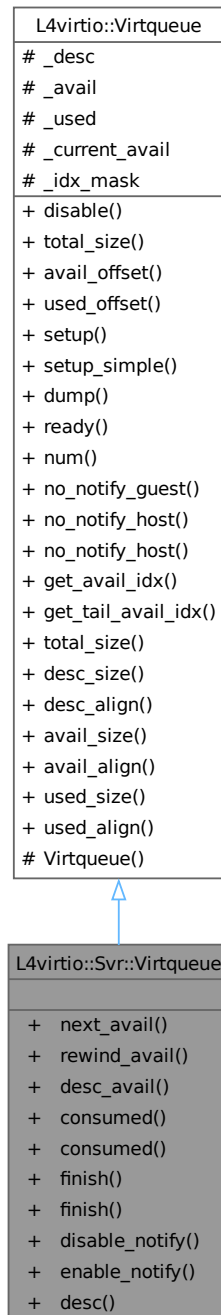
- `I4/I4virtio/server/virtio-spi-device`

15.444 L4virtio::Svr::Virtqueue Class Reference

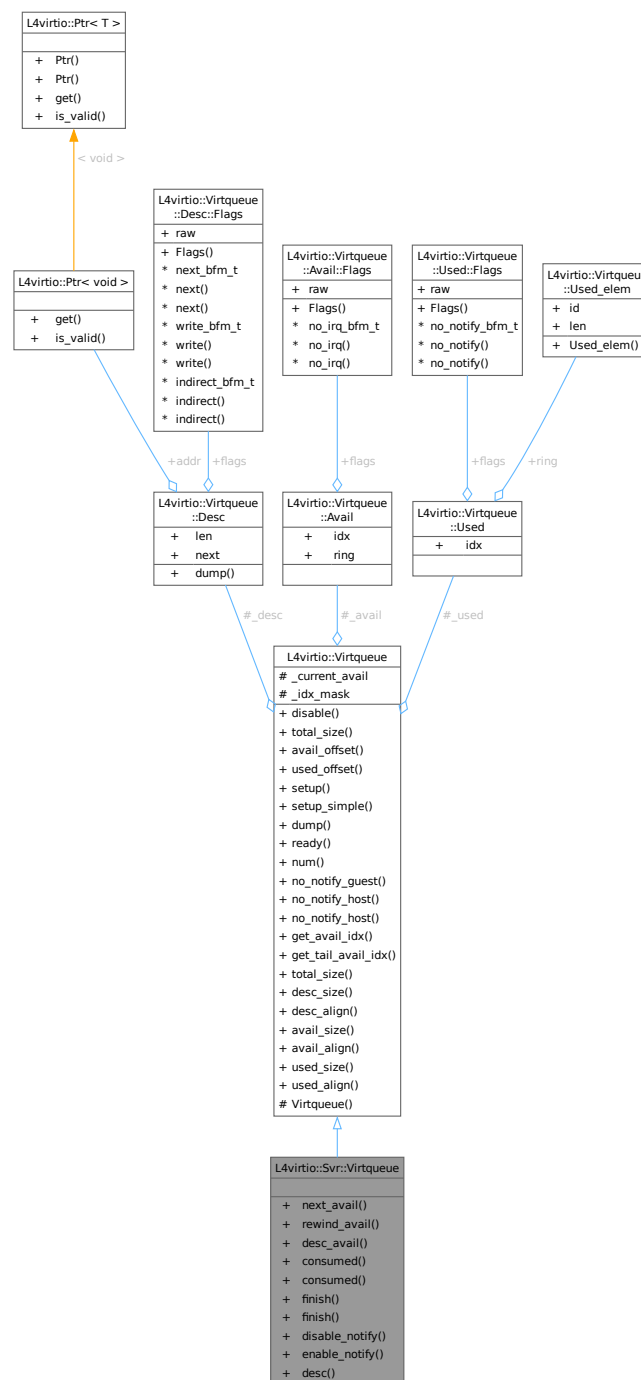
[Virtqueue](#) implementation for the device.

```
#include <virtio>
```

Inheritance diagram for L4virtio::Svr::Virtqueue:



Collaboration diagram for L4virtio::Svr::Virtqueue:



Data Structures

- class [Head_desc](#)
VIRTIO request, essentially a descriptor from the available ring.

Public Member Functions

- Request [next_avail](#) ()

- Get the next available descriptor from the available ring.*

 - void `rewind_avail` (`Head_desc` const &d)

Return unfinished descriptors to the available ring, i.e.

 - bool `desc_avail` () const
- Test for available descriptors.*
- void `consumed` (`Head_desc` const &r, `l4_uint32_t` len=0)
- Put the given descriptor into the used ring.*
- template<typename ITER>
void `consumed` (ITER const &begin, ITER const &end)
- Put multiple descriptors into the used ring.*
- template<typename QUEUE_OBSERVER>
void `finish` (`Head_desc` &d, QUEUE_OBSERVER *o, `l4_uint32_t` len=0)
- Add a descriptor to the used ring, and notify an observer.*
- template<typename ITER, typename QUEUE_OBSERVER>
void `finish` (ITER const &begin, ITER const &end, QUEUE_OBSERVER *o)
- Add a range of descriptors to the used ring, and notify an observer once.*
- void `disable_notify` ()
- Set the 'no notify' flag for this queue.*
- void `enable_notify` ()
- Clear the 'no notify' flag for this queue.*
- `Desc` const * `desc` (unsigned idx) const
- Get a descriptor from the descriptor list.*

Public Member Functions inherited from `L4virtio::Virtqueue`

- void `disable` ()
- Completely disable the queue.*
- unsigned long `total_size` () const
- Calculate the total size of this virtqueue.*
- unsigned long `avail_offset` () const
- Get the offset of the available ring from the descriptor table.*
- unsigned long `used_offset` () const
- Get the offset of the used ring from the descriptor table.*
- void `setup` (unsigned `num`, void *desc, void *avail, void *used)
- Enable this queue.*
- void `setup_simple` (unsigned `num`, void *ring)
- Enable this queue.*
- void `dump` (`Desc` const *d) const
- Dump descriptors for this queue.*
- bool `ready` () const
- Test if this queue is in working state.*
- unsigned `num` () const
- bool `no_notify_guest` () const
- Get the no IRQ flag of this queue.*
- bool `no_notify_host` () const
- Get the no notify flag of this queue.*
- void `no_notify_host` (bool value)
- Set the no-notify flag for this queue.*
- `l4_uint16_t` `get_avail_idx` () const
- Get available index from available ring (for debugging).*
- `l4_uint16_t` `get_tail_avail_idx` () const
- Get tail-available index stored in local state (for debugging).*

Additional Inherited Members

Public Types inherited from [L4virtio::Virtqueue](#)

- enum
Fixed alignment values for different parts of a virtqueue.

Static Public Member Functions inherited from [L4virtio::Virtqueue](#)

- static unsigned long [total_size](#) (unsigned [num](#))
Calculate the total size for a virtqueue of the given dimensions.
- static unsigned long [desc_size](#) (unsigned [num](#))
Calculate the size of the descriptor table for [num](#) entries.
- static unsigned long [desc_align](#) ()
Get the alignment in zero LSBs needed for the descriptor table.
- static unsigned long [avail_size](#) (unsigned [num](#))
Calculate the size of the available ring for [num](#) entries.
- static unsigned long [avail_align](#) ()
Get the alignment in zero LSBs needed for the available ring.
- static unsigned long [used_size](#) (unsigned [num](#))
Calculate the size of the used ring for [num](#) entries.
- static unsigned long [used_align](#) ()
Get the alignment in zero LSBs needed for the used ring.

Protected Member Functions inherited from [L4virtio::Virtqueue](#)

- [Virtqueue](#) ()=default
Create a disabled virtqueue.

Protected Attributes inherited from [L4virtio::Virtqueue](#)

- [Desc](#) * [_desc](#) = nullptr
pointer to descriptor table, NULL if queue is off.
- [Avail](#) * [_avail](#) = nullptr
pointer to available ring.
- [Used](#) * [_used](#) = nullptr
pointer to used ring.
- [l4_uint16_t](#) [_current_avail](#) = 0
The life counter for the queue.
- [l4_uint16_t](#) [_idx_mask](#) = 0
mask used for indexing into the descriptor table and the rings.

15.444.1 Detailed Description

[Virtqueue](#) implementation for the device.

This class represents a single virtqueue, with a local running available index.

Note

The [Virtqueue](#) implementation is not thread-safe.

Definition at line 87 of file [virtio](#).

15.444.2 Member Function Documentation

15.444.2.1 consumed() [1/2]

```
void L4virtio::Svr::Virtqueue::consumed (
    Head_desc const & r,
    14_uint32_t len = 0) [inline]
```

Put the given descriptor into the used ring.

Parameters

| | |
|------------|---|
| <i>r</i> | Request that shall be marked as finished. |
| <i>len</i> | The total number of bytes written. |

Precondition

queue must be in working state.

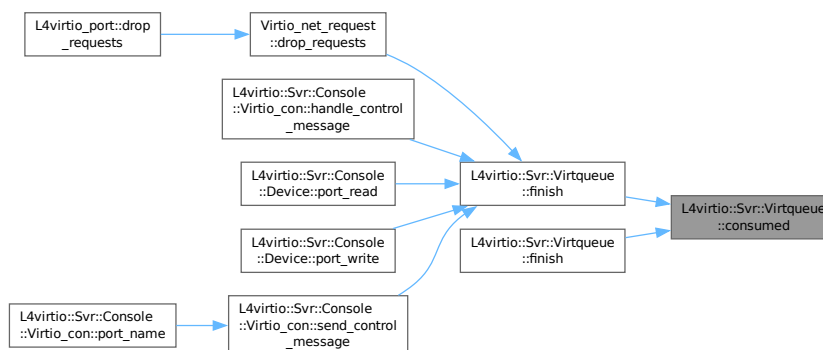
r must be a valid request from this queue.

Definition at line 190 of file [virtio](#).

References [L4virtio::Virtqueue::_desc](#), [L4virtio::Virtqueue::_idx_mask](#), and [L4virtio::Virtqueue::_used](#).

Referenced by [finish\(\)](#), and [finish\(\)](#).

Here is the caller graph for this function:



15.444.2.2 consumed() [2/2]

```
template<typename ITER>
void L4virtio::Svr::Virtqueue::consumed (
    ITER const & begin,
    ITER const & end) [inline]
```

Put multiple descriptors into the used ring.

A range of descriptors, specified by *begin* and *end* iterators is added. Each iterator points to a struct that has a *first* member that is a [Head_desc](#) and a *second* member that is the corresponding number of bytes written.

Template Parameters

| | |
|-------------|--------------------------------------|
| <i>ITER</i> | The type of the iterator (inferred). |
|-------------|--------------------------------------|

Parameters

| | |
|--------------|--|
| <i>begin</i> | Iterator pointing to first new descriptor. |
| <i>end</i> | Iterator pointing to one past last entry. |

Precondition

queue must be in working state.

Definition at line 213 of file [virtio](#).

References [L4virtio::Virtqueue::_desc](#), [L4virtio::Virtqueue::_idx_mask](#), and [L4virtio::Virtqueue::_used](#).

15.444.2.3 desc()

```
Desc const * L4virtio::Svr::Virtqueue::desc (
    unsigned idx) const [inline]
```

Get a descriptor from the descriptor list.

Parameters

| | |
|------------|------------------------------|
| <i>idx</i> | The index of the descriptor. |
|------------|------------------------------|

Precondition

`idx < num`

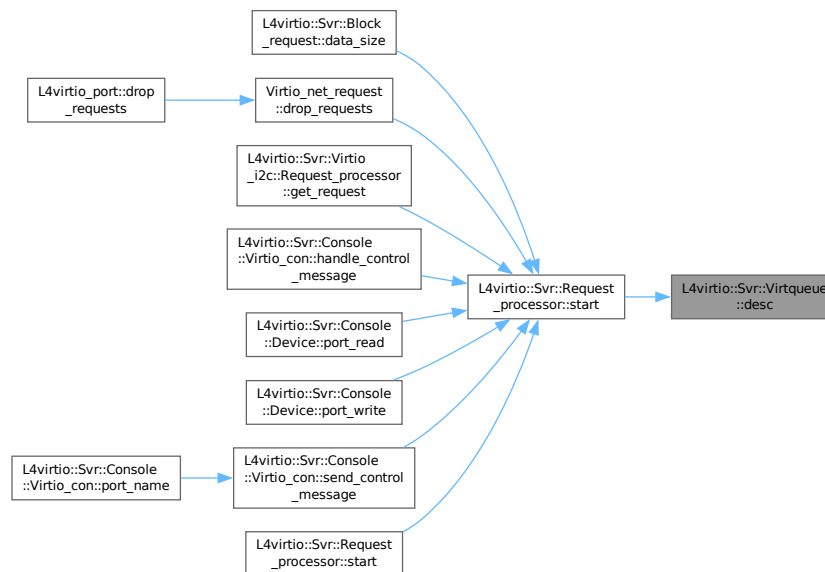
queue must be in working state

Definition at line 298 of file [virtio](#).

References [L4virtio::Virtqueue::_desc](#).

Referenced by [L4virtio::Svr::Request_processor::start\(\)](#).

Here is the caller graph for this function:

**15.444.2.4 desc_avail()**

```
bool L4virtio::Svr::Virtqueue::desc_avail () const [inline]
```

Test for available descriptors.

Returns

true if there are descriptors available, false if not.

Precondition

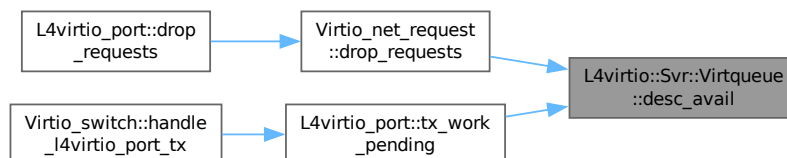
The queue must be in working state.

Definition at line 175 of file [virtio](#).

References [L4virtio::Virtqueue::_avail](#), and [L4virtio::Virtqueue::_current_avail](#).

Referenced by [Virtio_net_request::drop_requests\(\)](#), and [L4virtio_port::tx_work_pending\(\)](#).

Here is the caller graph for this function:

**15.444.2.5 disable_notify()**

```
void L4virtio::Svr::Virtqueue::disable_notify () [inline]
```

Set the 'no notify' flag for this queue.

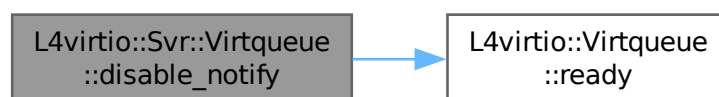
This function may be called on a disabled queue.

Definition at line 273 of file [virtio](#).

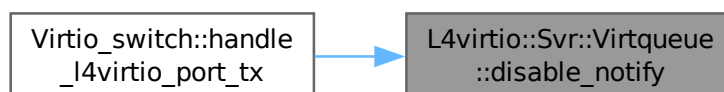
References [L4virtio::Virtqueue::_used](#), [L4_LIKELY](#), and [L4virtio::Virtqueue::ready\(\)](#).

Referenced by [Virtio_switch::handle_l4virtio_port_tx\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.444.2.6 enable_notify()

```
void L4virtio::Svr::Virtqueue::enable_notify () [inline]
```

Clear the 'no notify' flag for this queue.

This function may be called on a disabled queue.

Definition at line 284 of file [virtio](#).

References [L4virtio::Virtqueue::_used](#), [L4_LIKELY](#), and [L4virtio::Virtqueue::ready\(\)](#).

Referenced by [Virtio_switch::handle_l4virtio_port_tx\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.444.2.7 finish() [1/2]

```
template<typename QUEUE_OBSERVER>
void L4virtio::Svr::Virtqueue::finish (
    Head_desc & d,
    QUEUE_OBSERVER * o,
    l4_uint32_t len = 0) [inline]
```

Add a descriptor to the used ring, and notify an observer.

Template Parameters

| | |
|-----------------------------|--------------------------------------|
| <code>QUEUE_OBSERVER</code> | The type of the observer (inferred). |
|-----------------------------|--------------------------------------|

Parameters

| | |
|------------|---|
| <i>d</i> | descriptor of the request that is to be marked as finished. |
| <i>o</i> | Pointer to the observer that is notified. |
| <i>len</i> | Number of bytes written for this request. |

Precondition

queue must be in working state.

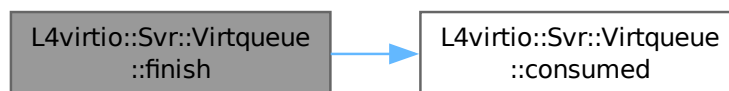
d must be a valid request from this queue.

Definition at line 240 of file [virtio](#).

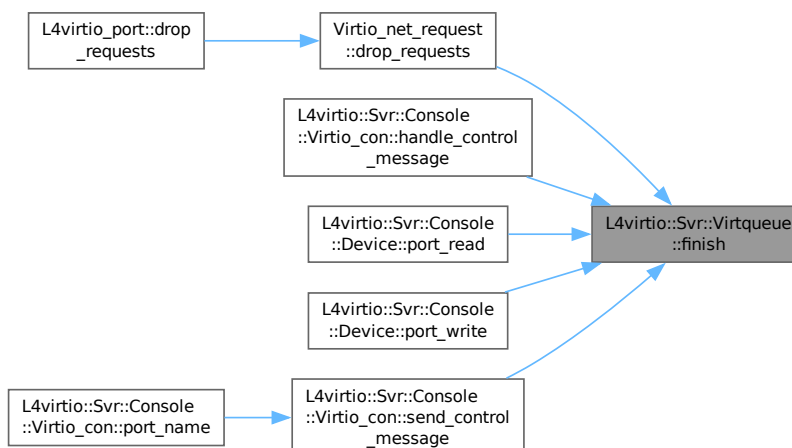
References [consumed\(\)](#).

Referenced by [Virtio_net_request::drop_requests\(\)](#), [L4virtio::Svr::Console::Virtio_con::handle_control_message\(\)](#), [L4virtio::Svr::Console::Device::port_read\(\)](#), [L4virtio::Svr::Console::Device::port_write\(\)](#), and [L4virtio::Svr::Console::Virtio_con::send_control_message\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.444.2.8 finish() [2/2]

```
template<typename ITER, typename QUEUE_OBSERVER>
void L4virtio::Svr::Virtqueue::finish (
    ITER const & begin,
    ITER const & end,
    QUEUE_OBSERVER * o) [inline]
```

Add a range of descriptors to the used ring, and notify an observer once.

The iterators are passed to [consumed<ITER>\(ITER const &, ITER const &\)](#), and the requirements detailed there apply.

Template Parameters

| | |
|-----------------------|--------------------------------------|
| <i>ITER</i> | type of the iterator (inferred) |
| <i>QUEUE_OBSERVER</i> | the type of the observer (inferred). |

Parameters

| | |
|--------------|---|
| <i>begin</i> | iterator pointing to first element. |
| <i>end</i> | iterator pointing to one past last element. |
| <i>o</i> | pointer to the observer that is notified. |

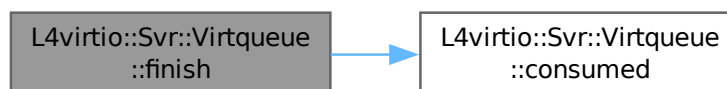
Precondition

queue must be in working state.

Definition at line [262](#) of file [virtio](#).

References [consumed\(\)](#).

Here is the call graph for this function:



15.444.2.9 next_avail()

Request L4virtio::Svr::Virtqueue::next_avail () [inline]

Get the next available descriptor from the available ring.

Precondition

The queue must be in working state.

Returns

A Request for the next available descriptor, the Request is invalid if there are no descriptors in the available ring.

Note

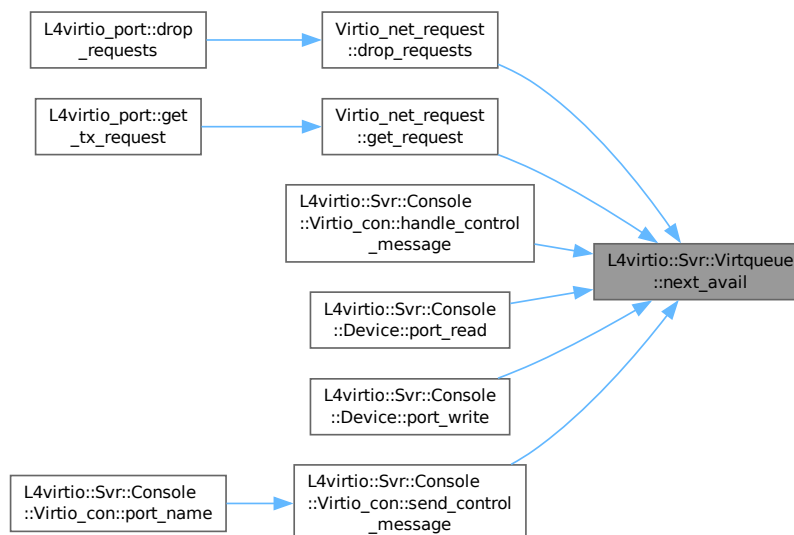
The return value must be checked even when a previous [desc_avail\(\)](#) returned true.

Definition at line 136 of file [virtio](#).

References [L4virtio::Virtqueue::_avail](#), [L4virtio::Virtqueue::_current_avail](#), [L4virtio::Virtqueue::_idx_mask](#), and [L4_LIKELY](#).

Referenced by [Virtio_net_request::drop_requests\(\)](#), [Virtio_net_request::get_request\(\)](#), [L4virtio::Svr::Console::Virtio_con::handle_control_message](#), [L4virtio::Svr::Console::Device::port_read\(\)](#), [L4virtio::Svr::Console::Device::port_write\(\)](#), and [L4virtio::Svr::Console::Virtio_con::send_control_message](#).

Here is the caller graph for this function:



15.444.2.10 `rewind_avail()`

```
void L4virtio::Svr::Virtqueue::rewind_avail (
    Head\_desc const & d) [inline]
```

Return unfinished descriptors to the available ring, i.e.

reset the local next index of the available ring to the given descriptor.

Parameters

| | |
|----------|---|
| <i>d</i> | descriptor of the request that is to be marked as finished. |
|----------|---|

Precondition

queue must be in working state.

d must be a valid request from this queue, obtained via [next_avail\(\)](#), that has not yet been finished, and in addition, no descriptors following it have been finished.

Definition at line 160 of file [virtio](#).

References [L4virtio::Virtqueue::_current_avail](#), [L4virtio::Virtqueue::_desc](#), and [L4virtio::Virtqueue::_idx_mask](#).

The documentation for this class was generated from the following file:

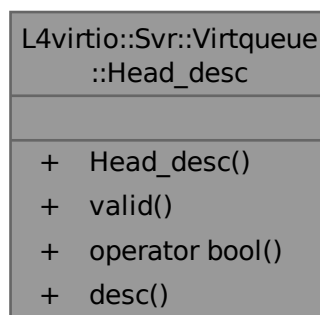
- I4/I4virtio/server/virtio

15.445 `L4virtio::Svr::Virtqueue::Head_desc` Class Reference

VIRTIO request, essentially a descriptor from the available ring.

```
#include <virtio>
```

Collaboration diagram for `L4virtio::Svr::Virtqueue::Head_desc`:



Public Member Functions

- **Head_desc** ()
Make invalid (NULL) request.
- bool **valid** () const
- operator bool () const
- Desc const * **desc** () const

15.445.1 Detailed Description

VIRTIO request, essentially a descriptor from the available ring.

Definition at line 93 of file [virtio](#).

15.445.2 Member Function Documentation

15.445.2.1 desc()

```
Desc const * L4virtio::Svr::Virtqueue::Head_desc::desc () const [inline]
```

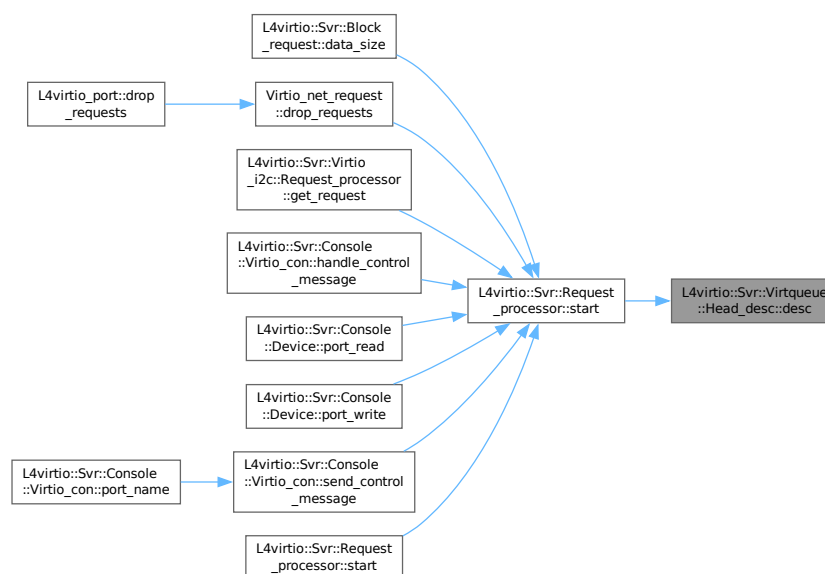
Returns

Pointer to the head descriptor of the request.

Definition at line 112 of file [virtio](#).

Referenced by [L4virtio::Svr::Request_processor::start\(\)](#).

Here is the caller graph for this function:



15.445.2.2 operator bool()

```
L4virtio::Svr::Virtqueue::Head_desc::operator bool () const [inline], [explicit]
```

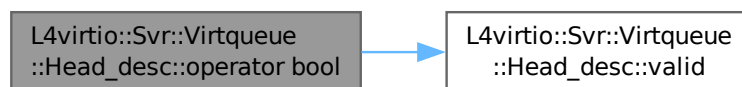
Returns

True if the request is valid (not NULL).

Definition at line 108 of file [virtio](#).

References [valid\(\)](#).

Here is the call graph for this function:



15.445.2.3 valid()

```
bool L4virtio::Svr::Virtqueue::Head_desc::valid () const [inline]
```

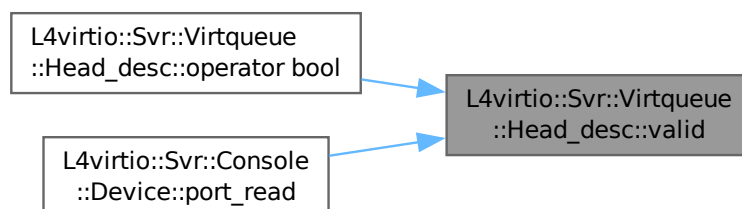
Returns

True if the request is valid (not NULL).

Definition at line 105 of file [virtio](#).

Referenced by [operator bool\(\)](#), and [L4virtio::Svr::Console::Device::port_read\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

- `I4/I4virtio/server/virtio`

15.446 L4virtio::Virtqueue Class Reference

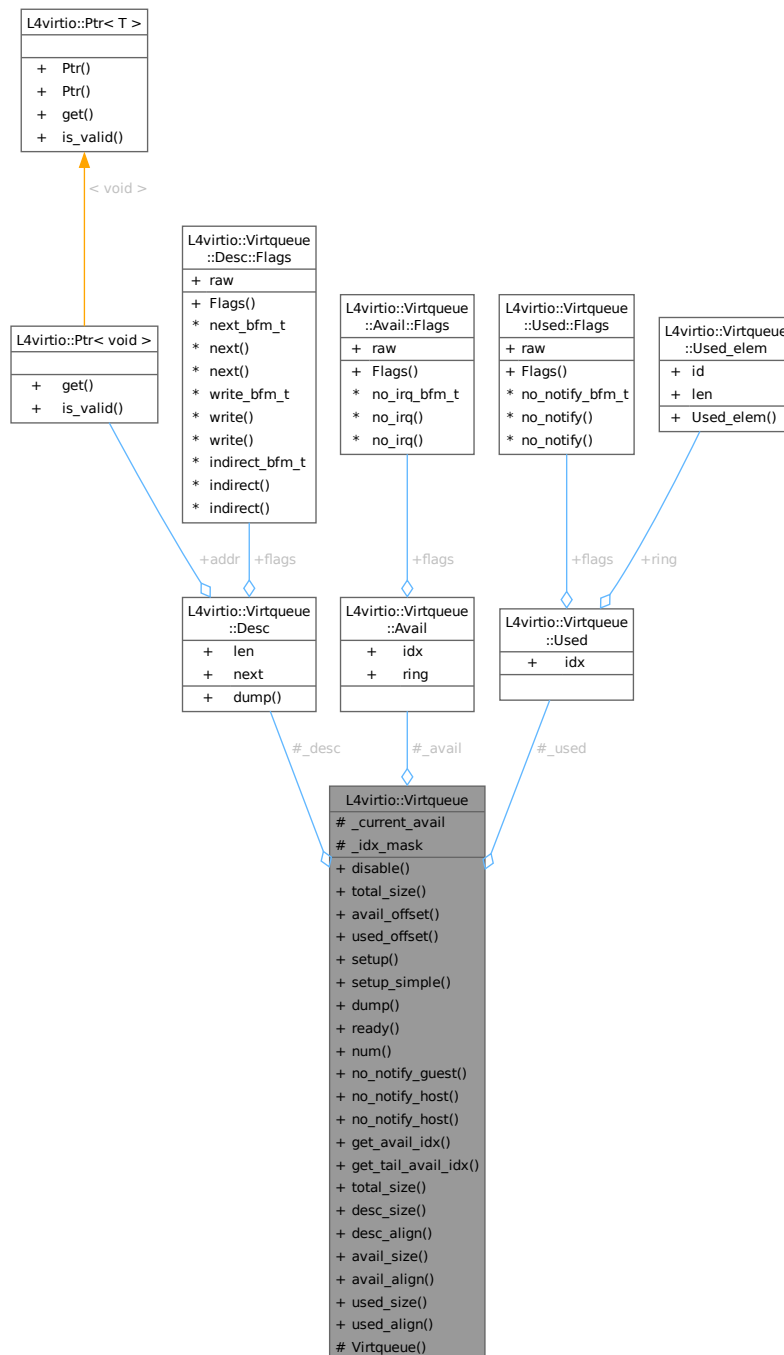
Low-level [Virtqueue](#).

```
#include <virtqueue>
```

Inheritance diagram for L4virtio::Virtqueue:



Collaboration diagram for L4virtio::Virtqueue:



Data Structures

- class [Desc](#)
Descriptor in the descriptor table.
- class [Avail](#)
Type of available ring, this is read-only for the host.
- struct [Used_elem](#)

Type of an element of the used ring.

- class [Used](#)
Used ring.

Public Types

- enum
Fixed alignment values for different parts of a virtqueue.

Public Member Functions

- void [disable](#) ()
Completely disable the queue.
- unsigned long [total_size](#) () const
Calculate the total size of this virtqueue.
- unsigned long [avail_offset](#) () const
Get the offset of the available ring from the descriptor table.
- unsigned long [used_offset](#) () const
Get the offset of the used ring from the descriptor table.
- void [setup](#) (unsigned [num](#), void *desc, void *avail, void *used)
Enable this queue.
- void [setup_simple](#) (unsigned [num](#), void *ring)
Enable this queue.
- void [dump](#) ([Desc](#) const *d) const
Dump descriptors for this queue.
- bool [ready](#) () const
Test if this queue is in working state.
- unsigned [num](#) () const
- bool [no_notify_guest](#) () const
Get the no IRQ flag of this queue.
- bool [no_notify_host](#) () const
Get the no notify flag of this queue.
- void [no_notify_host](#) (bool value)
Set the no-notify flag for this queue.
- [l4_uint16_t](#) [get_avail_idx](#) () const
Get available index from available ring (for debugging).
- [l4_uint16_t](#) [get_tail_avail_idx](#) () const
Get tail-available index stored in local state (for debugging).

Static Public Member Functions

- static unsigned long [total_size](#) (unsigned [num](#))
Calculate the total size for a virtqueue of the given dimensions.
- static unsigned long [desc_size](#) (unsigned [num](#))
Calculate the size of the descriptor table for [num](#) entries.
- static unsigned long [desc_align](#) ()
Get the alignment in zero LSBs needed for the descriptor table.
- static unsigned long [avail_size](#) (unsigned [num](#))
Calculate the size of the available ring for [num](#) entries.
- static unsigned long [avail_align](#) ()
Get the alignment in zero LSBs needed for the available ring.
- static unsigned long [used_size](#) (unsigned [num](#))
Calculate the size of the used ring for [num](#) entries.
- static unsigned long [used_align](#) ()
Get the alignment in zero LSBs needed for the used ring.

Protected Member Functions

- **Virtqueue** ()=default
Create a disabled virtqueue.

Protected Attributes

- **Desc** * **_desc** = nullptr
pointer to descriptor table, NULL if queue is off.
- **Avail** * **_avail** = nullptr
pointer to available ring.
- **Used** * **_used** = nullptr
pointer to used ring.
- **l4_uint16_t** **_current_avail** = 0
The life counter for the queue.
- **l4_uint16_t** **_idx_mask** = 0
mask used for indexing into the descriptor table and the rings.

15.446.1 Detailed Description

Low-level [Virtqueue](#).

This class represents a single virtqueue, with a local running available index.

Note

The [Virtqueue](#) implementation is not thread-safe.

Definition at line 86 of file [virtqueue](#).

15.446.2 Member Function Documentation

15.446.2.1 `avail_align()`

```
unsigned long L4virtio::Virtqueue::avail_align () [inline], [static]
```

Get the alignment in zero LSBs needed for the available ring.

Returns

The alignment in zero LSBs needed for an available ring.

Definition at line 293 of file [virtqueue](#).

15.446.2.2 avail_size()

```
unsigned long L4virtio::Virtqueue::avail_size (
    unsigned num) [inline], [static]
```

Calculate the size of the available ring for `num` entries.

Parameters

| | |
|------------------|--|
| <code>num</code> | The number of entries in the available ring. |
|------------------|--|

Returns

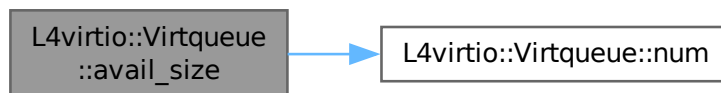
The size in bytes needed for an available ring with `num` entries.

Definition at line 285 of file `virtqueue`.

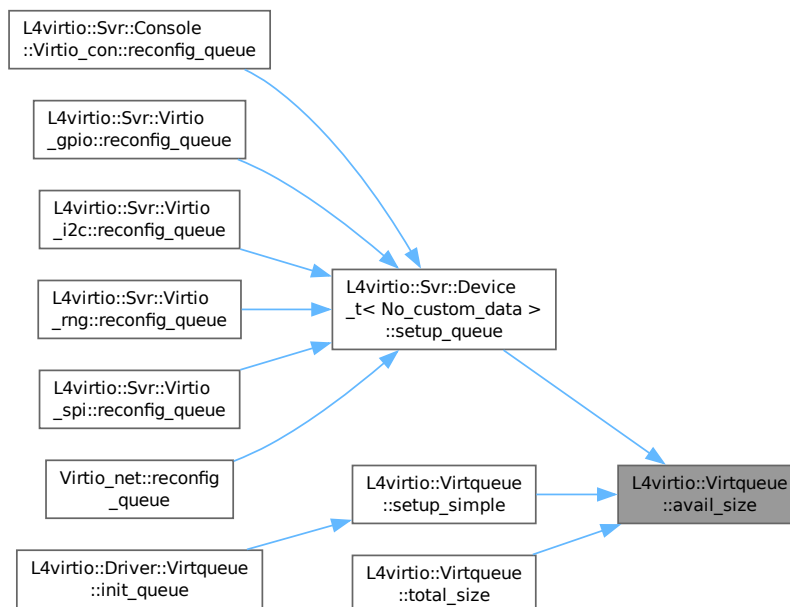
References `num()`.

Referenced by `L4virtio::Svr::Device_t< No_custom_data >::setup_queue()`, `setup_simple()`, and `total_size()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.446.2.3 desc_align()

```
unsigned long L4virtio::Virtqueue::desc_align () [inline], [static]
```

Get the alignment in zero LSBs needed for the descriptor table.

Returns

The alignment in zero LSBs needed for a descriptor table.

Definition at line 275 of file [virtqueue](#).

15.446.2.4 desc_size()

```
unsigned long L4virtio::Virtqueue::desc_size (
    unsigned num) [inline], [static]
```

Calculate the size of the descriptor table for [num](#) entries.

Parameters

| | |
|------------|--|
| <i>num</i> | The number of entries in the descriptor table. |
|------------|--|

Returns

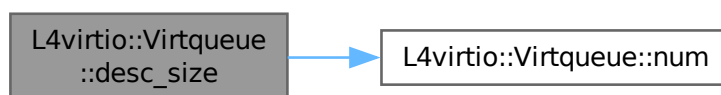
The size in bytes needed for a descriptor table with [num](#) entries.

Definition at line 267 of file [virtqueue](#).

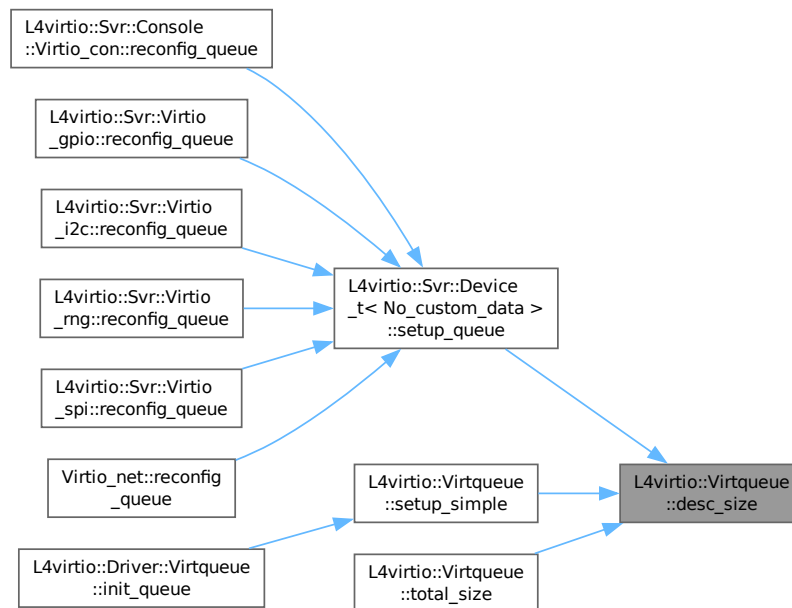
References [num\(\)](#).

Referenced by [L4virtio::Svr::Device_t< No_custom_data >::setup_queue\(\)](#), [setup_simple\(\)](#), and [total_size\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.446.2.5 disable()

```
void L4virtio::Virtqueue::disable () [inline]
```

Completely disable the queue.

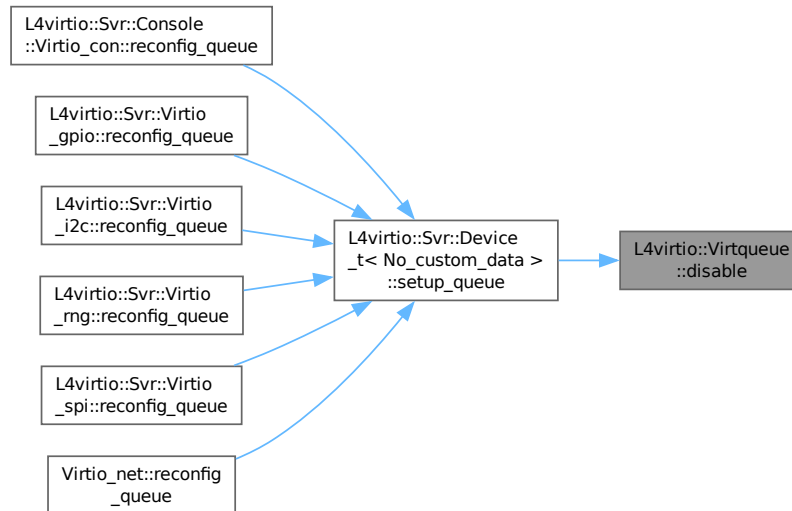
[setup\(\)](#) must be used to enable the queue again.

Definition at line 230 of file [virtqueue](#).

References [_desc](#).

Referenced by [L4virtio::Svr::Device_t< No_custom_data >::setup_queue\(\)](#).

Here is the caller graph for this function:



15.446.2.6 dump()

```
void L4virtio::Virtqueue::dump (
    Desc const * d) const [inline]
```

Dump descriptors for this queue.

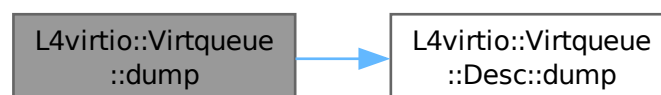
Precondition

the queue must be in working state.

Definition at line 398 of file `virtqueue`.

References `_desc`, and `L4virtio::Virtqueue::Desc::dump()`.

Here is the call graph for this function:



15.446.2.7 get_avail_idx()

```
l4_uint16_t L4virtio::Virtqueue::get_avail_idx () const [inline]
```

Get available index from available ring (for debugging).

Precondition

Queue must be in a working state.

Returns

current index in the available ring (shared between device model and device driver).

Definition at line 456 of file [virtqueue](#).

References [_avail](#).

15.446.2.8 get_tail_avail_idx()

```
l4_uint16_t L4virtio::Virtqueue::get_tail_avail_idx () const [inline]
```

Get tail-available index stored in local state (for debugging).

Returns

current tail index for the the available ring.

Definition at line 463 of file [virtqueue](#).

References [_current_avail](#).

15.446.2.9 no_notify_guest()

```
bool L4virtio::Virtqueue::no_notify_guest () const [inline]
```

Get the no IRQ flag of this queue.

Precondition

queue must be in working state.

Returns

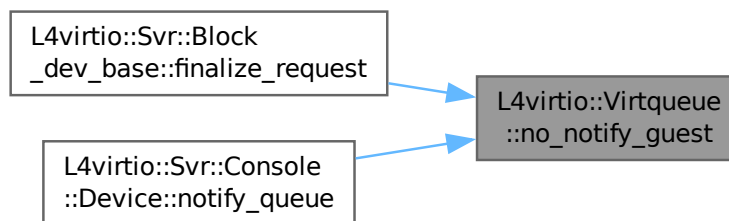
true if the guest does not want to get IRQs (currently).

Definition at line 420 of file [virtqueue](#).

References [_avail](#).

Referenced by [L4virtio::Svr::Block_dev_base<Ds_data>::finalize_request\(\)](#), and [L4virtio::Svr::Console::Device::notify_queue\(\)](#).

Here is the caller graph for this function:

**15.446.2.10 no_notify_host() [1/2]**

```
bool L4virtio::Virtqueue::no_notify_host () const [inline]
```

Get the no notify flag of this queue.

Precondition

queue must be in working state.

Returns

true if the host does not want to get IRQs (currently).

Definition at line 433 of file [virtqueue](#).

References [_used](#).

15.446.2.11 no_notify_host() [2/2]

```
void L4virtio::Virtqueue::no_notify_host (
    bool value) [inline]
```

Set the no-notify flag for this queue.

Precondition

Queue must be in a working state.

Definition at line 443 of file [virtqueue](#).

References [_used](#).

15.446.2.12 num()

```
unsigned L4virtio::Virtqueue::num () const [inline]
```

Returns

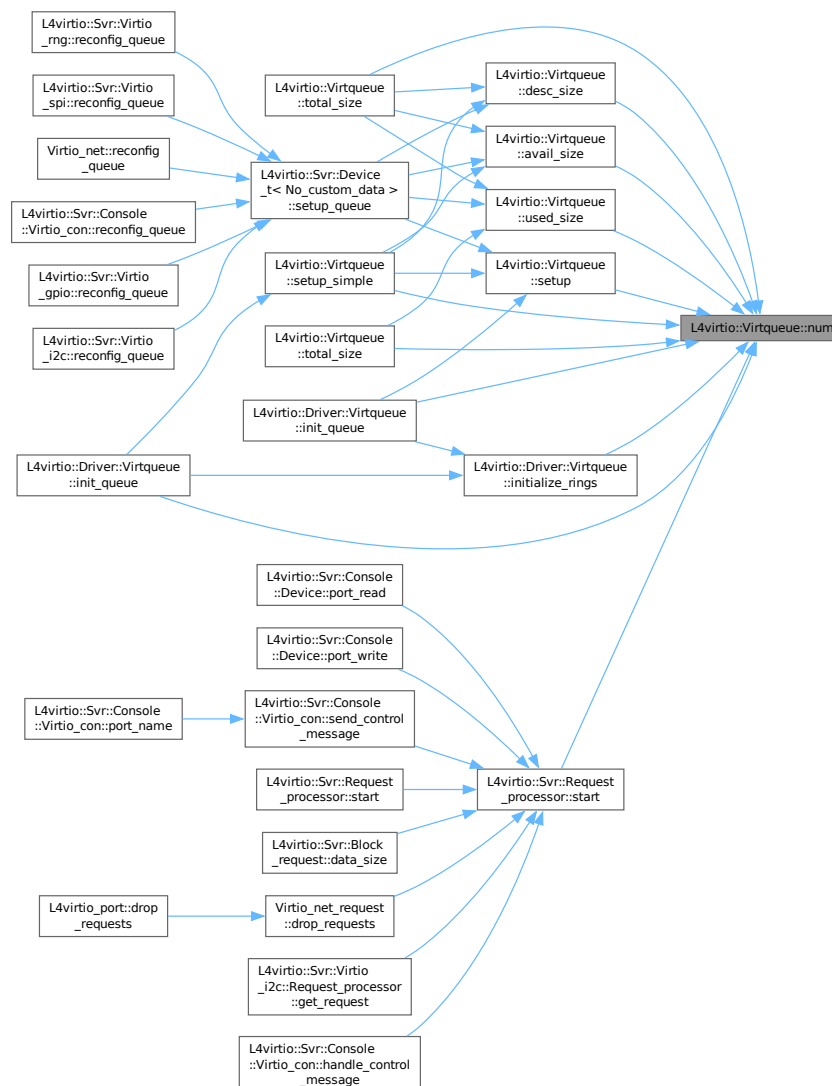
The number of entries in the ring.

Definition at line 410 of file [virtqueue](#).

References [_idx_mask](#).

Referenced by [avail_size\(\)](#), [desc_size\(\)](#), [L4virtio::Driver::Virtqueue::init_queue\(\)](#), [L4virtio::Driver::Virtqueue::init_queue\(\)](#), [L4virtio::Driver::Virtqueue::initialize_rings\(\)](#), [setup\(\)](#), [setup_simple\(\)](#), [L4virtio::Svr::Request_processor::start\(\)](#), [total_size\(\)](#), [total_size\(\)](#), and [used_size\(\)](#).

Here is the caller graph for this function:



15.446.2.13 ready()

```
bool L4virtio::Virtqueue::ready () const [inline]
```

Test if this queue is in working state.

Returns

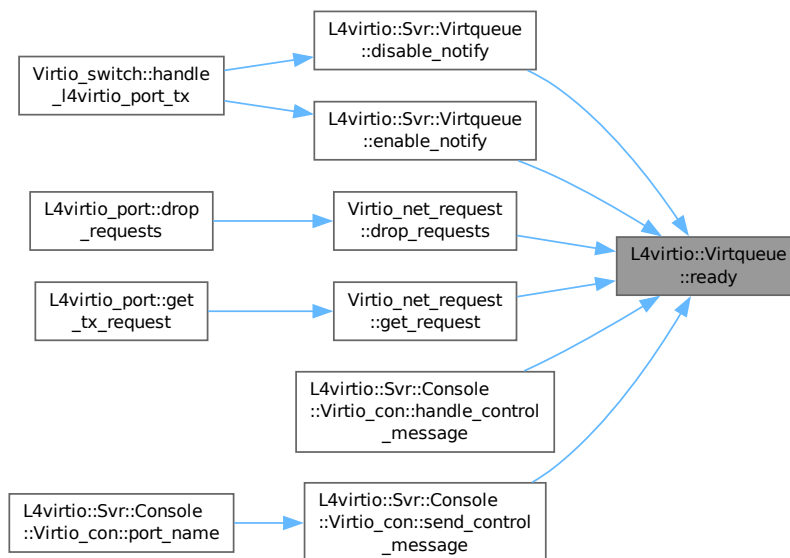
true when the queue is in working state, false else.

Definition at line 406 of file [virtqueue](#).

References [_desc](#), and [L4_LIKELY](#).

Referenced by [L4virtio::Svr::Virtqueue::disable_notify\(\)](#), [Virtio_net_request::drop_requests\(\)](#), [L4virtio::Svr::Virtqueue::enable_notify\(\)](#), [Virtio_net_request::get_request\(\)](#), [L4virtio::Svr::Console::Virtio_con::handle_control_message\(\)](#), and [L4virtio::Svr::Console::Virtio_con::port_name](#).

Here is the caller graph for this function:



15.446.2.14 setup()

```
void L4virtio::Virtqueue::setup (
    unsigned num,
    void * desc,
    void * avail,
    void * used) [inline]
```

Enable this queue.

Parameters

| | |
|--------------|--|
| <i>num</i> | The number of entries in the descriptor table, the available ring, and the used ring (must be a power of 2). |
| <i>desc</i> | The address of the descriptor table. (Must be <code>Desc_align</code> aligned and at least <code>desc_size(num)</code> bytes in size.) |
| <i>avail</i> | The address of the available ring. (Must be <code>Avail_align</code> aligned and at least <code>avail_size(num)</code> bytes in size.) |
| <i>used</i> | The address of the used ring. (Must be <code>Used_align</code> aligned and at least <code>used_size(num)</code> bytes in size.) |

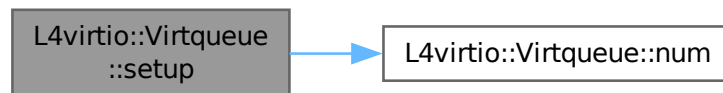
Due to the data type of the descriptors, the queue can have a maximum size of 2^{16} .

Definition at line 355 of file `virtqueue`.

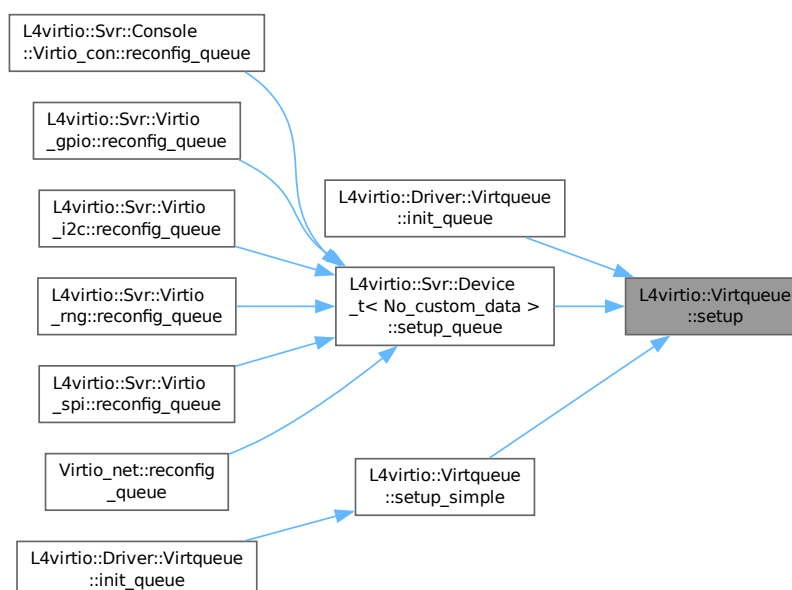
References `_avail`, `_current_avail`, `_desc`, `_idx_mask`, `_used`, `L4_EINVAL`, and `num()`.

Referenced by `L4virtio::Driver::Virtqueue::init_queue()`, `L4virtio::Svr::Device_t< No_custom_data >::setup_queue()`, and `setup_simple()`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.446.2.15 setup_simple()

```
void L4virtio::Virtqueue::setup_simple (
    unsigned num,
    void * ring) [inline]
```

Enable this queue.

Parameters

| | |
|-------------|---|
| <i>num</i> | The number of entries in the descriptor table, the available ring, and the used ring (must be a power of 2). |
| <i>ring</i> | The base address for the queue data structure. The memory block at <i>ring</i> must be at least <code>total_size(num)</code> bytes in size and have an alignment of <code>Desc_align(desc_align())</code> bits. |

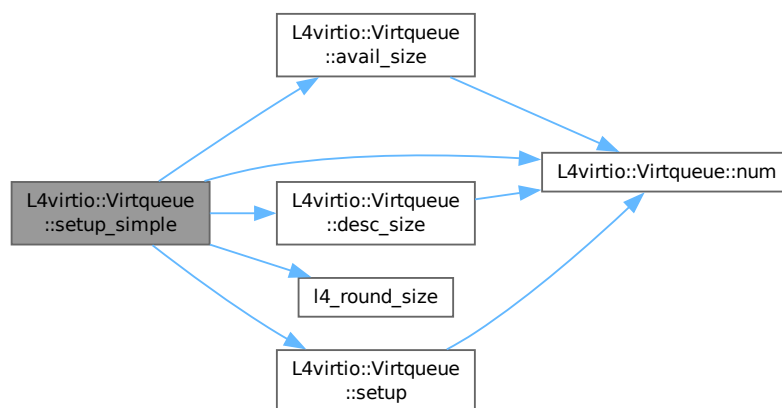
Due to the data type of the descriptors, the queue can have a maximum size of 2^{16} .

Definition at line 384 of file [virtqueue](#).

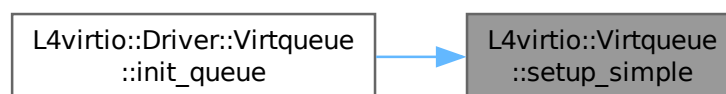
References [avail_size\(\)](#), [desc_size\(\)](#), [l4_round_size\(\)](#), [num\(\)](#), and [setup\(\)](#).

Referenced by [L4virtio::Driver::Virtqueue::init_queue\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.446.2.16 total_size() [1/2]

```
unsigned long L4virtio::Virtqueue::total_size () const [inline]
```

Calculate the total size of this virtqueue.

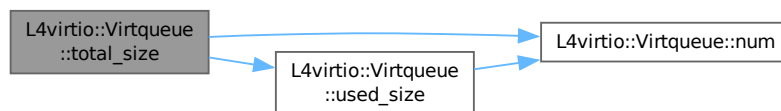
Precondition

The queue has been set up.

Definition at line 320 of file [virtqueue](#).

References [_desc](#), [_used](#), [num\(\)](#), and [used_size\(\)](#).

Here is the call graph for this function:

**15.446.2.17 total_size()** [2/2]

```
unsigned long L4virtio::Virtqueue::total_size (
    unsigned num) [inline], [static]
```

Calculate the total size for a virtqueue of the given dimensions.

Parameters

| | |
|------------|--|
| <i>num</i> | The number of entries in the descriptor table, the available ring, and the used ring (must be a power of 2). |
|------------|--|

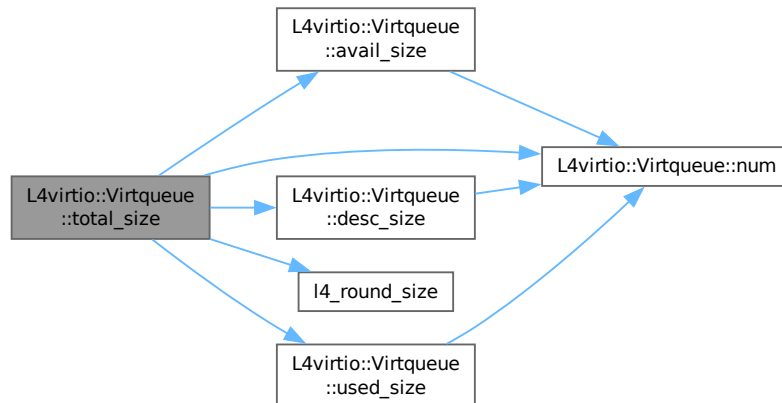
Returns

The total size in bytes of the queue data structures.

Definition at line 251 of file [virtqueue](#).

References [avail_size\(\)](#), [desc_size\(\)](#), [l4_round_size\(\)](#), [num\(\)](#), and [used_size\(\)](#).

Here is the call graph for this function:



15.446.2.18 `used_align()`

```
unsigned long L4virtio::Virtqueue::used_align () [inline], [static]
```

Get the alignment in zero LSBs needed for the used ring.

Returns

The alignment in zero LSBs needed for an used ring.

Definition at line 312 of file [virtqueue](#).

15.446.2.19 `used_size()`

```
unsigned long L4virtio::Virtqueue::used_size (
    unsigned num) [inline], [static]
```

Calculate the size of the used ring for `num` entries.

Parameters

| | |
|------------|---|
| <i>num</i> | The number of entries in the used ring. |
|------------|---|

Returns

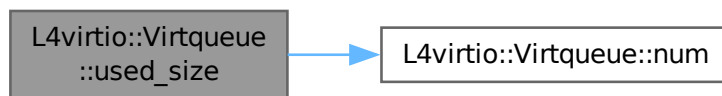
The size in bytes needed for an used ring with `num` entries.

Definition at line 304 of file [virtqueue](#).

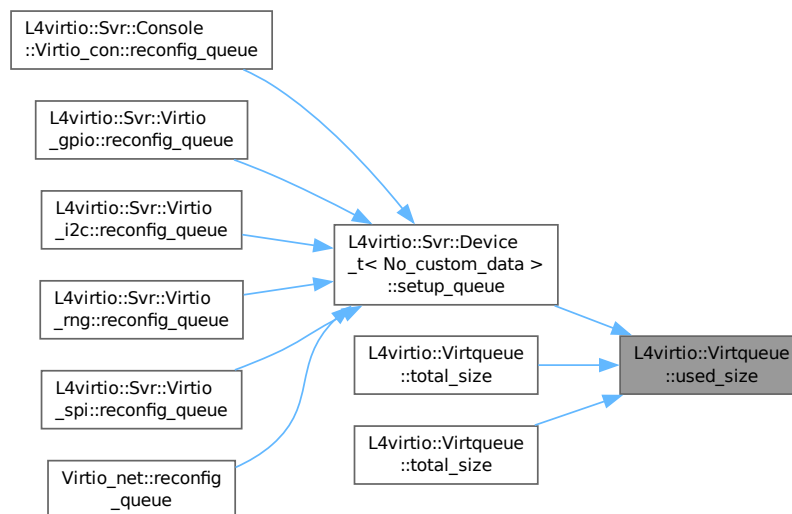
References [num\(\)](#).

Referenced by [L4virtio::Svr::Device_t< No_custom_data >::setup_queue\(\)](#), [total_size\(\)](#), and [total_size\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

- `I4/I4virtio/virtqueue`

15.447 L4virtio::Virtqueue::Avail Class Reference

Type of available ring, this is read-only for the host.

```
#include <virtqueue>
```

Collaboration diagram for L4virtio::Virtqueue::Avail:



Data Structures

- struct [Flags](#)
Flags of the available ring.

Data Fields

- [Flags flags](#)
flags of available ring
- [l4_uint16_t idx](#)
available index written by guest
- [l4_uint16_t ring \[\]](#)
array of available descriptor indexes.

15.447.1 Detailed Description

Type of available ring, this is read-only for the host.

Definition at line 134 of file [virtqueue](#).

The documentation for this class was generated from the following file:

- [l4/l4virtio/virtqueue](#)

15.448 L4virtio::Virtqueue::Avail::Flags Struct Reference

[Flags](#) of the available ring.

```
#include <virtqueue>
```

Collaboration diagram for L4virtio::Virtqueue::Avail::Flags:

| L4virtio::Virtqueue ::Avail::Flags |
|---------------------------------------|
| + raw |
| + Flags() |
| * no_irq_bfm_t |
| * no_irq() |
| * no_irq() |

Public Member Functions

- **Flags** ([l4_uint16_t](#) v)
Make [Flags](#) from the raw value.

Data Fields

- [l4_uint16_t](#) raw
raw 16bit flags value of the available ring.
- typedef [cxx::Bitfield](#)< decltype(raw), 0, 0 > no_irq_bfm_t
Guest does not want to receive interrupts when requests are finished.
- constexpr [no_irq_bfm_t::Val](#) no_irq () const
Get the [no_irq](#) bits (0 to 0) of raw.
- constexpr [no_irq_bfm_t::Ref](#) no_irq ()
Get a reference to the [no_irq](#) bits (0 to 0) of raw.

15.448.1 Detailed Description

[Flags](#) of the available ring.

Definition at line [140](#) of file [virtqueue](#).

15.448.2 Member Typedef Documentation

15.448.2.1 no_irq_bfm_t

```
typedef cxx::Bitfield<decltype(raw), 0, 0> L4virtio::Virtqueue::Avail::Flags::no\_irq\_bfm\_t
```

Guest does not want to receive interrupts when requests are finished.

Type to access the [no_irq](#) bits (0 to 0) of [raw](#).

Definition at line [149](#) of file [virtqueue](#).

The documentation for this struct was generated from the following file:

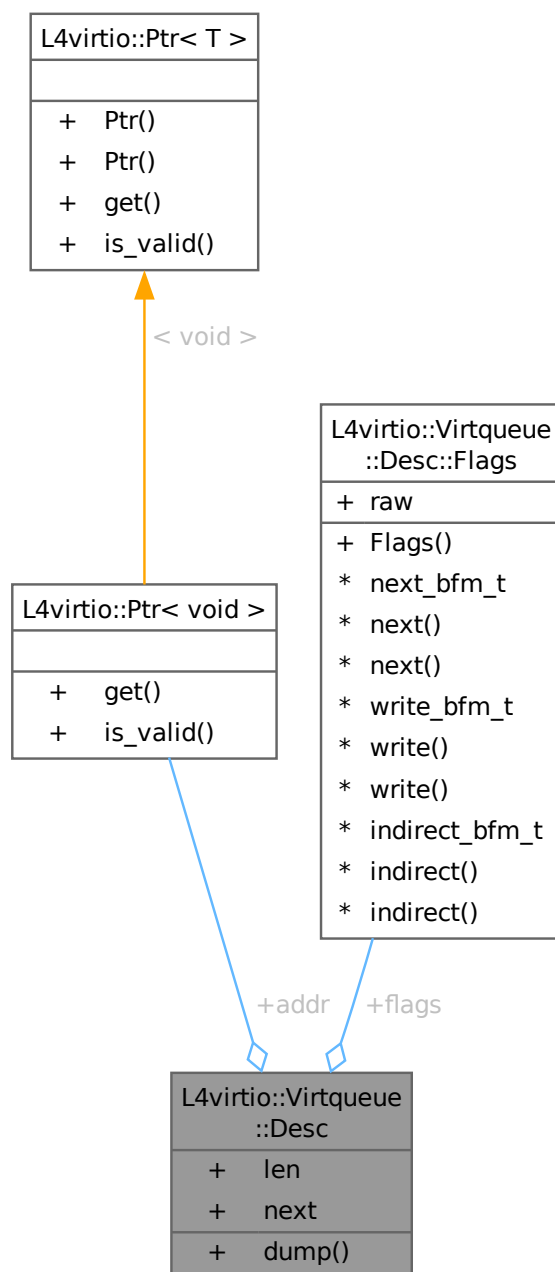
- [l4/l4virtio/virtqueue](#)

15.449 L4virtio::Virtqueue::Desc Class Reference

Descriptor in the descriptor table.

```
#include <virtqueue>
```

Collaboration diagram for L4virtio::Virtqueue::Desc:



Data Structures

- struct [Flags](#)
Type for descriptor flags.

Public Member Functions

- void **dump** (unsigned idx) const

Dump a single descriptor.

Data Fields

- [Ptr](#)< void > **addr**
Address stored in descriptor.
- [l4_uint32_t](#) **len**
Length of described buffer.
- [Flags](#) **flags**
Descriptor flags.
- [l4_uint16_t](#) **next**
Index of the next chained descriptor.

15.449.1 Detailed Description

Descriptor in the descriptor table.

Definition at line 92 of file [virtqueue](#).

The documentation for this class was generated from the following file:

- l4/l4virtio/virtqueue

15.450 L4virtio::Virtqueue::Desc::Flags Struct Reference

Type for descriptor flags.

```
#include <virtqueue>
```

Collaboration diagram for L4virtio::Virtqueue::Desc::Flags:

| L4virtio::Virtqueue ::Desc::Flags |
|--------------------------------------|
| + raw |
| + Flags() |
| * next_bfm_t |
| * next() |
| * next() |
| * write_bfm_t |
| * write() |
| * write() |
| * indirect_bfm_t |
| * indirect() |
| * indirect() |

Public Member Functions

- **Flags** ([l4_uint16_t](#) v)
Make [Flags](#) from raw 16bit value.

Data Fields

- [l4_uint16_t](#) **raw**
raw flags value of a virtio descriptor.
- typedef [cxx::Bitfield](#)< decltype([raw](#)), 0, 0 > [next_bfm_t](#)
Part of a descriptor chain which is continued with the next field.
- constexpr [next_bfm_t::Val](#) **next** () const
Get the [next](#) bits (0 to 0) of [raw](#).
- constexpr [next_bfm_t::Ref](#) **next** ()
Get a reference to the [next](#) bits (0 to 0) of [raw](#).
- typedef [cxx::Bitfield](#)< decltype([raw](#)), 1, 1 > [write_bfm_t](#)
Block described by this descriptor is writeable.
- constexpr [write_bfm_t::Val](#) **write** () const
Get the [write](#) bits (1 to 1) of [raw](#).
- constexpr [write_bfm_t::Ref](#) **write** ()
Get a reference to the [write](#) bits (1 to 1) of [raw](#).
- typedef [cxx::Bitfield](#)< decltype([raw](#)), 2, 2 > [indirect_bfm_t](#)
Indirect descriptor, block contains a list of descriptors.
- constexpr [indirect_bfm_t::Val](#) **indirect** () const
Get the [indirect](#) bits (2 to 2) of [raw](#).
- constexpr [indirect_bfm_t::Ref](#) **indirect** ()
Get a reference to the [indirect](#) bits (2 to 2) of [raw](#).

15.450.1 Detailed Description

Type for descriptor flags.

Definition at line 98 of file [virtqueue](#).

15.450.2 Member Typedef Documentation

15.450.2.1 indirect_bfm_t

```
typedef cxx::Bitfield<decltype(raw), 2, 2> L4virtio::Virtqueue::Desc::Flags::indirect\_bfm\_t
```

Indirect descriptor, block contains a list of descriptors.

Type to access the [indirect](#) bits (2 to 2) of [raw](#).

Definition at line 111 of file [virtqueue](#).

15.450.2.2 next_bfm_t

```
typedef cxx::Bitfield<decltype(raw), 0, 0> L4virtio::Virtqueue::Desc::Flags::next_bfm_t
```

Part of a descriptor chain which is continued with the next field.

Type to access the `next` bits (0 to 0) of `raw`.

Definition at line 107 of file `virtqueue`.

15.450.2.3 write_bfm_t

```
typedef cxx::Bitfield<decltype(raw), 1, 1> L4virtio::Virtqueue::Desc::Flags::write_bfm_t
```

Block described by this descriptor is writeable.

Type to access the `write` bits (1 to 1) of `raw`.

Definition at line 109 of file `virtqueue`.

The documentation for this struct was generated from the following file:

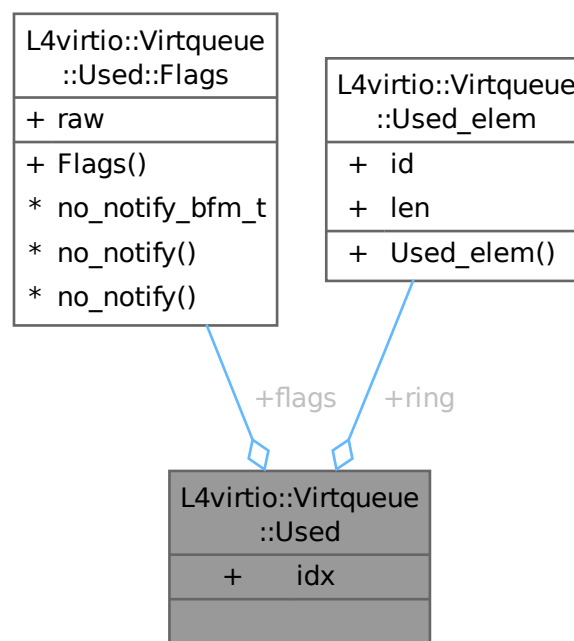
- `l4/l4virtio/virtqueue`

15.451 L4virtio::Virtqueue::Used Class Reference

`Used` ring.

```
#include <virtqueue>
```

Collaboration diagram for `L4virtio::Virtqueue::Used`:



Data Structures

- struct [Flags](#)
flags for the used ring.

Data Fields

- [Flags](#) **flags**
flags of the used ring.
- [l4_uint16_t](#) **idx**
index of the last entry in the ring.
- [Used_elem](#) **ring** []
array of used descriptors.

15.451.1 Detailed Description

[Used](#) ring.

Definition at line 179 of file [virtqueue](#).

The documentation for this class was generated from the following file:

- l4/l4virtio/virtqueue

15.452 L4virtio::Virtqueue::Used::Flags Struct Reference

flags for the used ring.

```
#include <virtqueue>
```

Collaboration diagram for L4virtio::Virtqueue::Used::Flags:

| L4virtio::Virtqueue ::Used::Flags |
|--------------------------------------|
| + raw |
| + Flags() |
| * no_notify_bfm_t |
| * no_notify() |
| * no_notify() |

Public Member Functions

- **Flags** ([l4_uint16_t](#) v)
make [Flags](#) from raw value

Data Fields

- [l4_uint16_t](#) **raw**
raw flags value as specified by virtio.
- typedef [cxx::Bitfield](#)< decltype([raw](#)), 0, 0 > **no_notify_bfm_t**
host does not want to be notified when new requests have been queued.
- constexpr [no_notify_bfm_t::Val](#) **no_notify** () const
Get the [no_notify](#) bits (0 to 0) of [raw](#).
- constexpr [no_notify_bfm_t::Ref](#) **no_notify** ()
Get a reference to the [no_notify](#) bits (0 to 0) of [raw](#).

15.452.1 Detailed Description

flags for the used ring.

Definition at line [185](#) of file [virtqueue](#).

15.452.2 Member Typedef Documentation

15.452.2.1 no_notify_bfm_t

```
typedef cxx::Bitfield<decltype(raw), 0, 0> L4virtio::Virtqueue::Used::Flags::no\_notify\_bfm\_t
```

host does not want to be notified when new requests have been queued.

Type to access the [no_notify](#) bits (0 to 0) of [raw](#).

Definition at line [194](#) of file [virtqueue](#).

The documentation for this struct was generated from the following file:

- [l4/l4virtio/virtqueue](#)

15.453 L4virtio::Virtqueue::Used_elem Struct Reference

Type of an element of the used ring.

```
#include <virtqueue>
```

Collaboration diagram for L4virtio::Virtqueue::Used_elem:

| L4virtio::Virtqueue ::Used_elem |
|------------------------------------|
| + id |
| + len |
| + Used_elem() |

Public Member Functions

- [Used_elem](#) ([l4_uint16_t](#) id, [l4_uint32_t](#) len)
Initialize a used ring element.

Data Fields

- [l4_uint32_t](#) id
descriptor index
- [l4_uint32_t](#) len
length field

15.453.1 Detailed Description

Type of an element of the used ring.

Definition at line [160](#) of file [virtqueue](#).

15.453.2 Constructor & Destructor Documentation

15.453.2.1 Used_elem()

```
L4virtio::Virtqueue::Used_elem::Used_elem (
    14_uint16_t id,
    14_uint32_t len) [inline]
```

Initialize a used ring element.

Parameters

| | |
|------------|--|
| <i>id</i> | The index of the descriptor to be marked as used. |
| <i>len</i> | The total bytes written into the buffer of the descriptor chain. |

Definition at line 171 of file [virtqueue](#).

References [id](#), and [len](#).

The documentation for this struct was generated from the following file:

- I4/I4virtio/virtqueue

15.454 I4virtio_block_config_t Struct Reference

Device configuration for block devices.

```
#include <virtio_block.h>
```

Collaboration diagram for I4virtio_block_config_t:

| I4virtio_block_config_t | |
|-------------------------|----------|
| + | capacity |
| + | size_max |
| + | seg_max |
| + | blk_size |
| | |

Data Fields

- [l4_uint64_t](#) **capacity**
Capacity of device in 512-byte sectors.
- [l4_uint32_t](#) **size_max**
Maximum size of a single segment.
- [l4_uint32_t](#) **seg_max**
Maximum number of segments per request.
- [l4_uint32_t](#) **blk_size**
Block size of underlying disk.

15.454.1 Detailed Description

Device configuration for block devices.

Definition at line 68 of file [virtio_block.h](#).

The documentation for this struct was generated from the following file:

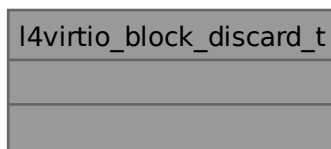
- l4/l4virtio/virtio_block.h

15.455 l4virtio_block_discard_t Struct Reference

Structure used for the write zeroes and discard commands.

```
#include <virtio_block.h>
```

Collaboration diagram for l4virtio_block_discard_t:



15.455.1 Detailed Description

Structure used for the write zeroes and discard commands.

Definition at line 58 of file [virtio_block.h](#).

The documentation for this struct was generated from the following file:

- l4/l4virtio/virtio_block.h

15.456 l4virtio_block_header_t Struct Reference

Header structure of a request for a block device.

```
#include <virtio_block.h>
```

Collaboration diagram for l4virtio_block_header_t:

| l4virtio_block_header_t | |
|-------------------------|--------|
| + | type |
| + | ioprio |
| + | sector |
| | |

Data Fields

- [l4_uint32_t](#) **type**
Kind of request, see [L4virtio_block_operations](#).
- [l4_uint32_t](#) **ioprio**
Priority (unused).
- [l4_uint64_t](#) **sector**
First sector to read/write.

15.456.1 Detailed Description

Header structure of a request for a block device.

Definition at line 42 of file [virtio_block.h](#).

The documentation for this struct was generated from the following file:

- l4/l4virtio/virtio_block.h

15.457 l4virtio_config_hdr_t Struct Reference

L4-VIRTIO config header, provided in shared data space.

```
#include <virtio.h>
```

Collaboration diagram for l4virtio_config_hdr_t:

| l4virtio_config_hdr_t |
|-----------------------|
| + magic |
| + version |
| + device |
| + vendor |
| + dev_features |
| + num_queues |
| + queues_offset |

Data Fields

- [l4_uint32_t](#) **magic**
magic value (must be 'virt').
- [l4_uint32_t](#) **version**
VIRTIO version.
- [l4_uint32_t](#) **device**
device ID
- [l4_uint32_t](#) **vendor**
vendor ID
- [l4_uint32_t](#) **dev_features**
device features windows selected by device_feature_sel
- [l4_uint32_t](#) **num_queues**
number of virtqueues
- [l4_uint32_t](#) **queues_offset**
offset of virtqueue config array

15.457.1 Detailed Description

L4-VIRTIO config header, provided in shared data space.

Definition at line 135 of file [virtio.h](#).

The documentation for this struct was generated from the following file:

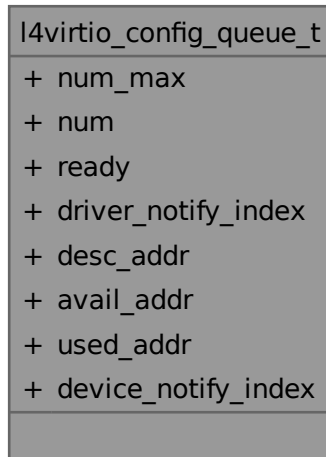
- l4/l4virtio/virtio.h

15.458 l4virtio_config_queue_t Struct Reference

Queue configuration entry.

```
#include <virtio.h>
```

Collaboration diagram for l4virtio_config_queue_t:



Data Fields

- [l4_uint16_t](#) **num_max**
R: maximum number of descriptors supported by this queue.
- [l4_uint16_t](#) **num**
RW: number of descriptors configured for this queue.
- [l4_uint16_t](#) **ready**
RW: queue ready flag (read-write).
- [l4_uint16_t](#) **driver_notify_index**
W: Event index to be used for device notifications (device to driver).
- [l4_uint64_t](#) **desc_addr**
W: address of descriptor table.
- [l4_uint64_t](#) **avail_addr**
W: address of available ring.
- [l4_uint64_t](#) **used_addr**
W: address of used ring.
- [l4_uint16_t](#) **device_notify_index**
R: Event index to be used by the driver (driver to device).

15.458.1 Detailed Description

Queue configuration entry.

An array of such entries is available at the [l4virtio_config_hdr_t::queues_offset](#) in the config data space.

Consistency rules for the queue config are:

- A driver might read [num_max](#) at any time.
- A driver must write to [num](#), [desc_addr](#), [avail_addr](#), and [used_addr](#) only when [ready](#) is zero (0). Values in these fields are validated and used by the device only after successfully setting [ready](#) to one (1), either by the IPC or by L4VIRTIO_CMD_CFG_QUEUE.
- The value of [device_notify_index](#) is valid only when [ready](#) is one.
- The driver might write to [device_notify_index](#) at any time, however the change is guaranteed to take effect after a successful L4VIRTIO_CMD_CFG_QUEUE or after a config_queue IPC. Note, the change might also have immediate effect, depending on the device implementation.

Definition at line [226](#) of file [virtio.h](#).

The documentation for this struct was generated from the following file:

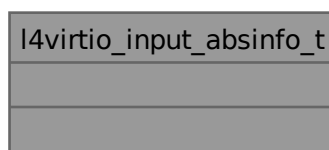
- [l4/l4virtio/virtio.h](#)

15.459 l4virtio_input_absinfo_t Struct Reference

Information about the absolute axis in the underlying evdev implementation.

```
#include <virtio_input.h>
```

Collaboration diagram for l4virtio_input_absinfo_t:



15.459.1 Detailed Description

Information about the absolute axis in the underlying evdev implementation.

Definition at line [33](#) of file [virtio_input.h](#).

The documentation for this struct was generated from the following file:

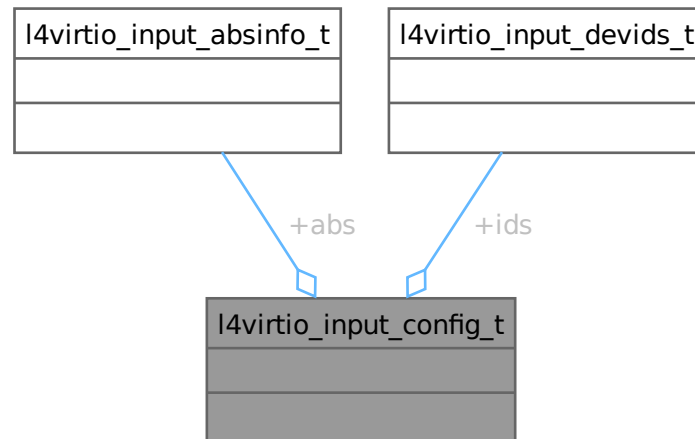
- [l4/l4virtio/virtio_input.h](#)

15.460 l4virtio_input_config_t Struct Reference

Device configuration for input devices.

```
#include <virtio_input.h>
```

Collaboration diagram for l4virtio_input_config_t:



15.460.1 Detailed Description

Device configuration for input devices.

Definition at line 56 of file [virtio_input.h](#).

The documentation for this struct was generated from the following file:

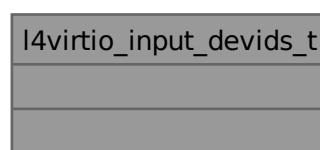
- l4/l4virtio/virtio_input.h

15.461 l4virtio_input_devids_t Struct Reference

Device ID information for the device.

```
#include <virtio_input.h>
```

Collaboration diagram for l4virtio_input_devids_t:



15.461.1 Detailed Description

Device ID information for the device.

Definition at line 45 of file [virtio_input.h](#).

The documentation for this struct was generated from the following file:

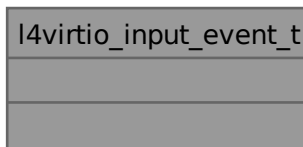
- l4/l4virtio/virtio_input.h

15.462 l4virtio_input_event_t Struct Reference

Single event in event or status queue.

```
#include <virtio_input.h>
```

Collaboration diagram for l4virtio_input_event_t:



15.462.1 Detailed Description

Single event in event or status queue.

Definition at line 74 of file [virtio_input.h](#).

The documentation for this struct was generated from the following file:

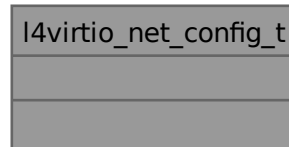
- l4/l4virtio/virtio_input.h

15.463 l4virtio_net_config_t Struct Reference

Device configuration for network devices.

```
#include <virtio_net.h>
```

Collaboration diagram for l4virtio_net_config_t:



15.463.1 Detailed Description

Device configuration for network devices.

Definition at line 34 of file [virtio_net.h](#).

The documentation for this struct was generated from the following file:

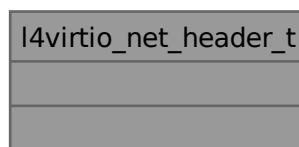
- I4/I4virtio/virtio_net.h

15.464 l4virtio_net_header_t Struct Reference

Header structure of a request for a network device.

```
#include <virtio_net.h>
```

Collaboration diagram for l4virtio_net_header_t:



15.464.1 Detailed Description

Header structure of a request for a network device.

Definition at line 20 of file [virtio_net.h](#).

The documentation for this struct was generated from the following file:

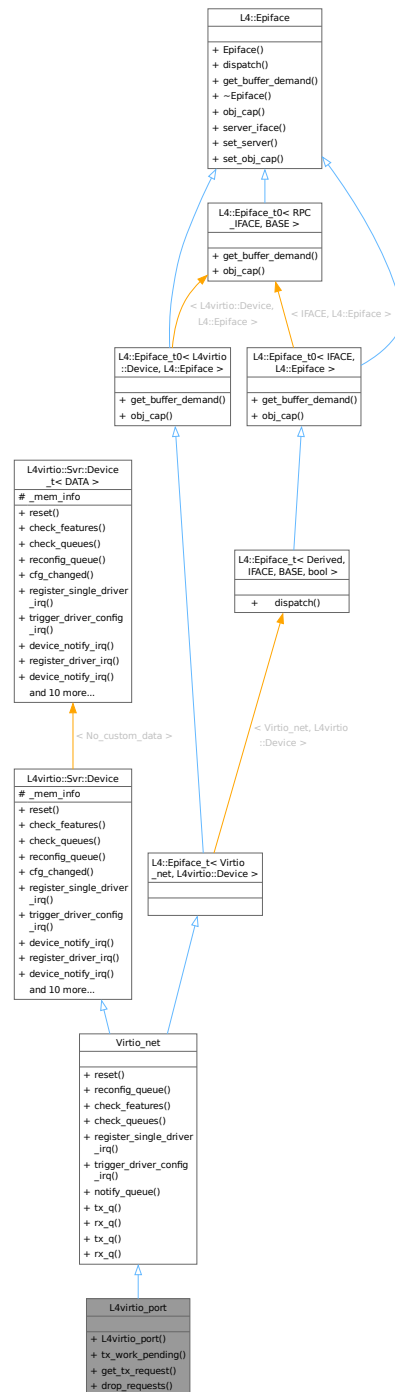
- l4/l4virtio/virtio_net.h

15.465 L4virtio_port Class Reference

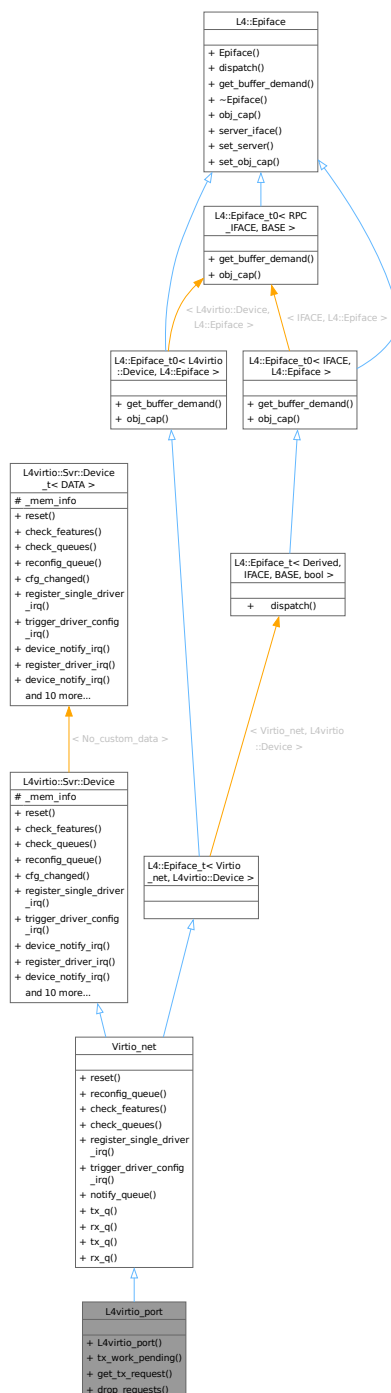
A Port on the Virtio Net Switch.

```
#include <port_l4virtio.h>
```

Inheritance diagram for L4virtio_port:



Collaboration diagram for L4virtio_port:



Public Member Functions

- **L4virtio_port** (unsigned vq_max, unsigned num_ds, char const *name, [l4_uint8_t](#) const *mac)
Create a Virtio net port object.
- bool **tx_work_pending** () const
Check whether there is any work pending on the transmission queue.
- std::optional< [Virtio_net_request](#) > **get_tx_request** ()

Get one request from the transmission queue.

- void **drop_requests** ()

Drop all requests pending in the transmission queue.

Public Member Functions inherited from [Virtio_net](#)

- void **reset** () override
reset callback, called for doing a device reset
- int **reconfig_queue** (unsigned index) override
callback for client queue-config request
- bool **check_features** () override
callback for checking the subset of accepted features
- bool **check_queues** () override
Check whether both virtqueues are ready.
- void **register_single_driver_irq** () override
Save the `_kick_guest_irq` that the client sent via `device_notification_irq()`.
- void **trigger_driver_config_irq** () override
callback for triggering configuration change notification IRQ
- void **notify_queue** ([L4virtio::Svr::Virtqueue](#) *queue)
Trigger the `_kick_guest_irq` IRQ.
- [Virtqueue](#) * **tx_q** ()
Getter for the transmission queue.
- [Virtqueue](#) * **rx_q** ()
Getter for the receive queue.
- [Virtqueue](#) const * **tx_q** () const
Getter for the transmission queue.
- [Virtqueue](#) const * **rx_q** () const
Getter for the receive queue.

Public Member Functions inherited from [L4virtio::Svr::Device_t< No_custom_data >](#)

- virtual void **cfg_changed** (unsigned)
callback for client device configuration changes
- virtual [L4::Cap< L4::Irq >](#) **device_notify_irq** () const
callback to gather the device notification IRQ (old-style)
- virtual void **register_driver_irq** (unsigned idx)
Callback for registering an notification IRQ (multi IRQ).
- virtual [L4::Cap< L4::Irq >](#) **device_notify_irq** (unsigned idx)
Callback to gather the device notification IRQ (multi IRQ).
- virtual unsigned **num_events_supported** () const
Return the highest notification index supported.
- **Device_t** ([Dev_config](#) *dev_config)
Make a device for the given config.
- [Mem_list](#) const * **mem_info** () const
Get the memory region list used for this device.
- void **reset_queue_config** (unsigned idx, unsigned num_max, bool inc_generation=false, unsigned device_↔
notify_index=0)
Trigger reset for the configuration space for queue idx.
- void **init_mem_info** (unsigned num)

- Initialize the memory region list to the given maximum.*
- void [device_error](#) ()
Transition device into DEVICE_NEEDS_RESET state.
- bool [setup_queue](#) (Virtqueue *q, unsigned qn, unsigned num_max)
Enable/disable the specified queue.
- bool [handle_mem_cmd_write](#) ()
Check for a value in the cmd register and handle a write.
- void [enable_trusted_ds_validation](#) ()
Enable trusted dataspace validation.
- void [add_trusted_dataspaces](#) (std::shared_ptr< Ds_vector const > ds)
Provide a list of trusted dataspaces that can be used for validation.

Public Member Functions inherited from [L4::Epiface_t0](#)< [L4virtio::Device](#), [L4::Epiface](#) >

- [Type_info::Demand](#) [get_buffer_demand](#) () const
Get the server-side buffer demand based in IFACE.
- [Cap](#)< [L4virtio::Device](#) > [obj_cap](#) () const
Get the (typed) capability to this object.

Public Member Functions inherited from [L4::Epiface](#)

- [Epiface](#) ()
Make a server object.
- virtual ~[Epiface](#) ()=0
Destroy the object.
- Stored_cap [obj_cap](#) () const
Get the capability to the kernel object belonging to this object.
- [Server_iface](#) * [server_iface](#) () const
Get pointer to server interface at which the object is currently registered.
- int [set_server](#) ([Server_iface](#) *srv, [Cap](#)< void > cap, bool managed=false)
Set server registration info for the object.
- void [set_obj_cap](#) ([Cap](#)< void > const &cap)
Deprecated server registration function.

Additional Inherited Members

Public Types inherited from [L4::Epiface_t0](#)< [L4virtio::Device](#), [L4::Epiface](#) >

- using [Interface](#)
Data type of the IPC interface definition.

Public Types inherited from [L4::Epiface](#)

- using [Server_iface](#) = [lpc_svr::Server_iface](#)
Type for abstract server interface.
- using [Demand](#) = [lpc_svr::Server_iface::Demand](#)
Type for server-side receive buffer demand.

Protected Attributes inherited from [L4virtio::Svr::Device_t< No_custom_data >](#)

- `Mem_list_mem_info`
Memory region list.

15.465.1 Detailed Description

A Port on the Virtio Net Switch.

A Port object gets created by `Virtio_factory::op_create()`. This function actually only instantiates objects of the types `Switch_port` and `Monitor_port`. The created Port registers itself at the switch's server. Usually, the IPC call for port creation comes from ned. To finalize the setup, the client has to initialize the port during the virtio initialization phase. To do this, the client registers a dataspace for queues and buffers and provides an IRQ to notify the client on incoming network requests.

Definition at line 36 of file [port_l4virtio.h](#).

15.465.2 Member Function Documentation

15.465.2.1 `drop_requests()`

```
void L4virtio_port::drop_requests () [inline]
```

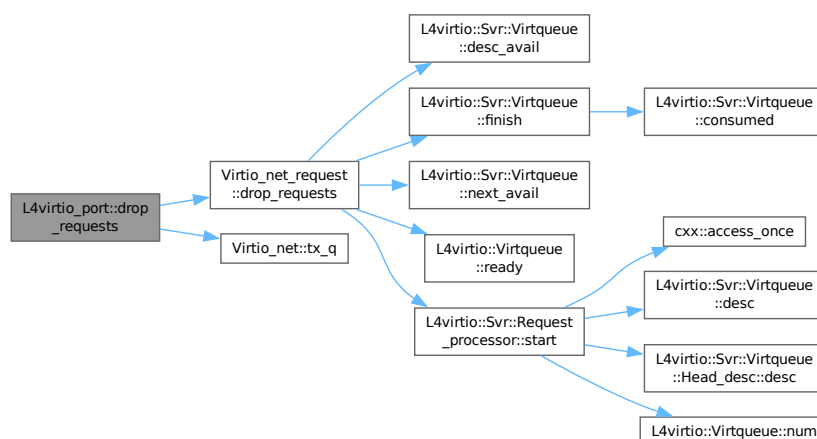
Drop all requests pending in the transmission queue.

This is used for monitor ports, which are not allowed to send packets.

Definition at line 103 of file [port_l4virtio.h](#).

References [Virtio_net_request::drop_requests\(\)](#), and [Virtio_net::tx_q\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

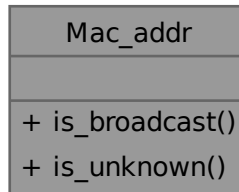
- `pkg/virtio-net-switch/server/switch/port_l4virtio.h`

15.466 Mac_addr Class Reference

A wrapper class around the value of a MAC address.

```
#include <mac_addr.h>
```

Collaboration diagram for Mac_addr:



Public Member Functions

- bool **is_broadcast** () const
Check if MAC address is a broadcast or multicast address.
- bool **is_unknown** () const
Check if the MAC address is not yet known.

15.466.1 Detailed Description

A wrapper class around the value of a MAC address.

Definition at line 19 of file [mac_addr.h](#).

The documentation for this class was generated from the following file:

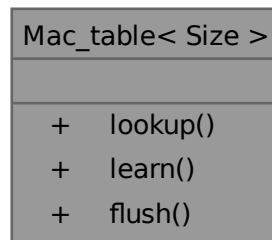
- [pkg/virtio-net-switch/server/switch/mac_addr.h](#)

15.467 Mac_table< Size > Class Template Reference

[Mac_table](#) manages a 1:n association between ports and MAC addresses.

```
#include <mac_table.h>
```

Collaboration diagram for `Mac_table< Size >`:



Public Member Functions

- `Port_iface * lookup (Mac_addr dst, l4_uint16_t vlan_id) const`
Find the destination port for a MAC address and VLAN id.
- `void learn (Mac_addr src, Port_iface *port, l4_uint16_t vlan_id)`
Learn a MAC address (add it to the MAC table).
- `void flush (Port_iface *port)`
Flush all associations with a given port.

15.467.1 Detailed Description

```
template<std::size_t Size = 1024U>
class Mac_table< Size >
```

`Mac_table` manages a 1:n association between ports and MAC addresses.

There are different types of devices which might be attached to a port. For a normal device the switch sees exactly one MAC address per port - the MAC address of the device attached to it. But there might be other devices like software bridges attached to the port sending packets with different MAC addresses to the port. Therefore the switch has to manage a 1:n association between ports and MAC addresses. The MAC table manages this association.

When a packet comes in we need to find the destination port for the packet and therefore perform a lookup based on the MAC address.

To prevent unbounded growth of the lookup table, the number of entries is limited. Replacement is done on a round-robin basis. If the capacity was reached, the oldest entry is evicted.

Definition at line 40 of file `mac_table.h`.

15.467.2 Member Function Documentation

15.467.2.1 flush()

```
template<std::size_t Size = 1024U>
void Mac_table< Size >::flush (
    Port_iface * port) [inline]
```

Flush all associations with a given port.

Parameters

| | |
|-------------|---------------------------------------|
| <i>port</i> | Pointer to port that is to be flushed |
|-------------|---------------------------------------|

This function removes all references to a given port from the MAC table. Since we manage a 1:n association between ports and MAC addresses there might be more than one entry for a given port and we have to iterate over the whole array to delete every reference to the port.

Definition at line 129 of file [mac_table.h](#).

15.467.2.2 learn()

```
template<std::size_t Size = 1024U>
void Mac_table< Size >::learn (
    Mac_addr src,
    Port_iface * port,
    14_uint16_t vlan_id) [inline]
```

Learn a MAC address (add it to the MAC table).

Parameters

| | |
|----------------|---|
| <i>src</i> | MAC address |
| <i>port</i> | Pointer to the port object that can be used to reach MAC address <i>src</i> |
| <i>vlan_id</i> | VLAN id of the packet destination. |

Will evict the oldest learned address from the table if the maximum capacity was reached and if the MAC address was not known yet. The source port of the table entry is always updated to cope with clients that move between ports.

Definition at line 78 of file [mac_table.h](#).

References [L4_UNLIKELY](#), and [lookup\(\)](#).

Here is the call graph for this function:



15.467.2.3 lookup()

```
template<std::size_t Size = 1024U>
Port_iface * Mac_table< Size >::lookup (
    Mac_addr dst,
    14_uint16_t vlan_id) const [inline]
```

Find the destination port for a MAC address and VLAN id.

Parameters

| | |
|----------------------|-------------|
| <i>dst</i> | MAC address |
| <i>vlan↔ _id</i> | VLAN id |

Return values

| | |
|----------------|------------------------------------|
| <i>nullptr</i> | The MAC address is not known (yet) |
| <i>other</i> | Pointer to the destination port |

Definition at line 58 of file [mac_table.h](#).

Referenced by [learn\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

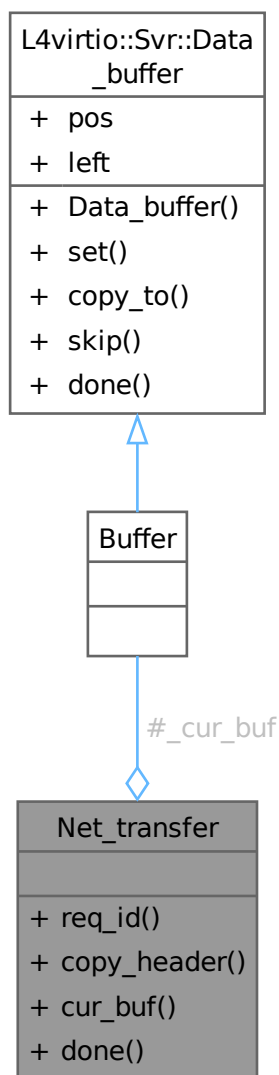
- [pkg/virtio-net-switch/server/switch/mac_table.h](#)

15.468 Net_transfer Class Reference

A network request to only a single destination.

```
#include <request.h>
```


Collaboration diagram for Net_transfer:



Public Member Functions

- void const * **req_id** () const
Identifier for the underlying `Net_request`, used for logging purposes.
- virtual void **copy_header** (Virtio_net::Hdr *dst_header) const =0
Populate the virtio-net header for the destination.
- [Buffer](#) & **cur_buf** ()
[Buffer](#) containing (a part of) the packet data.
- virtual bool **done** ()=0
Check whether the transfer has been completed, i.e.

15.468.1 Detailed Description

A network request to only a single destination.

A `Net_request` can have multiple destinations (being a broadcast request, for example). That is why it is processed by multiple `Net_transfers`, each representing the delivery to a single destination port.

`Port_iface::handle_request` uses the `Net_transfer` to move one packet to the destination of the request.

Definition at line 33 of file `request.h`.

15.468.2 Member Function Documentation

15.468.2.1 `cur_buf()`

```
Buffer & Net_transfer::cur_buf () [inline]
```

`Buffer` containing (a part of) the packet data.

Once emptied, a call to `done()` might replenish the buffer, in case the net request consisted of multiple chained buffers.

Definition at line 54 of file `request.h`.

15.468.2.2 `done()`

```
virtual bool Net_transfer::done () [pure virtual]
```

Check whether the transfer has been completed, i.e.

the entire packet data has been copied.

Return values

| | |
|--------------|---|
| <i>false</i> | There is remaining packet data that needs to be copied. |
| <i>true</i> | The entire packet data has been copied. |

Exceptions

| | |
|--|-------------------------------------|
| <code>L4virtio::Svr::Bad_descriptor</code> | Exception raised in SRC port queue. |
|--|-------------------------------------|

The documentation for this class was generated from the following file:

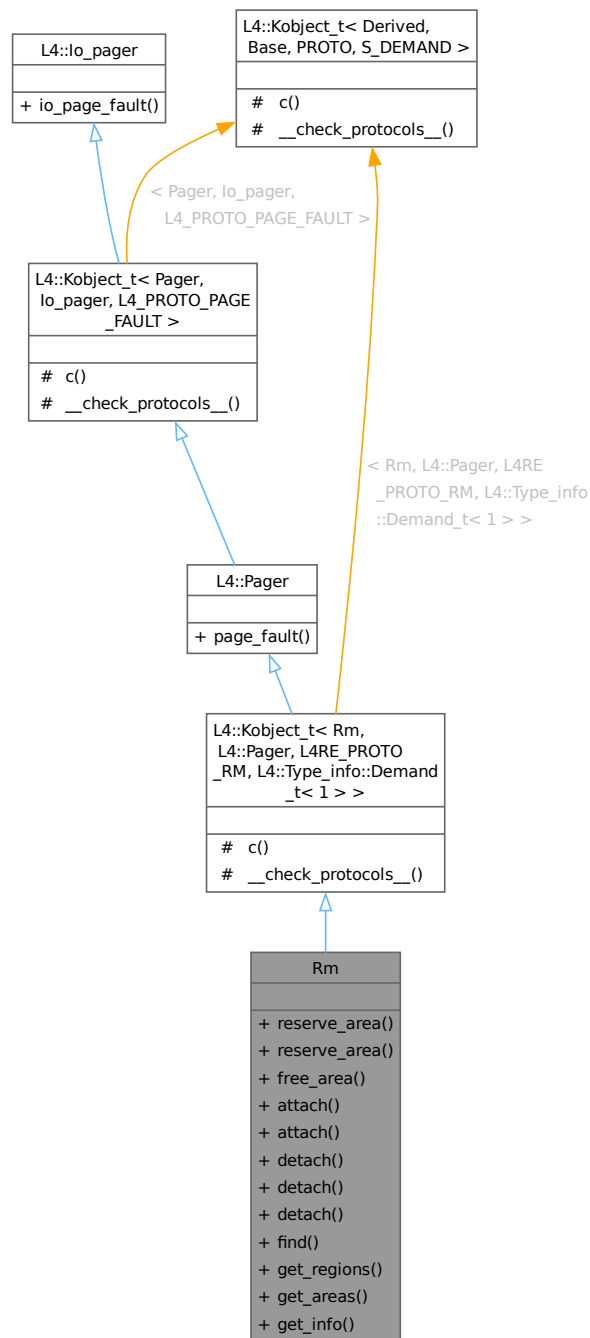
- `pkg/virtio-net-switch/server/switch/request.h`

15.469 Rm Class Reference

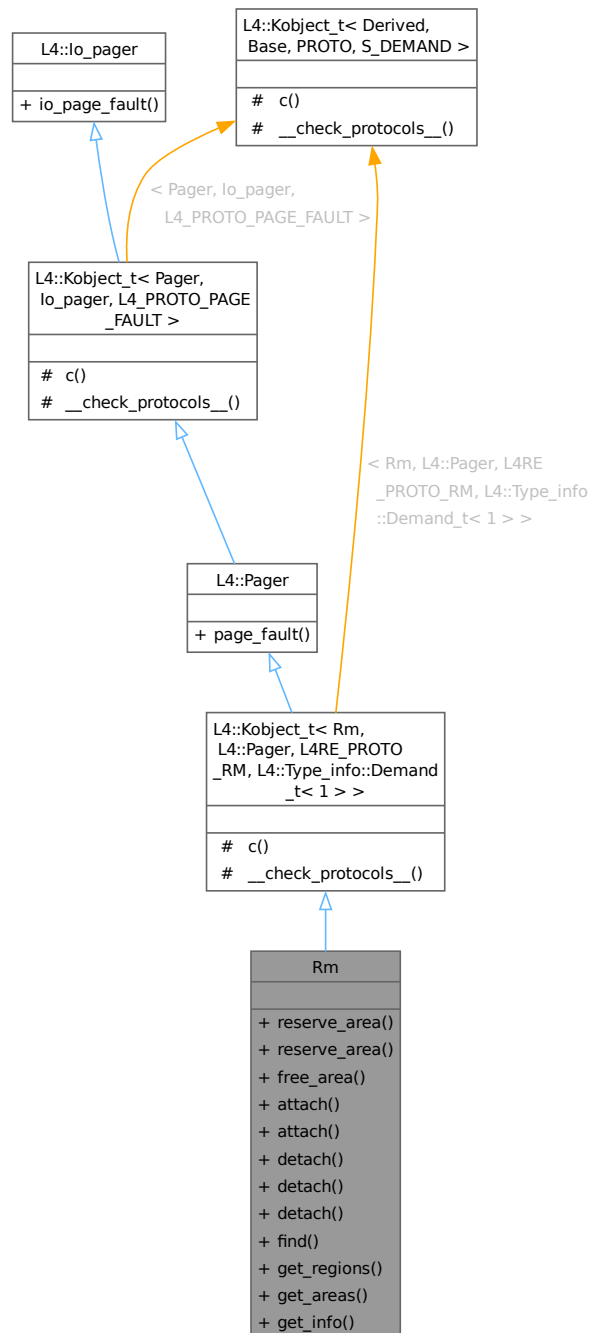
Region map.

```
#include <l4/re/rm>
```

Inheritance diagram for Rm:



Collaboration diagram for Rm:



Data Structures

- struct [F](#)
Rm flags definitions.
- class [Unique_region](#)
Unique region.
- struct [Region](#)

A region is a range of virtual addresses which is backed by content.

- struct [Area](#)

An area is a range of virtual addresses which is reserved, see [L4Re::Rm::reserve_area\(\)](#).

Public Types

- enum [Detach_result](#) { [Detached_ds](#) = 0 , [Kept_ds](#) = 1 , [Split_ds](#) = 2 , [Detach_result_mask](#) = 3 , [Detach_again](#) = 4 }
- Result values for detach operation.*
- enum [Region_flag_shifts](#) { [Caching_shift](#) = Dataspace::F::Caching_shift }
- Region flag shifts.*
- enum [Detach_flags](#) { [Detach_exact](#) = 1 , [Detach_overlap](#) = 2 , [Detach_keep](#) = 4 }
- Flags for detach operation.*

Public Member Functions

- [l4_ret_t reserve_area](#) ([l4_addr_t](#) *start, unsigned long size, Flags flags=[Flags](#)(0), unsigned char align=[L4_PAGESHIFT](#)) const noexcept
- Reserve the given area in the region map.*
- template<typename T>
[l4_ret_t reserve_area](#) (T **start, unsigned long size, Flags flags=[Flags](#)(0), unsigned char align=[L4_PAGESHIFT](#)) const noexcept
- Reserve the given area in the region map.*
- [l4_ret_t free_area](#) ([l4_addr_t](#) addr)
- Free an area from the region map.*
- [l4_ret_t attach](#) ([l4_addr_t](#) *start, unsigned long size, Flags flags, [L4::lpc::Cap](#)< Dataspace > mem, Offset offs=0, unsigned char align=[L4_PAGESHIFT](#), [L4::Cap](#)< [L4::Task](#) > const task=[L4::Cap](#)< [L4::Task](#) >::Invalid, char const *name=nullptr, Offset backing_offset=0) const noexcept
- Attach a data space to a region.*
- template<typename T>
[l4_ret_t attach](#) (T **start, unsigned long size, Flags flags, [L4::lpc::Cap](#)< Dataspace > mem, Offset offs=0, unsigned char align=[L4_PAGESHIFT](#), [L4::Cap](#)< [L4::Task](#) > const task=[L4::Cap](#)< [L4::Task](#) >::Invalid, char const *name=nullptr, Offset backing_offset=0) const noexcept
- Attach a data space to a region.*
- [l4_ret_t detach](#) ([l4_addr_t](#) addr, [L4::Cap](#)< Dataspace > *mem, [L4::Cap](#)< [L4::Task](#) > const &task=[This_task](#)) const noexcept
- Detach and unmap a region from the address space.*
- [l4_ret_t detach](#) (void *addr, [L4::Cap](#)< Dataspace > *mem, [L4::Cap](#)< [L4::Task](#) > const &task=[This_task](#)) const noexcept
- Detach and unmap a region from the address space.*
- [l4_ret_t detach](#) ([l4_addr_t](#) start, unsigned long size, [L4::Cap](#)< Dataspace > *mem, [L4::Cap](#)< [L4::Task](#) > const &task) const noexcept
- Detach and unmap all parts of the regions within the specified interval.*
- [l4_ret_t find](#) ([l4_addr_t](#) *addr, unsigned long *size, Offset *offset, [L4Re::Rm::Flags](#) *flags, [L4::Cap](#)< Dataspace > *m) noexcept
- Find a region given an address and size.*
- [l4_ret_t get_regions](#) ([l4_addr_t](#) start, [L4::lpc::Ret_array](#)< [Region](#) > regions)
- Return the list of regions whose starting addresses are higher or equal to start in the address space managed by this region map.*
- long [get_areas](#) ([l4_addr_t](#) start, [L4::lpc::Ret_array](#)< [Area](#) > areas)
- Return the list of areas whose starting addresses are higher or equal to start in the address space managed by this region map.*
- [l4_ret_t get_info](#) ([l4_addr_t](#) addr, [L4::lpc::String](#)< char > &name, Offset &backing_offset)
- Return auxiliary information of a region.*

Public Member Functions inherited from [L4::Pager](#)

- [l4_msgtag_t page_fault](#) ([l4_umword_t](#) pfa, [l4_umword_t](#) pc, [L4::lpc::Rcv_fpage](#) rwin, [L4::lpc::Opt<L4::lpc::Snd_fpage & > fp](#))

Page-fault protocol message.

Public Member Functions inherited from [L4::io_pager](#)

- [l4_msgtag_t io_page_fault](#) ([l4_fpage_t](#) io_pfa, [l4_umword_t](#) pc, [L4::lpc::Rcv_fpage](#) rwin, [L4::lpc::Opt<L4::lpc::Snd_fpage & > fp](#))

IO page fault protocol message.

Additional Inherited Members

Protected Types inherited from

[L4::Kobject_t< Rm, L4::Pager, L4RE_PROTO_RM, L4::Type_info::Demand_t< 1 > >](#)

- typedef [Rm](#) **Class**
The target interface type (inheriting from [Kobject_t](#)).
- typedef Typeid::Iface< PROTO, [Rm](#) > **__Iface**
The interface description for the derived class.
- typedef Typeid::Merge_list< Typeid::Iface_list< **__Iface** >, typename [L4::Pager::__Iface_list](#) > **__Iface_list**
The list of all RPC interfaces provided directly or through inheritance.

Protected Types inherited from

[L4::Kobject_t< Pager, io_pager, L4_PROTO_PAGE_FAULT >](#)

- typedef [Pager](#) **Class**
The target interface type (inheriting from [Kobject_t](#)).
- typedef Typeid::Iface< PROTO, [Pager](#) > **__Iface**
The interface description for the derived class.
- typedef Typeid::Merge_list< Typeid::Iface_list< **__Iface** >, typename [io_pager::__Iface_list](#) > **__Iface_list**
The list of all RPC interfaces provided directly or through inheritance.

Protected Member Functions inherited from

[L4::Kobject_t< Rm, L4::Pager, L4RE_PROTO_RM, L4::Type_info::Demand_t< 1 > >](#)

- [L4::Cap< Class > c](#) () const noexcept
Get the capability to ourselves.

Protected Member Functions inherited from

[L4::Kobject_t< Pager, io_pager, L4_PROTO_PAGE_FAULT >](#)

- [L4::Cap< Class > c](#) () const noexcept
Get the capability to ourselves.

Static Protected Member Functions inherited from**[L4::Kobject_t< Rm, L4::Pager, L4RE_PROTO_RM, L4::Type_info::Demand_t< 1 > >](#)**

- static void **`__check_protocols__`** () noexcept
Helper to check for protocol conflicts.

Static Protected Member Functions inherited from**[L4::Kobject_t< Pager, lo_pager, L4_PROTO_PAGE_FAULT >](#)**

- static void **`__check_protocols__`** () noexcept
Helper to check for protocol conflicts.

15.469.1 Detailed Description[Region](#) map.

See also

[Region map API](#) .Definition at line [81](#) of file [rm](#).**15.469.2 Member Enumeration Documentation****15.469.2.1 Detach_flags**enum [L4Re::Rm::Detach_flags](#)

Flags for detach operation.

Enumerator

| | |
|--------------------------------|--|
| Detach_exact | Do an unmap of the exact region given. |
| Detach_overlap | Do an unmap of all overlapping regions. |
| Detach_keep | Do not free the detached data space, ignore the F::Detach_free . |

Definition at line [216](#) of file [rm](#).

15.469.2.2 Detach_result

enum `L4Re::Rm::Detach_result`

Result values for detach operation.

Enumerator

| | |
|--------------|----------------------------------|
| Detached_ds | Detached data space. |
| Kept_ds | Kept data space. |
| Split_ds | Split data space, and done. |
| Detach_again | Detached data space, more to do. |

Definition at line 89 of file `rm`.

15.469.2.3 Region_flag_shifts

enum `L4Re::Rm::Region_flag_shifts`

`Region` flag shifts.

Enumerator

| | |
|---------------|--------------------------------------|
| Caching_shift | Start of <code>Rm</code> cache bits. |
|---------------|--------------------------------------|

Definition at line 100 of file `rm`.

15.469.3 Member Function Documentation

15.469.3.1 attach() [1/2]

```
l4_ret_t L4Re::Rm::attach (
    l4_addr_t * start,
    unsigned long size,
    Rm::Flags flags,
    L4::Ipc::Cap< Dataspace > mem,
    Rm::Offset offs = 0,
    unsigned char align = L4_PAGESHIFT,
    L4::Cap< L4::Task > const task = L4::Cap<L4::Task>::Invalid,
    char const * name = nullptr,
    Rm::Offset backing_offset = 0) const [noexcept]
```

Attach a data space to a region.

Parameters

| | | |
|----------------|-----------------------|---|
| <i>in, out</i> | <i>start</i> | Virtual start address where the region manager shall attach the data space. Will be rounded down to the nearest start of a page. If L4Re::Rm::F::Search_addr is given this value is used as the start address to search for a free virtual memory region and the resulting address is returned here. If L4Re::Rm::F::In_area is given the value is used as a selector for the area (see L4Re::Rm::reserve_area) to attach the data space to. |
| | <i>size</i> | Size of the data space to attach (in bytes). Will be rounded up to the nearest multiple of the page size. |
| | <i>flags</i> | The flags control how and with which rights the dataspace is attached to the region. See L4Re::Rm::F::Attach_flags and L4Re::Rm::F::Region_flags . The caller must specify the desired rights of the attached region explicitly. The default set of rights is empty. If the F::Eager_map flag is set this function may also return L4Re::Dataspace::map error codes if the mapping fails. |
| | <i>mem</i> | Data space. |
| | <i>offs</i> | Offset into the data space to use. |
| | <i>align</i> | Alignment of the virtual region, log2-size, default: a page (L4_PAGESHIFT). This is only meaningful if the L4Re::Rm::F::Search_addr flag is used. |
| | <i>task</i> | Optional destination task of mapping if F::Eager_map flag was passed. If invalid, the mapping is established in the current task. This parameter is only useful if the region manager is for a foreign task. |
| | <i>name</i> | Optional name of the region. |
| | <i>backing_offset</i> | Optional value describing an offset into the backing store of this region. |

Return values

| | |
|--------------------------|--|
| <i>0</i> | Success |
| <i>-L4_ENOENT</i> | No area could be found (see L4Re::Rm::F::In_area) |
| <i>-L4_EPERM</i> | Operation not allowed. |
| <i>-L4_EINVAL</i> | |
| <i>-L4_EADDRNOTAVAIL</i> | The given address is not available. |
| <i><0</i> | IPC errors |

Makes the whole or parts of a data space visible in the virtual memory of the corresponding task. The corresponding region in the virtual address space is backed with the contents of the dataspace.

Note

When searching for a free place in the virtual address space, the space between *start* and the end of the virtual address space is searched.

There is no region object created, instead the region is defined by a virtual address within this range (see [L4Re::Rm::find](#)).

Definition at line 35 of file [rm_impl.h](#).

15.469.3.2 attach() [2/2]

```
template<typename T>
l4_ret_t L4Re::Rm::attach (
```

```

T ** start,
unsigned long size,
Flags flags,
L4::Ipc::Cap< Dataspace > mem,
Offset offs = 0,
unsigned char align = L4_PAGESHIFT,
L4::Cap< L4::Task > const task = L4::Cap<L4::Task>::Invalid,
char const * name = nullptr,
Offset backing_offset = 0) const [inline], [noexcept]

```

Attach a data space to a region.

Parameters

| | | |
|---------|-----------------------|---|
| in, out | <i>start</i> | Virtual start address where the region manager shall attach the data space. Will be rounded down to the nearest start of a page. If L4Re::Rm::F::Search_addr is given this value is used as the start address to search for a free virtual memory region and the resulting address is returned here. If L4Re::Rm::F::In_area is given the value is used as a selector for the area (see L4Re::Rm::reserve_area) to attach the data space to. |
| | <i>size</i> | Size of the data space to attach (in bytes). Will be rounded up to the nearest multiple of the page size. |
| | <i>flags</i> | The flags control how and with which rights the dataspace is attached to the region. See L4Re::Rm::F::Attach_flags and L4Re::Rm::F::Region_flags . The caller must specify the desired rights of the attached region explicitly. The default set of rights is empty. If the F::Eager_map flag is set this function may also return L4Re::Dataspace::map error codes if the mapping fails. |
| | <i>mem</i> | Data space. |
| | <i>offs</i> | Offset into the data space to use. |
| | <i>align</i> | Alignment of the virtual region, log2-size, default: a page (L4_PAGESHIFT). This is only meaningful if the L4Re::Rm::F::Search_addr flag is used. |
| | <i>task</i> | Optional destination task of mapping if F::Eager_map flag was passed. If invalid, the mapping is established in the current task. This parameter is only useful if the region manager is for a foreign task. |
| | <i>name</i> | Optional name of the region. |
| | <i>backing_offset</i> | Optional value describing an offset into the backing store of this region. |

Return values

| | |
|-------------------|--|
| 0 | Success |
| -L4_ENOENT | No area could be found (see L4Re::Rm::F::In_area) |
| -L4_EPERM | Operation not allowed. |
| -L4_EINVAL | |
| -L4_EADDRNOTAVAIL | The given address is not available. |
| <0 | IPC errors |

Makes the whole or parts of a data space visible in the virtual memory of the corresponding task. The corresponding region in the virtual address space is backed with the contents of the dataspace.

Note

When searching for a free place in the virtual address space, the space between *start* and the end of the virtual address space is searched.

There is no region object created, instead the region is defined by a virtual address within this range (see [L4Re::Rm::find](#)).

Definition at line 409 of file [rm](#).

15.469.3.3 detach() [1/3]

```
l4_ret_t L4Re::Rm::detach (
    l4_addr_t addr,
    L4::Cap< Dataspace > * mem,
    L4::Cap< L4::Task > const & task = This_task) const [inline], [noexcept]
```

Detach and unmap a region from the address space.

Parameters

| | | |
|-----|-------------|--|
| | <i>addr</i> | Virtual address of region, any address within the region is valid. |
| out | <i>mem</i> | Dataspace that is affected. Give 0 if not interested. |
| | <i>task</i> | This argument specifies the task where the pages are unmapped. Provide L4::Cap<L4::Task>::Invalid for none. The default is the current task. |

Return values

| | |
|---|------------------|
| L4Re::Rm::Detach_result | On success. |
| <code>-L4_ENOENT</code> | No region found. |
| <code><0</code> | IPC errors |

Frees a region in the virtual address space given by *addr* (address type). The corresponding part of the address space is now available again.

Definition at line 766 of file [rm](#).

15.469.3.4 detach() [2/3]

```
l4_ret_t L4Re::Rm::detach (
    l4_addr_t start,
    unsigned long size,
    L4::Cap< Dataspace > * mem,
    L4::Cap< L4::Task > const & task) const [inline], [noexcept]
```

Detach and unmap all parts of the regions within the specified interval.

Parameters

| | | |
|-----|--------------|--|
| | <i>start</i> | Start of area to detach, must be within region. |
| | <i>size</i> | Size of of area to detach (in bytes). |
| out | <i>mem</i> | Dataspace that is affected. Give 0 if not interested. |
| | <i>task</i> | This argument specifies the task where the pages are unmapped. Provide L4::Cap<L4::Task>::Invalid for none. The default is the current task. |

Return values

| | |
|---|------------------|
| L4Re::Rm::Detach_result | On success. |
| -L4_ENOENT | No region found. |
| <0 | IPC errors |

Frees all regions within the interval given by start and size. If a region overlaps the start or the end of the interval this region is only detached partly. If the interval is within one region the original region is split up into two separate regions.

Definition at line [779](#) of file [rm](#).

15.469.3.5 detach() [3/3]

```
l4_ret_t L4Re::Rm::detach (
    void * addr,
    L4::Cap< Dataspace > * mem,
    L4::Cap< L4::Task > const & task = This_task) const [inline], [noexcept]
```

Detach and unmap a region from the address space.

Parameters

| | | |
|-----|-------------|--|
| | <i>addr</i> | Virtual address of region, any address within the region is valid. |
| out | <i>mem</i> | Dataspace that is affected. Give 0 if not interested. |
| | <i>task</i> | This argument specifies the task where the pages are unmapped. Provide L4::Cap<L4::Task>::Invalid for none. The default is the current task. |

Return values

| | |
|---|------------------|
| L4Re::Rm::Detach_result | On success. |
| -L4_ENOENT | No region found. |
| <0 | IPC errors |

Frees a region in the virtual address space given by addr (address type). The corresponding part of the address space is now available again.

Definition at line [771](#) of file [rm](#).

15.469.3.6 find()

```
l4_ret_t L4Re::Rm::find (
    l4_addr_t * addr,
    unsigned long * size,
    Offset * offset,
    L4Re::Rm::Flags * flags,
    L4::Cap< Dataspace > * m) [inline], [noexcept]
```

Find a region given an address and size.

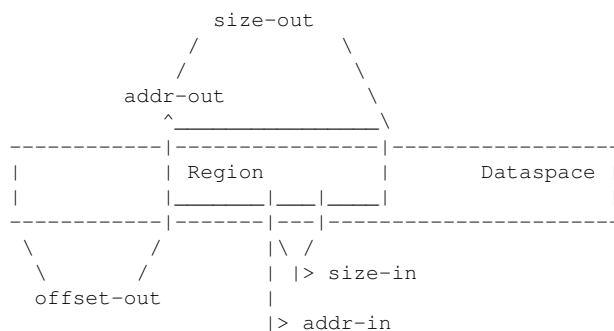
Parameters

| | | |
|---------|---------------|---|
| in, out | <i>addr</i> | Address to look for. Returns the start address of the found region. |
| in, out | <i>size</i> | Size of the area to look for (in bytes). Returns the size of the found region (in bytes). |
| out | <i>offset</i> | Offset at the beginning of the region within the associated dataspace. |
| out | <i>flags</i> | Region flags, see F::Region_flags (and F::In_area). |
| out | <i>m</i> | Associated dataspace or paging service. |

Return values

| | |
|------------|------------------------|
| 0 | Success |
| -L4_EPERM | Operation not allowed. |
| -L4_ENOENT | No region found. |
| <0 | IPC errors |

This function returns the properties of the region that contains the area described by the *addr* and *size* parameter. If no such region is found but a reserved area, the area is returned and [F::In_area](#) is set in *flags*. Note, in the case of an area the *offset* and *m* return values are invalid.



Note

The value of the *size* input parameter should be 1 to assure that a region can be determined unambiguously.

Definition at line 675 of file [rm](#).

15.469.3.7 free_area()

```
l4_ret_t L4Re::Rm::free_area (
    l4_addr_t addr)
```

Free an area from the region map.

Parameters

| | |
|-------------|-------------------------------------|
| <i>addr</i> | An address within the area to free. |
|-------------|-------------------------------------|

Return values

| | |
|------------|----------------|
| 0 | Success |
| -L4_ENOENT | No area found. |
| <0 | IPC errors |

Note

The data spaces that are attached to that area are not detached by this operation.

See also

[reserve_area\(\)](#) for more information about areas.

15.469.3.8 get_areas()

```
long L4Re::Rm::get_areas (
    l4_addr_t start,
    L4::Ipc::Ret_array< Area > areas)
```

Return the list of areas whose starting addresses are higher or equal to *start* in the address space managed by this region map.

Parameters

| | | |
|-----|--------------|--|
| | <i>start</i> | Virtual address from where to start searching. |
| out | <i>areas</i> | List of areas found in this region map. |

Return values

| | |
|-----|---|
| >=0 | Number of returned areas in the <i>areas</i> array. |
| <0 | IPC errors |

Note

The returned list of areas might not be complete and the caller shall use the function repeatedly with a start address one larger than the end address of the last area from the previous call.

15.469.3.9 get_info()

```
l4_ret_t L4Re::Rm::get_info (
    l4_addr_t addr,
    L4::Ipc::String< char > & name,
    Offset & backing_offset)
```

Return auxiliary information of a region.

This is a debugging feature and might not be available.

Parameters

| | | |
|-----|-----------------------|--------------------------------|
| | <i>addr</i> | Virtual address of the region. |
| out | <i>name</i> | Name of the region. |
| out | <i>backing_offset</i> | Backing offset information. |

Return values

| | |
|-------------------|-----------------------------------|
| <i>0</i> | Success |
| <i>-L4_ENOENT</i> | Region not found. |
| <i>-L4_ENOSYS</i> | Function not available. |
| <i><0</i> | IPC errors |

15.469.3.10 get_regions()

```
l4_ret_t L4Re::Rm::get_regions (
    l4_addr_t start,
    L4::Ipc::Ret_array< Region > regions)
```

Return the list of regions whose starting addresses are higher or equal to *start* in the address space managed by this region map.

Parameters

| | | |
|-----|----------------|--|
| | <i>start</i> | Virtual address from where to start searching. |
| out | <i>regions</i> | List of regions found in this region map. |

Return values

| | |
|---------------|---|
| <i>>=0</i> | Number of returned regions in the <i>regions</i> array. |
| <i><0</i> | IPC errors |

Note

The returned list of regions might not be complete and the caller shall use the function repeatedly with a start address one larger than the end address of the last region from the previous call.

15.469.3.11 reserve_area() [1/2]

```
l4_ret_t L4Re::Rm::reserve_area (
    l4_addr_t * start,
    unsigned long size,
    Flags flags = Flags(0),
    unsigned char align = L4_PAGESHIFT) const [inline], [noexcept]
```

Reserve the given area in the region map.

Parameters

| | | |
|----------------|--------------|---|
| <i>in, out</i> | <i>start</i> | The virtual start address of the area to reserve. Returns the start address of the area. |
| | <i>size</i> | The size of the area to reserve (in bytes). |
| | <i>flags</i> | Flags for the reserved area (see L4Re::Rm::F::Region_flags and L4Re::Rm::F::Attach_flags). |
| | <i>align</i> | Alignment of area if searched as bits (log2 value). |

Return values

| | |
|--------------------------|------------------------------------|
| <i>0</i> | Success |
| <i>-L4_EADDRNOTAVAIL</i> | The given area cannot be reserved. |
| <i><0</i> | IPC errors |

This function reserves an area within the virtual address space managed by the region map. There are two kinds of areas available:

- Reserved areas (*flags* = [L4Re::Rm::F::Reserved](#)), where no data spaces can be attached
- Special purpose areas (*flags* = 0), where data spaces can be attached to the area via the [L4Re::Rm::F::In_area](#) flag and a start address within the area itself.

Note

When searching for a free place in the virtual address space (with *flags* = [L4Re::Rm::F::Search_addr](#)), the space between *start* and the end of the virtual address space is searched.

Definition at line [277](#) of file [rm](#).

15.469.3.12 reserve_area() [2/2]

```
template<typename T>
l4_ret_t L4Re::Rm::reserve_area (
    T ** start,
    unsigned long size,
    Flags flags = Flags(0),
    unsigned char align = L4_PAGESHIFT) const [inline], [noexcept]
```

Reserve the given area in the region map.

Parameters

| | | |
|----------------|--------------|---|
| <i>in, out</i> | <i>start</i> | The virtual start address of the area to reserve. Returns the start address of the area. |
| | <i>size</i> | The size of the area to reserve (in bytes). |
| | <i>flags</i> | Flags for the reserved area (see F::Region_flags and F::Attach_flags). |
| | <i>align</i> | Alignment of area if searched as bits (log2 value). |

Return values

| | |
|-------------------|------------------------------------|
| 0 | Success |
| -L4_EADDRNOTAVAIL | The given area cannot be reserved. |
| <0 | IPC errors |

For more information, please refer to the analogous function

See also

[L4Re::Rm::reserve_area](#).

Definition at line 305 of file [rm](#).

The documentation for this class was generated from the following files:

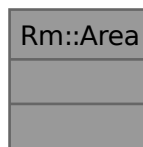
- [l4/re/rm](#)
- [l4/re/impl/rm_impl.h](#)

15.470 Rm::Area Struct Reference

An area is a range of virtual addresses which is reserved, see [L4Re::Rm::reserve_area\(\)](#).

```
#include <rm>
```

Collaboration diagram for Rm::Area:



15.470.1 Detailed Description

An area is a range of virtual addresses which is reserved, see [L4Re::Rm::reserve_area\(\)](#).

See also

[Region map API](#)

Definition at line 702 of file [rm](#).

The documentation for this struct was generated from the following file:

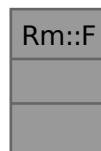
- [l4/re/rm](#)

15.471 Rm::F Struct Reference

[Rm](#) flags definitions.

```
#include <rm>
```

Collaboration diagram for [Rm::F](#):



Public Types

- enum [Attach_flags](#) : [l4_uint32_t](#) {
[Search_addr](#) = 0x20000 , [In_area](#) = 0x40000 , [Eager_map](#) = 0x80000 , [No_eager_map](#) = 0x100000 ,
[Attach_mask](#) = 0x1f0000 }
Flags for attach operation.
- enum [Region_flags](#) : [l4_uint16_t](#) {
[Rights_mask](#) = 0x0f , [R](#) = [Dataspace::F::R](#) , [W](#) = [Dataspace::F::W](#) , [X](#) = [Dataspace::F::X](#) ,
[RW](#) = [Dataspace::F::RW](#) , [RX](#) = [Dataspace::F::RX](#) , [RWX](#) = [Dataspace::F::RWX](#) , [Kernel](#) = 0x100 ,
[Detach_free](#) = 0x200 , [Pager](#) = 0x400 , [Reserved](#) = 0x800 , [Caching_mask](#) = [Dataspace::F::Caching_mask](#) ,
[Cache_normal](#) = [Dataspace::F::Normal](#) , [Cache_buffered](#) = [Dataspace::F::Bufferable](#) , [Cache_uncached](#) =
[Dataspace::F::Uncacheable](#) , [Ds_map_mask](#) = 0xff ,
[Region_flags_mask](#) = 0xffff }
Region flags (permissions, cacheability, special).

15.471.1 Detailed Description

[Rm](#) flags definitions.

Definition at line 107 of file [rm](#).

15.471.2 Member Enumeration Documentation

15.471.2.1 Attach_flags

```
enum L4Re::Rm::F::Attach_flags : 14_uint32_t
```

Flags for attach operation.

Enumerator

| | |
|--------------|---|
| Search_addr | Search for a suitable address range. |
| In_area | Search only in area, or map into area. |
| Eager_map | Eagerly map the attached data space in. |
| No_eager_map | Prevent eager mapping of the attached data space. |
| Attach_mask | Mask of all attach flags. |

Definition at line 110 of file [rm](#).

15.471.2.2 Region_flags

```
enum L4Re::Rm::F::Region_flags : 14_uint16_t
```

[Region](#) flags (permissions, cacheability, special).

Enumerator

| | |
|-------------|---|
| Rights_mask | Region rights. |
| R | Readable region. |
| W | Writable region. |
| X | Executable region. |
| RW | Readable and writable region. |
| RX | Readable and executable region. |
| RWX | Readable, writable and executable region. |
| Kernel | Kernel-provided memory (KUMEM). |
| Detach_free | Free the portion of the data space after detach. |
| Pager | Region has a pager. Page faults in this region are not handled by ITAS but rather forwarded to the external region manager / pager. |
| Reserved | Region is reserved (blocked). |

| | |
|-------------------|---|
| Caching_mask | Mask of all Rm cache bits. |
| Cache_normal | Cache bits for normal cacheable memory. This is the default if no other cache-related flag was specified. |
| Cache_buffered | Cache bits for buffered (write combining) memory. |
| Cache_uncached | Cache bits for uncached memory. |
| Ds_map_mask | Mask for all bits for cache options and rights. |
| Region_flags_mask | Mask of all region flags. |

Definition at line [128](#) of file [rm](#).

The documentation for this struct was generated from the following file:

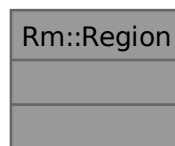
- [l4/re/rm](#)

15.472 Rm::Region Struct Reference

A region is a range of virtual addresses which is backed by content.

```
#include <rm>
```

Collaboration diagram for Rm::Region:



15.472.1 Detailed Description

A region is a range of virtual addresses which is backed by content.

See also

[Region map API](#)

Definition at line [689](#) of file [rm](#).

The documentation for this struct was generated from the following file:

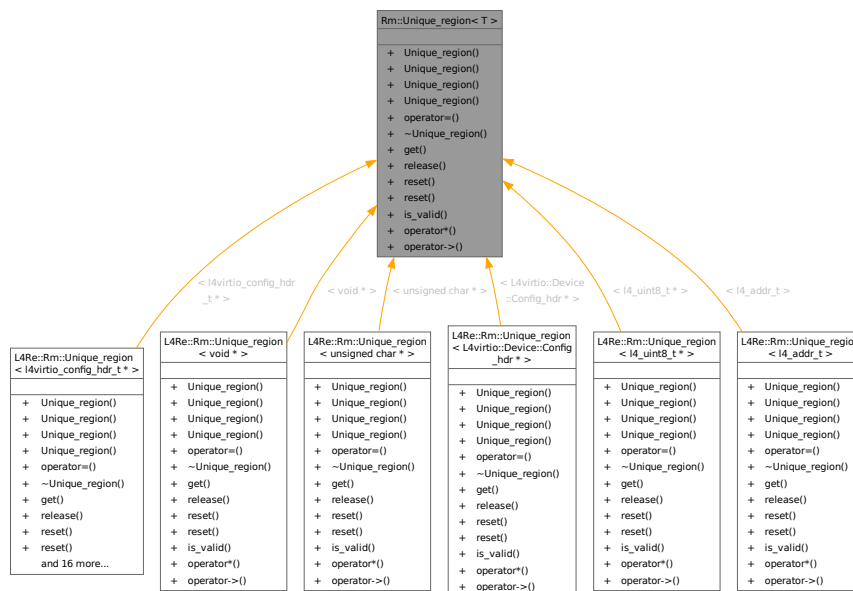
- [l4/re/rm](#)

15.473 Rm::Unique_region< T > Class Template Reference

Unique region.

```
#include <rm>
```

Inheritance diagram for Rm::Unique_region< T >:



Collaboration diagram for Rm::Unique_region< T >:

| Rm::Unique_region< T > |
|---|
| <ul style="list-style-type: none"> + Unique_region() + Unique_region() + Unique_region() + Unique_region() + operator=() + ~Unique_region() + get() + release() + reset() + reset() + is_valid() + operator*() + operator->() |

Public Member Functions

- **Unique_region** () noexcept
Construct an invalid [Unique_region](#).
- [Unique_region](#) (T addr) noexcept
Construct a [Unique_region](#) from an address.
- [Unique_region](#) (T addr, [L4::Cap](#)< [Rm](#) > const &rm) noexcept
Construct a valid [Unique_region](#) from an address and a region manager.
- [Unique_region](#) (Unique_region &&o) noexcept
Move-Construct a [Unique_region](#).
- Unique_region & [operator=](#) (Unique_region &&o) noexcept
Move-assign a [Unique_region](#).
- [~Unique_region](#) () noexcept
Destructor.
- T [get](#) () const noexcept
Return the address.
- T [release](#) () noexcept
Return the address and invalidate the [Unique_region](#).
- void [reset](#) (T addr, [L4::Cap](#)< [Rm](#) > const &rm) noexcept
Set new address and region manager.
- void **reset** () noexcept
Make the [Unique_region](#) invalid.

- bool `is_valid` () const noexcept
Check if the `Unique_region` is valid.
- T `operator*` () const noexcept
Dereference the address.
- T `operator->` () const noexcept
Member access for the address.

15.473.1 Detailed Description

```
template<typename T>
class Rm::Unique_region< T >
```

Unique region.

Capture a single region with automatic detach on destruction and unique ownership. Stores the start address and the region-mapper capability internally. A unique region is valid precisely if the internal region-mapper capability is valid. The features for unique ownership and automatic detach are only active for valid unique regions.

Definition at line 435 of file `rm`.

15.473.2 Constructor & Destructor Documentation

15.473.2.1 Unique_region() [1/3]

```
template<typename T>
L4Re::Rm::Unique_region< T >::Unique_region (
    T addr) [inline], [explicit], [noexcept]
```

Construct a `Unique_region` from an address.

No region manager is set.

Parameters

| | |
|-------------------|-----------------|
| <code>addr</code> | The new address |
|-------------------|-----------------|

Definition at line 456 of file `rm`.

15.473.2.2 Unique_region() [2/3]

```
template<typename T>
L4Re::Rm::Unique_region< T >::Unique_region (
    T addr,
    L4::Cap< Rm > const & rm) [inline], [noexcept]
```

Construct a valid [Unique_region](#) from an address and a region manager.

Parameters

| | |
|-------------|--------------------|
| <i>addr</i> | The address |
| <i>rm</i> | The region manager |

Definition at line [465](#) of file [rm](#).

15.473.2.3 Unique_region() [3/3]

```
template<typename T>
L4Re::Rm::Unique_region< T >::Unique_region (
    Unique_region< T > && o) [inline], [noexcept]
```

Move-Construct a [Unique_region](#).

Parameters

| | |
|----------|------------------------------------|
| <i>o</i> | L-value reference to other region. |
|----------|------------------------------------|

Definition at line [473](#) of file [rm](#).

15.473.2.4 ~Unique_region()

```
template<typename T>
L4Re::Rm::Unique_region< T >::~~Unique_region () [inline], [noexcept]
```

Destructor.

If the region is valid, call [detach](#).

Definition at line [498](#) of file [rm](#).

15.473.3 Member Function Documentation

15.473.3.1 get()

```
template<typename T>
T L4Re::Rm::Unique_region< T >::get () const [inline], [noexcept]
```

Return the address.

Returns

the address

Definition at line [509](#) of file [rm](#).

15.473.3.2 is_valid()

```
template<typename T>
bool L4Re::Rm::Unique_region< T >::is_valid () const [inline], [noexcept]
```

Check if the [Unique_region](#) is valid.

Returns

true iff the [Unique_region](#) is valid

Definition at line [549](#) of file [rm](#).

15.473.3.3 operator=()

```
template<typename T>
Unique_region & L4Re::Rm::Unique_region< T >::operator= (
    Unique_region< T > && o) [inline], [noexcept]
```

Move-assign a [Unique_region](#).

Parameters

| | |
|----------|--|
| <i>o</i> | L-value reference to region to assign from |
|----------|--|

Definition at line [481](#) of file [rm](#).

15.473.3.4 release()

```
template<typename T>
T L4Re::Rm::Unique_region< T >::release () [inline], [noexcept]
```

Return the address and invalidate the [Unique_region](#).

Returns

the address

Definition at line [517](#) of file [rm](#).

15.473.3.5 reset()

```
template<typename T>
void L4Re::Rm::Unique_region< T >::reset (
    T addr,
    L4::Cap< Rm > const & rm) [inline], [noexcept]
```

Set new address and region manager.

Parameters

| | |
|-------------|------------------------|
| <i>addr</i> | The new address |
| <i>rm</i> | The new region manager |

Definition at line [529](#) of file [rm](#).

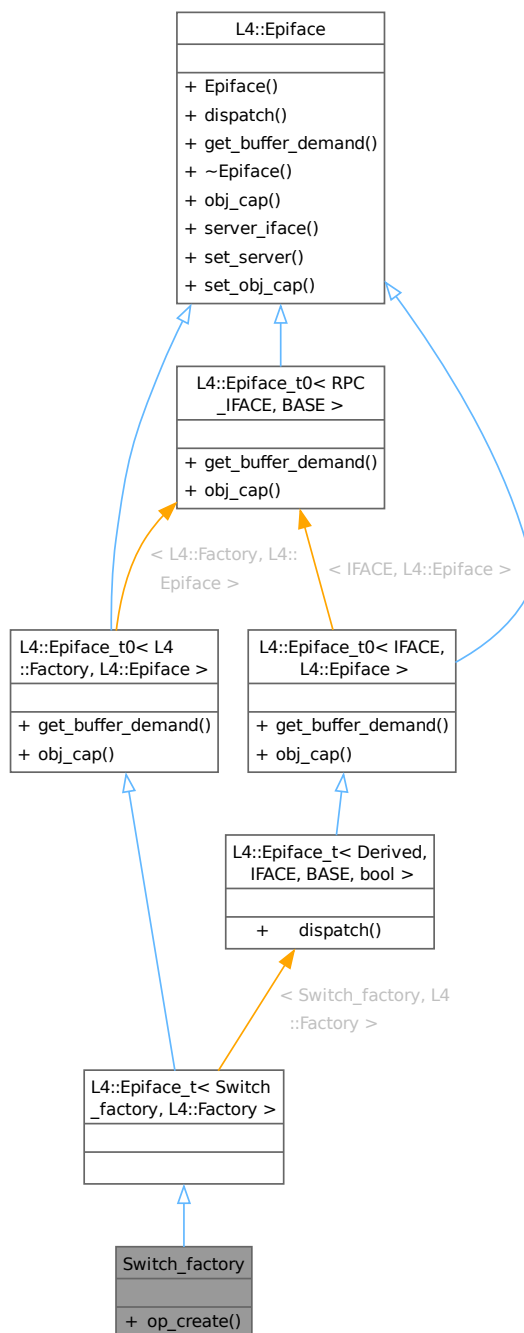
The documentation for this class was generated from the following file:

- [l4/re/rm](#)

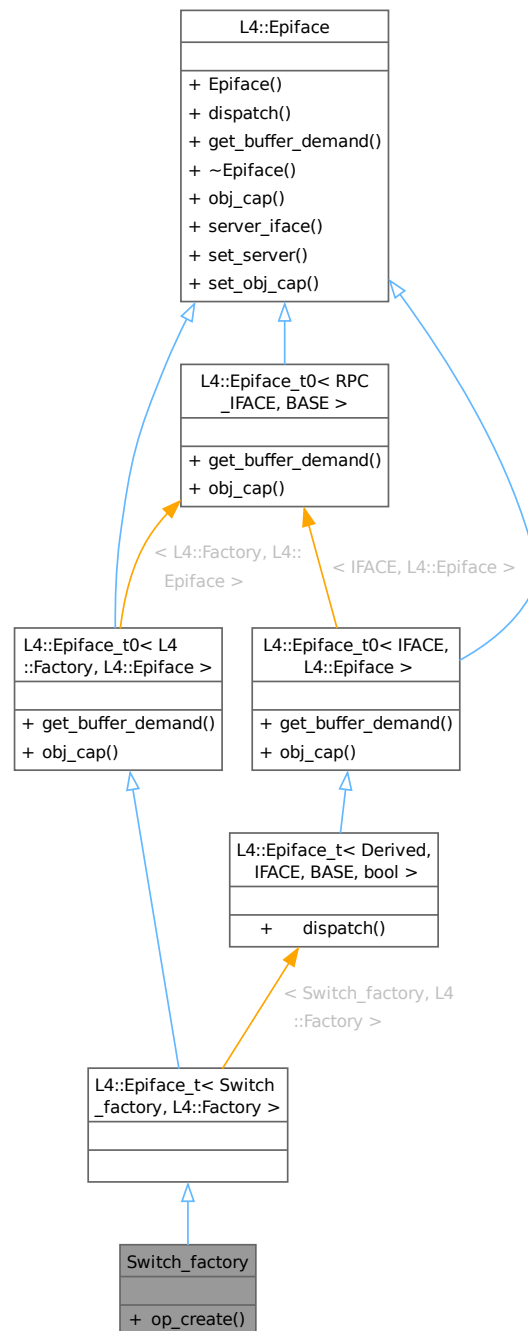
15.474 Switch_factory Class Reference

The IPC interface for creating ports.

Inheritance diagram for Switch_factory:



Collaboration diagram for Switch_factory:



Public Member Functions

- long `op_create` (L4::Factory::Rights, L4::ipc::Cap< void > &res, l4_umword_t type, L4::ipc::Varg_list_ref va)
Handle factory protocol.

Public Member Functions inherited from L4::Epiface_t0< L4::Factory, L4::Epiface >

- Type_info::Demand `get_buffer_demand` () const

Get the server-side buffer demand based in IFACE.

- `Cap< L4::Factory > obj_cap () const`

Get the (typed) capability to this object.

Public Member Functions inherited from `L4::Epiface`

- `Epiface ()`

Make a server object.

- `virtual ~Epiface ()=0`

Destroy the object.

- `Stored_cap obj_cap () const`

Get the capability to the kernel object belonging to this object.

- `Server_iface * server_iface () const`

Get pointer to server interface at which the object is currently registered.

- `int set_server (Server_iface *srv, Cap< void > cap, bool managed=false)`

Set server registration info for the object.

- `void set_obj_cap (Cap< void > const &cap)`

Deprecated server registration function.

Additional Inherited Members

Public Types inherited from `L4::Epiface_t0< L4::Factory, L4::Epiface >`

- using `Interface`

Data type of the IPC interface definition.

Public Types inherited from `L4::Epiface`

- using `Server_iface = lpc_svr::Server_iface`

Type for abstract server interface.

- using `Demand = lpc_svr::Server_iface::Demand`

Type for server-side receive buffer demand.

15.474.1 Detailed Description

The IPC interface for creating ports.

The Switch factory provides an IPC interface to create ports. Ports are the only option for a client to communicate with the switch and, thus, with other network devices.

The `Switch_factory` gets constructed when the net switch application gets started. It thereafter gets registered on the switch's server to serve IPC `create` calls.

Definition at line 114 of file `main.cc`.

15.474.2 Member Function Documentation

15.474.2.1 op_create()

```
long Switch_factory::op_create (
    L4::Factory::Rights ,
    L4::Ipc::Cap< void > & res,
    l4_umword_t type,
    L4::Ipc::Varg_list_ref va) [inline]
```

Handle factory protocol.

This function is invoked after an incoming factory::create request and creates a new port or statistics interface if possible.

Definition at line 533 of file [main.cc](#).

References [L4_EINVAL](#).

The documentation for this class was generated from the following file:

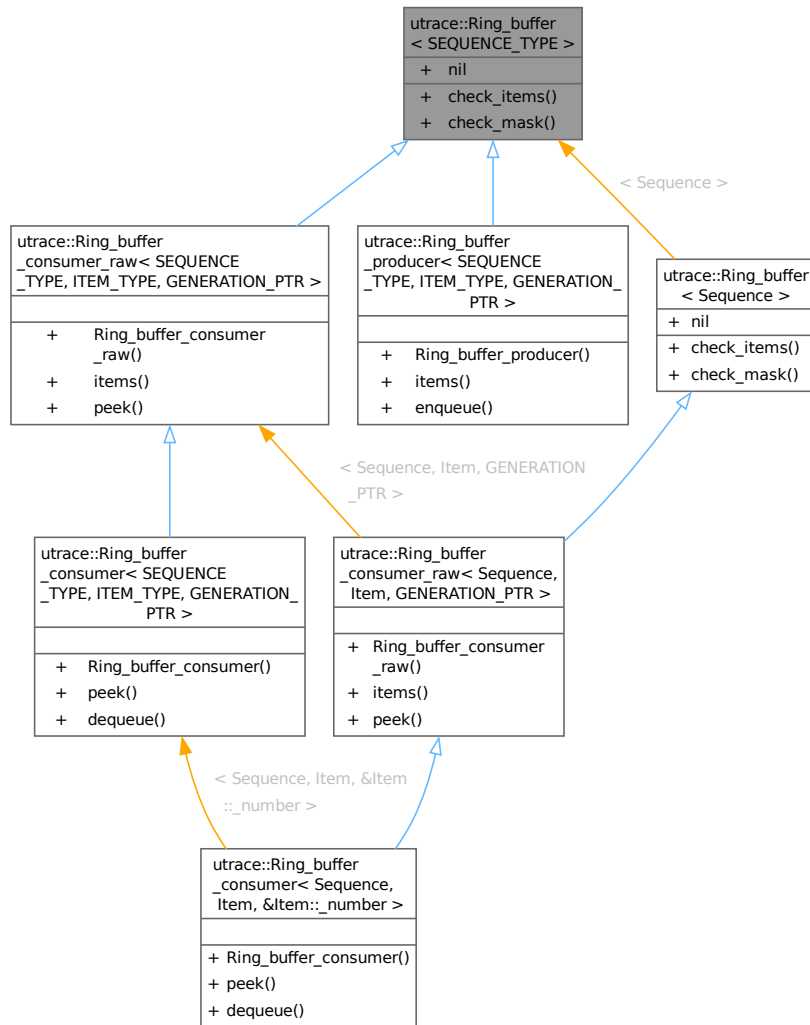
- pkg/virtio-net-switch/server/switch/main.cc

15.475 utrace::Ring_buffer< SEQUENCE_TYPE > Class Template Reference

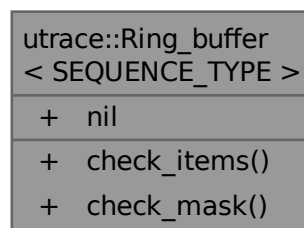
Ring buffer.

```
#include <ring_buffer>
```

Inheritance diagram for utrace::Ring_buffer< SEQUENCE_TYPE >:



Collaboration diagram for utrace::Ring_buffer< SEQUENCE_TYPE >:



Static Public Member Functions

- static void [check_items](#) (size_t const items)
Check that the number of items is a power of two.
- static void [check_mask](#) (size_t const mask)
Check that the item mask is a power of two minus one.

Static Public Attributes

- static constexpr Sequence const **nil** = 0U
Invalid (non-committed) items set their sequence counter to 0.

15.475.1 Detailed Description

```
template<typename SEQUENCE_TYPE>
class utrace::Ring_buffer< SEQUENCE_TYPE >
```

Ring buffer.

Base class of all ring buffer objects. Not directly usable by the user.

Template Parameters

| | |
|----------------------|--|
| <i>SEQUENCE_TYPE</i> | Numerical type for storing sequence and generation counters. Must be suitable for atomic access. |
|----------------------|--|

Definition at line 66 of file [ring_buffer](#).

15.475.2 Member Function Documentation

15.475.2.1 check_items()

```
template<typename SEQUENCE_TYPE>
void utrace::Ring_buffer< SEQUENCE_TYPE >::check_items (
    size_t const items) [inline], [static]
```

Check that the number of items is a power of two.

This ring buffer implementation supports a non-zero power of two number of items.

Parameters

| | |
|--------------|------------------|
| <i>items</i> | Number of items. |
|--------------|------------------|

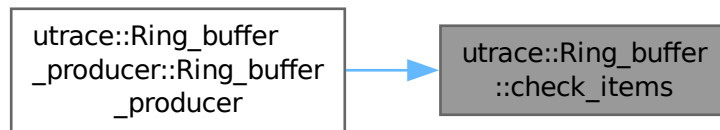
Exceptions

| | |
|------------------------------------|---|
| <code>std::invalid_argument</code> | If the number of items is zero or not a power of two. |
|------------------------------------|---|

Definition at line 86 of file [ring_buffer](#).

Referenced by [utrace::Ring_buffer_producer< SEQUENCE_TYPE, ITEM_TYPE, GENERATION_PTR >::Ring_buffer_producer\(\)](#).

Here is the caller graph for this function:

**15.475.2.2 check_mask()**

```

template<typename SEQUENCE_TYPE>
void utrace::Ring_buffer< SEQUENCE_TYPE >::check_mask (
    size_t const mask) [inline], [static]
  
```

Check that the item mask is a power of two minus one.

This ring buffer implementation uses the bitwise AND for implementing a modulo operation with the number of items. Thus the item mask must be a power of two minus one.

Parameters

| | |
|-------------------|------------|
| <code>mask</code> | Item mask. |
|-------------------|------------|

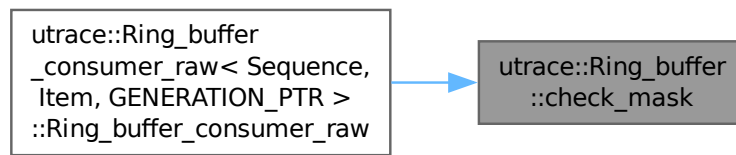
Exceptions

| | |
|-------------------------------|---|
| <code>td::length_error</code> | If the item mask is not a power of two minus one. |
|-------------------------------|---|

Definition at line 106 of file [ring_buffer](#).

Referenced by [utrace::Ring_buffer_consumer_raw< Sequence, Item, GENERATION_PTR >::Ring_buffer_consumer_raw\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

- [l4/utrace/ring_buffer](#)

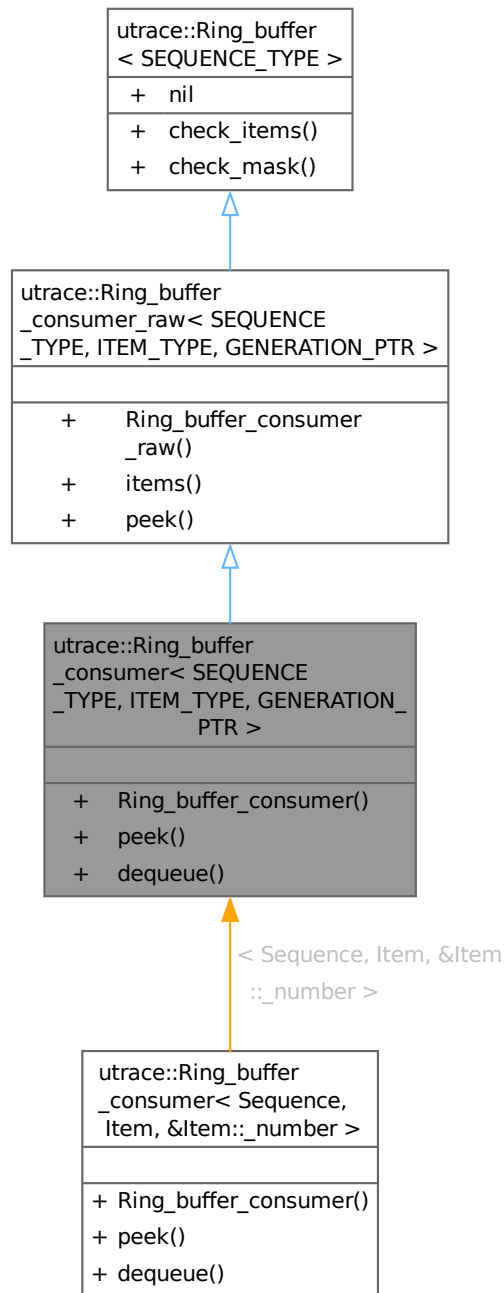
15.476 utrace::Ring_buffer_consumer< SEQUENCE_TYPE, ITEM_TYPE, GENERATION_PTR > Class Template Reference

Blocking ring buffer consumer.

```
#include <ring_buffer>
```

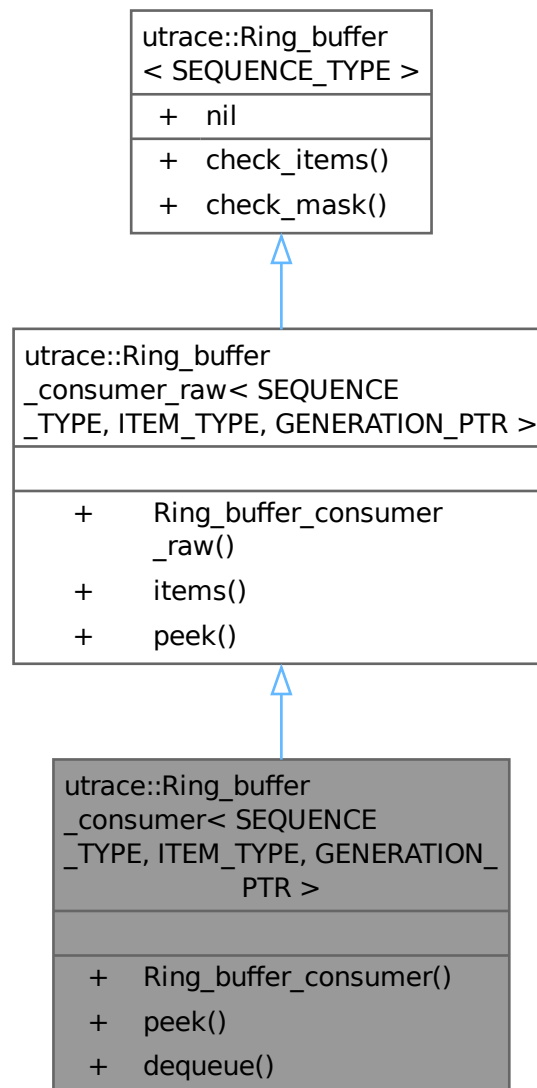
Inheritance diagram for utrace::Ring_buffer_consumer< SEQUENCE_TYPE, ITEM_TYPE, GENERATION_PTR

>:



Collaboration diagram for `utrace::Ring_buffer_consumer< SEQUENCE_TYPE, ITEM_TYPE, GENERATION_PTR`

>:



Public Member Functions

- [Ring_buffer_consumer](#) (Status const &status, Slot const *slots)
Construct ring buffer consumer.
- [State peek](#) (Item &item, [Drop_policy](#) policy, Sequence *drops=nullptr)
Poll and possibly dequeue an item from the ring buffer.
- `size_t` [dequeue](#) (Item [items](#)[], `size_t` capacity, `size_t` burst, [Drop_policy](#) policy, Yield const &yield, Sequence *drops=nullptr)
Dequeue items from the ring buffer.

Public Member Functions inherited from

`utrace::Ring_buffer_consumer_raw< SEQUENCE_TYPE, ITEM_TYPE, GENERATION_PTR >`

- `Ring_buffer_consumer_raw` (Status const &status, Slot const *slots)
Construct ring buffer consumer.
- `size_t items` () const
Get the number of items.
- `State peek` (Item &item, `Drop_policy` policy, Sequence ¤t, Sequence *drops=nullptr) const
Poll and possibly dequeue an item from the ring buffer.

Additional Inherited Members

Public Types inherited from

`utrace::Ring_buffer_consumer_raw< SEQUENCE_TYPE, ITEM_TYPE, GENERATION_PTR >`

- enum class `Drop_policy` { `Conservative` = 0 , `Minimal` }
- enum class `State`

Item drop policy.

Status of the dequeue operation.

Static Public Member Functions inherited from

`utrace::Ring_buffer< SEQUENCE_TYPE >`

- static void `check_items` (size_t const items)
Check that the number of items is a power of two.
- static void `check_mask` (size_t const mask)
Check that the item mask is a power of two minus one.

Static Public Attributes inherited from `utrace::Ring_buffer< SEQUENCE_TYPE >`

- static constexpr Sequence const `nil` = 0U
Invalid (non-committed) items set their sequence counter to 0.

15.476.1 Detailed Description

```
template<typename SEQUENCE_TYPE, typename ITEM_TYPE, auto GENERATION_PTR>
class utrace::Ring_buffer_consumer< SEQUENCE_TYPE, ITEM_TYPE, GENERATION_PTR >
```

Blocking ring buffer consumer.

Both the ring buffer status area and ring buffer slots are accessed in a read-only fashion.

Template Parameters

| | |
|----------------------------|--|
| <code>SEQUENCE_TYPE</code> | Numerical type for storing sequence and generation counters. Must be suitable for atomic access. |
|----------------------------|--|

| | |
|-----------------------|--|
| <i>ITEM_TYPE</i> | Actual ring buffer item type. |
| <i>GENERATION_PTR</i> | Class member pointer to the member that contains the sequence number within the item. The member needs to be suitable for an atomic reference. |

Definition at line 510 of file [ring_buffer](#).

15.476.2 Constructor & Destructor Documentation

15.476.2.1 Ring_buffer_consumer()

```
template<typename SEQUENCE_TYPE, typename ITEM_TYPE, auto GENERATION_PTR>
utrace::Ring_buffer_consumer< SEQUENCE_TYPE, ITEM_TYPE, GENERATION_PTR >::Ring_buffer_consumer
(
    Status const & status,
    Slot const * slots) [inline]
```

Construct ring buffer consumer.

Parameters

| | |
|---------------|--------------------------|
| <i>status</i> | Ring buffer status area. |
| <i>slots</i> | Ring buffer slots. |

Exceptions

| | |
|-------------------------|--|
| <i>td::length_error</i> | If the item mask (in the ring buffer status area) is not a power of two minus one. |
|-------------------------|--|

Definition at line 535 of file [ring_buffer](#).

15.476.3 Member Function Documentation

15.476.3.1 dequeue()

```
template<typename SEQUENCE_TYPE, typename ITEM_TYPE, auto GENERATION_PTR>
size_t utrace::Ring_buffer_consumer< SEQUENCE_TYPE, ITEM_TYPE, GENERATION_PTR >::dequeue (
    Item items[],
    size_t capacity,
    size_t burst,
    Drop_policy policy,
    Yield const & yield,
    Sequence * drops = nullptr) [inline]
```

Dequeue items from the ring buffer.

Dequeue multiple items from the ring buffer. The goal is to dequeue as many items as available from the ring buffer in a single call to avoid overhead. This operation blocks until at least *burst* items have been dequeued (but returns immediately if an item is dropped).

The implementation implicitly spins actively while waiting for items to be dequeued from the ring buffer. The *yield* function can implement passive waiting. Its argument is the number of idle spinning cycles that already happened and if the function returns true, this number is reset.

Parameters

| | | |
|-----|-----------------|--|
| out | <i>items</i> | Array to dequeue the next items from the ring buffer into (as copies). |
| | <i>capacity</i> | Capacity of <i>items</i> (in items). |
| | <i>burst</i> | Minimal number of items that should be collected before returning in case of waiting for items. |
| | <i>policy</i> | Item drop policy. |
| | <i>yield</i> | Yield function to implement passive waiting. Executed in every spinning cycle when no next item has been produced yet. Its argument is the current counter of the idle spinning cycles and that counter is reset when the function returns true. |
| out | <i>drops</i> | Pointer to store the number of items that have been dropped. Can be nullptr. |

Returns

Number of items dequeued and stored into *items*.

Definition at line 595 of file [ring_buffer](#).

15.476.3.2 peek()

```
template<typename SEQUENCE_TYPE, typename ITEM_TYPE, auto GENERATION_PTR>
State utrace::Ring_buffer_consumer< SEQUENCE_TYPE, ITEM_TYPE, GENERATION_PTR >::peek (
    Item & item,
    Drop_policy policy,
    Sequence * drops = nullptr) [inline]
```

Poll and possibly dequeue an item from the ring buffer.

This operation never blocks. However, since the expected sequence counter of the next item is managed internally with mutual exclusion, multiple concurrent calls to this method might affect the latency.

Parameters

| | | |
|-----|---------------|--|
| out | <i>item</i> | Reference to dequeue the next item from the ring buffer into (as a copy). |
| | <i>policy</i> | Item drop policy. |
| out | <i>drops</i> | Pointer to store the number of items that have been dropped. Can be nullptr. |

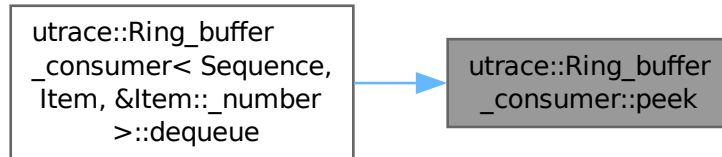
Return values

| | |
|---------------------|---|
| <i>State::Idle</i> | No next item has been produced yet. No item has been dequeued, no item has been dropped. |
| <i>State::Ready</i> | A next item has been dequeued from the ring buffer and stored in <i>item</i> . No item has been dropped. |
| <i>State::Drop</i> | Some items have been dropped (missed). No item has been dequeued. The number of dropped items has been stored to <i>drops</i> (if non-nullptr). |

Definition at line 560 of file [ring_buffer](#).

Referenced by [utrace::Ring_buffer_consumer< Sequence, Item, &Item::_number >::dequeue\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

- [l4/utrace/ring_buffer](#)

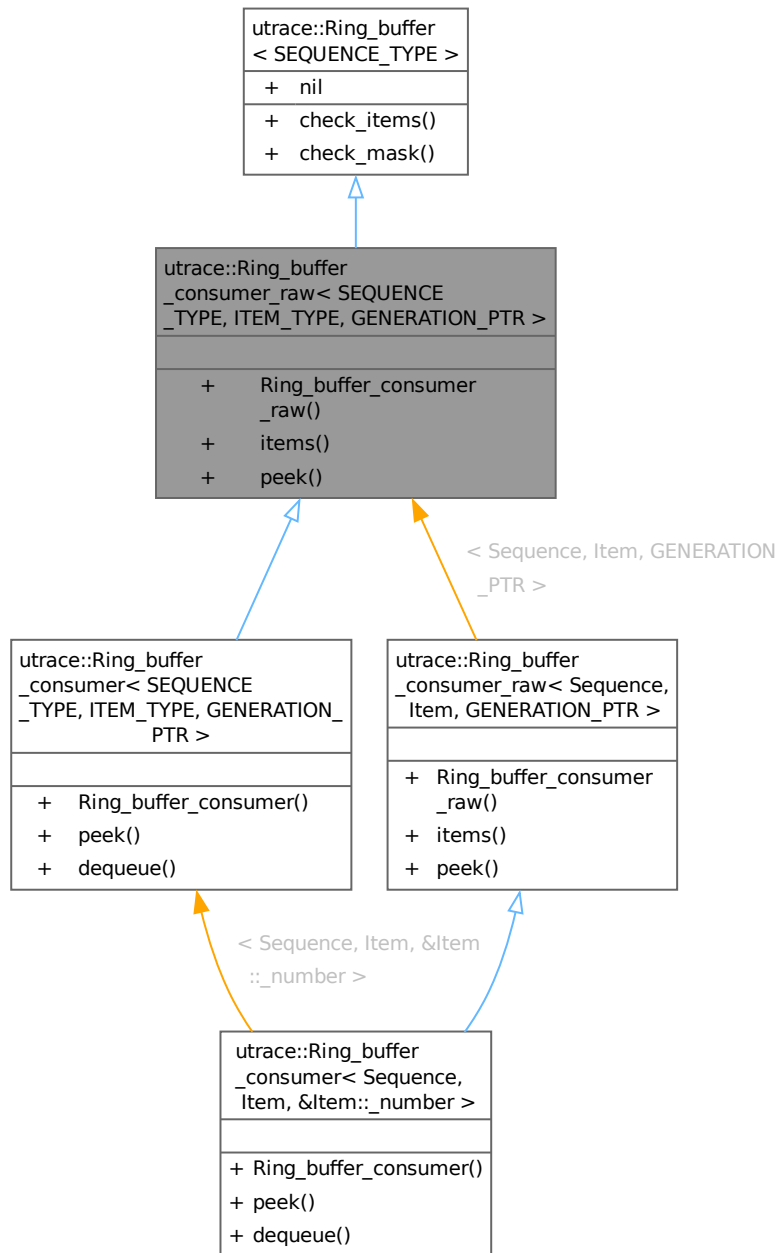
15.477 `utrace::Ring_buffer_consumer_raw< SEQUENCE_TYPE, ITEM_TYPE, GENERATION_PTR >` Class Template Reference

Polling ring buffer consumer.

```
#include <ring_buffer>
```

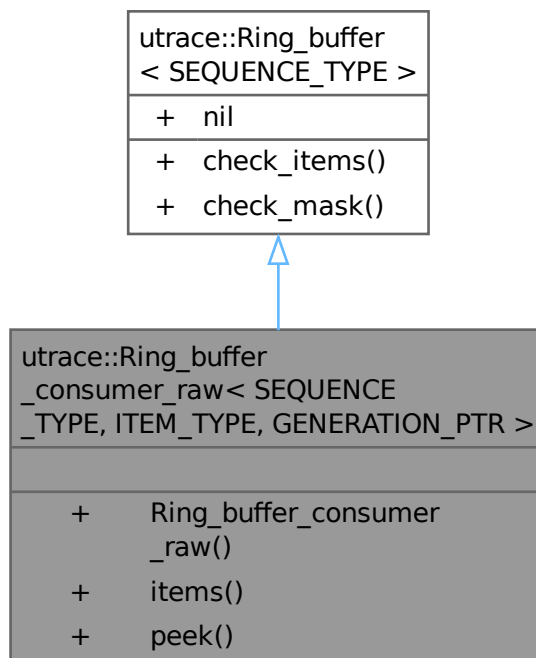
Inheritance diagram for `utrace::Ring_buffer_consumer_raw< SEQUENCE_TYPE, ITEM_TYPE, GENERATION_PTR >`

PTR >:



Collaboration diagram for utrace::Ring_buffer_consumer_raw< SEQUENCE_TYPE, ITEM_TYPE, GENERATION_PTR >:

_PTR >:



Public Types

- enum class [Drop_policy](#) { [Conservative](#) = 0 , [Minimal](#) }
Item drop policy.
- enum class [State](#)
Status of the dequeue operation.

Public Member Functions

- [Ring_buffer_consumer_raw](#) (Status const &status, Slot const *slots)
Construct ring buffer consumer.
- [size_t items](#) () const
Get the number of items.
- [State peek](#) (Item &item, [Drop_policy](#) policy, Sequence ¤t, Sequence *drops=nullptr) const
Poll and possibly dequeue an item from the ring buffer.

Additional Inherited Members

Static Public Member Functions inherited from [utrace::Ring_buffer](#)< [SEQUENCE_TYPE](#) >

- static void [check_items](#) (size_t const items)
Check that the number of items is a power of two.
- static void [check_mask](#) (size_t const mask)
Check that the item mask is a power of two minus one.

Static Public Attributes inherited from [utrace::Ring_buffer< SEQUENCE_TYPE >](#)

- static constexpr Sequence const **nil** = 0U
Invalid (non-committed) items set their sequence counter to 0.

15.477.1 Detailed Description

```
template<typename SEQUENCE_TYPE, typename ITEM_TYPE, auto GENERATION_PTR>
class utrace::Ring_buffer_consumer_raw< SEQUENCE_TYPE, ITEM_TYPE, GENERATION_PTR >
```

Polling ring buffer consumer.

Both the ring buffer status area and ring buffer slots are accessed in a read-only fashion.

Template Parameters

| | |
|-----------------------|--|
| <i>SEQUENCE_TYPE</i> | Numerical type for storing sequence and generation counters. Must be suitable for atomic access. |
| <i>ITEM_TYPE</i> | Actual ring buffer item type. |
| <i>GENERATION_PTR</i> | Class member pointer to the member that contains the sequence number within the item. The member needs to be suitable for an atomic reference. |

Definition at line [334](#) of file [ring_buffer](#).

15.477.2 Member Enumeration Documentation

15.477.2.1 Drop_policy

```
template<typename SEQUENCE_TYPE, typename ITEM_TYPE, auto GENERATION_PTR>
enum class utrace::Ring_buffer_consumer_raw::Drop_policy [strong]
```

Item drop policy.

This affects the behavior of the consumer when an item drop is detected (i.e. the consumer loses access to an item or multiple items in the ring buffer that have been overwritten by a producer in the meantime).

Enumerator

| | |
|--------------|--|
| Conservative | Conservative item drop policy. Continue with the newest item in the ring buffer (i.e. proactively drop all the items except the newest one). This policy is suitable when the production of items is always faster than the consumption and it gives the consumer a chance to dequeue at least some items from the ring buffer before it falls behind again. |
| Minimal | Minimal item drop policy. Continue with the oldest item in the ring buffer (i.e. do not proactively drop more items than those that have been lost already). This policy is suitable when the production of items is mostly slower than the consumption and the consumer is able to catch up after an occasional drop of items. |

Definition at line [353](#) of file [ring_buffer](#).

15.477.3 Constructor & Destructor Documentation

15.477.3.1 Ring_buffer_consumer_raw()

```
template<typename SEQUENCE_TYPE, typename ITEM_TYPE, auto GENERATION_PTR>
utrace::Ring_buffer_consumer_raw< SEQUENCE_TYPE, ITEM_TYPE, GENERATION_PTR >::Ring_buffer_consumer_raw (
    Status const & status,
    Slot const * slots) [inline]
```

Construct ring buffer consumer.

Parameters

| | |
|---------------|--------------------------|
| <i>status</i> | Ring buffer status area. |
| <i>slots</i> | Ring buffer slots. |

Exceptions

| | |
|-------------------------|--|
| <i>td::length_error</i> | If the item mask (in the ring buffer status area) is not a power of two minus one. |
|-------------------------|--|

Definition at line 394 of file [ring_buffer](#).

15.477.4 Member Function Documentation

15.477.4.1 peek()

```
template<typename SEQUENCE_TYPE, typename ITEM_TYPE, auto GENERATION_PTR>
State utrace::Ring_buffer_consumer_raw< SEQUENCE_TYPE, ITEM_TYPE, GENERATION_PTR >::peek (
    Item & item,
    Drop_policy policy,
    Sequence & current,
    Sequence * drops = nullptr) const [inline]
```

Poll and possibly dequeue an item from the ring buffer.

This operation never blocks.

Parameters

| | | |
|---------|----------------|--|
| out | <i>item</i> | Reference to dequeue the next item from the ring buffer into (as a copy). |
| | <i>policy</i> | Item drop policy. |
| in, out | <i>current</i> | Sequence counter of the next item to be dequeued from the ring buffer. Initialize to Ring_buffer::nil to get the sequence counter of the oldest or the newest item currently available in the ring buffer (depending on the item drop policy). |
| out | <i>drops</i> | Pointer to store the number of items that have been dropped. Can be nullptr. |

Return values

| | |
|---------------------|--|
| <i>State::Idle</i> | The next item with the expected sequence counter has not been produced yet. No item has been dequeued, no item has been dropped, the sequence counter is unchanged. |
| <i>State::Ready</i> | The next item (with the expected sequence counter) has been dequeued from the ring buffer and stored in <i>item</i> . The sequence counter <i>current</i> has been advanced to the next expected item. No item has been dropped. |
| <i>State::Drop</i> | Some items have been dropped (missed). No item has been dequeued. The number of dropped items has been stored to <i>drops</i> (if non-nullptr). The sequence counter <i>current</i> has been advanced to the oldest (if the drop policy is Minimal) or to the newest (if the drop policy is Conservative), respectively. |

Definition at line 435 of file [ring_buffer](#).

The documentation for this class was generated from the following file:

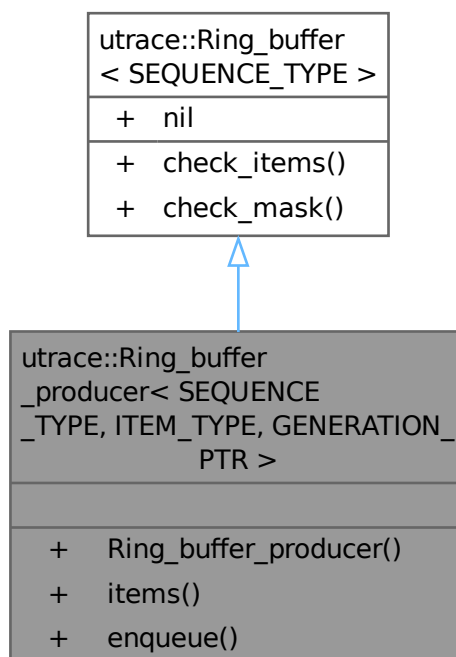
- [I4/utrace/ring_buffer](#)

15.478 `utrace::Ring_buffer_producer< SEQUENCE_TYPE, ITEM_TYPE, GENERATION_PTR >` Class Template Reference

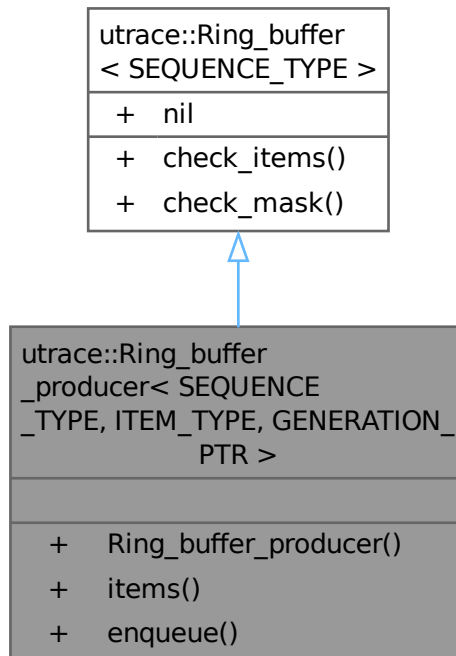
Ring buffer producer.

```
#include <ring_buffer>
```

Inheritance diagram for `utrace::Ring_buffer_producer< SEQUENCE_TYPE, ITEM_TYPE, GENERATION_PTR >`:



Collaboration diagram for `utrace::Ring_buffer_producer< SEQUENCE_TYPE, ITEM_TYPE, GENERATION_PTR >`:



Public Member Functions

- [Ring_buffer_producer](#) (Status &status, Slot &slots, unsigned const version, size_t const [items](#))
Construct ring buffer producer.
- size_t [items](#) () const
Get the number of items.
- void [enqueue](#) (Item const &item) const
Enqueue an item in the ring buffer.

Additional Inherited Members

Static Public Member Functions inherited from [utrace::Ring_buffer< SEQUENCE_TYPE >](#)

- static void [check_items](#) (size_t const items)
Check that the number of items is a power of two.
- static void [check_mask](#) (size_t const mask)
Check that the item mask is a power of two minus one.

Static Public Attributes inherited from [utrace::Ring_buffer< SEQUENCE_TYPE >](#)

- static constexpr Sequence const **nil** = 0U
Invalid (non-committed) items set their sequence counter to 0.

15.478.1 Detailed Description

```
template<typename SEQUENCE_TYPE, typename ITEM_TYPE, auto GENERATION_PTR>
class utrace::Ring_buffer_producer< SEQUENCE_TYPE, ITEM_TYPE, GENERATION_PTR >
```

Ring buffer producer.

Template Parameters

| | |
|-----------------------|--|
| <i>SEQUENCE_TYPE</i> | Numerical type for storing sequence and generation counters. Must be suitable for atomic access. |
| <i>ITEM_TYPE</i> | Actual ring buffer item type. |
| <i>GENERATION_PTR</i> | Class member pointer to the member that contains the sequence number within the item. The member needs to be suitable for an atomic reference. |

Definition at line [248](#) of file [ring_buffer](#).

15.478.2 Constructor & Destructor Documentation**15.478.2.1 Ring_buffer_producer()**

```
template<typename SEQUENCE_TYPE, typename ITEM_TYPE, auto GENERATION_PTR>
utrace::Ring_buffer_producer< SEQUENCE_TYPE, ITEM_TYPE, GENERATION_PTR >::Ring_buffer_producer
(
    Status & status,
    Slot & slots,
    unsigned const version,
    size_t const items) [inline]
```

Construct ring buffer producer.

This initializes the ring buffer status area and slots.

Parameters

| | |
|----------------|---|
| <i>status</i> | Ring buffer status area. |
| <i>slots</i> | Ring buffer slots. |
| <i>version</i> | Version of the ring buffer items. The semantics of the version number is defined by the user. |
| <i>items</i> | Number of items in the ring buffer slots. Must be a non-zero power of two. |

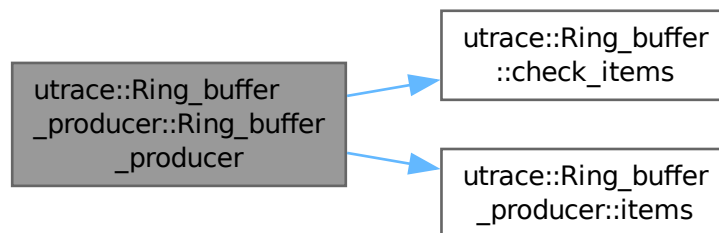
Exceptions

| | |
|------------------------------------|---|
| <code>std::invalid_argument</code> | If the number of items is zero or not a power of two. |
|------------------------------------|---|

Definition at line 274 of file [ring_buffer](#).

References [utrace::Ring_buffer< SEQUENCE_TYPE >::check_items\(\)](#), [items\(\)](#), and [utrace::Ring_buffer< SEQUENCE_TYPE >::nil](#).

Here is the call graph for this function:



15.478.3 Member Function Documentation

15.478.3.1 enqueue()

```
template<typename SEQUENCE_TYPE, typename ITEM_TYPE, auto GENERATION_PTR>
void utrace::Ring_buffer_producer< SEQUENCE_TYPE, ITEM_TYPE, GENERATION_PTR >::enqueue (
    Item const & item) const [inline]
```

Enqueue an item in the ring buffer.

This operation never blocks and implements the head-drop policy (i.e. the oldest item is overwritten).

Parameters

| | |
|-------------|------------------------------|
| <i>item</i> | Item to enqueue (as a copy). |
|-------------|------------------------------|

Definition at line 300 of file [ring_buffer](#).

References [utrace::Ring_buffer< SEQUENCE_TYPE >::nil](#).

The documentation for this class was generated from the following file:

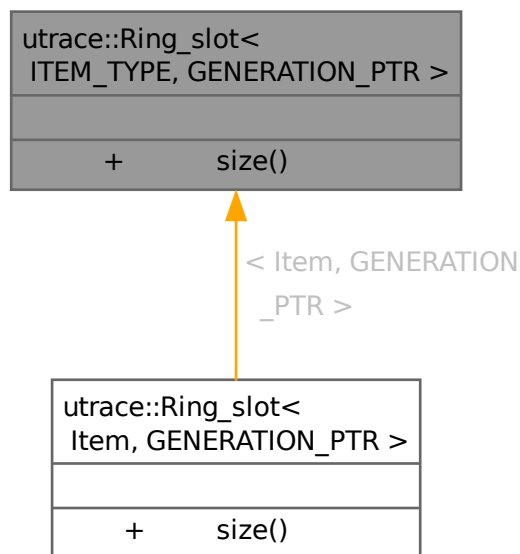
- [I4/utrace/ring_buffer](#)

15.479 utrace::Ring_slot< ITEM_TYPE, GENERATION_PTR > Class Template Reference

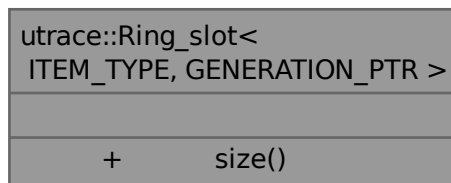
Ring buffer slot.

```
#include <ring_buffer>
```

Inheritance diagram for utrace::Ring_slot< ITEM_TYPE, GENERATION_PTR >:



Collaboration diagram for utrace::Ring_slot< ITEM_TYPE, GENERATION_PTR >:



Static Public Member Functions

- static `size_t` [size](#) (`size_t` const items)
Get the size (in bytes) of the given count of items.

15.479.1 Detailed Description

```
template<typename ITEM_TYPE, auto GENERATION_PTR>
class utrace::Ring_slot< ITEM_TYPE, GENERATION_PTR >
```

Ring buffer slot.

The ring buffer slot encapsulates one ring buffer item and makes sure that the item is aligned to the nearest larger power-of-two of its size.

Another purpose of this class is to abstract away the access to the sequence number of the item.

Template Parameters

| | |
|-----------------------|--|
| <i>ITEM_TYPE</i> | Actual ring buffer item type. |
| <i>GENERATION_PTR</i> | Class member pointer to the member that contains the sequence number within the item. The member needs to be suitable for an atomic reference. |

Definition at line 207 of file [ring_buffer](#).

15.479.2 Member Function Documentation

15.479.2.1 size()

```
template<typename ITEM_TYPE, auto GENERATION_PTR>
size_t utrace::Ring_slot< ITEM_TYPE, GENERATION_PTR >::size (
    size_t const items) [inline], [static]
```

Get the size (in bytes) of the given count of items.

Parameters

| | |
|--------------|-----------------|
| <i>items</i> | Count of items. |
|--------------|-----------------|

Returns

Size (in bytes) of the given count of items.

Definition at line 224 of file [ring_buffer](#).

The documentation for this class was generated from the following file:

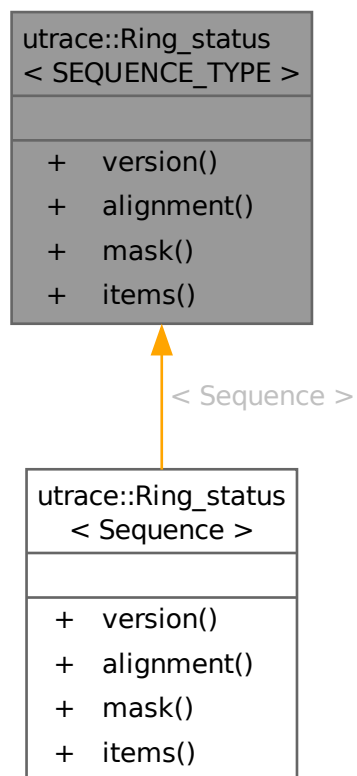
- I4/utrace/[ring_buffer](#)

15.480 utrace::Ring_status< SEQUENCE_TYPE > Class Template Reference

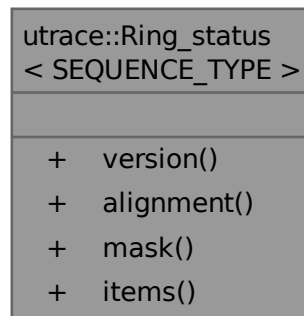
Ring buffer status area.

```
#include <ring_buffer>
```

Inheritance diagram for utrace::Ring_status< SEQUENCE_TYPE >:



Collaboration diagram for `utrace::Ring_status< SEQUENCE_TYPE >`:



Public Member Functions

- unsigned `version` () const
Get the version of the ring buffer items.
- `size_t alignment` () const
Get the stable alignment of the ring buffer status members.
- `size_t mask` () const
Get the item mask.
- `size_t items` () const
Get the number of items.

15.480.1 Detailed Description

```
template<typename SEQUENCE_TYPE>
class utrace::Ring_status< SEQUENCE_TYPE >
```

Ring buffer status area.

The ring buffer status area contains run-time configuration information of the ring buffer, most notably the current atomic generation counter (i.e. the last sequence number allocated to an item).

This structure is only written by the producers, it is read-only for the consumers.

Template Parameters

| | |
|----------------------------|--|
| <code>SEQUENCE_TYPE</code> | Numerical type for storing sequence and generation counters. Must be suitable for atomic access. |
|----------------------------|--|

Definition at line 128 of file `ring_buffer`.

15.480.2 Member Function Documentation

15.480.2.1 alignment()

```
template<typename SEQUENCE_TYPE>
size_t utrace::Ring_status< SEQUENCE_TYPE >::alignment () const [inline]
```

Get the stable alignment of the ring buffer status members.

This is relevant for the `_mask` and `_tail` members.

Returns

Stable alignment of the ring buffer status members.

Definition at line 155 of file [ring_buffer](#).

15.480.2.2 version()

```
template<typename SEQUENCE_TYPE>
unsigned utrace::Ring_status< SEQUENCE_TYPE >::version () const [inline]
```

Get the version of the ring buffer items.

The semantics of the version number is defined by the user.

Returns

Version of the ring buffer items.

Definition at line 145 of file [ring_buffer](#).

The documentation for this class was generated from the following file:

- I4/utrace/[ring_buffer](#)

15.481 utrace::Tracebuffer Class Reference

[Tracebuffer](#) abstraction.

```
#include <utrace>
```

Collaboration diagram for utrace::Tracebuffer:

| utrace::Tracebuffer |
|---|
| <ul style="list-style-type: none"> + items() + dequeue() + validate() + version() + endianness() + index() + indexes() + instance() |

Data Structures

- struct [Index_desc](#)
Mapping of the dynamic tracebuffer log indexes to names.

Public Member Functions

- size_t **items** () const
Get the capacity (in items) of the tracebuffer ring buffer.
- size_t [dequeue](#) (Item *[items](#), size_t capacity, size_t burst, [Drop_policy](#) policy, Sequence *drops=nullptr)
Dequeue items from the tracebuffer.

Static Public Member Functions

- static void [validate](#) ()
Check that the base debugger (JDB) capability is valid and accessible.
- static unsigned **version** ()
Get the tracebuffer format version supported by this class.
- static std::endian **endianness** ()
Get the endianness supported by this class.
- static std::optional< [Index_desc](#) > **index** (unsigned idx)
Map a dynamic tracebuffer log index to names.
- static std::vector< [Index_desc](#) > **indexes** ()
Get all mappings of dynamic tracebuffer log indexes to names.
- static Tracebuffer & **instance** ()
Get the singleton instance of this class.

15.481.1 Detailed Description

[Tracebuffer](#) abstraction.

Since there is only a single global tracebuffer available in the current implementation, this is a singleton class.

Definition at line 44 of file [utrace](#).

15.481.2 Member Function Documentation

15.481.2.1 dequeue()

```
size_t utrace::Tracebuffer::dequeue (
    Item * items,
    size_t capacity,
    size_t burst,
    Drop_policy policy,
    Sequence * drops = nullptr)
```

Dequeue items from the tracebuffer.

Dequeue multiple items from the tracebuffer. The goal is to dequeue as many items as available from the tracebuffer in a single call to avoid overhead. This operation blocks until at least *burst* items have been dequeued (but returns immediately if an item is dropped).

Parameters

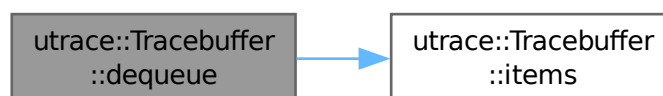
| | | |
|-----|-----------------|---|
| out | <i>items</i> | Array to dequeue the next items from the tracebuffer into (as copies). |
| | <i>capacity</i> | Capacity of <i>items</i> (in items). |
| | <i>burst</i> | Minimal number of items that should be collected before returning in case of waiting for items. |
| | <i>policy</i> | Item drop policy. |
| out | <i>drops</i> | Pointer to store the number of items that have been dropped. Can be nullptr. |

Returns

Number of items dequeued and stored into *items*.

References [items\(\)](#).

Here is the call graph for this function:



15.481.2.2 index()

```
std::optional< Index_desc > utrace::Tracebuffer::index (
    unsigned idx) [static]
```

Map a dynamic tracebuffer log index to names.

Parameters

| | |
|------------|---------------------------------------|
| <i>idx</i> | Dynamic tracebuffer log index to map. |
|------------|---------------------------------------|

Returns

Mapping of the dynamic tracebuffer index to names.

Return values

| | |
|---------------------|---|
| <i>std::nullopt</i> | The dynamic tracebuffer log index is not defined. |
|---------------------|---|

15.481.2.3 indexes()

```
std::vector< Index_desc > utrace::Tracebuffer::indexes () [static]
```

Get all mappings of dynamic tracebuffer log indexes to names.

Returns

Generator (if available) or a vector of all defined mappings of dynamic tracebuffer log indexes to names.

15.481.2.4 validate()

```
void utrace::Tracebuffer::validate () [static]
```

Check that the base debugger (JDB) capability is valid and accessible.

The base debugger (JDB) capability is used to access the single global tracebuffer in the current implementation.

Exceptions

| | |
|------------------------------|---|
| <i>std::invalid_argument</i> | If the base debugger (JDB) capability is invalid or inaccessible. |
|------------------------------|---|

The documentation for this class was generated from the following file:

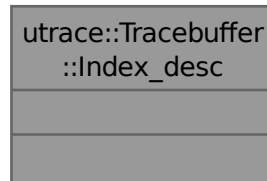
- I4/utrace/[utrace](#)

15.482 utrace::Tracebuffer::Index_desc Struct Reference

Mapping of the dynamic tracebuffer log indexes to names.

```
#include <utrace>
```

Collaboration diagram for utrace::Tracebuffer::Index_desc:



15.482.1 Detailed Description

Mapping of the dynamic tracebuffer log indexes to names.

Definition at line 58 of file [utrace](#).

The documentation for this struct was generated from the following file:

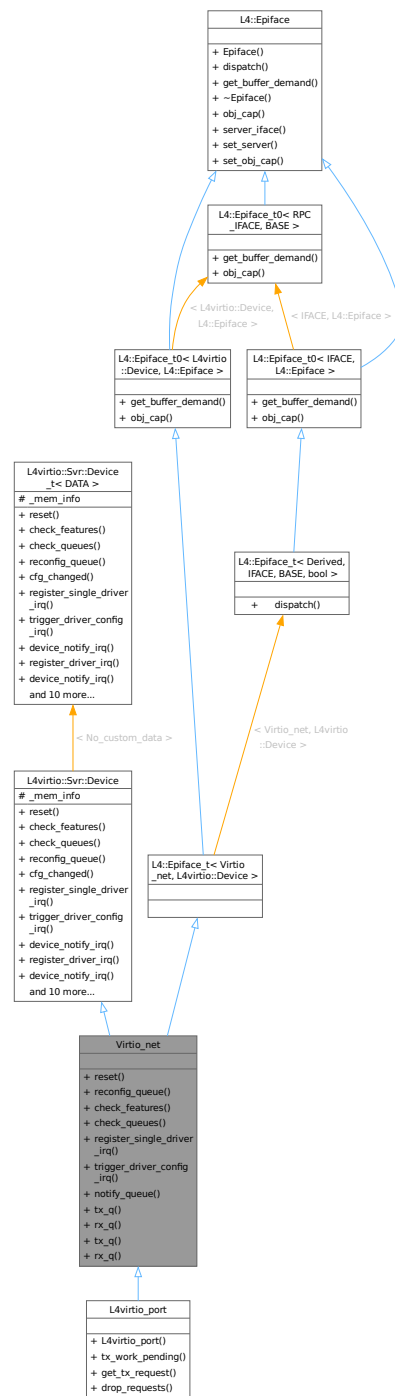
- I4/utrace/[utrace](#)

15.483 Virtio_net Class Reference

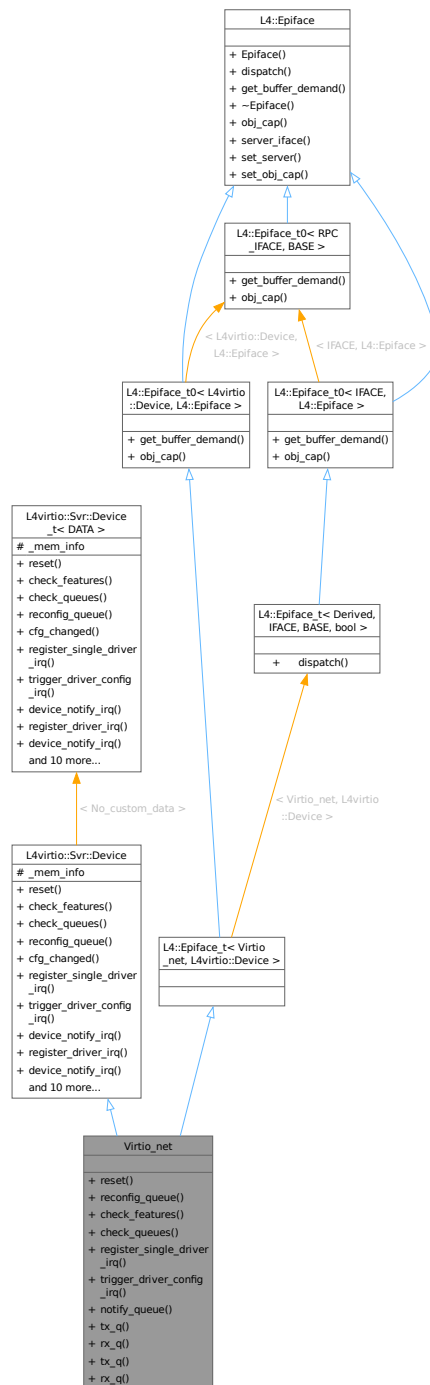
The Base class of a Port.

```
#include <virtio_net.h>
```

Inheritance diagram for Virtio_net:



Collaboration diagram for Virtio_net:



Public Member Functions

- void **reset** () override
reset callback, called for doing a device reset
- int **reconfig_queue** (unsigned index) override
callback for client queue-config request
- bool **check_features** () override

- callback for checking the subset of accepted features*
- bool **check_queues** () override
Check whether both virtqueues are ready.
- void **register_single_driver_irq** () override
Save the `_kick_guest_irq` that the client sent via `device_notification_irq()`.
- void **trigger_driver_config_irq** () override
callback for triggering configuration change notification IRQ
- void **notify_queue** (L4virtio::Svr::Virtqueue *queue)
Trigger the `_kick_guest_irq` IRQ.
- Virtqueue * **tx_q** ()
Getter for the transmission queue.
- Virtqueue * **rx_q** ()
Getter for the receive queue.
- Virtqueue const * **tx_q** () const
Getter for the transmission queue.
- Virtqueue const * **rx_q** () const
Getter for the receive queue.

Public Member Functions inherited from L4virtio::Svr::Device_t< No_custom_data >

- virtual void **cfg_changed** (unsigned)
callback for client device configuration changes
- virtual L4::Cap< L4::Irq > **device_notify_irq** () const
callback to gather the device notification IRQ (old-style)
- virtual void **register_driver_irq** (unsigned idx)
Callback for registering an notification IRQ (multi IRQ).
- virtual L4::Cap< L4::Irq > **device_notify_irq** (unsigned idx)
Callback to gather the device notification IRQ (multi IRQ).
- virtual unsigned **num_events_supported** () const
Return the highest notification index supported.
- **Device_t** (Dev_config *dev_config)
Make a device for the given config.
- Mem_list const * **mem_info** () const
Get the memory region list used for this device.
- void **reset_queue_config** (unsigned idx, unsigned num_max, bool inc_generation=false, unsigned device_notify_index=0)
Trigger reset for the configuration space for queue idx.
- void **init_mem_info** (unsigned num)
Initialize the memory region list to the given maximum.
- void **device_error** ()
Transition device into `DEVICE_NEEDS_RESET` state.
- bool **setup_queue** (Virtqueue *q, unsigned qn, unsigned num_max)
Enable/disable the specified queue.
- bool **handle_mem_cmd_write** ()
Check for a value in the `cmd` register and handle a write.
- void **enable_trusted_ds_validation** ()
Enable trusted dataspace validation.
- void **add_trusted_dataspaces** (std::shared_ptr< Ds_vector const > ds)
Provide a list of trusted dataspaces that can be used for validation.

Public Member Functions inherited from [L4::Epiface_t0< L4virtio::Device, L4::Epiface >](#)

- [Type_info::Demand](#) **get_buffer_demand** () const
Get the server-side buffer demand based in IFACE.
- [Cap< L4virtio::Device >](#) **obj_cap** () const
Get the (typed) capability to this object.

Public Member Functions inherited from [L4::Epiface](#)

- **Epiface** ()
Make a server object.
- virtual **~Epiface** ()=0
Destroy the object.
- Stored_cap [obj_cap](#) () const
Get the capability to the kernel object belonging to this object.
- [Server_iface](#) * **server_iface** () const
Get pointer to server interface at which the object is currently registered.
- int **set_server** ([Server_iface](#) *srv, [Cap< void >](#) cap, bool managed=false)
Set server registration info for the object.
- void **set_obj_cap** ([Cap< void >](#) const &cap)
Deprecated server registration function.

Additional Inherited Members

Public Types inherited from [L4::Epiface_t0< L4virtio::Device, L4::Epiface >](#)

- using **Interface**
Data type of the IPC interface definition.

Public Types inherited from [L4::Epiface](#)

- using **Server_iface** = [lpc_svr::Server_iface](#)
Type for abstract server interface.
- using **Demand** = [lpc_svr::Server_iface::Demand](#)
Type for server-side receive buffer demand.

Protected Attributes inherited from [L4virtio::Svr::Device_t< No_custom_data >](#)

- **Mem_list_mem_info**
Memory region list.

15.483.1 Detailed Description

The Base class of a Port.

This class provides the Virtio network protocol specific implementation aspects of a port.

`Virtio_net` comprises the virtqueues for both, the incoming and the outgoing network requests:

- The transmission queue, containing requests to be transmitted to other ports. The transmission queue is filled by the client, this port relates to.
- The receive queue, containing requests that have been transmitted from other ports. The receive queue is filled by the switch.

Definition at line 71 of file `virtio_net.h`.

15.483.2 Member Function Documentation

15.483.2.1 `notify_queue()`

```
void Virtio_net::notify_queue (
    L4virtio::Svr::Virtqueue * queue) [inline]
```

Trigger the `_kick_guest_irq` IRQ.

This function gets called on the receiving port, when a request was successfully transmitted by the switch.

Definition at line 269 of file `virtio_net.h`.

References `L4VIRTIO_IRQ_STATUS_VRING`.

The documentation for this class was generated from the following file:

- `pkg/virtio-net-switch/server/switch/virtio_net.h`

15.484 `Virtio_net_request` Class Reference

Abstraction for a network request.

```
#include <request_l4virtio.h>
```

Collaboration diagram for `Virtio_net_request`:



Public Member Functions

- [Mac_addr dst_mac](#) () const
Get the Mac address of the destination port.
- [Mac_addr src_mac](#) () const
Get the Mac address of the source port.

Static Public Member Functions

- static void [drop_requests](#) ([Virtio_net](#) *dev, [L4virtio::Svr::Virtqueue](#) *queue)
Drop all requests of a specific queue.
- static std::optional< [Virtio_net_request](#) > [get_request](#) ([Virtio_net](#) *dev, [L4virtio::Svr::Virtqueue](#) *queue)
Construct a request from the next entry of a provided queue.

15.484.1 Detailed Description

Abstraction for a network request.

A [Virtio_net_request](#) is constructed by the source port, using the static function [get_request](#) () as part of [Port_iface::get_tx_request](#) ().

On destruction, [finish](#) () will be called, which, will trigger the client IRQ of the source client.

Definition at line 35 of file [request_l4virtio.h](#).

15.484.2 Member Function Documentation

15.484.2.1 drop_requests()

```
void Virtio_net_request::drop_requests (
    Virtio_net * dev,
    L4virtio::Svr::Virtqueue * queue) [inline], [static]
```

Drop all requests of a specific queue.

This function is used for example to drop all requests in the transmission queue of a monitor port, since monitor ports are not allowed to transmit data.

Parameters

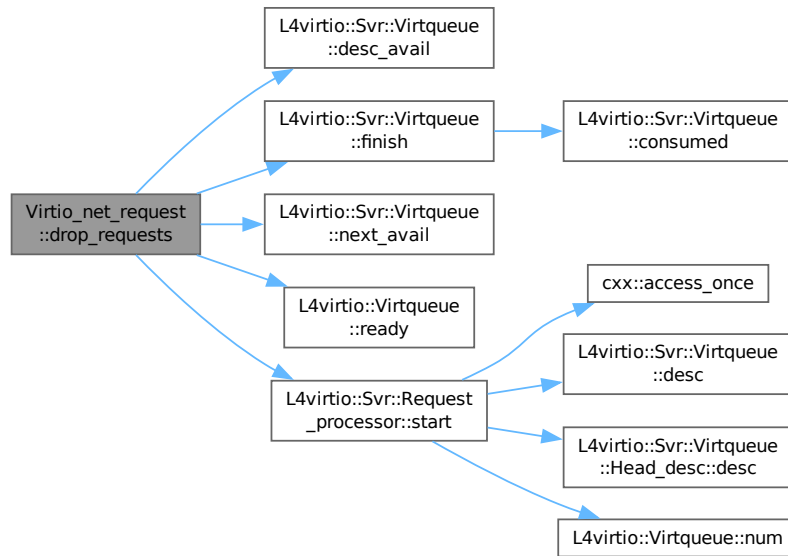
| | |
|--------------|------------------------------------|
| <i>dev</i> | Port of the provided virtqueue. |
| <i>queue</i> | Virtqueue to drop all requests of. |

Definition at line 172 of file [request_l4virtio.h](#).

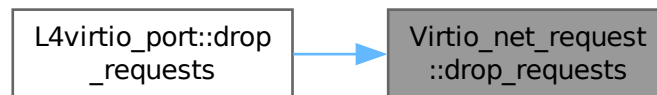
References [L4virtio::Svr::Virtqueue::desc_avail](#)(), [L4virtio::Svr::Virtqueue::finish](#)(), [L4_UNLIKELY](#), [L4virtio::Svr::Virtqueue::next_avail](#)(), [L4virtio::Virtqueue::ready](#)(), and [L4virtio::Svr::Request_processor::start](#)().

Referenced by [L4virtio_port::drop_requests](#)().

Here is the call graph for this function:



Here is the caller graph for this function:



15.484.2.2 get_request()

```

std::optional< Virtio_net_request > Virtio_net_request::get_request (
    Virtio_net * dev,
    L4virtio::Svr::Virtqueue * queue) [inline], [static]
  
```

Construct a request from the next entry of a provided queue.

Parameters

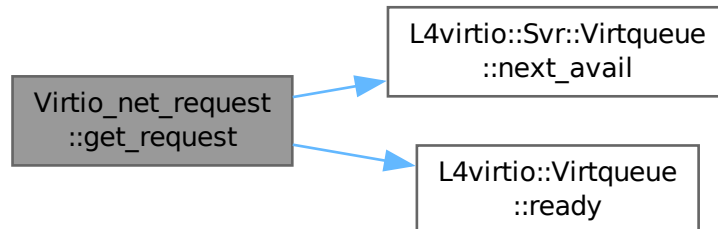
| | |
|--------------|---------------------------------------|
| <i>dev</i> | Port of the provided virtqueue. |
| <i>queue</i> | Virtqueue to extract next entry from. |

Definition at line 199 of file [request_l4virtio.h](#).

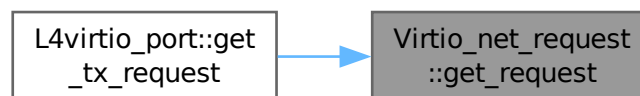
References [L4_UNLIKELY](#), [L4virtio::Svr::Virtqueue::next_avail\(\)](#), and [L4virtio::Virtqueue::ready\(\)](#).

Referenced by [L4virtio_port::get_tx_request\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

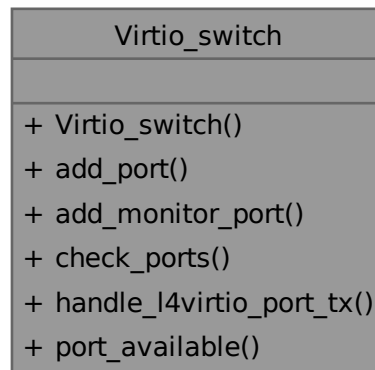
- `pkg/virtio-net-switch/server/switch/request_l4virtio.h`

15.485 Virtio_switch Class Reference

The Virtio switch contains all ports and processes network requests.

```
#include <switch.h>
```

Collaboration diagram for Virtio_switch:



Public Member Functions

- [Virtio_switch](#) (unsigned max_ports)
Create a switch with n ports.
- bool [add_port](#) (Port_iface *port)
Add a port to the switch.
- bool [add_monitor_port](#) (Port_iface *port)
Add a monitor port to the switch.
- void [check_ports](#) ()
Check validity of ports.
- bool [handle_l4virtio_port_tx](#) (L4virtio_port *port)
Handle TX queue of the given port.
- bool [port_available](#) (bool monitor)
Is there still a free port on this switch available?

15.485.1 Detailed Description

The Virtio switch contains all ports and processes network requests.

A Port on its own is not capable to process an incoming network request because it has no knowledge about other ports. The processing of an incoming request therefore gets delegated to the switch.

The [Virtio_switch](#) is constructed at the start of the Virtio Net Switch application. The factory saves a reference to it to pass it to the `Kick_irq` on port creation.

Definition at line 35 of file [switch.h](#).

15.485.2 Constructor & Destructor Documentation

15.485.2.1 Virtio_switch()

```
Virtio_switch::Virtio_switch (  
    unsigned max_ports) [explicit]
```

Create a switch with n ports.

Parameters

| | |
|------------------|----------------------------------|
| <i>max_ports</i> | maximal number of provided ports |
|------------------|----------------------------------|

Definition at line 12 of file [switch.cc](#).

15.485.3 Member Function Documentation

15.485.3.1 add_monitor_port()

```
bool Virtio_switch::add_monitor_port (  
    Port_iface * port)
```

Add a monitor port to the switch.

Parameters

| | |
|-------------|--|
| <i>port</i> | A pointer to an already constructed Port_iface object. |
|-------------|--|

Return values

| | |
|--------------|--------------------------------------|
| <i>true</i> | Port was added successfully. |
| <i>false</i> | Switch was not able to add the port. |

Definition at line 41 of file [switch.cc](#).

15.485.3.2 add_port()

```
bool Virtio_switch::add_port (  
    Port_iface * port)
```

Add a port to the switch.

Parameters

| | |
|-------------|--|
| <i>port</i> | A pointer to an already constructed Port_iface object. |
|-------------|--|

Return values

| | |
|--------------|--------------------------------------|
| <i>true</i> | Port was added successfully. |
| <i>false</i> | Switch was not able to add the port. |

Definition at line 17 of file [switch.cc](#).

References [Mac_addr::is_unknown\(\)](#).

Here is the call graph for this function:

**15.485.3.3 check_ports()**

```
void Virtio_switch::check_ports ()
```

Check validity of ports.

Check whether all ports are still used and remove any unused (unreferenced) ports. Shall be invoked after an incoming cap deletion irq to remove ports without clients.

Definition at line 56 of file [switch.cc](#).

15.485.3.4 handle_l4virtio_port_tx()

```
bool Virtio_switch::handle_l4virtio_port_tx (
    L4virtio_port * port)
```

Handle TX queue of the given port.

Parameters

| | |
|-------------|--|
| <i>port</i> | L4virtio_port to handle pending TX work for. |
|-------------|--|

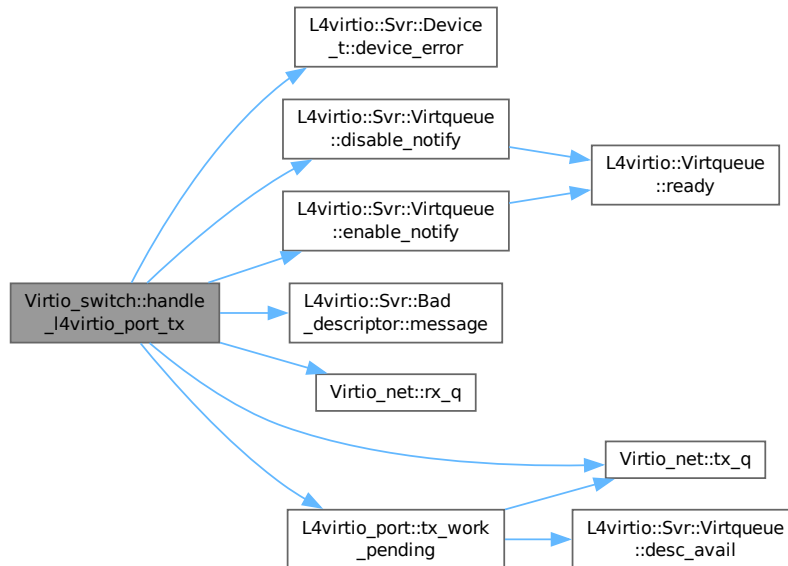
Return values

| | |
|--------------|--|
| <i>false</i> | Port hit its TX burst limit, and thus a TX pending reschedule notification was queued. |
| <i>true</i> | Port's entire TX queue was processed. |

Definition at line 180 of file [switch.cc](#).

References [L4virtio::Svr::Device_t< DATA >::device_error\(\)](#), [L4virtio::Svr::Virtqueue::disable_notify\(\)](#), [L4virtio::Svr::Virtqueue::enable_notify\(\)](#), [L4virtio::Svr::Bad_descriptor::message\(\)](#), [Virtio_net::rx_q\(\)](#), [Virtio_net::tx_q\(\)](#), and [L4virtio_port::tx_work_pending\(\)](#).

Here is the call graph for this function:



15.485.3.5 port_available()

```
bool Virtio_switch::port_available (
    bool monitor) [inline]
```

Is there still a free port on this switch available?

Parameters

| | |
|----------------|-------------------------------------|
| <i>monitor</i> | True if we look for a monitor slot. |
|----------------|-------------------------------------|

Return values

| | |
|--------------|--------------------|
| <i>true</i> | Port is available. |
| <i>false</i> | No port available. |

Definition at line 146 of file [switch.h](#).

The documentation for this class was generated from the following files:

- `pkg/virtio-net-switch/server/switch/switch.h`
- `pkg/virtio-net-switch/server/switch/switch.cc`

15.486 Virtio_vlan_mangle Class Reference

Class for VLAN packet rewriting.

```
#include <vlan.h>
```

Collaboration diagram for Virtio_vlan_mangle:



Public Member Functions

- [Virtio_vlan_mangle](#) ()
Default constructor.
- [l4_uint32_t copy_pkt](#) ([Buffer](#) &dst, [Buffer](#) &src)
Copy packet from src to dst.
- void [rewrite_hdr](#) ([Virtio_net::Hdr](#) *hdr)
Rewrite the virtio network header.

Static Public Member Functions

- static constexpr [Virtio_vlan_mangle add](#) ([l4_uint16_t](#) tci)
Construct an object that adds a VLAN tag.
- static constexpr [Virtio_vlan_mangle remove](#) ()
Construct an object that removes the VLAN tag.

15.486.1 Detailed Description

Class for VLAN packet rewriting.

Definition at line 36 of file [vlan.h](#).

15.486.2 Constructor & Destructor Documentation

15.486.2.1 Virtio_vlan_mangle()

`Virtio_vlan_mangle::Virtio_vlan_mangle () [inline]`

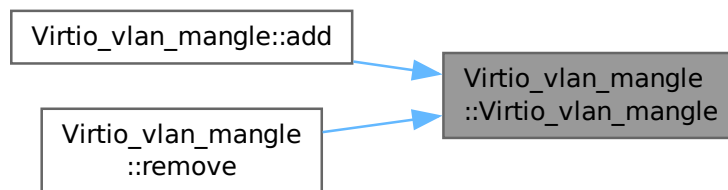
Default constructor.

The packet is not touched in any way.

Definition at line 52 of file [vlan.h](#).

Referenced by [add\(\)](#), and [remove\(\)](#).

Here is the caller graph for this function:



15.486.3 Member Function Documentation

15.486.3.1 add()

```
constexpr Virtio_vlan_mangle Virtio_vlan_mangle::add (
    14_uint16_t tci) [inline], [static], [constexpr]
```

Construct an object that adds a VLAN tag.

Parameters

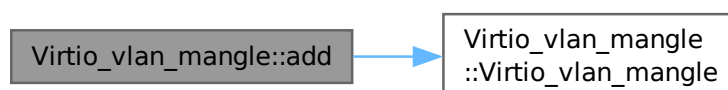
| | |
|------------|---------------------------------------|
| <i>tci</i> | The TCI field of the VLAN tag to add. |
|------------|---------------------------------------|

It is the callers responsibility to ensure that the packet is not already tagged.

Definition at line 64 of file [vlan.h](#).

References [Virtio_vlan_mangle\(\)](#).

Here is the call graph for this function:



15.486.3.2 copy_pkt()

```
l4_uint32_t Virtio_vlan_mangle::copy_pkt (
    Buffer & dst,
    Buffer & src) [inline]
```

Copy packet from *src* to *dst*.

Parameters

| | |
|------------|---------------------------|
| <i>src</i> | Source packet buffer |
| <i>dst</i> | Destination packet buffer |

Returns

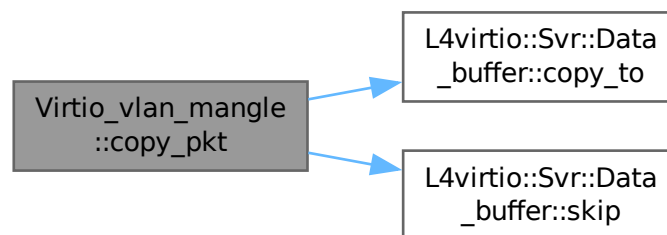
The number of bytes copied

Copy the data from *src* to *dst*, possibly rewriting parts of the packet. The method is expected to be called repeatedly until the source packet is finished. Partial copies are allowed (including reading nothing from the source buffer) as long as progress is made, i.e. repeatedly calling this function eventually consumes the source buffer.

Definition at line 93 of file [vlan.h](#).

References [L4virtio::Svr::Data_buffer::copy_to\(\)](#), [L4_LIKELY](#), [L4virtio::Svr::Data_buffer::left](#), [L4virtio::Svr::Data_buffer::pos](#), and [L4virtio::Svr::Data_buffer::skip\(\)](#).

Here is the call graph for this function:



15.486.3.3 remove()

```
constexpr Virtio_vlan_mangle Virtio_vlan_mangle::remove () [inline], [static], [constexpr]
```

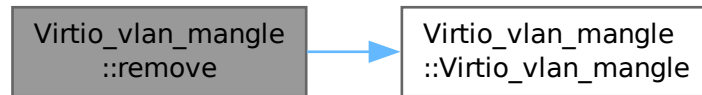
Construct an object that removes the VLAN tag.

This object assumes that the Ethernet packet has a VLAN tag and will slavishly remove the necessary bytes from the packet.

Definition at line 75 of file [vlan.h](#).

References [Virtio_vlan_mangle\(\)](#).

Here is the call graph for this function:



15.486.3.4 `rewrite_hdr()`

```
void Virtio_vlan_mangle::rewrite_hdr (  
    Virtio_net::Hdr * hdr) [inline]
```

Rewrite the virtio network header.

Parameters

| | |
|------------|---------------------------------|
| <i>hdr</i> | The virtio header of the packet |
|------------|---------------------------------|

This method is called exactly once for every virtio network packet. Any necessary changes to the header are done in-place.

Definition at line 142 of file [vlan.h](#).

References [L4_UNLIKELY](#).

The documentation for this class was generated from the following file:

- `pkg/virtio-net-switch/server/switch/vlan.h`

Chapter 16

File Documentation

16.1 asm_access.h

```
00001 /*
00002  * Copyright (C) 2021, 2024 Kernkonzept GmbH.
00003  * Author(s): Jakub Jermar <jakub.jermar@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00008 #pragma once
00009
00010 #include <l4/sys/l4int.h>
00011
00012 namespace Asm_access {
00013
00014 inline
00015 l4_uint8_t
00016 read(l4_uint8_t const *mem)
00017 {
00018     l4_uint8_t val;
00019
00020     asm volatile ("movb %[mem], %[val]" : [val] "=q" (val) : [mem] "m" (*mem));
00021
00022     return val;
00023 }
00024
00025 inline
00026 l4_uint16_t
00027 read(l4_uint16_t const *mem)
00028 {
00029     l4_uint16_t val;
00030
00031     asm volatile ("movw %[mem], %[val]" : [val] "=r" (val) : [mem] "m" (*mem));
00032
00033     return val;
00034 }
00035
00036 inline
00037 l4_uint32_t
00038 read(l4_uint32_t const *mem)
00039 {
00040     l4_uint32_t val;
00041
00042     asm volatile ("movl %[mem], %[val]" : [val] "=r" (val) : [mem] "m" (*mem));
00043
00044     return val;
00045 }
00046
00047 inline
00048 l4_uint64_t
00049 read(l4_uint64_t const *mem)
00050 {
00051     l4_uint64_t val;
00052
00053     asm volatile ("movq %[mem], %[val]" : [val] "=r" (val) : [mem] "m" (*mem));
00054
00055     return val;
00056 }
00057 }
```

```

00058 inline
00059 void
00060 write(l4_uint8_t val, l4_uint8_t *mem)
00061 {
00062     asm volatile ("movb %[val], %[mem]" : [mem] "=m" (*mem) : [val] "qi" (val));
00063 }
00064
00065 inline
00066 void
00067 write(l4_uint16_t val, l4_uint16_t *mem)
00068 {
00069     asm volatile ("movw %[val], %[mem]" : [mem] "=m" (*mem) : [val] "ri" (val));
00070 }
00071
00072 inline
00073 void
00074 write(l4_uint32_t val, l4_uint32_t *mem)
00075 {
00076     asm volatile ("movl %[val], %[mem]" : [mem] "=m" (*mem) : [val] "ri" (val));
00077 }
00078
00079 inline
00080 void
00081 write(l4_uint64_t val, l4_uint64_t *mem)
00082 {
00083     asm volatile ("movq %[val], %[mem]" : [mem] "=m" (*mem) : [val] "r" (val));
00084 }
00085
00086 }

```

16.2 asm_access.h

```

00001 /*
00002  * Copyright (C) 2021, 2024 Kernkonzept GmbH.
00003  * Author(s): Jakub Jermar <jakub.jermar@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00008 #pragma once
00009
00010 #include <l4/sys/l4int.h>
00011
00012 namespace Asm_access {
00013
00014 inline
00015 l4_uint8_t
00016 read(l4_uint8_t const *mem)
00017 {
00018     l4_uint8_t val;
00019
00020     asm volatile ("ldrb %[val], %[mem]" : [val] "=r" (val) : [mem] "m" (*mem));
00021
00022     return val;
00023 }
00024
00025 inline
00026 l4_uint16_t
00027 read(l4_uint16_t const *mem)
00028 {
00029     l4_uint16_t val;
00030
00031     asm volatile ("ldrh %[val], %[mem]" : [val] "=r" (val) : [mem] "m" (*mem));
00032
00033     return val;
00034 }
00035
00036 inline
00037 l4_uint32_t
00038 read(l4_uint32_t const *mem)
00039 {
00040     l4_uint32_t val;
00041
00042     asm volatile ("ldr %[val], %[mem]" : [val] "=r" (val) : [mem] "m" (*mem));
00043
00044     return val;
00045 }
00046
00047 inline
00048 void
00049 write(l4_uint8_t val, l4_uint8_t *mem)
00050 {
00051     asm volatile ("strb %[val], %[mem]" : [mem] "=m" (*mem) : [val] "r" (val));

```

```

00052 }
00053
00054 inline
00055 void
00056 write(l4_uint16_t val, l4_uint16_t *mem)
00057 {
00058     asm volatile ("strh %[val], %[mem]" : [mem] "=m" (*mem) : [val] "r" (val));
00059 }
00060
00061 inline
00062 void
00063 write(l4_uint32_t val, l4_uint32_t *mem)
00064 {
00065     asm volatile ("str %[val], %[mem]" : [mem] "=m" (*mem) : [val] "r" (val));
00066 }
00067
00068 }

```

16.3 asm_access.h

```

00001 /*
00002  * Copyright (C) 2021, 2024 Kernkonzept GmbH.
00003  * Author(s): Jakub Jermar <jakub.jermar@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00008 #pragma once
00009
00010 #include <l4/sys/l4int.h>
00011
00012 namespace Asm_access {
00013
00014     inline
00015     l4_uint8_t
00016     read(l4_uint8_t const *mem)
00017     {
00018         l4_uint8_t val;
00019
00020         asm volatile ("ldrb %w[val], %[mem]" : [val] "=r" (val) : [mem] "m" (*mem));
00021
00022         return val;
00023     }
00024
00025     inline
00026     l4_uint16_t
00027     read(l4_uint16_t const *mem)
00028     {
00029         l4_uint16_t val;
00030
00031         asm volatile ("ldrh %w[val], %[mem]" : [val] "=r" (val) : [mem] "m" (*mem));
00032
00033         return val;
00034     }
00035
00036     inline
00037     l4_uint32_t
00038     read(l4_uint32_t const *mem)
00039     {
00040         l4_uint32_t val;
00041
00042         asm volatile ("ldr %w[val], %[mem]" : [val] "=r" (val) : [mem] "m" (*mem));
00043
00044         return val;
00045     }
00046
00047     inline
00048     l4_uint64_t
00049     read(l4_uint64_t const *mem)
00050     {
00051         l4_uint64_t val;
00052
00053         asm volatile ("ldr %[val], %[mem]" : [val] "=r" (val) : [mem] "m" (*mem));
00054
00055         return val;
00056     }
00057
00058     inline
00059     void
00060     write(l4_uint8_t val, l4_uint8_t *mem)
00061     {
00062         asm volatile ("strb %w[val], %[mem]" : [mem] "=m" (*mem) : [val] "r" (val));
00063     }

```

```

00064
00065 inline
00066 void
00067 write(l4_uint16_t val, l4_uint16_t *mem)
00068 {
00069     asm volatile ("strh %w[val], %[mem]" : [mem] "=m" (*mem) : [val] "r" (val));
00070 }
00071
00072 inline
00073 void
00074 write(l4_uint32_t val, l4_uint32_t *mem)
00075 {
00076     asm volatile ("str %w[val], %[mem]" : [mem] "=m" (*mem) : [val] "r" (val));
00077 }
00078
00079 inline
00080 void
00081 write(l4_uint64_t val, l4_uint64_t *mem)
00082 {
00083     asm volatile ("str %[val], %[mem]" : [mem] "=m" (*mem) : [val] "r" (val));
00084 }
00085 }
00086 }

```

16.4 asm_access.h

```

00001 /*
00002  * Copyright (C) 2021, 2024 Kernkonzept GmbH.
00003  * Author(s): Jakub Jermar <jakub.jermar@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00008 #pragma once
00009
00010 #include <l4/drivers/asm_access_gen.h>

```

16.5 asm_access.h

```

00001 /*
00002  * Copyright (C) 2021, 2024 Kernkonzept GmbH.
00003  * Author(s): Jakub Jermar <jakub.jermar@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00008 #pragma once
00009
00010 #include <l4/drivers/asm_access_gen.h>

```

16.6 asm_access.h

```

00001 /*
00002  * Copyright (C) 2021, 2024 Kernkonzept GmbH.
00003  * Author(s): Georg Kotheimer <georg.kotheimer@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00008 #pragma once
00009
00010 #include <l4/sys/l4int.h>
00011
00012 namespace Asm_access {
00013
00014 inline
00015 l4_uint8_t
00016 read(l4_uint8_t const *mem)
00017 {
00018     l4_uint8_t val;
00019
00020     asm volatile ("lb %[val], %[mem]" : [val] "=r" (val) : [mem] "m" (*mem));
00021 }

```

```

00022     return val;
00023 }
00024
00025 inline
00026 l4_uint16_t
00027 read(l4_uint16_t const *mem)
00028 {
00029     l4_uint16_t val;
00030
00031     asm volatile ("lh %[val], %[mem]" : [val] "=r" (val) : [mem] "m" (*mem));
00032
00033     return val;
00034 }
00035
00036 inline
00037 l4_uint32_t
00038 read(l4_uint32_t const *mem)
00039 {
00040     l4_uint32_t val;
00041
00042     asm volatile ("lw %[val], %[mem]" : [val] "=r" (val) : [mem] "m" (*mem));
00043
00044     return val;
00045 }
00046
00047 inline
00048 void
00049 write(l4_uint8_t val, l4_uint8_t *mem)
00050 {
00051     asm volatile ("sb %[val], %[mem]" : [mem] "=m" (*mem) : [val] "r" (val));
00052 }
00053
00054 inline
00055 void
00056 write(l4_uint16_t val, l4_uint16_t *mem)
00057 {
00058     asm volatile ("sh %[val], %[mem]" : [mem] "=m" (*mem) : [val] "r" (val));
00059 }
00060
00061 inline
00062 void
00063 write(l4_uint32_t val, l4_uint32_t *mem)
00064 {
00065     asm volatile ("sw %[val], %[mem]" : [mem] "=m" (*mem) : [val] "r" (val));
00066 }
00067
00068 #if __riscv_xlen == 64
00069
00070 inline
00071 l4_uint64_t
00072 read(l4_uint64_t const *mem)
00073 {
00074     l4_uint64_t val;
00075
00076     asm volatile ("ld %[val], %[mem]" : [val] "=r" (val) : [mem] "m" (*mem));
00077
00078     return val;
00079 }
00080
00081 inline
00082 void
00083 write(l4_uint64_t val, l4_uint64_t *mem)
00084 {
00085     asm volatile ("sd %[val], %[mem]" : [mem] "=m" (*mem) : [val] "r" (val));
00086 }
00087
00088 #endif
00089
00090 }

```

16.7 asm_access.h

```

00001 /*
00002  * Copyright (C) 2021, 2024 Kernkonzept GmbH.
00003  * Author(s): Jakub Jermar <jakub.jermar@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00008 #pragma once
00009
00010 #include <l4/drivers/asm_access_gen.h>

```

16.8 asm_access.h

```

00001 /*
00002  * Copyright (C) 2021, 2024 Kernkonzept GmbH.
00003  * Author(s): Jakub Jermar <jakub.jermar@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00008 #pragma once
00009
00010 #include <l4/sys/l4int.h>
00011
00012 namespace Asm_access {
00013
00014 inline
00015 l4_uint8_t
00016 read(l4_uint8_t const *mem)
00017 {
00018     l4_uint8_t val;
00019
00020     asm volatile ("movb %[mem], %[val]" : [val] "=q" (val) : [mem] "m" (*mem));
00021
00022     return val;
00023 }
00024
00025 inline
00026 l4_uint16_t
00027 read(l4_uint16_t const *mem)
00028 {
00029     l4_uint16_t val;
00030
00031     asm volatile ("movw %[mem], %[val]" : [val] "=r" (val) : [mem] "m" (*mem));
00032
00033     return val;
00034 }
00035
00036 inline
00037 l4_uint32_t
00038 read(l4_uint32_t const *mem)
00039 {
00040     l4_uint32_t val;
00041
00042     asm volatile ("movl %[mem], %[val]" : [val] "=r" (val) : [mem] "m" (*mem));
00043
00044     return val;
00045 }
00046
00047 inline
00048 void
00049 write(l4_uint8_t val, l4_uint8_t *mem)
00050 {
00051     asm volatile ("movb %[val], %[mem]" : [mem] "=m" (*mem) : [val] "qi" (val));
00052 }
00053
00054 inline
00055 void
00056 write(l4_uint16_t val, l4_uint16_t *mem)
00057 {
00058     asm volatile ("movw %[val], %[mem]" : [mem] "=m" (*mem) : [val] "ri" (val));
00059 }
00060
00061 inline
00062 void
00063 write(l4_uint32_t val, l4_uint32_t *mem)
00064 {
00065     asm volatile ("movl %[val], %[mem]" : [mem] "=m" (*mem) : [val] "ri" (val));
00066 }
00067 }
00068

```

16.9 asm_access_gen.h

```

00001 /*
00002  * Copyright (C) 2021, 2024 Kernkonzept GmbH.
00003  * Author(s): Jakub Jermar <jakub.jermar@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include <l4/sys/l4int.h>

```



```

00010 #include <l4/cxx/type_traits>
00011
00012 namespace Asm_access {
00013
00014 template <typename T>
00015 struct is_supported_type
00016 {
00017     static const bool value = cxx::is_same<T, l4_uint8_t>::value
00018                             || cxx::is_same<T, l4_uint16_t>::value
00019                             || cxx::is_same<T, l4_uint32_t>::value
00020                             || cxx::is_same<T, l4_uint64_t>::value;
00021 };
00022
00023 template <typename T>
00024 inline
00025 typename cxx::enable_if<is_supported_type<T>::value, T>::type
00026 read(T const *mem)
00027 {
00028     return *reinterpret_cast<volatile T const *>(mem);
00029 }
00030
00031 template <typename T>
00032 inline
00033 typename cxx::enable_if<is_supported_type<T>::value, void>::type
00034 write(T val, T *mem)
00035 {
00036     *reinterpret_cast<volatile T *>(mem) = val;
00037 }
00038
00039 }

```

16.10 hw_mmio_register_block

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * Copyright (C) 2014-2021, 2023-2024 Kernkonzept GmbH.
00004  * Author(s): Alexander Warg <alexander.warg@kernkonzept.com>
00005  *            Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009 #pragma once
00010
00011 #include <l4/drivers/hw_register_block>
00012 #include <l4/drivers/asm_access.h>
00013
00014 namespace L4drivers {
00015
00016 class Mmio_register_block_base
00017 {
00018 protected:
00019     l4_addr_t _base;
00020     l4_addr_t _shift;
00021
00022 public:
00023     explicit Mmio_register_block_base(l4_addr_t base = 0, l4_addr_t shift = 0)
00024         : _base(base), _shift(shift) {}
00025
00026     template< typename T >
00027     T read(l4_addr_t reg) const
00028     { return Asm_access::read(reinterpret_cast<T const *>(_base + (reg « _shift))); }
00029
00030     template< typename T >
00031     void write(T value, l4_addr_t reg) const
00032     { Asm_access::write(value, reinterpret_cast<T *>(_base + (reg « _shift))); }
00033
00034     void set_base(l4_addr_t base) { _base = base; }
00035     void set_shift(l4_addr_t shift) { _shift = shift; }
00036
00037     l4_addr_t get_base() const { return _base; }
00038 };
00039
00040 template< unsigned MAX_BITS = 32 >
00041 struct Mmio_register_block
00042 : Register_block_impl<Mmio_register_block<MAX_BITS>, MAX_BITS>,
00043   Mmio_register_block_base
00044 {
00045     explicit Mmio_register_block(l4_addr_t base = 0, l4_addr_t shift = 0)
00046         : Mmio_register_block_base(base, shift) {}
00047 };
00048
00049 }
00050
00051 }
00052
00053 }

```

16.11 hw_register_block

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * Copyright (C) 2014-2021, 2023-2024 Kernkonzept GmbH.
00004  * Author(s): Alexander Warg <alexander.warg@kernkonzept.com>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include <l4/sys/types.h>
00011 #include <l4/cxx/type_traits>
00012
00013 namespace L4drivers {
00014
00015
00062
00063
00071 template< unsigned MAX_BITS = 32 >
00072 struct Register_block_base;
00073
00074 template<>
00075 struct Register_block_base<8>
00076 {
00077     virtual l4_uint8_t do_read_8(l4_addr_t reg) const = 0;
00078     virtual void do_write_8(l4_uint8_t value, l4_addr_t reg) = 0;
00079     virtual ~Register_block_base() = 0;
00080 };
00081
00082 inline Register_block_base<8>::~~Register_block_base() {}
00083
00084 template<>
00085 struct Register_block_base<16> : Register_block_base<8>
00086 {
00087     virtual l4_uint16_t do_read_16(l4_addr_t reg) const = 0;
00088     virtual void do_write_16(l4_uint16_t value, l4_addr_t reg) = 0;
00089 };
00090
00091 template<>
00092 struct Register_block_base<32> : Register_block_base<16>
00093 {
00094     virtual l4_uint32_t do_read_32(l4_addr_t reg) const = 0;
00095     virtual void do_write_32(l4_uint32_t value, l4_addr_t reg) = 0;
00096 };
00097
00098 template<>
00099 struct Register_block_base<64> : Register_block_base<32>
00100 {
00101     virtual l4_uint64_t do_read_64(l4_addr_t reg) const = 0;
00102     virtual void do_write_64(l4_uint64_t value, l4_addr_t reg) = 0;
00103 };
00104 #undef REGBLK_READ_TEMPLATE
00105 #undef REGBLK_WRITE_TEMPLATE
00106
00107 template<typename CHILD>
00108 struct Register_block_modify_mixin
00109 {
00110     template< typename T >
00111     T modify(T clear_bits, T set_bits, l4_addr_t reg) const
00112     {
00113         CHILD const *c = static_cast<CHILD const *>(this);
00114         T r = (c->template read<T>(reg) & ~clear_bits) | set_bits;
00115         c->template write<T>(r, reg);
00116         return r;
00117     }
00118
00119     template< typename T >
00120     T set(T set_bits, l4_addr_t reg) const
00121     { return this->template modify<T>(T(0), set_bits, reg); }
00122
00123     template< typename T >
00124     T clear(T clear_bits, l4_addr_t reg) const
00125     { return this->template modify<T>(clear_bits, T(0), reg); }
00126 };
00127
00128
00129 #define REGBLK_READ_TEMPLATE(sz) \
00130     template< typename T > \
00131     typename cxx::enable_if<sizeof(T) == (sz / 8), T>::type read(l4_addr_t reg) const \
00132     { \
00133         union X { T t; l4_uint##sz##_t v; } m; \
00134         m.v = _b->do_read_##sz (reg); \
00135         return m.t; \
00136     }
00137

```

```

00138 #define REGBLK_WRITE_TEMPLATE(sz) \
00139     template< typename T > \
00140     void write(T value, l4_addr_t reg, typename cxx::enable_if<sizeof(T) == (sz / 8), T>::type = T())
00141     const \
00142     { \
00143         union X { T t; l4_uint##sz##_t v; } m; \
00144         m.t = value; \
00145         _b->do_write_##sz(m.v, reg); \
00146     }
00147
00155 template< typename BLOCK >
00156 class Register_block_tmpl
00157 : public Register_block_modify_mixin<Register_block_tmpl<BLOCK> >
00158 {
00159 private:
00160     BLOCK *_b;
00161
00162 public:
00163     Register_block_tmpl(BLOCK *blk) : _b(blk) {}
00164     Register_block_tmpl() = default;
00165
00166     operator BLOCK * () const { return _b; }
00167
00168     REGBLK_READ_TEMPLATE(8)
00169     REGBLK_WRITE_TEMPLATE(8)
00170     REGBLK_READ_TEMPLATE(16)
00171     REGBLK_WRITE_TEMPLATE(16)
00172     REGBLK_READ_TEMPLATE(32)
00173     REGBLK_WRITE_TEMPLATE(32)
00174     REGBLK_READ_TEMPLATE(64)
00175     REGBLK_WRITE_TEMPLATE(64)
00176 };
00177
00178 #undef REGBLK_READ_TEMPLATE
00179 #undef REGBLK_WRITE_TEMPLATE
00180
00181 namespace __Type_helper {
00182     template<unsigned> struct Unsigned;
00183     template<> struct Unsigned<8> { typedef l4_uint8_t type; };
00184     template<> struct Unsigned<16> { typedef l4_uint16_t type; };
00185     template<> struct Unsigned<32> { typedef l4_uint32_t type; };
00186     template<> struct Unsigned<64> { typedef l4_uint64_t type; };
00187 };
00188
00189
00190
00200 template< unsigned BITS, typename BLOCK >
00201 class Ro_register_tmpl
00202 {
00203 protected:
00204     BLOCK _b;
00205     unsigned _o;
00206
00207 public:
00208     typedef typename __Type_helper::Unsigned<BITS>::type value_type;
00209
00210     Ro_register_tmpl(BLOCK const &blk, unsigned offset) : _b(blk), _o(offset) {}
00211     Ro_register_tmpl() = default;
00212
00213     operator value_type () const
00214     { return _b.template read<value_type>(_o); }
00215
00216     value_type read() const
00217     { return _b.template read<value_type>(_o); }
00218 };
00219
00220
00221
00222
00223
00224
00225
00226
00227
00228
00229
00230
00231
00232
00233
00234
00235
00236 template< unsigned BITS, typename BLOCK >
00237 class Register_tmpl : public Ro_register_tmpl<BITS, BLOCK>
00238 {
00239 public:
00240     typedef typename Ro_register_tmpl<BITS, BLOCK>::value_type value_type;
00241
00242     Register_tmpl(BLOCK const &blk, unsigned offset)
00243     : Ro_register_tmpl<BITS, BLOCK>(blk, offset)
00244     {}
00245
00246     Register_tmpl() = default;
00247
00248     Register_tmpl &operator = (value_type val)
00249     { this->_b.template write<value_type>(val, this->_o); return *this; }
00250
00251     void write(value_type val)
00252     { this->_b.template write<value_type>(val, this->_o); }
00253
00254     value_type set(value_type set_bits)
00255     { return this->_b.template set<value_type>(set_bits, this->_o); }

```

```

00276
00290 value_type clear(value_type clear_bits)
00291 { return this->_b.template clear<value_type>(clear_bits, this->_o); }
00292
00308 value_type modify(value_type clear_bits, value_type set_bits)
00309 { return this->_b.template modify<value_type>(clear_bits, set_bits, this->_o); }
00310 };
00311
00312
00324 template<
00325     unsigned MAX_BITS,
00326     typename BLOCK = Register_block_tmpl<
00327         Register_block_base<MAX_BITS>
00328     >
00329 >
00330 class Register_block
00331 {
00332 private:
00333     template< unsigned B, typename BLK > friend class Register_block;
00334     template< unsigned B, typename BLK > friend class Ro_register_block;
00335     typedef BLOCK Block;
00336     Block _b;
00337
00338 public:
00339     Register_block() = default;
00340     Register_block(Block const &blk) : _b(blk) {}
00341     Register_block &operator = (Block const &blk)
00342     { _b = blk; return *this; }
00343
00344     template< unsigned BITS >
00345     Register_block(Register_block<BITS> blk) : _b(blk._b) {}
00346
00347     typedef Register_tmpl<MAX_BITS, Block> Register;
00348     typedef Ro_register_tmpl<MAX_BITS, Block> Ro_register;
00349
00356     template< unsigned BITS >
00357     Ro_register_tmpl<BITS, Block> r(unsigned offset) const
00358     { return Ro_register_tmpl<BITS, Block>(this->_b, offset); }
00359
00365     Ro_register operator [] (unsigned offset) const
00366     { return this->r<MAX_BITS>(offset); }
00367
00375     template< unsigned BITS >
00376     Register_tmpl<BITS, Block> r(unsigned offset)
00377     { return Register_tmpl<BITS, Block>(this->_b, offset); }
00378
00385     Register operator [] (unsigned offset)
00386     { return this->r<MAX_BITS>(offset); }
00387 };
00388
00398 template<
00399     unsigned MAX_BITS,
00400     typename BLOCK = Register_block_tmpl<
00401         Register_block_base<MAX_BITS> const
00402     >
00403 >
00404 class Ro_register_block
00405 {
00406 private:
00407     template< unsigned B, typename BLK > friend class Ro_register_block;
00408     typedef BLOCK Block;
00409     Block _b;
00410
00411 public:
00412     Ro_register_block() = default;
00413     Ro_register_block(BLOCK const &blk) : _b(blk) {}
00414
00415     template< unsigned BITS >
00416     Ro_register_block(Register_block<BITS> const &blk) : _b(blk._b) {}
00417
00418     typedef Ro_register_tmpl<MAX_BITS, Block> Ro_register;
00419     typedef Ro_register Register;
00420
00426     Ro_register operator [] (unsigned offset) const
00427     { return Ro_register(this->_b, offset); }
00428
00435     template< unsigned BITS >
00436     Ro_register_tmpl<BITS, Block> r(unsigned offset) const
00437     { return Ro_register_tmpl<BITS, Block>(this->_b, offset); }
00438 };
00439
00440
00454 template< typename BASE, unsigned MAX_BITS = 32 >
00455 struct Register_block_impl;
00456
00457 #define REGBLK_IMPL_RW_TEMPLATE(sz, ...) \

```

```

00458     l4_uint##sz##_t do_read_##sz(l4_addr_t reg) const override \
00459     { return static_cast<BASE const *>(this)->template read<l4_uint##sz##_t>(reg); } \
00460     \
00461     void do_write_##sz(l4_uint##sz##_t value, l4_addr_t reg) override \
00462     { static_cast<BASE*>(this)->template write<l4_uint##sz##_t>(value, reg); }
00463
00464
00465 template< typename BASE >
00466 struct Register_block_impl<BASE, 8> : public Register_block_base<8>
00467 {
00468     REGBLK_IMPL_RW_TEMPLATE(8);
00469 };
00470
00471 template< typename BASE >
00472 struct Register_block_impl<BASE, 16> : public Register_block_base<16>
00473 {
00474     REGBLK_IMPL_RW_TEMPLATE(8);
00475     REGBLK_IMPL_RW_TEMPLATE(16);
00476 };
00477
00478 template< typename BASE >
00479 struct Register_block_impl<BASE, 32> : public Register_block_base<32>
00480 {
00481     REGBLK_IMPL_RW_TEMPLATE(8);
00482     REGBLK_IMPL_RW_TEMPLATE(16);
00483     REGBLK_IMPL_RW_TEMPLATE(32);
00484 };
00485
00486 template< typename BASE >
00487 struct Register_block_impl<BASE, 64> : public Register_block_base<64>
00488 {
00489     REGBLK_IMPL_RW_TEMPLATE(8);
00490     REGBLK_IMPL_RW_TEMPLATE(16);
00491     REGBLK_IMPL_RW_TEMPLATE(32);
00492     REGBLK_IMPL_RW_TEMPLATE(64);
00493 };
00494
00495 #undef REGBLK_IMPL_RW_TEMPLATE
00496
00497 }

```

16.12 io_regblock.h

```

00001 /*
00002  * Copyright (C) 2012 Technische Universität Dresden.
00003  * Author(s): Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 namespace L4
00010 {
00011     class Io_register_block
00012     {
00013     public:
00017         virtual unsigned char read8(unsigned long reg) const = 0;
00018
00022         virtual unsigned short read16(unsigned long reg) const = 0;
00023
00027         virtual unsigned int read32(unsigned long reg) const = 0;
00028
00029         /*
00030          * \brief Read register with an 8 byte access.
00031          */
00032         //virtual unsigned long long read64(unsigned long reg) const = 0;
00033
00037         virtual void write8(unsigned long reg, unsigned char value) const = 0;
00038
00042         virtual void write16(unsigned long reg, unsigned short value) const = 0;
00043
00047         virtual void write32(unsigned long reg, unsigned int value) const = 0;
00048
00049         /*
00050          * \brief Write register with an 8 byte access.
00051          */
00052         //virtual void write64(unsigned long reg, unsigned long long value) const = 0;
00053
00057         virtual unsigned long addr(unsigned long reg) const = 0;
00058
00064         virtual void delay() const = 0;
00065
00066         virtual ~Io_register_block() = 0;

```

```

00067
00075     template< typename R >
00076     R read(unsigned long reg) const
00077     {
00078         static_assert(sizeof(R) == 1 || sizeof(R) == 2 || sizeof(R) == 4,
00079             "Invalid size");
00080
00081         switch (sizeof(R))
00082         {
00083             case 1: return read8(reg);
00084             case 2: return read16(reg);
00085             case 4: return read32(reg);
00086         };
00087     }
00088
00096     template< typename R >
00097     void write(unsigned long reg, R value) const
00098     {
00099         static_assert(sizeof(R) == 1 || sizeof(R) == 2 || sizeof(R) == 4,
00100             "Invalid size");
00101
00102         switch (sizeof(R))
00103         {
00104             case 1: write8(reg, value); return;
00105             case 2: writel6(reg, value); return;
00106             case 4: write32(reg, value); return;
00107         };
00108     }
00109
00120     template< typename R >
00121     R modify(unsigned long reg, R clear_bits, R set_bits) const
00122     {
00123         R r = (read<R>(reg) & ~clear_bits) | set_bits;
00124         write(reg, r);
00125         return r;
00126     }
00127
00134     template< typename R >
00135     R set(unsigned long reg, R set_bits) const
00136     {
00137         return modify<R>(reg, 0, set_bits);
00138     }
00139
00146     template< typename R >
00147     R clear(unsigned long reg, R clear_bits) const
00148     {
00149         return modify<R>(reg, clear_bits, 0);
00150     }
00151 };
00152
00153 inline Io_register_block::~Io_register_block() {}
00154
00155
00156
00157 class Io_register_block_mmio : public Io_register_block
00158 {
00159 private:
00160     template< typename R >
00161     R _read(unsigned long reg) const
00162     { return *reinterpret_cast<volatile R*>(_base + (reg << _shift)); }
00163
00164     template< typename R >
00165     void _write(unsigned long reg, R val) const
00166     { *reinterpret_cast<volatile R*>(_base + (reg << _shift)) = val; }
00167
00168 public:
00169     Io_register_block_mmio(unsigned long base, unsigned char shift = 0)
00170     : _base(base), _shift(shift)
00171     {}
00172
00173     unsigned long addr(unsigned long reg) const override
00174     { return _base + (reg << _shift); }
00175
00176     unsigned char read8(unsigned long reg) const override
00177     { return _read<unsigned char>(reg); }
00178     unsigned short read16(unsigned long reg) const override
00179     { return _read<unsigned short>(reg); }
00180     unsigned int read32(unsigned long reg) const override
00181     { return _read<unsigned int>(reg); }
00182
00183     void write8(unsigned long reg, unsigned char val) const override
00184     { _write(reg, val); }
00185     void writel6(unsigned long reg, unsigned short val) const override
00186     { _write(reg, val); }
00187     void write32(unsigned long reg, unsigned int val) const override
00188     { _write(reg, val); }
00189

```

```

00190     void delay() const override
00191     {}
00192
00193     void set_base(unsigned long base) { _base = base; }
00194     void set_shift(unsigned char shift) { _shift = shift; }
00195
00196     unsigned long get_base() const { return _base; }
00197
00198 private:
00199     unsigned long _base;
00200     unsigned char _shift;
00201 };
00202
00203 template<typename ACCESS_TYPE>
00204 class Io_register_block_mmio_fixed_width : public Io_register_block
00205 {
00206 private:
00207     template< typename R >
00208     R _read(unsigned long reg) const
00209     { return *reinterpret_cast<volatile ACCESS_TYPE *>(_base + (reg « _shift)); }
00210
00211     template< typename R >
00212     void _write(unsigned long reg, R val) const
00213     { *reinterpret_cast<volatile ACCESS_TYPE *>(_base + (reg « _shift)) = val; }
00214
00215 public:
00216     Io_register_block_mmio_fixed_width(unsigned long base, unsigned char shift = 0)
00217     : _base(base), _shift(shift)
00218     {}
00219
00220     unsigned long addr(unsigned long reg) const
00221     { return _base + (reg « _shift); }
00222
00223     unsigned char read8(unsigned long reg) const override
00224     { return _read<unsigned char>(reg); }
00225     unsigned short read16(unsigned long reg) const override
00226     { return _read<unsigned short>(reg); }
00227     unsigned int read32(unsigned long reg) const override
00228     { return _read<unsigned int>(reg); }
00229
00230     void write8(unsigned long reg, unsigned char val) const override
00231     { _write(reg, val); }
00232     void write16(unsigned long reg, unsigned short val) const override
00233     { _write(reg, val); }
00234     void write32(unsigned long reg, unsigned int val) const override
00235     { _write(reg, val); }
00236
00237     void delay() const override
00238     {}
00239
00240 private:
00241     unsigned long _base;
00242     unsigned char _shift;
00243 };
00244 }

```

16.13 io_regblock_port.h

```

00001 /*
00002  * Copyright (C) 2012 Technische Universität Dresden.
00003  * Author(s): Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include "io_regblock.h"
00010
00011 namespace L4
00012 {
00013     class Io_register_block_port : public Io_register_block
00014     {
00015     public:
00016         Io_register_block_port(unsigned long base)
00017         : _base(base)
00018         {}
00019
00020         unsigned long addr(unsigned long reg) const override { return _base + reg; }
00021
00022         unsigned char read8(unsigned long reg) const override
00023         {
00024             unsigned char val;
00025             asm volatile("inb %w1, %b0" : "=a" (val) : "Nd" (_base + reg));

```

```

00026     return val;
00027 }
00028
00029 unsigned short read16(unsigned long reg) const override
00030 {
00031     unsigned short val;
00032     asm volatile("inw %w1, %w0" : "=a" (val) : "Nd" (_base + reg));
00033     return val;
00034 }
00035
00036 unsigned int read32(unsigned long reg) const override
00037 {
00038     unsigned int val;
00039     asm volatile("in %w1, %0" : "=a" (val) : "Nd" (_base + reg));
00040     return val;
00041 }
00042
00043 void write8(unsigned long reg, unsigned char val) const override
00044 { asm volatile("outb %b0, %w1" : : "a" (val), "Nd" (_base + reg)); }
00045
00046 void write16(unsigned long reg, unsigned short val) const override
00047 { asm volatile("outw %w0, %w1" : : "a" (val), "Nd" (_base + reg)); }
00048
00049 void write32(unsigned long reg, unsigned int val) const override
00050 { asm volatile("out %0, %w1" : : "a" (val), "Nd" (_base + reg)); }
00051
00052 void delay() const override
00053 { asm volatile ("outb %al,$0x80"); }
00054
00055 private:
00056     unsigned long _base;
00057 };
00058 }

```

16.14 poll_timeout_counter.h

```

00001 /*
00002  * Copyright (C) 2012 Technische Universität Dresden.
00003  * Author(s): Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 namespace L4 {
00010
00011     00034 class Poll_timeout_counter
00012     {
00013     public:
00042     Poll_timeout_counter(unsigned counter_val)
00014     {
00043     {
00044     set(counter_val);
00045     }
00046
00053     void set(unsigned counter_val)
00015     {
00054     {
00055     _c = counter_val;
00056     }
00057
00061     bool test(bool expression = true)
00016     {
00062     {
00063     if (!expression)
00064     return false;
00065
00066     if (_c)
00067     {
00068     --_c;
00069     return true;
00070     }
00071
00072     return false;
00073     }
00074
00081     bool timed_out() const { return _c == 0; }
00082
00083 private:
00084     unsigned _c;
00085 };
00086
00087 }

```


16.15 device.h

```

00001 #pragma once
00002
00003 #include <stdint.h>
00004
00005 struct l4re_device_spec_dt_ids
00006 {
00007     const char *compatible;
00008 };
00009
00010 struct l4re_device_spec_pcidev_pciids
00011 {
00012     uint16_t vendor;
00013     uint16_t function;
00014 };
00015
00016 struct l4re_device_spec_pcidev_ids
00017 {
00018     struct l4re_device_spec_pcidev_pciids *pcidevs;
00019 };
00020
00021 struct l4re_device_ids
00022 {
00023     struct l4re_device_spec_dt_ids *dt = nullptr;
00024     struct l4re_device_spec_pcidev_ids *pcidev = nullptr;
00025 };
00026
00027 // ---- Internal device
00028 namespace L4 {
00029
00030     class Dev_obj
00031     {
00032     };
00033
00034 }
00035
00036 struct l4re_device_type_base
00037 {
00038     virtual L4::Dev_obj *create(void *obj_mem, unsigned arg1) = 0;
00039     virtual unsigned obj_size() = 0;
00040     virtual const l4re_device_ids *ids() = 0;
00041 };
00042
00043 #include <stdio.h>
00044 #include <string.h>
00045
00046 static inline
00047 L4::Dev_obj *
00048 l4re_dev_create_by_dt_compatible(const char *dt_compatible,
00049                                 void *obj_store, unsigned obj_store_size,
00050                                 unsigned arg1)
00051 {
00052     l4re_device_type_base *device = nullptr;
00053
00054     extern l4re_device_type_base *__L4RE_DEVICE_BEGIN[];
00055     extern l4re_device_type_base *__L4RE_DEVICE_END[];
00056
00057     for (auto const *d = __L4RE_DEVICE_BEGIN; d < __L4RE_DEVICE_END; ++d)
00058     {
00059         const l4re_device_ids *ids = (*d)->ids();
00060         if (ids->dt)
00061             for (l4re_device_spec_dt_ids *dt_id = ids->dt; dt_id->compatible; ++dt_id)
00062                 if (!strcmp(dt_compatible, dt_id->compatible))
00063                 {
00064                     device = *d;
00065                     break;
00066                 }
00067     }
00068
00069     if (!device)
00070         return nullptr;
00071
00072     if (device->obj_size() > obj_store_size)
00073         return nullptr;
00074
00075     return device->create(reinterpret_cast<void *>(obj_store), arg1);
00076 }
00077
00078 #define l4re_register_device(Device_class_base, Device_class, \
00079                             instance_name, __ids) \
00080     struct l4re_device_type_##instance_name \
00081     : public l4re_device_type_base \
00082     { \
00083         Device_class_base *create(void *obj_mem, unsigned arg1) override \
00084         { return new (obj_mem) Device_class(arg1); } \

```

```

00085     \
00086     unsigned obj_size() override { return sizeof(Device_class); } \
00087     const l4re_device_ids *ids() override { return &__ids; } \
00088 }; \
00089 static const l4re_device_type_##instance_name \
00090     l4re_device_inst_##instance_name; \
00091 static const l4re_device_type_base * const \
00092     __attribute__((section(".data.l4redevice"),used)) \
00093     l4re_device_inst_p_##instance_name \
00094     = &l4re_device_inst_##instance_name

```

16.16 uart_16550.h

```

00001 /*
00002  * Copyright (C) 2008-2021 Technische Universität Dresden.
00003  * Copyright (C) 2023-2025 Kernkonzept GmbH.
00004  * Author(s): Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00005  *             Alexander Warg <alexander.warg@os.inf.tu-dresden.de>
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009 #pragma once
00010
00011 #include "uart_base.h"
00012
00013 namespace L4 {
00014
00015 class Uart_16550 : public Uart
00016 {
00017 protected:
00018     enum Registers
00019     {
00020         TRB      = 0x00, // Transmit/Receive Buffer (read/write)
00021         BRD_LOW  = 0x00, // Baud Rate Divisor LSB if bit 7 of LCR is set (read/write)
00022         IER      = 0x01, // Interrupt Enable Register (read/write)
00023         BRD_HIGH = 0x01, // Baud Rate Divisor MSB if bit 7 of LCR is set (read/write)
00024         IIR      = 0x02, // Interrupt Identification Register (read only)
00025         FCR      = 0x02, // 16550 FIFO Control Register (write only)
00026         LCR      = 0x03, // Line Control Register (read/write)
00027         MCR      = 0x04, // Modem Control Register (read/write)
00028         LSR      = 0x05, // Line Status Register (read only)
00029         MSR      = 0x06, // Modem Status Register (read only)
00030         SPR      = 0x07, // Scratch Pad Register (read/write)
00031     };
00032
00033     enum Register_value_iir
00034     {
00035         IIR_BUSY = 7,
00036     };
00037
00038     enum Register_value_lsr
00039     {
00040         LSR_DR      = 0x01, // Receiver data ready
00041         LSR_THRE    = 0x20, // Transmit hold register empty
00042         LSR_TSRE    = 0x40, // Transmitter empty
00043     };
00044
00045     enum Init_values
00046     {
00047 #ifdef UART_16550_INIT_MCR
00048         Init_mcr = UART_16550_INIT_MCR,
00049 #else
00050         Init_mcr = 0,
00051 #endif
00052 #ifdef UART_16550_INIT_IER
00053         Init_ier = UART_16550_INIT_IER,
00054 #else
00055         Init_ier = 0,
00056 #endif
00057 #ifdef UART_16550_INIT_FCR
00058         Init_fcr = UART_16550_INIT_FCR,
00059 #else
00060         Init_fcr = 0,
00061 #endif
00062     };
00063
00064 public:
00065     enum
00066     {
00067         PAR_NONE = 0x00,
00068         PAR_EVEN = 0x18,
00069         PAR_ODD  = 0x08,
00070         DAT_5    = 0x00,

```

```

00071     DAT_6     = 0x01,
00072     DAT_7     = 0x02,
00073     DAT_8     = 0x03,
00074     STOP_1    = 0x00,
00075     STOP_2    = 0x04,
00076
00077     MODE_8N1 = PAR_NONE | DAT_8 | STOP_1,
00078     MODE_7E1 = PAR_EVEN | DAT_7 | STOP_1,
00079
00080     // these two values are to leave either mode
00081     // or baud rate unchanged on a call to change_mode
00082     MODE_NC   = 0x1000000,
00083     BAUD_NC    = 0x1000000,
00084
00085     Base_rate_x86 = 115200,
00086     Base_rate_pxa = 921600,
00087 };
00088
00089 explicit Uart_16550(unsigned long base_rate, unsigned char init_flags = 0,
00090                    unsigned char ier_bits = Init_ier,
00091                    unsigned char mcr_bits = Init_mcr,
00092                    unsigned char fcr_bits = Init_fcr)
00093 : _base_rate(base_rate), _init_flags(init_flags), _mcr_bits(mcr_bits),
00094   _ier_bits(ier_bits), _fcr_bits(fcr_bits)
00095 {}
00096
00097 bool startup(Io_register_block const *regs) override;
00098 void shutdown() override;
00099 bool change_mode(Transfer_mode m, Baud_rate r) override;
00100 bool tx_avail() const;
00101 void wait_tx_done() const;
00102 inline void out_char(char c) const;
00103 int write(char const *s, unsigned long count,
00104          bool blocking = true) const override;
00105
00106 #ifndef UART_WITHOUT_INPUT
00107 bool enable_rx_irq(bool enable = true) override;
00108 int char_avail() const override;
00109 int get_char(bool blocking = true) const override;
00110 #endif
00111
00112 private:
00113 unsigned long _base_rate;
00114 unsigned char _init_flags;
00115 unsigned char _mcr_bits;
00116 unsigned char _ier_bits;
00117 unsigned char _fcr_bits;
00118 };
00119
00120 } // namespace L4

```

16.17 uart_16550_dw.h

```

00001 /*
00002  * Copyright (C) 2015 Technische Universität Dresden.
00003  * Copyright (C) 2023-2025 Kernkonzept GmbH.
00004  * Author(s): Adam Lackorzynski <adam@l4re.org>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include "uart_16550.h"
00011
00012 namespace L4 {
00013
00014 class Uart_16550_dw : public Uart_16550
00015 {
00016 public:
00017     explicit Uart_16550_dw(unsigned long base_rate)
00018     : Uart_16550(base_rate)
00019     {}
00020
00021 #ifndef UART_WITHOUT_INPUT
00022     void irq_ack() override;
00023 #endif
00024 };
00025
00026 } // namespace L4

```

16.18 uart_apb.h

```

00001 /*
00002  * Copyright (C) 2017, 2019, 2023-2025 Kernkonzept GmbH.
00003  * Author(s): Georg Kotheimer <georg.kotheimer@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include "uart_base.h"
00010
00011 namespace L4 {
00012
00017 class Uart_apb : public Uart
00018 {
00019 public:
00021   Uart_apb(unsigned freq) : _freq(freq) {}
00022   bool startup(Io_register_block const *) override;
00023   void shutdown() override;
00024   bool change_mode(Transfer_mode m, Baud_rate r) override;
00025   bool tx_avail() const;
00026   void wait_tx_done() const;
00027   inline void out_char(char c) const;
00028   int write(char const *s, unsigned long count,
00029            bool blocking = true) const override;
00030
00031 #ifndef UART_WITHOUT_INPUT
00032   bool enable_rx_irq(bool enable) override;
00033   int char_avail() const override;
00034   int get_char(bool blocking = true) const override;
00035 #endif
00036
00037 private:
00038   void set_rate(Baud_rate r);
00039   unsigned _freq;
00040 };
00041
00042 } // namespace L4

```

16.19 uart_base.h

```

00001 /*
00002  * Copyright (C) 2009-2012 Technische Universität Dresden.
00003  * Copyright (C) 2023-2025 Kernkonzept GmbH.
00004  * Author(s): Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include <stddef.h>
00011 #include <l4/drivers/io_regblock.h>
00012 #include <l4/sys/types.h>
00013
00014 #include <l4/drivers/device.h>
00015 #include "poll_timeout_counter.h"
00016
00017 namespace L4 {
00018
00022 class Uart : public Dev_obj
00023 {
00024 protected:
00025   unsigned _mode;
00026   unsigned _rate;
00027   Io_register_block const *_regs;
00028
00029 public:
00030   void *operator new (size_t, void *p) { return p; }
00031
00032 public:
00033   typedef unsigned Transfer_mode;
00034   typedef unsigned Baud_rate;
00035
00036   Uart()
00037   : _mode(~0U), _rate(~0U)
00038   {}
00039
00047   virtual bool reg_shift(unsigned char *shift __attribute__((unused))) const
00048   { return false; }
00049
00057   virtual bool startup(Io_register_block const *regs) = 0;

```

```

00058
00059     virtual ~Uart() {}
00060
00064     virtual void shutdown() = 0;
00065
00077     virtual bool change_mode(Transfer_mode m, Baud_rate r) = 0;
00078
00089     virtual int write(char const *s, unsigned long count,
00090                      bool blocking = true) const = 0;
00091
00097     Transfer_mode mode() const { return _mode; }
00098
00104     Baud_rate rate() const { return _rate; }
00105
00106 #ifndef UART_WITHOUT_INPUT
00107
00115     virtual bool enable_rx_irq(bool = true) { return false; }
00116
00120     virtual void irq_ack() {}
00121
00128     virtual int char_avail() const = 0;
00129
00138     virtual int get_char(bool blocking = true) const = 0;
00139
00140 #endif // !UART_WITHOUT_INPUT
00141
00142 protected:
00154     template <typename Uart_driver, bool Timeout_guard = true>
00155     int generic_write(char const *s, unsigned long count,
00156                      bool blocking = true) const
00157     {
00158         auto *self = static_cast<Uart_driver const*>(this);
00159
00160         unsigned long c;
00161         for (c = 0; c < count; ++c)
00162         {
00163             if (!blocking && !self->tx_avail())
00164                 break;
00165
00166             if constexpr (Timeout_guard)
00167             {
00168                 Poll_timeout_counter i(3000000);
00169                 while (i.test(!self->tx_avail()))
00170                     ;
00171             }
00172             else
00173             {
00174                 while (!self->tx_avail())
00175                     ;
00176             }
00177
00178             self->out_char(*s++);
00179         }
00180
00181         if (blocking)
00182             self->wait_tx_done();
00183
00184         return c;
00185     }
00186 };
00187
00188 } // namespace L4
00189
00190 /*
00191  * Instantiate a UART, given the memory.
00192  */
00193 static inline
00194 L4::Uart *
00195 l4re_dev_uart_create_by_dt_compatible(const char *dt_compatible,
00196                                       void *obj_buf, unsigned obj_buf_size,
00197                                       unsigned freq)
00198 {
00199     L4::Dev_obj *dev = l4re_dev_create_by_dt_compatible(dt_compatible,
00200                                                         obj_buf, obj_buf_size,
00201                                                         freq);
00202     return static_cast<L4::Uart *>(dev);
00203 }
00204
00205 /*
00206  * Instantiate a UART a single time, use an internal static buffer.
00207  */
00208 static inline
00209 L4::Uart *
00210 l4re_dev_uart_create_by_dt_compatible_once(const char *dt_compatible, unsigned freq)
00211 {
00212     static bool once_done = false;
00213

```

```

00214     if (once_done)
00215         return nullptr;
00216
00217     static char __attribute__((aligned(sizeof(long) * 2))) obj_buf[64];
00218     L4::Uart *uart
00219         = l4re_dev_uart_create_by_dt_compatible(dt_compatible,
00220                                                 obj_buf, sizeof(obj_buf),
00221                                                 freq);
00222     if (uart)
00223         once_done = true;
00224
00225     return uart;
00226 }
00227
00228 #define l4re_register_device_uart(Device_class, instance_name, \
00229                                 device_dt_ids, pci_ids) \
00230     static const struct l4re_device_ids __dev_spec_##instance_name \
00231         = { .dt = device_dt_ids, .pcidev = pci_ids }; \
00232     l4re_register_device(L4::Uart, Device_class, instance_name, \
00233                         __dev_spec_##instance_name)
00234
00235 #define l4re_register_device_uart_dt(Device_class, instance_name, dt_ids) \
00236     l4re_register_device_uart(Device_class, instance_name, dt_ids, nullptr)
00237
00238 #define l4re_register_device_uart_pci(Device_class, instance_name, pci_ids) \
00239     l4re_register_device_uart(Device_class, instance_name, nullptr, pci_ids)
00240

```

16.20 uart_bcm2835.h

```

00001 /*
00002  * Copyright (C) 2026 Kernkonzept GmbH.
00003  * Author(s): Jakub Jermar <jakub.jermar@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00008 #pragma once
00009
00010 #include "uart_16550.h"
00011
00012 namespace L4 {
00013
00014     /*
00015      * According to the BCM2835 ARM Peripherals document, the mini UART is not
00016      * compatible with the 16550 UART, but has a 16550-like register layout.
00017      * The document is known to have many errors, including the mini UART part, as
00018      * described in this errata:
00019      *
00020      * https://elinux.org/BCM2835\_datasheet\_errata#p12
00021      *
00022      * Even from the errata the meaning of some bits in the IER is not very clear.
00023      */
00024     class Uart_bcm2835 : public Uart_16550
00025     {
00026     public:
00027         explicit Uart_bcm2835(unsigned long base_rate)
00028             : Uart_16550(base_rate, 0, 8)
00029         {}
00030
00031         bool reg_shift(unsigned char *shift) const override
00032         {
00033             *shift = 2;
00034             return true;
00035         }
00036     };
00037
00038 } // namespace L4

```

16.21 uart_cadence.h

```

00001 /*
00002  * Copyright (C) 2013 Technische Universität Dresden.
00003  * Copyright (C) 2023-2025 Kernkonzept GmbH.
00004  * Author(s): Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */

```

```

00008 #pragma once
00009
00010 #include "uart_base.h"
00011
00012 namespace L4 {
00013
00014 class Uart_cadence : public Uart
00015 {
00016 public:
00017     explicit Uart_cadence(unsigned base_rate) : _base_rate(base_rate) {}
00018     bool startup(Io_register_block const *) override;
00019     void shutdown() override;
00020     bool change_mode(Transfer_mode m, Baud_rate r) override;
00021     bool tx_avail() const;
00022     void wait_tx_done() const {}
00023     inline void out_char(char c) const;
00024     int write(char const *s, unsigned long count,
00025              bool blocking = true) const override;
00026
00027 #ifndef UART_WITHOUT_INPUT
00028     bool enable_rx_irq(bool) override;
00029     void irq_ack() override;
00030     int char_avail() const override;
00031     int get_char(bool blocking = true) const override;
00032 #endif
00033
00034 private:
00035     unsigned _base_rate;
00036 };
00037
00038 } // namespace L4

```

16.22 uart_dcc-v6.h

```

00001 /*
00002  * Copyright (C) 2009 Technische Universität Dresden.
00003  * Copyright (C) 2023-2025 Kernkonzept GmbH.
00004  * Author(s): Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include "uart_base.h"
00011
00012 namespace L4 {
00013
00014 class Uart_dcc_v6 : public Uart
00015 {
00016 public:
00017     explicit Uart_dcc_v6() {}
00018     explicit Uart_dcc_v6(unsigned /*base_rate*/) {}
00019     bool startup(Io_register_block const *) override;
00020     void shutdown() override;
00021     bool change_mode(Transfer_mode m, Baud_rate r) override;
00022     bool tx_avail() const;
00023     void wait_tx_done() const;
00024     inline void out_char(char c) const;
00025     int write(char const *s, unsigned long count,
00026              bool blocking = true) const override;
00027
00028 #ifndef UART_WITHOUT_INPUT
00029     int char_avail() const override;
00030     int get_char(bool blocking = true) const override;
00031 #endif
00032
00033 private:
00034     unsigned get_status() const;
00035 };
00036
00037 } // namespace L4

```

16.23 uart_dm.h

```

00001 /*
00002  * Copyright (C) 2021-2022 Stephan Gerhold <stephan@gerhold.net>
00003  * Copyright (C) 2022-2025 Kernkonzept GmbH.
00004  * Author(s): Stephan Gerhold <stephan@gerhold.net>

```

```

00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include "uart_base.h"
00011
00012 namespace L4 {
00013
00014 class Uart_dm : public Uart
00015 {
00016 public:
00017     explicit Uart_dm(unsigned /*base_rate*/) {}
00018     bool startup(Io_register_block const *) override;
00019     void shutdown() override;
00020     bool change_mode(Transfer_mode m, Baud_rate r) override;
00021     bool tx_avail() const;
00022     void wait_tx_done() const;
00023     inline void out_char(char c) const;
00024     int write(char const *s, unsigned long count,
00025              bool blocking = true) const override;
00026
00027 #ifndef UART_WITHOUT_INPUT
00028     bool enable_rx_irq(bool enable = true) override;
00029     int char_avail() const override;
00030     int get_char(bool blocking = true) const override;
00031 #endif
00032 };
00033
00034 } // namespace L4

```

16.24 uart_dummy.h

```

00001 /*
00002  * Copyright 2009 Technische Universität Dresden.
00003  * Copyright (C) 2023-2025 Kernkonzept GmbH.
00004  * Author(s): Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include "uart_base.h"
00011
00012 namespace L4 {
00013
00014 class Uart_dummy : public Uart
00015 {
00016 public:
00017     explicit Uart_dummy() {}
00018     explicit Uart_dummy(unsigned /*base_rate*/) {}
00019     bool startup(Io_register_block const *) override { return true; }
00020     void shutdown() override {}
00021     bool change_mode(Transfer_mode, Baud_rate) override { return true; }
00022     inline void out_char(char /*ch*/) const {}
00023     int write(char const * /*str*/, unsigned long /*count*/,
00024              bool /*blocking*/ = true) const override
00025     { return 0; }
00026
00027 #ifndef UART_WITHOUT_INPUT
00028     int char_avail() const override { return false; }
00029     int get_char(bool /*blocking*/ = true) const override { return 0; }
00030 #endif
00031 };
00032
00033 } // namespace L4

```

16.25 uart_geni.h

```

00001 /*
00002  * Copyright (C) 2022-2025 Kernkonzept GmbH.
00003  * Author(s): Stephan Gerhold <stephan.gerhold@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include "uart_base.h"

```



```

00010
00011 namespace L4 {
00012
00013 class Uart_geni : public Uart
00014 {
00015 public:
00016     explicit Uart_geni(unsigned /*base_rate*/) {}
00017     bool startup(Io_register_block const *) override;
00018     void shutdown() override;
00019     bool change_mode(Transfer_mode m, Baud_rate r) override;
00020     bool tx_avail() const;
00021     void wait_tx_done() const;
00022     void out_char(char c) const;
00023     int write(char const *s, unsigned long count,
00024              bool blocking = true) const override;
00025
00026 #ifndef UART_WITHOUT_INPUT
00027     bool enable_rx_irq(bool enable = true) override;
00028     void irq_ack() override;
00029     int char_avail() const override;
00030     int get_char(bool blocking = true) const override;
00031 #endif
00032 };
00033
00034 } // namespace L4

```

16.26 uart_imx.h

```

00001 /*
00002  * Copyright (C) 2008-2009 Technische Universität Dresden.
00003  * Copyright (C) 2023-2025 Kernkonzept GmbH.
00004  * Author(s): Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include "uart_base.h"
00011
00012 namespace L4 {
00013
00014 class Uart_imx : public Uart
00015 {
00016 public:
00017     enum platform_type
00018     {
00019         Type_imx21,
00020         Type_imx35,
00021         Type_imx51,
00022         Type_imx6,
00023         Type_imx7,
00024         Type_imx8,
00025     };
00026     explicit Uart_imx(enum platform_type type) : _type(type) {}
00027     explicit Uart_imx(enum platform_type type, unsigned /*base_rate*/)
00028         : _type(type) {}
00029     bool startup(Io_register_block const *) override;
00030     void shutdown() override;
00031     bool change_mode(Transfer_mode m, Baud_rate r) override;
00032     bool tx_avail() const;
00033     void wait_tx_done() const;
00034     inline void out_char(char c) const;
00035     int write(char const *s, unsigned long count,
00036              bool blocking = true) const override;
00037
00038 #ifndef UART_WITHOUT_INPUT
00039     bool enable_rx_irq(bool enable = true) override;
00040     int char_avail() const override;
00041     int get_char(bool blocking = true) const override;
00042 #endif
00043 private:
00044     enum platform_type _type;
00045 };
00046
00047 class Uart_imx21 : public Uart_imx
00048 {
00049 public:
00050     Uart_imx21() : Uart_imx(Type_imx21) {}
00051     explicit Uart_imx21(unsigned base_rate) : Uart_imx(Type_imx21, base_rate) {}
00052 };
00053
00054 class Uart_imx35 : public Uart_imx

```

```

00056 {
00057 public:
00058   Uart_imx35() : Uart_imx(Type_imx35) {}
00059   explicit Uart_imx35(unsigned base_rate) : Uart_imx(Type_imx35, base_rate) {}
00060 };
00061
00062 class Uart_imx51 : public Uart_imx
00063 {
00064 public:
00065   Uart_imx51() : Uart_imx(Type_imx51) {}
00066   explicit Uart_imx51(unsigned base_rate) : Uart_imx(Type_imx51, base_rate) {}
00067 };
00068
00069 class Uart_imx6 : public Uart_imx
00070 {
00071 public:
00072   Uart_imx6() : Uart_imx(Type_imx6) {}
00073   explicit Uart_imx6(unsigned base_rate) : Uart_imx(Type_imx6, base_rate) {}
00074
00075 #ifndef UART_WITHOUT_INPUT
00076   void irq_ack() override;
00077 #endif
00078 };
00079
00080 class Uart_imx7 : public Uart_imx
00081 {
00082 public:
00083   Uart_imx7() : Uart_imx(Type_imx7) {}
00084   explicit Uart_imx7(unsigned base_rate) : Uart_imx(Type_imx7, base_rate) {}
00085 };
00086
00087 class Uart_imx8 : public Uart_imx
00088 {
00089 public:
00090   Uart_imx8() : Uart_imx(Type_imx8) {}
00091   explicit Uart_imx8(unsigned base_rate) : Uart_imx(Type_imx8, base_rate) {}
00092 };
00093
00094 } // namespace L4

```

16.27 uart_leon3.h

```

00001 /*
00002  * Copyright (C) 2011 Technische Universität Dresden.
00003  * Copyright (C) 2023-2025 Kernkonzept GmbH.
00004  * Author(s): Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00005  *           Björn Döbel <doebel@os.inf.tu-dresden.de>
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009 #pragma once
00010
00011 #include "uart_base.h"
00012
00013 namespace L4 {
00014
00015 class Uart_leon3 : public Uart
00016 {
00017 public:
00018   explicit Uart_leon3() {}
00019   explicit Uart_leon3(unsigned /*base_rate*/) {}
00020   bool startup(Io_register_block const *) override;
00021   void shutdown() override;
00022   bool change_mode(Transfer_mode m, Baud_rate r) override;
00023   bool tx_avail() const;
00024   void wait_tx_done() const;
00025   inline void out_char(char c) const;
00026   int write(char const *s, unsigned long count,
00027            bool blocking = true) const override;
00028
00029 #ifndef UART_WITHOUT_INPUT
00030   bool enable_rx_irq(bool = true) override;
00031   int char_avail() const override;
00032   int get_char(bool blocking = true) const override;
00033 #endif
00034 };
00035
00036 } // namespace L4

```

16.28 uart_linflex.h

```

00001 /*
00002  * Copyright (C) 2018 Technische Universität Dresden.
00003  * Copyright (C) 2023-2025 Kernkonzept GmbH.
00004  * Author(s): Adam Lackorzynski <adam@l4re.org>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include "uart_base.h"
00011
00012 namespace L4 {
00013
00014 class Uart_linflex : public Uart
00015 {
00016 public:
00017     explicit Uart_linflex(unsigned) {}
00018     bool startup(Io_register_block const *) override;
00019     void shutdown() override;
00020     bool change_mode(Transfer_mode m, Baud_rate r) override;
00021     bool tx_avail() const;
00022     void wait_tx_done() const;
00023     inline void out_char(char c) const;
00024     int write(char const *s, unsigned long count,
00025              bool blocking = true) const override;
00026
00027 #ifndef UART_WITHOUT_INPUT
00028     bool enable_rx_irq(bool enable = true) override;
00029     int char_avail() const override;
00030     int get_char(bool blocking = true) const override;
00031 #endif
00032
00033 private:
00034     void set_uartcr(bool fifo);
00035     bool is_tx_fifo_enabled() const;
00036     bool is_rx_fifo_enabled() const;
00037 };
00038
00039 } // namespace L4

```

16.29 uart_lpuart.h

```

00001 /*
00002  * Copyright (C) 2019, 2023-2025 Kernkonzept GmbH.
00003  * Author(s): Adam Lackorzynski <adam@l4re.org>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include "uart_base.h"
00010
00011 namespace L4 {
00012
00013 class Uart_lpuart : public Uart
00014 {
00015 public:
00016     Uart_lpuart(unsigned freq = 0) : _freq(freq) {}
00017     bool startup(Io_register_block const *) override;
00018     void shutdown() override;
00019     bool change_mode(Transfer_mode m, Baud_rate r) override;
00020     bool tx_avail() const;
00021     void wait_tx_done() const {}
00022     inline void out_char(char c) const;
00023     int write(char const *s, unsigned long count,
00024              bool blocking = true) const override;
00025
00026 #ifndef UART_WITHOUT_INPUT
00027     bool enable_rx_irq(bool enable = true) override;
00028     int char_avail() const override;
00029     int get_char(bool blocking = true) const override;
00030 #endif
00031
00032 private:
00033     unsigned _freq;
00034 };
00035
00036 } // namespace L4

```

16.30 uart_mvebu.h

```

00001 /*
00002  * Copyright (C) 2017, 2023-2025 Kernkonzept GmbH.
00003  * Author(s) Adam Lackorzynski <adam@l4re.org>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include "uart_base.h"
00010
00011 namespace L4 {
00012
00013 class Uart_mvebu : public Uart
00014 {
00015 public:
00016     explicit Uart_mvebu(unsigned baserate) : _baserate(baserate) {}
00017     bool startup(Io_register_block const *) override;
00018     void shutdown() override;
00019     bool change_mode(Transfer_mode m, Baud_rate r) override;
00020     bool tx_avail() const;
00021     void wait_tx_done() const {}
00022     inline void out_char(char c) const;
00023     int write(char const *s, unsigned long count,
00024              bool blocking = true) const override;
00025
00026 #ifndef UART_WITHOUT_INPUT
00027     bool enable_rx_irq(bool enable = true) override;
00028     int char_avail() const override;
00029     int get_char(bool blocking = true) const override;
00030 #endif
00031
00032 private:
00033     unsigned _baserate;
00034 };
00035
00036 } // namespace L4

```

16.31 uart_of.h

```

00001 /*
00002  * Copyright (C) 2009 Technische Universität Dresden.
00003  * Copyright (C) 2023-2025 Kernkonzept GmbH.
00004  * Author(s): Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include "uart_base.h"
00011 #include <stdarg.h>
00012 #include <string.h>
00013 #include <l4/drivers/of.h>
00014
00015 namespace L4 {
00016
00017 class Uart_of : public Uart, public L4_drivers::Of
00018 {
00019 private:
00020     ihandle_t _serial;
00021
00022 public:
00023     Uart_of() : Of(), _serial(0) {}
00024     explicit Uart_of(unsigned /*base_rate*/) : Of(), _serial(0) {}
00025     bool startup(Io_register_block const *) override;
00026     void shutdown() override;
00027     bool change_mode(Transfer_mode m, Baud_rate r) override;
00028     bool tx_avail() const;
00029     void out_char(char c) const;
00030     int write(char const *s, unsigned long count,
00031              bool blocking = true) const override;
00032
00033 #ifndef UART_WITHOUT_INPUT
00034     int char_avail() const override;
00035     int get_char(bool blocking = true) const override;
00036 #endif
00037 };
00038
00039 } // namespace L4

```

16.32 uart_omap35x.h

```

00001 /*
00002  * Copyright (C) 2009 Technische Universität Dresden.
00003  * Copyright (C) 2023-2025 Kernkonzept GmbH.
00004  * Author(s) Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include "uart_base.h"
00011
00012 namespace L4 {
00013
00014 class Uart_omap35x : public Uart
00015 {
00016 public:
00017     explicit Uart_omap35x() {}
00018     explicit Uart_omap35x(unsigned /*base_rate*/) {}
00019     bool startup(Io_register_block const *) override;
00020     void shutdown() override;
00021     bool change_mode(Transfer_mode m, Baud_rate r) override;
00022     bool tx_avail() const;
00023     void wait_tx_done() const;
00024     inline void out_char(char c) const;
00025     int write(char const *s, unsigned long count,
00026              bool blocking = true) const override;
00027
00028 #ifndef UART_WITHOUT_INPUT
00029     bool enable_rx_irq(bool) override;
00030     int char_avail() const override;
00031     int get_char(bool blocking = true) const override;
00032 #endif
00033 };
00034
00035 } // namespace L4

```

16.33 uart_pl011.h

```

00001 /*
00002  * Copyright (C) 2009 Technische Universität Dresden.
00003  * Copyright (C) 2023-2025 Kernkonzept GmbH.
00004  * Author(s) Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include "uart_base.h"
00011
00012 namespace L4 {
00013
00014 class Uart_pl011 : public Uart
00015 {
00016 public:
00017     Uart_pl011(unsigned freq) : _freq(freq) {}
00018     bool startup(Io_register_block const *) override;
00019     void shutdown() override;
00020     bool change_mode(Transfer_mode m, Baud_rate r) override;
00021     bool tx_avail() const;
00022     void wait_tx_done() const;
00023     inline void out_char(char c) const;
00024     int write(char const *s, unsigned long count,
00025              bool blocking = true) const override;
00026
00027 #ifndef UART_WITHOUT_INPUT
00028     bool enable_rx_irq(bool enable) override;
00029     int char_avail() const override;
00030     int get_char(bool blocking = true) const override;
00031 #endif
00032 private:
00033     void set_rate(Baud_rate r);
00034     unsigned _freq;
00035 };
00036
00037 } // namespace L4

```

16.34 uart_s3c2410.h

```

00001 /*
00002  * Copyright (C) 2009-2012 Technische Universität Dresden.
00003  * Copyright (C) 2023-2025 Kernkonzept GmbH.
00004  * Author(s): Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include "uart_base.h"
00011
00012 namespace L4 {
00013
00014 class Uart_s3c : public Uart
00015 {
00016 protected:
00017     enum Uart_type
00018     {
00019         Type_24xx, Type_64xx, Type_s5pv210,
00020     };
00021
00022     Uart_type type() const { return _type; }
00023
00024 public:
00025     explicit Uart_s3c(Uart_type type) : _type(type) {}
00026     explicit Uart_s3c(Uart_type type, unsigned /*base_rate*/) : _type(type) {}
00027     bool startup(Io_register_block const *) override;
00028     void shutdown() override;
00029     bool change_mode(Transfer_mode m, Baud_rate r) override;
00030     bool tx_avail() const;
00031     void wait_tx_done() const;
00032     inline void out_char(char c) const;
00033     int write(char const *s, unsigned long count,
00034              bool blocking = true) const override;
00035     void fifo_reset();
00036
00037 #ifndef UART_WITHOUT_INPUT
00038     int char_avail() const override;
00039     int get_char(bool blocking = true) const override;
00040 #endif
00041
00042 protected:
00043     virtual void ack_rx_irq() const = 0;
00044     virtual void wait_for_empty_tx_fifo() const = 0;
00045     virtual unsigned is_rx_fifo_non_empty() const = 0;
00046     virtual bool is_tx_fifo_not_full() const = 0;
00047
00048 private:
00049     Uart_type _type;
00050 };
00051
00052 class Uart_s3c2410 : public Uart_s3c
00053 {
00054 public:
00055     Uart_s3c2410() : Uart_s3c(Type_24xx) {}
00056     explicit Uart_s3c2410(unsigned base_rate) : Uart_s3c(Type_24xx, base_rate) {}
00057
00058 protected:
00059     void ack_rx_irq() const override {}
00060     void wait_for_empty_tx_fifo() const override;
00061     unsigned is_rx_fifo_non_empty() const override;
00062     bool is_tx_fifo_not_full() const override;
00063
00064     void auto_flow_control(bool on);
00065 };
00066
00067 class Uart_s3c64xx : public Uart_s3c
00068 {
00069 public:
00070     Uart_s3c64xx() : Uart_s3c(Type_64xx) {}
00071     explicit Uart_s3c64xx(unsigned base_rate) : Uart_s3c(Type_64xx, base_rate) {}
00072
00073 protected:
00074     void ack_rx_irq() const override;
00075     void wait_for_empty_tx_fifo() const override;
00076     unsigned is_rx_fifo_non_empty() const override;
00077     bool is_tx_fifo_not_full() const override;
00078 };
00079
00080 class Uart_s5pv210 : public Uart_s3c
00081 {
00082 public:
00083     Uart_s5pv210() : Uart_s3c(Type_s5pv210) {}
00084     explicit Uart_s5pv210(unsigned base_rate) : Uart_s3c(Type_s5pv210, base_rate) {}

```

```

00085
00086 protected:
00087     void ack_rx_irq() const override;
00088     void wait_for_empty_tx_fifo() const override;
00089     unsigned is_rx_fifo_non_empty() const override;
00090     bool is_tx_fifo_not_full() const override;
00091 };
00092
00093 } // namespace L4

```

16.35 uart_sa1000.h

```

00001 /*
00002  * Copyright (C) 2008-2012 Technische Universität Dresden.
00003  * Copyright (C) 2023-2025 Kernkonzept GmbH.
00004  * Author(s): Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00005  *             Alexander Warg <alexander.warg@os.inf.tu-dresden.de>
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009 #pragma once
00010
00011 #include "uart_base.h"
00012
00013 namespace L4 {
00014
00015     class Uart_sa1000 : public Uart
00016     {
00017     public:
00018         explicit Uart_sa1000() {}
00019         explicit Uart_sa1000(unsigned /*base_rate*/) {}
00020         bool startup(Io_register_block const *) override;
00021         void shutdown() override;
00022         bool change_mode(Transfer_mode m, Baud_rate r) override;
00023         bool tx_avail() const;
00024         void wait_tx_done() const;
00025         inline void out_char(char c) const;
00026         int write(char const *s, unsigned long count,
00027                 bool blocking = true) const override;
00028
00029         #ifndef UART_WITHOUT_INPUT
00030             int char_avail() const override;
00031             int get_char(bool blocking = true) const override;
00032         #endif
00033     };
00034
00035 } // namespace L4

```

16.36 uart_sbi.h

```

00001 /*
00002  * Copyright (C) 2021, 2024-2025 Kernkonzept GmbH.
00003  * Author(s): Georg Kotheimer <georg.kotheimer@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include "uart_base.h"
00010
00011 namespace L4 {
00012
00013     class Uart_sbi : public Uart
00014     {
00015     public:
00016         Uart_sbi();
00017         explicit Uart_sbi(unsigned /*base_rate*/) : Uart_sbi() {}
00018         bool startup(Io_register_block const *) override;
00019         void shutdown() override;
00020         bool change_mode(Transfer_mode m, Baud_rate r) override;
00021         bool tx_avail() const { return true; }
00022         void wait_tx_done() const {}
00023         inline void out_char(char c) const;
00024         int write(char const *s, unsigned long count,
00025                 bool blocking = true) const override;
00026
00027         #ifndef UART_WITHOUT_INPUT
00028             int char_avail() const override;

```

```

00029     int get_char(bool blocking = true) const override;
00030 #endif
00031
00032 private:
00033     mutable int _bufchar;
00034 };
00035
00036 } // namespace L4

```

16.37 uart_sh.h

```

00001 /*
00002  * Copyright (C) 2016 Technische Universität Dresden.
00003  * Copyright (C) 2023-2025 Kernkonzept GmbH.
00004  * Author(s): Adam Lackorzynski <adam@l4re.org>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include "uart_base.h"
00011
00012 namespace L4 {
00013
00014     class Uart_sh : public Uart
00015     {
00016     public:
00017         explicit Uart_sh() {}
00018         explicit Uart_sh(unsigned /*base_rate*/) {}
00019         bool startup(Io_register_block const *) override;
00020         void shutdown() override;
00021         bool change_mode(Transfer_mode m, Baud_rate r) override;
00022         bool tx_avail() const;
00023         void wait_tx_done() const {}
00024         inline void out_char(char c) const;
00025         int write(char const *s, unsigned long count,
00026                 bool blocking = true) const override;
00027
00028     #ifndef UART_WITHOUT_INPUT
00029         bool enable_rx_irq(bool enable = true) override;
00030         void irq_ack() override;
00031         int char_avail() const override;
00032         int get_char(bool blocking = true) const override;
00033     #endif
00034     };
00035
00036 } // namespace L4

```

16.38 uart_sifive.h

```

00001 /*
00002  * Copyright (C) 2021-2025 Kernkonzept GmbH.
00003  * Author(s): Georg Kotheimer <georg.kotheimer@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include "uart_base.h"
00010
00011 namespace L4 {
00012
00013     class Uart_sifive : public Uart
00014     {
00015     public:
00016         Uart_sifive(unsigned freq) : _freq(freq), _bufchar(-1) {}
00017         bool startup(Io_register_block const *) override;
00018         void shutdown() override;
00019         bool change_mode(Transfer_mode m, Baud_rate r) override;
00020         bool tx_avail() const;
00021         void wait_tx_done() const;
00022         inline void out_char(char c) const;
00023         int write(char const *s, unsigned long count,
00024                 bool blocking = true) const override;
00025
00026     #ifndef UART_WITHOUT_INPUT
00027         bool enable_rx_irq(bool enable) override;
00028         int char_avail() const override;

```



```

00029 int get_char(bool blocking = true) const override;
00030 #endif
00031
00032 private:
00033     unsigned _freq;
00034     mutable int _bufchar;
00035 };
00036
00037 } // namespace L4

```

16.39 uart_tegra-tcu.h

```

00001 /*
00002  * Copyright (C) 2025 Kernkonzept GmbH.
00003  * Author(s): Adam Lackorzynski <adam@l4re.org>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include "uart_base.h"
00010
00011 namespace L4 {
00012
00013     class Uart_tegra_tcu : public Uart
00014     {
00015     public:
00016         Uart_tegra_tcu() {}
00017         Uart_tegra_tcu(unsigned long) {}
00018         bool startup(Io_register_block const *tx_mbox) override;
00019         void set_rx_mbox(Io_register_block const *rx_mbox);
00020         void shutdown() override;
00021         bool change_mode(Transfer_mode m, Baud_rate r) override;
00022         bool tx_avail() const;
00023         void wait_tx_done() const {}
00024         inline void out_char(char c) const;
00025         int write(char const *s, unsigned long count,
00026                 bool blocking = true) const override;
00027
00028         #ifndef UART_WITHOUT_INPUT
00029             int char_avail() const override;
00030             int get_char(bool blocking = true) const override;
00031         #endif
00032
00033     private:
00034         mutable unsigned _current_rx_val = 0u;
00035         Io_register_block const *_rx_mbox = nullptr;
00036     };
00037
00038 } // namespace L4

```

16.40 tutorial.lua

```

00001 local _doc = [=[
00002
00003 Tutorial lua script for Ned
00004 =====
00005
00006 Firstly we have a set of definitions available. Some come from 'ned.lua'
00007 embedded script and others from the C++ bindings within Ned. The whole L4
00008 functionality is in the lua module "L4" (use 'local L4 = require("L4");').
00009 The L4 module classes and functions cope with L4 capabilities and
00010 their invocations, provide a set of constants and access to the L4Re environment of
00011 the running program. Finally, of course it can also start L4 applications.
00012
00013 L4 Capabilities
00014 =====
00015
00016 The L4 module defines a user data type for capabilities. A capability in lua
00017 carries a typed L4 capability and is accompanied with a set of type specific
00018 methods that may be called on the object behind the capability. There also
00019 exists a way to cast a capability to a capability to a different type of
00020 object using L4.cast(type, cap).
00021
00022 L4.cast(type, cap)
00023
00024 Returns a cap transformed to a capability of the given type, whereas type
00025 is either the fully qualified C++ name of the class encapsulating the object

```

```

00026 or the L4 protocol ID assigned to all L4Re and L4 system objects.
00027 If the type is unknown then nil is returned.
00028
00029 Generic capabilities provide the following methods:
00030
00031 is_valid()
00032
00033 Returns true if the capability is not the invalid cap (L4_INVALID_CAP), and
00034 false if it is the invalid cap.
00035
00036
00037 L4Re::Namespace object
00038 =====
00039
00040 There is a lua type for name spaces that has the following methods:
00041
00042 query(name), or q(name)
00043
00044 Returns a closure (function) that initiates a query for the given name
00045 within the name space. The function takes no arguments and returns
00046 a capability if successful or nil if name is not found.
00047
00048
00049 link(name), or l(name)
00050
00051 Returns a function that can create a link to the given object in the name
00052 space if put into another name space. The function takes two parameters
00053 the target name space and the name in the target name space.
00054 Loader:create_namespace and Loader.fill_namespace calls this function
00055 when things are really put into an L4Re::Namespace.
00056
00057
00058 register(name, cap), or r(name, cap)
00059
00060 Registers the given object capability under the given name. cap can
00061 be either a real capability (note query returns a function), a string, or
00062 nil. If it is a capability it is just put into the name space.
00063 In the case cap is a string a placeholder will be put into the name space
00064 that will be replaced with a real capability later by some program.
00065 And if nil is use the name will be deleted from the name space.
00066
00067
00068 L4::Factory object
00069 =====
00070
00071 The factory object provides an interface to the generic create method of a
00072 factory.
00073
00074 create(proto, ...)
00075
00076 This method calls the factory to create an object of the given type,
00077 via the L4 protocol number (see L4.Proto table for more) all further
00078 arguments are passed to the factory.
00079
00080
00081 Access to the L4Re Env capabilities
00082 =====
00083
00084 The L4 module defines a table L4.Env that contains the capabilities
00085 of the L4Re::Env::env() environment. Namely:
00086
00087 factory      The kernel factory of Ned
00088 log          The log object of Ned
00089 user_factory The factory provided to Ned, including memory
00090 parent       The parent of Ned
00091 rm           The region map of Ned
00092 scheduler    The scheduler of Ned
00093
00094
00095 Some useful constants
00096 =====
00097
00098 L4.Proto table contains the most important protocol values for
00099 L4 and L4Re objects.
00100
00101 Namespace
00102 Goos
00103 Rm
00104 Irq
00105 Sigma0
00106 Factory
00107 Log
00108 Scheduler
00109
00110 The L4.Info table contains the following functions:
00111
00112 Kip.str()    The banner string found in the kernel info page

```

```

00113     arch()          Architecture name, such as: x86, amd64, arm, ppc32
00114     platform()      Platform name, such as: pc, ux, realview, beagleboard
00115     mword_bits()    Number of native machine word bits (32, 64)
00116
00117
00118 Support for starting L4 programs
00119 =====
00120
00121 The L4 module defines two classes that are useful for starting L4 applications.
00122 The class L4.Loader that encapsulates a fairly high level policy
00123 that is useful for starting a whole set of processes. And the class L4.App_env
00124 that encapsulates a more fine-grained policy.
00125
00126 L4.Loader
00127 -----
00128
00129 The class L4.Loader encapsulates the policy for starting programs with the
00130 basic building blocks for the application coming from a dedicated loader,
00131 such as Moe or a Loader instance. These building blocks are a region map (Rm),
00132 a scheduler, a memory allocator, and a logging facility.
00133 A L4.Loader object is typically used to start multiple applications. There
00134 is a L4.default_loader instance of L4.Loader that uses the L4.Env.mem_alloc
00135 factory of the current Ned instance to create the objects for a new program.
00136 However you may also use a more restricted factory for applications and
00137 instantiate a loader for them. The L4.Loader objects can already be used
00138 to start a program with L4.Loader:start(app_spec, cmd, ...). Where app_spec
00139 is a table containing parameters for the new application. cmd is the
00140 command to run and the remaining arguments are the command-line options for
00141 the application.
00142
00143 ]==]
00144
00145 L4.default_loader:start({}, "rom/hello");
00146
00147 local _doc = [=[
00148
00149 This statement does the following:
00150 1. Create a new environment for the application
00151 2. Add the rom name-space into the new environment (thus shares Ned's
00152    'rom' directory with the new program).
00153 3. Creates all the building blocks for the new process and starts the
00154    'l4re' support kernel in the new process which in turn starts 'rom/hello'
00155    in the new process.
00156
00157 Using the app_spec parameter you can modify the behavior in two ways. There are
00158 two supported options 'caps' for providing more capabilities for the
00159 application. And 'log' for modifying the logger tag and color.
00160
00161 ]==]
00162
00163 local my_caps = {
00164     fb = L4.Env.vesa;
00165 };
00166
00167 L4.default_loader:start({caps = my_caps, log = {"APP", "blue"}}, "rom/hello");
00168
00169 local _doc = [=[
00170
00171 This snippet creates a caps template (my_caps) and uses it for the
00172 new process and also sets user-defined log tags. The L4.Loader:start method,
00173 however, automatically adds the 'rom' directory to the caps environment if
00174 not already specified in the template.
00175
00176 Environment variables may be given as a table in the third argument to
00177 start. Argument to the program are given space separated after the program
00178 name within a single string.
00179
00180 ]==]
00181
00182 L4.default_loader:start({}, "rom/program arg1 " .. arg2, { LD_DEBUG = 1 });
00183
00184 local _doc = [=[
00185
00186 L4.default_loader:startv is a variant of the start function that takes the
00187 arguments of the program as a single argument each. If the last argument to
00188 startv is a table it is interpreted as environment variables for the program.
00189 The above example would translate to:
00190
00191 ]==]
00192
00193 L4.default_loader:startv({}, "rom/program", "arg1", arg2, { LD_DEBUG = 1 });
00194
00195 local _doc = [=[
00196
00197 To create a new L4.Loader instance you may use a generic factory for all
00198 building blocks or set individual factories.
00199

```

```

00200 ]==]
00201
00202 l = L4.Loader.new({
00203     mem = L4.Env.user_factory:create(L4.Proto.Factory, 512*1024):m("rs")
00204 });
00205
00206 local _doc = [=[
00207
00208 Creates a loader instance that uses the newly created 512 KB factory for
00209 all building blocks. To set individual factories use the options:
00210     'mem'          as memory allocator for the new processes and as
00211                   default factory for all objects not explicitly set to a
00212                   different factory
00213     'log_fab'      for creating log objects.
00214     'ns_fab'       for creating name-space objects.
00215     'rm_fab'       for creating region-map objects.
00216
00217
00218
00219 L4.App_env
00220 -----
00221
00222 L4.App_env provides a more fine-grained control for a single process or for a
00223 limited number of processes. L4.App_env uses an L4.Loader object as basic
00224 facility. However you can override the memory allocator 'mem' for for the new
00225 process as well as the kernel factory 'factory', the log capability etc.
00226
00227 ]==]
00228
00229 local e = L4.App_env.new({
00230     loader = l,
00231     mem = L4.Env.user_factory:create(L4.Proto.Factory, 128*1024):m("rs")
00232 });
00233
00234 e:start("rom/hello");

```

16.41 cmd_control

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00003 /*
00004  * Copyright (C) 2016, 2024 Kernkonzept GmbH.
00005  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009 #pragma once
00010
00011 #include <l4/sys/cxx/ipc_epiface>
00012 #include <l4/sys/cxx/ipc_string>
00013
00014 namespace L4Re { namespace Ned {
00015
00019 class Cmd_control : public L4::Kobject_0t<Cmd_control>
00020 {
00021     L4_INLINE_RPC_NF(long, execute, (L4::Ipc::String<> cmd,
00022                                     L4::Ipc::Array<char> &result));
00023
00024 public:
00042     long execute(L4::Ipc::String<> cmd) noexcept
00043     {
00044         L4::Ipc::Array<char> res(0, NULL);
00045         return execute_t::call(c(), cmd, res);
00046     }
00047
00064     long execute(L4::Ipc::String<> cmd,
00065                 L4::Ipc::String<char> *result) noexcept
00066     {
00067         L4::Ipc::Array<char> res(result->length, result->data);
00068         long r = execute_t::call(c(), cmd, res);
00069         if (r >= 0)
00070             result->length = res.length;
00071         return r;
00072     }
00073
00074     typedef L4::Typeid::Rpc<execute_t> Rpc;
00075 };
00076
00077 } } // namespace

```

16.42 pkg/uvmm/configs/vmm.lua File Reference

Functions

- [new_sched](#) (prio, cpu_mask, ...)
- [start_io](#) (busses, cmdline, opts)
- [set_sched](#) (opts, prio, cpus, ...)
- [start_virtio_switch](#) (ports, prio, cpus, switch_type, ext_caps)
- [start_virtio_switch_tbl](#) (options)
- [start_vm](#) (options)

16.42.1 Function Documentation

16.42.1.1 new_sched()

```
new_sched (
    prio,
    cpu_mask,
    ... )
```

Creates a new scheduler proxy at moe.

Parameters

| | |
|-----------------|---|
| <i>prio</i> | Base priority of the threads running in the scheduler proxy |
| <i>cpu_mask</i> | First of a list of CPU masks for the first 64 CPUs to use for the scheduler proxy |
| ... | more CPU masks |

Returns

created scheduler

16.42.1.2 set_sched()

```
set_sched (
    opts,
    prio,
    cpus,
    ... )
```

Create scheduler proxy and add it into the `opts` table under the key `scheduler`.

Parameters

| | | |
|----------------|-------------|--|
| <i>in, out</i> | <i>opts</i> | option table |
| | <i>prio</i> | thread priority (or <code>nil</code>) |

| | | |
|--|-------------|-------------------|
| | <i>cpus</i> | cpu mask (or nil) |
| | ... | more CPU masks |

There are four possibilities for values of prio and cpus:

- No prio and no cpus: No scheduler proxy created.
- A prio, but no cpus: Create a scheduler proxy with only a priority limit.
- No Prio, but cpus: Create a scheduler proxy with default prio and cpus limit.
- A prio and cpus: Create a scheduler proxy with given limits.

16.42.1.3 start_io()

```
start_io (
    busses,
    cmdline,
    opts)
```

Start IO service with the given options

Parameters

| | | |
|----------------|----------------|---|
| <i>in, out</i> | <i>busses</i> | table of vBus names as keys. Uses io config <name> . vbus to fill vBus <name>. |
| | <i>cmdline</i> | io command line parameters |
| | <i>opts</i> | Option table for loader.start function, e.g. scheduler or ext_caps. Entries from ext_caps have precedence over default caps created by this function. |

After this function returns the created vBusses are located in the table passed as *busses*.

16.42.1.4 start_virtio_switch()

```
start_virtio_switch (
    ports,
    prio,
    cpus,
    switch_type,
    ext_caps)
```

Start virtio network application.

Deprecated This function exists for backwards compatibility reasons and calls [start_virtio_switch_tbl](#) with an appropriate *options* table

Parameters

| | | |
|----------------------|--------------------------|--|
| <code>in, out</code> | <code>ports</code> | table with port names as keys |
| | <code>prio</code> | priority for started thread |
| | <code>cpus</code> | cpu mask for started thread |
| | <code>switch_type</code> | Selects application to start. Either <code>switch</code> or <code>p2p</code> |
| | <code>ext_caps</code> | Extra capabilities to pass to the started application |

The `switch_type` `switch` can take additional arguments to create a port at the switch. To pass these arguments for a specific port, pass a table as value for a key in the `ports` table.

16.42.1.5 start_virtio_switch_tbl()

```
start_virtio_switch_tbl (
    options)
```

Start virtio network application.

Parameters

| | |
|----------------------|-----------------------|
| <code>options</code> | A table of parameters |
|----------------------|-----------------------|

The following keys are supported in the `options` table:

| table key | value |
|--------------------------|--|
| <code>ports</code> | table with port names as keys |
| <code>scheduler</code> | scheduler (e.g. created with <code>new_sched</code>) |
| <code>switch_type</code> | selects application to start. Either <code>switch</code> or <code>p2p</code> |
| <code>ext_caps</code> | Extra capabilities to pass to the started application |
| <code>svr_cap</code> | cap slot to be used for the server interface |
| <code>port_limit</code> | the maximum number of dynamic ports the switch shall support |

The `switch_type` `switch` can take additional arguments to create a port at the switch. To pass these arguments for a specific port, pass a table as value for a key in the `ports` table.

Note

The `svr_cap` capability requires server rights, use `":svr()"`.

16.42.1.6 start_vm()

```
start_vm (
    options)
```

Start UVMM

Parameters

| | |
|----------------|-----------------------|
| <i>options</i> | A table of parameters |
|----------------|-----------------------|

The following keys are supported in the `options` table:

| table key | value |
|------------------------|--|
| <code>bootargs</code> | command line for guest kernel |
| <code>cpus</code> | cpu mask |
| <code>ext_args</code> | additional arguments to pass to UVMM |
| <code>fdt</code> | file name of the device tree |
| <code>id</code> | an integer identifying the VM |
| <code>jdb</code> | jdb capability |
| <code>kernel</code> | file name of the guest kernel binary |
| <code>mem</code> | RAM size in MiB <i>or</i> dataspace cap for guest memory. |
| <code>mem_align</code> | alignment for the guest memory in bits. Ignored if <code>mem</code> is a cap. |
| <code>mon</code> | monitor application file name |
| <code>net</code> | a virtio cap, e.g. for network |
| <code>prio</code> | thread priority |
| <code>ram_base</code> | start of guest memory |
| <code>rd</code> | file name of the ramdisk |
| <code>scheduler</code> | a scheduler cap. If used, <code>prio</code> and <code>cpus</code> are ignored. |
| <code>vbus</code> | the vBus to attach to the VM |

16.43 vmm.lua

[Go to the documentation of this file.](#)

```
00001 --! \file vmm.lua
00002
00003 local L4 = require "L4";
00004
00005 local l = L4.Loader.new({mem = L4.Env.user_factory});
00006 loader = l;
00007
00008 --[[!
00009 \internal
00010
00011 Utility function to merge several lua tables
00012
00013 \param ... one or more tables
00014
00015 \note Later tables are given more priority when merging
00016
00017 \return a new combined table
00018 ]]
00019 function table_override(...)
00020     local combined = {}
00021     for _, tab in ipairs({...}) do
```



```

00022     for k, v in pairs(tab) do
00023         combined[k] = v
00024     end
00025 end
00026 return combined
00027 end
00028
00029 --[[!
00030  Creates a new scheduler proxy at moe.
00031
00032  \param prio      Base priority of the threads running in the scheduler proxy
00033  \param cpu_mask  First of a list of CPU masks for the first 64 CPUs to use for
00034                    the scheduler proxy
00035  \param ...       more CPU masks
00036
00037  \return created scheduler
00038 ]]
00039 function new_sched(prio, cpu_mask, ...)
00040     return L4.Env.user_factory:create(L4.Proto.Scheduler, prio + 10, prio,
00041                                       cpu_mask, ...);
00042 end
00043
00044 --[[!
00045  Start IO service with the given options
00046
00047  \param[in,out] busses  table of vBus names as keys. Uses io config
00048                        `<name>.vbus` to fill vBus `<name>`.
00049  \param          cmdline io command line parameters
00050  \param          opts    Option table for loader.start function, e.g. scheduler
00051                        or ext_caps. Entries from ext_caps have precedence over
00052                        default caps created by this function.
00053
00054  After this function returns the created vBusses are located in the table
00055  passed as `busses`.
00056 ]]
00057 function start_io(busses, cmdline, opts)
00058     if opts == nil then opts = {} end
00059
00060     if opts.caps ~= nil then
00061         print("Warning: use opts.ext_caps to pass custom/additional capabilities.")
00062     end
00063
00064     if opts.scheduler == nil then
00065         print("IO started with base priority. Risk of priority related deadlocks! "
00066               .. "Provide an opts.scheduler entry.")
00067     end
00068
00069     local caps = {
00070         sigma0 = L4.cast(L4.Proto.Factory, L4.Env.sigma0):create(L4.Proto.Sigma0);
00071         icu     = L4.Env.icu;
00072         iommu   = L4.Env.iommu;
00073     };
00074
00075     local files = "";
00076
00077     for k, v in pairs(busses) do
00078         if caps[k] ~= nil then
00079             print("Warning: overwriting caps." .. k .. " with vbus of same name.")
00080         end
00081         local c = l:new_channel();
00082         busses[k] = c
00083         caps[k] = c:svr();
00084         files = files .. " rom/" .. k .. ".vbus";
00085     end
00086
00087     opts.caps = table_override(caps, opts.caps or {}, opts.ext_caps or {})
00088     opts.log  = opts.log or { "io", "red" }
00089
00090     return l:start(opts, "rom/io " .. cmdline .. files)
00091 end
00092
00093 --[[!
00094  Create scheduler proxy and add it into the `opts` table under
00095  the key `scheduler`.
00096
00097  \param[in,out] opts  option table
00098  \param          prio  thread priority (or `nil`)
00099  \param          cpus  cpu mask (or `nil`)
00100  \param          ...   more CPU masks
00101
00102  There are four possibilities for values of prio and cpus:
00103
00104  \li No prio and no cpus: No scheduler proxy created.
00105  \li A prio, but no cpus: Create a scheduler proxy with only a priority limit.
00106  \li No Prio, but cpus: Create a scheduler proxy with default prio and cpus
00107      limit.
00108  \li A prio and cpus: Create a scheduler proxy with given limits.

```

```

00109 ]]
00110 function set_sched(opts, prio, cpus, ...)
00111   if cpus == nil and prio == nil then
00112     return
00113   end
00114
00115   if prio == nil then
00116     -- Default to zero to use the L4Re Default_thread_prio
00117     prio = 0
00118   end
00119
00120   local sched = new_sched(prio, cpus, ...);
00121   opts["scheduler"] = sched;
00122 end
00123
00124 --[[!
00125   Start virtio network application.
00126
00127   \deprecated This function exists for backwards compatibility reasons and calls
00128     \ref start_virtio_switch_tbl with an appropriate `options` table
00129
00130   \param[in,out] ports      table with port names as keys
00131   \param          prio      priority for started thread
00132   \param          cpus      cpu mask for started thread
00133   \param          switch_type Selects application to start. Either `switch` or `p2p`
00134   \param          ext_caps   Extra capabilities to pass to the started application
00135
00136   The switch_type `switch` can take additional arguments to create a port at the
00137   switch. To pass these arguments for a specific port, pass a table as value for
00138   a key in the ports table.
00139 ]]
00140 ]]
00141 function start_virtio_switch(ports, prio, cpus, switch_type, ext_caps)
00142   local opts = {
00143     ports = ports,
00144     switch_type = switch_type,
00145     ext_caps = ext_caps,
00146   }
00147   set_sched(opts, prio, cpus)
00148   return start_virtio_switch_tbl(opts)
00149 end
00150
00151 --[[!
00152   Start virtio network application.
00153
00154   \param options  A table of parameters
00155
00156   The following keys are supported in the `options` table:
00157
00158   | table key      | value
00159   | ----- | -----
00160   | `ports`       | table with port names as keys
00161   | `scheduler`   | scheduler (e.g. created with new_sched)
00162   | `switch_type` | selects application to start. Either `switch` or `p2p`
00163   | `ext_caps`    | Extra capabilities to pass to the started application
00164   | `svr_cap`     | cap slot to be used for the server interface
00165   | `port_limit`  | the maximum number of dynamic ports the switch shall support
00166
00167   The switch_type `switch` can take additional arguments to create a port at the
00168   switch. To pass these arguments for a specific port, pass a table as value for
00169   a key in the ports table.
00170
00171   \note The `svr_cap` capability requires server rights, use ":svr()".
00172 ]]
00173 function start_virtio_switch_tbl(options)
00174   local ports = options.ports;
00175   local scheduler = options.scheduler;
00176   local switch_type = options.switch_type;
00177   local ext_caps = options.ext_caps;
00178   local svr_cap = options.svr_cap;
00179   local port_limit = options.port_limit;
00180
00181   if svr_cap and port_limit == nil then
00182     print("Warning: start_virtio_switch_tbl(): 'svr_cap' defined, but no '..
00183           "'port_limit' set. The svr_cap will not support dynamic port '..
00184           'creation.")
00185   end
00186
00187   if port_limit and svr_cap == nil then
00188     error("start_virtio_switch_tbl(): 'port_limit' set, but no 'svr_cap'. '..
00189           'This is not supported")
00190   end
00191
00192   local switch
00193
00194   if svr_cap then
00195     switch = svr_cap:svr()

```

```

00196     else
00197         switch = l:new_channel()
00198     end
00199
00200     local opts = {
00201         log = { "switch", "Blue" },
00202         caps = table_override({ svr = switch:svr() }, ext_caps or {});
00203     };
00204
00205     if scheduler then
00206         opts["scheduler"] = scheduler;
00207     end
00208
00209     if switch_type == "switch" then
00210         local port_count = 0;
00211         for k, v in pairs(ports) do
00212             port_count = port_count + 1;
00213         end
00214         if port_limit then
00215             port_count = port_count + port_limit
00216         end
00217
00218         svr = l:start(opts, "rom/l4vio_switch -v -p " .. port_count );
00219
00220         for k, extra_opts in pairs(ports) do
00221             if type(extra_opts) ~= "table" then
00222                 extra_opts = {}
00223             end
00224
00225             ports[k] = L4.cast(L4.Proto.Factory, switch):create(
00226                 0,
00227                 "ds-max=4",
00228                 "name=" .. k,
00229                 table.unpack(extra_opts)
00230             )
00231         end
00232     else
00233         svr = l:start(opts, "rom/l4vio_net_p2p");
00234
00235         for k, v in pairs(ports) do
00236             ports[k] = L4.cast(L4.Proto.Factory, switch):create(0, "ds-max=4");
00237         end
00238     end
00239
00240     return svr;
00241 end
00242
00243 --[[!
00244 Start UVMM
00245
00246 \param options  A table of parameters
00247
00248 The following keys are supported in the `options` table:
00249
00250 | table key | value |
00251 | ----- | ----- |
00252 | `bootargs` | command line for guest kernel |
00253 | `cpus` | cpu mask |
00254 | `ext_args` | additional arguments to pass to UVMM |
00255 | `fdt` | file name of the device tree |
00256 | `id` | an integer identifying the VM |
00257 | `jdb` | jdb capability |
00258 | `kernel` | file name of the guest kernel binary |
00259 | `mem` | RAM size in MiB \e or dataspace cap for guest memory. |
00260 | `mem_align` | alignment for the guest memory in bits. Ignored if mem is a cap. |
00261 | `mon` | monitor application file name |
00262 | `net` | a virtio cap, e.g. for network |
00263 | `prio` | thread priority |
00264 | `ram_base` | start of guest memory |
00265 | `rd` | file name of the ramdisk |
00266 | `scheduler` | a scheduler cap. If used, prio and cpus are ignored. |
00267 | `vbus` | the vBus to attach to the VM |
00268 ]]
00269 function start_vm(options)
00270     local nr = options.id;
00271     local size_mb = 0;
00272     local vbus = options.vbus;
00273     local vnet = options.net;
00274     local prio = options.prio;
00275     local cpus = options.cpus;
00276     local scheduler = options.scheduler;
00277     local nonidentmem = options.nonidentmem;
00278
00279     local align = 10;
00280     if L4.Info.arch() == "arm" then
00281         align = 28;
00282     elseif L4.Info.arch() == "arm64" then

```

```

00283     align = 21;
00284 end
00285 align = options.mem_align or align;
00286
00287 local cmdline = {};
00288 if options.fdt then
00289     if type(options.fdt) ~= "table" then
00290         options.fdt = { options.fdt }
00291     end
00292     for _,v in ipairs(options.fdt) do
00293         cmdline[#cmdline+1] = "-d" .. v;
00294     end
00295 end
00296
00297 if options.bootargs then
00298     cmdline[#cmdline+1] = "-c" .. options.bootargs;
00299 end
00300
00301 if options.rd then
00302     cmdline[#cmdline+1] = "-r" .. options.rd;
00303 end
00304
00305 if options.kernel then
00306     cmdline[#cmdline+1] = "-k" .. options.kernel;
00307 end
00308
00309 if options.ram_base then
00310     cmdline[#cmdline+1] = "-b" .. options.ram_base;
00311 end
00312
00313 if L4.Info.arch() == "arm" or L4.Info.arch() == "arm64" then
00314     if not options.nonidentmem then
00315         cmdline[#cmdline+1] = "-i";
00316     end
00317 end
00318
00319 local keyb_shortcut = nil;
00320 if nr ~= nil then
00321     keyb_shortcut = "key=" .. nr;
00322 end
00323
00324 local vm_ram;
00325 if type(options.mem) == "userdata" then
00326     -- User gave us a cap. Using this as dataspace for guest RAM.
00327     vm_ram = options.mem
00328 elseif type(options.mem) == "number" then
00329     -- User gave us a number. Using this as size for a new Dataspace.
00330     size_mb = options.mem
00331 elseif type(options.mem) == "string" then
00332     print("start_vm: mem parameter '" .. options.mem .. "' is of type string, "
00333           .. "please use integer.");
00334     size_mb = tonumber(options.mem)
00335 else
00336     -- User did not give us any valid value.
00337     size_mb = 16
00338 end
00339
00340 if size_mb > 0 then
00341     local mem_flags = L4.Mem_alloc_flags.Continuous
00342                     | L4.Mem_alloc_flags.Pinned
00343                     | L4.Mem_alloc_flags.Super_pages;
00344
00345     vm_ram = L4.Env.user_factory:create(L4.Proto.Dataspace,
00346                                         size_mb * 1024 * 1024,
00347                                         mem_flags, align):m("rw");
00348 end
00349
00350 local caps = {
00351     net = vnet;
00352     vbus = vbus;
00353     ram = vm_ram;
00354 };
00355
00356 if options.jdb then
00357     caps["jdb"] = L4.Env.jdb
00358 end
00359
00360 if options.ext_args then
00361     for _,v in ipairs(options.ext_args) do
00362         cmdline[#cmdline+1] = v
00363     end
00364 end
00365
00366 local opts = {
00367     log = options.log or l.log_fab:create(L4.Proto.Log, "vm" .. nr, "w",
00368                                           keyb_shortcut);
00369     caps = table_override(caps, options.ext_caps or {});

```

```

00370     };
00371
00372     if scheduler then
00373         opts["scheduler"] = scheduler;
00374     else
00375         set_sched(opts, prio, cpus);
00376     end
00377
00378     if type(options.mon) == 'string' then
00379         -- assume 'mon' is the name of a server binary which implements the uvmm
00380         -- CLI interface
00381         mon = l:new_channel()
00382
00383         l:start({
00384             scheduler = opts.scheduler;
00385             log = l.log_fab:create(L4.Proto.Log, "mon" .. nr),
00386             caps = { mon = mon:svr() }
00387         }, "rom/" .. options.mon)
00388
00389         opts.caps["mon"] = mon
00390     elseif options.mon ~= false then
00391         opts.caps["mon"] = l.log_fab:create(L4.Proto.Log, "mon" .. nr, "g");
00392     end
00393
00394     return l:startv(opts, "rom/uvmm", table.unpack(cmdline));
00395 end
00396
00397 return _ENV

```

16.44 debug.h

```

00001 /*
00002  * (c) 2013-2014 Alexander Warg <warg@os.inf.tu-dresden.de>
00003  *     economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include <l4/re/util/debug>
00010
00011 struct Err : L4Re::Util::Err
00012 {
00013     Err(Level l = Fatal) : L4Re::Util::Err(l, "VSwitch") {}
00014 };
00015
00016 class Dbg : public L4Re::Util::Dbg
00017 {
00018     enum
00019     {
00020         Verbosity_shift = 4,
00021         Verbosity_mask = (1UL « Verbosity_shift) - 1
00022     };
00023
00024 public:
00025     enum Verbosity : unsigned long
00026     {
00027         Quiet = 0,
00028         Warn = 1,
00029         Info = 2,
00030         Debug = 4,
00031         Trace = 8,
00032         Max_verbosity = 8
00033     };
00034
00035     enum Component
00036     {
00037         Core = 0,
00038         Virtio,
00039         Port,
00040         Request,
00041         Queue,
00042         Packet,
00043         Max_component
00044     };
00045
00046 #ifndef NDEBUG
00047     static_assert(Max_component * Verbosity_shift <= sizeof(level) * 8,
00048         "Too many components for level mask");
00049     static_assert((Max_verbosity & Verbosity_mask) == Max_verbosity,
00050         "Verbosity_shift to small for verbosity levels");
00051 #endif

```

```

00062 static void set_verbosity(unsigned mask)
00063 {
00064     for (unsigned i = 0; i < Max_component; ++i)
00065         set_verbosity(i, mask);
00066 }
00067
00074 static void set_verbosity(unsigned c, unsigned mask)
00075 {
00076     level &= ~(Verbosity_mask < (Verbosity_shift * c));
00077     level |= (mask & Verbosity_mask) < (Verbosity_shift * c);
00078 }
00079
00089 static bool is_active(unsigned c, unsigned mask)
00090 { return level & (mask & Verbosity_mask) < (Verbosity_shift * c); }
00091
00098 using L4Re::Util::Dbg::is_active;
00099
00100 Dbg(Component c = Core, Verbosity v = Warn, char const *subsys = nullptr)
00101 : L4Re::Util::Dbg(v < (Verbosity_shift * c), "SWI", subsys)
00102 {}
00103
00104 #else
00105
00106 static void set_verbosity(unsigned) {}
00107 static void set_verbosity(unsigned, unsigned) {}
00108
00109 static bool is_active(unsigned, unsigned) { return false; }
00110 using L4Re::Util::Dbg::is_active;
00111
00112 Dbg(Component c = Core, Verbosity v = Warn, char const *subsys = nullptr)
00113 : L4Re::Util::Dbg(v < (Verbosity_shift * c), "", subsys)
00114 {}
00115
00116 #endif
00117 };

```

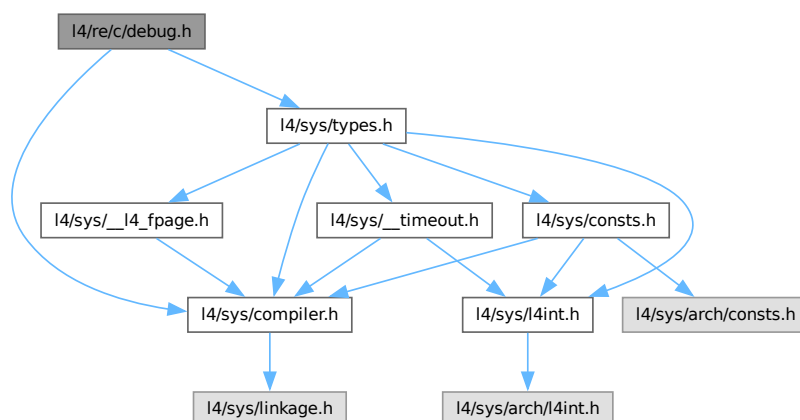
16.45 l4/re/c/debug.h File Reference

Debug C interface.

```
#include <l4/sys/compiler.h>
```

```
#include <l4/sys/types.h>
```

Include dependency graph for debug.h:



Functions

- [L4_BEGIN_DECLS](#) [l4_re_t](#) [l4re_debug_obj_debug](#) ([l4_cap_idx_t](#) srv, unsigned long function) [L4_NOTHROW](#)
Call debug function of [L4Re](#) service.

16.45.1 Detailed Description

Debug C interface.

Definition in file [debug.h](#).

16.46 debug.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *      economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011 #pragma once
00012
00017
00018 #include <l4/sys/compiler.h>
00019 #include <l4/sys/types.h>
00020
00021 L4_BEGIN_DECLS
00022
00030 L4_CV l4_ret_t
00031 l4re_debug_obj_debug(l4_cap_idx_t srv, unsigned long function) L4_NOTHROW;
00032
00033 L4_END_DECLS

```

16.47 filter.cc

```

00001 /*
00002  * Copyright (C) 2016-2017, 2024 Kernkonzept GmbH.
00003  * Author(s): Jean Wolter <jean.wolter@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #include "filter.h"
00008
00009 /* This is an example filter and therefore rather verbose. A real
00010    filter would not produce any output */
00011
00012 bool
00013 filter(const uint8_t *buf, size_t size)
00014 {
00015     // Packet large enough to apply filter condition?
00016     if (size <= 13)
00017         return false;
00018
00019     uint16_t ether_type = (uint16_t)*(buf + 12) << 8
00020                          | (uint16_t)*(buf + 13);
00021     char const *protocol;
00022     switch (ether_type)
00023     {
00024         case 0x0800: protocol = "IPv4"; break;
00025         case 0x0806: protocol = "ARP"; break;
00026         case 0x8100: protocol = "Vlan"; break;
00027         case 0x86dd: protocol = "IPv6"; break;
00028         case 0x8863: protocol = "PPPoE Discovery"; break;
00029         case 0x8864: protocol = "PPPoE Session"; break;
00030         default: protocol = nullptr;
00031     }
00032     if (protocol)
00033         printf("%s\n", protocol);
00034     else
00035         printf("%04x\n", ether_type);
00036
00037     if (ether_type == 0x0806)
00038     {
00039         printf("Do not filter arp\n");
00040         return false;
00041     }
00042
00043     return true;
00044 }

```

16.48 filter.h

```

00001 /*
00002  * Copyright (C) 2016-2017, 2023-2025 Kernkonzept GmbH.
00003  * Author(s): Jean Wolter <jean.wolter@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include "request.h"
00010 #include <l4/bid_config.h>
00011
00024 #ifdef CONFIG_VNS_PORT_FILTER
00025 bool filter(const uint8_t *buf, size_t size);
00026 #else
00027 inline bool filter(const uint8_t *, size_t)
00028 {
00029     // default implementation filtering out no packets, see filter.cc for
00030     // other examples
00031     return false;
00032 }
00033 #endif
00034
00043 inline bool filter_request(Net_request const &req)
00044 {
00045     size_t size;
00046     const uint8_t *buf = req.buffer(&size);
00047     return filter(buf, size);
00048 }

```

16.49 mac_addr.h

```

00001 /*
00002  * Copyright (C) 2016-2017, 2020, 2022-2025 Kernkonzept GmbH.
00003  * Author(s): Jean Wolter <jean.wolter@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include <cstring>
00010 #include <inttypes.h>
00015
00019 class Mac_addr
00020 {
00021 public:
00022     enum
00023     {
00024         Addr_length = 6,
00025         Addr_unknown = 0ULL
00026     };
00027
00028     explicit Mac_addr(char const *_src)
00029     {
00030         /* A mac address is 6 bytes long, it is transmitted in big endian
00031            order over the network. For our internal representation we
00032            focus on easy testability of broadcast/multicast and reorder
00033            the bytes that the most significant byte becomes the least
00034            significant one. */
00035         unsigned char const *src = reinterpret_cast<unsigned char const *>(_src);
00036         _mac = ((uint64_t)src[0]) | (((uint64_t)src[1]) << 8)
00037             | (((uint64_t)src[2]) << 16) | (((uint64_t)src[3]) << 24)
00038             | (((uint64_t)src[4]) << 32) | (((uint64_t)src[5]) << 40);
00039     }
00040
00041     static Mac_addr from_uncached(char volatile const *src)
00042     { return Mac_addr(src); }
00043
00044     explicit Mac_addr(uint64_t mac) : _mac{mac} {}
00045
00046     Mac_addr(Mac_addr const &other) : _mac{other._mac} {}
00047
00049     bool is_broadcast() const
00050     {
00051         /* There are broadcast and multicast addresses, both are supposed
00052            to be delivered to all station and the local network (layer 2).
00053
00054            Broadcast address is FF:FF:FF:FF:FF:FF, multicast addresses have
00055            the LSB of the first octet set. Since this holds for both
00056            broadcast and multicast we test for the multicast bit here.
00057

```



```

00058         In our internal representation we store the bytes in reverse
00059         order, so we test the least significant bit of the least
00060         significant byte.
00061     */
00062     return _mac & 1;
00063 }
00064
00066 bool is_unknown() const
00067 { return _mac == Addr_unknown; }
00068
00069 bool operator == (Mac_addr const &other) const
00070 { return _mac == other._mac; }
00071
00072 bool operator != (Mac_addr const &other) const
00073 { return _mac != other._mac; }
00074
00075 bool operator < (Mac_addr const &other) const
00076 { return _mac < other._mac; }
00077
00078 Mac_addr& operator = (Mac_addr const &other)
00079 { _mac = other._mac; return *this; }
00080
00081 Mac_addr& operator = (uint64_t mac)
00082 { _mac = mac; return *this; }
00083
00084 template<typename T>
00085 void print(T &stream) const
00086 {
00087     stream.cprintf("%02x:%02x:%02x:%02x:%02x:%02x",
00088         (int)(_mac & 0xff), (int)((_mac >> 8) & 0xff),
00089         (int)((_mac >> 16) & 0xff), (int)((_mac >> 24) & 0xff),
00090         (int)((_mac >> 32) & 0xff), (int)((_mac >> 40) & 0xff));
00091 }
00092
00093 void to_array(unsigned char mac[6]) const
00094 {
00095     mac[0] = _mac & 0xffU;
00096     mac[1] = (_mac >> 8) & 0xffU;
00097     mac[2] = (_mac >> 16) & 0xffU;
00098     mac[3] = (_mac >> 24) & 0xffU;
00099     mac[4] = (_mac >> 32) & 0xffU;
00100     mac[5] = (_mac >> 40) & 0xffU;
00101 }
00102
00103 private:
00104     explicit Mac_addr(char volatile const *_src)
00105     {
00106         /* A mac address is 6 bytes long, it is transmitted in big endian
00107         order over the network. For our internal representation we
00108         focus on easy testability of broadcast/multicast and reorder
00109         the bytes that the most significant byte becomes the least
00110         significant one. */
00111         volatile unsigned char const *src = reinterpret_cast<volatile unsigned char const *>(_src);
00112         _mac = ((uint64_t)src[0]) | (((uint64_t)src[1]) << 8)
00113             | (((uint64_t)src[2]) << 16) | (((uint64_t)src[3]) << 24)
00114             | (((uint64_t)src[4]) << 32) | (((uint64_t)src[5]) << 40);
00115     }
00116
00118     uint64_t _mac;
00119 };

```

16.50 mac_table.h

```

00001 /*
00002  * Copyright (C) 2016-2017, 2020, 2022-2025 Kernkonzept GmbH.
00003  * Author(s): Jean Wolter <jean.wolter@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include "mac_addr.h"
00010 #include "port.h"
00011
00012 #include <array>
00013 #include <map>
00014 #include <tuple>
00015 #include <algorithm>
00020
00039 template<std::size_t Size = 1024U>
00040 class Mac_table
00041 {
00042 public:

```

```

00043 Mac_table()
00044 : _mac_table(),
00045   _entries(),
00046   _rr_index(0U)
00047 {}
00048
00058 Port_iface *lookup(Mac_addr dst, l4_uint16_t vlan_id) const
00059 {
00060     auto entry = _mac_table.find(std::tuple(dst, vlan_id));
00061     return (entry != _mac_table.end()) ? entry->second->port : nullptr;
00062 }
00063
00078 void learn(Mac_addr src, Port_iface *port, l4_uint16_t vlan_id)
00079 {
00080     Dbg info(Dbg::Port, Dbg::Info);
00081
00082     if (L4_UNLIKELY(info.is_active()))
00083     {
00084         // check whether we already know about src mac and vlan_id
00085         auto *p = lookup(src, vlan_id);
00086         if (!p || p != port)
00087         {
00088             info.printf("%s %-20s -> ", !p ? "learned " : "replaced",
00089                         port->get_name());
00090             src.print(info);
00091             info.cprintf("\n");
00092         }
00093     }
00094
00095     auto status = _mac_table.emplace(std::tuple(src, vlan_id),
00096                                     &_entries[_rr_index]);
00097     if (L4_UNLIKELY(status.second))
00098     {
00099         if (_entries[_rr_index].port)
00100         {
00101             // remove old entry
00102             _mac_table.erase(std::tuple(_entries[_rr_index].addr,
00103                                         _entries[_rr_index].vlan_id));
00104         }
00105         // Set/Replace port and mac address
00106         _entries[_rr_index].port = port;
00107         _entries[_rr_index].addr = src;
00108         _entries[_rr_index].vlan_id = vlan_id;
00109         _rr_index = (_rr_index + 1U) % Size;
00110     }
00111     else
00112     {
00113         // Update port to allow for movement of client between ports
00114         status.first->second->port = port;
00115     }
00116 }
00117
00129 void flush(Port_iface *port)
00130 {
00131     typedef std::pair<std::tuple<const Mac_addr, l4_uint16_t>, Entry*> TableEntry;
00132
00133     auto iter = _mac_table.begin();
00134     while ((iter = std::find_if(iter, _mac_table.end(),
00135                                [port](TableEntry const &p)
00136                                { return p.second->port == port; })))
00137     {
00138         iter->second->port = nullptr;
00139         iter->second->addr = Mac_addr::Addr_unknown;
00140         iter->second->vlan_id = 0;
00141         iter = _mac_table.erase(iter);
00142     }
00143
00144     assert(std::find_if(_mac_table.begin(), _mac_table.end(),
00145                        [port](TableEntry const &p)
00146                        { return p.second->port == port; }) == _mac_table.end());
00147 }
00148
00150 private:
00157 struct Entry {
00158     Port_iface *port;
00159     Mac_addr addr;
00160     l4_uint16_t vlan_id;
00161
00162     Entry()
00163     : port(nullptr),
00164       addr(Mac_addr::Addr_unknown),
00165       vlan_id(0)
00166     {}
00167 };
00168
00169 std::map<std::tuple<Mac_addr, l4_uint16_t>, Entry*> _mac_table;

```

```

00170     std::array<Entry, Size> _entries;
00171     size_t _rr_index;
00172 };

```

16.51 main.cc

```

00001  /*
00002  * Copyright (C) 2016-2020, 2022-2025 Kernkonzept GmbH.
00003  * Author(s): Jean Wolter <jean.wolter@kernkonzept.com>
00004  *             Manuel von Oltersdorff-Kalettkka <manuel.kalettkka@kernkonzept.com>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008  #include <l4/re/util/meta>
00009  #include <l4/re/util/object_registry>
00010  #include <l4/re/util/br_manager>
00011
00012  #include <l4/sys/factory>
00013  #include <l4/sys/task>
00014
00015  #include <l4/sys/cxx/ipc_epiface>
00016  #include <l4/sys/cxx/ipc_varg>
00017  #include <l4/cxx/dlist>
00018  #include <l4/cxx/string>
00019
00020  #include <stdlib.h>
00021  #include <string>
00022  #include <terminate_handler-l4>
00023  #include <vector>
00024
00025  #include "debug.h"
00026  #include "options.h"
00027  #include "switch.h"
00028  #include "vlan.h"
00029  #include <l4/virtio-net-switch/stats.h>
00030
00047
00048
00049  /*
00050  * Registry for our server, used to register
00051  * - factory capability
00052  * - irq object for capability deletion irqs
00053  * - virtio host kick irqs
00054  */
00055  static L4Re::Util::Registry_server<L4Re::Util::Br_manager_hooks> server;
00056
00057  using Ds_vector = std::vector<L4::Cap<L4Re::Dataspace>;
00058  static std::shared_ptr<Ds_vector> trusted_dataspaces;
00059
00060  static bool
00061  parse_int_param(L4::Ipc::Varg const &param, char const *prefix, int *out)
00062  {
00063      l4_size_t headlen = strlen(prefix);
00064
00065      if (param.length() < headlen)
00066          return false;
00067
00068      char const *pstr = param.value<char const *>();
00069
00070      if (strncmp(pstr, prefix, headlen) != 0)
00071          return false;
00072
00073      std::string tail(pstr + headlen, param.length() - headlen);
00074
00075      if (!parse_int_optstring(tail.c_str(), out))
00076      {
00077          Err(Err::Normal).printf("Bad parameter '%s'. Invalid number specified.\n",
00078                                  prefix);
00079          throw L4::Runtime_error(-L4_EINVAL);
00080      }
00081
00082      return true;
00083  }
00084
00085  static void
00086  assign_random_mac(l4_uint8_t mac[6])
00087  {
00088      static bool initialized = false;
00089
00090      if (!initialized)
00091      {
00092          srandom(l4_kip_clock(l4re_kip()));
00093          initialized = true;

```



```

00215     server.registry()->unregister_obj(&_reschedule_tx_irq);
00216 }
00217 };
00218
00222 class Monitor_port : public Port
00223 {
00229     class Kick_irq : public L4::Irqep_t<Kick_irq>
00230     {
00231         L4virtio_port *_port;
00232
00233     public:
00241         void handle_irq()
00242         {
00243             do
00244             {
00245                 _port->tx_q()->disable_notify();
00246                 _port->rx_q()->disable_notify();
00247
00248                 _port->drop_requests();
00249
00250                 _port->tx_q()->enable_notify();
00251                 _port->rx_q()->enable_notify();
00252
00253                 L4virtio::wmb();
00254                 L4virtio::rmb();
00255             }
00256             while (_port->tx_work_pending());
00257         }
00258
00259         Kick_irq(L4virtio_port *port) : _port{port} {}
00260     };
00261
00262     Kick_irq _kick_irq;
00263
00264 public:
00265     Monitor_port(L4Re::Util::Object_registry* registry,
00266                 unsigned vq_max, unsigned num_ds, char const *name,
00267                 l4_uint8_t const *mac)
00268     : Port(vq_max, num_ds, name, mac), _kick_irq(this)
00269     { register_end_points(registry, &_kick_irq); }
00270
00271     virtual ~Monitor_port()
00272     {
00273         // We need to delete the IRQ object created in register_irq_obj() ourselves
00274         L4::Cap<L4::Task> (L4Re::This_task)
00275         ->unmap(_kick_irq.obj_cap().fpage(),
00276                L4_FP_ALL_SPACES | L4_FP_DELETE_OBJ);
00277         server.registry()->unregister_obj(&_kick_irq);
00278     }
00279 };
00280
00284 class Stats_reader
00285 : public cxx::D_list_item,
00286   public L4::Epiface_t<Stats_reader, Virtio_net_switch::Statistics_if>
00287 {
00288     L4Re::Util::Unique_cap<L4Re::Dataspace> _ds;
00289     l4_addr_t _addr;
00290
00291 public:
00292     Stats_reader()
00293     {
00294         l4_size_t size = Switch_statistics::get_instance().size();
00295         _ds = L4Re::Util::make_unique_cap<L4Re::Dataspace>();
00296         L4Re::chksys(L4Re::Env::env()->mem_alloc()->alloc(size, _ds.get()),
00297                     "Could not allocate shared mem ds.");
00298         L4Re::chksys(L4Re::Env::env()->rm()->attach(&_addr, _ds->size(),
00299                                                    L4Re::Rm::F::Search_addr
00300                                                    | L4Re::Rm::F::RW,
00301                                                    L4::Ipc::make_cap_rw(_ds.get())));
00302
00303         memset(reinterpret_cast<void*>(_addr), 0, _ds->size());
00304     }
00305
00306     ~Stats_reader()
00307     {
00308         L4Re::Env::env()->rm()->detach(reinterpret_cast<l4_addr_t>(_addr), 0);
00309         server.registry()->unregister_obj(this);
00310     }
00311
00312     long op_get_buffer(Virtio_net_switch::Statistics_if::Rights,
00313                      L4::Ipc::Cap<L4Re::Dataspace> &ds)
00314     {
00315         // We hand out the dataspace in a read only manner. Clients must not be
00316         // able to modify information as that would create an unwanted data
00317         // channel.
00318         ds = L4::Ipc::Cap<L4Re::Dataspace>(_ds.get(), L4_CAP_FPAGE_RO);
00319         return L4_EOK;

```

```

00320     }
00321
00322     long op_sync(Virtio_net_switch::Statistics_if::Rights)
00323     {
00324         memcpy(reinterpret_cast<void *>(_addr),
00325                reinterpret_cast<void *>(Switch_statistics::get_instance().stats()),
00326                Switch_statistics::get_instance().size());
00327         return L4_EOK;
00328     }
00329
00330     bool is_valid()
00331     { return obj_cap() && obj_cap().validate().label(); }
00332 };
00333
00334 class Stats_reader_list
00335 {
00336     cxx::D_list<Stats_reader> _readers;
00337
00338 public:
00339     void check_readers()
00340     {
00341         auto it = _readers.begin();
00342         while (it != _readers.end())
00343         {
00344             auto *reader = *it;
00345             if (!reader->is_valid())
00346             {
00347                 it = _readers.erase(it);
00348                 delete reader;
00349             }
00350             else
00351                 ++it;
00352         }
00353     }
00354
00355     void push_back(cxx::unique_ptr<Stats_reader> reader)
00356     {
00357         _readers.push_back(reader.release());
00358     }
00359 };
00360
00361 /*
00362  * Handle vanishing caps by telling the switch that a port might have gone
00363  */
00364 struct Del_cap_irq : public L4::Irqep_t<Del_cap_irq>
00365 {
00366 public:
00367     void handle_irq()
00368     {
00369         _switch->check_ports();
00370         _stats_readers->check_readers();
00371     }
00372
00373     Del_cap_irq(Virtio_switch *virtio_switch, Stats_reader_list *stats_readers)
00374     : _switch{virtio_switch},
00375       _stats_readers{stats_readers}
00376     {}
00377
00378 private:
00379     Virtio_switch *_switch;
00380     Stats_reader_list *_stats_readers;
00381 };
00382
00383 Virtio_switch *_virtio_switch;
00384
00386 unsigned _vq_max_num;
00387 Stats_reader_list _stats_readers;
00388 Del_cap_irq _del_cap_irq;
00393 unsigned _dbg_port_num = 0;
00394
00410 bool handle_opt_arg(L4::Ipc::Varg const &opt, bool &monitor,
00411                   char *name, size_t size,
00412                   l4_uint16_t &vlan_access,
00413                   std::vector<l4_uint16_t> &vlan_trunk,
00414                   bool *vlan_trunk_all,
00415                   l4_uint8_t mac[6], bool &mac_set)
00416 {
00417     assert(opt.is_of<char const *>());
00418     unsigned len = opt.length();
00419     const char *opt_str = opt.data();
00420     Err err(Err::Normal);
00421
00422
00423     if (len > 5)
00424     {
00425         if (!strncmp("type=", opt_str, 5))
00426     
```

```

00427         if (!strcmp("type=monitor", opt_str, len))
00428         {
00429             monitor = true;
00430             return true;
00431         }
00432         else if (!strcmp("type=none", opt_str, len))
00433             return true;
00434
00435         err.printf("Unknown type '%.*s'\n", opt.length() - 5, opt.data() + 5);
00436         return false;
00437     }
00438     else if (!strcmp("name=", opt_str, 5))
00439     {
00440         snprintf(name, size, "%.*s", opt.length() - 5, opt.data() + 5);
00441         return true;
00442     }
00443     else if (!strcmp("vlan=", opt_str, 5))
00444     {
00445         cxx::String str(opt_str + 5, strlen(opt_str + 5, len - 5));
00446         cxx::String::Index idx;
00447
00448         if ((idx = str.starts_with("access=")))
00449         {
00450             str = str.substr(idx);
00451             l4_uint16_t vid;
00452             int next = str.from_dec(&vid);
00453             if (next && next == str.len() && vlan_valid_id(vid))
00454                 vlan_access = vid;
00455             else
00456             {
00457                 err.printf("Invalid VLAN access port id '%.*s'\n",
00458                     opt.length(), opt.data());
00459                 return false;
00460             }
00461         }
00462         else if ((idx = str.starts_with("trunk=")))
00463         {
00464             int next;
00465             l4_uint16_t vid;
00466             str = str.substr(idx);
00467             if (str == cxx::String("all"))
00468             {
00469                 *vlan_trunk_all = true;
00470                 return true;
00471             }
00472             while ((next = str.from_dec(&vid)))
00473             {
00474                 if (!vlan_valid_id(vid))
00475                     break;
00476                 vlan_trunk.push_back(vid);
00477                 if (next < str.len() && str[next] != ',')
00478                     break;
00479                 str = str.substr(next+1);
00480             }
00481
00482             if (vlan_trunk.empty() || !str.empty())
00483             {
00484                 err.printf("Invalid VLAN trunk port spec '%.*s'\n",
00485                     opt.length(), opt.data());
00486                 return false;
00487             }
00488         }
00489         else
00490         {
00491             err.printf("Invalid VLAN specification..\n");
00492             return false;
00493         }
00494     }
00495     return true;
00496 }
00497 else if (!strcmp("mac=", opt_str, 4))
00498 {
00499     size_t const OPT_LEN = 4 /* mac= */ + 6*2 /* digits */ + 5 /* : */;
00500     // expect NUL terminated string for simplicity
00501     if (len > OPT_LEN && opt_str[OPT_LEN] == '\0' &&
00502         sscanf(opt_str+4, "%hhx:%hhx:%hhx:%hhx:%hhx:%hhx", &mac[0],
00503             &mac[1], &mac[2], &mac[3], &mac[4], &mac[5]) == 6)
00504     {
00505         mac_set = true;
00506         return true;
00507     }
00508
00509     err.printf("Invalid mac address '%.*s'\n", len - 4, opt_str + 4);
00510     return false;
00511 }
00512 }
00513

```

```

00514     err.printf("Unknown option '%.*s'\n", opt.length(), opt.data());
00515     return false;
00516 }
00517
00518 public:
00519 Switch_factory(Virtio_switch *virtio_switch, unsigned vq_max_num)
00520 : _virtio_switch{virtio_switch}, _vq_max_num{vq_max_num},
00521   _del_cap_irq{virtio_switch, &stats_readers}
00522 {
00523     auto c = L4Re::chkcap(server.registry()->register_irq_obj(&_del_cap_irq));
00524     L4Re::chksys(L4Re::Env::env()->main_thread()->register_del_irq(c));
00525 };
00526
00533 long op_create(L4::Factory::Rights, L4::Ipc::Cap<void> &res,
00534               l4_umword_t type, L4::Ipc::Varg_list_ref va)
00535 {
00536     switch (type)
00537     {
00538     case 0:
00539         return create_port(res, va);
00540     case 1:
00541         return create_stats(res);
00542     default:
00543         Dbg(Dbg::Core, Dbg::Warn).printf("op_create: Invalid object type\n");
00544         return -L4_EINVAL;
00545     }
00546 }
00547
00548 long create_port(L4::Ipc::Cap<void> &res, L4::Ipc::Varg_list_ref va)
00549 {
00550     Dbg warn(Dbg::Port, Dbg::Warn, "Port");
00551     Dbg info(Dbg::Port, Dbg::Info, "Port");
00552
00553     info.printf("Incoming port request\n");
00554
00555     bool monitor = false;
00556     char name[20] = "";
00557     unsigned arg_n = 2;
00558     l4_uint16_t vlan_access = 0;
00559     std::vector<l4_uint16_t> vlan_trunk;
00560     bool vlan_trunk_all = false;
00561
00562     l4_uint8_t mac[6];
00563     bool mac_set = false;
00564     int num_ds = 2;
00565
00566     for (L4::Ipc::Varg opt: va)
00567     {
00568         if (!opt.is_of<char const *>())
00569         {
00570             warn.printf("Unexpected type for argument %d\n", arg_n);
00571             return -L4_EINVAL;
00572         }
00573
00574         if (parse_int_param(opt, "ds-max=", &num_ds))
00575         {
00576             if (num_ds <= 0 || num_ds > 80)
00577             {
00578                 Err(Err::Normal).printf("warning: client requested invalid number"
00579                                         " of data spaces: 0 < %d <= 80\n", num_ds);
00580                 return -L4_EINVAL;
00581             }
00582         }
00583         else if (!handle_opt_arg(opt, monitor, name, sizeof(name), vlan_access,
00584                                 vlan_trunk, &vlan_trunk_all, mac, mac_set))
00585             return -L4_EINVAL;
00586
00587         ++arg_n;
00588     }
00589
00590     if (!_virtio_switch->port_available(monitor))
00591     {
00592         warn.printf("No port available\n");
00593         return -L4_ENOMEM;
00594     }
00595
00596     if (vlan_access && (!vlan_trunk.empty() || vlan_trunk_all))
00597     {
00598         warn.printf("VLAN port cannot be access and trunk simultaneously.\n");
00599         return -L4_EINVAL;
00600     }
00601
00602     if (!name[0])
00603         snprintf(name, sizeof(name), "%s[%u]", monitor ? "monitor": "",
00604                 _dbg_port_num++);
00605
00606     info.printf("    Creating port %s\n", name,

```



```

00607         monitor ? " as monitor port" : "");
00608
00609         // Assign a random MAC address if we assign one to our devices but the
00610         // user has not passed an explicit one for a port.
00611         if (!mac_set && Options::get_options()->assign_mac())
00612             assign_random_mac(mac);
00613
00614         l4_uint8_t *mac_ptr = (mac_set || Options::get_options()->assign_mac())
00615             ? mac : nullptr;
00616
00617         // create port
00618         Port *port;
00619         if (monitor)
00620         {
00621             port = new Monitor_port(server.registry(), _vq_max_num, num_ds, name,
00622                                     mac_ptr);
00623             port->set_monitor();
00624
00625             if (vlan_access)
00626                 warn.printf("vlan=access=<id> ignored on monitor ports!\n");
00627             if (!vlan_trunk.empty())
00628                 warn.printf("vlan=trunk=... ignored on monitor ports!\n");
00629         }
00630         else
00631         {
00632             port = new Switch_port(server.registry(), _virtio_switch, _vq_max_num,
00633                                   num_ds, name, mac_ptr);
00634
00635             if (vlan_access)
00636                 port->set_vlan_access(vlan_access);
00637             else if (vlan_trunk_all)
00638                 port->set_vlan_trunk_all();
00639             else if (!vlan_trunk.empty())
00640                 port->set_vlan_trunk(vlan_trunk);
00641         }
00642
00643         port->add_trusted_dataspaces(trusted_dataspaces);
00644         if (!trusted_dataspaces->empty())
00645             port->enable_trusted_ds_validation();
00646
00647         // hand port over to the switch
00648         bool added = monitor ? _virtio_switch->add_monitor_port(port)
00649                               : _virtio_switch->add_port(port);
00650         if (!added)
00651         {
00652             delete port;
00653             return -L4_ENOMEM;
00654         }
00655         res = L4::Ipc::make_cap(port->obj_cap(), L4_CAP_FPAGE_RWSD);
00656
00657         info.printf("    Created port %s\n", name);
00658         return L4_EOK;
00659     }
00660
00661     long create_stats(L4::Ipc::Cap<void> &res)
00662     {
00663         // Create a stats reader and throw away our reference to get a notification
00664         // when the external reference vanishes.
00665         auto reader = cxx::make_unique<Stats_reader>();
00666         L4Re::chkcap(server.registry()->register_obj(reader.get()));
00667         reader->obj_cap()->dec_refcnt(1);
00668         res = L4::Ipc::make_cap(reader->obj_cap(),
00669                                 L4_CAP_FPAGE_R | L4_CAP_FPAGE_D);
00670         _stats_readers.push_back(cxx::move(reader));
00671         return L4_EOK;
00672     }
00673 };
00674
00675 #if CONFIG_VNS_IXL
00676 class Ixl_hw_port : public Ixl_port
00677 {
00678     template<typename Derived>
00679     class Port_irq : public L4::Irqep_t<Derived>
00680     {
00681     public:
00682         Port_irq(Virtio_switch *virtio_switch, Ixl_port *port)
00683             : _switch{virtio_switch}, _port{port} {}
00684     protected:
00685         Virtio_switch *_switch;
00686         Ixl_port *_port;
00687     };
00688
00689     class Receive_irq : public Port_irq<Receive_irq>
00690     {
00691     public:
00692         using Port_irq::Port_irq;
00693     };
00694
00695     using Receive_irq;
00696

```

```

00697
00704     void handle_irq()
00705     {
00706         if (!_port->dev()->check_recv_irq(0))
00707             return;
00708
00709         if (_switch->handle_ixl_port_tx(_port))
00710             _port->dev()->ack_recv_irq(0);
00711     }
00712 };
00713
00714 class Reschedule_tx_irq : public Port_irq<Reschedule_tx_irq>
00715 {
00716 public:
00717     using Port_irq::Port_irq;
00718
00719     void handle_irq()
00720     {
00721         if (_switch->handle_ixl_port_tx(_port))
00722             // Entire TX queue handled, re-enable the recv IRQ again.
00723             _port->dev()->ack_recv_irq(0);
00724     }
00725 };
00726
00727 Receive_irq _recv_irq;
00728 Reschedule_tx_irq _reschedule_tx_irq;
00729
00730 public:
00731     Ixl_hw_port(L4Re::Util::Object_registry *registry,
00732               Virtio_switch *virtio_switch, Ixl::Ixl_device *dev)
00733     : Ixl_port(dev),
00734       _recv_irq(virtio_switch, this),
00735       _reschedule_tx_irq(virtio_switch, this)
00736     {
00737         L4::Cap<L4::Irx> recv_irq_cap = L4Re::chkcap(dev->get_recv_irq(0), "Get receive IRQ");
00738         L4Re::chkcap(registry->register_obj(&_recv_irq, recv_irq_cap),
00739                     "Register receive IRQ.");
00740         recv_irq_cap->unmask();
00741
00742         _pending_tx_reschedule =
00743             L4Re::chkcap(registry->register_irq_obj(&_reschedule_tx_irq,
00744                                                  "Register TX reschedule IRQ."),
00745                         _pending_tx_reschedule->unmask());
00746     }
00747
00748     ~Ixl_hw_port() override
00749     {
00750         server.registry()->unregister_obj(&_recv_irq);
00751     }
00752 };
00753
00754 static void
00755 discover_ixl_devices(L4::Cap<L4vbus::Vbus> vbus, Virtio_switch *virtio_switch)
00756 {
00757     struct Ixl::Dev_cfg cfg;
00758     // Configure the device in asynchronous notify mode.
00759     cfg.irq_timeout_ms = -1;
00760
00761     // TODO: Support detecting multiple devices on a Vbus.
00762     // Setup the driver (also resets and initializes the NIC).
00763     Ixl::Ixl_device *dev = Ixl::Ixl_device::ixl_init(vbus, 0, cfg);
00764     if (!dev)
00765         // No Ixl supported device found, Ixl already printed an error message.
00766         return;
00767
00768     Ixl_hw_port *hw_port = new Ixl_hw_port(server.registry(), virtio_switch, dev);
00769     if (!virtio_switch->add_port(hw_port))
00770     {
00771         Err().printf("error adding ixl port\n");
00772         delete hw_port;
00773     }
00774 }
00775 #endif
00776
00777 int main(int argc, char *argv[])
00778 {
00779     trusted_dataspaces = std::make_shared<Ds_vector>();
00780     auto *opts = Options::parse_options(argc, argv, trusted_dataspaces);
00781     if (!opts)
00782     {
00783         Err().printf("Error during command line parsing.\n");
00784         return 1;
00785     }
00786
00787     // Show welcome message if debug level is not set to quiet
00788     if (Dbg(Dbg::Core, Dbg::Warn).is_active())
00789         printf("Hello from l4virtio switch\n");

```

```

00790
00791     Virtio_switch *virtio_switch = new Virtio_switch(opts->get_max_ports());
00792
00793 #ifdef CONFIG_VNS_STATS
00794     Switch_statistics::get_instance().initialize(opts->get_max_ports());
00795 #endif
00796
00797 #if CONFIG_VNS_IXL
00798     auto vbus = L4Re::Env::env()->get_cap<L4vbus::Vbus>("vbus");
00799     if (vbus.is_valid())
00800         discover_ixl_devices(vbus, virtio_switch);
00801 #endif
00802
00803     Switch_factory *factory = new Switch_factory(virtio_switch,
00804                                                  opts->get_virtq_max_num());
00805
00806     L4::Cap<void> cap = server.registry()->register_obj(factory, "svr");
00807     if (!cap.is_valid())
00808     {
00809         Err().printf("error registering switch\n");
00810         return 2;
00811     }
00812
00813     /*
00814     * server loop will handle 4 types of events
00815     * - Switch_factory
00816     * - factory protocol
00817     * - capability deletion
00818     * - delegated to Virtio_switch::check_ports()
00819     * - Switch_factory::Switch_port
00820     * - irqs triggered by clients
00821     * - delegated to Virtio_switch::handle_l4virtio_port_tx()
00822     * - Virtio_net_transfer
00823     * - timeouts for pending transfer requests added by
00824     *   Port_iface::handle_request() via registered via
00825     *   L4::Epiface::server_iface()->add_timeout()
00826     */
00827     server.loop();
00828     return 0;
00829 }
00830

```

16.52 Makefile

```

00001 PKGDIR = ..
00002 L4DIR ?= $(PKGDIR)/../..
00003
00004 PKGNAME = drivers
00005 PC_FILENAME = drivers-frst
00006 EXTRA_TARGET += hw_mmio_register_block hw_register_block
00007
00008 include $(L4DIR)/mk/include.mk

```

16.53 Makefile

```

00001 PKGDIR = ../..
00002 L4DIR ?= $(PKGDIR)/../..
00003
00004 PKGNAME = drivers
00005
00006 include $(L4DIR)/mk/include.mk

```

16.54 Makefile

```

00001 PKGDIR = ../..
00002 L4DIR ?= $(PKGDIR)/../..
00003
00004 EXTRA_TARGET += cmd_control
00005
00006 include $(L4DIR)/mk/include.mk

```

16.55 Makefile

```

00001 PKGDIR          ?= ../..
00002 L4DIR            ?= $(PKGDIR)/../..
00003
00004 TARGET            = l4vio_switch
00005
00006 REQUIRES_LIBS      = libstdc++ l4virtio
00007 REQUIRES_LIBS-$(CONFIG_VNS_IXL) += ixl
00008
00009 SRC_CC-$(CONFIG_VNS_PORT_FILTER) += filter.cc
00010
00011 SRC_CC = main.cc switch.cc options.cc
00012
00013 include $(L4DIR)/mk/prog.mk

```

16.56 options.cc

```

00001 /*
00002  * Copyright (C) 2016-2017, 2019, 2022-2025 Kernkonzept GmbH.
00003  * Author(s): Jean Wolter <jean.wolter@kernkonzept.com>
00004  *             Manuel von Oltersdorff-Kalettkka <manuel.kalettkka@kernkonzept.de>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008
00009 #include <getopt.h>
00010 #include <stdlib.h>
00011 #include <cstring>
00012 #include <type_traits>
00013
00014 #include <l4/cxx/exceptions>
00015 #include <l4/re/error_helper>
00016 #include <l4/re/env>
00017
00018 #include "debug.h"
00019 #include "options.h"
00020
00021 bool
00022 parse_int_optstring(char const *optstring, int *out)
00023 {
00024     char *endp;
00025
00026     errno = 0;
00027     long num = strtol(optstring, &endp, 10);
00028
00029     // check that long can be converted to int
00030     if (errno || *endp != '\0' || num < INT_MIN || num > INT_MAX)
00031         return false;
00032
00033     *out = num;
00034
00035     return true;
00036 }
00037
00038 static int
00039 verbosity_mask_from_string(char const *str, unsigned *mask)
00040 {
00041     if (strcmp("quiet", str) == 0)
00042     {
00043         *mask = Dbg::Quiet;
00044         return 0;
00045     }
00046     if (strcmp("warn", str) == 0)
00047     {
00048         *mask = Dbg::Warn;
00049         return 0;
00050     }
00051     if (strcmp("info", str) == 0)
00052     {
00053         *mask = Dbg::Warn | Dbg::Info;
00054         return 0;
00055     }
00056     if (strcmp("debug", str) == 0)
00057     {
00058         *mask = Dbg::Warn | Dbg::Info | Dbg::Debug;
00059         return 0;
00060     }
00061     if (strcmp("trace", str) == 0)
00062     {
00063         *mask = Dbg::Warn | Dbg::Info | Dbg::Debug | Dbg::Trace;
00064         return 0;

```

```

00065     }
00066
00067     return -L4_ENOENT;
00068 }
00069
00099 static void
00100 set_verbosity(char const *str)
00101 {
00102     unsigned mask;
00103     if (verbosity_mask_from_string(str, &mask) == 0)
00104     {
00105         Dbg::set_verbosity(mask);
00106         return;
00107     }
00108
00109     static char const *const components[] =
00110     { "core", "virtio", "port", "request", "queue", "packet" };
00111
00112     static_assert(std::extent<decltype(components)>::value == Dbg::Max_component,
00113         "Component names must match 'enum Component'.");
00114
00115     for (unsigned i = 0; i < Dbg::Max_component; ++i)
00116     {
00117         auto len = strlen(components[i]);
00118         if (strncmp(components[i], str, len) == 0 && str[len] == '='
00119             && verbosity_mask_from_string(str + len + 1, &mask) == 0)
00120         {
00121             Dbg::set_verbosity(i, mask);
00122             return;
00123         }
00124     }
00125 }
00126
00127 int
00128 Options::parse_cmd_line(int argc, char **argv,
00129     std::shared_ptr<Ds_vector> trusted_dataspaces)
00130 {
00131     int opt, index;
00132
00133     struct option options[] =
00134     {
00135         {"size",          1, 0, 's' }, // size of in/out queue == #buffers in queue
00136         {"ports",         1, 0, 'p' }, // number of ports
00137         {"mac",           0, 0, 'm' }, // switch sets MAC address for each client
00138         {"debug",         1, 0, 'D' }, // configure debug levels
00139         {"verbose",       0, 0, 'v' },
00140         {"quiet",         0, 0, 'q' },
00141         {"register-ds",   1, 0, 'd' }, // register a trusted dataspace
00142         {0, 0, 0, 0}
00143     };
00144
00145     unsigned long verbosity = Dbg::Warn;
00146
00147     Dbg info(Dbg::Core, Dbg::Info);
00148
00149     Dbg::set_verbosity(Dbg::Core, Dbg::Info);
00150     info.printf("Arguments:\n");
00151     for (int i = 0; i < argc; ++i)
00152         info.printf("\t%s\n", argv[i]);
00153
00154     Dbg::set_verbosity(verbosity);
00155     while ( (opt = getopt_long(argc, argv, "s:p:mMqvD:d:", options, &index)) != -1)
00156     {
00157         switch (opt)
00158         {
00159             case 's':
00160
00161                 // QueueNumMax must be power of 2 between 1 and 0x8000
00162                 if (!parse_int_optstring(optarg, &_virtq_max_num)
00163                     || _virtq_max_num < 1 || _virtq_max_num > 32768
00164                     || (_virtq_max_num & (_virtq_max_num - 1)))
00165                 {
00166                     Err().printf("Max number of virtqueue buffers must be power of 2"
00167                         " between 1 and 32768. Invalid value %i or argument "
00168                         "%s\n",
00169                         _virtq_max_num, optarg);
00170                     return -1;
00171                 }
00172                 info.printf("Max number of buffers in virtqueue: %i\n",
00173                     _virtq_max_num);
00174                 break;
00175             case 'p':
00176                 if (parse_int_optstring(optarg, &_max_ports))
00177                     info.printf("Max number of ports: %u\n", _max_ports);
00178             else
00179             {
00180                 Err().printf("Invalid number of ports argument: %s\n", optarg);

```

```

00181         return -1;
00182     }
00183     break;
00184     case 'q':
00185         verbosity = Dbg::Quiet;
00186         Dbg::set_verbosity(verbosity);
00187         break;
00188     case 'v':
00189         verbosity = (verbosity < 1) | 1;
00190         Dbg::set_verbosity(verbosity);
00191         break;
00192     case 'D':
00193         set_verbosity(optarg);
00194         break;
00195     case 'm':
00196         info.printf("Option -m ignored to compatibility.\n");
00197         break;
00198     case 'M':
00199         _assign_mac = false;
00200         break;
00201     case 'd':
00202     {
00203         L4Re::Cap<L4Re::Dataspace> ds =
00204             L4Re::chkcap(L4Re::Env::env()->get_cap<L4Re::Dataspace>(optarg),
00205                 "Find a dataspace capability.\n");
00206         trusted_dataspaces->push_back(ds);
00207         break;
00208     }
00209     default:
00210         Err().printf("Unknown command line option '%c' (%d)\n", opt, opt);
00211         return -1;
00212     }
00213 }
00214 return 0;
00215 }
00216
00217 static Options options;
00218
00219 Options const *
00220 Options::get_options()
00221 { return &options; }
00222
00223 Options const *
00224 Options::parse_options(int argc, char **argv,
00225     std::shared_ptr<Ds_vector> trusted_dataspaces)
00226 {
00227     if (options.parse_cmd_line(argc, argv, trusted_dataspaces) < 0)
00228         return nullptr;
00229
00230     return &options;
00231 }

```

16.57 options.h

```

00001 /*
00002  * Copyright (C) 2016-2017, 2022, 2024-2025 Kernkonzept GmbH.
00003  * Author(s): Jean Wolter <jean.wolter@kernkonzept.com>
00004  *             Manuel von Oltersdorff-Kalettka <manuel.kalettka@kernkonzept.com>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include <memory>
00011 #include <vector>
00012 #include <cerrno>
00013 #include <climits>
00014
00015 #include <l4/re/dataspace>
00016
00017 bool
00018 parse_int_optstring(char const *optstring, int *out);
00019
00020 class Options
00021 {
00022     using Ds_vector = std::vector<L4Re::Cap<L4Re::Dataspace>;
00023 public:
00024     int get_max_ports() const
00025     { return _max_ports; }
00026
00027     int get_virtq_max_num() const
00028     { return _virtq_max_num; }
00029

```

```

00030 int get_portq_max_num() const
00031 { return _portq_max_num; }
00032
00033 int get_request_timeout() const
00034 { return _request_timeout; }
00035
00036 int assign_mac() const
00037 { return _assign_mac; }
00038
00039 static Options const *
00040 parse_options(int argc, char **argv,
00041               std::shared_ptr<Ds_vector> trusted_dataspaces);
00042 static Options const *get_options();
00043
00044 private:
00045 int _max_ports = 256;
00046 int _virtq_max_num = 0x100; // default value for data queues
00047 int _portq_max_num = 50;    // default value for port queues
00048 int _request_timeout = 1 * 1000 * 1000; // default packet timeout 1 second
00049 bool _assign_mac = true;
00050
00051 int parse_cmd_line(int argc, char **argv,
00052                   std::shared_ptr<Ds_vector> trusted_dataspaces);
00053 };

```

16.58 port.h

```

00001 /*
00002  * Copyright (C) 2016-2018, 2020, 2022-2025 Kernkonzept GmbH.
00003  * Author(s): Jean Wolter <jean.wolter@kernkonzept.com>
00004  *            Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include "request.h"
00011 #include "mac_addr.h"
00012 #include "vlan.h"
00013 #include "stats.h"
00014
00015 #include <cassert>
00016 #include <set>
00017 #include <vector>
00018
00023
00024 class Port_iface
00025 {
00026 protected:
00027     Virtio_net_switch::Port_statistics *_stats;
00028
00029 public:
00030     Port_iface(char const *name)
00031     {
00032         strncpy(_name, name, sizeof(_name));
00033         _name[sizeof(_name) - 1] = '\0';
00034 #ifdef CONFIG_VNS_STATS
00035         _stats = Switch_statistics::get_instance().allocate_port_statistics(name);
00036         if (!_stats)
00037             throw L4::Runtime_error(-L4_ENOMEM,
00038                                     "Could not allocate port statistics.\n");
00039 #endif
00040     }
00041
00042     virtual ~Port_iface()
00043     {
00044 #ifdef CONFIG_VNS_STATS
00045         _stats->in_use = false;
00046 #endif
00047     }
00048
00049     // delete copy and assignment
00050     Port_iface(Port_iface const &) = delete;
00051     Port_iface &operator = (Port_iface const &) = delete;
00052
00053     char const *get_name() const
00054     { return _name; }
00055
00056     l4_uint16_t get_vlan() const
00057     { return _vlan_id; }
00058
00059     inline bool is_trunk() const
00060     { return _vlan_id == VLAN_ID_TRUNK; }

```

```

00061
00062 inline bool is_native() const
00063 { return _vlan_id == VLAN_ID_NATIVE; }
00064
00065 inline bool is_access() const
00066 { return !is_trunk() && !is_native(); }
00067
00075 void set_vlan_access(l4_uint16_t id)
00076 {
00077     assert(vlan_valid_id(id));
00078     _vlan_id = id;
00079     _vlan_bloom_filter = 0;
00080     _vlan_ids.clear();
00081 }
00082
00092 void set_vlan_trunk(const std::vector<l4_uint16_t> &ids)
00093 {
00094     // bloom filter to quickly reject packets that do not belong to this port
00095     l4_uint32_t filter = 0;
00096
00097     _vlan_ids.clear();
00098     for (const auto id : ids)
00099     {
00100         assert(vlan_valid_id(id));
00101         filter |= vlan_bloom_hash(id);
00102         _vlan_ids.insert(id);
00103     }
00104
00105     _vlan_id = VLAN_ID_TRUNK;
00106     _vlan_bloom_filter = filter;
00107 }
00108
00112 void set_vlan_trunk_all()
00113 {
00114     _vlan_all = true;
00115     _vlan_id = VLAN_ID_TRUNK;
00116     _vlan_bloom_filter = -1;
00117 }
00118
00125 void set_monitor()
00126 {
00127     _vlan_id = VLAN_ID_TRUNK;
00128     _vlan_bloom_filter = 0;
00129 }
00130
00139 bool match_vlan(uint16_t id)
00140 {
00141     // Regular case native/access port
00142     if (id == _vlan_id)
00143         return true;
00144
00145     // This port participates in all VLANs
00146     if (_vlan_all)
00147         return true;
00148
00149     // Quick check: does port probably accept this VLAN?
00150     if ((_vlan_bloom_filter & vlan_bloom_hash(id)) == 0)
00151         return false;
00152
00153     return _vlan_ids.find(id) != _vlan_ids.end();
00154 }
00155
00162 inline Mac_addr mac() const
00163 { return _mac; }
00164
00165 Virtio_vlan_mangle create_vlan_mangle(Port_iface *src_port) const
00166 {
00167     Virtio_vlan_mangle mangle;
00168
00169     if (is_trunk())
00170     {
00171         /*
00172          * Add a VLAN tag only if the packet does not already have one (by
00173          * coming from another trunk port) or if the packet does not belong to
00174          * any VLAN (by coming from a native port). The latter case is only
00175          * relevant if this is a monitor port. Otherwise traffic from native
00176          * ports is never forwarded to trunk ports.
00177          */
00178         if (!src_port->is_trunk() && !src_port->is_native())
00179             mangle = Virtio_vlan_mangle::add(src_port->get_vlan());
00180     }
00181     else
00182     {
00183         /*
00184          * Remove VLAN tag only if the packet actually has one (by coming from a
00185          * trunk port).
00186          */
00187         if (src_port->is_trunk())

```



```

00187         mangle = Virtio_vlan_mangle::remove();
00188
00189     return mangle;
00190 }
00191
00192 virtual void rx_notify_disable_and_remember() = 0;
00193 virtual void rx_notify_emit_and_enable() = 0;
00194
00195 virtual bool is_gone() const = 0;
00196
00197 // std::optional<Net_request> get_tx_request() = 0;
00198
00199 enum class Result
00200 {
00201     Delivered, Exception, Dropped,
00202 };
00203
00204 virtual Result handle_request(Port_iface *src_port,
00205                               Net_transfer &src,
00206                               l4_uint64_t *bytes_transferred) = 0;
00207
00208 void reschedule_pending_tx()
00209 { _pending_tx_reschedule->trigger(); }
00210
00211 protected:
00212 /*
00213  * VLAN related management information.
00214  *
00215  * A port may either be
00216  * - a native port (_vlan_id == VLAN_ID_NATIVE), or
00217  * - an access port (_vlan_id set accordingly), or
00218  * - a trunk port (_vlan_id == VLAN_ID_TRUNK, _vlan_bloom_filter and
00219  *   _vlan_ids populated accordingly, or _vlan_all == true).
00220  */
00221 l4_uint16_t _vlan_id = VLAN_ID_NATIVE; // VID for native/access port
00222 l4_uint32_t _vlan_bloom_filter = 0; // Bloom filter for trunk ports
00223 std::set<l4_uint16_t> _vlan_ids; // Authoritative list of trunk VLANs
00224 bool _vlan_all = false; // This port participates in all VLANs (ignoring _vlan_ids)
00225
00226 inline l4_uint32_t vlan_bloom_hash(l4_uint16_t vid)
00227 { return LUL << (vid & 31U); }
00228
00229 L4::Cap<L4::Irq> _pending_tx_reschedule;
00230
00231 Mac_addr _mac = Mac_addr(Mac_addr::Addr_unknown);
00232 char _name[20];
00233
00234 public:
00235 #ifdef CONFIG_VNS_STATS
00236     inline void stat_inc_tx_num()
00237     { _stats->tx_num++; }
00238     inline void stat_inc_tx_dropped()
00239     { _stats->tx_dropped++; }
00240     inline void stat_inc_tx_bytes(l4_uint64_t bytes)
00241     { _stats->tx_bytes += bytes; }
00242     inline void stat_inc_rx_num()
00243     { _stats->rx_num++; }
00244     inline void stat_inc_rx_dropped()
00245     { _stats->rx_dropped++; }
00246     inline void stat_inc_rx_bytes(l4_uint64_t bytes)
00247     { _stats->rx_bytes += bytes; }
00248 #else
00249     inline void stat_inc_tx_num()
00250     {}
00251     inline void stat_inc_tx_dropped()
00252     {}
00253     inline void stat_inc_tx_bytes(l4_uint64_t /*bytes*/)
00254     {}
00255     inline void stat_inc_rx_num()
00256     {}
00257     inline void stat_inc_rx_dropped()
00258     {}
00259     inline void stat_inc_rx_bytes(l4_uint64_t /*bytes*/)
00260     {}
00261 #endif
00262 };
00263
00264
00265

```

16.59 port_ixl.h

```

00001 /*
00002  * Copyright (C) 2024-2025 Kernkonzept GmbH.
00003  * Author(s): Georg Kotheimer <georg.kotheimer@kernkonzept.com>

```

```

00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include "port.h"
00010 #include "request_ixl.h"
00011
00012 #include <l4/ixl/device.h>
00013 #include <l4/ixl/memory.h>
00014
00015 #include <optional>
00016
00021
00022 class Ixl_port : public Port_iface
00023 {
00024 public:
00025     static constexpr unsigned Tx_batch_size = 32;
00026     static constexpr unsigned Num_bufs = 1024;
00027     static constexpr unsigned Buf_size = 2048;
00028     static constexpr l4_uint64_t Max_mem_size = 1ULL << 28;
00029
00030     Ixl_port(Ixl::Ixl_device *dev)
00031     : Port_iface(dev->get_driver_name().c_str()),
00032       _dev(dev),
00033       _mempool(*_dev, Num_bufs, Buf_size, Max_mem_size)
00034     {
00035         Ixl::mac_address mac_addr = _dev->get_mac_addr();
00036         _mac = Mac_addr(reinterpret_cast<char const *>(mac_addr.addr));
00037         #if CONFIG_VNS_STATS
00038         _mac.to_array(_stats->mac);
00039         #endif
00040     }
00041
00042     // OPTIMIZE: Could use this information for rx batching, i.e. collect while
00043     //             rx_notify is disabled, then flush the collected buffers when
00044     //             rx_notify is enabled again.
00045     void rx_notify_disable_and_remember() override {}
00046     void rx_notify_emit_and_enable() override {}
00047     bool is_gone() const override { return false; }
00048
00050     bool tx_work_pending()
00051     {
00052         fetch_tx_requests();
00053         return _tx_batch_idx < _tx_batch_len;
00054     }
00055
00057     std::optional<Ixl_net_request> get_tx_request()
00058     {
00059         fetch_tx_requests();
00060         if (_tx_batch_idx < _tx_batch_len)
00061             return std::make_optional<Ixl_net_request>(_tx_batch[_tx_batch_idx++]);
00062         else
00063             return std::nullopt;
00064     }
00065
00066     Result handle_request(Port_iface *src_port, Net_transfer &src,
00067                          l4_uint64_t *bytes_transferred) override
00068     {
00069         Virtio_vlan_mangle mangle = create_vlan_mangle(src_port);
00070
00071         Dbg trace(Dbg::Request, Dbg::Trace, "REQ-IXL");
00072         trace.printf("%s: Transfer request %p.\n", _name, src.req_id());
00073
00074         struct Ixl::pkt_buf *buf = _mempool.pkt_buf_alloc();
00075         if (!buf)
00076         {
00077             trace.printf("\tTransfer failed, out-of-memory, dropping.\n");
00078             return Result::Dropped;
00079         }
00080
00081         // NOTE: Currently, the switch does not offer checksum or segmentation
00082         //         offloading to its l4virtio clients, so it is fine to simply ignore
00083         //         the Virtio_net::Hdr of the request here.
00084
00085         // Copy the request to the pkt_buf.
00086         Buffer dst_buf(reinterpret_cast<char *>(buf->data),
00087                      Buf_size - offsetof(Ixl::pkt_buf, data));
00088         unsigned max_size = Buf_size - offsetof(Ixl::pkt_buf, data);
00089         for (;;)
00090         {
00091             try
00092             {
00093                 if (src.done())
00094                     // Request completely copied to destination.
00095                     break;
00096             }

```

```

00097         catch (L4virtio::Svr::Bad_descriptor &e)
00098         {
00099             trace.printf("\tTransfer failed, bad descriptor exception, dropping.\n");
00100
00101             // Handle partial transfers to destination port.
00102             Ixl::pkt_buf_free(buf);
00103             throw;
00104         }
00105
00106         if (dst_buf.done())
00107         {
00108             trace.printf(
00109                 "\tTransfer failed, exceeds max packet-size, dropping.\n");
00110             Ixl::pkt_buf_free(buf);
00111             return Result::Dropped;
00112         }
00113
00114         auto &src_buf = src.cur_buf();
00115         trace.printf("\tCopying %p#%p:%u (%x) -> %p#%p:%u (%x)\n",
00116             src_port, src_buf.pos, src_buf.left, src_buf.left,
00117             static_cast<Port_iface *>(this),
00118             dst_buf.pos, dst_buf.left, dst_buf.left);
00119
00120         mangle.copy_pkt(dst_buf, src_buf);
00121     }
00122     buf->size = max_size - dst_buf.left;
00123     *bytes_transferred = buf->size;
00124
00125     // Enqueue the pkt_buf at the device.
00126     if (_dev->tx_batch(0, &buf, 1) == 1)
00127     {
00128         trace.printf("\tTransfer queued at device.\n");
00129         return Result::Delivered;
00130     }
00131     else
00132     {
00133         trace.printf("\tTransfer failed, dropping.\n");
00134         Ixl::pkt_buf_free(buf);
00135         return Result::Dropped;
00136     }
00137 }
00138
00139 Ixl::Ixl_device *dev() { return _dev; }
00140
00141 private:
00142 void fetch_tx_requests()
00143 {
00144     if (_tx_batch_idx < _tx_batch_len)
00145         // Previous batch not yet fully processed.
00146         return;
00147
00148     // Batch receive, then cache in member array, to avoid frequent interactions
00149     // with the hardware.
00150     _tx_batch_len = _dev->rx_batch(0, _tx_batch, Tx_batch_size);
00151     _tx_batch_idx = 0;
00152 }
00153
00154 Ixl::Ixl_device *_dev;
00155 Ixl::Mempool _mempool;
00156 Ixl::pkt_buf *_tx_batch[Tx_batch_size];
00157 unsigned _tx_batch_idx = 0;
00158 unsigned _tx_batch_len = 0;
00159 };
00160

```

16.60 port_l4virtio.h

```

00001 /*
00002  * Copyright (C) 2016-2017, 2020, 2022-2025 Kernkonzept GmbH.
00003  * Author(s): Jean Wolter <jean.wolter@kernkonzept.com>
00004  *            Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *            Georg Kotheimer <georg.kotheimer@kernkonzept.com>
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009 #pragma once
00010
00011 #include "port.h"
00012 #include "request_l4virtio.h"
00013 #include "virtio_net.h"
00014
00015 #include <l4/cxx/pair>
00016

```

```

00017 #include <vector>
00018
00023
00036 class L4virtio_port : public Port_iface, public Virtio_net
00037 {
00038 public:
00042 explicit L4virtio_port(unsigned vq_max, unsigned num_ds, char const *name,
00043                        l4_uint8_t const *mac)
00044 : Port_iface(name), Virtio_net(vq_max)
00045 {
00046     init_mem_info(num_ds);
00047
00048     Features hf = _dev_config.host_features(0);
00049     if (mac)
00050     {
00051         _mac = Mac_addr((char const *)mac);
00052         memcpy((void *)_dev_config.priv_config()->mac, mac,
00053               sizeof(_dev_config.priv_config()->mac));
00054
00055         hf.mac() = true;
00056         Dbg d(Dbg::Port, Dbg::Info);
00057         d.cprintf("%s: Adding Mac '", _name);
00058         _mac.print(d);
00059         d.cprintf("' to host features to %x\n", hf.raw);
00060     }
00061     _dev_config.host_features(0) = hf.raw;
00062     _dev_config.reset_hdr();
00063     Dbg(Dbg::Port, Dbg::Info)
00064         .printf("%s: Set host features to %x\n", _name,
00065               _dev_config.host_features(0));
00066 #if CONFIG_VNS_STATS
00067     _mac.to_array(_stats->mac);
00068 #endif
00069 }
00070
00071 void rx_notify_disable_and_remember() override
00072 {
00073     kick_disable_and_remember();
00074 }
00075
00076 void rx_notify_emit_and_enable() override
00077 {
00078     kick_emit_and_enable();
00079 }
00080
00081 bool is_gone() const override
00082 {
00083     return obj_cap() && !obj_cap().validate().label();
00084 }
00085
00087 bool tx_work_pending() const
00088 {
00089     return L4_LIKELY(tx_q()->ready()) && tx_q()->desc_avail();
00090 }
00091
00093 std::optional<Virtio_net_request> get_tx_request()
00094 {
00095     return Virtio_net_request::get_request(this, tx_q());
00096 }
00097
00103 void drop_requests()
00104 { Virtio_net_request::drop_requests(this, tx_q()); }
00105
00106 Result handle_request(Port_iface *src_port, Net_transfer &src,
00107                     l4_uint64_t *bytes_transferred) override
00108 {
00109     Virtio_vlan_mangle mangle = create_vlan_mangle(src_port);
00110
00111     Dbg trace(Dbg::Request, Dbg::Trace, "REQ-VIO");
00112     trace.printf("%s: Transfer request %p.\n", _name, src.req_id());
00113
00114     Buffer dst;
00115     int total = 0;
00116     l4_uint16_t num_merged = 0;
00117     l4_uint64_t total_merged = 0;
00118     typedef cxx::Pair<L4virtio::Svr::Virtqueue::Head_desc, l4_uint32_t> Consumed_entry;
00119     std::vector<Consumed_entry> consumed;
00120
00121     Virtio_net *dst_dev = this;
00122     Virtqueue *dst_queue = rx_q();
00123     L4virtio::Svr::Virtqueue::Head_desc dst_head;
00124     L4virtio::Svr::Request_processor dst_req_proc;
00125     Virtio_net::Hdr *dst_header = nullptr;
00126
00127     for (;;)
00128     {
00129         try

```

```

00130         {
00131             if (src.done())
00132                 // Request completely copied to destination.
00133                 break;
00134         }
00135     catch (L4virtio::Svr::Bad_descriptor &e)
00136     {
00137         trace.printf("\tTransfer failed, bad descriptor exception, dropping.\n");
00138
00139         // Handle partial transfers to destination port.
00140         if (!consumed.empty())
00141             // Partial transfer, rewind to before first descriptor of transfer.
00142             dst_queue->rewind_avail(consumed.at(0).first);
00143         else if (dst_head)
00144             // Partial transfer, still at first _dst_head.
00145             dst_queue->rewind_avail(dst_head);
00146         throw;
00147     }
00148
00149     /* The source data structures are already initialized, the header
00150     is consumed and src stands at the very first real buffer.
00151     Initialize the target data structures if necessary and fill the
00152     header. */
00153     if (!dst_head)
00154     {
00155         if (!dst_queue->ready())
00156             return Result::Dropped;
00157
00158         auto r = dst_queue->next_avail();
00159
00160         if (L4_UNLIKELY(!r))
00161         {
00162             trace.printf("\tTransfer failed, destination queue depleted, dropping.\n");
00163             // Abort incomplete transfer.
00164             if (!consumed.empty())
00165                 dst_queue->rewind_avail(consumed.front().first);
00166             return Result::Dropped;
00167         }
00168
00169         try
00170         {
00171             dst_head = dst_req_proc.start(dst_dev->mem_info(), r, &dst);
00172         }
00173     catch (L4virtio::Svr::Bad_descriptor &e)
00174     {
00175         Dbg(Dbg::Request, Dbg::Warn, "REQ")
00176             .printf("%s: bad descriptor exception: %s - %i"
00177                 " -- signal device error in destination device %p.\n",
00178                 __PRETTY_FUNCTION__, e.message(), e.error, dst_dev);
00179
00180         dst_dev->device_error();
00181         return Result::Exception; // Must not touch the dst queues anymore.
00182     }
00183
00184     if (!dst_header)
00185     {
00186         if (dst.left < sizeof(Virtio_net::Hdr))
00187             throw L4::Runtime_error(-L4_EINVAL,
00188                 "Target buffer too small for header");
00189         dst_header = reinterpret_cast<Virtio_net::Hdr *>(dst.pos);
00190         trace.printf("\tCopying header to %p (size: %u)\n",
00191             dst.pos, dst.left);
00192
00193         /*
00194          * Header and csum offloading/general segmentation offloading
00195          *
00196          * We just copy the original header from source to
00197          * destination and have to consider three different
00198          * cases:
00199          * - no flags are set
00200          *   - we got a packet that is completely checksummed
00201          *     and correctly fragmented, there is nothing to
00202          *     do other then copying.
00203          * - virtio_net_hdr_f_needs_csum set
00204          *   - the packet is partially checksummed; if we would
00205          *     send the packet out on the wire we would have
00206          *     to calculate checksums now. But here we rely on
00207          *     the ability of our guest to handle partially
00208          *     checksummed packets and simply delegate the
00209          *     checksum calculation to them.
00210          * - gso_type != gso_none
00211          *   - the packet needs to be segmented; if we would
00212          *     send it out on the wire we would have to
00213          *     segment it now. But again we rely on the
00214          *     ability of our guest to handle gso
00215          *
00216          * We currently assume that our guests negotiated
00217          * virtio_net_f_guest_*, this needs to be checked in

```

```

00217         * the future.
00218         *
00219         * We also discussed the usage of
00220         * virtio_net_hdr_f_data_valid to remove the need to
00221         * checksum packets at all. But since our clients send
00222         * partially checksummed packets anyway the only
00223         * interesting case would be a packet without
00224         * net_hdr_f_needs_checksum set. In that case we would
00225         * signal that we checked the checksum and the
00226         * checksum is actually correct. Since we do not know
00227         * the origin of the packet (it could have been send
00228         * by an external node and could have been routed to
00229         * u) we can not signal this without actually
00230         * verifying the checksum. Otherwise a packet with an
00231         * invalid checksum could be successfully delivered.
00232         */
00233         total = sizeof(Virtio_net::Hdr);
00234         src.copy_header(dst_header);
00235         mangle.rewrite_hdr(dst_header);
00236         dst.skip(total);
00237     }
00238     ++num_merged;
00239 }
00240
00241 bool has_dst_buffer = !dst.done();
00242 if (!has_dst_buffer)
00243     try
00244     {
00245         // The current dst buffer is full, try to get next chained buffer.
00246         has_dst_buffer = dst_req_proc.next(dst_dev->mem_info(), &dst);
00247     }
00248     catch (L4virtio::Svr::Bad_descriptor &e)
00249     {
00250         Dbg(Dbg::Request, Dbg::Warn, "REQ")
00251         .printf("%s: bad descriptor exception: %s - %i"
00252             " -- signal device error in destination device %p.\n",
00253             __PRETTY_FUNCTION__, e.message(), e.error, dst_dev);
00254         dst_dev->device_error();
00255         return Result::Exception; // Must not touch the dst queues anymore.
00256     }
00257
00258 if (has_dst_buffer)
00259 {
00260     auto &src_buf = src.cur_buf();
00261     trace.printf("\tCopying %p#%p:%u (%x) -> %p#%p:%u (%x)\n",
00262         src_port, src_buf.pos, src_buf.left, src_buf.left,
00263         static_cast<Port_iface*>(this),
00264         dst.pos, dst.left, dst.left);
00265
00266     total += mangle.copy_pkt(dst, src_buf);
00267 }
00268 else if (negotiated_features().mrg_rxbuf())
00269 {
00270     // save descriptor information for later
00271     trace.printf("\tSaving descriptor for later\n");
00272     consumed.push_back(Consumed_entry(dst_head, total));
00273     total_merged += total;
00274     total = 0;
00275     dst_head = L4virtio::Svr::Virtqueue::Head_desc();
00276 }
00277 else
00278 {
00279     trace.printf("\tTransfer failed, destination buffer too small, dropping.\n");
00280     // Abort incomplete transfer.
00281     dst_queue->rewind_avail(dst_head);
00282     return Result::Dropped;
00283 }
00284 }
00285
00286 /*
00287 * Finalize the Request delivery. Call `finish()` on the destination
00288 * port's receive queue, which will result in triggering the destination
00289 * client IRQ.
00290 */
00291 if (!dst_header)
00292 {
00293     if (!total)
00294         trace.printf("\tTransfer - not started yet, dropping\n");
00295     return Result::Dropped;
00296 }
00297
00298 if (consumed.empty())
00299 {
00300     assert(dst_head);
00301     assert(num_merged == 1);
00302     trace.printf("\tTransfer - Invoke dst_queue->finish()\n");

```

```

00304         dst_header->num_buffers = 1;
00305         dst_queue->finish(dst_head, dst_dev, total);
00306         *bytes_transferred = total;
00307     }
00308     else
00309     {
00310         assert(dst_head);
00311         dst_header->num_buffers = num_merged;
00312         consumed.push_back(Consumed_entry(dst_head, total));
00313         trace.printf("\tTransfer - Invoke dst_queue->finish(iter)\n");
00314         *bytes_transferred = total + total_merged;
00315         dst_queue->finish(consumed.begin(), consumed.end(), dst_dev);
00316     }
00317     return Result::Delivered;
00318 }
00319 };
00320

```

16.61 request.h

```

00001 /*
00002  * Copyright (C) 2016-2017, 2020, 2022, 2024-2025 Kernkonzept GmbH.
00003  * Author(s): Jean Wolter <jean.wolter@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include "mac_addr.h"
00010 #include "virtio_net.h"
00011 #include "virtio_net_buffer.h"
00012 #include "vlan.h"
00013
00014 #include <l4/l4virtio/server/virtio>
00015
00016
00021
00033 class Net_transfer
00034 {
00035 public:
00036     virtual ~Net_transfer() = default;
00037
00041     void const *req_id() const { return _req_id; }
00042
00046     virtual void copy_header(Virtio_net::Hdr *dst_header) const = 0;
00047
00054     Buffer &cur_buf() { return _cur_buf; }
00055
00065     virtual bool done() = 0;
00066
00067 protected:
00068     Buffer _cur_buf;
00069     void const *_req_id;
00070 };
00071
00072 class Net_request
00073 {
00074 public:
00075
00076     bool has_vlan() const
00077     {
00078         if (!_pkt.pos || _pkt.left < 14)
00079             return false;
00080
00081         uint8_t *p = reinterpret_cast<uint8_t *>(_pkt.pos);
00082         return p[12] == 0x81U && p[13] == 0x00U;
00083     }
00084
00085     uint16_t vlan_id() const
00086     {
00087         if (!has_vlan() || _pkt.left < 16)
00088             return VLAN_ID_NATIVE;
00089
00090         uint8_t *p = reinterpret_cast<uint8_t *>(_pkt.pos);
00091         return (uint16_t){p[14]} << 8 | p[15] & 0xffU;
00092     }
00093
00105     uint8_t const *buffer(size_t *size) const
00106     {
00107         *size = _pkt.left;
00108         return reinterpret_cast<uint8_t const *>(_pkt.pos);
00109     }
00110

```

```

00111 void dump_pkt() const
00112 {
00113     Dbg pkt_debug(Dbg::Packet, Dbg::Debug, "PKT");
00114     if (pkt_debug.is_active())
00115     {
00116         //pkt_debug.cprintf("\t");
00117         //src_mac().print(pkt_debug);
00118         //pkt_debug.cprintf(" -> ");
00119         //dst_mac().print(pkt_debug);
00120         //pkt_debug.cprintf("\n");
00121
00122         Dbg pkt_trace(Dbg::Packet, Dbg::Trace, "PKT");
00123         if (pkt_trace.is_active() && _pkt.left >= 14)
00124         {
00125             uint8_t const *packet = reinterpret_cast<uint8_t const *>(_pkt.pos);
00126             pkt_trace.cprintf("\n\tEthertype: ");
00127             uint16_t ether_type = uint16_t{packet[12]} << 8 | packet[13];
00128             char const *protocol;
00129             switch (ether_type)
00130             {
00131                 case 0x0800: protocol = "IPv4"; break;
00132                 case 0x0806: protocol = "ARP"; break;
00133                 case 0x8100: protocol = "Vlan"; break;
00134                 case 0x86dd: protocol = "IPv6"; break;
00135                 case 0x8863: protocol = "PPPoE Discovery"; break;
00136                 case 0x8864: protocol = "PPPoE Session"; break;
00137                 default: protocol = nullptr;
00138             }
00139             if (protocol)
00140                 pkt_trace.cprintf("%s\n", protocol);
00141             else
00142                 pkt_trace.cprintf("%04x\n", ether_type);
00143         }
00144     }
00145 }
00146
00147 protected:
00148     Buffer _pkt;
00149 };
00150

```

16.62 request_ixl.h

```

00001 /*
00002  * Copyright (C) 2024-2025 Kernkonzept GmbH.
00003  * Author(s): Georg Kotheimer <georg.kotheimer@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include "port.h"
00010 #include "request.h"
00011
00012 #include <14/ixl/memory.h>
00013
00014 #include <utility>
00015
00020
00021 class Ixl_net_request final : public Net_request
00022 {
00023 public:
00024     class Ixl_net_transfer final : public Net_transfer
00025     {
00026     public:
00027         explicit Ixl_net_transfer(Ixl_net_request const &request)
00028             : _request(request)
00029         {
00030             _cur_buf = Buffer(reinterpret_cast<char *>(request.buf()->data),
00031                             request.buf()->size);
00032             _req_id = _request.buf();
00033         }
00034
00035         // delete copy constructor and copy assignment operator
00036         Ixl_net_transfer(Ixl_net_transfer const &) = delete;
00037         Ixl_net_transfer &operator = (Ixl_net_transfer const &) = delete;
00038
00039         void copy_header(Virtio_net::Hdr *dst_header) const override
00040         {
00041             dst_header->flags.data_valid() = 0;
00042             dst_header->flags.need_csum() = 0;
00043             dst_header->gso_type = 0; // GSO_NONE
00044             dst_header->hdr_len = sizeof(Virtio_net::Hdr);
00045         }
00046     };
00047 };

```



```

00045     dst_header->gso_size = 0;
00046     dst_header->csum_start = 0;
00047     dst_header->csum_offset = 0;
00048     dst_header->num_buffers = 1;
00049 }
00050
00051     bool done() override { return _cur_buf.done(); }
00052
00053 private:
00054     Ixl_net_request const &_request;
00055 };
00056
00057 void dump_request(Port_iface *port) const
00058 {
00059     Dbg debug(Dbg::Request, Dbg::Debug, "REQ-IXL");
00060     if (debug.is_active())
00061     {
00062         debug.printf("%s: Next packet: %p - %x bytes\n",
00063             port->get_name(), _pkt.pos, _pkt.left);
00064     }
00065     dump_pkt();
00066 }
00067
00068 explicit Ixl_net_request(Ixl::pkt_buf *buf) : _buf(buf)
00069 {
00070     _pkt = Buffer(reinterpret_cast<char *>(buf->data), buf->size);
00071 }
00072
00073 // delete copy constructor and copy assignment operator
00074 Ixl_net_request(Ixl_net_request const &) = delete;
00075 Ixl_net_request &operator=(Ixl_net_request const &) = delete;
00076
00077 // define move constructor and copy assignment operator
00078 Ixl_net_request(Ixl_net_request &&other)
00079 : _buf(other._buf)
00080 {
00081     _pkt = std::move(other._pkt);
00082
00083     // Invalidate other.
00084     other._buf = nullptr;
00085 }
00086
00087 Ixl_net_request &operator=(Ixl_net_request &&other)
00088 {
00089     // Invalidate self.
00090     if (_buf != nullptr)
00091         Ixl::pkt_buf_free(_buf);
00092
00093     _buf = other._buf;
00094     _pkt = std::move(other._pkt);
00095
00096     // Invalidate other.
00097     other._buf = nullptr;
00098
00099     return *this;
00100 }
00101
00102 ~Ixl_net_request()
00103 {
00104     if (_buf != nullptr)
00105     {
00106         Ixl::pkt_buf_free(_buf);
00107         _buf = nullptr;
00108     }
00109 }
00110
00112 Mac_addr dst_mac() const
00113 {
00114     return (_pkt.pos && _pkt.left >= Mac_addr::Addr_length)
00115         ? Mac_addr::from_uncached(_pkt.pos)
00116         : Mac_addr(Mac_addr::Addr_unknown);
00117 }
00118
00120 Mac_addr src_mac() const
00121 {
00122     return (_pkt.pos && _pkt.left >= Mac_addr::Addr_length * 2)
00123         ? Mac_addr::from_uncached(_pkt.pos + Mac_addr::Addr_length)
00124         : Mac_addr(Mac_addr::Addr_unknown);
00125 }
00126
00127 Ixl::pkt_buf *buf() const { return _buf; }
00128
00129 Ixl_net_transfer transfer_src() const
00130 { return Ixl_net_transfer(*this); }
00131
00132 private:
00133     Ixl::pkt_buf *_buf;

```



```

00092         _header->flags.raw,
00093         _header->gso_type, _header->hdr_len,
00094         _header->gso_size,
00095         _header->csum_start, _header->csum_offset,
00096         _header->num_buffers);
00097     }
00098 }
00099 dump_pkt();
00100 }
00101
00102 // delete copy constructor and copy assignment operator
00103 Virtio_net_request(Virtio_net_request const &) = delete;
00104 Virtio_net_request &operator = (Virtio_net_request const &) = delete;
00105
00106 // define move constructor and copy assignment operator
00107 Virtio_net_request(Virtio_net_request &&other)
00108 : _dev(other._dev),
00109   _queue(other._queue),
00110   _head(std::move(other._head)),
00111   _req_proc(std::move(other._req_proc)),
00112   _header(other._header)
00113 {
00114     _pkt = std::move(other._pkt);
00115
00116     // Invalidate other.
00117     other._queue = nullptr;
00118 }
00119
00120 Virtio_net_request &operator = (Virtio_net_request &&other)
00121 {
00122     // Invalidate self.
00123     finish();
00124
00125     _dev = other._dev;
00126     _queue = other._queue;
00127     _head = std::move(other._head);
00128     _req_proc = std::move(other._req_proc);
00129     _header = other._header;
00130     _pkt = std::move(other._pkt);
00131
00132     // Invalidate other.
00133     other._queue = nullptr;
00134
00135     return *this;
00136 }
00137
00138 Virtio_net_request(Virtio_net *dev, L4virtio::Svr::Virtqueue *queue,
00139                   L4virtio::Svr::Virtqueue::Request const &req)
00140 : _dev(dev), _queue(queue)
00141 {
00142     _head = _req_proc.start(_dev->mem_info(), req, &_pkt);
00143
00144     _header = (Virtio_net::Hdr *)_pkt.pos;
00145     l4_uint32_t skipped = _pkt.skip(sizeof(Virtio_net::Hdr));
00146
00147     if (L4_UNLIKELY( (skipped != sizeof(Virtio_net::Hdr))
00148                     || (_pkt.done() && !_next_buffer(&_pkt))))
00149     {
00150         _header = 0;
00151         Dbg(Dbg::Queue, Dbg::Warn).printf("Invalid request\n");
00152         return;
00153     }
00154 }
00155
00156 ~Virtio_net_request()
00157 { finish(); }
00158
00159 bool valid() const
00160 { return _header != 0; }
00161
00172 static void drop_requests(Virtio_net *dev,
00173                           L4virtio::Svr::Virtqueue *queue)
00174 {
00175     if (L4_UNLIKELY(!queue->ready()))
00176         return;
00177
00178     if (queue->desc_avail())
00179         Dbg(Dbg::Request, Dbg::Debug)
00180             .printf("Dropping incoming packets on monitor port\n");
00181
00182     L4virtio::Svr::Request_processor req_proc;
00183     Buffer pkt;
00184
00185     while (auto req = queue->next_avail())
00186     {
00187         auto head = req_proc.start(dev->mem_info(), req, &pkt);
00188         queue->finish(head, dev, 0);
00189     }

```

```

00189     }
00190 }
00191
00192 static std::optional<Virtio_net_request>
00193 get_request(Virtio_net *dev, L4virtio::Svr::Virtqueue *queue)
00194 {
00195     if (L4_UNLIKELY(!queue->ready()))
00196         return std::nullopt;
00197
00198     if (auto r = queue->next_avail())
00199     {
00200         // Virtio_net_request keeps "a lot of internal state",
00201         // therefore we create the object before creating the
00202         // state.
00203         // We might check later on whether it is possible to
00204         // save the state when we actually have to because a
00205         // transfer is blocking on a port.
00206         auto request = Virtio_net_request(dev, queue, r);
00207         if (request.valid())
00208             return request;
00209     }
00210     return std::nullopt;
00211 }
00212
00213 Buffer const &first_buffer() const
00214 { return _pkt; }
00215
00216 Virtio_net::Hdr const *header() const
00217 { return _header; }
00218
00219 L4virtio::Svr::Request_processor const &get_request_processor() const
00220 { return _req_proc; }
00221
00222 Virtio_net const *dev() const
00223 { return _dev; }
00224
00225 Virtio_net_transfer transfer_src() const
00226 { return Virtio_net_transfer(*this); }
00227
00228 Mac_addr dst_mac() const
00229 {
00230     return (_pkt.pos && _pkt.left >= Mac_addr::Addr_length)
00231         ? Mac_addr(_pkt.pos)
00232         : Mac_addr(Mac_addr::Addr_unknown);
00233 }
00234
00235 Mac_addr src_mac() const
00236 {
00237     return (_pkt.pos && _pkt.left >= Mac_addr::Addr_length * 2)
00238         ? Mac_addr(_pkt.pos + Mac_addr::Addr_length)
00239         : Mac_addr(Mac_addr::Addr_unknown);
00240 }
00241
00242 private:
00243     /* needed for Virtqueue::finish() */
00244     Virtio_net *_dev;
00245     L4virtio::Svr::Virtqueue *_queue;
00246     L4virtio::Svr::Virtqueue::Head_desc _head;
00247
00248     /* the actual request processor, encapsulates the decoding of the request */
00249     L4virtio::Svr::Request_processor _req_proc;
00250
00251     /* A request to the virtio net layer consists of one or more buffers
00252     containing the Virtio_net::Hdr and the actual packet. To make a
00253     switching decision we need to be able to look at the packet while
00254     still being able access the Virtio_net::Hdr for the actual copy
00255     operation. Therefore we keep track of two locations, the header
00256     location and the start of the packet (which might be in a
00257     different buffer) */
00258     Virtio_net::Hdr *_header;
00259
00260     bool _next_buffer(Buffer *buf)
00261     { return _req_proc.next(_dev->mem_info(), buf); }
00262
00263     void finish()
00264     {
00265         if (_queue == nullptr || !_queue->ready())
00266             return;
00267
00268         Dbg(Dbg::Virtio, Dbg::Trace).printf("%s(%p)\n", __PRETTY_FUNCTION__, this);
00269         _queue->finish(_head, _dev, 0);
00270         _queue = nullptr;
00271     }
00272 };
00273
00274
00275

```

16.64 stats.h

```

00001 #include <l4/re/env>
00002 #include <l4/re/dataspace>
00003 #include <l4/re/error_helper>
00004 #include <l4/re/util/cap_alloc>
00005 #include <l4/virtio-net-switch/stats.h>
00006
00007 class Switch_statistics
00008 {
00009 private:
00010     L4Re::Util::Ref_cap<L4Re::Dataspace>::_Cap _ds;
00011     Virtio_net_switch::Statistics *_stats;
00012     bool _initialized = false;
00013
00014     Switch_statistics() {}
00015
00016     ~Switch_statistics()
00017     {
00018         if (_initialized)
00019             L4Re::Env::env()->rm()->detach(reinterpret_cast<l4_addr_t>(_stats), 0);
00020     }
00021
00022     l4_size_t _size;
00023
00024 public:
00025     Virtio_net_switch::Statistics *stats()
00026     {
00027         if (_initialized)
00028             return _stats;
00029         else
00030             throw L4::Runtime_error(-L4_EAGAIN, "Statistics not set up.");
00031     }
00032
00033     static Switch_statistics& get_instance()
00034     {
00035         static Switch_statistics instance;
00036         return instance;
00037     }
00038
00039     void initialize(l4_uint64_t num_max_ports)
00040     {
00041         _size = l4_round_page(sizeof(Virtio_net_switch::Statistics)
00042                                + sizeof(Virtio_net_switch::Port_statistics) * num_max_ports);
00043         void *addr = malloc(_size);
00044         if (!addr)
00045             throw L4::Runtime_error(-L4_ENOMEM,
00046                                     "Could not allocate statistics memory.");
00047
00048         memset(addr, 0, _size);
00049         _stats = reinterpret_cast<Virtio_net_switch::Statistics *>(addr);
00050         _initialized = true;
00051         _stats->max_ports = num_max_ports;
00052     }
00053
00054     Virtio_net_switch::Port_statistics *
00055     allocate_port_statistics(char const* name)
00056     {
00057         for (unsigned i = 0; i < _stats->max_ports; ++i)
00058         {
00059             if (!_stats->port_stats[i].in_use)
00060             {
00061                 memset(reinterpret_cast<void*>(&_stats->port_stats[i]), 0,
00062                         sizeof(Virtio_net_switch::Port_statistics));
00063                 _stats->port_stats[i].in_use = 1;
00064                 size_t len = std::min(strlen(name), sizeof(_stats->port_stats[i].name) - 1);
00065                 memcpy(_stats->port_stats[i].name, name, len);
00066                 _stats->port_stats[i].name[len] = '\0';
00067                 _stats->age++;
00068                 return &_stats->port_stats[i];
00069             }
00070         }
00071         return nullptr;
00072     }
00073
00074     inline l4_size_t size()
00075     { return _size; }
00076
00077     Switch_statistics(Switch_statistics const&) = delete;
00078     void operator=(Switch_statistics const &) = delete;
00079 };

```

16.65 switch.cc

```

00001 /*
00002  * Copyright (C) 2016-2018, 2020, 2023-2025 Kernkonzept GmbH.
00003  * Author(s): Jean Wolter <jean.wolter@kernkonzept.com>
00004  *             Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #include "debug.h"
00009 #include "switch.h"
00010 #include "filter.h"
00011
00012 Virtio_switch::Virtio_switch(unsigned max_ports)
00013 : _max_ports(max_ports)
00014 {}
00015
00016 bool
00017 Virtio_switch::add_port(Port_iface *port)
00018 {
00019     if (_ports.size() == _max_ports)
00020     {
00021         Dbg(Dbg::Port, Dbg::Warn)
00022             .printf("Port limit (%u) has been reached.\n", _max_ports);
00023         return false;
00024     }
00025
00026     if (!port->mac().is_unknown())
00027         for (auto const *p: _ports)
00028             if (p->mac() == port->mac())
00029             {
00030                 Dbg(Dbg::Port, Dbg::Warn)
00031                     .printf("Rejecting port '%s'. MAC address already in use.\n",
00032                             port->get_name());
00033                 return false;
00034             }
00035
00036     _ports.push_back(port);
00037     return true;
00038 }
00039
00040 bool
00041 Virtio_switch::add_monitor_port(Port_iface *port)
00042 {
00043     if (!_monitor)
00044     {
00045         _monitor = port;
00046         return true;
00047     }
00048
00049     Dbg(Dbg::Port, Dbg::Warn).printf("'%' already defined as monitor port,"
00050                                     " rejecting monitor port '%s'\n",
00051                                     _monitor->get_name(), port->get_name());
00052     return false;
00053 }
00054
00055 void
00056 Virtio_switch::check_ports()
00057 {
00058     for (std::vector<Port_iface *>::iterator it = _ports.begin();
00059          it != _ports.end();)
00060     {
00061         auto *port = *it;
00062         if (port->is_gone())
00063         {
00064             Dbg(Dbg::Port, Dbg::Info)
00065                 .printf("Client on port %p has gone. Deleting...\n", port);
00066
00067             _mac_table.flush(port);
00068             it = _ports.erase(it);
00069             delete port;
00070         }
00071         else
00072             ++it;
00073     }
00074
00075     if (_monitor && _monitor->is_gone())
00076     {
00077         delete(_monitor);
00078         _monitor = nullptr;
00079     }
00080 }
00081
00082 template<typename REQ>
00083 void
00084 Virtio_switch::handle_tx_request(Port_iface *port, REQ const &request)

```

```

00085 {
00086     // Trunk ports are required to have a VLAN tag and only accept packets that
00087     // belong to a configured VLAN.
00088     if (port->is_trunk() && !port->match_vlan(request.vlan_id()))
00089     {
00090         // Drop packet.
00091         port->stat_inc_tx_dropped();
00092         return;
00093     }
00094
00095     // Access ports must not be VLAN tagged to prevent double tagging attacks.
00096     if (port->is_access() && request.has_vlan())
00097     {
00098         // Drop packet.
00099         port->stat_inc_tx_dropped();
00100         return;
00101     }
00102
00103     auto handle_request = [](Port_iface *dst_port, Port_iface *src_port,
00104                             REQ const &req)
00105     {
00106         auto transfer_src = req.transfer_src();
00107         l4_uint64_t bytes;
00108         auto res = dst_port->handle_request(src_port, transfer_src, &bytes);
00109         switch (res)
00110         {
00111             case Port_iface::Result::Delivered:
00112                 dst_port->stat_inc_tx_num();
00113                 dst_port->stat_inc_tx_bytes(bytes);
00114                 src_port->stat_inc_rx_num();
00115                 src_port->stat_inc_rx_bytes(bytes);
00116                 break;
00117             case Port_iface::Result::Dropped:
00118                 [[fallthrough]];
00119             case Port_iface::Result::Exception:
00120                 [[fallthrough]];
00121             default:
00122                 dst_port->stat_inc_tx_dropped();
00123                 break;
00124         }
00125     };
00126
00127     Mac_addr src = request.src_mac();
00128
00129     auto dst = request.dst_mac();
00130     bool is_broadcast = dst.is_broadcast();
00131     uint16_t vlan = request.has_vlan() ? request.vlan_id() : port->get_vlan();
00132     _mac_table.learn(src, port, vlan);
00133     if (L4_LIKELY(!is_broadcast))
00134     {
00135         auto *target = _mac_table.lookup(dst, vlan);
00136         if (target)
00137         {
00138             // Do not send packets to the port they came in; they might
00139             // be sent to us by another switch which does not know how
00140             // to reach the target.
00141             if (target != port)
00142             {
00143                 handle_request(target, port, request);
00144                 if (_monitor && !filter_request(request))
00145                     handle_request(_monitor, port, request);
00146             }
00147             return;
00148         }
00149     }
00150
00151     // It is either a broadcast or an unknown destination - send to all
00152     // known ports except the source port
00153     for (auto *target: _ports)
00154     {
00155         if (target != port && target->match_vlan(vlan))
00156             handle_request(target, port, request);
00157     }
00158
00159     // Send a copy to the monitor port
00160     if (_monitor && !filter_request(request))
00161         handle_request(_monitor, port, request);
00162 }
00163
00164 template<typename PORT>
00165 void
00166 Virtio_switch::handle_tx_requests(PORT *port, unsigned &num_reqs_handled)
00167 {
00168     while (auto req = port->get_tx_request())
00169     {
00170         req->dump_request(port);
00171         handle_tx_request(port, *req);

```

```

00172
00173     if (++num_reqs_handled >= Tx_burst)
00174         // Port has hit its TX burst limit.
00175         break;
00176     }
00177 }
00178
00179 bool
00180 Virtio_switch::handle_l4virtio_port_tx(L4virtio_port *port)
00181 {
00182     /* handle IRQ on one port for the time being */
00183     if (!port->tx_work_pending())
00184         Dbg(Dbg::Port, Dbg::Debug)
00185             .printf("%s: Irq without pending work\n", port->get_name());
00186
00187     unsigned num_reqs_handled = 0;
00188     do
00189     {
00190         port->tx_q()->disable_notify();
00191         port->rx_q()->disable_notify();
00192
00193         if (num_reqs_handled >= Tx_burst)
00194         {
00195             Dbg(Dbg::Port, Dbg::Debug)
00196                 .printf(
00197                     "%s: Tx burst limit hit, reschedule remaining Tx work.\n",
00198                     port->get_name());
00199
00200             // Port has hit its TX burst limit, so for fairness reasons, stop
00201             // processing TX work from this port, and instead reschedule the
00202             // pending work for later.
00203             port->reschedule_pending_tx();
00204             // NOTE: Notifications for this port remain disabled, until eventually
00205             // the reschedule handler calls `handle_l4virtio_port_tx` again.
00206             return false;
00207         }
00208
00209         // Within the loop, to trigger before enabling notifications again.
00210         all_rx_notify_disable_and_remember();
00211
00212         try
00213         {
00214             // throws Bad_descriptor exceptions raised on SRC port
00215             handle_tx_requests(port, num_reqs_handled);
00216         }
00217         catch (L4virtio::Svr::Bad_descriptor &e)
00218         {
00219             Dbg(Dbg::Port, Dbg::Warn, "REQ")
00220                 .printf("%s: caught bad descriptor exception: %s - %i"
00221                     " -- Signal device error on device %p.\n",
00222                     __PRETTY_FUNCTION__, e.message(), e.error, port);
00223             port->device_error();
00224             all_rx_notify_emit_and_enable();
00225             return false;
00226         }
00227
00228         all_rx_notify_emit_and_enable();
00229
00230         port->tx_q()->enable_notify();
00231         port->rx_q()->enable_notify();
00232
00233         L4virtio::wmb();
00234         L4virtio::rmb();
00235     }
00236     while (port->tx_work_pending());
00237
00238     return true;
00239 }
00240
00241 #if CONFIG_VNS_IXL
00242 bool
00243 Virtio_switch::handle_ixl_port_tx(Ixl_port *port)
00244 {
00245     unsigned num_reqs_handled = 0;
00246
00247     all_rx_notify_disable_and_remember();
00248     handle_tx_requests(port, num_reqs_handled);
00249     all_rx_notify_emit_and_enable();
00250
00251     if (num_reqs_handled >= Tx_burst && port->tx_work_pending())
00252     {
00253         Dbg(Dbg::Port, Dbg::Info)
00254             .printf("%s: Tx burst limit hit, reschedule remaining Tx work.\n",
00255                 port->get_name());
00256
00257         // Port has hit its TX burst limit, so for fairness reasons, stop
00258         // processing TX work from this port, and instead reschedule the

```



```

00259         // pending work for later.
00260         port->reschedule_pending_tx();
00261         return false;
00262     }
00263
00264     return true;
00265 }
00266 #endif
00267

```

16.66 switch.h

```

00001 /*
00002  * Copyright (C) 2016-2017, 2020, 2022-2025 Kernkonzept GmbH.
00003  * Author(s): Jean Wolter <jean.wolter@kernkonzept.com>
00004  *             Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include "port.h"
00011 #include "port_l4virtio.h"
00012 #include "mac_table.h"
00013
00014 #if CONFIG_VNS_IXL
00015 #include "port_ixl.h"
00016 #endif
00017
00018 #include <vector>
00019
00035 class Virtio_switch
00036 {
00037 private:
00038     std::vector<Port_iface *> _ports;
00039     Port_iface *_monitor = nullptr;
00040
00041     unsigned _max_ports;
00042     Mac_table<> _mac_table;
00043
00044     // Limits the number of consecutive TX requests a port can process before
00045     // being interrupted to ensure fairness to other ports.
00046     static constexpr unsigned Tx_burst = 128;
00047
00057     template<typename REQ>
00058     void handle_tx_request(Port_iface *port, REQ const &request);
00059
00060     template<typename PORT>
00061     void handle_tx_requests(PORT *port, unsigned &num_reqs_handled);
00062
00063
00064     void all_rx_notify_emit_and_enable()
00065     {
00066         for (auto *port : _ports)
00067             port->rx_notify_emit_and_enable();
00068     }
00069
00070     void all_rx_notify_disable_and_remember()
00071     {
00072         for (auto *port: _ports)
00073             port->rx_notify_disable_and_remember();
00074     }
00075
00076 public:
00083     explicit Virtio_switch(unsigned max_ports);
00084
00093     bool add_port(Port_iface *port);
00094
00103     bool add_monitor_port(Port_iface *port);
00104
00112     void check_ports();
00113
00123     bool handle_l4virtio_port_tx(L4virtio_port *port);
00124
00125 #if CONFIG_VNS_IXL
00135     bool handle_ixl_port_tx(Ixl_port *port);
00136 #endif
00137
00146     bool port_available(bool monitor)
00147     {
00148         if (monitor)
00149             return _monitor == 0;
00150

```

```

00151     return _ports.size() < _max_ports;
00152 }
00153 };

```

16.67 virtio_net.h

```

00001 /*
00002  * Copyright (C) 2016-2017, 2019, 2022-2025 Kernkonzept GmbH.
00003  * Author(s): Jean Wolter <jean.wolter@kernkonzept.com>
00004  *           Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include <l4/re/dataspace>
00011 #include <l4/re/util/unique_cap>
00012
00013 #include <l4/sys/cxx/ipc_epiface>
00014
00015 #include <l4/l4virtio/server/virtio>
00016 #include <l4/l4virtio/server/l4virtio>
00017 #include <l4/l4virtio/l4virtio>
00018
00019 #include "debug.h"
00024 class Virtqueue : public L4virtio::Svr::Virtqueue
00025 {
00026 public:
00027     bool kick_queue()
00028     {
00029         if (no_notify_guest())
00030             return false;
00031
00032         if (_do_kick)
00033             return true;
00034
00035         _kick_pending = true;
00036         return false;
00037     }
00038
00039     bool kick_enable_get_pending()
00040     {
00041         _do_kick = true;
00042         return _kick_pending;
00043     }
00044
00045     void kick_disable_and_remember()
00046     {
00047         _do_kick = false;
00048         _kick_pending = false;
00049     }
00050
00051 private:
00052     bool _do_kick = true;
00053     bool _kick_pending = false;
00054 };
00055
00071 class Virtio_net :
00072     public L4virtio::Svr::Device,
00073     public L4::Epiface_t<Virtio_net, L4virtio::Device>
00074 {
00075 public:
00076     struct Hdr_flags
00077     {
00078         l4_uint8_t raw;
00079         CXX_BITFIELD_MEMBER( 0, 0, need_csum, raw);
00080         CXX_BITFIELD_MEMBER( 1, 1, data_valid, raw);
00081     };
00082
00083     struct Hdr
00084     {
00085         Hdr_flags flags;
00086         l4_uint8_t gso_type;
00087         l4_uint16_t hdr_len;
00088         l4_uint16_t gso_size;
00089         l4_uint16_t csum_start;
00090         l4_uint16_t csum_offset;
00091         l4_uint16_t num_buffers;
00092     };
00093
00094     struct Features : L4virtio::Svr::Dev_config::Features
00095     {
00096         Features() = default;

```

```

00097     Features(l4_uint32_t raw) : L4virtio::Svr::Dev_config::Features(raw) {}
00098
00099     CXX_BITFIELD_MEMBER( 0, 0, csum, raw); // host handles partial csum
00100     CXX_BITFIELD_MEMBER( 1, 1, guest_csum, raw); // guest handles partial csum
00101     CXX_BITFIELD_MEMBER( 5, 5, mac, raw); // host has given mac
00102     CXX_BITFIELD_MEMBER( 6, 6, gso, raw); // host handles packets /w any GSO
00103     CXX_BITFIELD_MEMBER( 7, 7, guest_tso4, raw); // guest handles TSOv4 in
00104     CXX_BITFIELD_MEMBER( 8, 8, guest_tso6, raw); // guest handles TSOv6 in
00105     CXX_BITFIELD_MEMBER( 9, 9, guest_ecn, raw); // guest handles TSO[6] with ECN in
00106     CXX_BITFIELD_MEMBER(10, 10, guest_ufo, raw); // guest handles UFO in
00107     CXX_BITFIELD_MEMBER(11, 11, host_tso4, raw); // host handles TSOv4 in
00108     CXX_BITFIELD_MEMBER(12, 12, host_tso6, raw); // host handles TSOv6 in
00109     CXX_BITFIELD_MEMBER(13, 13, host_ecn, raw); // host handles TSO[6] with ECN in
00110     CXX_BITFIELD_MEMBER(14, 14, host_ufo, raw); // host handles UFO
00111     CXX_BITFIELD_MEMBER(15, 15, mrg_rxbuf, raw); // host can merge receive buffers
00112     CXX_BITFIELD_MEMBER(16, 16, status, raw); // virtio_net_config.status available
00113     CXX_BITFIELD_MEMBER(17, 17, ctrl_vq, raw); // Control channel available
00114     CXX_BITFIELD_MEMBER(18, 18, ctrl_rx, raw); // Control channel RX mode support
00115     CXX_BITFIELD_MEMBER(19, 19, ctrl_vlan, raw); // Control channel VLAN filtering
00116     CXX_BITFIELD_MEMBER(20, 20, ctrl_rx_extra, raw); // Extra RX mode control support
00117     CXX_BITFIELD_MEMBER(21, 21, guest_announce, raw); // Guest can announce device on the network
00118     CXX_BITFIELD_MEMBER(22, 22, mq, raw); // Device supports Receive Flow Steering
00119     CXX_BITFIELD_MEMBER(23, 23, ctrl_mac_addr, raw); // Set MAC address
00120 };
00121
00122 enum
00123 {
00124     Rx = 0,
00125     Tx = 1,
00126 };
00127
00128 struct Net_config_space
00129 {
00130     // The config defining mac address (if VIRTIO_NET_F_MAC aka Features::mac)
00131     l4_uint8_t mac[6];
00132     // currently not used ...
00133     l4_uint16_t status;
00134     l4_uint16_t max_virtqueue_pairs;
00135 };
00136
00137 L4virtio::Svr::Dev_config_t<Net_config_space> _dev_config;
00138
00139 explicit Virtio_net(unsigned vq_max)
00140 : L4virtio::Svr::Device(&_dev_config),
00141   _dev_config(L4VIRTIO_VENDOR_KK, L4VIRTIO_ID_NET, 2),
00142   _vq_max(vq_max)
00143 {
00144     Features hf(0);
00145     hf.ring_indirect_desc() = true;
00146     hf.mrg_rxbuf() = true;
00147 #if 0
00148     // disable currently unsupported options, but leave them in for
00149     // documentation purposes
00150     hf.csum() = true;
00151     hf.host_tso4() = true;
00152     hf.host_tso6() = true;
00153     hf.host_ufo() = true;
00154     hf.host_ecn() = true;
00155
00156     hf.guest_csum() = true;
00157     hf.guest_tso4() = true;
00158     hf.guest_tso6() = true;
00159     hf.guest_ufo() = true;
00160     hf.guest_ecn() = true;
00161 #endif
00162     _dev_config.host_features(0) = hf.raw;
00163     _dev_config.set_host_feature(L4VIRTIO_FEATURE_VERSION_1);
00164     _dev_config.reset_hdr();
00165
00166     reset_queue_config(Rx, vq_max);
00167     reset_queue_config(Tx, vq_max);
00168 }
00169
00170 void reset() override
00171 {
00172     for (L4virtio::Svr::Virtqueue &q: _q)
00173         q.disable();
00174
00175     reset_queue_config(Rx, _vq_max);
00176     reset_queue_config(Tx, _vq_max);
00177     _dev_config.reset_hdr();
00178 }
00179
00180 template<typename T, unsigned N >
00181 static unsigned array_length(T (&)[N]) { return N; }
00182
00183

```

```

00184 int reconfig_queue(unsigned index) override
00185 {
00186     Dbg(Dbg::Virtio, Dbg::Info, "Virtio")
00187     .printf("(%p): Reconfigure queue %d (%p): Status: %02x\n",
00188         this, index, _q + index, _dev_config.status().raw);
00189
00190     if (index >= array_length(_q))
00191         return -L4_ERANGE;
00192
00193     if (setup_queue(_q + index, index, _vq_max))
00194         return 0;
00195
00196     return -L4_EINVAL;
00197 }
00198
00199 void dump_features(Dbg const &dbg, const volatile l4_uint32_t *p)
00200 {
00201     dbg.cprintf("%08x:%08x:%08x:%08x:%08x:%08x:%08x\n",
00202         p[0], p[1], p[2], p[3], p[4], p[5], p[6], p[17]);
00203 }
00204
00205 void dump_features()
00206 {
00207     Dbg info(Dbg::Virtio, Dbg::Info, "Virtio");
00208     if (!info.is_active())
00209         return;
00210
00211     auto *hdr = _dev_config.hdr();
00212
00213     info.printf("Device %p running (%02x)\n\thost features: ",
00214         this, _dev_config.status().raw);
00215     dump_features(info, hdr->dev_features_map);
00216     info.printf("\tguest features: ");
00217     dump_features(info, hdr->driver_features_map);
00218 }
00219
00220 bool check_features() override
00221 {
00222     _negotiated_features = _dev_config.negotiated_features(0);
00223     return true;
00224 }
00225
00226 bool device_needs_reset() const
00227 { return _dev_config.status().device_needs_reset(); }
00228
00230 bool check_queues() override
00231 {
00232     for (L4virtio::Svr::Virtqueue &q: _q)
00233         if (!q.ready())
00234         {
00235             reset();
00236             Err().printf("failed to start queues\n");
00237             return false;
00238         }
00239     dump_features();
00240     return true;
00241 }
00242
00243 Server_iface *server_iface() const override
00244 { return L4::Epiface::server_iface(); }
00245
00250 void register_single_driver_irq() override
00251 {
00252     _kick_guest_irq = L4Re::Util::Unique_cap<L4::Irq>(
00253         L4Re::chkcap(server_iface()->template rcv_cap<L4::Irq>(0)));
00254     L4Re::chksys(server_iface()->realloc_rcv_cap(0));
00255 }
00256
00257 void trigger_driver_config_irq() override
00258 {
00259     _dev_config.add_irq_status(L4VIRTIO_IRQ_STATUS_CONFIG);
00260     _kick_guest_irq->trigger();
00261 }
00262
00269 void notify_queue(L4virtio::Svr::Virtqueue *queue)
00270 {
00271     // Downcast to Virtqueue to access kick_queue() - we know that our
00272     // queues have the type Virtqueue.
00273     Virtqueue *q = static_cast<Virtqueue*>(queue);
00274     if (q->kick_queue())
00275     {
00276         _dev_config.add_irq_status(L4VIRTIO_IRQ_STATUS_VRING);
00277         _kick_guest_irq->trigger();
00278     }
00279 }
00280
00281 void kick_emit_and_enable()

```

```

00282 {
00283     bool kick_pending = false;
00284
00285     for (auto &q : _q)
00286         kick_pending |= q.kick_enable_get_pending();
00287
00288     if (kick_pending)
00289     {
00290         _dev_config.add_irq_status(L4VIRTIO_IRQ_STATUS_VRING);
00291         _kick_guest_irq->trigger();
00292     }
00293 }
00294
00295 void kick_disable_and_remember()
00296 {
00297     for (auto &q : _q)
00298         q.kick_disable_and_remember();
00299 }
00300
00301 Features negotiated_features() const
00302 { return _negotiated_features; }
00303
00305 Virtqueue *tx_q() { return &_amp;_q[Tx]; }
00307 Virtqueue *rx_q() { return &_amp;_q[Rx]; }
00309 Virtqueue const *tx_q() const { return &_amp;_q[Tx]; }
00311 Virtqueue const *rx_q() const { return &_amp;_q[Rx]; }
00312
00313 private:
00314     Features _negotiated_features;
00316     unsigned _vq_max;
00318     Virtqueue _q[2];
00323     L4Re::Util::Unique_cap<L4::Irq> _kick_guest_irq;
00324 };

```

16.68 virtio_net.h

```

00001 /* SPDX-License-Identifier: MIT */
00002 /*
00003  * Copyright (C) 2022, 2024 Kernkonzept GmbH.
00004  * Author(s): Stephan Gerhold <stephan.gerhold@kernkonzept.com>
00005  */
00006
00007 #pragma once
00008
00014
00015 #include <l4/sys/types.h>
00016
00020 typedef struct l4virtio_net_header_t
00021 {
00022     l4_uint8_t flags;
00023     l4_uint8_t gso_type;
00024     l4_uint16_t hdr_len;
00025     l4_uint16_t gso_size;
00026     l4_uint16_t csum_start;
00027     l4_uint16_t csum_offset;
00028     l4_uint16_t num_buffers;
00029 } l4virtio_net_header_t;
00030
00034 typedef struct l4virtio_net_config_t
00035 {
00036     l4_uint8_t mac[6];
00037     l4_uint16_t status;
00038     l4_uint16_t max_virtqueue_pairs;
00039     l4_uint16_t mtu;
00040     l4_uint32_t speed;
00041     l4_uint8_t duplex;
00042 } l4virtio_net_config_t;
00043
00045 enum L4virtio_net_feature_bits
00046 {
00047     L4VIRTIO_NET_F_CSUM = 0,
00048     L4VIRTIO_NET_F_GUEST_CSUM = 1,
00049     L4VIRTIO_NET_F_MTU = 3,
00050     L4VIRTIO_NET_F_MAC = 5,
00051     L4VIRTIO_NET_F_GUEST_TSO4 = 7,
00052     L4VIRTIO_NET_F_GUEST_TSO6 = 8,
00053     L4VIRTIO_NET_F_GUEST_ECN = 9,
00054     L4VIRTIO_NET_F_GUEST_UFO = 10,
00055     L4VIRTIO_NET_F_HOST_TSO4 = 11,
00056     L4VIRTIO_NET_F_HOST_TSO6 = 12,
00057     L4VIRTIO_NET_F_HOST_ECN = 13,
00058     L4VIRTIO_NET_F_HOST_UFO = 14,
00059     L4VIRTIO_NET_F_MRG_RXBUF = 15,

```

```

00060     L4VIRTIO_NET_F_STATUS = 16,
00061     L4VIRTIO_NET_F_CTRL_VQ = 17,
00062     L4VIRTIO_NET_F_CTRL_RX = 18,
00063     L4VIRTIO_NET_F_CTRL_VLAN = 19,
00064     L4VIRTIO_NET_F_GUEST_ANNOUNCE = 21,
00065     L4VIRTIO_NET_F_MQ = 22,
00066     L4VIRTIO_NET_F_CTRL_MAC_ADDR = 23,
00067 };
00068

```

16.69 virtio_net_buffer.h

```

00001 /*
00002  * Copyright (C) 2016-2017, 2022, 2024-2025 Kernkonzept GmbH.
00003  * Author(s): Jean Wolter <jean.wolter@kernkonzept.com>
00004  *           Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include <l4/l4virtio/server/l4virtio>
00015
00019 struct Buffer : L4virtio::Svr::Data_buffer
00020 {
00021     Buffer() = default;
00022     Buffer(L4virtio::Svr::Driver_mem_region const *r,
00023           L4virtio::Svr::Virtqueue::Desc const &d,
00024           L4virtio::Svr::Request_processor const *)
00025     {
00026         pos = static_cast<char *>(r->local(d.addr));
00027         left = d.len;
00028     }
00029
00030     Buffer(char *data, l4_uint32_t size)
00031     {
00032         pos = data;
00033         left = size;
00034     }
00035
00036     template<typename T>
00037     explicit Buffer(T *p) : Data_buffer(p) {};
00038 };
00039

```

16.70 vlan.h

```

00001 /*
00002  * Copyright (C) 2020, 2022-2024 Kernkonzept GmbH.
00003  * Author(s): Jan Klötzke <jan.kloetzke@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include <l4/cxx/minmax>
00010 #include <l4/l4virtio/server/virtio>
00011 #include <l4/sys/types.h>
00012 #include <string.h>
00013
00014 #include "virtio_net.h"
00015 #include "virtio_net_buffer.h"
00016
00017 namespace {
00018
00019     const l4_uint16_t VLAN_ID_NATIVE = 0xffffU;
00020     const l4_uint16_t VLAN_ID_TRUNK = 0xfffeU;
00021
00022     inline bool vlan_valid_id(l4_uint16_t id)
00023     {
00024         return id > 0U && id < 0xffffU;
00025     }
00026
00027 }
00032
00036 class Virtio_vlan_mangle
00037 {
00038     l4_uint16_t _tci;

```

```

00039  l4_uint8_t _mac_remaining;
00040  l4_int8_t _tag_remaining;
00041
00042  constexpr Virtio_vlan_mangle(l4_uint16_t tci, l4_int8_t tag_remaining)
00043  : _tci{tci}, _mac_remaining{12}, _tag_remaining{tag_remaining}
00044  {}
00045
00046  public:
00052  Virtio_vlan_mangle()
00053  : _tci{0}, _mac_remaining{0}, _tag_remaining{0}
00054  {}
00055
00064  static constexpr Virtio_vlan_mangle add(l4_uint16_t tci)
00065  {
00066      return Virtio_vlan_mangle(tci, 4);
00067  }
00068
00075  static constexpr Virtio_vlan_mangle remove()
00076  {
00077      return Virtio_vlan_mangle(0xffffU, -4);
00078  }
00079
00093  l4_uint32_t copy_pkt(Buffer &dst, Buffer &src)
00094  {
00095      l4_uint32_t ret;
00096
00097      if (L4_LIKELY(_tci == 0))
00098      {
00099          // pass through (no tag or keep tag)
00100          ret = src.copy_to(&dst);
00101      }
00102      else if (_mac_remaining)
00103      {
00104          // copy initial MAC addresses
00105          ret = src.copy_to(&dst, _mac_remaining);
00106          _mac_remaining -= ret;
00107      }
00108      else if (_tag_remaining > 0)
00109      {
00110          // add VLAN tag
00111          l4_uint8_t tag[4] = {
00112              0x81, 0x00,
00113              static_cast<l4_uint8_t>(_tci >> 8),
00114              static_cast<l4_uint8_t>(_tci & 0xffU)
00115          };
00116
00117          ret = cxx::min(static_cast<l4_uint32_t>(_tag_remaining), dst.left());
00118          memcpy(dst.pos, &tag[4 - _tag_remaining], ret);
00119          dst.skip(ret);
00120          _tag_remaining -= (int)ret;
00121      }
00122      else if (_tag_remaining < 0)
00123      {
00124          // remove VLAN tag
00125          _tag_remaining += static_cast<int>(src.skip(-_tag_remaining));
00126          ret = 0;
00127      }
00128      else
00129          ret = src.copy_to(&dst);
00130
00131      return ret;
00132  }
00133
00142  void rewrite_hdr(Virtio_net::Hdr *hdr)
00143  {
00144      if (L4_UNLIKELY(_tci != 0 && hdr->flags.need_csum()))
00145      {
00146          if (_tci == 0xffffU)
00147              hdr->csum_start -= 4U;
00148          else
00149              hdr->csum_start += 4U;
00150      }
00151  }
00152 };
00153

```

16.71 amd64/l4/sys/segment.h File Reference

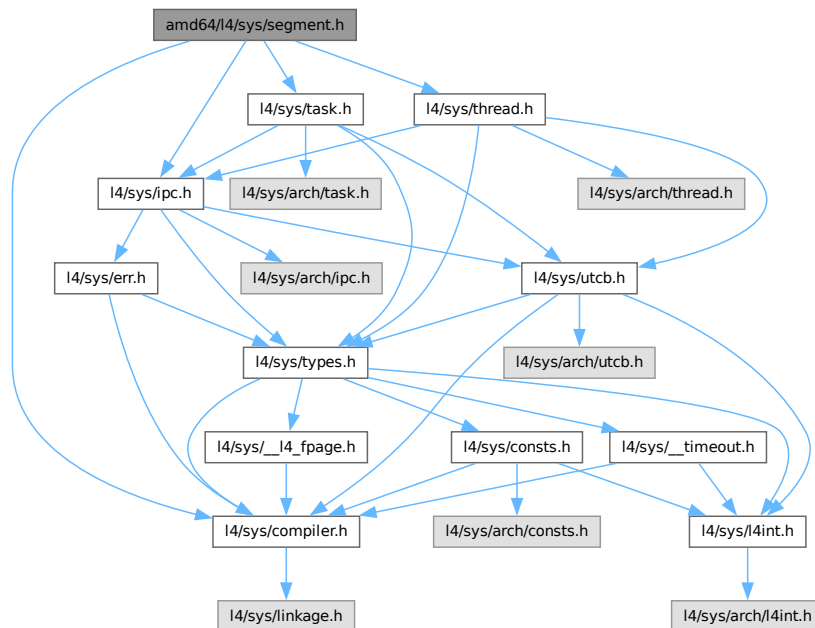
Segment handling (AMD64).

```

#include <l4/sys/ipc.h>
#include <l4/sys/task.h>

```

```
#include <l4/sys/thread.h>
#include <l4/sys/compiler.h>
Include dependency graph for segment.h:
```



Enumerations

- enum [L4_task_ldt_x86_consts](#) { [L4_TASK_LDT_X86_ENTRY_SIZE](#) = 8 , [L4_TASK_LDT_X86_MAX_ENTRIES](#) }
- Constants for LDT handling.*
- enum [L4_sys_segment](#) { [L4_AMD64_SEGMENT_FS](#) = 0 , [L4_AMD64_SEGMENT_GS](#) = 1 }
- Constants for identifying segments.*

Functions

- long [fiasco_ldt_set](#) ([l4_cap_idx_t](#) task, void *ldt, unsigned int num_desc, unsigned int entry_number_start, [l4_utcb_t](#) *utcb)
- Set LDT segments descriptors.*
- long [fiasco_gdt_set](#) ([l4_cap_idx_t](#) thread, void *desc, unsigned int size, unsigned int entry_number_start, [l4_utcb_t](#) *utcb)
- Set GDT segment descriptors.*
- unsigned [fiasco_gdt_get_entry_offset](#) ([l4_cap_idx_t](#) thread, [l4_utcb_t](#) *utcb)
- Return the offset of the entry in the GDT.*
- long [fiasco_amd64_set_fs](#) ([l4_cap_idx_t](#) thread, [l4_umword_t](#) base, [l4_utcb_t](#) *utcb)
- Set the base address for the FS segment.*
- long [fiasco_amd64_set_segment_base](#) ([l4_cap_idx_t](#) thread, enum [L4_sys_segment](#) segr, [l4_umword_t](#) base, [l4_utcb_t](#) *utcb)
- Set the base address for a segment.*
- long [fiasco_amd64_segment_info](#) ([l4_cap_idx_t](#) thread, unsigned *user_ds, unsigned *user_cs, unsigned *user32_cs, [l4_utcb_t](#) *utcb)
- Get segment information.*

16.71.1 Detailed Description

Segment handling (AMD64).

Definition in file [segment.h](#).

16.71.2 Enumeration Type Documentation

16.71.2.1 L4_sys_segment

```
enum L4_sys_segment
```

Constants for identifying segments.

Enumerator

| | |
|---------------------|--------------------------------------|
| L4_AMD64_SEGMENT_FS | Constant identifying the FS segment. |
| L4_AMD64_SEGMENT_GS | Constant identifying the GS segment. |

Definition at line 106 of file [segment.h](#).

16.71.2.2 L4_task_ldt_x86_consts

```
enum L4_task_ldt_x86_consts
```

Constants for LDT handling.

Enumerator

| | |
|-----------------------------|--|
| L4_TASK_LDT_X86_ENTRY_SIZE | Size of an LDT entry. |
| L4_TASK_LDT_X86_MAX_ENTRIES | Maximum number of LDT entries that can be written with one call. |

Definition at line 75 of file [segment.h](#).

16.71.3 Function Documentation

16.71.3.1 fiasco_amd64_segment_info()

```
long fiasco_amd64_segment_info (  
    l4_cap_idx_t thread,  
    unsigned * user_ds,  
    unsigned * user_cs,  
    unsigned * user32_cs,  
    l4_utcb_t * utcb) [inline]
```

Get segment information.

Parameters

| | | |
|-----|------------------|--|
| in | <i>thread</i> | Thread to get info from. |
| out | <i>user_ds</i> | DS segment selector. |
| out | <i>user_cs</i> | 64-bit CS segment selector. |
| out | <i>user32_cs</i> | 32-bit CS segment selector. |
| | <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |

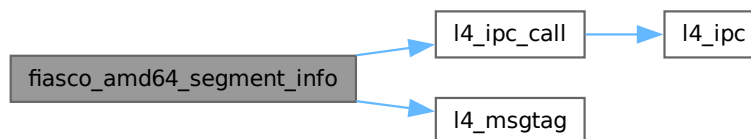
Returns

System call error

Definition at line 175 of file [segment.h](#).

References [l4_ipc_call\(\)](#), [L4_IPC_NEVER](#), [l4_msgtag\(\)](#), [L4_PROTO_THREAD](#), [L4_THREAD_AMD64_GET_SEGMENT_INFO_OP](#), and [l4_msg_regs_t::mr](#).

Here is the call graph for this function:

**16.71.3.2 fiasco_amd64_set_fs()**

```

long fiasco_amd64_set_fs (
    l4_cap_idx_t thread,
    l4_umword_t base,
    l4_utcb_t * utcb) [inline]

```

Set the base address for the FS segment.

Parameters

| | |
|---------------|--|
| <i>thread</i> | Thread for which the FS base address shall be modified. |
| <i>base</i> | Base address. |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |

Return values

| | |
|-------------------|--|
| <i>L4_EOK</i> | Success. |
| <i>-L4_EINVAL</i> | Invalid base address (<i>base</i>). |
| <i>-L4_ENOSYS</i> | Operation not supported with current kernel configuration. |

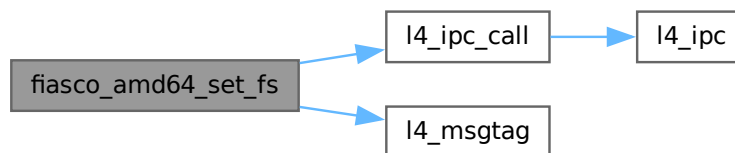
Note

Calling this function is equivalent to calling `fiasco_amd64_set_segment_base(thread, L4_↔AMD64_SEGMENT_FS, base, utcb)`.

Definition at line 203 of file [segment.h](#).

References [L4_AMD64_SEGMENT_FS](#), [l4_ipc_call\(\)](#), [L4_IPC_NEVER](#), [l4_msgtag\(\)](#), [L4_PROTO_THREAD](#), and [L4_THREAD_AMD64_SET_SEGMENT_BASE_OP](#).

Here is the call graph for this function:

**16.71.3.3 fiasco_amd64_set_segment_base()**

```

long fiasco_amd64_set_segment_base (
    l4_cap_idx_t thread,
    enum L4_sys_segment segr,
    l4_umword_t base,
    l4_utcb_t * utcb) [inline]

```

Set the base address for a segment.

Parameters

| | |
|---------------|--|
| <i>thread</i> | Thread for which the base address of the selected segment shall be modified. |
| <i>segr</i> | Segment to modify (one of L4_sys_segment). |
| <i>base</i> | Base address. |
| <i>utcb</i> | UTCB to be used for this operation, shall be the UTCB of the calling thread. See l4_utcb . |

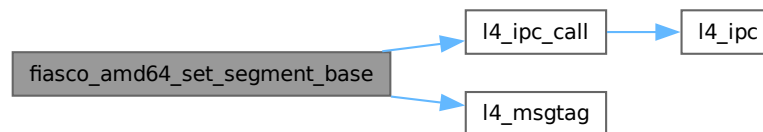
Return values

| | |
|-------------------------|--|
| <code>L4_EOK</code> | Success. |
| <code>-L4_EINVAL</code> | Invalid segment (<code>segr</code>) or base address (<code>base</code>). |
| <code>-L4_ENOSYS</code> | Operation not supported with current kernel configuration. |

Definition at line 211 of file [segment.h](#).

References [l4_ipc_call\(\)](#), [L4_IPC_NEVER](#), [l4_msgtag\(\)](#), [L4_PROTO_THREAD](#), and [L4_THREAD_AMD64_SET_SEGMENT_BASE](#).

Here is the call graph for this function:



16.72 segment.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 /*****
00013 #ifndef __L4_SYS__ARCH_X86__SEGMENT_H__
00014 #define __L4_SYS__ARCH_X86__SEGMENT_H__
00015
00016 #ifndef L4API_l4f
00017 #error This header file can only be used with a L4API version!
00018 #endif
00019
00020 #include <l4/sys/ipc.h>
00021
00039 L4_INLINE long
00040 fiasco_ldt_set(l4_cap_idx_t task, void *ldt, unsigned int num_desc,
00041               unsigned int entry_number_start, l4_utcb_t *utcb);
00042
00059 L4_INLINE long
00060 fiasco_gdt_set(l4_cap_idx_t thread, void *desc, unsigned int size,
00061               unsigned int entry_number_start, l4_utcb_t *utcb);
00062
00069 L4_INLINE unsigned
00070 fiasco_gdt_get_entry_offset(l4_cap_idx_t thread, l4_utcb_t *utcb);
00071
00075 enum L4_task_ldt_x86_consts
00076 {
00078     L4_TASK_LDT_X86_ENTRY_SIZE = 8,
00080     L4_TASK_LDT_X86_MAX_ENTRIES
00081         = (L4_UTCB_GENERIC_DATA_SIZE - 2)
00082           / (L4_TASK_LDT_X86_ENTRY_SIZE / (L4_MWORD_BITS / 8)),
00083 };
00084
00100 L4_INLINE long
00101 fiasco_amd64_set_fs(l4_cap_idx_t thread, l4_umword_t base, l4_utcb_t *utcb);
00102
00106 enum L4_sys_segment
00107 {
00109     L4_AMD64_SEGMENT_FS = 0,
00111     L4_AMD64_SEGMENT_GS = 1

```

```

00112 };
00113
00127 L4_INLINE long
00128 fiasco_amd64_set_segment_base(l4_cap_idx_t thread, enum L4_sys_segment segr,
00129                               l4_umword_t base, l4_utcb_t *utcb);
00130
00140 L4_INLINE long
00141 fiasco_amd64_segment_info(l4_cap_idx_t thread, unsigned *user_ds,
00142                           unsigned *user_cs, unsigned *user32_cs,
00143                           l4_utcb_t *utcb);
00144
00145 /*****
00146 *** Implementation
00147 *****/
00148
00149 #include <l4/sys/task.h>
00150 #include <l4/sys/thread.h>
00151
00152 L4_INLINE long
00153 fiasco_ldt_set(l4_cap_idx_t task, void *ldt, unsigned int num_desc,
00154               unsigned int entry_number_start, l4_utcb_t *utcb)
00155 {
00156     if (num_desc > L4_TASK_LDT_X86_MAX_ENTRIES)
00157         return -L4_EINVAL;
00158     l4_utcb_mr_u(utcb)->mr[0] = L4_TASK_LDT_SET_X86_OP;
00159     l4_utcb_mr_u(utcb)->mr[1] = entry_number_start;
00160     __builtin_memcpy(&l4_utcb_mr_u(utcb)->mr[2], ldt,
00161                     num_desc * L4_TASK_LDT_X86_ENTRY_SIZE);
00162     return l4_error_u(l4_ipc_call(task, utcb, l4_msgtag(L4_PROTO_TASK, 2 + num_desc * 2, 0, 0),
00163                          L4_IPC_NEVER), utcb);
00164 }
00165
00165 L4_INLINE unsigned
00166 fiasco_gdt_get_entry_offset(l4_cap_idx_t thread, l4_utcb_t *utcb)
00167 {
00168     l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_X86_GDT_OP;
00169     if (l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 1, 0, 0), L4_IPC_NEVER), utcb))
00170         return -1;
00171     return l4_utcb_mr_u(utcb)->mr[0];
00172 }
00173
00174 L4_INLINE long
00175 fiasco_amd64_segment_info(l4_cap_idx_t thread, unsigned *user_ds,
00176                           unsigned *user_cs, unsigned *user32_cs,
00177                           l4_utcb_t *utcb)
00178 {
00179     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00180     int r;
00181
00182     m->mr[0] = L4_THREAD_AMD64_GET_SEGMENT_INFO_OP;
00183
00184     r = l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 1, 0, 0),
00185                          L4_IPC_NEVER), utcb);
00186     if (r < 0)
00187         return r;
00188
00189     *user_ds = m->mr[0];
00190     *user_cs = m->mr[1];
00191     *user32_cs = m->mr[2];
00192
00193     return 0;
00194 }
00195
00196 #include <l4/sys/compiler.h>
00197
00198 /*****
00199 *** Implementation
00200 *****/
00201
00202 L4_INLINE long
00203 fiasco_amd64_set_fs(l4_cap_idx_t thread, l4_umword_t base, l4_utcb_t *utcb)
00204 {
00205     l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_AMD64_SET_SEGMENT_BASE_OP | ((l4_umword_t)L4_AMD64_SEGMENT_FS
00206     « 16);
00207     l4_utcb_mr_u(utcb)->mr[1] = base;
00208     return l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 2, 0, 0), L4_IPC_NEVER),
00209                      utcb);
00210 }
00210
00210 L4_INLINE long
00211 fiasco_amd64_set_segment_base(l4_cap_idx_t thread, enum L4_sys_segment segr,
00212                               l4_umword_t base, l4_utcb_t *utcb)
00213 {
00214     l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_AMD64_SET_SEGMENT_BASE_OP | ((l4_umword_t)segr « 16);
00215     l4_utcb_mr_u(utcb)->mr[1] = base;
00216     return l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 2, 0, 0), L4_IPC_NEVER),
00217                      utcb);

```

```

00217 }
00218
00219 L4_INLINE long
00220 fiasco_gdt_set(l4_cap_idx_t thread, void *desc, unsigned int size,
00221               unsigned int entry_number_start, l4_utcb_t *utcb)
00222 {
00223     l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_X86_GDT_OP;
00224     l4_utcb_mr_u(utcb)->mr[1] = entry_number_start;
00225     __builtin_memcpy(&l4_utcb_mr_u(utcb)->mr[2], desc, size);
00226     return l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 2 + (size / 8), 0, 0),
00227                       L4_IPC_NEVER), utcb);
00227 }
00228
00229
00230 #endif /* ! __L4_SYS__ARCH_X86__SEGMENT_H__ */

```

16.73 x86/l4/sys/segment.h File Reference

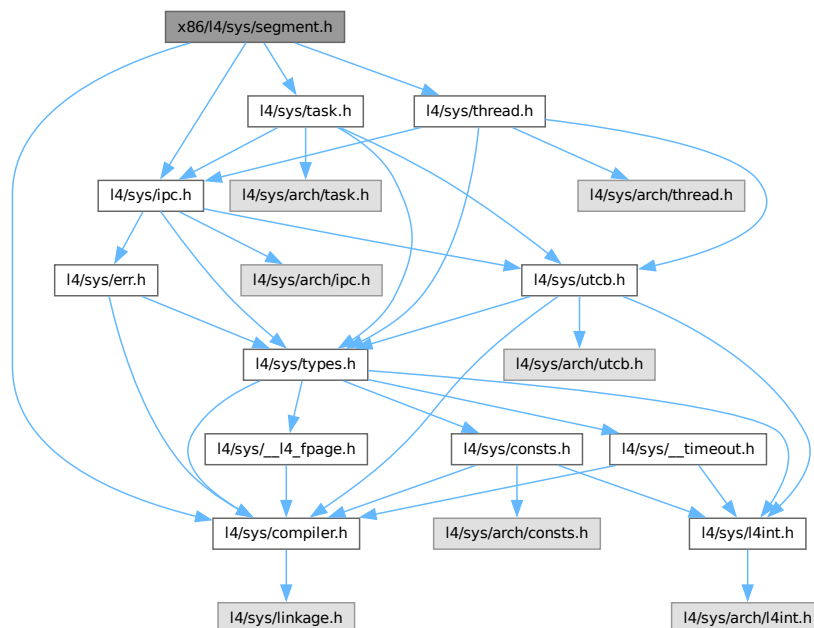
Segment handling (x86).

```

#include <l4/sys/ipc.h>
#include <l4/sys/compiler.h>
#include <l4/sys/task.h>
#include <l4/sys/thread.h>

```

Include dependency graph for segment.h:



Enumerations

- enum `L4_task_ldt_x86_consts` { `L4_TASK_LDT_X86_ENTRY_SIZE` = 8 , `L4_TASK_LDT_X86_MAX_ENTRIES` }

Constants for LDT handling.

Functions

- long `fiasco_ldt_set` (`l4_cap_idx_t` task, void *ldt, unsigned int num_desc, unsigned int entry_number_start, `l4_utcb_t` *utcb)
Set LDT segments descriptors.
- long `fiasco_gdt_set` (`l4_cap_idx_t` thread, void *desc, unsigned int size, unsigned int entry_number_start, `l4_utcb_t` *utcb)
Set GDT segment descriptors.
- unsigned `fiasco_gdt_get_entry_offset` (`l4_cap_idx_t` thread, `l4_utcb_t` *utcb)
Return the offset of the entry in the GDT.

16.73.1 Detailed Description

Segment handling (x86).

Definition in file [segment.h](#).

16.73.2 Enumeration Type Documentation

16.73.2.1 L4_task_ldt_x86_consts

enum `L4_task_ldt_x86_consts`

Constants for LDT handling.

Enumerator

| | |
|--|--|
| <code>L4_TASK_LDT_X86_ENTRY_SIZE</code> | Size of an LDT entry. |
| <code>L4_TASK_LDT_X86_MAX_ENTRIES</code> | Maximum number of LDT entries that can be written with one call. |

Definition at line 75 of file [segment.h](#).

16.74 segment.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 /*****
00013  #ifndef __L4_SYS__ARCH_X86__SEGMENT_H__
00014  #define __L4_SYS__ARCH_X86__SEGMENT_H__
00015
00016  #ifndef L4API_L4f
00017  #error This header file can only be used with a L4API version!
00018  #endif
00019
00020  #include <l4/sys/ipc.h>
00021
00039  L4_INLINE long
00040  fiasco_ldt_set(l4_cap_idx_t task, void *ldt, unsigned int num_desc,
00041                unsigned int entry_number_start, l4_utcb_t *utcb);

```

```

00042
00059 L4_INLINE long
00060 fiasco_gdt_set(l4_cap_idx_t thread, void *desc, unsigned int size,
00061               unsigned int entry_number_start, l4_utcb_t *utcb);
00062
00069 L4_INLINE unsigned
00070 fiasco_gdt_get_entry_offset(l4_cap_idx_t thread, l4_utcb_t *utcb);
00071
00075 enum L4_task_ldt_x86_consts
00076 {
00078     L4_TASK_LDT_X86_ENTRY_SIZE = 8,
00080     L4_TASK_LDT_X86_MAX_ENTRIES
00081         = (L4_UTCB_GENERIC_DATA_SIZE - 2)
00082           / (L4_TASK_LDT_X86_ENTRY_SIZE / (L4_MWORD_BITS / 8)),
00083 };
00084
00085 /*****
00086  *** Implementation
00087  *****/
00088
00089 #include <l4/sys/compiler.h>
00090 #include <l4/sys/task.h>
00091 #include <l4/sys/thread.h>
00092
00093 L4_INLINE long
00094 fiasco_ldt_set(l4_cap_idx_t task, void *ldt, unsigned int num_desc,
00095               unsigned int entry_number_start, l4_utcb_t *utcb)
00096 {
00097     if (num_desc > L4_TASK_LDT_X86_MAX_ENTRIES)
00098         return -L4_EINVAL;
00099     l4_utcb_mr_u(utcb)->mr[0] = L4_TASK_LDT_SET_X86_OP;
00100     l4_utcb_mr_u(utcb)->mr[1] = entry_number_start;
00101     __builtin_memcpy(&l4_utcb_mr_u(utcb)->mr[2], ldt,
00102                     num_desc * L4_TASK_LDT_X86_ENTRY_SIZE);
00103     return l4_error_u(l4_ipc_call(task, utcb, l4_msgtag(L4_PROTO_TASK, 2 + num_desc * 2, 0, 0),
00104                        L4_IPC_NEVER), utcb);
00104 }
00105
00106 L4_INLINE unsigned
00107 fiasco_gdt_get_entry_offset(l4_cap_idx_t thread, l4_utcb_t *utcb)
00108 {
00109     l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_X86_GDT_OP;
00110     if (l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 1, 0, 0), L4_IPC_NEVER), utcb))
00111         return -1;
00112     return l4_utcb_mr_u(utcb)->mr[0];
00113 }
00114
00115 L4_INLINE long
00116 fiasco_gdt_set(l4_cap_idx_t thread, void *desc, unsigned int size,
00117               unsigned int entry_number_start, l4_utcb_t *utcb)
00118 {
00119     l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_X86_GDT_OP;
00120     l4_utcb_mr_u(utcb)->mr[1] = entry_number_start;
00121     __builtin_memcpy(&l4_utcb_mr_u(utcb)->mr[2], desc, size);
00122     return l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 2 + (size >> 2), 0, 0),
00123                        L4_IPC_NEVER), utcb);
00123 }
00124
00125 #endif /* ! __L4_SYS_ARCH_X86_SEGMENT_H__ */

```

16.75 amd64/l4/util/perform.h File Reference

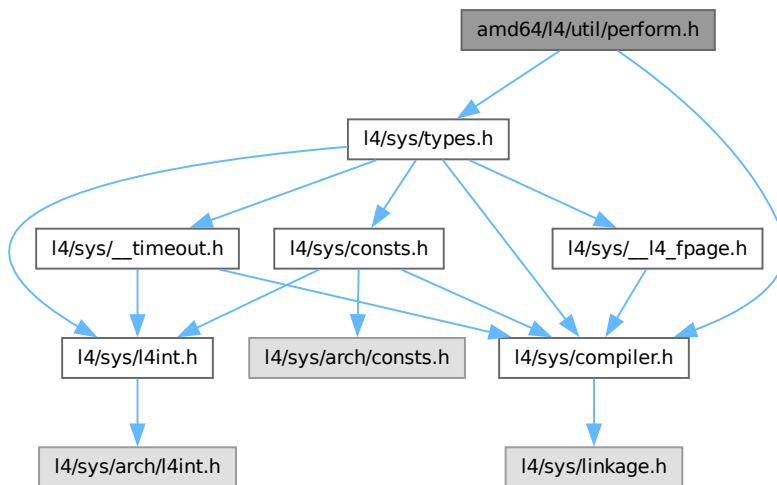
Performance Monitoring using P5/P6 Measurement Counters.

```

#include <l4/sys/types.h>
#include <l4/sys/compiler.h>

```


Include dependency graph for perform.h:



16.75.1 Detailed Description

Performance Monitoring using P5/P6 Measurement Counters.

Define either CPU_PENTIUM or CPU_P6

Definition in file [perform.h](#).

16.76 perform.h

[Go to the documentation of this file.](#)

```

00001
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #ifndef __L4UTIL_PERFORM_H
00009 #define __L4UTIL_PERFORM_H
00010
00011 #include <l4/sys/types.h>
00012 #include <l4/sys/compiler.h>
00013
00014 L4_BEGIN_DECLS
00015
00016 extern const char*strp6pmc_event(l4_uint32_t event);
00017
00018 #ifndef CONFIG_PERFORM_ONLY_PROTOTYPES
00019 #if ! (defined CPU_PENTIUM ^ defined CPU_P6 ^ defined CPU_K7)
00020 #error You must define your target architecture.
00021 #error Define EITHER CPU_PENTIUM for Intel Pentium or CPU_P6 for Intel PPro/PII/PIII.
00022 #else
00023 /* P5/P6/K7 section */
00024 #endif
00025 #endif

```

```

00034 /* Makros for access to model specific registers (MSR) */
00035
00036 /* Write the 64-Bit Model Specific Register. First argument is the register,
00037    second the 64-Bit value. This can only be called at privilege level 0.
00038    With L4, the kernel emulates the WRMSR when calling in PL 3.
00039    */
00040 static inline void l4_i586_wrmsr(unsigned reg,unsigned long long*val){
00041     unsigned long dummyeax, dummyecx, dummyedx;
00042
00043     asm volatile(
00044         ".byte 0xf; .byte 0x30\n" /* wrmsr */
00045         : "=a" (dummyeax), "=d" (dummyedx), "=c" (dummyecx)
00046         : "2" (reg), "0" (*(unsigned *)val), "1" (*(unsigned *)val+1))
00047     );
00048 }
00049
00050 /* Read the 64-Bit Model Specific Register. First argument is the register,
00051    second the address to a 64-Bit value. This can only be called at
00052    privilege level 0. With L4, the kernel emulates the RDMSR when calling
00053    in PL 3.
00054    */
00055 static inline void l4_i586_rdmsr(unsigned reg,unsigned long long*val){
00056     unsigned dummy;
00057
00058     asm volatile(
00059         ".byte 0xf; .byte 0x32\n" /* rdmsr */
00060         : "=a" (*(unsigned *)val), "=d" (*(unsigned *)val+1), "=c" (dummy)
00061         : "2" (reg)
00062     );
00063 }
00064
00065 #ifdef CPU_PENTIUM
00066 /* Pentium section */
00067
00068 /* functions and events defined here are only usable at Pentium
00069    Processors. P6 architecture does NOT support this kind of measuring and
00070    these events. P6 architecture has its own counters and its own events.
00071    See P6-section for details. */
00072
00073 /* from l4linux/arch/l4-i386/include/perform.h */
00074
00075 static inline void
00076 l4_i586_reset_event_counter(void){
00077     asm volatile("xor %%rax, %%rax\n"
00078         "xor %%rdx, %%rdx\n"
00079         "mov $0x12, %%rcx\n"
00080         ".byte 0x0f, 0x30\n"
00081         "movl $0x13, %%rcx\n"
00082         ".byte 0x0f, 0x30\n"
00083         : : : "cx", "ax", "dx"
00084     );
00085 };
00086
00087 static inline void
00088 l4_i586_read_event_counter_long(long long *counter0, long long *counter1)
00089 {
00090     asm volatile(
00091         /* "movl $0, %%eax\n"
00092         "movl $0x11, %%ecx\n"
00093         ".byte 0x0f, 0x30\n" */ /* stop event counting */
00094         "mov $0x12, %%rcx\n"
00095         ".byte 0x0f, 0x32\n"
00096         "mov %%rax, (%%%rbx)\n"
00097         "mov %%rdx, 4(%%rbx)\n"
00098         "mov $0x13, %%ecx\n"
00099         ".byte 0x0f, 0x32\n"
00100         "mov %%rax, (%%%rsi)\n"
00101         "mov %%rdx, 4(%%rsi)\n"
00102         : /* no output */
00103         : "b" (counter0), "S" (counter1)
00104         : "ax", "cx", "dx"
00105     );
00106 }
00107
00108 static inline void
00109 l4_i586_read_event_counter(int *counter0, int *counter1)
00110 {
00111     asm volatile("push %%rdx \n"
00112         ".byte 0x0f, 0x30 \n"
00113         "mov $0x12, %%rcx \n"
00114         ".byte 0x0f, 0x32 \n"
00115         "mov %%rax, %%rbx \n"
00116         "movl $0x13, %%rcx \n"
00117         ".byte 0x0f, 0x32\n"
00118         "popl %%edx\n"
00119         : "=b" (*counter0), "=a" (*counter1)

```

```

00121         : "l" (0), "c" (0x11)
00122     );
00123 }
00124
00125 static inline void
00126 l4_i586_select_event(int event0, int event1)
00127 {
00128     asm volatile(".byte 0x0f, 0x30\n"
00129         :
00130         :
00131         "a" (event0 + (event1 < 16)),
00132         "d" (0),
00133         "c" (0x11)
00134     );
00135 };
00136
00137 #define P5_RD_MISS          0x003 /* 000011B */
00138 #define P5_WR_MISS          0x008 /* 000100B */
00139 #define P5_RW_MISS          0x029 /* 101001B */
00140 #define P5_EX_MISS          0x00e /* 001110B */
00141
00142 #define P5_D_WBACK          0x006 /* 000110B */
00143
00144 #define P5_RW_TLB           0x002 /* 00010B */
00145 #define P5_EX_TLB           0x00d /* 01101B */
00146
00147 #define P5_A_STALL          0x01f /* 11111B */
00148 #define P5_W_STALL          0x019 /* 11001B */
00149 #define P5_R_STALL          0x01a /* 11010B */
00150 #define P5_X_STALL          0x01b /* 11011B */
00151
00152 #define P5_AGI_STALL         0x01f /* 11111B */
00153
00154 #define P5_PIPELINE_FLUSH   0x015 /* 10101B */
00155
00156 #define P5_NON_CACHE_RD     0x01e /* 11110B */
00157 #define P5_NCACHE_REFS      0x01e /* 11110B */
00158 #define P5_LOCKED_BUS       0x01c /* 11100B */
00159
00160 #define P5_MEM2PIPE         0x009 /* 01001B */
00161 #define P5_BANK_CONF        0x00a /* 01010B */
00162
00163
00164 #define P5_INSTRS_EX         0x016 /* 10110B */
00165 #define P5_INSTRS_EX_V      0x017 /* 10111B */
00166
00167
00168 #define P5_CNT_NOTHING       (0x00 < 6) /* 00B < 6 */
00169 #define P5_CNT_EVENT_PL0     (0x01 < 6) /* 01B < 6 */
00170 #define P5_CNT_EVENT_PL3     (0x02 < 6) /* 10B < 6 */
00171 #define P5_CNT_EVENT         (0x03 < 6) /* 11B < 6 */
00172 #define P5_CNT_CLOCKS_PL0    (0x05 < 6) /* 101B < 6 */
00173 #define P5_CNT_CLOCKS_PL3    (0x06 < 6) /* 110B < 6 */
00174 #define P5_CNT_CLOCKS        (0x07 < 6) /* 111B < 6 */
00175
00176
00177 #else
00178 #if defined CPU_P6
00179 /* PPro/PII/PIII section */
00180
00181 /*-
00182  * Copyright (c) 1997 The President and Fellows of Harvard College.
00183  * All rights reserved.
00184  * Copyright (c) 1997 Aaron B. Brown.
00185  *
00186  * Redistribution and use in source and binary forms, with or without
00187  * modification, are permitted provided that the following conditions
00188  * are met:
00189  * 1. Redistributions of source code must retain the above copyright
00190  *   notice, this list of conditions and the following disclaimer.
00191  * 2. Redistributions in binary form must reproduce the above copyright
00192  *   notice, this list of conditions and the following disclaimer in the
00193  *   documentation and/or other materials provided with the distribution.
00194  * 3. All advertising materials mentioning features or use of this software
00195  *   must display the following acknowledgement:
00196  *       This product includes software developed by Harvard University
00197  *       and its contributors.
00198  * 4. Neither the name of the University nor the names of its contributors
00199  *   may be used to endorse or promote products derived from this software
00200  *   without specific prior written permission.
00201  *
00202  * THIS SOFTWARE IS PROVIDED BY HARVARD AND CONTRIBUTORS ``AS IS" AND
00203  * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
00204  * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
00205  * ARE DISCLAIMED. IN NO EVENT SHALL HARVARD UNIVERSITY OR CONTRIBUTORS BE
00206  * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
00207  * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF

```

```

00208 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
00209 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
00210 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
00211 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
00212 * POSSIBILITY OF SUCH DAMAGE.
00213 */
00214
00215 /*****
00216 ** Symbolic names for counter numbers (used in select_p6counter()) **
00217 *****/
00218 *
00219 * These correspond in order to the Pentium Pro counters. Add new counters at
00220 * the end. These agree with the mnemonics in the Pentium Pro Family
00221 * Developer's Manual, vol 3.
00222 *
00223 * Those events marked with a $ require a MESI unit field; those marked with
00224 * a @ require a self/any unit field. Those marked with a 0 are only supported
00225 * in counter 0; those marked with 1 are only supported in counter 1.
00226 */
00227
00228 /* Data cache unit */
00229 #define P6_DATA_MEM_REFS 0x43 /* total memory refs */
00230 #define P6_DCU_LINES_IN 0x45 /* all lines allocated in cache unit */
00231 #define P6_DCU_M_LINES_IN 0x46 /* M lines allocated in cache unit */
00232 #define P6_DCU_M_LINES_OUT 0x47 /* M lines evicted from cache */
00233 #define P6_DCU_MISS_OUTSTANDING 0x48 /* #cycles a miss is outstanding */
00234
00235 /* Instruction fetch unit */
00236 #define P6_IFU_IFETCH 0x80 /* instruction fetches */
00237 #define P6_IFU_IFETCH_MISS 0x81 /* instruction fetch misses */
00238 #define P6_ITLB_MISS 0x85 /* ITLB misses */
00239 #define P6_IFU_MEM_STALL 0x86 /* number of cycles IFU is stalled */
00240 #define P6_ILD_STALL 0x87 /* #stalls in instr length decode */
00241
00242 /* L2 Cache */
00243 #define P6_L2_IFETCH 0x28 /* ($) 12 ifetches */
00244 #define P6_L2_LD 0x29 /* ($) 12 data loads */
00245 #define P6_L2_ST 0x2a /* ($) 12 data stores */
00246 #define P6_L2_LINES_IN 0x24 /* lines allocated in L2 */
00247 #define P6_L2_LINES_OUT 0x26 /* lines removed from L2 */
00248 #define P6_L2_M_LINES_INM 0x25 /* modified lines allocated in L2 */
00249 #define P6_L2_M_LINES_OUTM 0x27 /* modified lines removed from L2 */
00250 #define P6_L2_RQSTS 0x2e /* ($) number of l2 requests */
00251 #define P6_L2_ADS 0x21 /* number of l2 addr strobes */
00252 #define P6_L2_DBUS_BUSY 0x22 /* number of data bus busy cycles */
00253 #define P6_L2_DBUS_BUSY_RD 0x23 /* #bus cycles xferring l2->cpu */
00254
00255 /* External bus logic */
00256 #define P6_BUS_DRDY_CLOCKS 0x62 /* (@) #clocks DRDY is asserted */
00257 #define P6_BUS_LOCK_CLOCKS 0x63 /* (@) #clocks LOCK is asserted */
00258 #define P6_BUS_REQ_OUTSTANDING 0x60 /* #bus requests outstanding */
00259 #define P6_BUS_TRAN_BRD 0x65 /* (@) bus burst read txns */
00260 #define P6_BUS_TRAN_RFO 0x66 /* (@) bus read for ownership txns */
00261 #define P6_BUS_TRAN_WB 0x67 /* (@) bus writeback txns */
00262 #define P6_BUS_TRAN_IFETCH 0x68 /* (@) bus instr fetch txns */
00263 #define P6_BUS_TRAN_INVALID 0x69 /* (@) bus invalidate txns */
00264 #define P6_BUS_TRAN_PWR 0x6a /* (@) bus partial write txns */
00265 #define P6_BUS_TRANS_P 0x6b /* (@) bus partial txns */
00266 #define P6_BUS_TRANS_IO 0x6c /* (@) bus I/O txns */
00267 #define P6_BUS_TRAN_DEF 0x6d /* (@) bus deferred txns */
00268 #define P6_BUS_TRAN_BURST 0x6e /* (@) bus burst txns */
00269 #define P6_BUS_TRAN_ANY 0x70 /* (@) total bus txns */
00270 #define P6_BUS_TRAN_MEM 0x6f /* (@) total memory txns */
00271 #define P6_BUS_DATA_RCV 0x64 /* #busclocks CPU is receiving data */
00272 #define P6_BUS_BNR_DRV 0x61 /* #busclocks CPU is driving BNR pin */
00273 #define P6_BUS_HIT_DRV 0x7a /* #busclocks CPU is driving HIT pin */
00274 #define P6_BUS_HITM_DRV 0x7b /* #busclocks CPU is driving HITM pin */
00275 #define P6_BUS_SNOOP_STALL 0x7e /* #clkcycles bus is snoop-stalled */
00276
00277 /* FPU */
00278 #define P6_FLOPS 0xc1 /* (0) number of FP ops retired */
00279 #define P6_FP_COMP_OPS 0x10 /* (0) computational FPOPS exec'd */
00280 #define P6_FP_ASSIST 0x11 /* (1) FP excep's handled in ucode */
00281 #define P6_MUL 0x12 /* (1) number of FP multiplies */
00282 #define P6_DIV 0x13 /* (1) number of FP divides */
00283 #define P6_CYCLES_DIV_BUSY 0x14 /* (0) number of cycles divider busy */
00284
00285 /* Memory ordering */
00286 #define P6_LD_BLOCKS 0x03 /* number of store buffer blocks */
00287 #define P6_SB_DRAINS 0x04 /* # of store buffer drain cycles */
00288 #define P6_MISALING_MEM_REF 0x05 /* # misaligned data memory refs */
00289
00290 /* Instruction decoding and retirement */
00291 #define P6_INST_RETIRED 0xc0 /* number of instrs retired */
00292 #define P6_OPS_RETIRED 0xc2 /* number of micro-ops retired */
00293 #define P6_INST_DECODER 0xd0 /* number of instructions decoded */
00294

```

```

00295 /* Interrupts */
00296 #define P6_HW_INT_RX 0xc8 /* number of hardware interrupts */
00297 #define P6_CYCLES_INT_MASKED 0xc6 /* number of cycles hardints masked */
00298 #define P6_CYCLES_INT_PENDING_AND_MASKED 0xc7 /* #cycles masked but pending */
00299
00300 /* Branches */
00301 #define P6_BR_INST_RETIRED 0xc4 /* number of branch instrs retired */
00302 #define P6_BR_MISS_PRED_RETIRED 0xc5 /* number of mispred'd brs retired */
00303 #define P6_BR_TAKEN_RETIRED 0xc9 /* number of taken branches retired */
00304 #define P6_BR_MISS_PRED_TAKEN_RET 0xca /* #taken mispredictions br's retired */
00305 #define P6_BR_INST_DECODED 0xe0 /* number of branch instrs decoded */
00306 #define P6_BTBMISSES 0xe2 /* # of branches that missed in BTB */
00307 #define P6_BR_BOGUS 0xe4 /* number of bogus branches */
00308 #define P6_BACLEAR 0xe6 /* # times BACLEAR is asserted */
00309
00310 /* Stalls */
00311 #define P6_RESOURCE_STALLS 0xa2 /* # resource-related stall cycles */
00312 #define P6_PARTIAL_RAT_STALLS 0xd2 /* # cycles/events for partial stalls */
00313
00314 /* Segment register loads */
00315 #define P6_SEGMENT_REG_LOADS 0x06 /* number of segment register loads */
00316
00317 /* Clocks */
00318 #define P6_CPU_CLK_UNHALTED 0x79 /* #clocks CPU is not halted */
00319
00320 /* Unit field tags */
00321 #define P6_UNIT_M 0x0800
00322 #define P6_UNIT_E 0x0400
00323 #define P6_UNIT_S 0x0200
00324 #define P6_UNIT_I 0x0100
00325 #define P6_UNIT_MESI 0x0f00
00326
00327 #define P6_UNIT_SELF 0x0000
00328 #define P6_UNIT_ANY 0x2000
00329
00330 /*****
00331  ** Flag bit definitions (used for the 'flag' field in select_p6counter()) **
00332  *****/
00333 *
00334 * The driver accepts fully-formed counter specifications from user-level.
00335 * The following flags are mnemonics for the bits that get set in the
00336 * PerfEvtSel0 and PerfEvtSel1 MSR's
00337 *
00338 */
00339 #define P6CNT_U 0x010000 /* Monitor user-level events */
00340 #define P6CNT_K 0x020000 /* Monitor kernel-level events */
00341 #define P6CNT_E 0x040000 /* Edge detect: count state transitions */
00342 #define P6CNT_PC 0x080000 /* Pin control: ?? */
00343 #define P6CNT_IE 0x100000 /* Int enable: enable interrupt on overflow */
00344 #define P6CNT_F 0x200000 /* Freeze counter (handled in software) */
00345 #define P6CNT_EN 0x400000 /* enable counters (in PerfEvtSel0) */
00346 #define P6CNT_IV 0x800000 /* Invert counter mask comparison result */
00347
00348 /*****
00349  ** Miscellaneous constants **
00350  *****/
00351 *
00352 * Number of Pentium Pro programable hardware counters.
00353 */
00354 #define NUM_P6HWC 2
00355
00356 /*****
00357  *
00358  * End of Copyright by Harvard College
00359  *
00360  *****/
00361
00362
00363 #define MSR_P6_EVTSEL0 0x186
00364 #define MSR_P6_EVTSEL1 0x187
00365 #define MSR_P6_PERFCTR0 0xc1
00366 #define MSR_P6_PERFCTR1 0xc2
00367
00368 /* P6-specific Makros to manipulate and read counters */
00369
00370 /* Read the 40 bit performance monitoring counter. This requires
00371 the PCE-flag in CR4 to be set. Otherwise GP0 is raised. Works only
00372 at P6.
00373 */
00374 #define l4_i686_rdpmc(cntnr, res_p) \
00375     __asm __volatile( \
00376         "mov %2, %%rcx    # put counter number in  \n\
00377         .byte 0xf; .byte 0x33 # RDPMC instruction  \n\
00378         mov %%rdx, %1     # High order 32 bits    \n\
00379         mov %%rax, %0     # Low order 32 bits" \
00380         : "=g" (*(int *) (res_p)), "=g" (((int *) res_p)+1)) \
00381         : "g" (cntnr) \

```

```

00382 : "ecx", "eax", "edx")
00383
00384 static inline l4_uint32_t l4_i686_rdpmc_32(int cnt){
00385     l4_uint32_t x;
00386
00387     __asm__ __volatile__(
00388         ".byte 0xf; .byte 0x33 # RDPMC instruction"
00389         : "=a" (x)
00390         : "c" (cnt)
00391         : "rcx", "rax", "rdx");
00392     return x;
00393 }
00394
00395 static inline void l4_i686_select_perfctr_event(int counter,
00396                                                 unsigned long long val){
00397     l4_i586_wrmsr(MSR_P6_EVNTSEL0+counter, &val);
00398 }
00399
00400 static inline void l4_i686_select_perfctr0_event(long long *val){
00401     __asm volatile(
00402         "mov $MSR_P6_EVNTSEL0, %%rcx\n"
00403         "mov (%%rbx), %%rax\n"
00404         "mov 4(%%rbx), %%rdx\n"
00405         /* ".byte 0xcc, 0xeb, 0x01, 0x21\n"
00406         ".byte 0x0f, 0x30\n" // wrmsr
00407         /* ".byte 0xcc, 0xeb, 0x01, 0x21\n"
00408         : /* no output */
00409         : "b" (val)
00410         : "ax", "cx", "dx", "bx"
00411         );
00412
00413 }
00414
00415 /* end of P6 section */
00416 #else
00417
00418 #define K7CNT_U 0x010000 /* Monitor user-level events */
00419 #define K7CNT_K 0x020000 /* Monitor kernel-level events */
00420 #define K7CNT_E 0x040000 /* Edge detect: count state transitions */
00421 #define K7CNT_PC 0x080000 /* Pin control: ?? */
00422 #define K7CNT_IE 0x100000 /* Int enable: enable interrupt on overflow */
00423 #define K7CNT_F 0x200000 /* Freeze counter (handled in software) */
00424 #define K7CNT_EN 0x400000 /* enable counters (in PerfEvtSel0) */
00425 #define K7CNT_IV 0x800000 /* Invert counter mask comparison result */
00426
00427 #define MSR_K7_EVNTSEL0 0xC0010000
00428 #define MSR_K7_EVNTSEL1 0xC0010001
00429 #define MSR_K7_EVNTSEL2 0xC0010002
00430 #define MSR_K7_EVNTSEL3 0xC0010003
00431 #define MSR_K7_PERFCTR0 0xC0010004
00432 #define MSR_K7_PERFCTR1 0xC0010005
00433 #define MSR_K7_PERFCTR2 0xC0010006
00434 #define MSR_K7_PERFCTR3 0xC0010007
00435
00436 #endif
00437
00438 #endif
00439
00440 /* end of P5/P6/K7 section*/
00441 #endif
00442
00443 /* end of not only lib-prototypes section */
00444 #endif
00445
00446 L4_END_DECLS
00447
00448 #endif

```

16.77 x86/i4/util/perform.h File Reference

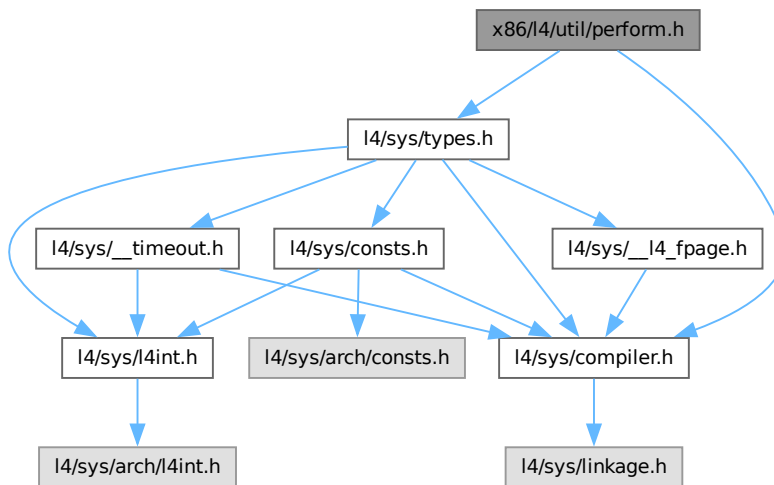
Performance Monitoring using P5/P6 Measurement Counters.

```

#include <l4/sys/types.h>
#include <l4/sys/compiler.h>

```

Include dependency graph for perform.h:



16.77.1 Detailed Description

Performance Monitoring using P5/P6 Measurement Counters.

Define either `CPU_PENTIUM` or `CPU_P6`

Definition in file [perform.h](#).

16.78 perform.h

[Go to the documentation of this file.](#)

```

00001
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *               Frank Mehnert <fm3@os.inf.tu-dresden.de>,
00010  *               Lars Reuther <reuther@os.inf.tu-dresden.de>
00011  *               economic rights: Technische Universität Dresden (Germany)
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014
00015 #ifndef __L4UTIL_PERFORM_H
00016 #define __L4UTIL_PERFORM_H
00017
00018 #include <l4/sys/types.h>
00019 #include <l4/sys/compiler.h>
00020
00021 L4_BEGIN_DECLS
00022
00023 extern const char*strp6pmc_event(l4_uint32_t event);
00024
00025 #ifndef CONFIG_PERFORM_ONLY_PROTOTYPES
00026
00027 #if ! (defined CPU_PENTIUM ^ defined CPU_P6 ^ defined CPU_K7)
00028
00029 #error You must define your target architecture.
00030 #error Define EITHER CPU_PENTIUM for Intel Pentium or CPU_P6 for Intel PPro/PII/PIII.
00031
00032 #else
00033

```

```

00034 /* P5/P6/K7 section */
00035
00036 /* Makros for access to model specific registers (MSR) */
00037
00038 /* Write the 64-Bit Model Specific Register. First argument is the register,
00039    second the 64-Bit value. This can only be called at privilege level 0.
00040    With L4, the kernel emulates the WRMSR when calling in PL 3.
00041    */
00042 static inline void l4_i586_wrmsr(unsigned reg,unsigned long long*val){
00043     unsigned long dummyeax, dummyecx, dummyedx;
00044
00045     asm volatile(
00046         ".byte 0xf; .byte 0x30\n" /* wrmsr */
00047         : "=a" (dummyeax), "=d" (dummyedx), "=c" (dummyecx)
00048         : "2" (reg), "0" (*(unsigned *)val), "1" (*(unsigned *)val+1))
00049     );
00050 }
00051
00052 /* Read the 64-Bit Model Specific Register. First argument is the register,
00053    second the address to a 64-Bit value. This can only be called at
00054    privilege level 0. With L4, the kernel emulates the RDMSR when calling
00055    in PL 3.
00056    */
00057 static inline void l4_i586_rdmsr(unsigned reg,unsigned long long*val){
00058     unsigned dummy;
00059
00060     asm volatile(
00061         ".byte 0xf; .byte 0x32\n" /* rdmsr */
00062         : "=a" (*(unsigned *)val), "=d" (*(unsigned *)val+1), "=c" (dummy)
00063         : "2" (reg)
00064     );
00065 }
00066
00067
00068 #ifdef CPU_PENTIUM
00069 /* Pentium section */
00070
00071 /* functions and events defined here are only usable at Pentium
00072    Processors. P6 architecture does NOT support this kind of measuring and
00073    these events. P6 architecture has its own counters and its own events.
00074    See P6-section for details. */
00075
00076 /* from l4linux/arch/l4-i386/include/perform.h */
00077
00078 static inline void
00079 l4_i586_reset_event_counter(void){
00080     asm volatile("xor %%eax, %%eax\n"
00081         "xor %%edx, %%edx\n"
00082         "movl $0x12, %%ecx\n"
00083         ".byte 0x0f, 0x30\n"
00084         "movl $0x13, %%ecx\n"
00085         ".byte 0x0f, 0x30\n"
00086         : : : "cx", "ax", "dx"
00087     );
00088 };
00089
00090 static inline void
00091 l4_i586_read_event_counter_long(long long *counter0, long long *counter1)
00092 {
00093     asm volatile(
00094         /*      "movl $0, %%eax\n"
00095         "movl $0x11, %%ecx\n"
00096         ".byte 0x0f, 0x30\n" *//* stop event counting */
00097         "movl $0x12, %%ecx\n"
00098         ".byte 0x0f, 0x32\n"
00099         "movl %%eax, (%%ebx)\n"
00100         "movl %%edx, 4(%%ebx)\n"
00101         "movl $0x13, %%ecx\n"
00102         ".byte 0x0f, 0x32\n"
00103         "movl %%eax, (%%esi)\n"
00104         "movl %%edx, 4(%%esi)\n"
00105         : /* no output */
00106         : "b" (counter0), "S" (counter1)
00107         : "ax", "cx", "dx"
00108     );
00109 }
00110
00111 static inline void
00112 l4_i586_read_event_counter(int *counter0, int *counter1)
00113 {
00114     asm volatile("pushl %%edx\n"
00115         ".byte 0x0f, 0x30\n"
00116         "movl $0x12, %%ecx\n"
00117         ".byte 0x0f, 0x32\n"
00118         "movl %%eax, %%ebx\n"
00119         "movl $0x13, %%ecx\n"
00120         ".byte 0x0f, 0x32\n"

```



```

00121         "popl  %%edx\n"
00122         : "=b" (*counter0), "=a" (*counter1)
00123         : "1" (0), "c" (0x11)
00124         );
00125     }
00126
00127     static inline void
00128     l4_i586_select_event(int event0, int event1)
00129     {
00130         asm volatile(".byte 0x0f, 0x30\n"
00131             :
00132             :
00133             "a" (event0 + (event1 < 16)),
00134             "d" (0),
00135             "c" (0x11)
00136             );
00137     };
00138
00139     #define P5_RD_MISS          0x003 /* 000011B */
00140     #define P5_WR_MISS          0x008 /* 000100B */
00141     #define P5_RW_MISS          0x029 /* 101001B */
00142     #define P5_EX_MISS          0x00e /* 001110B */
00143
00144     #define P5_D_WBACK          0x006 /* 000110B */
00145
00146     #define P5_RW_TLB           0x002 /* 00010B */
00147     #define P5_EX_TLB           0x00d /* 01101B */
00148
00149     #define P5_A_STALL           0x01f /* 11111B */
00150     #define P5_W_STALL           0x019 /* 11001B */
00151     #define P5_R_STALL           0x01a /* 11010B */
00152     #define P5_X_STALL           0x01b /* 11011B */
00153
00154     #define P5_AGI_STALL         0x01f /* 11111B */
00155
00156     #define P5_PIPELINE_FLUSH    0x015 /* 10101B */
00157
00158     #define P5_NON_CACHE_RD      0x01e /* 11110B */
00159     #define P5_NCACHE_REFS       0x01e /* 11110B */
00160     #define P5_LOCKED_BUS        0x01c /* 11100B */
00161
00162     #define P5_MEM2PIPE           0x009 /* 01001B */
00163     #define P5_BANK_CONF         0x00a /* 01010B */
00164
00165
00166     #define P5_INSTRS_EX          0x016 /* 10110B */
00167     #define P5_INSTRS_EX_V        0x017 /* 10111B */
00168
00169
00170     #define P5_CNT_NOTHING        (0x00 < 6) /* 00B < 6 */
00171     #define P5_CNT_EVENT_PL0      (0x01 < 6) /* 01B < 6 */
00172     #define P5_CNT_EVENT_PL3      (0x02 < 6) /* 10B < 6 */
00173     #define P5_CNT_EVENT          (0x03 < 6) /* 11B < 6 */
00174     #define P5_CNT_CLOCKS_PL0     (0x05 < 6) /* 101B < 6 */
00175     #define P5_CNT_CLOCKS_PL3     (0x06 < 6) /* 110B < 6 */
00176     #define P5_CNT_CLOCKS         (0x07 < 6) /* 111B < 6 */
00177
00178
00179     #else
00180     #if defined CPU_P6
00181     /* PPro/PII/PIII section */
00182
00183     /*-
00184     * Copyright (c) 1997 The President and Fellows of Harvard College.
00185     * All rights reserved.
00186     * Copyright (c) 1997 Aaron B. Brown.
00187     *
00188     * Redistribution and use in source and binary forms, with or without
00189     * modification, are permitted provided that the following conditions
00190     * are met:
00191     * 1. Redistributions of source code must retain the above copyright
00192     *   notice, this list of conditions and the following disclaimer.
00193     * 2. Redistributions in binary form must reproduce the above copyright
00194     *   notice, this list of conditions and the following disclaimer in the
00195     *   documentation and/or other materials provided with the distribution.
00196     * 3. All advertising materials mentioning features or use of this software
00197     *   must display the following acknowledgement:
00198     *     This product includes software developed by Harvard University
00199     *     and its contributors.
00200     * 4. Neither the name of the University nor the names of its contributors
00201     *   may be used to endorse or promote products derived from this software
00202     *   without specific prior written permission.
00203     *
00204     * THIS SOFTWARE IS PROVIDED BY HARVARD AND CONTRIBUTORS ``AS IS'' AND
00205     * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
00206     * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
00207     * ARE DISCLAIMED.  IN NO EVENT SHALL HARVARD UNIVERSITY OR CONTRIBUTORS BE

```

```

00208 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
00209 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
00210 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
00211 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
00212 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
00213 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
00214 * POSSIBILITY OF SUCH DAMAGE.
00215 */
00216
00217 /*****
00218 ** Symbolic names for counter numbers (used in select_p6counter()) **
00219 *****/
00220 *
00221 * These correspond in order to the Pentium Pro counters. Add new counters at
00222 * the end. These agree with the mnemonics in the Pentium Pro Family
00223 * Developer's Manual, vol 3.
00224 *
00225 * Those events marked with a $ require a MESI unit field; those marked with
00226 * a @ require a self/any unit field. Those marked with a 0 are only supported
00227 * in counter 0; those marked with 1 are only supported in counter 1.
00228 */
00229
00230 /* Data cache unit */
00231 #define P6_DATA_MEM_REFS 0x43 /* total memory refs */
00232 #define P6_DCU_LINES_IN 0x45 /* all lines allocated in cache unit */
00233 #define P6_DCU_M_LINES_IN 0x46 /* M lines allocated in cache unit */
00234 #define P6_DCU_M_LINES_OUT 0x47 /* M lines evicted from cache */
00235 #define P6_DCU_MISS_OUTSTANDING 0x48 /* #cycles a miss is outstanding */
00236
00237 /* Instruction fetch unit */
00238 #define P6_IFU_IFETCH 0x80 /* instruction fetches */
00239 #define P6_IFU_IFETCH_MISS 0x81 /* instruction fetch misses */
00240 #define P6_ITLB_MISS 0x85 /* ITLB misses */
00241 #define P6_IFU_MEM_STALL 0x86 /* number of cycles IFU is stalled */
00242 #define P6_ILD_STALL 0x87 /* #stalls in instr length decode */
00243
00244 /* L2 Cache */
00245 #define P6_L2_IFETCH 0x28 /* ($) l2 ifetches */
00246 #define P6_L2_LD 0x29 /* ($) l2 data loads */
00247 #define P6_L2_ST 0x2a /* ($) l2 data stores */
00248 #define P6_L2_LINES_IN 0x24 /* lines allocated in l2 */
00249 #define P6_L2_LINES_OUT 0x26 /* lines removed from l2 */
00250 #define P6_L2_M_LINES_INM 0x25 /* modified lines allocated in l2 */
00251 #define P6_L2_M_LINES_OUTM 0x27 /* modified lines removed from l2 */
00252 #define P6_L2_RQSTS 0x2e /* ($) number of l2 requests */
00253 #define P6_L2_ADS 0x21 /* number of l2 addr strobes */
00254 #define P6_L2_DBUS_BUSY 0x22 /* number of data bus busy cycles */
00255 #define P6_L2_DBUS_BUSY_RD 0x23 /* #bus cycles xferring l2->cpu */
00256
00257 /* External bus logic */
00258 #define P6_BUS_DRDY_CLOCKS 0x62 /* (@) #clocks DRDY is asserted */
00259 #define P6_BUS_LOCK_CLOCKS 0x63 /* (@) #clocks LOCK is asserted */
00260 #define P6_BUS_REQ_OUTSTANDING 0x60 /* #bus requests outstanding */
00261 #define P6_BUS_TRAN_BRD 0x65 /* (@) bus burst read txns */
00262 #define P6_BUS_TRAN_RFO 0x66 /* (@) bus read for ownership txns */
00263 #define P6_BUS_TRAN_WB 0x67 /* (@) bus writeback txns */
00264 #define P6_BUS_TRAN_IFETCH 0x68 /* (@) bus instr fetch txns */
00265 #define P6_BUS_TRAN_INVALID 0x69 /* (@) bus invalidate txns */
00266 #define P6_BUS_TRAN_PWR 0x6a /* (@) bus partial write txns */
00267 #define P6_BUS_TRANS_P 0x6b /* (@) bus partial txns */
00268 #define P6_BUS_TRANS_IO 0x6c /* (@) bus I/O txns */
00269 #define P6_BUS_TRAN_DEF 0x6d /* (@) bus deferred txns */
00270 #define P6_BUS_TRAN_BURST 0x6e /* (@) bus burst txns */
00271 #define P6_BUS_TRAN_ANY 0x70 /* (@) total bus txns */
00272 #define P6_BUS_TRAN_MEM 0x6f /* (@) total memory txns */
00273 #define P6_BUS_DATA_RCV 0x64 /* #busclocks CPU is receiving data */
00274 #define P6_BUS_BNR_DRV 0x61 /* #busclocks CPU is driving BNR pin */
00275 #define P6_BUS_HIT_DRV 0x7a /* #busclocks CPU is driving HIT pin */
00276 #define P6_BUS_HITM_DRV 0x7b /* #busclocks CPU is driving HITM pin */
00277 #define P6_BUS_SNOOP_STALL 0x7e /* #clkcycles bus is snoop-stalled */
00278
00279 /* FPU */
00280 #define P6_FLOPS 0xc1 /* (0) number of FP ops retired */
00281 #define P6_FP_COMP_OPS 0x10 /* (0) computational FPOPS exec'd */
00282 #define P6_FP_ASSIST 0x11 /* (1) FP excep's handled in ucode */
00283 #define P6_MUL 0x12 /* (1) number of FP multiplies */
00284 #define P6_DIV 0x13 /* (1) number of FP divides */
00285 #define P6_CYCLES_DIV_BUSY 0x14 /* (0) number of cycles divider busy */
00286
00287 /* Memory ordering */
00288 #define P6_LD_BLOCKS 0x03 /* number of store buffer blocks */
00289 #define P6_SB_DRAINS 0x04 /* # of store buffer drain cycles */
00290 #define P6_MISALING_MEM_REF 0x05 /* # misaligned data memory refs */
00291
00292 /* Instruction decoding and retirement */
00293 #define P6_INST_RETIRED 0xc0 /* number of instrs retired */
00294 #define P6_UOPS_RETIRED 0xc2 /* number of micro-ops retired */

```

Generated for L4Re by Doxygen

```

00382 : "=g" (*(int *) (res_p)), "=g" (*((int *)res_p)+1)) \
00383 : "g" (cntr) \
00384 : "ecx", "eax", "edx")
00385
00386 static inline l4_uint32_t l4_i686_rdpmc_32(int cntr){
00387     l4_uint32_t x;
00388
00389     __asm__ __volatile__(
00390         ".byte 0xf; .byte 0x33 # RDPMC instruction"
00391         : "=a" (x)
00392         : "c" (cntr)
00393         : "ecx", "eax", "edx");
00394     return x;
00395 }
00396
00397 static inline void l4_i686_select_perfctr_event(int counter,
00398                                                 unsigned long long val){
00399     l4_i586_wrmsr(MSR_P6_EVNTSEL0+counter, &val);
00400 }
00401
00402 static inline void l4_i686_select_perfctr0_event(long long *val){
00403     __asm volatile(
00404         "movl $MSR_P6_EVNTSEL0, %%ecx\n"
00405         "movl (%%ebx), %%eax\n"
00406         "movl 4(%%ebx), %%edx\n"
00407         /* ".byte 0xcc, 0xeb, 0x01, 0x21\n"
00408         ".byte 0x0f, 0x30\n" // wrmsr
00409         /* ".byte 0xcc, 0xeb, 0x01, 0x21\n"
00410         : /* no output */
00411         : "b" (val)
00412         : "ax", "cx", "dx", "bx"
00413         );
00414
00415 }
00416
00417 /* end of P6 section */
00418 #else
00419
00420 #define K7CNT_U 0x010000 /* Monitor user-level events */
00421 #define K7CNT_K 0x020000 /* Monitor kernel-level events */
00422 #define K7CNT_E 0x040000 /* Edge detect: count state transitions */
00423 #define K7CNT_PC 0x080000 /* Pin control: ?? */
00424 #define K7CNT_IE 0x100000 /* Int enable: enable interrupt on overflow */
00425 #define K7CNT_F 0x200000 /* Freeze counter (handled in software) */
00426 #define K7CNT_EN 0x400000 /* enable counters (in PerfEvtSel0) */
00427 #define K7CNT_IV 0x800000 /* Invert counter mask comparison result */
00428
00429 #define MSR_K7_EVNTSEL0 0xC0010000
00430 #define MSR_K7_EVNTSEL1 0xC0010001
00431 #define MSR_K7_EVNTSEL2 0xC0010002
00432 #define MSR_K7_EVNTSEL3 0xC0010003
00433 #define MSR_K7_PERFCTR0 0xC0010004
00434 #define MSR_K7_PERFCTR1 0xC0010005
00435 #define MSR_K7_PERFCTR2 0xC0010006
00436 #define MSR_K7_PERFCTR3 0xC0010007
00437
00438 #endif
00439
00440 #endif
00441
00442 /* end of P5/P6/K7 section*/
00443 #endif
00444
00445 /* end of not only lib-prototypes section */
00446 #endif
00447
00448 L4_END_DECLS
00449
00450 #endif

```

16.79 amd64/l4/util/port_io.h File Reference

x86 port I/O

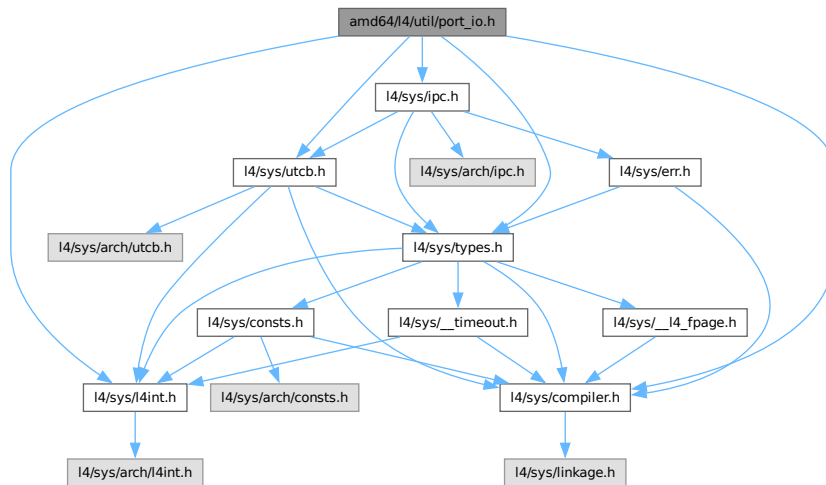
```

#include <l4/sys/l4int.h>
#include <l4/sys/compiler.h>
#include <l4/sys/types.h>
#include <l4/sys/utcb.h>

```

```
#include <l4/sys/ipc.h>
```

Include dependency graph for port_io.h:



Functions

- `l4_uint8_t l4util_in8 (l4_uint16_t port)`
Read byte from I/O port.
- `l4_uint16_t l4util_in16 (l4_uint16_t port)`
Read 16-bit-value from I/O port.
- `l4_uint32_t l4util_in32 (l4_uint16_t port)`
Read 32-bit-value from I/O port.
- `void l4util_ins8 (l4_uint16_t port, l4_umword_t addr, l4_umword_t count)`
Read a block of 8-bit-values from I/O ports.
- `void l4util_ins16 (l4_uint16_t port, l4_umword_t addr, l4_umword_t count)`
Read a block of 16-bit-values from I/O ports.
- `void l4util_ins32 (l4_uint16_t port, l4_umword_t addr, l4_umword_t count)`
Read a block of 32-bit-values from I/O ports.
- `void l4util_out8 (l4_uint8_t value, l4_uint16_t port)`
Write byte to I/O port.
- `void l4util_out16 (l4_uint16_t value, l4_uint16_t port)`
Write 16-bit-value to I/O port.
- `void l4util_out32 (l4_uint32_t value, l4_uint16_t port)`
Write 32-bit-value to I/O port.
- `void l4util_outs8 (l4_uint16_t port, l4_umword_t addr, l4_umword_t count)`
Write a block of bytes to I/O port.
- `void l4util_outs16 (l4_uint16_t port, l4_umword_t addr, l4_umword_t count)`
Write a block of 16-bit-values to I/O port.
- `void l4util_outs32 (l4_uint16_t port, l4_umword_t addr, l4_umword_t count)`
Write block of 32-bit-values to I/O port.
- `void l4util_iodelay (void)`
delay I/O port access by writing to port 0x80
- `int l4util_ioport_map (l4_cap_idx_t sigma0id, unsigned port_start, unsigned log2size)`
Map a range of I/O ports.

16.79.1 Detailed Description

x86 port I/O

Date

06/2003

Author

Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file [port_io.h](#).

16.80 port_io.h

[Go to the documentation of this file.](#)

```

00001  /*****
00009  /*****
00010
00011  /*
00012   * (c) 2003-2009 Author(s)
00013   *     economic rights: Technische Universität Dresden (Germany)
00014   * License: see LICENSE.spdx (in this directory or the directories above)
00015   */
00016
00017 #ifndef _L4UTIL_PORT_IO_H
00018 #define _L4UTIL_PORT_IO_H
00019
00024
00025 /* L4 includes */
00026 #include <l4/sys/l4int.h>
00027 #include <l4/sys/compiler.h>
00028 #include <l4/sys/types.h>
00029
00030 /*****
00031  *** Prototypes
00032  *****/
00033
00034 L4_BEGIN_DECLS
00045 L4_INLINE l4_uint8_t
00046 l4util_in8(l4_uint16_t port);
00047
00054 L4_INLINE l4_uint16_t
00055 l4util_in16(l4_uint16_t port);
00056
00063 L4_INLINE l4_uint32_t
00064 l4util_in32(l4_uint16_t port);
00065
00073 L4_INLINE void
00074 l4util_ins8(l4_uint16_t port, l4_umword_t addr, l4_umword_t count);
00075
00083 L4_INLINE void
00084 l4util_ins16(l4_uint16_t port, l4_umword_t addr, l4_umword_t count);
00085
00093 L4_INLINE void
00094 l4util_ins32(l4_uint16_t port, l4_umword_t addr, l4_umword_t count);
00095
00102 L4_INLINE void
00103 l4util_out8(l4_uint8_t value, l4_uint16_t port);
00104
00111 L4_INLINE void
00112 l4util_out16(l4_uint16_t value, l4_uint16_t port);
00113
00120 L4_INLINE void
00121 l4util_out32(l4_uint32_t value, l4_uint16_t port);
00122
00130 L4_INLINE void
00131 l4util_outs8(l4_uint16_t port, l4_umword_t addr, l4_umword_t count);
00132
00140 L4_INLINE void
00141 l4util_outs16(l4_uint16_t port, l4_umword_t addr, l4_umword_t count);

```

```

00142
00150 L4_INLINE void
00151 l4util_outs32(l4_uint16_t port, l4_umword_t addr, l4_umword_t count);
00152
00156 L4_INLINE void
00157 l4util_iodelay(void);
00158
00170 L4_INLINE int
00171 l4util_ioport_map(l4_cap_idx_t sigma0id,
00172                  unsigned port_start, unsigned log2size);
00173
00175
00176 L4_END_DECLS
00177
00178
00179 /***** Implementation *****/
00180 *** Implementation
00181 *****/
00182
00183 #include <l4/sys/utcb.h>
00184 #include <l4/sys/ipc.h>
00185
00186 L4_INLINE l4_uint8_t
00187 l4util_in8(l4_uint16_t port)
00188 {
00189     l4_uint8_t value;
00190     asm volatile ("inb %w1, %b0" : "=a" (value) : "Nd" (port));
00191     return value;
00192 }
00193
00194 L4_INLINE l4_uint16_t
00195 l4util_in16(l4_uint16_t port)
00196 {
00197     l4_uint16_t value;
00198     asm volatile ("inw %w1, %w0" : "=a" (value) : "Nd" (port));
00199     return value;
00200 }
00201
00202 L4_INLINE l4_uint32_t
00203 l4util_in32(l4_uint16_t port)
00204 {
00205     l4_uint32_t value;
00206     asm volatile ("inl %w1, %0" : "=a" (value) : "Nd" (port));
00207     return value;
00208 }
00209
00210 L4_INLINE void
00211 l4util_ins8(l4_uint16_t port, l4_umword_t addr, l4_umword_t count)
00212 {
00213     l4_umword_t dummy1, dummy2;
00214     asm volatile ("rep insb" : "=D" (dummy1), "=c" (dummy2)
00215                  : "d" (port), "D" (addr), "c" (count)
00216                  : "memory");
00217 }
00218
00219 L4_INLINE void
00220 l4util_ins16(l4_uint16_t port, l4_umword_t addr, l4_umword_t count)
00221 {
00222     l4_umword_t dummy1, dummy2;
00223     asm volatile ("rep insw" : "=D" (dummy1), "=c" (dummy2)
00224                  : "d" (port), "D" (addr), "c" (count)
00225                  : "memory");
00226 }
00227
00228 L4_INLINE void
00229 l4util_ins32(l4_uint16_t port, l4_umword_t addr, l4_umword_t count)
00230 {
00231     l4_umword_t dummy1, dummy2;
00232     asm volatile ("rep insl" : "=D" (dummy1), "=c" (dummy2)
00233                  : "d" (port), "D" (addr), "c" (count)
00234                  : "memory");
00235 }
00236
00237 L4_INLINE void
00238 l4util_out8(l4_uint8_t value, l4_uint16_t port)
00239 {
00240     asm volatile ("outb %b0, %w1" : : "a" (value), "Nd" (port));
00241 }
00242
00243 L4_INLINE void
00244 l4util_out16(l4_uint16_t value, l4_uint16_t port)
00245 {
00246     asm volatile ("outw %w0, %w1" : : "a" (value), "Nd" (port));
00247 }
00248
00249 L4_INLINE void
00250 l4util_out32(l4_uint32_t value, l4_uint16_t port)

```

```

00251 {
00252     asm volatile ("outl %0, %w1" : : "a" (value), "Nd" (port));
00253 }
00254
00255 L4_INLINE void
00256 l4util_outs8(l4_uint16_t port, l4_umword_t addr, l4_umword_t count)
00257 {
00258     l4_umword_t dummy1, dummy2;
00259     asm volatile ("rep outsb" : "=S"(dummy1), "=c"(dummy2)
00260                  : "d" (port), "S" (addr), "c"(count)
00261                  : "memory");
00262 }
00263
00264 L4_INLINE void
00265 l4util_outs16(l4_uint16_t port, l4_umword_t addr, l4_umword_t count)
00266 {
00267     l4_umword_t dummy1, dummy2;
00268     asm volatile ("rep outsw" : "=S"(dummy1), "=c"(dummy2)
00269                  : "d" (port), "S" (addr), "c"(count)
00270                  : "memory");
00271 }
00272
00273 L4_INLINE void
00274 l4util_outs32(l4_uint16_t port, l4_umword_t addr, l4_umword_t count)
00275 {
00276     l4_umword_t dummy1, dummy2;
00277     asm volatile ("rep outsl" : "=S"(dummy1), "=c"(dummy2)
00278                  : "d" (port), "S" (addr), "c"(count)
00279                  : "memory");
00280 }
00281
00282 L4_INLINE void
00283 l4util_iodelay(void)
00284 {
00285     asm volatile ("outb %a1,$0x80");
00286 }
00287
00288 L4_INLINE int
00289 l4util_ioport_map(l4_cap_idx_t sigma0id,
00290                  unsigned port_start, unsigned log2size)
00291 {
00292     l4_fpage_t iofp;
00293     l4_msgtag_t tag;
00294     long err;
00295
00296     iofp = l4_iofpage(port_start, log2size);
00297     l4_utcb_mr()->mr[0] = iofp.raw;
00298     l4_utcb_br()->bdr = 0;
00299     l4_utcb_br()->br[0] = L4_ITEM_MAP;
00300     l4_utcb_br()->br[1] = iofp.raw;
00301     tag = l4_ipc_call(sigma0id, l4_utcb(),
00302                     l4_msgtag(L4_PROTO_IO_PAGE_FAULT, 1, 0, 0),
00303                     L4_IPC_NEVER);
00304
00305     if ((err = l4_ipc_error(tag, l4_utcb())))
00306         return err;
00307
00308     return l4_msgtag_items(tag) > 0 ? 0 : -L4_ENOENT;
00309 }
00310
00311 #endif

```

16.81 x86/l4/util/port_io.h File Reference

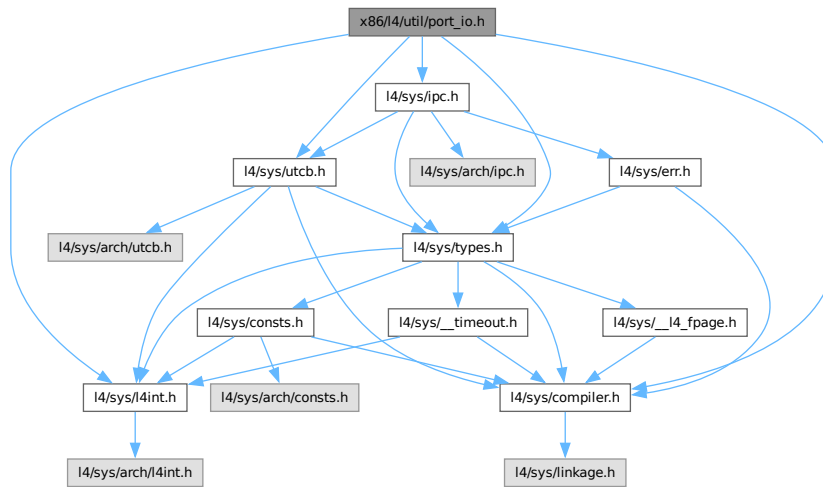
x86 port I/O

```

#include <l4/sys/l4int.h>
#include <l4/sys/compiler.h>
#include <l4/sys/types.h>
#include <l4/sys/utcb.h>
#include <l4/sys/ipc.h>

```


Include dependency graph for port_io.h:



Functions

- [i4_uint8_t i4util_in8 \(i4_uint16_t port\)](#)
Read byte from I/O port.
- [i4_uint16_t i4util_in16 \(i4_uint16_t port\)](#)
Read 16-bit-value from I/O port.
- [i4_uint32_t i4util_in32 \(i4_uint16_t port\)](#)
Read 32-bit-value from I/O port.
- [void i4util_ins8 \(i4_uint16_t port, i4_umword_t addr, i4_umword_t count\)](#)
Read a block of 8-bit-values from I/O ports.
- [void i4util_ins16 \(i4_uint16_t port, i4_umword_t addr, i4_umword_t count\)](#)
Read a block of 16-bit-values from I/O ports.
- [void i4util_ins32 \(i4_uint16_t port, i4_umword_t addr, i4_umword_t count\)](#)
Read a block of 32-bit-values from I/O ports.
- [void i4util_out8 \(i4_uint8_t value, i4_uint16_t port\)](#)
Write byte to I/O port.
- [void i4util_out16 \(i4_uint16_t value, i4_uint16_t port\)](#)
Write 16-bit-value to I/O port.
- [void i4util_out32 \(i4_uint32_t value, i4_uint16_t port\)](#)
Write 32-bit-value to I/O port.
- [void i4util_outs8 \(i4_uint16_t port, i4_umword_t addr, i4_umword_t count\)](#)
Write a block of bytes to I/O port.
- [void i4util_outs16 \(i4_uint16_t port, i4_umword_t addr, i4_umword_t count\)](#)
Write a block of 16-bit-values to I/O port.
- [void i4util_outs32 \(i4_uint16_t port, i4_umword_t addr, i4_umword_t count\)](#)
Write block of 32-bit-values to I/O port.
- [void i4util_iodelay \(void\)](#)
delay I/O port access by writing to port 0x80
- [int i4util_ioport_map \(i4_cap_idx_t sigma0id, unsigned port_start, unsigned log2size\)](#)
Map a range of I/O ports.

16.81.1 Detailed Description

x86 port I/O

Date

06/2003

Author

Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file [port_io.h](#).

16.82 port_io.h

[Go to the documentation of this file.](#)

```

00001  /*****
00009  /*****
00010
00011  /*
00012   * (c) 2003-2009 Author(s)
00013   *     economic rights: Technische Universität Dresden (Germany)
00014   * License: see LICENSE.spdx (in this directory or the directories above)
00015   */
00016
00017 #ifndef _L4UTIL_PORT_IO_H
00018 #define _L4UTIL_PORT_IO_H
00019
00024
00025 /* L4 includes */
00026 #include <l4/sys/l4int.h>
00027 #include <l4/sys/compiler.h>
00028 #include <l4/sys/types.h>
00029
00030 /*****
00031  *** Prototypes
00032  *****/
00033
00034 L4_BEGIN_DECLS
00045 L4_INLINE l4_uint8_t
00046 l4util_in8(l4_uint16_t port);
00047
00054 L4_INLINE l4_uint16_t
00055 l4util_in16(l4_uint16_t port);
00056
00063 L4_INLINE l4_uint32_t
00064 l4util_in32(l4_uint16_t port);
00065
00073 L4_INLINE void
00074 l4util_ins8(l4_uint16_t port, l4_umword_t addr, l4_umword_t count);
00075
00083 L4_INLINE void
00084 l4util_ins16(l4_uint16_t port, l4_umword_t addr, l4_umword_t count);
00085
00093 L4_INLINE void
00094 l4util_ins32(l4_uint16_t port, l4_umword_t addr, l4_umword_t count);
00095
00102 L4_INLINE void
00103 l4util_out8(l4_uint8_t value, l4_uint16_t port);
00104
00111 L4_INLINE void
00112 l4util_out16(l4_uint16_t value, l4_uint16_t port);
00113
00120 L4_INLINE void
00121 l4util_out32(l4_uint32_t value, l4_uint16_t port);
00122
00130 L4_INLINE void
00131 l4util_outs8(l4_uint16_t port, l4_umword_t addr, l4_umword_t count);
00132
00140 L4_INLINE void
00141 l4util_outs16(l4_uint16_t port, l4_umword_t addr, l4_umword_t count);

```

```

00142
00150 L4_INLINE void
00151 l4util_outs32(l4_uint16_t port, l4_umword_t addr, l4_umword_t count);
00152
00156 L4_INLINE void
00157 l4util_iodelay(void);
00158
00170 L4_INLINE int
00171 l4util_ioport_map(l4_cap_idx_t sigma0id,
00172                  unsigned port_start, unsigned log2size);
00173
00175
00176 L4_END_DECLS
00177
00178
00179 /***** Implementation *****/
00180 *** Implementation
00181 *****/
00182
00183 #include <l4/sys/utcb.h>
00184 #include <l4/sys/ipc.h>
00185
00186 L4_INLINE l4_uint8_t
00187 l4util_in8(l4_uint16_t port)
00188 {
00189     l4_uint8_t value;
00190     asm volatile ("inb %w1, %b0" : "=a" (value) : "Nd" (port));
00191     return value;
00192 }
00193
00194 L4_INLINE l4_uint16_t
00195 l4util_in16(l4_uint16_t port)
00196 {
00197     l4_uint16_t value;
00198     asm volatile ("inw %w1, %w0" : "=a" (value) : "Nd" (port));
00199     return value;
00200 }
00201
00202 L4_INLINE l4_uint32_t
00203 l4util_in32(l4_uint16_t port)
00204 {
00205     l4_uint32_t value;
00206     asm volatile ("inl %w1, %0" : "=a" (value) : "Nd" (port));
00207     return value;
00208 }
00209
00210 L4_INLINE void
00211 l4util_ins8(l4_uint16_t port, l4_umword_t addr, l4_umword_t count)
00212 {
00213     l4_umword_t dummy1, dummy2;
00214     asm volatile ("rep insb" : "=D" (dummy1), "=c" (dummy2)
00215                  : "d" (port), "D" (addr), "c" (count)
00216                  : "memory");
00217 }
00218
00219 L4_INLINE void
00220 l4util_ins16(l4_uint16_t port, l4_umword_t addr, l4_umword_t count)
00221 {
00222     l4_umword_t dummy1, dummy2;
00223     asm volatile ("rep insw" : "=D" (dummy1), "=c" (dummy2)
00224                  : "d" (port), "D" (addr), "c" (count)
00225                  : "memory");
00226 }
00227
00228 L4_INLINE void
00229 l4util_ins32(l4_uint16_t port, l4_umword_t addr, l4_umword_t count)
00230 {
00231     l4_umword_t dummy1, dummy2;
00232     asm volatile ("rep insl" : "=D" (dummy1), "=c" (dummy2)
00233                  : "d" (port), "D" (addr), "c" (count)
00234                  : "memory");
00235 }
00236
00237 L4_INLINE void
00238 l4util_out8(l4_uint8_t value, l4_uint16_t port)
00239 {
00240     asm volatile ("outb %b0, %w1" : : "a" (value), "Nd" (port));
00241 }
00242
00243 L4_INLINE void
00244 l4util_out16(l4_uint16_t value, l4_uint16_t port)
00245 {
00246     asm volatile ("outw %w0, %w1" : : "a" (value), "Nd" (port));
00247 }
00248
00249 L4_INLINE void
00250 l4util_out32(l4_uint32_t value, l4_uint16_t port)

```

```

00251 {
00252     asm volatile ("outl %0, %w1" : : "a" (value), "Nd" (port));
00253 }
00254
00255 L4_INLINE void
00256 l4util_outs8(l4_uint16_t port, l4_umword_t addr, l4_umword_t count)
00257 {
00258     l4_umword_t dummy1, dummy2;
00259     asm volatile ("rep outsb" : "=S"(dummy1), "=c"(dummy2)
00260                  : "d" (port), "S" (addr), "c"(count)
00261                  : "memory");
00262 }
00263
00264 L4_INLINE void
00265 l4util_outs16(l4_uint16_t port, l4_umword_t addr, l4_umword_t count)
00266 {
00267     l4_umword_t dummy1, dummy2;
00268     asm volatile ("rep outsw" : "=S"(dummy1), "=c"(dummy2)
00269                  : "d" (port), "S" (addr), "c"(count)
00270                  : "memory");
00271 }
00272
00273 L4_INLINE void
00274 l4util_outs32(l4_uint16_t port, l4_umword_t addr, l4_umword_t count)
00275 {
00276     l4_umword_t dummy1, dummy2;
00277     asm volatile ("rep outsl" : "=S"(dummy1), "=c"(dummy2)
00278                  : "d" (port), "S" (addr), "c"(count)
00279                  : "memory");
00280 }
00281
00282 L4_INLINE void
00283 l4util_iodelay(void)
00284 {
00285     asm volatile ("outb %a1,$0x80");
00286 }
00287
00288 L4_INLINE int
00289 l4util_ioport_map(l4_cap_idx_t sigma0id,
00290                  unsigned port_start, unsigned log2size)
00291 {
00292     l4_fpage_t iofp;
00293     l4_msgtag_t tag;
00294     long err;
00295
00296     iofp = l4_iofpage(port_start, log2size);
00297     l4_utcb_mr()->mr[0] = iofp.raw;
00298     l4_utcb_br()->bdr = 0;
00299     l4_utcb_br()->br[0] = L4_ITEM_MAP;
00300     l4_utcb_br()->br[1] = iofp.raw;
00301     tag = l4_ipc_call(sigma0id, l4_utcb(),
00302                      l4_msgtag(L4_PROTO_IO_PAGE_FAULT, 1, 0, 0),
00303                      L4_IPC_NEVER);
00304
00305     if ((err = l4_ipc_error(tag, l4_utcb())))
00306         return err;
00307
00308     return l4_msgtag_items(tag) > 0 ? 0 : -L4_ENOENT;
00309 }
00310
00311 #endif

```

16.83 amd64/l4/util/rdtsc.h File Reference

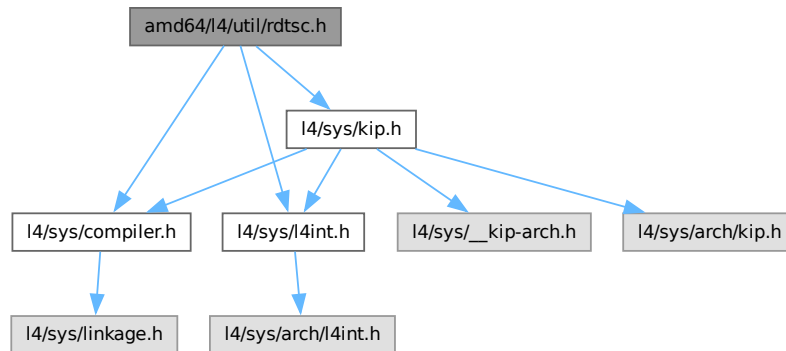
Timestamp counter related functions.

```

#include <l4/sys/compiler.h>
#include <l4/sys/l4int.h>
#include <l4/sys/kip.h>

```

Include dependency graph for rdtsc.h:



Functions

- `l4_cpu_time_t l4_rdtsc` (void)
Read current value of CPU-internal timestamp counter.
- `l4_uint32_t l4_rdtsc_32` (void)
Read the lest significant 32 bit of the TSC.
- `l4_uint64_t l4_rdpmc` (int ecx)
Return current value of CPU-internal performance measurement counter.
- `l4_uint32_t l4_rdpmc_32` (int ecx)
Return the least significant 32 bit of a performance counter.
- `l4_uint64_t l4_tsc_to_ns` (`l4_cpu_time_t` tsc)
Convert timestamp to ns value.
- `l4_uint64_t l4_tsc_to_us` (`l4_cpu_time_t` tsc)
Convert timestamp into micro seconds value.
- `void l4_tsc_to_s_and_ns` (`l4_cpu_time_t` tsc, `l4_uint32_t` *s, `l4_uint32_t` *ns)
Convert timestamp to s.ns value.
- `l4_cpu_time_t l4_ns_to_tsc` (`l4_uint64_t` ns)
Convert nano seconds into CPU ticks.
- `void l4_busy_wait_ns` (`l4_uint64_t` ns)
Wait busy for a small amount of time.
- `void l4_busy_wait_us` (`l4_uint64_t` us)
Wait busy for a small amount of time.
- `l4_uint32_t l4_calibrate_tsc` (`l4_kernel_info_t` const *kip)
Determine scalers for timestamp calculations.
- `l4_uint32_t l4_tsc_init` (`l4_kernel_info_t` const *kip)
Initialize scaler for TSC calibrations from the kernel.
- `l4_uint32_t l4_get_hz` (void)
Get CPU frequency in Hz.

16.83.1 Detailed Description

Timestamp counter related functions.

Author

Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file [rdtsc.h](#).

16.84 rdtsc.h

[Go to the documentation of this file.](#)

```

00001
00008
00009 /*
00010  * (c) 2003-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00011  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00012  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00013  *      economic rights: Technische Universität Dresden (Germany)
00014  * License: see LICENSE.spdx (in this directory or the directories above)
00015  */
00016
00017 #ifndef __l4_rdtsc_h
00018 #define __l4_rdtsc_h
00019
00024
00025 #include <l4/sys/compiler.h>
00026 #include <l4/sys/l4int.h>
00027 #include <l4/sys/kip.h>
00028
00029 L4_BEGIN_DECLS
00030
00031 /* interface */
00036
00037 extern l4_uint32_t l4_scaler_tsc_to_ns;
00038 extern l4_uint32_t l4_scaler_tsc_to_us;
00039 extern l4_uint32_t l4_scaler_ns_to_tsc;
00040 extern l4_uint32_t l4_scaler_tsc_linux;
00041
00046 L4_INLINE l4_cpu_time_t
00047 l4_rdtsc (void);
00048
00054 L4_INLINE
00055 l4_uint32_t l4_rdtsc_32 (void);
00056
00063 L4_INLINE l4_uint64_t
00064 l4_rdpmc (int ecx);
00065
00071 L4_INLINE
00072 l4_uint32_t l4_rdpmc_32 (int ecx);
00073
00078 L4_INLINE l4_uint64_t
00079 l4_tsc_to_ns (l4_cpu_time_t tsc);
00080
00085 L4_INLINE l4_uint64_t
00086 l4_tsc_to_us (l4_cpu_time_t tsc);
00087
00093 L4_INLINE void
00094 l4_tsc_to_s_and_ns (l4_cpu_time_t tsc, l4_uint32_t *s, l4_uint32_t *ns);
00095
00101 L4_INLINE l4_cpu_time_t
00102 l4_ns_to_tsc (l4_uint64_t ns);
00103
00109 L4_INLINE void
00110 l4_busy_wait_ns (l4_uint64_t ns);
00111
00117 L4_INLINE void
00118 l4_busy_wait_us (l4_uint64_t us);
00119
00126 L4_INLINE l4_uint32_t
00127 l4_calibrate_tsc (l4_kernel_info_t const *kip);
00128
00142 L4_CV l4_uint32_t
00143 l4_tsc_init (l4_kernel_info_t const *kip);

```

```

00144
00149 L4_CV l4_uint32_t
00150 l4_get_hz (void);
00151
00153
00154 L4_END_DECLS
00155
00156 /* implementation */
00157
00158 L4_INLINE l4_uint32_t
00159 l4_calibrate_tsc(l4_kernel_info_t const *kip)
00160 {
00161     return l4_tsc_init(kip);
00162 }
00163
00164 L4_INLINE l4_cpu_time_t
00165 l4_rdtsc (void)
00166 {
00167     l4_umword_t lo, hi;
00168
00169     __asm__ __volatile__ ("rdtsc" : "=a"(lo), "=d"(hi));
00170
00171     return ((l4_cpu_time_t)hi << 32) | lo;
00172 }
00173
00174 L4_INLINE l4_uint64_t
00175 l4_rdpmc (int ecx)
00176 {
00177     l4_umword_t lo, hi;
00178
00179     __asm__ __volatile__ ("rdpmc" : "=a"(lo), "=d"(hi) : "c"(ecx));
00180
00181     return ((l4_cpu_time_t)hi << 32) | lo;
00182 }
00183
00184 L4_INLINE
00185 l4_uint32_t l4_rdtsc_32(void)
00186 {
00187     l4_umword_t lo, hi;
00188
00189     __asm__ __volatile__ ("rdtsc" : "=a"(lo), "=d"(hi));
00190
00191     return lo;
00192 }
00193
00194 L4_INLINE
00195 l4_uint32_t l4_rdpmc_32(int ecx)
00196 {
00197     l4_umword_t lo, hi;
00198
00199     __asm__ __volatile__ ("rdpmc" : "=a"(lo), "=d"(hi) : "c"(ecx));
00200
00201     return lo;
00202 }
00203
00204 L4_INLINE l4_uint64_t
00205 l4_tsc_to_ns (l4_cpu_time_t tsc)
00206 {
00207     l4_uint64_t ns, dummy;
00208     __asm__
00209     ("
00210      \n\t"
00211      "mulq  %3      \n\t"
00212      "shrd  $27, %%rdx, %%rax      \n\t"
00213      : "=a" (ns), "=d" (dummy)
00214      : "a" (tsc), "r" ((l4_uint64_t)l4_scaler_tsc_to_ns)
00215      );
00216     return ns;
00217 }
00218 L4_INLINE l4_uint64_t
00219 l4_tsc_to_us (l4_cpu_time_t tsc)
00220 {
00221     l4_uint64_t ns, dummy;
00222     __asm__
00223     ("
00224      \n\t"
00225      "mulq  %3      \n\t"
00226      "shrd  $32, %%rdx, %%rax      \n\t"
00227      : "=a" (ns), "=d" (dummy)
00228      : "a" (tsc), "r" ((l4_uint64_t)l4_scaler_tsc_to_us)
00229      );
00230     return ns;
00231 }
00232 L4_INLINE void
00233 l4_tsc_to_s_and_ns (l4_cpu_time_t tsc, l4_uint32_t *s, l4_uint32_t *ns)
00234 {
00235     __asm__

```

```

00236     ("        \n\t"
00237     "mulq  %3        \n\t"
00238     "shrd  $27, %%rdx, %%rax        \n\t"
00239     "xorq  %%rdx, %%rdx        \n\t"
00240     "divq  %4        \n\t"
00241     : "=a" (*s), "=d" (*ns)
00242     : "a" (tsc), "r" ((l4_uint64_t)l4_scaler_tsc_to_ns),
00243     "rm"(1000000000ULL)
00244     );
00245 }
00246
00247 L4_INLINE l4_cpu_time_t
00248 l4_ns_to_tsc (l4_uint64_t ns)
00249 {
00250     l4_uint64_t tsc, dummy;
00251     __asm__
00252     (
00253     "mulq  %3        \n\t"
00254     "shrd  $27, %%rdx, %%rax        \n\t"
00255     : "=a" (tsc), "=d" (dummy)
00256     : "a" (ns), "r" ((l4_uint64_t)l4_scaler_ns_to_tsc)
00257     );
00258     return tsc;
00259 }
00260
00261 L4_INLINE void
00262 l4_busy_wait_ns (l4_uint64_t ns)
00263 {
00264     l4_cpu_time_t stop = l4_rdtsc();
00265     stop += l4_ns_to_tsc(ns);
00266
00267     while (l4_rdtsc() < stop)
00268         ;
00269 }
00270
00271 L4_INLINE void
00272 l4_busy_wait_us (l4_uint64_t us)
00273 {
00274     l4_cpu_time_t stop = l4_rdtsc ();
00275     stop += l4_ns_to_tsc(us*1000ULL);
00276
00277     while (l4_rdtsc() < stop)
00278         ;
00279 }
00280
00281 #endif /* __l4_rdtsc_h */
00282

```

16.85 x86/i4/util/rdtsc.h File Reference

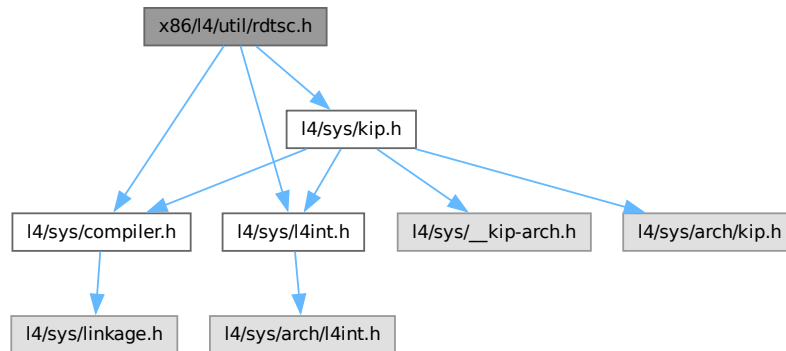
Timestamp counter related functions.

```

#include <l4/sys/compiler.h>
#include <l4/sys/l4int.h>
#include <l4/sys/kip.h>

```


Include dependency graph for rdtsc.h:



Functions

- `l4_cpu_time_t l4_rdtsc` (void)
Read current value of CPU-internal timestamp counter.
- `l4_uint32_t l4_rdtsc_32` (void)
Read the lest significant 32 bit of the TSC.
- `l4_uint64_t l4_rdpmc` (int ecx)
Return current value of CPU-internal performance measurement counter.
- `l4_uint32_t l4_rdpmc_32` (int ecx)
Return the least significant 32 bit of a performance counter.
- `l4_uint64_t l4_tsc_to_ns` (`l4_cpu_time_t` tsc)
Convert timestamp to ns value.
- `l4_uint64_t l4_tsc_to_us` (`l4_cpu_time_t` tsc)
Convert timestamp into micro seconds value.
- `void l4_tsc_to_s_and_ns` (`l4_cpu_time_t` tsc, `l4_uint32_t` *s, `l4_uint32_t` *ns)
Convert timestamp to s.ns value.
- `l4_cpu_time_t l4_ns_to_tsc` (`l4_uint64_t` ns)
Convert nano seconds into CPU ticks.
- `void l4_busy_wait_ns` (`l4_uint64_t` ns)
Wait busy for a small amount of time.
- `void l4_busy_wait_us` (`l4_uint64_t` us)
Wait busy for a small amount of time.
- `l4_uint32_t l4_calibrate_tsc` (`l4_kernel_info_t` const *kip)
Determine scalers for timestamp calculations.
- `l4_uint32_t l4_tsc_init` (`l4_kernel_info_t` const *kip)
Initialize scaler for TSC calibrations from the kernel.
- `l4_uint32_t l4_get_hz` (void)
Get CPU frequency in Hz.

16.85.1 Detailed Description

Timestamp counter related functions.

Author

Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file [rdtsc.h](#).

16.86 rdtsc.h

[Go to the documentation of this file.](#)

```

00001
00008
00009 /*
00010  * (c) 2003-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00011  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00012  *      Frank Mehnert <fm3@os.inf.tu-dresden.de>,
00013  *      Jork Löser <jork@os.inf.tu-dresden.de>,
00014  *      Martin Pohlack <mp26@os.inf.tu-dresden.de>
00015  *      economic rights: Technische Universität Dresden (Germany)
00016  * License: see LICENSE.spdx (in this directory or the directories above)
00017  */
00018
00019 #ifndef __l4_rdtsc_h
00020 #define __l4_rdtsc_h
00021
00026
00027 #include <l4/sys/compiler.h>
00028 #include <l4/sys/l4int.h>
00029 #include <l4/sys/kip.h>
00030
00031 L4_BEGIN_DECLS
00032
00033 /* interface */
00038
00039 extern l4_uint32_t l4_scaler_tsc_to_ns;
00040 extern l4_uint32_t l4_scaler_tsc_to_us;
00041 extern l4_uint32_t l4_scaler_ns_to_tsc;
00042 extern l4_uint32_t l4_scaler_tsc_linux;
00043
00048 L4_INLINE l4_cpu_time_t
00049 l4_rdtsc (void);
00050
00056 L4_INLINE
00057 l4_uint32_t l4_rdtsc_32 (void);
00058
00065 L4_INLINE l4_uint64_t
00066 l4_rdpmc (int ecx);
00067
00073 L4_INLINE
00074 l4_uint32_t l4_rdpmc_32 (int ecx);
00075
00080 L4_INLINE l4_uint64_t
00081 l4_tsc_to_ns (l4_cpu_time_t tsc);
00082
00087 L4_INLINE l4_uint64_t
00088 l4_tsc_to_us (l4_cpu_time_t tsc);
00089
00095 L4_INLINE void
00096 l4_tsc_to_s_and_ns (l4_cpu_time_t tsc, l4_uint32_t *s, l4_uint32_t *ns);
00097
00103 L4_INLINE l4_cpu_time_t
00104 l4_ns_to_tsc (l4_uint64_t ns);
00105
00111 L4_INLINE void
00112 l4_busy_wait_ns (l4_uint64_t ns);
00113
00119 L4_INLINE void
00120 l4_busy_wait_us (l4_uint64_t us);
00121
00128 L4_INLINE l4_uint32_t
00129 l4_calibrate_tsc (l4_kernel_info_t const *kip);
00130

```

```

00144 L4_CV l4_uint32_t
00145 l4_tsc_init(l4_kernel_info_t const *kip);
00146
00151 L4_CV l4_uint32_t
00152 l4_get_hz (void);
00153
00155
00156 L4_END_DECLS
00157
00158 /* implementation */
00159
00160 L4_INLINE l4_uint32_t
00161 l4_calibrate_tsc(l4_kernel_info_t const *kip)
00162 {
00163     return l4_tsc_init(kip);
00164 }
00165
00166 L4_INLINE l4_cpu_time_t
00167 l4_rdtsc (void)
00168 {
00169     l4_cpu_time_t v;
00170
00171     __asm__ __volatile__ (
00172         ("    \n\t"
00173          ".byte 0x0f, 0x31    \n\t"
00174          /*"rdtsc\n\t"*/
00175          :
00176          "=A" (v)
00177          : /* no inputs */
00178          );
00179
00180     return v;
00181 }
00182
00183 /* the same, but only 32 bit. Useful for smaller differences,
00184    needs less cycles. */
00185 L4_INLINE
00186 l4_uint32_t l4_rdtsc_32(void)
00187 {
00188     l4_uint32_t x;
00189
00190     __asm__ __volatile__ (
00191         ".byte 0x0f, 0x31\n\t" // rdtsc
00192         : "=a" (x)
00193         :
00194         : "edx");
00195
00196     return x;
00197 }
00198
00199 L4_INLINE l4_uint64_t
00200 l4_rdpmc (int ecx)
00201 {
00202     l4_cpu_time_t v;
00203
00204     __asm__ __volatile__ (
00205         "rdpmc    \n\t"
00206         :
00207         "=A" (v)
00208         : "c" (ecx)
00209         );
00210
00211     return v;
00212 }
00213
00214 /* the same, but only 32 bit. Useful for smaller differences,
00215    needs less cycles. */
00216 L4_INLINE
00217 l4_uint32_t l4_rdpmc_32(int ecx)
00218 {
00219     l4_uint32_t x;
00220
00221     __asm__ __volatile__ (
00222         "rdpmc    \n\t"
00223         : "=a" (x)
00224         : "c" (ecx)
00225         : "edx");
00226
00227     return x;
00228 }
00229
00230 L4_INLINE l4_uint64_t
00231 l4_tsc_to_ns (l4_cpu_time_t tsc)
00232 {
00233     l4_uint32_t dummy;
00234     l4_uint64_t ns;
00235     __asm__

```

```

00236 ("      \n\t"
00237 "movl %%edx, %%ecx \n\t"
00238 "mull %3 \n\t"
00239 "movl %%ecx, %%eax \n\t"
00240 "movl %%edx, %%ecx \n\t"
00241 "mull %3 \n\t"
00242 "addl %%ecx, %%eax \n\t"
00243 "adcl $0, %%edx \n\t"
00244 "shld $5, %%eax, %%edx \n\t"
00245 "shll $5, %%eax \n\t"
00246 : "=A" (ns),
00247 "=&c" (dummy)
00248 : "0" (tsc),
00249 "g" (l4_scaler_tsc_to_ns)
00250 );
00251 return ns;
00252 }
00253
00254 L4_INLINE l4_uint64_t
00255 l4_tsc_to_us (l4_cpu_time_t tsc)
00256 {
00257     l4_uint32_t dummy;
00258     l4_uint64_t us;
00259     __asm__
00260 ("      \n\t"
00261 "movl %%edx, %%ecx \n\t"
00262 "mull %3 \n\t"
00263 "movl %%ecx, %%eax \n\t"
00264 "movl %%edx, %%ecx \n\t"
00265 "mull %3 \n\t"
00266 "addl %%ecx, %%eax \n\t"
00267 "adcl $0, %%edx \n\t"
00268 : "=A" (us),
00269 "=&c" (dummy)
00270 : "0" (tsc),
00271 "g" (l4_scaler_tsc_to_us)
00272 );
00273 return us;
00274 }
00275
00276 L4_INLINE void
00277 l4_tsc_to_s_and_ns (l4_cpu_time_t tsc, l4_uint32_t *s, l4_uint32_t *ns)
00278 {
00279     l4_uint32_t dummy;
00280     __asm__
00281 ("      \n\t"
00282 "movl %%edx, %%ecx \n\t"
00283 "mull %4 \n\t"
00284 "movl %%ecx, %%eax \n\t"
00285 "movl %%edx, %%ecx \n\t"
00286 "mull %4 \n\t"
00287 "addl %%ecx, %%eax \n\t"
00288 "adcl $0, %%edx \n\t"
00289 "movl $1000000000, %%ecx \n\t"
00290 "shld $5, %%eax, %%edx \n\t"
00291 "shll $5, %%eax \n\t"
00292 "divl %%ecx \n\t"
00293 : "=a" (*s), "=d" (*ns), "=&c" (dummy)
00294 : "A" (tsc), "g" (l4_scaler_tsc_to_ns)
00295 );
00296 }
00297
00298 L4_INLINE l4_cpu_time_t
00299 l4_ns_to_tsc (l4_uint64_t ns)
00300 {
00301     l4_uint32_t dummy;
00302     l4_cpu_time_t tsc;
00303     __asm__
00304 ("      \n\t"
00305 "movl %%edx, %%ecx \n\t"
00306 "mull %3 \n\t"
00307 "movl %%ecx, %%eax \n\t"
00308 "movl %%edx, %%ecx \n\t"
00309 "mull %3 \n\t"
00310 "addl %%ecx, %%eax \n\t"
00311 "adcl $0, %%edx \n\t"
00312 "shld $5, %%eax, %%edx \n\t"
00313 "shll $5, %%eax \n\t"
00314 : "=A" (tsc),
00315 "=&c" (dummy)
00316 : "0" (ns),
00317 "g" (l4_scaler_ns_to_tsc)
00318 );
00319 return tsc;
00320 }
00321
00322 L4_INLINE void

```

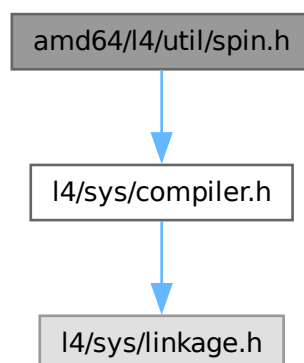
```
00323 l4_busy_wait_ns (l4_uint64_t ns)
00324 {
00325     l4_cpu_time_t stop = l4_rdtsc();
00326     stop += l4_ns_to_tsc(ns);
00327
00328     while (l4_rdtsc() < stop)
00329         ;
00330 }
00331
00332 L4_INLINE void
00333 l4_busy_wait_us (l4_uint64_t us)
00334 {
00335     l4_cpu_time_t stop = l4_rdtsc ();
00336     stop += l4_ns_to_tsc(us*1000ULL);
00337
00338     while (l4_rdtsc() < stop)
00339         ;
00340 }
00341
00342 #endif /* __l4_rdtsc_h */
00343
```

16.87 amd64/l4/util/spin.h File Reference

Spinning for amd64.

```
#include <l4/sys/compiler.h>
```

Include dependency graph for spin.h:



16.87.1 Detailed Description

Spinning for amd64.

Definition in file [spin.h](#).

16.88 spin.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  * Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00008  * economic rights: Technische Universität Dresden (Germany)
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011 #ifndef __l4util_spin_h
00012 #define __l4util_spin_h
00013
00014 #include <l4/sys/compiler.h>
00015
00016 L4_BEGIN_DECLS
00017
00018 L4_CV void l4_spin(int x,int y);
00019 L4_CV void l4_spin_vga(int x,int y);
00020 L4_CV void l4_spin_n_text(int x, int y, int len, const char*s);
00021 L4_CV void l4_spin_n_text_vga(int x, int y, int len, const char*s);
00022
00023 /*****
00024  *
00025  * spin_text() - spinning wheel at the hercules screen. The given text
00026  * must be a text constant, no variables or arrays. Its
00027  * size is determined with the sizeof operator, it's much
00028  * faster than the strlen function.
00029  * spin_text_vga() - same for vga.
00030  *
00031  *****/
00032 #define l4_spin_text(x, y, text) \
00033     l4_spin_n_text((x), (y), sizeof(text)-1, "" text)
00034 #define l4_spin_text_vga(x, y, text) \
00035     l4_spin_n_text_vga((x), (y), sizeof(text)-1, "" text)
00036
00037 L4_END_DECLS
00038
00039 #endif

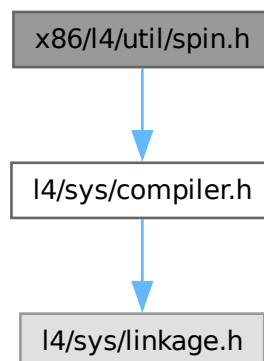
```

16.89 x86/l4/util/spin.h File Reference

Spinning for x86.

```
#include <l4/sys/compiler.h>
```

Include dependency graph for spin.h:



16.89.1 Detailed Description

Spinning for x86.

Definition in file [spin.h](#).

16.90 spin.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Frank Mehnert <fm3@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011 #ifndef __l4util_spin_h
00012 #define __l4util_spin_h
00013
00014 #include <l4/sys/compiler.h>
00015
00016 L4_BEGIN_DECLS
00017
00018 L4_CV void l4_spin(int x,int y);
00019 L4_CV void l4_spin_vga(int x,int y);
00020 L4_CV void l4_spin_n_text(int x, int y, int len, const char*s);
00021 L4_CV void l4_spin_n_text_vga(int x, int y, int len, const char*s);
00022
00023 /*****
00024  *
00025  * spin_text()      - spinning wheel at the hercules screen. The given text
00026  *                   must be a text constant, no variables or arrays. Its
00027  *                   size is determined with the sizeof operator, it's much
00028  *                   faster than the strlen function.
00029  * spin_text_vga() - same for vga.
00030  *
00031  *****/
00032 #define l4_spin_text(x, y, text) \
00033     l4_spin_n_text((x), (y), sizeof(text)-1, "" text)
00034 #define l4_spin_text_vga(x, y, text) \
00035     l4_spin_n_text_vga((x), (y), sizeof(text)-1, "" text)
00036
00037 L4_END_DECLS
00038
00039 #endif

```

16.91 __kip-arch.h

```

00001 /*
00002  * (c) 2013 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 struct l4_kip_platform_info_arch
00010 {};

```

16.92 __kip-arch.h

```

00001 /*
00002  * (c) 2013 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008

```

```

00014 struct l4_kip_platform_info_arch
00015 {
00016     struct
00017     {
00018         l4_uint32_t MIDR, CTR, TCMTR, TLBTR, MPIDR, REVIDR;
00019         l4_uint32_t ID_PFR[2], ID_DFR0, ID_AFR0, ID_MMFR[4], ID_ISAR[6];
00020     } cpuinfo;
00021 };

```

16.93 __kip-arch.h

```

00001 /*
00002  * (c) 2013 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *     economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00014 struct l4_kip_platform_info_arch
00015 {
00016     struct
00017     {
00018         l4_uint64_t MIDR, MPIDR, REVIDR;
00019         l4_uint64_t ID_PFR[3], ID_DFR0, ID_AFR0, ID_MMFR[4], ID_ISAR[7], ID_MVFR[3];
00020         l4_uint64_t ID_AA64DFR[2], ID_AA64ISAR[3], ID_AA64MMFR[3], ID_AA64PFR[2];
00021     } cpuinfo;
00022 };

```

16.94 __kip-arch.h

```

00001 /*
00002  * Copyright (C) 2021, 2024 Kernkonzept GmbH.
00003  * Author(s): Georg Kotheimer <georg.kotheimer@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 struct l4_kip_platform_info_arch
00010 {
00011     l4_uint32_t isa_ext[7];
00012     l4_uint32_t timebase_frequency;
00013     l4_uint32_t hart_num;
00014     l4_uint32_t hart_ids[16];
00015     l4_umword_t plic_addr;
00016     l4_uint32_t plic_nr_irqs;
00017     l4_uint32_t plic_hart_irq_targets[16];
00018 };
00019
00020 typedef enum L4_riscv_isa_ext
00021 {
00022     // IDs 0-25 are reserved for single-letter RISC-V ISA extensions.
00023     L4_riscv_isa_ext_a = ('a' - 'a'), // Atomics
00024     L4_riscv_isa_ext_b = ('b' - 'a'), // Bit Manipulation
00025     L4_riscv_isa_ext_c = ('c' - 'a'), // Quad-Precision Floating-Point
00026     L4_riscv_isa_ext_d = ('d' - 'a'), // Double-Precision Floating-Point
00027     L4_riscv_isa_ext_e = ('e' - 'a'), // Reduced Integer
00028     L4_riscv_isa_ext_f = ('f' - 'a'), // Single-Precision Floating-Point
00029     L4_riscv_isa_ext_g = ('g' - 'a'), // General
00030     L4_riscv_isa_ext_h = ('h' - 'a'), // Hypervisor
00031     L4_riscv_isa_ext_i = ('i' - 'a'), // Integer
00032     L4_riscv_isa_ext_m = ('m' - 'a'), // Integer Multiplication and Division
00033     L4_riscv_isa_ext_p = ('p' - 'a'), // Packed-SIMD Extensions
00034     L4_riscv_isa_ext_q = ('q' - 'a'), // Quad-Precision Floating-Point
00035     L4_riscv_isa_ext_v = ('v' - 'a'), // Vector
00036
00037     // IDs starting from 26 represent multi-letter extensions. The assignment does
00038     // not follow a defined order, IDs are simply assigned incrementally when new
00039     // extensions are added.
00040     L4_riscv_isa_ext_base = 26,
00041
00042     L4_riscv_isa_ext_sstc = 27, // stimecmp / vstimecmp
00043
00044     // Maximum number of extensions that can be represented, corresponds to the
00045     // size of the `l4_kip_platform_info_arch.isa_ext` bitmap.
00046     L4_riscv_isa_ext_max = 224,
00047 } L4_riscv_isa_ext;

```


16.95 __kip-arch.h

```

00001 /*
00002  * (c) 2013 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *     economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 struct l4_kip_platform_info_arch
00010 {};

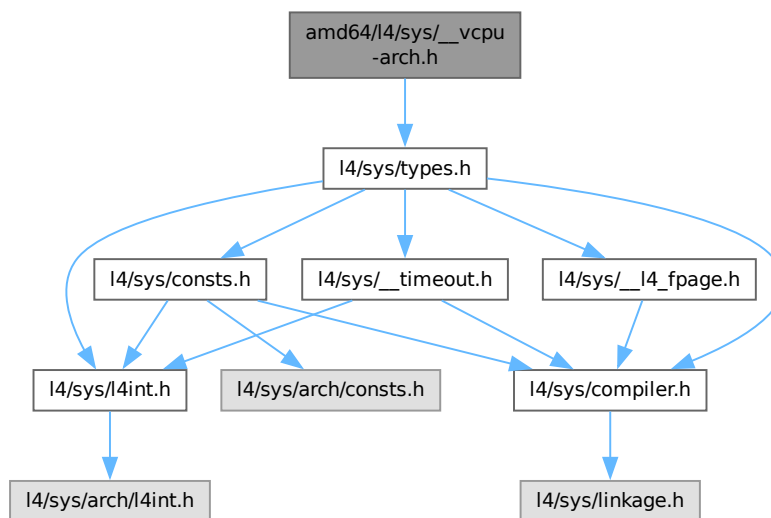
```

16.96 amd64/l4/sys/__vcpu-arch.h File Reference

AMD64-specific vCPU interface.

```
#include <l4/sys/types.h>
```

Include dependency graph for __vcpu-arch.h:



Data Structures

- struct `l4_vcpu_arch_state_t`
Architecture-specific vCPU state.
- struct `l4_vcpu_regs_t`
vCPU registers.
- struct `l4_vcpu_ipc_regs_t`
vCPU message registers.

Typedefs

- typedef struct `l4_vcpu_arch_state_t` `l4_vcpu_arch_state_t`
Architecture-specific vCPU state.
- typedef struct `l4_vcpu_regs_t` `l4_vcpu_regs_t`
vCPU registers.
- typedef struct `l4_vcpu_ipc_regs_t` `l4_vcpu_ipc_regs_t`
vCPU message registers.

Enumerations

- enum { [L4_VCPU_STATE_VERSION](#) = 0x26 , [L4_VCPU_STATE_SIZE](#) = 0x200 , [L4_VCPU_STATE_EXT_SIZE](#) = [L4_PAGESIZE](#) }
- enum [L4_vcpu_state_offset](#) { [L4_VCPU_OFFSET_EXT_STATE](#) = 0x400 , [L4_VCPU_OFFSET_EXT_INFOS](#) = 0x200 }

Offsets for vCPU state layouts.

16.96.1 Detailed Description

AMD64-specific vCPU interface.

Definition in file [__vcpu-arch.h](#).

16.96.2 Enumeration Type Documentation

16.96.2.1 anonymous enum

anonymous enum

Enumerator

| | |
|---------------------------------------|--|
| L4_VCPU_STATE_VERSION | Architecture-specific version ID. This ID must match the version field in the l4_vcpu_state_t structure after enabling vCPU mode or extended vCPU mode for a thread. |
|---------------------------------------|--|

Definition at line 16 of file [__vcpu-arch.h](#).

16.97 __vcpu-arch.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00012 #pragma once
00013
00014 #include <linux/types.h>
00015
00016 enum
00017 {
00024     L4_VCPU_STATE_VERSION = 0x26,
00025
00026     L4_VCPU_STATE_SIZE = 0x200,
00027     L4_VCPU_STATE_EXT_SIZE = L4_PAGESIZE,
00028 };
00029
00034 enum L4_vcpu_state_offset
00035 {
00036     L4_VCPU_OFFSET_EXT_STATE = 0x400,
00037     L4_VCPU_OFFSET_EXT_INFOS = 0x200,
00038 };
00039
00043 typedef struct l4_vcpu_arch_state_t
00044 {

```

```

00045  l4_umword_t host_fs_base;
00046  l4_umword_t host_gs_base;
00047  l4_uint16_t host_ds, host_es, host_fs, host_gs;
00048
00049  l4_uint16_t const user_ds32;
00050  l4_uint16_t const user_cs64;
00051  l4_uint16_t const user_cs32;
00052 } l4_vcpu_arch_state_t;
00053
00054
00055 typedef struct l4_vcpu_regs_t
00056 {
00057     l4_umword_t r15;
00058     l4_umword_t r14;
00059     l4_umword_t r13;
00060     l4_umword_t r12;
00061     l4_umword_t r11;
00062     l4_umword_t r10;
00063     l4_umword_t r9;
00064     l4_umword_t r8;
00065
00066     l4_umword_t di;
00067     l4_umword_t si;
00068     l4_umword_t bp;
00069     l4_umword_t pfa;
00070     l4_umword_t bx;
00071     l4_umword_t dx;
00072     l4_umword_t cx;
00073     l4_umword_t ax;
00074
00075     l4_umword_t trapno;
00076     l4_umword_t err;
00077
00078     l4_umword_t ip;
00079     l4_umword_t cs;
00080     l4_umword_t flags;
00081     l4_umword_t sp;
00082     l4_umword_t ss;
00083     l4_umword_t fs_base;
00084     l4_umword_t gs_base;
00085     l4_uint16_t ds, es, fs, gs;
00086
00087 } l4_vcpu_regs_t;
00088
00089 typedef struct l4_vcpu_ipc_regs_t
00090 {
00091     l4_umword_t _res[1];
00092     l4_umword_t label;
00093     l4_umword_t _res2[5];
00094     l4_msgtag_t tag;
00095 } l4_vcpu_ipc_regs_t;

```

16.98 arm/l4/sys/__vcpu-arch.h File Reference

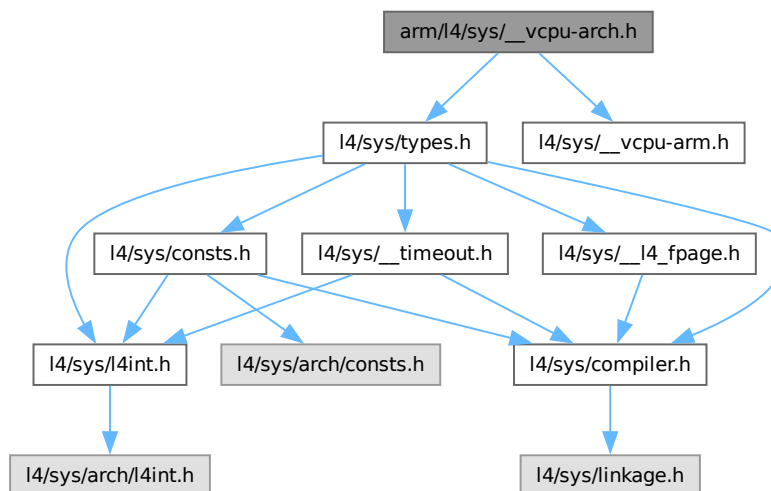
ARM-specific vCPU interface.

```

#include <l4/sys/types.h>
#include <l4/sys/__vcpu-arm.h>

```

Include dependency graph for `__vcpu-arch.h`:



Data Structures

- struct `l4_vcpu_regs_t`
vCPU registers.
- struct `l4_vcpu_arch_state_t`
Architecture-specific vCPU state.
- struct `l4_vcpu_ipc_regs_t`
vCPU message registers.

Typedefs

- typedef struct `l4_vcpu_regs_t` **`l4_vcpu_regs_t`**
vCPU registers.
- typedef struct `l4_vcpu_arch_state_t` **`l4_vcpu_arch_state_t`**
Architecture-specific vCPU state.
- typedef struct `l4_vcpu_ipc_regs_t` **`l4_vcpu_ipc_regs_t`**
vCPU message registers.

Enumerations

- enum { `L4_VCPU_STATE_VERSION` = 0x38 , `L4_VCPU_STATE_SIZE` = 0x100 , `L4_VCPU_STATE_EXT` ↔ `_SIZE` = 0x400 }
- enum `L4_vcpu_state_offset` { `L4_VCPU_OFFSET_EXT_STATE` = 0x180 , `L4_VCPU_OFFSET_EXT_INFOS` = 0x100 }
- enum `L4_vcpu_e_field_ids` { }
- *Offsets for vCPU state layouts.*
- *IDs for extended vCPU state fields.*

16.98.1 Detailed Description

ARM-specific vCPU interface.

Definition in file [__vcpu-arch.h](#).

16.98.2 Enumeration Type Documentation

16.98.2.1 anonymous enum

anonymous enum

Enumerator

| | |
|-----------------------|--|
| L4_VCPU_STATE_VERSION | Architecture-specific version ID. This ID must match the version field in the l4_vcpu_state_t structure after enabling vCPU mode or extended vCPU mode for a thread. |
|-----------------------|--|

Definition at line 17 of file [__vcpu-arch.h](#).

16.98.2.2 L4_vcpu_e_field_ids

enum [L4_vcpu_e_field_ids](#)

IDs for extended vCPU state fields.

Bits 14..15: are the field size:

- 0 = 32bit field
- 1 = register width field
- 2 = 64bit field

Enumerator

| | |
|--------------------|--------------------------|
| L4_VCPU_E_VTMR_CFG | vtimer irq configuration |
|--------------------|--------------------------|

Definition at line 99 of file [__vcpu-arch.h](#).

16.99 __vcpu-arch.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00012 #pragma once
00013
00014 #include <l4/sys/types.h>
00015 #include <l4/sys/__vcpu-arm.h>
00016
00017 enum
00018 {
00025     L4_VCPU_STATE_VERSION = 0x38,
00026
00027     L4_VCPU_STATE_SIZE = 0x100,
00028     L4_VCPU_STATE_EXT_SIZE = 0x400,
00029 };
00030
00035 enum L4_vcpu_state_offset
00036 {
00037     L4_VCPU_OFFSET_EXT_STATE = 0x180,
00038     L4_VCPU_OFFSET_EXT_INFOS = 0x100,
00039 };
00040
00041 L4_INLINE l4_arm_vcpu_e_info_t const *
00042 l4_vcpu_e_info(void const *vcpu) L4_NOTHROW
00043 {
00044     return (l4_arm_vcpu_e_info_t const *)((l4_addr_t)vcpu
00045                                           + L4_VCPU_OFFSET_EXT_INFOS);
00046 }
00047
00048 L4_INLINE void *l4_vcpu_e_ptr(void const *vcpu, unsigned id) L4_NOTHROW
00049 { return (void *)((l4_addr_t)vcpu + L4_VCPU_OFFSET_EXT_STATE + (id & 0xfff)); }
00050
00055 typedef struct l4_vcpu_regs_t
00056 {
00057     l4_umword_t pfa;
00058     l4_umword_t err;
00059
00060     l4_umword_t r[13];
00061
00062     l4_umword_t sp;
00063     l4_umword_t lr;
00064     l4_umword_t _dummy;
00065     l4_umword_t ip;
00066     l4_umword_t flags;
00067     l4_umword_t tpidruro;
00068     l4_umword_t tpidrurw;
00069 } l4_vcpu_regs_t;
00070
00074 typedef struct l4_vcpu_arch_state_t
00075 {
00076     l4_umword_t host_tpidruro;
00077 } l4_vcpu_arch_state_t;
00078
00083 typedef struct l4_vcpu_ipc_regs_t
00084 {
00085     l4_msgtag_t tag;
00086     l4_umword_t _d1[3];
00087     l4_umword_t label;
00088     l4_umword_t _d2[8];
00089 } l4_vcpu_ipc_regs_t;
00090
00099 enum L4_vcpu_e_field_ids
00100 {
00101     L4_VCPU_E_HCR           = 0x8008,
00102     L4_VCPU_E_TTBRO        = 0x8010,
00103     L4_VCPU_E_TTBRI        = 0x8018,
00104     L4_VCPU_E_TTBRCR       = 0x0020,
00105     L4_VCPU_E_SCTLR        = 0x0024,
00106     L4_VCPU_E_DACR         = 0x0028,
00107     L4_VCPU_E_FCSEIDR      = 0x002c,
00108
00109     L4_VCPU_E_CNTVCTL       = 0x0030,
00110     L4_VCPU_E_CNTVOFF      = 0x8038,
00111
00112     L4_VCPU_E_VMPIDR       = 0x0040,
00113     L4_VCPU_E_VPIDR        = 0x0044,
00114
00115     L4_VCPU_E_VTMR_CFG     = 0x0048,

```

```

00116
00117     L4_VCPU_E_GIC_HCR      = 0x0050,
00118     L4_VCPU_E_GIC_VTR      = 0x0054,
00119     L4_VCPU_E_GIC_VMCR      = 0x0058,
00120     L4_VCPU_E_GIC_MISR      = 0x005c,
00121     L4_VCPU_E_GIC_EISR      = 0x0060,
00122     L4_VCPU_E_GIC_ELSR      = 0x0064,
00123     L4_VCPU_E_GIC_V2_LR0    = 0x0068,
00124     L4_VCPU_E_GIC_V3_LR0    = 0x0068,
00125 };

```

16.100 arm64/l4/sys/__vcpu-arch.h File Reference

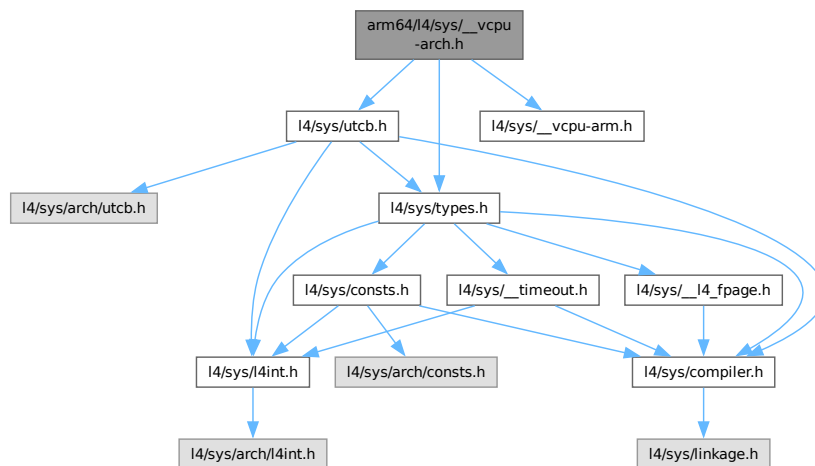
ARM64-specific vCPU interface.

```

#include <l4/sys/types.h>
#include <l4/sys/utcb.h>
#include <l4/sys/__vcpu-arm.h>

```

Include dependency graph for __vcpu-arch.h:



Data Structures

- struct [l4_vcpu_arch_state_t](#)
Architecture-specific vCPU state.
- struct [l4_vcpu_ipc_regs_t](#)
vCPU message registers.

Typedefs

- typedef [l4_exc_regs_t](#) [l4_vcpu_regs_t](#)
vCPU registers.
- typedef struct [l4_vcpu_arch_state_t](#) [l4_vcpu_arch_state_t](#)
Architecture-specific vCPU state.
- typedef struct [l4_vcpu_ipc_regs_t](#) [l4_vcpu_ipc_regs_t](#)
vCPU message registers.

Enumerations

- enum { [L4_VCPU_STATE_VERSION](#) = 0x38 , [L4_VCPU_STATE_SIZE](#) = 0x200 , [L4_VCPU_STATE_EXT_SIZE](#) = 0x800 }
- enum [L4_vcpu_state_offset](#) { [L4_VCPU_OFFSET_EXT_STATE](#) = 0x280 , [L4_VCPU_OFFSET_EXT_INFOS](#) = 0x200 }

Offsets for vCPU state layouts.

- enum [L4_vcpu_e_field_ids](#) { }

IDs for extended vCPU state fields.

16.100.1 Detailed Description

ARM64-specific vCPU interface.

Definition in file [__vcpu-arch.h](#).

16.100.2 Enumeration Type Documentation

16.100.2.1 anonymous enum

anonymous enum

Enumerator

| | |
|---------------------------------------|--|
| L4_VCPU_STATE_VERSION | Architecture-specific version ID. This ID must match the version field in the l4_vcpu_state_t structure after enabling vCPU mode or extended vCPU mode for a thread. |
|---------------------------------------|--|

Definition at line 18 of file [__vcpu-arch.h](#).

16.100.2.2 [L4_vcpu_e_field_ids](#)

enum [L4_vcpu_e_field_ids](#)

IDs for extended vCPU state fields.

Bits 14..15: are the field size:

- 0 = 32bit field
- 1 = register width field
- 2 = 64bit field

Enumerator

| | |
|------------------------------------|--------------------------|
| L4_VCPU_E_VTMR_CFG | vtimer irq configuration |
|------------------------------------|--------------------------|

Definition at line 85 of file [__vcpu-arch.h](#).

16.101 __vcpu-arch.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00012 #pragma once
00013
00014 #include <l4/sys/types.h>
00015 #include <l4/sys/utcb.h>
00016 #include <l4/sys/__vcpu-arm.h>
00017
00018 enum
00019 {
00026     L4_VCPU_STATE_VERSION = 0x38,
00027
00028     L4_VCPU_STATE_SIZE = 0x200,
00029     L4_VCPU_STATE_EXT_SIZE = 0x800,
00030 };
00031
00036 enum L4_vcpu_state_offset
00037 {
00038     L4_VCPU_OFFSET_EXT_STATE = 0x280,
00039     L4_VCPU_OFFSET_EXT_INFOS = 0x200,
00040 };
00041
00042 L4_INLINE l4_arm_vcpu_e_info_t const *
00043 l4_vcpu_e_info(void const *vcpu) L4_NOTHROW
00044 {
00045     return (l4_arm_vcpu_e_info_t const *)((l4_addr_t)vcpu
00046                                           + L4_VCPU_OFFSET_EXT_INFOS);
00047 }
00048
00049 L4_INLINE void *l4_vcpu_e_ptr(void const *vcpu, unsigned id) L4_NOTHROW
00050 { return (void *)((l4_addr_t)vcpu + L4_VCPU_OFFSET_EXT_STATE + (id & 0xfff)); }
00051
00056 typedef l4_exc_regs_t l4_vcpu_regs_t;
00057
00061 typedef struct l4_vcpu_arch_state_t
00062 {
00063     l4_umword_t host_tpidruro;
00064 } l4_vcpu_arch_state_t;
00065
00070 typedef struct l4_vcpu_ipc_regs_t
00071 {
00072     l4_msgtag_t tag;
00073     l4_umword_t label;
00074     l4_umword_t _d1[3];
00075 } l4_vcpu_ipc_regs_t;
00076
00085 enum L4_vcpu_e_field_ids
00086 {
00087     L4_VCPU_E_HCR          = 0x8008,
00088     L4_VCPU_E_SCTLR       = 0x0010,
00089     L4_VCPU_E_CPACR       = 0x0014,
00090
00091     L4_VCPU_E_CNTVCTL      = 0x0018,
00092     L4_VCPU_E_CNTVOFF      = 0x8020,
00093
00094     L4_VCPU_E_VMPIDR       = 0x8028,
00095     L4_VCPU_E_VPIDR        = 0x0030,
00096
00097     L4_VCPU_E_VTMR_CFG     = 0x0034,
00098     L4_VCPU_E_VTCR         = 0x8038,
00099
00100     L4_VCPU_E_GIC_HCR      = 0x0040,
00101     L4_VCPU_E_GIC_VTR      = 0x0044,
00102     L4_VCPU_E_GIC_VMCR     = 0x0048,
00103     L4_VCPU_E_GIC_MISR     = 0x004c,
00104     L4_VCPU_E_GIC_EISR     = 0x0050,
00105     L4_VCPU_E_GIC_ELSR     = 0x0054,
00106     L4_VCPU_E_GIC_V2_LR0   = 0x0058,
00107     L4_VCPU_E_GIC_V3_LR0   = 0x8058,
00108 };

```

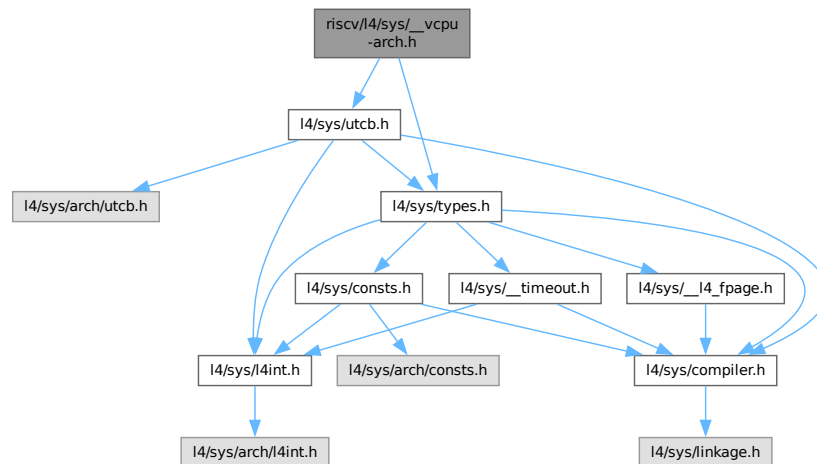
16.102 riscv/l4/sys/__vcpu-arch.h File Reference

RISC-V-specific vCPU interface.

```
#include <l4/sys/types.h>
```

```
#include <l4/sys/utcb.h>
```

Include dependency graph for __vcpu-arch.h:



Data Structures

- struct [l4_vcpu_arch_state_t](#)
Architecture-specific vCPU state.
- struct [l4_vcpu_ipc_regs_t](#)
vCPU message registers.

Typedefs

- typedef [l4_exc_regs_t](#) [l4_vcpu_regs_t](#)
vCPU registers.
- typedef struct [l4_vcpu_arch_state_t](#) [l4_vcpu_arch_state_t](#)
Architecture-specific vCPU state.
- typedef struct [l4_vcpu_ipc_regs_t](#) [l4_vcpu_ipc_regs_t](#)
vCPU message registers.

Enumerations

- enum { [L4_VCPU_STATE_VERSION](#) = 0x1 , [L4_VCPU_STATE_SIZE](#) = 0x200 , [L4_VCPU_STATE_EXT_SIZE](#) = [L4_PAGESIZE](#) }
 - enum [l4_vcpu_state_offset](#) { [L4_VCPU_OFFSET_EXT_STATE](#) = 0x400 , [L4_VCPU_OFFSET_EXT_INFOS](#) = 0x200 }
- Offsets for vCPU state layouts.*

16.102.1 Detailed Description

RISC-V-specific vCPU interface.

Definition in file [__vcpu-arch.h](#).

16.102.2 Enumeration Type Documentation

16.102.2.1 anonymous enum

anonymous enum

Enumerator

| | |
|-----------------------|--|
| L4_VCPU_STATE_VERSION | Architecture-specific version ID. This ID must match the version field in the l4_vcpu_state_t structure after enabling vCPU mode or extended vCPU mode for a thread. |
|-----------------------|--|

Definition at line 16 of file [__vcpu-arch.h](#).

16.103 __vcpu-arch.h

[Go to the documentation of this file.](#)

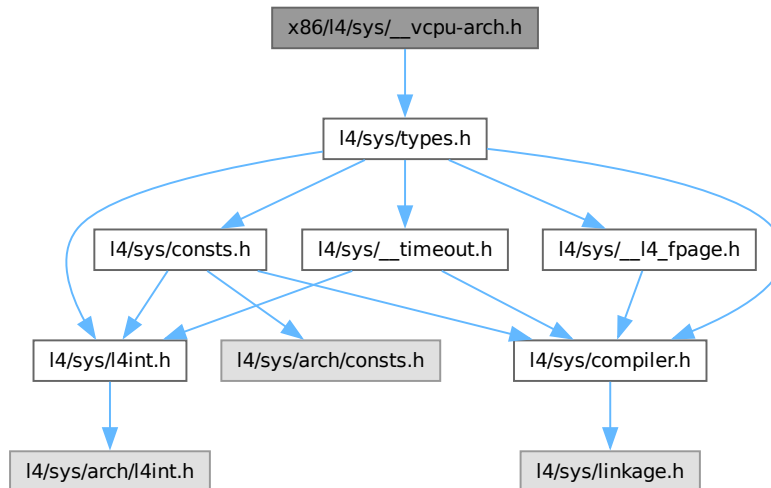
```
00001 /*
00002  * Copyright (C) 2021, 2024 Kernkonzept GmbH.
00003  * Author(s): Georg Kotheimer <georg.kotheimer@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00011 #pragma once
00012
00013 #include <l4/sys/types.h>
00014 #include <l4/sys/utcb.h>
00015
00016 enum
00017 {
00024     L4_VCPU_STATE_VERSION = 0x1,
00025
00026     L4_VCPU_STATE_SIZE = 0x200,
00027     L4_VCPU_STATE_EXT_SIZE = L4_PAGESIZE,
00028 };
00029
00034 enum l4_vcpu_state_offset
00035 {
00036     L4_VCPU_OFFSET_EXT_STATE = 0x400,
00037     L4_VCPU_OFFSET_EXT_INFOS = 0x200,
00038 };
00039
00044 typedef l4_exc_regs_t l4_vcpu_regs_t;
00045
00049 typedef struct l4_vcpu_arch_state_t
00050 {
00051     l4_umword_t host_tp;
00052     l4_umword_t host_gp;
00053 } l4_vcpu_arch_state_t;
00054
00059 typedef struct l4_vcpu_ipc_regs_t
00060 {
00061     l4_msgtag_t tag;        /* a0 */
00062     l4_umword_t dest;      /* a1 */
00063     l4_umword_t timeout;   /* a2 */
00064     l4_umword_t label;     /* a3 */
00065 } l4_vcpu_ipc_regs_t;
```

16.104 x86/l4/sys/__vcpu-arch.h File Reference

x86-specific vCPU interface.

```
#include <l4/sys/types.h>
```

Include dependency graph for __vcpu-arch.h:



Data Structures

- struct `l4_vcpu_regs_t`
vCPU registers.
- struct `l4_vcpu_arch_state_t`
Architecture-specific vCPU state.
- struct `l4_vcpu_ipc_regs_t`
vCPU message registers.

Typedefs

- typedef struct `l4_vcpu_regs_t` `l4_vcpu_regs_t`
vCPU registers.
- typedef struct `l4_vcpu_arch_state_t` `l4_vcpu_arch_state_t`
Architecture-specific vCPU state.
- typedef struct `l4_vcpu_ipc_regs_t` `l4_vcpu_ipc_regs_t`
vCPU message registers.

Enumerations

- enum { `L4_VCPU_STATE_VERSION` = 0x46 , `L4_VCPU_STATE_SIZE` = 0x200 , `L4_VCPU_STATE_EXT_SIZE` = `L4_PAGESIZE` }
 - enum `l4_vcpu_state_offset` { `L4_VCPU_OFFSET_EXT_STATE` = 0x400 , `L4_VCPU_OFFSET_EXT_INFOS` = 0x200 }
- Offsets for vCPU state layouts.*

16.104.1 Detailed Description

x86-specific vCPU interface.

Definition in file [__vcpu-arch.h](#).

16.104.2 Enumeration Type Documentation

16.104.2.1 anonymous enum

anonymous enum

Enumerator

| | |
|-----------------------|--|
| L4_VCPU_STATE_VERSION | Architecture-specific version ID. This ID must match the version field in the l4_vcpu_state_t structure after enabling vCPU mode or extended vCPU mode for a thread. |
|-----------------------|--|

Definition at line 16 of file [__vcpu-arch.h](#).

16.105 __vcpu-arch.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00012 #pragma once
00013
00014 #include <l4/sys/types.h>
00015
00016 enum
00017 {
00024     L4_VCPU_STATE_VERSION = 0x46,
00025
00026     L4_VCPU_STATE_SIZE = 0x200,
00027     L4_VCPU_STATE_EXT_SIZE = L4_PAGESIZE,
00028 };
00029
00034 enum L4_vcpu_state_offset
00035 {
00036     L4_VCPU_OFFSET_EXT_STATE = 0x400,
00037     L4_VCPU_OFFSET_EXT_INFOS = 0x200,
00038 };
00039
00044 typedef struct l4_vcpu_regs_t
00045 {
00046     l4_umword_t es;
00047     l4_umword_t ds;
00048     l4_umword_t gs;
00049     l4_umword_t fs;
00050
00051     l4_umword_t di;
00052     l4_umword_t si;
00053     l4_umword_t bp;
00054     l4_umword_t pfa;
00055     l4_umword_t bx;
00056     l4_umword_t dx;
00057     l4_umword_t cx;
00058     l4_umword_t ax;
00059
00060     l4_umword_t trapno;

```

```

00061     l4_umword_t err;
00062
00063     l4_umword_t ip;
00064     l4_umword_t dummy1;
00065     l4_umword_t flags;
00066     l4_umword_t sp;
00067     l4_umword_t ss;
00068 } l4_vcpu_regs_t;
00069
00073 typedef struct l4_vcpu_arch_state_t { } l4_vcpu_arch_state_t;
00074
00079 typedef struct l4_vcpu_ipc_regs_t
00080 {
00081     l4_umword_t _res[2];
00082     l4_umword_t label;
00083     l4_umword_t _res2[3];
00084     l4_msgtag_t tag;
00085 } l4_vcpu_ipc_regs_t;

```

16.106 ktrace_events.h

```

00001 /* Note, automatically generated from Fiasco binary */
00002 #pragma once
00003
00004 enum L4_ktrace_tbuf_entry_fixed
00005 {
00006     l4_ktrace_tbuf_unused = 0,
00007     l4_ktrace_tbuf_pf = 1,
00008     l4_ktrace_tbuf_ipc = 2,
00009     l4_ktrace_tbuf_ipc_res = 3,
00010     l4_ktrace_tbuf_ipc_trace = 4,
00011     l4_ktrace_tbuf_ke = 5,
00012     l4_ktrace_tbuf_ke_reg = 6,
00013     l4_ktrace_tbuf_breakpoint = 7,
00014     l4_ktrace_tbuf_ke_bin = 8,
00015     l4_ktrace_tbuf_dynentries = 9,
00016     l4_ktrace_tbuf_max = 128,
00017     l4_ktrace_tbuf_hidden = 128,
00018 };
00019
00020 typedef unsigned long L4_ktrace_t__Address;
00021 typedef unsigned long L4_ktrace_t__Cap_index;
00022 typedef void L4_ktrace_t__Context;
00023 typedef void L4_ktrace_t__Context__Drq;
00024 typedef unsigned L4_ktrace_t__Context__Drq_log__Type;
00025 typedef unsigned L4_ktrace_t__Cpu_number;
00026 typedef void L4_ktrace_t__Irq_base;
00027 typedef void L4_ktrace_t__Irq_chip;
00028 typedef void L4_ktrace_t__Kobject;
00029 typedef unsigned long L4_ktrace_t__L4_error;
00030 typedef unsigned long L4_ktrace_t__L4_msg_tag;
00031 typedef unsigned long L4_ktrace_t__L4_obj_ref;
00032 typedef unsigned L4_ktrace_t__L4_timeout_pair;
00033 typedef unsigned long L4_ktrace_t__Mword;
00034 typedef void L4_ktrace_t__Rcu_item;
00035 typedef void L4_ktrace_t__Sched_context;
00036 typedef long L4_ktrace_t__Smword;
00037 typedef void L4_ktrace_t__Space;
00038 typedef unsigned short L4_ktrace_t__Unsigned16;
00039 typedef unsigned int L4_ktrace_t__Unsigned32;
00040 typedef unsigned long long L4_ktrace_t__Unsigned64;
00041 typedef unsigned char L4_ktrace_t__Unsigned8;
00042 typedef void L4_ktrace_t__cxx__Type_info;
00043
00044 typedef struct __attribute__((packed))
00045 {
00046     L4_ktrace_t__Mword_number; /* 0+8 */
00047     L4_ktrace_t__Address_ip; /* 8+8 */
00048     L4_ktrace_t__Unsigned64_tsc; /* 16+8 */
00049     L4_ktrace_t__Context *ctx; /* 24+8 */
00050     L4_ktrace_t__Unsigned32_pmc1; /* 32+4 */
00051     L4_ktrace_t__Unsigned32_pmc2; /* 36+4 */
00052     L4_ktrace_t__Unsigned32_kclock; /* 40+4 */
00053     L4_ktrace_t__Unsigned8_type; /* 44+1 */
00054     L4_ktrace_t__Unsigned8_cpu; /* 45+1 */
00055     union __attribute__((packed))
00056     {
00057         struct __attribute__((packed))
00058         {
00059             char __pre_pad[2];
00060             void *func; /* 48+8 */
00061             L4_ktrace_t__Context *thread; /* 56+8 */
00062             L4_ktrace_t__Context__Drq *rq; /* 64+8 */

```

```

00063     L4_ktrace_t__Cpu_number target_cpu; /* 72+4 */
00064     L4_ktrace_t__Context__Drq_log__Type type; /* 76+4 */
00065     char wait; /* 80+1 */
00066 } drq; /* 88 */
00067 struct __attribute__((__packed__))
00068 {
00069     char __pre_pad[2];
00070     L4_ktrace_t__Mword state; /* 48+8 */
00071     L4_ktrace_t__Mword ip; /* 56+8 */
00072     L4_ktrace_t__Mword sp; /* 64+8 */
00073     L4_ktrace_t__Mword space; /* 72+8 */
00074     L4_ktrace_t__Mword err; /* 80+8 */
00075     unsigned char type; /* 88+1 */
00076     unsigned char trap; /* 89+1 */
00077 } vcpu; /* 96 */
00078 struct __attribute__((__packed__))
00079 {
00080     char __pre_pad[2];
00081     L4_ktrace_t__Sword op; /* 48+8 */
00082     L4_ktrace_t__Cap_index buffer; /* 56+8 */
00083     L4_ktrace_t__Mword id; /* 64+8 */
00084     L4_ktrace_t__Mword ram; /* 72+8 */
00085     L4_ktrace_t__Mword newo; /* 80+8 */
00086 } factory; /* 88 */
00087 struct __attribute__((__packed__))
00088 {
00089     char __pre_pad[2];
00090     L4_ktrace_t__Mword gate_dbg_id; /* 48+8 */
00091     L4_ktrace_t__Mword thread_dbg_id; /* 56+8 */
00092     L4_ktrace_t__Mword label; /* 64+8 */
00093 } gate; /* 72 */
00094 struct __attribute__((__packed__))
00095 {
00096     char __pre_pad[2];
00097     L4_ktrace_t__Irq_base *obj; /* 48+8 */
00098     L4_ktrace_t__Irq_chip *chip; /* 56+8 */
00099     L4_ktrace_t__Mword pin; /* 64+8 */
00100 } irq; /* 72 */
00101 struct __attribute__((__packed__))
00102 {
00103     char __pre_pad[2];
00104     L4_ktrace_t__Kobject *obj; /* 48+8 */
00105     L4_ktrace_t__Mword id; /* 56+8 */
00106     L4_ktrace_t__cxx__Type_info *type; /* 64+8 */
00107     L4_ktrace_t__Mword ram; /* 72+8 */
00108 } destroy; /* 80 */
00109 struct __attribute__((__packed__))
00110 {
00111     char __pre_pad[2];
00112     L4_ktrace_t__Cpu_number cpu; /* 48+4 */
00113     char __pad_l[4];
00114     L4_ktrace_t__Rcu_item *item; /* 56+8 */
00115     void *cb; /* 64+8 */
00116     unsigned char event; /* 72+1 */
00117 } rcu; /* 80 */
00118 struct __attribute__((__packed__))
00119 {
00120     char __pre_pad[2];
00121     L4_ktrace_t__Mword id; /* 48+8 */
00122     L4_ktrace_t__Mword mask; /* 56+8 */
00123     L4_ktrace_t__Mword fpage; /* 64+8 */
00124     char map; /* 72+1 */
00125 } tmap; /* 80 */
00126 struct __attribute__((__packed__))
00127 {
00128     char __pre_pad[2];
00129     L4_ktrace_t__Address _address; /* 48+8 */
00130     int _len; /* 56+4 */
00131     char __pad_l[4];
00132     L4_ktrace_t__Mword _value; /* 64+8 */
00133     int _mode; /* 72+4 */
00134 } bp; /* 80 */
00135 struct __attribute__((__packed__))
00136 {
00137     char __pre_pad[2];
00138     L4_ktrace_t__Context *dst; /* 48+8 */
00139     L4_ktrace_t__Context *dst_orig; /* 56+8 */
00140     L4_ktrace_t__Address kernel_ip; /* 64+8 */
00141     L4_ktrace_t__Mword lock_cnt; /* 72+8 */
00142     L4_ktrace_t__Space *from_space; /* 80+8 */
00143     L4_ktrace_t__Sched_context *from_sched; /* 88+8 */
00144     L4_ktrace_t__Mword from_prio; /* 96+8 */
00145 } context_switch; /* 104 */
00146 struct __attribute__((__packed__))
00147 {
00148     } empty; /* 48 */
00149 struct __attribute__((__packed__))

```

```

00150     {
00151         char __pre_pad[2];
00152         L4_ktrace_t__L4_msg_tag_tag; /* 48+8 */
00153         L4_ktrace_t__Mword_dword[2]; /* 56+16 */
00154         L4_ktrace_t__L4_obj_ref_dst; /* 72+8 */
00155         L4_ktrace_t__Mword_dbg_id; /* 80+8 */
00156         L4_ktrace_t__Mword_label; /* 88+8 */
00157         L4_ktrace_t__L4_timeout_pair_timeout; /* 96+4 */
00158         char __pad_1[4];
00159         L4_ktrace_t__Unsigned64_to_abs_rcv; /* 104+8 */
00160     } ipc; /* 112 */
00161     struct __attribute__((__packed__))
00162     {
00163         L4_ktrace_t__Unsigned8_have_snd; /* 46+1 */
00164         L4_ktrace_t__Unsigned8_is_np; /* 47+1 */
00165         L4_ktrace_t__L4_msg_tag_tag; /* 48+8 */
00166         L4_ktrace_t__Mword_dword[2]; /* 56+16 */
00167         L4_ktrace_t__L4_error_result; /* 72+8 */
00168         L4_ktrace_t__Mword_from; /* 80+8 */
00169         L4_ktrace_t__L4_obj_ref_dst; /* 88+8 */
00170         L4_ktrace_t__Mword_pair_event; /* 96+8 */
00171     } ipc_res; /* 104 */
00172     struct __attribute__((__packed__))
00173     {
00174         char __pre_pad[2];
00175         union __attribute__((__packed__)) {
00176             char msg[80]; /* 0+80 */
00177             struct __attribute__((__packed__)) {
00178                 char tag[2]; /* 0+2 */
00179                 char __pad_1[6];
00180                 char *ptr; /* 8+8 */
00181             } mptr; /* 0+16 */
00182         } msg; /* 48+80 */
00183     } ke; /* 128 */
00184     struct __attribute__((__packed__))
00185     {
00186         char _msg[80]; /* 46+80 */
00187     } ke_bin; /* 128 */
00188     struct __attribute__((__packed__))
00189     {
00190         char __pre_pad[2];
00191         L4_ktrace_t__Mword v[3]; /* 48+24 */
00192         union __attribute__((__packed__)) {
00193             char msg[56]; /* 0+56 */
00194             struct __attribute__((__packed__)) {
00195                 char tag[2]; /* 0+2 */
00196                 char __pad_1[6];
00197                 char *ptr; /* 8+8 */
00198             } mptr; /* 0+16 */
00199         } msg; /* 72+56 */
00200     } ke_reg; /* 128 */
00201     struct __attribute__((__packed__))
00202     {
00203         char __pre_pad[2];
00204         L4_ktrace_t__Address_pfa; /* 48+8 */
00205         L4_ktrace_t__Mword_error; /* 56+8 */
00206         L4_ktrace_t__Space *space; /* 64+8 */
00207     } pf; /* 72 */
00208     struct __attribute__((__packed__))
00209     {
00210         unsigned short mode; /* 46+2 */
00211         L4_ktrace_t__Context *owner; /* 48+8 */
00212         unsigned short id; /* 56+2 */
00213         unsigned short prio; /* 58+2 */
00214         char __pad_1[4];
00215         long left; /* 64+8 */
00216         unsigned long quantum; /* 72+8 */
00217     } sched; /* 80 */
00218     struct __attribute__((__packed__))
00219     {
00220         char _trapno; /* 46+1 */
00221         char __pad_1[1];
00222         L4_ktrace_t__Unsigned16_error; /* 48+2 */
00223         char __pad_2[6];
00224         L4_ktrace_t__Mword_rbp; /* 56+8 */
00225         L4_ktrace_t__Mword_cr2; /* 64+8 */
00226         L4_ktrace_t__Mword_rax; /* 72+8 */
00227         L4_ktrace_t__Mword_rflags; /* 80+8 */
00228         L4_ktrace_t__Mword_rsp; /* 88+8 */
00229         L4_ktrace_t__Unsigned16_cs; /* 96+2 */
00230         L4_ktrace_t__Unsigned16_ds; /* 98+2 */
00231     } trap; /* 104 */
00232     struct __attribute__((__packed__))
00233     {
00234         char _padding[80]; /* 46+80 */
00235         char __post_pad[2]; /* 126+2 */
00236     } fullsize; /* 128 */

```



```

00237     struct __attribute__((__packed__))
00238     {
00239         char __pre_pad[2];
00240         L4_ktrace_t__Cap_index cap_idx; /* 48+8 */
00241     } ieh; /* 56 */
00242     struct __attribute__((__packed__))
00243     {
00244         char __pre_pad[2];
00245         L4_ktrace_t__Mword pfa; /* 48+8 */
00246         L4_ktrace_t__Cap_index cap_idx; /* 56+8 */
00247         L4_ktrace_t__Mword err; /* 64+8 */
00248     } ipfh; /* 72 */
00249     struct __attribute__((__packed__))
00250     {
00251         char __pre_pad[2];
00252         L4_ktrace_t__Mword id; /* 48+8 */
00253         L4_ktrace_t__Mword ip; /* 56+8 */
00254         L4_ktrace_t__Mword sp; /* 64+8 */
00255         L4_ktrace_t__Mword op; /* 72+8 */
00256     } exregs; /* 80 */
00257     struct __attribute__((__packed__))
00258     {
00259         char __pre_pad[2];
00260         L4_ktrace_t__Mword state; /* 48+8 */
00261         L4_ktrace_t__Address user_ip; /* 56+8 */
00262         L4_ktrace_t__Cpu_number src_cpu; /* 64+4 */
00263         L4_ktrace_t__Cpu_number target_cpu; /* 68+4 */
00264     } migration; /* 72 */
00265     struct __attribute__((__packed__))
00266     {
00267         char __pre_pad[2];
00268         L4_ktrace_t__Address user_ip; /* 48+8 */
00269     } timer; /* 56 */
00270     struct __attribute__((__packed__))
00271     {
00272         char __pre_pad[2];
00273         L4_ktrace_t__Mword exitcode; /* 48+8 */
00274         L4_ktrace_t__Mword exitinfo1; /* 56+8 */
00275         L4_ktrace_t__Mword exitinfo2; /* 64+8 */
00276         L4_ktrace_t__Mword rip; /* 72+8 */
00277     } svm; /* 80 */
00278     } m;
00279 } l4_tracebuffer_entry_t;

```

16.107 ktrace_events.h

```

00001 /* Note, automatically generated from Fiasco binary */
00002 #pragma once
00003
00004 enum L4_ktrace_tbuf_entry_fixed
00005 {
00006     l4_ktrace_tbuf_unused = 0,
00007     l4_ktrace_tbuf_pf = 1,
00008     l4_ktrace_tbuf_ipc = 2,
00009     l4_ktrace_tbuf_ipc_res = 3,
00010     l4_ktrace_tbuf_ipc_trace = 4,
00011     l4_ktrace_tbuf_ke = 5,
00012     l4_ktrace_tbuf_ke_reg = 6,
00013     l4_ktrace_tbuf_breakpoint = 7,
00014     l4_ktrace_tbuf_ke_bin = 8,
00015     l4_ktrace_tbuf_dynentries = 9,
00016     l4_ktrace_tbuf_max = 128,
00017     l4_ktrace_tbuf_hidden = 128,
00018 };
00019
00020 typedef unsigned long L4_ktrace_t__Address;
00021 typedef unsigned long L4_ktrace_t__Cap_index;
00022 typedef void L4_ktrace_t__Context;
00023 typedef void L4_ktrace_t__Context__Drq;
00024 typedef unsigned L4_ktrace_t__Context__Drq_log__Type;
00025 typedef unsigned L4_ktrace_t__Cpu_number;
00026 typedef void L4_ktrace_t__Irq_base;
00027 typedef void L4_ktrace_t__Irq_chip;
00028 typedef void L4_ktrace_t__Kobject;
00029 typedef unsigned long L4_ktrace_t__L4_error;
00030 typedef unsigned long L4_ktrace_t__L4_msg_tag;
00031 typedef unsigned long L4_ktrace_t__L4_obj_ref;
00032 typedef unsigned L4_ktrace_t__L4_timeout_pair;
00033 typedef unsigned long L4_ktrace_t__Mword;
00034 typedef void L4_ktrace_t__Rcu_item;
00035 typedef void L4_ktrace_t__Sched_context;
00036 typedef long L4_ktrace_t__Sword;
00037 typedef void L4_ktrace_t__Space;

```

```

00038 typedef unsigned int L4_ktrace_t__Unsigned32;
00039 typedef unsigned long long L4_ktrace_t__Unsigned64;
00040 typedef unsigned char L4_ktrace_t__Unsigned8;
00041 typedef void L4_ktrace_t__cxx_Type_info;
00042
00043 typedef struct __attribute__((packed))
00044 {
00045     L4_ktrace_t__Mword _number; /* 0+4 */
00046     L4_ktrace_t__Address _ip; /* 4+4 */
00047     L4_ktrace_t__Unsigned64 _tsc; /* 8+8 */
00048     L4_ktrace_t__Context *_ctx; /* 16+4 */
00049     L4_ktrace_t__Unsigned32 _pmc1; /* 20+4 */
00050     L4_ktrace_t__Unsigned32 _pmc2; /* 24+4 */
00051     L4_ktrace_t__Unsigned32 _kclock; /* 28+4 */
00052     L4_ktrace_t__Unsigned8 _type; /* 32+1 */
00053     L4_ktrace_t__Unsigned8 _cpu; /* 33+1 */
00054     union __attribute__((__packed__))
00055     {
00056         struct __attribute__((__packed__))
00057         {
00058             char __pre_pad[2];
00059             void *func; /* 36+4 */
00060             L4_ktrace_t__Context *thread; /* 40+4 */
00061             L4_ktrace_t__Context__Drq *rq; /* 44+4 */
00062             L4_ktrace_t__Cpu_number target_cpu; /* 48+4 */
00063             L4_ktrace_t__Context__Drq_log__Type type; /* 52+4 */
00064             char wait; /* 56+1 */
00065         } drq; /* 64 */
00066         struct __attribute__((__packed__))
00067         {
00068             char __pre_pad[2];
00069             L4_ktrace_t__Mword state; /* 36+4 */
00070             L4_ktrace_t__Mword ip; /* 40+4 */
00071             L4_ktrace_t__Mword sp; /* 44+4 */
00072             L4_ktrace_t__Mword space; /* 48+4 */
00073             L4_ktrace_t__Mword err; /* 52+4 */
00074             unsigned char type; /* 56+1 */
00075             unsigned char trap; /* 57+1 */
00076         } vcpu; /* 64 */
00077         struct __attribute__((__packed__))
00078         {
00079             char __pre_pad[2];
00080             L4_ktrace_t__Smword op; /* 36+4 */
00081             L4_ktrace_t__Cap_index buffer; /* 40+4 */
00082             L4_ktrace_t__Mword id; /* 44+4 */
00083             L4_ktrace_t__Mword ram; /* 48+4 */
00084             L4_ktrace_t__Mword newo; /* 52+4 */
00085         } factory; /* 56 */
00086         struct __attribute__((__packed__))
00087         {
00088             char __pre_pad[2];
00089             L4_ktrace_t__Mword gate_dbg_id; /* 36+4 */
00090             L4_ktrace_t__Mword thread_dbg_id; /* 40+4 */
00091             L4_ktrace_t__Mword label; /* 44+4 */
00092         } gate; /* 48 */
00093         struct __attribute__((__packed__))
00094         {
00095             char __pre_pad[2];
00096             L4_ktrace_t__Irq_base *obj; /* 36+4 */
00097             L4_ktrace_t__Irq_chip *chip; /* 40+4 */
00098             L4_ktrace_t__Mword pin; /* 44+4 */
00099         } irq; /* 48 */
00100         struct __attribute__((__packed__))
00101         {
00102             char __pre_pad[2];
00103             L4_ktrace_t__Kobject *obj; /* 36+4 */
00104             L4_ktrace_t__Mword id; /* 40+4 */
00105             L4_ktrace_t__cxx_Type_info *type; /* 44+4 */
00106             L4_ktrace_t__Mword ram; /* 48+4 */
00107         } destroy; /* 56 */
00108         struct __attribute__((__packed__))
00109         {
00110             char __pre_pad[2];
00111             L4_ktrace_t__Cpu_number cpu; /* 36+4 */
00112             L4_ktrace_t__Rcu_item *item; /* 40+4 */
00113             void *cb; /* 44+4 */
00114             unsigned char event; /* 48+1 */
00115         } rcu; /* 56 */
00116         struct __attribute__((__packed__))
00117         {
00118             char __pre_pad[2];
00119             L4_ktrace_t__Mword id; /* 36+4 */
00120             L4_ktrace_t__Mword mask; /* 40+4 */
00121             L4_ktrace_t__Mword fpage; /* 44+4 */
00122             char map; /* 48+1 */
00123         } tmap; /* 56 */
00124         struct __attribute__((__packed__))

```

```

00125     {
00126         char __pre_pad[2];
00127         L4_ktrace_t__Address _address; /* 36+4 */
00128         int _len; /* 40+4 */
00129         L4_ktrace_t__Mword _value; /* 44+4 */
00130         int _mode; /* 48+4 */
00131     } bp; /* 56 */
00132     struct __attribute__((__packed__))
00133     {
00134         char __pre_pad[2];
00135         L4_ktrace_t__Context *dst; /* 36+4 */
00136         L4_ktrace_t__Context *dst_orig; /* 40+4 */
00137         L4_ktrace_t__Address kernel_ip; /* 44+4 */
00138         L4_ktrace_t__Mword lock_cnt; /* 48+4 */
00139         L4_ktrace_t__Space *from_space; /* 52+4 */
00140         L4_ktrace_t__Sched_context *from_sched; /* 56+4 */
00141         L4_ktrace_t__Mword from_prio; /* 60+4 */
00142     } context_switch; /* 64 */
00143     struct __attribute__((__packed__))
00144     {
00145     } empty; /* 40 */
00146     struct __attribute__((__packed__))
00147     {
00148         char __pre_pad[2];
00149         L4_ktrace_t__L4_msg_tag _tag; /* 36+4 */
00150         L4_ktrace_t__Mword _dword[2]; /* 40+8 */
00151         L4_ktrace_t__L4_obj_ref _dst; /* 48+4 */
00152         L4_ktrace_t__Mword _dbg_id; /* 52+4 */
00153         L4_ktrace_t__Mword _label; /* 56+4 */
00154         L4_ktrace_t__L4_timeout_pair _timeout; /* 60+4 */
00155     } ipc; /* 64 */
00156     struct __attribute__((__packed__))
00157     {
00158         L4_ktrace_t__Unsigned8 _have_snd; /* 34+1 */
00159         L4_ktrace_t__Unsigned8 _is_np; /* 35+1 */
00160         L4_ktrace_t__L4_msg_tag _tag; /* 36+4 */
00161         L4_ktrace_t__Mword _dword[2]; /* 40+8 */
00162         L4_ktrace_t__L4_error _result; /* 48+4 */
00163         L4_ktrace_t__Mword _from; /* 52+4 */
00164         L4_ktrace_t__L4_obj_ref _dst; /* 56+4 */
00165         L4_ktrace_t__Mword _pair_event; /* 60+4 */
00166     } ipc_res; /* 64 */
00167     struct __attribute__((__packed__))
00168     {
00169         char __pre_pad[2];
00170         union __attribute__((__packed__)) {
00171             char msg[24]; /* 0+24 */
00172             struct __attribute__((__packed__)) {
00173                 char tag[2]; /* 0+2 */
00174                 char __pad_1[2];
00175                 char *ptr; /* 4+4 */
00176             } mptr; /* 0+8 */
00177             } msg; /* 36+24 */
00178         } ke; /* 64 */
00179     struct __attribute__((__packed__))
00180     {
00181         char _msg[24]; /* 34+24 */
00182     } ke_bin; /* 64 */
00183     struct __attribute__((__packed__))
00184     {
00185         char __pre_pad[2];
00186         L4_ktrace_t__Mword v[3]; /* 36+12 */
00187         union __attribute__((__packed__)) {
00188             char msg[12]; /* 0+12 */
00189             struct __attribute__((__packed__)) {
00190                 char tag[2]; /* 0+2 */
00191                 char __pad_1[2];
00192                 char *ptr; /* 4+4 */
00193             } mptr; /* 0+8 */
00194             } msg; /* 48+12 */
00195         } ke_reg; /* 64 */
00196     struct __attribute__((__packed__))
00197     {
00198         char __pre_pad[2];
00199         L4_ktrace_t__Address _pfa; /* 36+4 */
00200         L4_ktrace_t__Mword _error; /* 40+4 */
00201         L4_ktrace_t__Space *_space; /* 44+4 */
00202     } pf; /* 48 */
00203     struct __attribute__((__packed__))
00204     {
00205         unsigned short mode; /* 34+2 */
00206         L4_ktrace_t__Context *owner; /* 36+4 */
00207         unsigned short id; /* 40+2 */
00208         unsigned short prio; /* 42+2 */
00209         long left; /* 44+4 */
00210         unsigned long quantum; /* 48+4 */
00211     } sched; /* 56 */

```

```

00212     struct __attribute__((__packed__))
00213     {
00214         char __pre_pad[2];
00215         L4_ktrace_t__Unsigned32 _error; /* 36+4 */
00216         L4_ktrace_t__Mword _cpsr; /* 40+4 */
00217         L4_ktrace_t__Mword _sp; /* 44+4 */
00218     } trap; /* 48 */
00219     struct __attribute__((__packed__))
00220     {
00221         char _padding[24]; /* 34+24 */
00222         char __post_pad[6]; /* 58+6 */
00223     } fullsize; /* 64 */
00224     struct __attribute__((__packed__))
00225     {
00226         char __pre_pad[2];
00227         L4_ktrace_t__Cap_index cap_idx; /* 36+4 */
00228     } ieh; /* 40 */
00229     struct __attribute__((__packed__))
00230     {
00231         char __pre_pad[2];
00232         L4_ktrace_t__Mword pfa; /* 36+4 */
00233         L4_ktrace_t__Cap_index cap_idx; /* 40+4 */
00234         L4_ktrace_t__Mword err; /* 44+4 */
00235     } ipfh; /* 48 */
00236     struct __attribute__((__packed__))
00237     {
00238         char __pre_pad[2];
00239         L4_ktrace_t__Mword id; /* 36+4 */
00240         L4_ktrace_t__Mword ip; /* 40+4 */
00241         L4_ktrace_t__Mword sp; /* 44+4 */
00242         L4_ktrace_t__Mword op; /* 48+4 */
00243     } exregs; /* 56 */
00244     struct __attribute__((__packed__))
00245     {
00246         char __pre_pad[2];
00247         L4_ktrace_t__Mword state; /* 36+4 */
00248         L4_ktrace_t__Address user_ip; /* 40+4 */
00249         L4_ktrace_t__Cpu_number src_cpu; /* 44+4 */
00250         L4_ktrace_t__Cpu_number target_cpu; /* 48+4 */
00251     } migration; /* 56 */
00252     struct __attribute__((__packed__))
00253     {
00254         char __pre_pad[2];
00255         L4_ktrace_t__Address user_ip; /* 36+4 */
00256     } timer; /* 40 */
00257     } m;
00258 } l4_tracebuffer_entry_t;

```

16.108 ktrace_events.h

```

00001 /* Note, automatically generated from Fiasco binary */
00002 #pragma once
00003
00004 enum L4_ktrace_tbuf_entry_fixed
00005 {
00006     l4_ktrace_tbuf_unused = 0,
00007     l4_ktrace_tbuf_pf = 1,
00008     l4_ktrace_tbuf_ipc = 2,
00009     l4_ktrace_tbuf_ipc_res = 3,
00010     l4_ktrace_tbuf_ipc_trace = 4,
00011     l4_ktrace_tbuf_ke = 5,
00012     l4_ktrace_tbuf_ke_reg = 6,
00013     l4_ktrace_tbuf_breakpoint = 7,
00014     l4_ktrace_tbuf_ke_bin = 8,
00015     l4_ktrace_tbuf_dynentries = 9,
00016     l4_ktrace_tbuf_max = 128,
00017     l4_ktrace_tbuf_hidden = 128,
00018 };
00019
00020 typedef unsigned long L4_ktrace_t__Address;
00021 typedef unsigned long L4_ktrace_t__Cap_index;
00022 typedef void L4_ktrace_t__Context;
00023 typedef void L4_ktrace_t__Context__Drq;
00024 typedef unsigned L4_ktrace_t__Context__Drq_log__Type;
00025 typedef unsigned L4_ktrace_t__Cpu_number;
00026 typedef void L4_ktrace_t__Irq_base;
00027 typedef void L4_ktrace_t__Irq_chip;
00028 typedef void L4_ktrace_t__Kobject;
00029 typedef unsigned long L4_ktrace_t__L4_error;
00030 typedef unsigned long L4_ktrace_t__L4_msg_tag;
00031 typedef unsigned long L4_ktrace_t__L4_obj_ref;
00032 typedef unsigned L4_ktrace_t__L4_timeout_pair;
00033 typedef unsigned long L4_ktrace_t__Mword;

```

```

00034 typedef void L4_ktrace_t__Rcu_item;
00035 typedef void L4_ktrace_t__Sched_context;
00036 typedef long L4_ktrace_t__Smword;
00037 typedef void L4_ktrace_t__Space;
00038 typedef unsigned int L4_ktrace_t__Unsigned32;
00039 typedef unsigned long long L4_ktrace_t__Unsigned64;
00040 typedef unsigned char L4_ktrace_t__Unsigned8;
00041 typedef void L4_ktrace_t__cxx__Type_info;
00042
00043 typedef struct __attribute__((packed))
00044 {
00045     L4_ktrace_t__Mword _number; /* 0+8 */
00046     L4_ktrace_t__Address _ip; /* 8+8 */
00047     L4_ktrace_t__Unsigned64 _tsc; /* 16+8 */
00048     L4_ktrace_t__Context *_ctx; /* 24+8 */
00049     L4_ktrace_t__Unsigned32 _pmc1; /* 32+4 */
00050     L4_ktrace_t__Unsigned32 _pmc2; /* 36+4 */
00051     L4_ktrace_t__Unsigned32 _kclock; /* 40+4 */
00052     L4_ktrace_t__Unsigned8 _type; /* 44+1 */
00053     L4_ktrace_t__Unsigned8 _cpu; /* 45+1 */
00054     union __attribute__((__packed__))
00055     {
00056         struct __attribute__((__packed__))
00057         {
00058             char __pre_pad[2];
00059             void *func; /* 48+8 */
00060             L4_ktrace_t__Context *thread; /* 56+8 */
00061             L4_ktrace_t__Context__Drq *rq; /* 64+8 */
00062             L4_ktrace_t__Cpu_number target_cpu; /* 72+4 */
00063             L4_ktrace_t__Context__Drq_log__Type type; /* 76+4 */
00064             char wait; /* 80+1 */
00065         } drq; /* 88 */
00066         struct __attribute__((__packed__))
00067         {
00068             char __pre_pad[2];
00069             L4_ktrace_t__Mword state; /* 48+8 */
00070             L4_ktrace_t__Mword ip; /* 56+8 */
00071             L4_ktrace_t__Mword sp; /* 64+8 */
00072             L4_ktrace_t__Mword space; /* 72+8 */
00073             L4_ktrace_t__Mword err; /* 80+8 */
00074             unsigned char type; /* 88+1 */
00075             unsigned char trap; /* 89+1 */
00076         } vcpu; /* 96 */
00077         struct __attribute__((__packed__))
00078         {
00079             char __pre_pad[2];
00080             L4_ktrace_t__Smword op; /* 48+8 */
00081             L4_ktrace_t__Cap_index buffer; /* 56+8 */
00082             L4_ktrace_t__Mword id; /* 64+8 */
00083             L4_ktrace_t__Mword ram; /* 72+8 */
00084             L4_ktrace_t__Mword newo; /* 80+8 */
00085         } factory; /* 88 */
00086         struct __attribute__((__packed__))
00087         {
00088             char __pre_pad[2];
00089             L4_ktrace_t__Mword gate_dbg_id; /* 48+8 */
00090             L4_ktrace_t__Mword thread_dbg_id; /* 56+8 */
00091             L4_ktrace_t__Mword label; /* 64+8 */
00092         } gate; /* 72 */
00093         struct __attribute__((__packed__))
00094         {
00095             char __pre_pad[2];
00096             L4_ktrace_t__Irq_base *obj; /* 48+8 */
00097             L4_ktrace_t__Irq_chip *chip; /* 56+8 */
00098             L4_ktrace_t__Mword pin; /* 64+8 */
00099         } irq; /* 72 */
00100         struct __attribute__((__packed__))
00101         {
00102             char __pre_pad[2];
00103             L4_ktrace_t__Kobject *obj; /* 48+8 */
00104             L4_ktrace_t__Mword id; /* 56+8 */
00105             L4_ktrace_t__cxx__Type_info *type; /* 64+8 */
00106             L4_ktrace_t__Mword ram; /* 72+8 */
00107         } destroy; /* 80 */
00108         struct __attribute__((__packed__))
00109         {
00110             char __pre_pad[2];
00111             L4_ktrace_t__Cpu_number cpu; /* 48+4 */
00112             char __pad_l[4];
00113             L4_ktrace_t__Rcu_item *item; /* 56+8 */
00114             void *cb; /* 64+8 */
00115             unsigned char event; /* 72+1 */
00116         } rcu; /* 80 */
00117         struct __attribute__((__packed__))
00118         {
00119             char __pre_pad[2];
00120             L4_ktrace_t__Mword id; /* 48+8 */

```

```

00121     L4_ktrace_t__Mword mask; /* 56+8 */
00122     L4_ktrace_t__Mword fpage; /* 64+8 */
00123     char map; /* 72+1 */
00124 } tmap; /* 80 */
00125 struct __attribute__((__packed__))
00126 {
00127     char __pre_pad[2];
00128     L4_ktrace_t__Address _address; /* 48+8 */
00129     int _len; /* 56+4 */
00130     char __pad_1[4];
00131     L4_ktrace_t__Mword _value; /* 64+8 */
00132     int _mode; /* 72+4 */
00133 } bp; /* 80 */
00134 struct __attribute__((__packed__))
00135 {
00136     char __pre_pad[2];
00137     L4_ktrace_t__Context *dst; /* 48+8 */
00138     L4_ktrace_t__Context *dst_orig; /* 56+8 */
00139     L4_ktrace_t__Address kernel_ip; /* 64+8 */
00140     L4_ktrace_t__Mword lock_cnt; /* 72+8 */
00141     L4_ktrace_t__Space *from_space; /* 80+8 */
00142     L4_ktrace_t__Sched_context *from_sched; /* 88+8 */
00143     L4_ktrace_t__Mword from_prio; /* 96+8 */
00144 } context_switch; /* 104 */
00145 struct __attribute__((__packed__))
00146 {
00147 } empty; /* 48 */
00148 struct __attribute__((__packed__))
00149 {
00150     char __pre_pad[2];
00151     L4_ktrace_t__L4_msg_tag _tag; /* 48+8 */
00152     L4_ktrace_t__Mword _dword[2]; /* 56+16 */
00153     L4_ktrace_t__L4_obj_ref _dst; /* 72+8 */
00154     L4_ktrace_t__Mword _dbg_id; /* 80+8 */
00155     L4_ktrace_t__Mword _label; /* 88+8 */
00156     L4_ktrace_t__L4_timeout_pair _timeout; /* 96+4 */
00157     char __pad_1[4];
00158     L4_ktrace_t__Unsigned64 _to_abs_rcv; /* 104+8 */
00159 } ipc; /* 112 */
00160 struct __attribute__((__packed__))
00161 {
00162     L4_ktrace_t__Unsigned8 _have_snd; /* 46+1 */
00163     L4_ktrace_t__Unsigned8 _is_np; /* 47+1 */
00164     L4_ktrace_t__L4_msg_tag _tag; /* 48+8 */
00165     L4_ktrace_t__Mword _dword[2]; /* 56+16 */
00166     L4_ktrace_t__L4_error_result; /* 72+8 */
00167     L4_ktrace_t__Mword _from; /* 80+8 */
00168     L4_ktrace_t__L4_obj_ref _dst; /* 88+8 */
00169     L4_ktrace_t__Mword _pair_event; /* 96+8 */
00170 } ipc_res; /* 104 */
00171 struct __attribute__((__packed__))
00172 {
00173     char __pre_pad[2];
00174     union __attribute__((__packed__)) {
00175         char msg[80]; /* 0+80 */
00176         struct __attribute__((__packed__)) {
00177             char tag[2]; /* 0+2 */
00178             char __pad_1[6];
00179             char *ptr; /* 8+8 */
00180         } mptr; /* 0+16 */
00181     } msg; /* 48+80 */
00182 } ke; /* 128 */
00183 struct __attribute__((__packed__))
00184 {
00185     char _msg[80]; /* 46+80 */
00186 } ke_bin; /* 128 */
00187 struct __attribute__((__packed__))
00188 {
00189     char __pre_pad[2];
00190     L4_ktrace_t__Mword v[3]; /* 48+24 */
00191     union __attribute__((__packed__)) {
00192         char msg[56]; /* 0+56 */
00193         struct __attribute__((__packed__)) {
00194             char tag[2]; /* 0+2 */
00195             char __pad_1[6];
00196             char *ptr; /* 8+8 */
00197         } mptr; /* 0+16 */
00198     } msg; /* 72+56 */
00199 } ke_reg; /* 128 */
00200 struct __attribute__((__packed__))
00201 {
00202     char __pre_pad[2];
00203     L4_ktrace_t__Address _pfa; /* 48+8 */
00204     L4_ktrace_t__Mword _error; /* 56+8 */
00205     L4_ktrace_t__Space *_space; /* 64+8 */
00206 } pf; /* 72 */
00207 struct __attribute__((__packed__))

```

```

00208     {
00209         unsigned short mode; /* 46+2 */
00210         L4_ktrace_t__Context *owner; /* 48+8 */
00211         unsigned short id; /* 56+2 */
00212         unsigned short prio; /* 58+2 */
00213         char __pad_l[4];
00214         long left; /* 64+8 */
00215         unsigned long quantum; /* 72+8 */
00216     } sched; /* 80 */
00217     struct __attribute__((__packed__))
00218     {
00219         char __pre_pad[2];
00220         L4_ktrace_t__Unsigned32 _error; /* 48+4 */
00221         char __pad_l[4];
00222         L4_ktrace_t__Mword _cpsr; /* 56+8 */
00223         L4_ktrace_t__Mword _sp; /* 64+8 */
00224     } trap; /* 72 */
00225     struct __attribute__((__packed__))
00226     {
00227         char _padding[80]; /* 46+80 */
00228         char __post_pad[2]; /* 126+2 */
00229     } fullsize; /* 128 */
00230     struct __attribute__((__packed__))
00231     {
00232         char __pre_pad[2];
00233         L4_ktrace_t__Cap_index cap_idx; /* 48+8 */
00234     } ieh; /* 56 */
00235     struct __attribute__((__packed__))
00236     {
00237         char __pre_pad[2];
00238         L4_ktrace_t__Mword pfa; /* 48+8 */
00239         L4_ktrace_t__Cap_index cap_idx; /* 56+8 */
00240         L4_ktrace_t__Mword err; /* 64+8 */
00241     } ipfh; /* 72 */
00242     struct __attribute__((__packed__))
00243     {
00244         char __pre_pad[2];
00245         L4_ktrace_t__Mword id; /* 48+8 */
00246         L4_ktrace_t__Mword ip; /* 56+8 */
00247         L4_ktrace_t__Mword sp; /* 64+8 */
00248         L4_ktrace_t__Mword op; /* 72+8 */
00249     } exregs; /* 80 */
00250     struct __attribute__((__packed__))
00251     {
00252         char __pre_pad[2];
00253         L4_ktrace_t__Mword state; /* 48+8 */
00254         L4_ktrace_t__Address user_ip; /* 56+8 */
00255         L4_ktrace_t__Cpu_number src_cpu; /* 64+4 */
00256         L4_ktrace_t__Cpu_number target_cpu; /* 68+4 */
00257     } migration; /* 72 */
00258     struct __attribute__((__packed__))
00259     {
00260         char __pre_pad[2];
00261         L4_ktrace_t__Address user_ip; /* 48+8 */
00262     } timer; /* 56 */
00263     } m;
00264 } l4_tracebuffer_entry_t;

```

16.109 ktrace_events.h

```

00001 /* Note, automatically generated from Fiasco binary */
00002 #pragma once
00003
00004 enum L4_ktrace_tbuf_entry_fixed
00005 {
00006     l4_ktrace_tbuf_unused = 0,
00007     l4_ktrace_tbuf_pf = 1,
00008     l4_ktrace_tbuf_ipc = 2,
00009     l4_ktrace_tbuf_ipc_res = 3,
00010     l4_ktrace_tbuf_ipc_trace = 4,
00011     l4_ktrace_tbuf_ke = 5,
00012     l4_ktrace_tbuf_ke_reg = 6,
00013     l4_ktrace_tbuf_breakpoint = 7,
00014     l4_ktrace_tbuf_ke_bin = 8,
00015     l4_ktrace_tbuf_dynentries = 9,
00016     l4_ktrace_tbuf_max = 128,
00017     l4_ktrace_tbuf_hidden = 128,
00018 };
00019
00020 typedef unsigned long L4_ktrace_t__Address;
00021 typedef unsigned long L4_ktrace_t__Cap_index;
00022 typedef void L4_ktrace_t__Context;
00023 typedef void L4_ktrace_t__Context__Drq;

```

```

00024 typedef unsigned L4_ktrace_t__Context__Drq_log__Type;
00025 typedef unsigned L4_ktrace_t__Cpu_number;
00026 typedef void L4_ktrace_t__Irq_base;
00027 typedef void L4_ktrace_t__Irq_chip;
00028 typedef void L4_ktrace_t__Kobject;
00029 typedef unsigned long L4_ktrace_t__L4_error;
00030 typedef unsigned long L4_ktrace_t__L4_msg_tag;
00031 typedef unsigned long L4_ktrace_t__L4_obj_ref;
00032 typedef unsigned L4_ktrace_t__L4_timeout_pair;
00033 typedef unsigned long L4_ktrace_t__Mword;
00034 typedef void L4_ktrace_t__Rcu_item;
00035 typedef void L4_ktrace_t__Sched_context;
00036 typedef long L4_ktrace_t__Smword;
00037 typedef void L4_ktrace_t__Space;
00038 typedef unsigned int L4_ktrace_t__Unsigned32;
00039 typedef unsigned long long L4_ktrace_t__Unsigned64;
00040 typedef unsigned char L4_ktrace_t__Unsigned8;
00041 typedef void L4_ktrace_t__cxx_Type_info;
00042
00043 #if __riscv_xlen == 32
00044 typedef struct __attribute__((packed))
00045 {
00046     L4_ktrace_t__Mword _number; /* 0+4 */
00047     L4_ktrace_t__Address _ip; /* 4+4 */
00048     L4_ktrace_t__Unsigned64 _tsc; /* 8+8 */
00049     L4_ktrace_t__Context *_ctx; /* 16+4 */
00050     L4_ktrace_t__Unsigned32 _pmc1; /* 20+4 */
00051     L4_ktrace_t__Unsigned32 _pmc2; /* 24+4 */
00052     L4_ktrace_t__Unsigned32 _kclock; /* 28+4 */
00053     L4_ktrace_t__Unsigned8 _type; /* 32+1 */
00054     L4_ktrace_t__Unsigned8 _cpu; /* 33+1 */
00055     union __attribute__((packed))
00056     {
00057         struct __attribute__((packed))
00058         {
00059             char __pre_pad[2];
00060             void *func; /* 36+4 */
00061             L4_ktrace_t__Context *thread; /* 40+4 */
00062             L4_ktrace_t__Context__Drq *rq; /* 44+4 */
00063             L4_ktrace_t__Cpu_number target_cpu; /* 48+4 */
00064             L4_ktrace_t__Context__Drq_log__Type type; /* 52+4 */
00065             char wait; /* 56+1 */
00066         } drq; /* 64 */
00067         struct __attribute__((packed))
00068         {
00069             char __pre_pad[2];
00070             L4_ktrace_t__Mword state; /* 36+4 */
00071             L4_ktrace_t__Mword ip; /* 40+4 */
00072             L4_ktrace_t__Mword sp; /* 44+4 */
00073             L4_ktrace_t__Mword space; /* 48+4 */
00074             L4_ktrace_t__Mword err; /* 52+4 */
00075             unsigned char type; /* 56+1 */
00076             unsigned char trap; /* 57+1 */
00077         } vcpu; /* 64 */
00078     } __attribute__((packed))
00079     {
00080         char __pre_pad[2];
00081         L4_ktrace_t__Smword op; /* 36+4 */
00082         L4_ktrace_t__Cap_index buffer; /* 40+4 */
00083         L4_ktrace_t__Mword id; /* 44+4 */
00084         L4_ktrace_t__Mword ram; /* 48+4 */
00085         L4_ktrace_t__Mword newo; /* 52+4 */
00086     } factory; /* 56 */
00087     struct __attribute__((packed))
00088     {
00089         char __pre_pad[2];
00090         L4_ktrace_t__Mword gate_dbg_id; /* 36+4 */
00091         L4_ktrace_t__Mword thread_dbg_id; /* 40+4 */
00092         L4_ktrace_t__Mword label; /* 44+4 */
00093     } gate; /* 48 */
00094     struct __attribute__((packed))
00095     {
00096         char __pre_pad[2];
00097         L4_ktrace_t__Irq_base *obj; /* 36+4 */
00098         L4_ktrace_t__Irq_chip *chip; /* 40+4 */
00099         L4_ktrace_t__Mword pin; /* 44+4 */
00100     } irq; /* 48 */
00101     struct __attribute__((packed))
00102     {
00103         char __pre_pad[2];
00104         L4_ktrace_t__Kobject *obj; /* 36+4 */
00105         L4_ktrace_t__Mword id; /* 40+4 */
00106         L4_ktrace_t__cxx_Type_info *type; /* 44+4 */
00107         L4_ktrace_t__Mword ram; /* 48+4 */
00108     } destroy; /* 56 */
00109     struct __attribute__((packed))
00110     {

```



```

00111     char __pre_pad[2];
00112     L4_ktrace_t__Cpu_number cpu; /* 36+4 */
00113     L4_ktrace_t__Rcu_item *item; /* 40+4 */
00114     void *cb; /* 44+4 */
00115     unsigned char event; /* 48+1 */
00116 } rcu; /* 56 */
00117 struct __attribute__((__packed__))
00118 {
00119     char __pre_pad[2];
00120     L4_ktrace_t__Mword id; /* 36+4 */
00121     L4_ktrace_t__Mword mask; /* 40+4 */
00122     L4_ktrace_t__Mword fpage; /* 44+4 */
00123     char map; /* 48+1 */
00124 } tmap; /* 56 */
00125 struct __attribute__((__packed__))
00126 {
00127     char __pre_pad[2];
00128     L4_ktrace_t__Address _address; /* 36+4 */
00129     int _len; /* 40+4 */
00130     L4_ktrace_t__Mword _value; /* 44+4 */
00131     int _mode; /* 48+4 */
00132 } bp; /* 56 */
00133 struct __attribute__((__packed__))
00134 {
00135     char __pre_pad[2];
00136     L4_ktrace_t__Context *dst; /* 36+4 */
00137     L4_ktrace_t__Context *dst_orig; /* 40+4 */
00138     L4_ktrace_t__Address kernel_ip; /* 44+4 */
00139     L4_ktrace_t__Mword lock_cnt; /* 48+4 */
00140     L4_ktrace_t__Space *from_space; /* 52+4 */
00141     L4_ktrace_t__Sched_context *from_sched; /* 56+4 */
00142     L4_ktrace_t__Mword from_prio; /* 60+4 */
00143 } context_switch; /* 64 */
00144 struct __attribute__((__packed__))
00145 {
00146 } empty; /* 40 */
00147 struct __attribute__((__packed__))
00148 {
00149     char __pre_pad[2];
00150     L4_ktrace_t__L4_msg_tag _tag; /* 36+4 */
00151     L4_ktrace_t__Mword _dword[2]; /* 40+8 */
00152     L4_ktrace_t__L4_obj_ref _dst; /* 48+4 */
00153     L4_ktrace_t__Mword _dbg_id; /* 52+4 */
00154     L4_ktrace_t__Mword _label; /* 56+4 */
00155     L4_ktrace_t__L4_timeout_pair _timeout; /* 60+4 */
00156 } ipc; /* 64 */
00157 struct __attribute__((__packed__))
00158 {
00159     L4_ktrace_t__Unsigned8 _have_snd; /* 34+1 */
00160     L4_ktrace_t__Unsigned8 _is_np; /* 35+1 */
00161     L4_ktrace_t__L4_msg_tag _tag; /* 36+4 */
00162     L4_ktrace_t__Mword _dword[2]; /* 40+8 */
00163     L4_ktrace_t__L4_error _result; /* 48+4 */
00164     L4_ktrace_t__Mword _from; /* 52+4 */
00165     L4_ktrace_t__L4_obj_ref _dst; /* 56+4 */
00166     L4_ktrace_t__Mword _pair_event; /* 60+4 */
00167 } ipc_res; /* 64 */
00168 struct __attribute__((__packed__))
00169 {
00170     char __pre_pad[2];
00171     union __attribute__((__packed__)) {
00172         char msg[24]; /* 0+24 */
00173         struct __attribute__((__packed__)) {
00174             char tag[2]; /* 0+2 */
00175             char __pad_1[2];
00176             char *ptr; /* 4+4 */
00177         } mptr; /* 0+8 */
00178     } msg; /* 36+24 */
00179 } ke; /* 64 */
00180 struct __attribute__((__packed__))
00181 {
00182     char _msg[24]; /* 34+24 */
00183 } ke_bin; /* 64 */
00184 struct __attribute__((__packed__))
00185 {
00186     char __pre_pad[2];
00187     L4_ktrace_t__Mword v[3]; /* 36+12 */
00188     union __attribute__((__packed__)) {
00189         char msg[12]; /* 0+12 */
00190         struct __attribute__((__packed__)) {
00191             char tag[2]; /* 0+2 */
00192             char __pad_1[2];
00193             char *ptr; /* 4+4 */
00194         } mptr; /* 0+8 */
00195     } msg; /* 48+12 */
00196 } ke_reg; /* 64 */
00197 struct __attribute__((__packed__))

```

```

00198     {
00199         char __pre_pad[2];
00200         L4_ktrace_t__Address _pfa; /* 36+4 */
00201         L4_ktrace_t__Mword _error; /* 40+4 */
00202         L4_ktrace_t__Space *_space; /* 44+4 */
00203     } pf; /* 48 */
00204     struct __attribute__((__packed__))
00205     {
00206         unsigned short mode; /* 34+2 */
00207         L4_ktrace_t__Context *owner; /* 36+4 */
00208         unsigned short id; /* 40+2 */
00209         unsigned short prio; /* 42+2 */
00210         long left; /* 44+4 */
00211         unsigned long quantum; /* 48+4 */
00212     } sched; /* 56 */
00213     struct __attribute__((__packed__))
00214     {
00215         char __pre_pad[2];
00216         L4_ktrace_t__Unsigned32 _cause; /* 36+4 */
00217         L4_ktrace_t__Mword _sp; /* 40+4 */
00218     } trap; /* 48 */
00219     struct __attribute__((__packed__))
00220     {
00221         char _padding[24]; /* 34+24 */
00222         char __post_pad[6]; /* 58+6 */
00223     } fullsize; /* 64 */
00224     struct __attribute__((__packed__))
00225     {
00226         char __pre_pad[2];
00227         L4_ktrace_t__Cap_index cap_idx; /* 36+4 */
00228     } ieh; /* 40 */
00229     struct __attribute__((__packed__))
00230     {
00231         char __pre_pad[2];
00232         L4_ktrace_t__Mword pfa; /* 36+4 */
00233         L4_ktrace_t__Cap_index cap_idx; /* 40+4 */
00234         L4_ktrace_t__Mword err; /* 44+4 */
00235     } ipfh; /* 48 */
00236     struct __attribute__((__packed__))
00237     {
00238         char __pre_pad[2];
00239         L4_ktrace_t__Mword id; /* 36+4 */
00240         L4_ktrace_t__Mword ip; /* 40+4 */
00241         L4_ktrace_t__Mword sp; /* 44+4 */
00242         L4_ktrace_t__Mword op; /* 48+4 */
00243     } exregs; /* 56 */
00244     struct __attribute__((__packed__))
00245     {
00246         char __pre_pad[2];
00247         L4_ktrace_t__Mword state; /* 36+4 */
00248         L4_ktrace_t__Address user_ip; /* 40+4 */
00249         L4_ktrace_t__Cpu_number src_cpu; /* 44+4 */
00250         L4_ktrace_t__Cpu_number target_cpu; /* 48+4 */
00251     } migration; /* 56 */
00252     struct __attribute__((__packed__))
00253     {
00254         char __pre_pad[2];
00255         L4_ktrace_t__Address user_ip; /* 36+4 */
00256     } timer; /* 40 */
00257 } m;
00258 } l4_tracebuffer_entry_t;
00259 #else
00260 typedef struct __attribute__((packed))
00261 {
00262     L4_ktrace_t__Mword _number; /* 0+8 */
00263     L4_ktrace_t__Address _ip; /* 8+8 */
00264     L4_ktrace_t__Unsigned64 _tsc; /* 16+8 */
00265     L4_ktrace_t__Context *_ctx; /* 24+8 */
00266     L4_ktrace_t__Unsigned32 _pmc1; /* 32+4 */
00267     L4_ktrace_t__Unsigned32 _pmc2; /* 36+4 */
00268     L4_ktrace_t__Unsigned32 _kclock; /* 40+4 */
00269     L4_ktrace_t__Unsigned8 _type; /* 44+1 */
00270     L4_ktrace_t__Unsigned8 _cpu; /* 45+1 */
00271     union __attribute__((__packed__))
00272     {
00273         struct __attribute__((__packed__))
00274         {
00275             char __pre_pad[2];
00276             void *func; /* 48+8 */
00277             L4_ktrace_t__Context *thread; /* 56+8 */
00278             L4_ktrace_t__Context__Drq *rq; /* 64+8 */
00279             L4_ktrace_t__Cpu_number target_cpu; /* 72+4 */
00280             L4_ktrace_t__Context__Drq_log_Type type; /* 76+4 */
00281             char wait; /* 80+1 */
00282         } drq; /* 88 */
00283     } __attribute__((__packed__))
00284     {

```

```

00285     char __pre_pad[2];
00286     L4_ktrace_t_Mword state; /* 48+8 */
00287     L4_ktrace_t_Mword ip; /* 56+8 */
00288     L4_ktrace_t_Mword sp; /* 64+8 */
00289     L4_ktrace_t_Mword space; /* 72+8 */
00290     L4_ktrace_t_Mword err; /* 80+8 */
00291     unsigned char type; /* 88+1 */
00292     unsigned char trap; /* 89+1 */
00293 } vcpu; /* 96 */
00294 struct __attribute__((__packed__))
00295 {
00296     char __pre_pad[2];
00297     L4_ktrace_t_Smword op; /* 48+8 */
00298     L4_ktrace_t_Cap_index buffer; /* 56+8 */
00299     L4_ktrace_t_Mword id; /* 64+8 */
00300     L4_ktrace_t_Mword ram; /* 72+8 */
00301     L4_ktrace_t_Mword newo; /* 80+8 */
00302 } factory; /* 88 */
00303 struct __attribute__((__packed__))
00304 {
00305     char __pre_pad[2];
00306     L4_ktrace_t_Mword gate_dbg_id; /* 48+8 */
00307     L4_ktrace_t_Mword thread_dbg_id; /* 56+8 */
00308     L4_ktrace_t_Mword label; /* 64+8 */
00309 } gate; /* 72 */
00310 struct __attribute__((__packed__))
00311 {
00312     char __pre_pad[2];
00313     L4_ktrace_t_Irq_base *obj; /* 48+8 */
00314     L4_ktrace_t_Irq_chip *chip; /* 56+8 */
00315     L4_ktrace_t_Mword pin; /* 64+8 */
00316 } irq; /* 72 */
00317 struct __attribute__((__packed__))
00318 {
00319     char __pre_pad[2];
00320     L4_ktrace_t_Kobject *obj; /* 48+8 */
00321     L4_ktrace_t_Mword id; /* 56+8 */
00322     L4_ktrace_t_cxx_Type_info *type; /* 64+8 */
00323     L4_ktrace_t_Mword ram; /* 72+8 */
00324 } destroy; /* 80 */
00325 struct __attribute__((__packed__))
00326 {
00327     char __pre_pad[2];
00328     L4_ktrace_t_Cpu_number cpu; /* 48+4 */
00329     char __pad_l[4];
00330     L4_ktrace_t_Rcu_item *item; /* 56+8 */
00331     void *cb; /* 64+8 */
00332     unsigned char event; /* 72+1 */
00333 } rcu; /* 80 */
00334 struct __attribute__((__packed__))
00335 {
00336     char __pre_pad[2];
00337     L4_ktrace_t_Mword id; /* 48+8 */
00338     L4_ktrace_t_Mword mask; /* 56+8 */
00339     L4_ktrace_t_Mword fpage; /* 64+8 */
00340     char map; /* 72+1 */
00341 } tmap; /* 80 */
00342 struct __attribute__((__packed__))
00343 {
00344     char __pre_pad[2];
00345     L4_ktrace_t_Address _address; /* 48+8 */
00346     int _len; /* 56+4 */
00347     char __pad_l[4];
00348     L4_ktrace_t_Mword _value; /* 64+8 */
00349     int _mode; /* 72+4 */
00350 } bp; /* 80 */
00351 struct __attribute__((__packed__))
00352 {
00353     char __pre_pad[2];
00354     L4_ktrace_t_Context *dst; /* 48+8 */
00355     L4_ktrace_t_Context *dst_orig; /* 56+8 */
00356     L4_ktrace_t_Address kernel_ip; /* 64+8 */
00357     L4_ktrace_t_Mword lock_cnt; /* 72+8 */
00358     L4_ktrace_t_Space *from_space; /* 80+8 */
00359     L4_ktrace_t_Sched_context *from_sched; /* 88+8 */
00360     L4_ktrace_t_Mword from_prio; /* 96+8 */
00361 } context_switch; /* 104 */
00362 struct __attribute__((__packed__))
00363 {
00364 } empty; /* 48 */
00365 struct __attribute__((__packed__))
00366 {
00367     char __pre_pad[2];
00368     L4_ktrace_t_L4_msg_tag _tag; /* 48+8 */
00369     L4_ktrace_t_Mword _dword[2]; /* 56+16 */
00370     L4_ktrace_t_L4_obj_ref _dst; /* 72+8 */
00371     L4_ktrace_t_Mword _dbg_id; /* 80+8 */

```

```

00372     L4_ktrace_t__Mword _label; /* 88+8 */
00373     L4_ktrace_t__L4_timeout_pair _timeout; /* 96+4 */
00374     char __pad_l[4];
00375     L4_ktrace_t__Unsigned64 _to_abs_rcv; /* 104+8 */
00376 } ipc; /* 112 */
00377 struct __attribute__((__packed__))
00378 {
00379     L4_ktrace_t__Unsigned8 _have_snd; /* 46+1 */
00380     L4_ktrace_t__Unsigned8 _is_np; /* 47+1 */
00381     L4_ktrace_t__L4_msg_tag _tag; /* 48+8 */
00382     L4_ktrace_t__Mword _dword[2]; /* 56+16 */
00383     L4_ktrace_t__L4_error_result; /* 72+8 */
00384     L4_ktrace_t__Mword _from; /* 80+8 */
00385     L4_ktrace_t__L4_obj_ref _dst; /* 88+8 */
00386     L4_ktrace_t__Mword _pair_event; /* 96+8 */
00387 } ipc_res; /* 104 */
00388 struct __attribute__((__packed__))
00389 {
00390     char __pre_pad[2];
00391     union __attribute__((__packed__)) {
00392         char msg[80]; /* 0+80 */
00393         struct __attribute__((__packed__)) {
00394             char tag[2]; /* 0+2 */
00395             char __pad_l[6];
00396             char *ptr; /* 8+8 */
00397         } mptr; /* 0+16 */
00398     } msg; /* 48+80 */
00399 } ke; /* 128 */
00400 struct __attribute__((__packed__))
00401 {
00402     char _msg[80]; /* 46+80 */
00403 } ke_bin; /* 128 */
00404 struct __attribute__((__packed__))
00405 {
00406     char __pre_pad[2];
00407     L4_ktrace_t__Mword v[3]; /* 48+24 */
00408     union __attribute__((__packed__)) {
00409         char msg[56]; /* 0+56 */
00410         struct __attribute__((__packed__)) {
00411             char tag[2]; /* 0+2 */
00412             char __pad_l[6];
00413             char *ptr; /* 8+8 */
00414         } mptr; /* 0+16 */
00415     } msg; /* 72+56 */
00416 } ke_reg; /* 128 */
00417 struct __attribute__((__packed__))
00418 {
00419     char __pre_pad[2];
00420     L4_ktrace_t__Address _pfa; /* 48+8 */
00421     L4_ktrace_t__Mword _error; /* 56+8 */
00422     L4_ktrace_t__Space *_space; /* 64+8 */
00423 } pf; /* 72 */
00424 struct __attribute__((__packed__))
00425 {
00426     unsigned short mode; /* 46+2 */
00427     L4_ktrace_t__Context *owner; /* 48+8 */
00428     unsigned short id; /* 56+2 */
00429     unsigned short prio; /* 58+2 */
00430     char __pad_l[4];
00431     long left; /* 64+8 */
00432     unsigned long quantum; /* 72+8 */
00433 } sched; /* 80 */
00434 struct __attribute__((__packed__))
00435 {
00436     char __pre_pad[2];
00437     L4_ktrace_t__Unsigned32 _cause; /* 48+4 */
00438     char __pad_l[4];
00439     L4_ktrace_t__Mword _sp; /* 56+8 */
00440 } trap; /* 64 */
00441 struct __attribute__((__packed__))
00442 {
00443     char _padding[80]; /* 46+80 */
00444     char __post_pad[2]; /* 126+2 */
00445 } fullsize; /* 128 */
00446 struct __attribute__((__packed__))
00447 {
00448     char __pre_pad[2];
00449     L4_ktrace_t__Cap_index cap_idx; /* 48+8 */
00450 } ieh; /* 56 */
00451 struct __attribute__((__packed__))
00452 {
00453     char __pre_pad[2];
00454     L4_ktrace_t__Mword pfa; /* 48+8 */
00455     L4_ktrace_t__Cap_index cap_idx; /* 56+8 */
00456     L4_ktrace_t__Mword err; /* 64+8 */
00457 } ipfh; /* 72 */
00458 struct __attribute__((__packed__))

```

```

00459     {
00460         char __pre_pad[2];
00461         L4_ktrace_t__Mword id; /* 48+8 */
00462         L4_ktrace_t__Mword ip; /* 56+8 */
00463         L4_ktrace_t__Mword sp; /* 64+8 */
00464         L4_ktrace_t__Mword op; /* 72+8 */
00465     } exregs; /* 80 */
00466     struct __attribute__((__packed__))
00467     {
00468         char __pre_pad[2];
00469         L4_ktrace_t__Mword state; /* 48+8 */
00470         L4_ktrace_t__Address user_ip; /* 56+8 */
00471         L4_ktrace_t__Cpu_number src_cpu; /* 64+4 */
00472         L4_ktrace_t__Cpu_number target_cpu; /* 68+4 */
00473     } migration; /* 72 */
00474     struct __attribute__((__packed__))
00475     {
00476         char __pre_pad[2];
00477         L4_ktrace_t__Address user_ip; /* 48+8 */
00478     } timer; /* 56 */
00479     } m;
00480 } l4_tracebuffer_entry_t;
00481 #endif

```

16.110 ktrace_events.h

```

00001 /* Note, automatically generated from Fiasco binary */
00002 #pragma once
00003
00004 enum L4_ktrace_tbuf_entry_fixed
00005 {
00006     l4_ktrace_tbuf_unused = 0,
00007     l4_ktrace_tbuf_pf = 1,
00008     l4_ktrace_tbuf_ipc = 2,
00009     l4_ktrace_tbuf_ipc_res = 3,
00010     l4_ktrace_tbuf_ipc_trace = 4,
00011     l4_ktrace_tbuf_ke = 5,
00012     l4_ktrace_tbuf_ke_reg = 6,
00013     l4_ktrace_tbuf_breakpoint = 7,
00014     l4_ktrace_tbuf_ke_bin = 8,
00015     l4_ktrace_tbuf_dynentries = 9,
00016     l4_ktrace_tbuf_max = 128,
00017     l4_ktrace_tbuf_hidden = 128,
00018 };
00019
00020 typedef unsigned long L4_ktrace_t__Address;
00021 typedef unsigned long L4_ktrace_t__Cap_index;
00022 typedef void L4_ktrace_t__Context;
00023 typedef void L4_ktrace_t__Context__Drq;
00024 typedef unsigned L4_ktrace_t__Context__Drq_log__Type;
00025 typedef unsigned L4_ktrace_t__Cpu_number;
00026 typedef void L4_ktrace_t__Irq_base;
00027 typedef void L4_ktrace_t__Irq_chip;
00028 typedef void L4_ktrace_t__Kobject;
00029 typedef unsigned long L4_ktrace_t__L4_error;
00030 typedef unsigned long L4_ktrace_t__L4_msg_tag;
00031 typedef unsigned long L4_ktrace_t__L4_obj_ref;
00032 typedef unsigned L4_ktrace_t__L4_timeout_pair;
00033 typedef unsigned long L4_ktrace_t__Mword;
00034 typedef void L4_ktrace_t__Rcu_item;
00035 typedef void L4_ktrace_t__Sched_context;
00036 typedef long L4_ktrace_t__Smword;
00037 typedef void L4_ktrace_t__Space;
00038 typedef unsigned short L4_ktrace_t__Unsigned16;
00039 typedef unsigned int L4_ktrace_t__Unsigned32;
00040 typedef unsigned long long L4_ktrace_t__Unsigned64;
00041 typedef unsigned char L4_ktrace_t__Unsigned8;
00042 typedef void L4_ktrace_t__cxx__Type_info;
00043
00044 typedef struct __attribute__((packed))
00045 {
00046     L4_ktrace_t__Mword _number; /* 0+4 */
00047     L4_ktrace_t__Address _ip; /* 4+4 */
00048     L4_ktrace_t__Unsigned64 _tsc; /* 8+8 */
00049     L4_ktrace_t__Context *_ctx; /* 16+4 */
00050     L4_ktrace_t__Unsigned32 _pmc1; /* 20+4 */
00051     L4_ktrace_t__Unsigned32 _pmc2; /* 24+4 */
00052     L4_ktrace_t__Unsigned32 _kclock; /* 28+4 */
00053     L4_ktrace_t__Unsigned8 _type; /* 32+1 */
00054     L4_ktrace_t__Unsigned8 _cpu; /* 33+1 */
00055     union __attribute__((__packed__))
00056     {
00057         struct __attribute__((__packed__))

```

```

00058     {
00059         char __pre_pad[2];
00060         void *func; /* 36+4 */
00061         L4_ktrace_t__Context *thread; /* 40+4 */
00062         L4_ktrace_t__Context__Drq *rq; /* 44+4 */
00063         L4_ktrace_t__Cpu_number target_cpu; /* 48+4 */
00064         L4_ktrace_t__Context__Drq_log__Type type; /* 52+4 */
00065         char wait; /* 56+1 */
00066     } drq; /* 64 */
00067     struct __attribute__((__packed__))
00068     {
00069         char __pre_pad[2];
00070         L4_ktrace_t__Mword state; /* 36+4 */
00071         L4_ktrace_t__Mword ip; /* 40+4 */
00072         L4_ktrace_t__Mword sp; /* 44+4 */
00073         L4_ktrace_t__Mword space; /* 48+4 */
00074         L4_ktrace_t__Mword err; /* 52+4 */
00075         unsigned char type; /* 56+1 */
00076         unsigned char trap; /* 57+1 */
00077     } vcpu; /* 64 */
00078     struct __attribute__((__packed__))
00079     {
00080         char __pre_pad[2];
00081         L4_ktrace_t__Sword op; /* 36+4 */
00082         L4_ktrace_t__Cap_index buffer; /* 40+4 */
00083         L4_ktrace_t__Mword id; /* 44+4 */
00084         L4_ktrace_t__Mword ram; /* 48+4 */
00085         L4_ktrace_t__Mword newo; /* 52+4 */
00086     } factory; /* 56 */
00087     struct __attribute__((__packed__))
00088     {
00089         char __pre_pad[2];
00090         L4_ktrace_t__Mword gate_dbg_id; /* 36+4 */
00091         L4_ktrace_t__Mword thread_dbg_id; /* 40+4 */
00092         L4_ktrace_t__Mword label; /* 44+4 */
00093     } gate; /* 48 */
00094     struct __attribute__((__packed__))
00095     {
00096         char __pre_pad[2];
00097         L4_ktrace_t__Irq_base *obj; /* 36+4 */
00098         L4_ktrace_t__Irq_chip *chip; /* 40+4 */
00099         L4_ktrace_t__Mword pin; /* 44+4 */
00100     } irq; /* 48 */
00101     struct __attribute__((__packed__))
00102     {
00103         char __pre_pad[2];
00104         L4_ktrace_t__Kobject *obj; /* 36+4 */
00105         L4_ktrace_t__Mword id; /* 40+4 */
00106         L4_ktrace_t__cxx__Type_info *type; /* 44+4 */
00107         L4_ktrace_t__Mword ram; /* 48+4 */
00108     } destroy; /* 56 */
00109     struct __attribute__((__packed__))
00110     {
00111         char __pre_pad[2];
00112         L4_ktrace_t__Cpu_number cpu; /* 36+4 */
00113         L4_ktrace_t__Rcu_item *item; /* 40+4 */
00114         void *cb; /* 44+4 */
00115         unsigned char event; /* 48+1 */
00116     } rcu; /* 56 */
00117     struct __attribute__((__packed__))
00118     {
00119         char __pre_pad[2];
00120         L4_ktrace_t__Mword id; /* 36+4 */
00121         L4_ktrace_t__Mword mask; /* 40+4 */
00122         L4_ktrace_t__Mword fpage; /* 44+4 */
00123         char map; /* 48+1 */
00124     } tmap; /* 56 */
00125     struct __attribute__((__packed__))
00126     {
00127         char __pre_pad[2];
00128         L4_ktrace_t__Address _address; /* 36+4 */
00129         int _len; /* 40+4 */
00130         L4_ktrace_t__Mword _value; /* 44+4 */
00131         int _mode; /* 48+4 */
00132     } bp; /* 56 */
00133     struct __attribute__((__packed__))
00134     {
00135         char __pre_pad[2];
00136         L4_ktrace_t__Context *dst; /* 36+4 */
00137         L4_ktrace_t__Context *dst_orig; /* 40+4 */
00138         L4_ktrace_t__Address kernel_ip; /* 44+4 */
00139         L4_ktrace_t__Mword lock_cnt; /* 48+4 */
00140         L4_ktrace_t__Space *from_space; /* 52+4 */
00141         L4_ktrace_t__Sched_context *from_sched; /* 56+4 */
00142         L4_ktrace_t__Mword from_prio; /* 60+4 */
00143     } context_switch; /* 64 */
00144     struct __attribute__((__packed__))

```

```

00145     {
00146     } empty; /* 40 */
00147     struct __attribute__((__packed__))
00148     {
00149         char __pre_pad[2];
00150         L4_ktrace_t__L4_msg_tag_tag; /* 36+4 */
00151         L4_ktrace_t__Mword_dword[2]; /* 40+8 */
00152         L4_ktrace_t__L4_obj_ref_dst; /* 48+4 */
00153         L4_ktrace_t__Mword_dbg_id; /* 52+4 */
00154         L4_ktrace_t__Mword_label; /* 56+4 */
00155         L4_ktrace_t__L4_timeout_pair_timeout; /* 60+4 */
00156     } ipc; /* 64 */
00157     struct __attribute__((__packed__))
00158     {
00159         L4_ktrace_t__Unsigned8_have_snd; /* 34+1 */
00160         L4_ktrace_t__Unsigned8_is_np; /* 35+1 */
00161         L4_ktrace_t__L4_msg_tag_tag; /* 36+4 */
00162         L4_ktrace_t__Mword_dword[2]; /* 40+8 */
00163         L4_ktrace_t__L4_error_result; /* 48+4 */
00164         L4_ktrace_t__Mword_from; /* 52+4 */
00165         L4_ktrace_t__L4_obj_ref_dst; /* 56+4 */
00166         L4_ktrace_t__Mword_pair_event; /* 60+4 */
00167     } ipc_res; /* 64 */
00168     struct __attribute__((__packed__))
00169     {
00170         char __pre_pad[2];
00171         union __attribute__((__packed__)) {
00172             char msg[24]; /* 0+24 */
00173             struct __attribute__((__packed__)) {
00174                 char tag[2]; /* 0+2 */
00175                 char __pad_1[2];
00176                 char *ptr; /* 4+4 */
00177             } mptr; /* 0+8 */
00178         } msg; /* 36+24 */
00179     } ke; /* 64 */
00180     struct __attribute__((__packed__))
00181     {
00182         char _msg[24]; /* 34+24 */
00183     } ke_bin; /* 64 */
00184     struct __attribute__((__packed__))
00185     {
00186         char __pre_pad[2];
00187         L4_ktrace_t__Mword v[3]; /* 36+12 */
00188         union __attribute__((__packed__)) {
00189             char msg[12]; /* 0+12 */
00190             struct __attribute__((__packed__)) {
00191                 char tag[2]; /* 0+2 */
00192                 char __pad_1[2];
00193                 char *ptr; /* 4+4 */
00194             } mptr; /* 0+8 */
00195         } msg; /* 48+12 */
00196     } ke_reg; /* 64 */
00197     struct __attribute__((__packed__))
00198     {
00199         char __pre_pad[2];
00200         L4_ktrace_t__Address_pfa; /* 36+4 */
00201         L4_ktrace_t__Mword_error; /* 40+4 */
00202         L4_ktrace_t__Space *space; /* 44+4 */
00203     } pf; /* 48 */
00204     struct __attribute__((__packed__))
00205     {
00206         unsigned short mode; /* 34+2 */
00207         L4_ktrace_t__Context *owner; /* 36+4 */
00208         unsigned short id; /* 40+2 */
00209         unsigned short prio; /* 42+2 */
00210         long left; /* 44+4 */
00211         unsigned long quantum; /* 48+4 */
00212     } sched; /* 56 */
00213     struct __attribute__((__packed__))
00214     {
00215         L4_ktrace_t__Unsigned8_trapno; /* 34+1 */
00216         char __pad_1[1];
00217         L4_ktrace_t__Unsigned16_error; /* 36+2 */
00218         char __pad_2[2];
00219         L4_ktrace_t__Mword_ebp; /* 40+4 */
00220         L4_ktrace_t__Mword_cr2; /* 44+4 */
00221         L4_ktrace_t__Mword_eax; /* 48+4 */
00222         L4_ktrace_t__Mword_eflags; /* 52+4 */
00223         L4_ktrace_t__Mword_esp; /* 56+4 */
00224         L4_ktrace_t__Unsigned16_cs; /* 60+2 */
00225         L4_ktrace_t__Unsigned16_ds; /* 62+2 */
00226     } trap; /* 64 */
00227     struct __attribute__((__packed__))
00228     {
00229         char __padding[24]; /* 34+24 */
00230         char __post_pad[6]; /* 58+6 */
00231     } fullsize; /* 64 */

```

```

00232     struct __attribute__((__packed__))
00233     {
00234         char __pre_pad[2];
00235         L4_ktrace_t__Cap_index cap_idx; /* 36+4 */
00236     } ieh; /* 40 */
00237     struct __attribute__((__packed__))
00238     {
00239         char __pre_pad[2];
00240         L4_ktrace_t__Mword pfa; /* 36+4 */
00241         L4_ktrace_t__Cap_index cap_idx; /* 40+4 */
00242         L4_ktrace_t__Mword err; /* 44+4 */
00243     } ipfh; /* 48 */
00244     struct __attribute__((__packed__))
00245     {
00246         char __pre_pad[2];
00247         L4_ktrace_t__Mword id; /* 36+4 */
00248         L4_ktrace_t__Mword ip; /* 40+4 */
00249         L4_ktrace_t__Mword sp; /* 44+4 */
00250         L4_ktrace_t__Mword op; /* 48+4 */
00251     } exregs; /* 56 */
00252     struct __attribute__((__packed__))
00253     {
00254         char __pre_pad[2];
00255         L4_ktrace_t__Mword state; /* 36+4 */
00256         L4_ktrace_t__Address user_ip; /* 40+4 */
00257         L4_ktrace_t__Cpu_number src_cpu; /* 44+4 */
00258         L4_ktrace_t__Cpu_number target_cpu; /* 48+4 */
00259     } migration; /* 56 */
00260     struct __attribute__((__packed__))
00261     {
00262         char __pre_pad[2];
00263         L4_ktrace_t__Address user_ip; /* 36+4 */
00264     } timer; /* 40 */
00265     struct __attribute__((__packed__))
00266     {
00267         char __pre_pad[2];
00268         L4_ktrace_t__Mword exitcode; /* 36+4 */
00269         L4_ktrace_t__Mword exitinfo1; /* 40+4 */
00270         L4_ktrace_t__Mword exitinfo2; /* 44+4 */
00271         L4_ktrace_t__Mword rip; /* 48+4 */
00272     } svm; /* 56 */
00273     } m;
00274 } l4_tracebuffer_entry_t;

```

16.111 amd64/l4/sys/linkage.h File Reference

Linkage.

Macros

- **#define L4_CV**
Define calling convention.

16.111.1 Detailed Description

Linkage.

Definition in file [linkage.h](#).

16.112 linkage.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014 #ifndef __L4__SYS__ARCH_AMD64__LINKAGE_H__
00015 #define __L4__SYS__ARCH_AMD64__LINKAGE_H__
00016
00017 #ifdef __ASSEMBLY__
00018
00019 #ifndef ENTRY
00020 #define ENTRY(name) \
00021     .globl name; \
00022     .p2align(2); \
00023     name:
00024
00025 #endif /* __ASSEMBLY__ */
00026 #endif /* ! ENTRY */
00027
00028 #define L4_FASTCALL(x)      x
00029 #define l4_fastcall
00030
00036 #define L4_CV
00037
00038 #endif /* ! __L4__SYS__ARCH_AMD64__LINKAGE_H__ */

```

16.113 arm/l4/sys/linkage.h File Reference

Linkage.

Macros

- **#define L4_CV**
Define calling convention.

16.113.1 Detailed Description

Linkage.

Definition in file [linkage.h](#).

16.114 linkage.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013 #ifndef __L4__SYS__ARCH_ARM__LINKAGE_H__
00014 #define __L4__SYS__ARCH_ARM__LINKAGE_H__

```

```

00015
00016 #ifdef __ASSEMBLY__
00017 #ifndef ENTRY
00018 #define ENTRY(name) \
00019     .globl name; \
00020     .p2align(2); \
00021     name:
00022 #endif
00023 #endif
00024
00025 #define L4_FASTCALL(x)  x
00026 #define l4_fastcall
00027
00033 #define L4_CV
00034
00035 #ifdef __PIC__
00036 # define L4_LONG_CALL
00037 #else
00038 # define L4_LONG_CALL __attribute__((long_call))
00039 #endif
00040
00041 #endif /* ! __L4__SYS__ARCH_ARM__LINKAGE_H__ */

```

16.115 linkage.h

```

00001 #pragma once
00002
00008 #define L4_CV
00009
00010 #define L4_FASTCALL(x)  x
00011 #define l4_fastcall

```

16.116 riscv/l4/sys/linkage.h File Reference

Linkage.

Macros

- **#define L4_CV**
Define calling convention.

16.116.1 Detailed Description

Linkage.

Definition in file [linkage.h](#).

16.117 linkage.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * Copyright (C) 2021, 2024-2025 Kernkonzept GmbH.
00008  * Author(s): Georg Kotheimer <georg.kotheimer@kernkonzept.com>
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #pragma once
00013
00014 #ifdef __ASSEMBLY__

```

```

00015
00016 #ifndef ENTRY
00017 #define ENTRY(name) \
00018     .globl name; \
00019     .p2align(2); \
00020     name:
00021
00022 #endif /* ! ENTRY */
00023 #endif /* __ASSEMBLY__ */
00024
00030 #define L4_CV
00031
00032 #define L4_FASTCALL(x)  x
00033 #define l4_fastcall

```

16.118 x86/l4/sys/linkage.h File Reference

Linkage.

Macros

- **#define L4_CV**
Define calling convention.

16.118.1 Detailed Description

Linkage.

Definition in file [linkage.h](#).

16.119 linkage.h

[Go to the documentation of this file.](#)

```

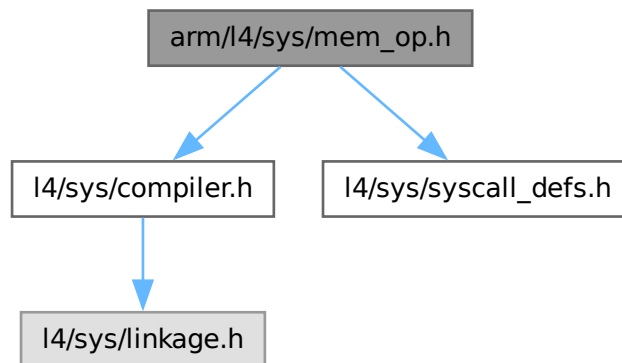
00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *      Frank Mehnert <fm3@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014 #ifndef __L4__SYS__ARCH_X86__LINKAGE_H__
00015 #define __L4__SYS__ARCH_X86__LINKAGE_H__
00016
00017 #ifdef __ASSEMBLY__
00018
00019 #ifndef ENTRY
00020 #define ENTRY(name) \
00021     .globl name; \
00022     .p2align(2); \
00023     name:
00024
00025 #endif /* ! ENTRY */
00026 #endif /* __ASSEMBLY__ */
00027
00028 #define L4_FASTCALL(x)  x __attribute__((regparm(3)))
00029 #define l4_fastcall __attribute__((regparm(3)))
00030
00036 #define L4_CV  __attribute__((regparm(0)))
00037
00038 #endif /* ! __L4__SYS__ARCH_X86__LINKAGE_H__ */

```

16.120 arm/l4/sys/mem_op.h File Reference

Memory access functions (ARM specific).

```
#include <l4/sys/compiler.h>
#include <l4/sys/syscall_defs.h>
Include dependency graph for mem_op.h:
```



Enumerations

- enum `L4_mem_op_widths` { `L4_MEM_WIDTH_1BYTE` = 0 , `L4_MEM_WIDTH_2BYTE` = 1 , `L4_MEM_WIDTH_4BYTE` = 2 }
- Memory access width definitions.*

Functions

- unsigned long `l4_mem_read` (unsigned long virtaddress, unsigned width)
Read user task memory from kernel privilege level.
- void `l4_mem_write` (unsigned long virtaddress, unsigned width, unsigned long value)
Write user task memory from kernel privilege level.
- unsigned long `l4_mem_arm_op_call` (unsigned long op, unsigned long va, unsigned long width, unsigned long value)
Implementations.

16.120.1 Detailed Description

Memory access functions (ARM specific).

Date

2010-10

Author

Adam Lackorzynski adam@os.inf.tu-dresden.de

Definition in file [mem_op.h](#).

16.121 mem_op.h

[Go to the documentation of this file.](#)

```

00001
00009 /*
00010  * (c) 2010 Author(s)
00011  *     economic rights: Technische Universität Dresden (Germany)
00012  *
00013  * License: see LICENSE.spdx (in this directory or the directories above)
00014  */
00015 #ifndef __L4SYS__INCLUDE__ARCH_ARM__MEM_OP_H__
00016 #define __L4SYS__INCLUDE__ARCH_ARM__MEM_OP_H__
00017
00018 #include <l4/sys/compiler.h>
00019 #include <l4/sys/syscall_defs.h>
00020
00021 L4_BEGIN_DECLS
00022
00023
00035
00040 enum L4_mem_op_widths
00041 {
00042     L4_MEM_WIDTH_1BYTE = 0,
00043     L4_MEM_WIDTH_2BYTE = 1,
00044     L4_MEM_WIDTH_4BYTE = 2,
00045 };
00046
00059 L4_INLINE unsigned long
00060 l4_mem_read(unsigned long virtaddress, unsigned width);
00061
00074 L4_INLINE void
00075 l4_mem_write(unsigned long virtaddress, unsigned width,
00076              unsigned long value);
00077
00078 enum L4_mem_ops
00079 {
00080     L4_MEM_OP_MEM_READ = 0x10,
00081     L4_MEM_OP_MEM_WRITE = 0x11,
00082 };
00083
00087 L4_INLINE unsigned long
00088 l4_mem_arm_op_call(unsigned long op,
00089                    unsigned long va,
00090                    unsigned long width,
00091                    unsigned long value);
00092
00094
00095 L4_INLINE unsigned long
00096 l4_mem_arm_op_call(unsigned long op,
00097                    unsigned long va,
00098                    unsigned long width,
00099                    unsigned long value)
00100 {
00101     register unsigned long _op    __asm__ ("r0") = op;
00102     register unsigned long _va    __asm__ ("r1") = va;
00103     register unsigned long _width __asm__ ("r2") = width;
00104     register unsigned long _value __asm__ ("r3") = value;
00105
00106     __asm__ __volatile__
00107     ("@ l4_cache_op_arm_call(start) \n\t"
00108      "mov     r5, %[sc] \n\t"
00109      "blx     __l4_sys_syscall \n\t"
00110      "@ l4_cache_op_arm_call(end) \n\t"
00111      :
00112      "=r" (_op),
00113      "=r" (_va),
00114      "=r" (_width),
00115      "=r" (_value)
00116      :
00117      [sc] "i" (L4_SYSCALL_MEM_OP),
00118      "0" (_op),
00119      "1" (_va),
00120      "2" (_width),
00121      "3" (_value)
00122      :
00123      "cc", "memory", "r5", "ip", "lr"
00124      );
00125
00126     return _value;
00127 }
00128
00129 L4_INLINE unsigned long
00130 l4_mem_read(unsigned long virtaddress, unsigned width)
00131 {
00132     return l4_mem_arm_op_call(L4_MEM_OP_MEM_READ, virtaddress, width, 0);
00133 }

```

```

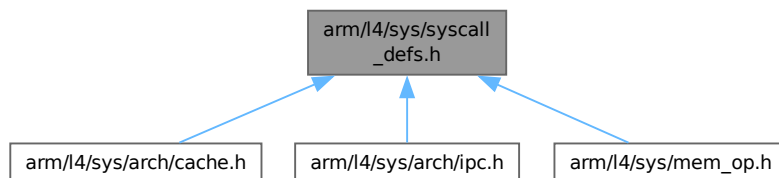
00134
00135 L4_INLINE void
00136 l4_mem_write(unsigned long virtaddress, unsigned width,
00137              unsigned long value)
00138 {
00139     l4_mem_arm_op_call(L4_MEM_OP_MEM_WRITE, virtaddress, width, value);
00140 }
00141
00142 L4_END_DECLS
00143
00144 #endif /* ! __L4SYS__INCLUDE__ARCH_ARM__MEM_OP_H__ */

```

16.122 arm/l4/sys/syscall_defs.h File Reference

Syscall entry definitions.

This graph shows which files directly or indirectly include this file:



16.122.1 Detailed Description

Syscall entry definitions.

Definition in file [syscall_defs.h](#).

16.123 syscall_defs.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011 #ifndef __L4SYS__ARCH_ARM__L4API_L4F__SYSCALL_DEFS_H__
00012 #define __L4SYS__ARCH_ARM__L4API_L4F__SYSCALL_DEFS_H__
00013
00014 #define L4_SYSCALL_INVOKE      (0)
00015 #define L4_SYSCALL_MEM_OP      (1)
00016
00017 #endif /* __L4SYS__ARCH_ARM__L4API_L4F__SYSCALL_DEFS_H__ */

```

16.124 vm.h

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #pragma once
00013
00014 #include <l4/sys/__vm-svm.h>
00015 #include <l4/sys/__vm-vmx.h>

```

16.125 arm/l4/sys/vm.h File Reference

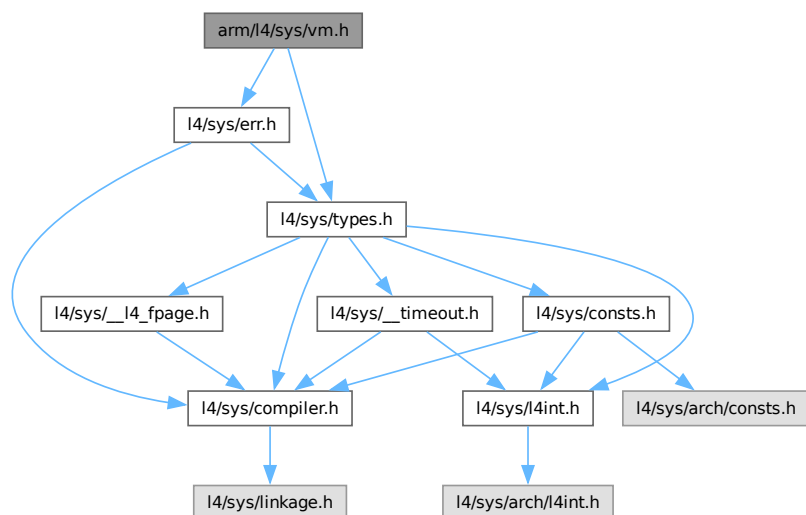
ARM virtualization interface.

```

#include <l4/sys/err.h>
#include <l4/sys/types.h>

```

Include dependency graph for vm.h:



Data Structures

- struct `l4_vm_tz_state`
state structure for TrustZone VMs

16.125.1 Detailed Description

ARM virtualization interface.

Definition in file `vm.h`.

16.126 vm.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #pragma once
00013
00014 #include <l4/sys/err.h>
00015 #include <l4/sys/types.h>
00016
00022
00027 struct l4_vm_tz_state_mode
00028 {
00029     l4_umword_t sp;
00030     l4_umword_t lr;
00031     l4_umword_t spsr;
00032 };
00033
00034 struct l4_vm_tz_state_irq_inject
00035 {
00036     l4_uint32_t group;
00037     l4_uint32_t irqs[8];
00038 };
00039
00044 struct l4_vm_tz_state
00045 {
00046     l4_umword_t r[13]; // r0 - r12
00047
00048     l4_umword_t sp_usr;
00049     l4_umword_t lr_usr;
00050
00051     struct l4_vm_tz_state_mode irq;
00052
00053     l4_umword_t r_fiq[5]; // r8 - r12
00054     struct l4_vm_tz_state_mode fiq;
00055     struct l4_vm_tz_state_mode abt;
00056     struct l4_vm_tz_state_mode und;
00057     struct l4_vm_tz_state_mode svc;
00058
00059     l4_umword_t pc;
00060     l4_umword_t cpsr;
00061
00062     l4_umword_t pending_events;
00063     l4_uint32_t cpacr;
00064     l4_umword_t cpl0_fpexc;
00065
00066     l4_umword_t pfs;
00067     l4_umword_t pfa;
00068     l4_umword_t exit_reason;
00069
00070     struct l4_vm_tz_state_irq_inject irq_inject;
00071 };
00072
00073 enum L4_vm_exit_reason
00074 {
00075     L4_vm_exit_reason_vmm_call    = 1,
00076     L4_vm_exit_reason_inst_abort  = 2,
00077     L4_vm_exit_reason_data_abort  = 3,
00078     L4_vm_exit_reason_irq         = 4,
00079     L4_vm_exit_reason_fiq         = 5,
00080     L4_vm_exit_reason_undef       = 6,
00081 };
00082
00083 L4_INLINE int
00084 l4_vm_tz_irq_inject(struct l4_vm_tz_state *state, unsigned irq);
00085
00086 L4_INLINE int
00087 l4_vm_tz_irq_inject(struct l4_vm_tz_state *state, unsigned irq)
00088 {
00089     if (irq > sizeof(state->irq_inject.irqs) * 8)
00090         return -L4_EINVAL;
00091
00092     unsigned g = irq / 32;
00093     state->irq_inject.group |= 1 << g;
00094     state->irq_inject.irqs[g] |= 1 << (irq & 31);
00095
00096     return 0;
00097 }

```


16.127 vm.h

```
00001
00002
```

16.128 vm.h

```
00001 /*
00002  * Copyright (C) 2024 Kernkonzept GmbH.
00003  * Author(s): Georg Kotheimer <georg.kotheimer@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include <l4/sys/vcpu.h>
00010
00011 enum
00012 {
00014     L4_vm_hstatus_vsbe      = 1UL < 5,
00016     L4_vm_hstatus_gva       = 1UL < 6,
00018     L4_vm_hstatus_spvp      = 1UL < 8,
00020     L4_vm_hstatus_vtw       = 1UL < 21,
00021
00022     // Controls the effective XLEN for VS-mode, only available on RV64 and only if
00023     // supported by the hardware.
00024     L4_vm_hstatus_vsxl_32   = 1UL,
00025     L4_vm_hstatus_vsxl_64   = 2UL,
00026     L4_vm_hstatus_vsxl_128  = 3UL,
00027     L4_vm_hstatus_vsxl_shift = 32,
00028 };
00029
00030 typedef enum L4_vm_rfnc
00031 {
00032     L4_vm_rfnc_none          = 0,
00033     L4_vm_rfnc_fence_i       = 1,
00034     L4_vm_rfnc_sfence_vma    = 2,
00035     L4_vm_rfnc_sfence_vma_asid = 3,
00036 } L4_vm_rfnc;
00037
00038 enum
00039 {
00040     L4_vm_hvip_vssip = 1UL < 2,
00041     L4_vm_hvip_vstip = 1UL < 6,
00042     L4_vm_hvip_vseip = 1UL < 10,
00043 };
00044
00050 typedef struct l4_vm_state_t
00051 {
00052     l4_umword_t hedeleg;
00053     l4_umword_t hideleg;
00054
00055     l4_umword_t hvip;
00056     l4_umword_t hip; // read-only
00057     l4_umword_t hie;
00058
00059     l4_uint64_t htimedelta;
00060
00061     l4_umword_t htval;
00062     l4_umword_t htinst;
00063
00064     l4_umword_t vsstatus;
00065     l4_umword_t vstvec;
00066     l4_umword_t vsscratch;
00067     l4_umword_t vsepc;
00068     l4_umword_t vscause;
00069     l4_umword_t vstval;
00070     l4_umword_t vsatp;
00071     l4_uint64_t vstimecmp;
00072
00073     // Indicates that a hypervisor load/store instruction failed. VMM is
00074     // responsible for resetting this value before executing a hypervisor/load
00075     // store instruction.
00076     l4_uint8_t hlasi_failed;
00077
00078     l4_uint8_t remote_fence;
00079     l4_umword_t remote_fence_hart_mask;
00080     l4_umword_t remote_fence_start_addr;
00081     l4_umword_t remote_fence_size;
00082     l4_umword_t remote_fence_asid;
00083 } l4_vm_state_t;
00084
```

```

00085
00086 L4_INLINE l4_vm_state_t *
00087 l4_vm_state(l4_vcpu_state_t *vcpu) L4_NOTHROW;
00088
00089 L4_INLINE l4_vm_state_t *
00090 l4_vm_state(l4_vcpu_state_t *vcpu) L4_NOTHROW
00091 { return (l4_vm_state_t *)((char *)vcpu + 0x400); }

```

16.129 vm.h

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Henning Schild <hschild@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013 #pragma once
00014
00015 #include <l4/sys/__vm-svm.h>
00016 #include <l4/sys/__vm-vmx.h>

```

16.130 amd64/l4/util/arch/l4_macros.h File Reference

Main function.

16.130.1 Detailed Description

Main function.

Date

08/29/2000

Author

Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file [l4_macros.h](#).

16.131 l4_macros.h

[Go to the documentation of this file.](#)

```

00001
00007
00008 /*
00009  * (c) 2006-2009 Author(s)
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *      License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013
00014 #ifndef _L4UTIL__ARCH_AMD64__L4_MACROS_H
00015 #define _L4UTIL__ARCH_AMD64__L4_MACROS_H
00016
00017 #ifndef l4_addr_fmt
00018 # define l4_addr_fmt    "%016lx"
00019 #endif
00020
00021 #endif /* !_L4UTIL__ARCH_AMD64__L4_MACROS_H */

```

16.132 arm/l4/util/arch/l4_macros.h File Reference

Main function.

16.132.1 Detailed Description

Main function.

Date

08/29/2000

Author

Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file [l4_macros.h](#).

16.133 l4_macros.h

[Go to the documentation of this file.](#)

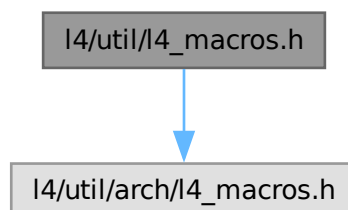
```
00001
00007
00008 /*
00009  * (c) 2006-2009 Author(s)
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013
00014 #ifndef _L4UTIL__ARCH_ARM__L4_MACROS_H
00015 #define _L4UTIL__ARCH_ARM__L4_MACROS_H
00016
00017 #ifndef l4_addr_fmt
00018 # define l4_addr_fmt    "%08lx"
00019 #endif
00020
00021 #endif /* !_L4UTIL__ARCH_ARM__L4_MACROS_H */
```

16.134 l4/util/l4_macros.h File Reference

Some useful generic macros, L4f version.

```
#include <l4/util/arch/l4_macros.h>
```

Include dependency graph for l4_macros.h:



16.134.1 Detailed Description

Some useful generic macros, L4f version.

Date

11/12/2002

Author

Lars Reuther reuther@os.inf.tu-dresden.de

Definition in file [l4_macros.h](#).

16.135 l4_macros.h

[Go to the documentation of this file.](#)

```
00001 /*****
00002 */
00003 * (c) 2000-2009 Author(s)
00004 *     economic rights: Technische Universität Dresden (Germany)
00005 * License: see LICENSE.spdx (in this directory or the directories above)
00006 */
00007
00008 /*****
00009 */
00010 #pragma once
00011 #include <l4/util/arch/l4_macros.h>
00012
00013 /*****
00014 *** generic macros
00015 *****/
00016
00017 /* generate L4 thread id printf string */
00018 #ifndef l4util_idstr
00019 #   define l4util_idfmt      "%lx"
00020 #   define l4util_idfmt_adjust "%04lx"
00021 #   define l4util_idstr(tid) (tid » L4_CAP_SHIFT)
00022 #endif
00023
00024
```

16.136 x86/l4/util/arch/l4_macros.h File Reference

Main function.

16.136.1 Detailed Description

Main function.

Date

08/29/2000

Author

Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file [l4_macros.h](#).

16.137 l4_macros.h

[Go to the documentation of this file.](#)

```
00001
00002
00003 /*
00004  * (c) 2006-2009 Author(s)
00005  *     economic rights: Technische Universität Dresden (Germany)
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008
00009 #ifndef __L4UTIL__ARCH_X86__L4_MACROS_H
00010 #define __L4UTIL__ARCH_X86__L4_MACROS_H
00011
00012 #ifndef l4_addr_fmt
00013 #define l4_addr_fmt "%08lx"
00014 #endif
00015
00016 #endif /* !__L4UTIL__ARCH_X86__L4_MACROS_H */
```

16.138 thread.h

```
00001 /*
00002  * Copyright (C) 2026 Kernkonzept GmbH.
00003  * Author(s): Jan Klötzke <jan.kloetzke@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
```

16.139 thread.h

```
00001 /*
00002  * Copyright (C) 2026 Kernkonzept GmbH.
00003  * Author(s): Jan Klötzke <jan.kloetzke@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
```

16.140 arm/l4/sys/arch/thread.h File Reference

ARM-specific thread related definitions.

Enumerations

- enum [L4_thread_ex_regs_flags_arm](#) { [L4_THREAD_EX_REGS_ARM_SET_EL_MASK](#) = 0x3 << 24 , [L4_THREAD_EX_REGS_ARM_SET_EL_KEEP](#) = 0x0 << 24 , [L4_THREAD_EX_REGS_ARM_SET_EL_EL0](#) = 0x1 << 24 , [L4_THREAD_EX_REGS_ARM_SET_EL_EL1](#) = 0x2 << 24 }

Arm specific [L4::Thread::ex_regs\(\)](#) flags.

Functions

- [l4_msgtag_t l4_thread_arm_set_tpidruro](#) ([l4_cap_idx_t](#) thread, [l4_addr_t](#) tpidruro) [L4_NOTHROW](#)

Set the [TPIDRURO](#) thread specific register.

16.140.1 Detailed Description

ARM-specific thread related definitions.

Definition in file [thread.h](#).

16.141 thread.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2013 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011 #pragma once
00012
00013 // Use the full documentation from ARM64. Otherwise \parameters and \return
00014 // would occur twice in the Doxygen documentation of this function.
00018 L4_INLINE l4_msgtag_t
00019 l4_thread_arm_set_tpidruru(l4_cap_idx_t thread, l4_addr_t tpidruru) L4_NOTHROW;
00020
00025 L4_INLINE l4_msgtag_t
00026 l4_thread_arm_set_tpidruru_u(l4_cap_idx_t thread, l4_addr_t tpidruru,
00027                             l4_utcb_t *utcb) L4_NOTHROW;
00028
00037 enum l4_thread_ex_regs_flags_arm
00038 {
00040     L4_THREAD_EX_REGS_ARM_SET_EL_MASK      = 0x3 << 24,
00042     L4_THREAD_EX_REGS_ARM_SET_EL_KEEP     = 0x0 << 24,
00044     L4_THREAD_EX_REGS_ARM_SET_EL_EL0      = 0x1 << 24,
00046     L4_THREAD_EX_REGS_ARM_SET_EL_EL1      = 0x2 << 24,
00047 };
00048
00049 /* IMPLEMENTATION -----*/
00050
00051 L4_INLINE l4_msgtag_t
00052 l4_thread_arm_set_tpidruru_u(l4_cap_idx_t thread, l4_addr_t tpidruru,
00053                             l4_utcb_t *utcb) L4_NOTHROW
00054 {
00055     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00056     v->mr[0] = L4_THREAD_ARM_TPIDRURO_OP;
00057     v->mr[1] = tpidruru;
00058     return l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 2, 0, 0),
00059                       L4_IPC_NEVER);
00060 }
00061
00062 L4_INLINE l4_msgtag_t
00063 l4_thread_arm_set_tpidruru(l4_cap_idx_t thread, l4_addr_t tpidruru) L4_NOTHROW
00064 {
00065     return l4_thread_arm_set_tpidruru_u(thread, tpidruru, l4_utcb());
00066 }

```

16.142 thread.h

```

00001 /*
00002  * Copyright (C) 2026 Kernkonzept GmbH.
00003  * Author(s): Jan Klötzke <jan.kloetzke@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once

```

16.143 arm64/l4/sys/arch/thread.h File Reference

ARM64-specific thread related definitions.

Enumerations

- enum `L4_thread_ex_regs_flags_arm64` { `L4_THREAD_EX_REGS_ARM64_SET_EL_MASK` = 0x3 << 24 , `L4_THREAD_EX_REGS_ARM64_SET_EL_KEEP` = 0x0 << 24 , `L4_THREAD_EX_REGS_ARM64_SET_EL_EL0` = 0x1 << 24 , `L4_THREAD_EX_REGS_ARM64_SET_EL_EL1` = 0x2 << 24 }

Arm64 specific `L4::Thread::ex_regs()` flags.

Functions

- `l4_msgtag_t l4_thread_arm_set_tpidruro` (`l4_cap_idx_t` thread, `l4_addr_t` tpidruro) `L4_NOTHROW`

Set the `TPIDRURO` thread specific register.

16.143.1 Detailed Description

ARM64-specific thread related definitions.

Definition in file [thread.h](#).

16.144 thread.h

[Go to the documentation of this file.](#)

```

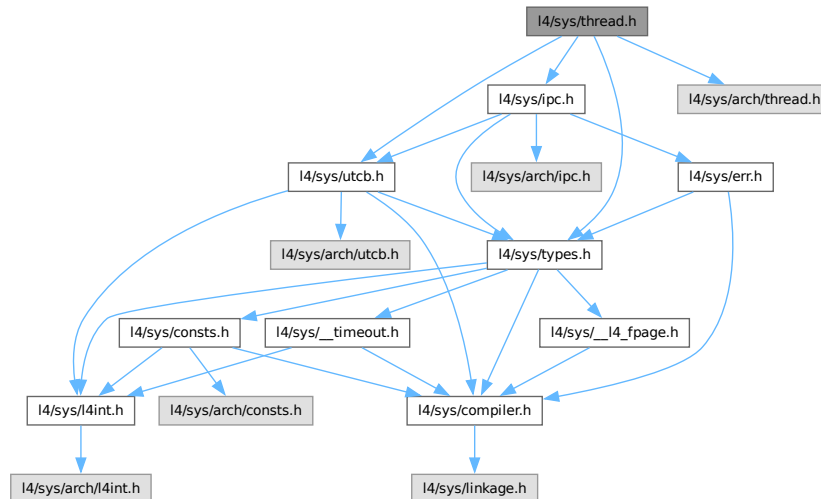
00001
00005 /*
00006  * (c) 2013 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011 #pragma once
00012
00025 L4_INLINE l4_msgtag_t
00026 l4_thread_arm_set_tpidruro(l4_cap_idx_t thread, l4_addr_t tpidruro) L4_NOTHROW;
00027
00032 L4_INLINE l4_msgtag_t
00033 l4_thread_arm_set_tpidruro_u(l4_cap_idx_t thread, l4_addr_t tpidruro,
00034                             l4_utcb_t *utcb) L4_NOTHROW;
00035
00044 enum L4_thread_ex_regs_flags_arm64
00045 {
00047     L4_THREAD_EX_REGS_ARM64_SET_EL_MASK      = 0x3 << 24,
00049     L4_THREAD_EX_REGS_ARM64_SET_EL_KEEP      = 0x0 << 24,
00051     L4_THREAD_EX_REGS_ARM64_SET_EL_EL0       = 0x1 << 24,
00053     L4_THREAD_EX_REGS_ARM64_SET_EL_EL1       = 0x2 << 24,
00054 };
00055
00056 /* IMPLEMENTATION -----*/
00057
00058 L4_INLINE l4_msgtag_t
00059 l4_thread_arm_set_tpidruro_u(l4_cap_idx_t thread, l4_addr_t tpidruro,
00060                             l4_utcb_t *utcb) L4_NOTHROW
00061 {
00062     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00063     v->mr[0] = L4_THREAD_ARM_TPIDRURO_OP;
00064     v->mr[1] = tpidruro;
00065     return l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 2, 0, 0),
00066                       L4_IPC_NEVER);
00067 }
00068
00069 L4_INLINE l4_msgtag_t
00070 l4_thread_arm_set_tpidruro(l4_cap_idx_t thread, l4_addr_t tpidruro) L4_NOTHROW
00071 {
00072     return l4_thread_arm_set_tpidruro_u(thread, tpidruro, l4_utcb());
00073 }

```

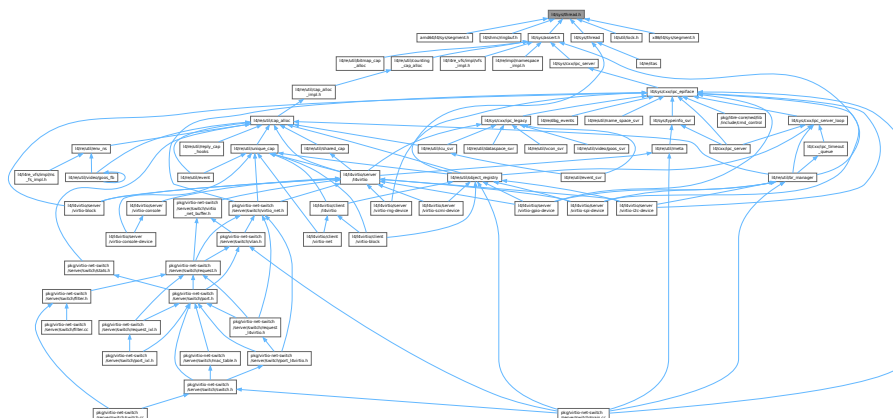
16.145 l4/sys/thread.h File Reference

Common thread related definitions.

```
#include <l4/sys/types.h>
#include <l4/sys/utcb.h>
#include <l4/sys/ipc.h>
#include <l4/sys/arch/thread.h>
Include dependency graph for thread.h:
```



This graph shows which files directly or indirectly include this file:



Enumerations

- enum `L4_thread_ops` {
 - `L4_THREAD_CONTROL_OP` = 0UL , `L4_THREAD_EX_REGS_OP` = 1UL , `L4_THREAD_SWITCH_OP` = 2UL , `L4_THREAD_STATS_OP` = 3UL ,
 - `L4_THREAD_VCPU_RESUME_OP` = 4UL , `L4_THREAD_REGISTER_DELETE_IRQ_OP` = 5UL ,


```

L4_THREAD_MODIFY_SENDER_OP = 6UL , L4_THREAD_VCPU_CONTROL_OP = 7UL ,
L4_THREAD_VCPU_CONTROL_EXT_OP = L4_THREAD_VCPU_CONTROL_OP | 0x10000 , L4_THREAD_REGISTER_DO
= 8UL , L4_THREAD_X86_GDT_OP = 0x10UL , L4_THREAD_ARM_TPIDRURO_OP = 0x10UL ,
L4_THREAD_AMD64_SET_SEGMENT_BASE_OP = 0x12UL , L4_THREAD_AMD64_GET_SEGMENT_INFO_OP
= 0x13UL , L4_THREAD_OPCODE_MASK = 0xffff }

```

Operations on thread objects.

- enum `L4_thread_control_flags` { `L4_THREAD_CONTROL_SET_PAGER` = 0x0010000 , `L4_THREAD_CONTROL_BIND_TASK` = 0x0200000 , `L4_THREAD_CONTROL_ALIEN` = 0x0400000 , `L4_THREAD_CONTROL_SET_EXC_HANDLER` = 0x1000000 }

Flags for the thread control operation.

- enum `L4_thread_control_mr_indices` { `L4_THREAD_CONTROL_MR_IDX_FLAGS` = 0 , `L4_THREAD_CONTROL_MR_IDX_PAGER` = 1 , `L4_THREAD_CONTROL_MR_IDX_EXC_HANDLER` = 2 , `L4_THREAD_CONTROL_MR_IDX_FLAG_VALS` = 4 , `L4_THREAD_CONTROL_MR_IDX_BIND_UTCB` = 5 , `L4_THREAD_CONTROL_MR_IDX_BIND_TASK` = 6 }

Indices for the values in the message register for thread control.

- enum `L4_thread_ex_regs_flags` { `L4_THREAD_EX_REGS_CANCEL` = 0x10000UL , `L4_THREAD_EX_REGS_TRIGGER_EXC` = 0x20000UL , `L4_THREAD_EX_REGS_ARCH_MASK` = 0xff000000UL }

Flags for the thread ex-regs operation.

Functions

- `l4_msgtag_t l4_thread_ex_regs` (`l4_cap_idx_t` thread, `l4_addr_t` ip, `l4_addr_t` sp, `l4_umword_t` flags) `L4_NOTHROW`

Exchange basic thread registers.

- `l4_msgtag_t l4_thread_ex_regs_u` (`l4_cap_idx_t` thread, `l4_addr_t` ip, `l4_addr_t` sp, `l4_umword_t` flags, `l4_utcb_t` *utcb) `L4_NOTHROW`

Exchange basic thread registers.

- `l4_msgtag_t l4_thread_ex_regs_ret` (`l4_cap_idx_t` thread, `l4_addr_t` *ip, `l4_addr_t` *sp, `l4_umword_t` *flags) `L4_NOTHROW`

Exchange basic thread registers and return previous values.

- `l4_msgtag_t l4_thread_ex_regs_ret_u` (`l4_cap_idx_t` thread, `l4_addr_t` *ip, `l4_addr_t` *sp, `l4_umword_t` *flags, `l4_utcb_t` *utcb) `L4_NOTHROW`

Exchange basic thread registers and return previous values.

- void `l4_thread_control_start` (void) `L4_NOTHROW`

Start a thread control API sequence.

- void `l4_thread_control_pager` (`l4_cap_idx_t` pager) `L4_NOTHROW`

Set the pager.

- void `l4_thread_control_exc_handler` (`l4_cap_idx_t` exc_handler) `L4_NOTHROW`

Set the exception handler.

- void `l4_thread_control_bind` (`l4_utcb_t` *thread_utcb, `l4_cap_idx_t` task) `L4_NOTHROW`

Bind the thread to a task.

- void `l4_thread_control_alien` (int on) `L4_NOTHROW`

Enable alien mode.

- `l4_msgtag_t l4_thread_control_commit` (`l4_cap_idx_t` thread) `L4_NOTHROW`

Commit the thread control parameters.

- `l4_msgtag_t l4_thread_yield` (void) `L4_NOTHROW`

Yield current time slice.

- `l4_msgtag_t l4_thread_switch` (`l4_cap_idx_t` to_thread) `L4_NOTHROW`

Switch to another thread (and donate the remaining time slice).

- `l4_msgtag_t l4_thread_stats_time` (`l4_cap_idx_t` thread, `l4_kernel_clock_t` *us) `L4_NOTHROW`

- Get consumed time of thread in μ s.*
- [l4_msgtag_t l4_thread_vcpu_resume_start](#) (void) [L4_NOTHROW](#)
vCPU return from event handler.
- [l4_msgtag_t l4_thread_vcpu_resume_commit](#) ([l4_cap_idx_t](#) thread, [l4_msgtag_t](#) tag) [L4_NOTHROW](#)
Commit vCPU resume.
- [l4_msgtag_t l4_thread_vcpu_control](#) ([l4_cap_idx_t](#) thread, [l4_addr_t](#) vcpu_state) [L4_NOTHROW](#)
Enable the vCPU feature for the thread.
- [l4_msgtag_t l4_thread_vcpu_control_u](#) ([l4_cap_idx_t](#) thread, [l4_addr_t](#) vcpu_state, [l4_utcb_t](#) *utcb) [L4_NOTHROW](#)
Enable the vCPU feature for the thread.
- [l4_msgtag_t l4_thread_vcpu_control_ext](#) ([l4_cap_idx_t](#) thread, [l4_addr_t](#) ext_vcpu_state) [L4_NOTHROW](#)
Enable the extended vCPU feature for the thread.
- [l4_msgtag_t l4_thread_vcpu_control_ext_u](#) ([l4_cap_idx_t](#) thread, [l4_addr_t](#) ext_vcpu_state, [l4_utcb_t](#) *utcb) [L4_NOTHROW](#)
Enable the extended vCPU feature for the thread.
- [l4_msgtag_t l4_thread_register_del_irq](#) ([l4_cap_idx_t](#) thread, [l4_cap_idx_t](#) irq) [L4_NOTHROW](#)
Register an IRQ that will trigger upon deletion events.
- [l4_msgtag_t l4_thread_modify_sender_start](#) (void) [L4_NOTHROW](#)
Start a thread sender modification sequence.
- [int l4_thread_modify_sender_add](#) ([l4_umword_t](#) match_mask, [l4_umword_t](#) match, [l4_umword_t](#) del_bits, [l4_umword_t](#) add_bits, [l4_msgtag_t](#) *tag) [L4_NOTHROW](#)
Add a modification pattern to a sender modification sequence.
- [l4_msgtag_t l4_thread_modify_sender_commit](#) ([l4_cap_idx_t](#) thread, [l4_msgtag_t](#) tag) [L4_NOTHROW](#)
Apply (commit) a sender modification sequence.
- [l4_msgtag_t l4_thread_register_doorbell_irq](#) ([l4_cap_idx_t](#) thread, [l4_cap_idx_t](#) irq) [L4_NOTHROW](#)
Register an IRQ that will trigger when a forwarded virtual interrupt is pending.

16.145.1 Detailed Description

Common thread related definitions.

Definition in file [thread.h](#).

16.146 thread.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *               Alexander Warg <warg@os.inf.tu-dresden.de>,
00008  *               Björn Döbel <doebel@os.inf.tu-dresden.de>,
00009  *               Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010  *               economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014 #pragma once
00015
00016 #include <l4/sys/types.h>
00017 #include <l4/sys/utcb.h>
00018 #include <l4/sys/ipc.h>
00019
00054
00055
00083 L4_INLINE l4_msgtag_t
00084 l4_thread_ex_regs(l4_cap_idx_t thread, l4_addr_t ip, l4_addr_t sp,
00085                  l4_umword_t flags) L4_NOTHROW;

```

```

00086
00093 L4_INLINE l4_msgtag_t
00094 l4_thread_ex_regs_u(l4_cap_idx_t thread, l4_addr_t ip, l4_addr_t sp,
00095                    l4_umword_t flags, l4_utcb_t *utcb) L4_NOTHROW;
00096
00129 L4_INLINE l4_msgtag_t
00130 l4_thread_ex_regs_ret(l4_cap_idx_t thread, l4_addr_t *ip, l4_addr_t *sp,
00131                     l4_umword_t *flags) L4_NOTHROW;
00132
00139 L4_INLINE l4_msgtag_t
00140 l4_thread_ex_regs_ret_u(l4_cap_idx_t thread, l4_addr_t *ip, l4_addr_t *sp,
00141                       l4_umword_t *flags, l4_utcb_t *utcb) L4_NOTHROW;
00142
00143
00144
00170
00190 L4_INLINE void
00191 l4_thread_control_start(void) L4_NOTHROW;
00192
00197 L4_INLINE void
00198 l4_thread_control_start_u(l4_utcb_t *utcb) L4_NOTHROW;
00199
00209 L4_INLINE void
00210 l4_thread_control_pager(l4_cap_idx_t pager) L4_NOTHROW;
00211
00216 L4_INLINE void
00217 l4_thread_control_pager_u(l4_cap_idx_t pager, l4_utcb_t *utcb) L4_NOTHROW;
00218
00228 L4_INLINE void
00229 l4_thread_control_exc_handler(l4_cap_idx_t exc_handler) L4_NOTHROW;
00230
00235 L4_INLINE void
00236 l4_thread_control_exc_handler_u(l4_cap_idx_t exc_handler,
00237                                l4_utcb_t *utcb) L4_NOTHROW;
00238
00266 L4_INLINE void
00267 l4_thread_control_bind(l4_utcb_t *thread_utcb,
00268                       l4_cap_idx_t task) L4_NOTHROW;
00269
00274 L4_INLINE void
00275 l4_thread_control_bind_u(l4_utcb_t *thread_utcb,
00276                         l4_cap_idx_t task, l4_utcb_t *utcb) L4_NOTHROW;
00277
00301 L4_INLINE void
00302 l4_thread_control_alien(int on) L4_NOTHROW;
00303
00308 L4_INLINE void
00309 l4_thread_control_alien_u(l4_utcb_t *utcb, int on) L4_NOTHROW;
00310
00311
00312
00313
00330 L4_INLINE l4_msgtag_t
00331 l4_thread_control_commit(l4_cap_idx_t thread) L4_NOTHROW;
00332
00337 L4_INLINE l4_msgtag_t
00338 l4_thread_control_commit_u(l4_cap_idx_t thread, l4_utcb_t *utcb) L4_NOTHROW;
00339
00346 L4_INLINE l4_msgtag_t
00347 l4_thread_yield(void) L4_NOTHROW;
00348
00357 L4_INLINE l4_msgtag_t
00358 l4_thread_switch(l4_cap_idx_t to_thread) L4_NOTHROW;
00359
00364 L4_INLINE l4_msgtag_t
00365 l4_thread_switch_u(l4_cap_idx_t to_thread, l4_utcb_t *utcb) L4_NOTHROW;
00366
00367
00368
00378 L4_INLINE l4_msgtag_t
00379 l4_thread_stats_time(l4_cap_idx_t thread, l4_kernel_clock_t *us) L4_NOTHROW;
00380
00385 L4_INLINE l4_msgtag_t
00386 l4_thread_stats_time_u(l4_cap_idx_t thread, l4_kernel_clock_t *us,
00387                       l4_utcb_t *utcb) L4_NOTHROW;
00388
00389
00400 L4_INLINE l4_msgtag_t
00401 l4_thread_vcpu_resume_start(void) L4_NOTHROW;
00402
00407 L4_INLINE l4_msgtag_t
00408 l4_thread_vcpu_resume_start_u(l4_utcb_t *utcb) L4_NOTHROW;
00409
00457 L4_INLINE l4_msgtag_t
00458 l4_thread_vcpu_resume_commit(l4_cap_idx_t thread,
00459                              l4_msgtag_t tag) L4_NOTHROW;
00460

```

```

00465 L4_INLINE l4_msgtag_t
00466 l4_thread_vcpu_resume_commit_u(l4_cap_idx_t thread,
00467                                l4_msgtag_t tag, l4_utcb_t *utcb) L4_NOTHROW;
00468
00469
00489 L4_INLINE l4_msgtag_t
00490 l4_thread_vcpu_control(l4_cap_idx_t thread, l4_addr_t vcpu_state) L4_NOTHROW;
00491
00499 L4_INLINE l4_msgtag_t
00500 l4_thread_vcpu_control_u(l4_cap_idx_t thread, l4_addr_t vcpu_state,
00501                          l4_utcb_t *utcb) L4_NOTHROW;
00502
00534 L4_INLINE l4_msgtag_t
00535 l4_thread_vcpu_control_ext(l4_cap_idx_t thread, l4_addr_t ext_vcpu_state) L4_NOTHROW;
00536
00544 L4_INLINE l4_msgtag_t
00545 l4_thread_vcpu_control_ext_u(l4_cap_idx_t thread, l4_addr_t ext_vcpu_state,
00546                              l4_utcb_t *utcb) L4_NOTHROW;
00547
00548
00572 L4_INLINE l4_msgtag_t
00573 l4_thread_register_del_irq(l4_cap_idx_t thread, l4_cap_idx_t irq) L4_NOTHROW;
00574
00579 L4_INLINE l4_msgtag_t
00580 l4_thread_register_del_irq_u(l4_cap_idx_t thread, l4_cap_idx_t irq,
00581                              l4_utcb_t *utcb) L4_NOTHROW;
00582
00604 L4_INLINE l4_msgtag_t
00605 l4_thread_modify_sender_start(void) L4_NOTHROW;
00606
00611 L4_INLINE l4_msgtag_t
00612 l4_thread_modify_sender_start_u(l4_utcb_t *u) L4_NOTHROW;
00613
00638 L4_INLINE int
00639 l4_thread_modify_sender_add(l4_umword_t match_mask,
00640                             l4_umword_t match,
00641                             l4_umword_t del_bits,
00642                             l4_umword_t add_bits,
00643                             l4_msgtag_t *tag) L4_NOTHROW;
00644
00649 L4_INLINE int
00650 l4_thread_modify_sender_add_u(l4_umword_t match_mask,
00651                               l4_umword_t match,
00652                               l4_umword_t del_bits,
00653                               l4_umword_t add_bits,
00654                               l4_msgtag_t *tag, l4_utcb_t *u) L4_NOTHROW;
00655
00681 L4_INLINE l4_msgtag_t
00682 l4_thread_modify_sender_commit(l4_cap_idx_t thread, l4_msgtag_t tag) L4_NOTHROW;
00683
00688 L4_INLINE l4_msgtag_t
00689 l4_thread_modify_sender_commit_u(l4_cap_idx_t thread, l4_msgtag_t tag,
00690                                  l4_utcb_t *u) L4_NOTHROW;
00691
00692
00713 L4_INLINE l4_msgtag_t
00714 l4_thread_register_doorbell_irq(l4_cap_idx_t thread,
00715                                 l4_cap_idx_t irq) L4_NOTHROW;
00716
00721 L4_INLINE l4_msgtag_t
00722 l4_thread_register_doorbell_irq_u(l4_cap_idx_t thread, l4_cap_idx_t irq,
00723                                   l4_utcb_t *u) L4_NOTHROW;
00724
00725
00732 enum L4_thread_ops
00733 {
00734     L4_THREAD_CONTROL_OP           = 0UL,
00735     L4_THREAD_EX_REGS_OP          = 1UL,
00736     L4_THREAD_SWITCH_OP           = 2UL,
00737     L4_THREAD_STATS_OP            = 3UL,
00738     L4_THREAD_VCPU_RESUME_OP      = 4UL,
00739     L4_THREAD_REGISTER_DELETE_IRQ_OP = 5UL,
00740     L4_THREAD_MODIFY_SENDER_OP     = 6UL,
00741     L4_THREAD_VCPU_CONTROL_OP     = 7UL,
00742     L4_THREAD_VCPU_CONTROL_EXT_OP  = L4_THREAD_VCPU_CONTROL_OP | 0x10000,
00743     L4_THREAD_REGISTER_DOORBELL_IRQ_OP = 8UL,
00744     L4_THREAD_X86_GDT_OP          = 0x10UL,
00745     L4_THREAD_ARM_TPIDRURO_OP     = 0x10UL,
00746     L4_THREAD_AMD64_SET_SEGMENT_BASE_OP = 0x12UL,
00747     L4_THREAD_AMD64_GET_SEGMENT_INFO_OP = 0x13UL,
00748     L4_THREAD_OPCODE_MASK        = 0xffff,
00749 };
00750
00761 enum L4_thread_control_flags
00762 {
00764     L4_THREAD_CONTROL_SET_PAGER      = 0x0010000,
00766     L4_THREAD_CONTROL_BIND_TASK     = 0x0200000,

```

```

00768     L4_THREAD_CONTROL_ALIEN                = 0x0400000,
00770     L4_THREAD_CONTROL_SET_EXC_HANDLER = 0x1000000,
00771 };
00772
00782 enum L4_thread_control_mr_indices
00783 {
00784     L4_THREAD_CONTROL_MR_IDX_FLAGS          = 0,
00785     L4_THREAD_CONTROL_MR_IDX_PAGER          = 1,
00786     L4_THREAD_CONTROL_MR_IDX_EXC_HANDLER    = 2,
00787     L4_THREAD_CONTROL_MR_IDX_FLAG_VALS      = 4,
00788     L4_THREAD_CONTROL_MR_IDX_BIND_UTCB      = 5,
00789     L4_THREAD_CONTROL_MR_IDX_BIND_TASK      = 6,
00790 };
00791
00797 enum L4_thread_ex_regs_flags
00798 {
00799     L4_THREAD_EX_REGS_CANCEL                = 0x10000UL,
00800     L4_THREAD_EX_REGS_TRIGGER_EXCEPTION    = 0x20000UL,
00801
00802     L4_THREAD_EX_REGS_ARCH_MASK            = 0xff000000UL,
00803 };
00804
00805
00806 /* IMPLEMENTATION -----*/
00807
00808 #include <l4/sys/ipc.h>
00809 #include <l4/sys/types.h>
00810
00811 L4_INLINE l4_msgtag_t
00812 l4_thread_ex_regs_u(l4_cap_idx_t thread, l4_addr_t ip, l4_addr_t sp,
00813                    l4_umword_t flags, l4_utcb_t *utcb) L4_NOTHROW
00814 {
00815     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00816     v->mr[0] = L4_THREAD_EX_REGS_OP | flags;
00817     v->mr[1] = ip;
00818     v->mr[2] = sp;
00819     return l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 3, 0, 0), L4_IPC_NEVER);
00820 }
00821
00822 L4_INLINE l4_msgtag_t
00823 l4_thread_ex_regs_ret_u(l4_cap_idx_t thread, l4_addr_t *ip, l4_addr_t *sp,
00824                        l4_umword_t *flags, l4_utcb_t *utcb) L4_NOTHROW
00825 {
00826     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00827     l4_msgtag_t ret = l4_thread_ex_regs_u(thread, *ip, *sp, *flags, utcb);
00828     if (l4_error_u(ret, utcb))
00829         return ret;
00830
00831     *flags = v->mr[0];
00832     *ip     = v->mr[1];
00833     *sp     = v->mr[2];
00834     return ret;
00835 }
00836
00837 L4_INLINE void
00838 l4_thread_control_start_u(l4_utcb_t *utcb) L4_NOTHROW
00839 {
00840     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00841     v->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS] = L4_THREAD_CONTROL_OP;
00842 }
00843
00844 L4_INLINE void
00845 l4_thread_control_pager_u(l4_cap_idx_t pager, l4_utcb_t *utcb) L4_NOTHROW
00846 {
00847     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00848     v->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS] |= L4_THREAD_CONTROL_SET_PAGER;
00849     v->mr[L4_THREAD_CONTROL_MR_IDX_PAGER] = pager;
00850 }
00851
00852 L4_INLINE void
00853 l4_thread_control_exc_handler_u(l4_cap_idx_t exc_handler,
00854                                l4_utcb_t *utcb) L4_NOTHROW
00855 {
00856     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00857     v->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS] |= L4_THREAD_CONTROL_SET_EXC_HANDLER;
00858     v->mr[L4_THREAD_CONTROL_MR_IDX_EXC_HANDLER] = exc_handler;
00859 }
00860
00861 L4_INLINE void
00862 l4_thread_control_bind_u(l4_utcb_t *thread_utcb, l4_cap_idx_t task,
00863                         l4_utcb_t *utcb) L4_NOTHROW
00864 {
00865     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00866     v->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS] |= L4_THREAD_CONTROL_BIND_TASK;
00867     v->mr[L4_THREAD_CONTROL_MR_IDX_BIND_UTCB] = (l4_addr_t)thread_utcb;
00868     v->mr[L4_THREAD_CONTROL_MR_IDX_BIND_TASK] = L4_ITEM_MAP;
00869     v->mr[L4_THREAD_CONTROL_MR_IDX_BIND_TASK + 1] = l4_obj_fpage(task, 0, L4_CAP_FPAGE_RWS).raw;

```

```

00870 }
00871
00872 L4_INLINE void
00873 l4_thread_control_alien_u(l4_utcb_t *utcb, int on) L4_NOTHROW
00874 {
00875     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00876     v->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS] |= L4_THREAD_CONTROL_ALIEN;
00877     v->mr[L4_THREAD_CONTROL_MR_IDX_FLAG_VALS] |= on ? L4_THREAD_CONTROL_ALIEN : 0;
00878 }
00879
00880 L4_INLINE l4_msgtag_t
00881 l4_thread_control_commit_u(l4_cap_idx_t thread, l4_utcb_t *utcb) L4_NOTHROW
00882 {
00883     int items = 0;
00884     if (l4_utcb_mr_u(utcb)->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS] & L4_THREAD_CONTROL_BIND_TASK)
00885         items = 1;
00886     return l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 6, items, 0), L4_IPC_NEVER);
00887 }
00888
00889
00890 L4_INLINE l4_msgtag_t
00891 l4_thread_yield(void) L4_NOTHROW
00892 {
00893     l4_ipc_receive(L4_INVALID_CAP, NULL, L4_IPC_BOTH_TIMEOUT_0);
00894     return l4_msgtag(0, 0, 0, 0);
00895 }
00896
00897 /* Preliminary, to be changed */
00898 L4_INLINE l4_msgtag_t
00899 l4_thread_switch_u(l4_cap_idx_t to_thread, l4_utcb_t *utcb) L4_NOTHROW
00900 {
00901     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00902     v->mr[0] = L4_THREAD_SWITCH_OP;
00903     return l4_ipc_call(to_thread, utcb, l4_msgtag(L4_PROTO_THREAD, 1, 0, 0), L4_IPC_NEVER);
00904 }
00905
00906
00907 L4_INLINE l4_msgtag_t
00908 l4_thread_stats_time_u(l4_cap_idx_t thread, l4_kernel_clock_t *us,
00909                       l4_utcb_t *utcb) L4_NOTHROW
00910 {
00911     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00912     l4_msgtag_t res;
00913
00914     v->mr[0] = L4_THREAD_STATS_OP;
00915
00916     res = l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 1, 0, 0), L4_IPC_NEVER);
00917
00918     if (l4_msgtag_has_error(res))
00919         return res;
00920
00921     *us = v->mr64[l4_utcb_mr64_idx(0)];
00922
00923     return res;
00924 }
00925
00926 L4_INLINE l4_msgtag_t
00927 l4_thread_vcpu_resume_start_u(l4_utcb_t *utcb) L4_NOTHROW
00928 {
00929     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00930     v->mr[0] = L4_THREAD_VCPU_RESUME_OP;
00931     return l4_msgtag(L4_PROTO_THREAD, 1, 0, 0);
00932 }
00933
00934 L4_INLINE l4_msgtag_t
00935 l4_thread_vcpu_resume_commit_u(l4_cap_idx_t thread,
00936                               l4_msgtag_t tag, l4_utcb_t *utcb) L4_NOTHROW
00937 {
00938     return l4_ipc_call(thread, utcb, tag, L4_IPC_NEVER);
00939 }
00940
00941 L4_INLINE l4_msgtag_t
00942 l4_thread_ex_regs(l4_cap_idx_t thread, l4_addr_t ip, l4_addr_t sp,
00943                  l4_umword_t flags) L4_NOTHROW
00944 {
00945     return l4_thread_ex_regs_u(thread, ip, sp, flags, l4_utcb());
00946 }
00947
00948 L4_INLINE l4_msgtag_t
00949 l4_thread_ex_regs_ret(l4_cap_idx_t thread, l4_addr_t *ip, l4_addr_t *sp,
00950                      l4_umword_t *flags) L4_NOTHROW
00951 {
00952     return l4_thread_ex_regs_ret_u(thread, ip, sp, flags, l4_utcb());
00953 }
00954
00955 L4_INLINE void
00956 l4_thread_control_start(void) L4_NOTHROW

```

```

00957 {
00958     l4_thread_control_start_u(l4_utcb());
00959 }
00960
00961 L4_INLINE void
00962 l4_thread_control_pager(l4_cap_idx_t pager) L4_NOTHROW
00963 {
00964     l4_thread_control_pager_u(pager, l4_utcb());
00965 }
00966
00967 L4_INLINE void
00968 l4_thread_control_exc_handler(l4_cap_idx_t exc_handler) L4_NOTHROW
00969 {
00970     l4_thread_control_exc_handler_u(exc_handler, l4_utcb());
00971 }
00972
00973
00974 L4_INLINE void
00975 l4_thread_control_bind(l4_utcb_t *thread_utcb, l4_cap_idx_t task) L4_NOTHROW
00976 {
00977     l4_thread_control_bind_u(thread_utcb, task, l4_utcb());
00978 }
00979
00980 L4_INLINE void
00981 l4_thread_control_alien(int on) L4_NOTHROW
00982 {
00983     l4_thread_control_alien_u(l4_utcb(), on);
00984 }
00985
00986 L4_INLINE l4_msgtag_t
00987 l4_thread_control_commit(l4_cap_idx_t thread) L4_NOTHROW
00988 {
00989     return l4_thread_control_commit_u(thread, l4_utcb());
00990 }
00991
00992
00993
00994
00995 L4_INLINE l4_msgtag_t
00996 l4_thread_switch(l4_cap_idx_t to_thread) L4_NOTHROW
00997 {
00998     return l4_thread_switch_u(to_thread, l4_utcb());
00999 }
01000
01001
01002
01003
01004 L4_INLINE l4_msgtag_t
01005 l4_thread_stats_time(l4_cap_idx_t thread, l4_kernel_clock_t *us) L4_NOTHROW
01006 {
01007     return l4_thread_stats_time_u(thread, us, l4_utcb());
01008 }
01009
01010 L4_INLINE l4_msgtag_t
01011 l4_thread_vcpu_resume_start(void) L4_NOTHROW
01012 {
01013     return l4_thread_vcpu_resume_start_u(l4_utcb());
01014 }
01015
01016 L4_INLINE l4_msgtag_t
01017 l4_thread_vcpu_resume_commit(l4_cap_idx_t thread,
01018                               l4_msgtag_t tag) L4_NOTHROW
01019 {
01020     return l4_thread_vcpu_resume_commit_u(thread, tag, l4_utcb());
01021 }
01022
01023
01024 L4_INLINE l4_msgtag_t
01025 l4_thread_register_del_irq_u(l4_cap_idx_t thread, l4_cap_idx_t irq,
01026                               l4_utcb_t *u) L4_NOTHROW
01027 {
01028     l4_msg_regs_t *m = l4_utcb_mr_u(u);
01029     m->mr[0] = L4_THREAD_REGISTER_DELETE_IRQ_OP;
01030     m->mr[1] = l4_map_obj_control(0,0);
01031     m->mr[2] = l4_obj_fpage(irq, 0, L4_CAP_FPAGE_RWS).raw;
01032     return l4_ipc_call(thread, u, l4_msgtag(L4_PROTO_THREAD, 1, 1, 0), L4_IPC_NEVER);
01033 }
01034
01035
01036 L4_INLINE l4_msgtag_t
01037 l4_thread_register_del_irq(l4_cap_idx_t thread, l4_cap_idx_t irq) L4_NOTHROW
01038 {
01039     return l4_thread_register_del_irq_u(thread, irq, l4_utcb());
01040 }
01041
01042
01043 L4_INLINE l4_msgtag_t

```

```

01044 l4_thread_vcpu_control_u(l4_cap_idx_t thread, l4_addr_t vcpu_state,
01045                          l4_utcb_t *utcb) L4_NOTHROW
01046 {
01047     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
01048     v->mr[0] = L4_THREAD_VCPU_CONTROL_OP;
01049     v->mr[1] = vcpu_state;
01050     return l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 2, 0, 0), L4_IPC_NEVER);
01051 }
01052
01053 L4_INLINE l4_msgtag_t
01054 l4_thread_vcpu_control(l4_cap_idx_t thread, l4_addr_t vcpu_state) L4_NOTHROW
01055 { return l4_thread_vcpu_control_u(thread, vcpu_state, l4_utcb()); }
01056
01057 L4_INLINE l4_msgtag_t
01058 l4_thread_vcpu_control_ext_u(l4_cap_idx_t thread, l4_addr_t ext_vcpu_state,
01059                             l4_utcb_t *utcb) L4_NOTHROW
01060 {
01061     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
01062     v->mr[0] = L4_THREAD_VCPU_CONTROL_EXT_OP;
01063     v->mr[1] = ext_vcpu_state;
01064     return l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 2, 0, 0), L4_IPC_NEVER);
01065 }
01066
01067 L4_INLINE l4_msgtag_t
01068 l4_thread_vcpu_control_ext(l4_cap_idx_t thread, l4_addr_t ext_vcpu_state) L4_NOTHROW
01069 { return l4_thread_vcpu_control_ext_u(thread, ext_vcpu_state, l4_utcb()); }
01070
01071 L4_INLINE l4_msgtag_t
01072 l4_thread_modify_sender_start_u(l4_utcb_t *u) L4_NOTHROW
01073 {
01074     l4_msg_regs_t *m = l4_utcb_mr_u(u);
01075     m->mr[0] = L4_THREAD_MODIFY_SENDER_OP;
01076     return l4_msgtag(L4_PROTO_THREAD, 1, 0, 0);
01077 }
01078
01079 L4_INLINE int
01080 l4_thread_modify_sender_add_u(l4_umword_t match_mask,
01081                              l4_umword_t match,
01082                              l4_umword_t del_bits,
01083                              l4_umword_t add_bits,
01084                              l4_msgtag_t *tag, l4_utcb_t *u) L4_NOTHROW
01085 {
01086     l4_msg_regs_t *m = l4_utcb_mr_u(u);
01087     unsigned w = l4_msgtag_words(*tag);
01088     if (w >= L4_UTCB_GENERIC_DATA_SIZE - 4)
01089         return -L4_ENOMEM;
01090
01091     m->mr[w] = match_mask;
01092     m->mr[w+1] = match;
01093     m->mr[w+2] = del_bits;
01094     m->mr[w+3] = add_bits;
01095
01096     *tag = l4_msgtag(l4_msgtag_label(*tag), w + 4, 0, 0);
01097
01098     return 0;
01099 }
01100
01101 L4_INLINE l4_msgtag_t
01102 l4_thread_modify_sender_commit_u(l4_cap_idx_t thread, l4_msgtag_t tag,
01103                                 l4_utcb_t *u) L4_NOTHROW
01104 {
01105     return l4_ipc_call(thread, u, tag, L4_IPC_NEVER);
01106 }
01107
01108 L4_INLINE l4_msgtag_t
01109 l4_thread_modify_sender_start(void) L4_NOTHROW
01110 {
01111     return l4_thread_modify_sender_start_u(l4_utcb());
01112 }
01113
01114 L4_INLINE int
01115 l4_thread_modify_sender_add(l4_umword_t match_mask,
01116                             l4_umword_t match,
01117                             l4_umword_t del_bits,
01118                             l4_umword_t add_bits,
01119                             l4_msgtag_t *tag) L4_NOTHROW
01120 {
01121     return l4_thread_modify_sender_add_u(match_mask, match,
01122                                          del_bits, add_bits, tag, l4_utcb());
01123 }
01124
01125 L4_INLINE l4_msgtag_t
01126 l4_thread_modify_sender_commit(l4_cap_idx_t thread, l4_msgtag_t tag) L4_NOTHROW
01127 {
01128     return l4_thread_modify_sender_commit_u(thread, tag, l4_utcb());
01129 }
01130

```



```

01131
01132
01133 L4_INLINE l4_msgtag_t
01134 l4_thread_register_doorbell_irq_u(l4_cap_idx_t thread, l4_cap_idx_t irq,
01135                                   l4_utcb_t *u) L4_NOTHROW
01136 {
01137     l4_msg_regs_t *m = l4_utcb_mr_u(u);
01138     m->mr[0] = L4_THREAD_REGISTER_DOORBELL_IRQ_OP;
01139     m->mr[1] = l4_map_obj_control(0,0);
01140     m->mr[2] = l4_obj_fpage(irq, 0, L4_CAP_FPAGE_RWS).raw;
01141     return l4_ipc_call(thread, u, l4_msgtag(L4_PROTO_THREAD, 1, 1, 0),
01142                       L4_IPC_NEVER);
01143 }
01144
01145 L4_INLINE l4_msgtag_t
01146 l4_thread_register_doorbell_irq(l4_cap_idx_t thread,
01147                                 l4_cap_idx_t irq) L4_NOTHROW
01148 {
01149     return l4_thread_register_doorbell_irq_u(thread, irq, l4_utcb());
01150 }
01151
01152 #include <l4/sys/arch/thread.h>

```

16.147 l4/util/thread.h File Reference

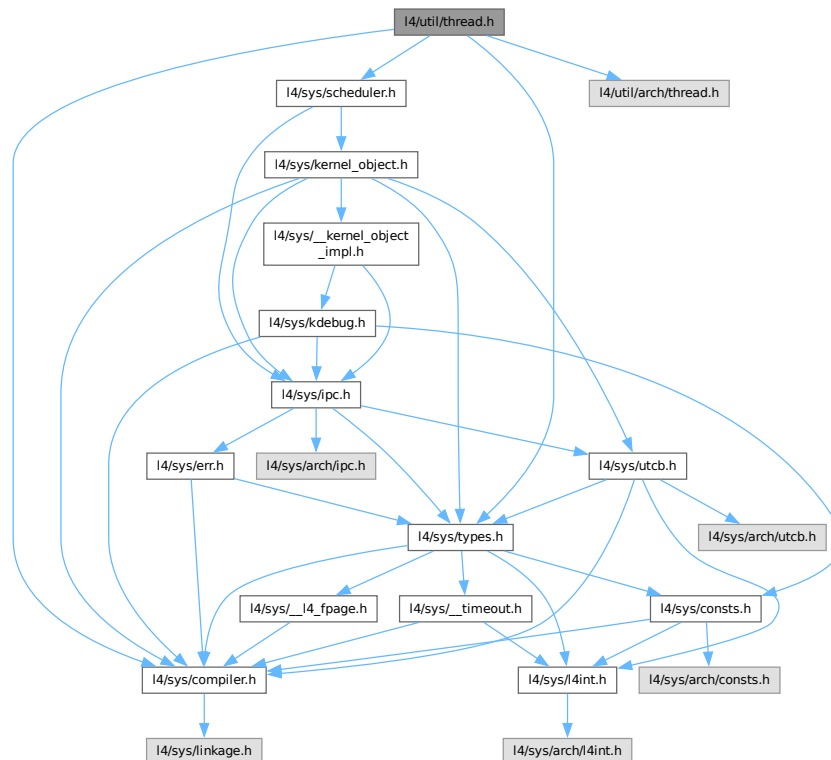
Low-level Thread Functions.

```

#include <l4/sys/compiler.h>
#include <l4/sys/types.h>
#include <l4/sys/scheduler.h>
#include <l4/util/arch/thread.h>

```

Include dependency graph for thread.h:



Macros

- `#define __L4UTIL_THREAD_FUNC(name)`

Defines a wrapper function that sets up the registers according to the calling conventions for the architecture.

16.147.1 Detailed Description

Low-level Thread Functions.

Date

1997

Author

Sebastian Schönberg

Definition in file [thread.h](#).

16.147.2 Macro Definition Documentation

16.147.2.1 __L4UTIL_THREAD_FUNC

```
#define __L4UTIL_THREAD_FUNC(  
    name)
```

Value:

```
void L4_NORETURN name(void)
```

Defines a wrapper function that sets up the registers according to the calling conventions for the architecture.

Use this as a function header when starting a low-level thread where only stack and instruction pointer are in a well-defined state.

Example:

```
L4UTIL_THREAD_FUNC(helper_thread) { l4_infinite_loop(); }
```

```
thread_cap->ex_regs((l4_umword_t)helper_thread, stack_addr);
```

Definition at line 73 of file [thread.h](#).

16.148 thread.h

[Go to the documentation of this file.](#)

```

00001
00002
00003 /*
00004  * (c) 2003-2009 Author(s)
00005  *     economic rights: Technische Universität Dresden (Germany)
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008
00009 #ifndef __L4_THREAD_H
00010 #define __L4_THREAD_H
00011
00012 #include <l4/sys/compiler.h>
00013 #include <l4/sys/types.h>
00014 #include <l4/sys/scheduler.h>
00015
00016 #include <l4/util/arch/thread.h>
00017
00018 L4_BEGIN_DECLS
00019
00020 L4_CV long
00021 l4util_create_thread(l4_cap_idx_t id, l4_utcb_t *thread_utcb,
00022                     l4_cap_idx_t factory,
00023                     l4_umword_t pc, l4_umword_t sp, l4_cap_idx_t pager,
00024                     l4_cap_idx_t task,
00025                     l4_cap_idx_t scheduler, l4_sched_param_t scp) L4_NOTHROW;
00026
00027 L4_END_DECLS
00028
00029 #ifndef L4UTIL_THREAD_FUNC
00030 #define __L4UTIL_THREAD_FUNC(name) void L4_NORETURN name(void)
00031 #define L4UTIL_THREAD_FUNC(name) __L4UTIL_THREAD_FUNC(name)
00032 #define __L4UTIL_THREAD_STATIC_FUNC(name) static L4_NORETURN void name(void)
00033 #define L4UTIL_THREAD_STATIC_FUNC(name) __L4UTIL_THREAD_STATIC_FUNC(name)
00034 #endif
00035
00036 #endif /* __L4_THREAD_H */

```

16.149 thread.h

```

00001 /*
00002  * Copyright (C) 2026 Kernkonzept GmbH.
00003  * Author(s): Jan Klötzke <jan.kloetzke@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once

```

16.150 thread.h

```

00001 /*
00002  * Copyright (C) 2026 Kernkonzept GmbH.
00003  * Author(s): Jan Klötzke <jan.kloetzke@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once

```

16.151 thread.h

```

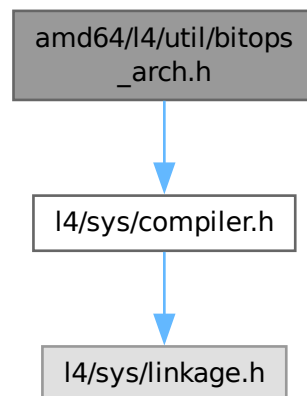
00001 /*
00002  * Copyright (C) 2026 Kernkonzept GmbH.
00003  * Author(s): Jan Klötzke <jan.kloetzke@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once

```

16.152 amd64/l4/util/bitops_arch.h File Reference

amd64 bit manipulation functions

#include <l4/sys/compiler.h>
Include dependency graph for bitops_arch.h:



16.152.1 Detailed Description

amd64 bit manipulation functions

Definition in file [bitops_arch.h](#).

16.153 bitops_arch.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (C) 2000-2009 Technische Universität Dresden (Germany)
00003  * Copyright (C) 2016, 2022, 2024-2025 Kernkonzept GmbH. All rights reserved.
00004  * Author(s): Lars Reuther <reuther@os.inf.tu-dresden.de>
00005  *            Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00006  *            Frank Mehnert <frank.mehnert@kernkonzept.com>
00007  *
00008  * License: see LICENSE.spdx (in this directory or the directories above)
00009  */
00010
00016
00017 #pragma once
00018
00019 #include <l4/sys/compiler.h>
00020
00021 /*
00022  * Note: The following Assembler statement may produce wrong code:
00023  *   asm volatile ("btsl %1, %2" : "=ccc"(r) : "Jr"(63), "m"(m) : "memory");
00024  *
00025  * The compiler might chose the first variant because the bit number is smaller
00026  * than 64. However, 'bts' is encoded as 32-bit variant ('btsl') and thus only
00027  * supports immediate bit values up to 31. Some assemblers support immediate
00028  * offsets > 31 by adapting the memory address accordingly but GAS does not.
00029  * With GAS, the instruction will encode an immediate value of 63 but the CPU

```

```

00030  * will set bit 31 instead of bit 63!
00031  *
00032  * Therefore, if we would use 'btsl' instead of 'btsq', the correct constraint
00033  * for the bit number parameter would be "Ir" instead of "Jr".
00034  */
00035
00036 L4_BEGIN_DECLS
00037
00038 /* set bit */
00039 #define __L4UTIL_BITOPS_HAVE_ARCH_SET_BIT
00040 L4_INLINE void
00041 l4util_set_bit(int b, volatile l4_umword_t * dest)
00042 {
00043     __asm__ __volatile__
00044     (
00045         "lock; btsq %1,%0    \n\t"
00046         :
00047         :
00048         "m"    (*dest), /* 0 mem, destination operand */
00049         "Jr"    ((l4_umword_t)b) /* 1, bit number */
00050         :
00051         "memory", "cc"
00052     );
00053 }
00054
00055 /* clear bit */
00056 #define __L4UTIL_BITOPS_HAVE_ARCH_CLEAR_BIT
00057 L4_INLINE void
00058 l4util_clear_bit(int b, volatile l4_umword_t * dest)
00059 {
00060     __asm__ __volatile__
00061     (
00062         "lock; btrq %1,%0    \n\t"
00063         :
00064         :
00065         "m"    (*dest), /* 0 mem, destination operand */
00066         "Jr"    ((l4_umword_t)b) /* 1, bit number */
00067         :
00068         "memory", "cc"
00069     );
00070 }
00071
00072 /* change bit */
00073 #define __L4UTIL_BITOPS_HAVE_ARCH_COMPLEMENT_BIT
00074 L4_INLINE void
00075 l4util_complement_bit(int b, volatile l4_umword_t * dest)
00076 {
00077     __asm__ __volatile__
00078     (
00079         "lock; btcq %1,%0    \n\t"
00080         :
00081         :
00082         "m"    (*dest), /* 0 mem, destination operand */
00083         "Jr"    ((l4_umword_t)b) /* 1, bit number */
00084         :
00085         "memory", "cc"
00086     );
00087 }
00088
00089 /* test bit */
00090 #define __L4UTIL_BITOPS_HAVE_ARCH_TEST_BIT
00091 L4_INLINE int
00092 l4util_test_bit(int b, const volatile l4_umword_t * dest)
00093 {
00094     l4_int8_t bit;
00095
00096     __asm__ __volatile__
00097     (
00098         "btq %2,%1    \n\t"
00099         :
00100         "=@ccc" (bit) /* 0, old bit value */
00101         :
00102         "m"    (*dest), /* 1 mem, destination operand */
00103         "Jr"    ((l4_umword_t)b) /* 2, bit number */
00104         :
00105         "memory"
00106     );
00107
00108     return bit;
00109 }
00110
00111 /* bit test and set */
00112 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_SET
00113 L4_INLINE int
00114 l4util_bts(int b, volatile l4_umword_t * dest)
00115 {
00116     l4_int8_t bit;

```

```

00117
00118 __asm__ __volatile__
00119 (
00120     "lock; btsq %2,%1    \n\t"
00121     :
00122     "=@ccc" (bit)      /* 0,      old bit value */
00123     :
00124     "m"      (*dest),   /* 1 mem, destination operand */
00125     "Jr"     ((l4_umword_t)b) /* 2, bit number */
00126     :
00127     "memory"
00128     );
00129
00130     return bit;
00131 }
00132
00133 /* bit test and reset */
00134 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_RESET
00135 L4_INLINE int
00136 l4util_btr(int b, volatile l4_umword_t * dest)
00137 {
00138     l4_int8_t bit;
00139
00140     __asm__ __volatile__
00141     (
00142         "lock; btrq %2,%1    \n\t"
00143         :
00144         "=@ccc" (bit)      /* 0,      old bit value */
00145         :
00146         "m"      (*dest),   /* 1 mem, destination operand */
00147         "Jr"     ((l4_umword_t)b) /* 2, bit number */
00148         :
00149         "memory"
00150         );
00151
00152     return bit;
00153 }
00154
00155 /* bit test and complement */
00156 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_COMPLEMENT
00157 L4_INLINE int
00158 l4util_btc(int b, volatile l4_umword_t * dest)
00159 {
00160     l4_int8_t bit;
00161
00162     __asm__ __volatile__
00163     (
00164         "lock; btcq %2,%1    \n\t"
00165         :
00166         "=@ccc" (bit)      /* 0,      old bit value */
00167         :
00168         "m"      (*dest),   /* 1 mem, destination operand */
00169         "Jr"     ((l4_umword_t)b) /* 2, bit number */
00170         :
00171         "memory"
00172         );
00173
00174     return bit;
00175 }
00176
00177 /* bit scan reverse */
00178 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_SCAN_REVERSE
00179 L4_INLINE int
00180 l4util_bsr(l4_umword_t word)
00181 {
00182     l4_umword_t tmp;
00183
00184     if (L4_UNLIKELY(word == 0))
00185         return -1;
00186
00187     __asm__ __volatile__
00188     (
00189         "bsrq %1,%0 \n\t"
00190         :
00191         "=r" (tmp)          /* 0, index of most significant set bit */
00192         :
00193         "r"   (word)        /* 1, argument */
00194         );
00195
00196     return tmp;
00197 }
00198
00199 /* bit scan forward */
00200 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_SCAN_FORWARD
00201 L4_INLINE int
00202 l4util_bsf(l4_umword_t word)
00203 {

```

```

00204     l4_umword_t tmp;
00205
00206     if (L4_UNLIKELY(word == 0))
00207         return -1;
00208
00209     __asm__ __volatile__
00210     (
00211         "bsfq %1,%0 \n\t"
00212         :
00213         "=r" (tmp)          /* 0, index of least significant set bit */
00214         :
00215         "r" (word)          /* 1, argument */
00216         );
00217
00218     return tmp;
00219 }
00220
00221 #define __L4UTIL_BITOPS_HAVE_ARCH_FIND_FIRST_SET_BIT
00222 L4_INLINE int
00223 l4util_find_first_set_bit(const void * dest, l4_size_t size)
00224 {
00225     l4_mword_t dummy0, dummy1, res;
00226
00227     __asm__ __volatile__
00228     (
00229         "repe; scasq          \n\t"
00230         "jz     1f            \n\t"
00231         "lea    -8(%%rdi),%%rdi \n\t"
00232         "bsf    (%%rdi),%%rax   \n\t"
00233         "1:          \n\t"
00234         "sub    %%rbx,%%rdi     \n\t"
00235         "shl    $3,%%rdi        \n\t"
00236         "add    %%rdi,%%rax     \n\t"
00237         :
00238         "=a" (res), "=c" (dummy0), "=D" (dummy1)
00239         :
00240         "a" (0), "b" (dest), "c" ((size + 63) >> 6), "D" (dest)
00241         :
00242         "cc", "memory");
00243
00244     return res;
00245 }
00246
00247 #define __L4UTIL_BITOPS_HAVE_ARCH_FIND_FIRST_ZERO_BIT
00248 L4_INLINE int
00249 l4util_find_first_zero_bit(const void * dest, l4_size_t size)
00250 {
00251     l4_mword_t dummy0, dummy1, dummy2, res;
00252
00253     if (!size)
00254         return 0;
00255
00256     __asm__ __volatile__
00257     (
00258         "repe; scasq          \n\t"
00259         "je     1f            \n\t"
00260         "xor    -8(%%rdi),%%rax \n\t"
00261         "sub    $8,%%rdi        \n\t"
00262         "bsf    %%rax,%%rdx     \n\t"
00263         "1:          \n\t"
00264         "sub    %%rsi,%%rdi     \n\t"
00265         "shl    $3,%%rdi        \n\t"
00266         "add    %%rdi,%%rdx     \n\t"
00267         :
00268         "=a" (dummy0), "=c" (dummy1), "=d" (res), "=D" (dummy2)
00269         :
00270         "a" (~0UL), "c" ((size + 63) >> 6), "d" (0), "D" (dest), "S" (dest)
00271         :
00272         "cc", "memory");
00273
00274     return res;
00275 }
00276
00277 L4_END_DECLS

```

16.154 arm/l4/util/bitops_arch.h File Reference

ARM specific implementation of bitops functions.

16.154.1 Detailed Description

ARM specific implementation of bitops functions.

Definition in file [bitops_arch.h](#).

16.155 bitops_arch.h

[Go to the documentation of this file.](#)

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  * License: see LICENSE.spdx (in this directory or the directories above)
00009  */
00010 #ifndef __L4UTIL__ARCH_ARM__BITOPS_ARCH_H__
00011 #define __L4UTIL__ARCH_ARM__BITOPS_ARCH_H__
00012
00013
00014 #endif /* ! __L4UTIL__ARCH_ARM__BITOPS_ARCH_H__ */
```

16.156 x86/l4/util/bitops_arch.h File Reference

x86 bit manipulation functions

16.156.1 Detailed Description

x86 bit manipulation functions

Definition in file [bitops_arch.h](#).

16.157 bitops_arch.h

[Go to the documentation of this file.](#)

```
00001 /*
00002  * Copyright (C) 2000-2009 Technische Universität Dresden (Germany)
00003  * Copyright (C) 2016, 2022, 2024-2025 Kernkonzept GmbH. All rights reserved.
00004  * Author(s): Lars Reuther <reuther@os.inf.tu-dresden.de>
00005  *             Frank Mehnert <frank.mehnert@kernkonzept.com>
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009
00015
00016 #pragma once
00017
00018 L4_BEGIN_DECLS
00019
00020 /* set bit */
00021 #define __L4UTIL_BITOPS_HAVE_ARCH_SET_BIT
00022 L4_INLINE void
00023 l4util_set_bit(int b, volatile l4_umword_t * dest)
00024 {
00025     __asm__ __volatile__
00026     (
00027         "lock; btsl  %1,%0  \n\t"
00028         :
00029         :
00030         "m"    (*dest),    /* 0 mem, destination operand */
00031         "Ir"    (b)        /* 1,    bit number */
00032     );
00033 }
```



```

00032     :
00033     "memory", "cc"
00034 );
00035 }
00036
00037 /* clear bit */
00038 #define __L4UTIL_BITOPS_HAVE_ARCH_CLEAR_BIT
00039 L4_INLINE void
00040 l4util_clear_bit(int b, volatile l4_umword_t * dest)
00041 {
00042     __asm__ __volatile__
00043     (
00044         "lock; btrl %1,%0    \n\t"
00045         :
00046         :
00047         "m"    (*dest), /* 0 mem, destination operand */
00048         "Ir"    (b)      /* 1,    bit number */
00049         :
00050         "memory", "cc"
00051         );
00052 }
00053
00054 /* change bit */
00055 #define __L4UTIL_BITOPS_HAVE_ARCH_COMPLEMENT_BIT
00056 L4_INLINE void
00057 l4util_complement_bit(int b, volatile l4_umword_t * dest)
00058 {
00059     __asm__ __volatile__
00060     (
00061         "lock; btc1 %1,%0    \n\t"
00062         :
00063         :
00064         "m"    (*dest), /* 0 mem, destination operand */
00065         "Ir"    (b)      /* 1,    bit number */
00066         :
00067         "memory", "cc"
00068         );
00069 }
00070
00071 /* test bit */
00072 #define __L4UTIL_BITOPS_HAVE_ARCH_TEST_BIT
00073 L4_INLINE int
00074 l4util_test_bit(int b, const volatile l4_umword_t * dest)
00075 {
00076     l4_int8_t bit;
00077
00078     __asm__ __volatile__
00079     (
00080         "btl %2,%1    \n\t"
00081         :
00082         "=@ccc" (bit) /* 0,    old bit value */
00083         :
00084         "m"    (*dest), /* 1 mem, destination operand */
00085         "Ir"    (b)      /* 2,    bit number */
00086         :
00087         "memory"
00088         );
00089
00090     return bit;
00091 }
00092
00093 /* bit test and set */
00094 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_SET
00095 L4_INLINE int
00096 l4util_bts(int b, volatile l4_umword_t * dest)
00097 {
00098     l4_int8_t bit;
00099
00100     __asm__ __volatile__
00101     (
00102         "lock; btsl %2,%1    \n\t"
00103         :
00104         "=@ccc" (bit) /* 0,    old bit value */
00105         :
00106         "m"    (*dest), /* 1 mem, destination operand */
00107         "Ir"    (b)      /* 2,    bit number */
00108         :
00109         "memory"
00110         );
00111
00112     return bit;
00113 }
00114
00115 /* bit test and reset */
00116 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_RESET
00117 L4_INLINE int
00118 l4util_btr(int b, volatile l4_umword_t * dest)

```

```

00119 {
00120     l4_int8_t bit;
00121
00122     __asm__ __volatile__
00123     (
00124         "lock; btrl %2,%1 \n\t"
00125         :
00126         "=@ccc" (bit) /* 0, old bit value */
00127         :
00128         "m" (*dest), /* 1 mem, destination operand */
00129         "Ir" (b) /* 2, bit number */
00130         :
00131         "memory"
00132     );
00133
00134     return bit;
00135 }
00136
00137 /* bit test and complement */
00138 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_COMPLEMENT
00139 L4_INLINE int
00140 l4util_btc(int b, volatile l4_umword_t * dest)
00141 {
00142     l4_int8_t bit;
00143
00144     __asm__ __volatile__
00145     (
00146         "lock; btc1 %2,%1 \n\t"
00147         :
00148         "=@ccc" (bit) /* 0, old bit value */
00149         :
00150         "m" (*dest), /* 1 mem, destination operand */
00151         "Ir" (b) /* 2, bit number */
00152         :
00153         "memory"
00154     );
00155
00156     return bit;
00157 }
00158
00159 /* bit scan reverse */
00160 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_SCAN_REVERSE
00161 L4_INLINE int
00162 l4util_bsr(l4_umword_t word)
00163 {
00164     int tmp;
00165
00166     if (L4_UNLIKELY(word == 0))
00167         return -1;
00168
00169     __asm__ __volatile__
00170     (
00171         "bsrl %1,%0 \n\t"
00172         :
00173         "=r" (tmp) /* 0, index of most significant set bit */
00174         :
00175         "r" (word) /* 1, argument */
00176     );
00177
00178     return tmp;
00179 }
00180
00181 /* bit scan forward */
00182 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_SCAN_FORWARD
00183 L4_INLINE int
00184 l4util_bsf(l4_umword_t word)
00185 {
00186     int tmp;
00187
00188     if (L4_UNLIKELY(word == 0))
00189         return -1;
00190
00191     __asm__ __volatile__
00192     (
00193         "bsfl %1,%0 \n\t"
00194         :
00195         "=r" (tmp) /* 0, index of least significant set bit */
00196         :
00197         "r" (word) /* 1, argument */
00198     );
00199
00200     return tmp;
00201 }
00202
00203 #define __L4UTIL_BITOPS_HAVE_ARCH_FIND_FIRST_SET_BIT
00204 L4_INLINE int
00205 l4util_find_first_set_bit(const void * dest, l4_size_t size)

```

```

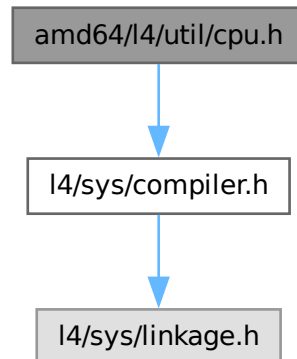
00206 {
00207     l4_mword_t dummy0, dummy1, res;
00208
00209     __asm__ __volatile__
00210     (
00211         "repe; scasl                \n\t"
00212         "jz     lf                 \n\t"
00213         "lea    -4(%%edi),%%edi    \n\t"
00214         "bsf    (%%edi),%%eax      \n\t"
00215         "1:                                     \n\t"
00216         "sub    %%esi,%%edi        \n\t"
00217         "shl    $3,%%edi          \n\t"
00218         "add    %%edi,%%eax        \n\t"
00219         :
00220         "=a" (res), "=c" (dummy0), "=D" (dummy1)
00221         :
00222         "a" (0), "c" ((size + 31) >> 5), "D" (dest), "S" (dest)
00223         :
00224         "cc", "memory");
00225
00226     return res;
00227 }
00228
00229 #define __L4UTIL_BITOPS_HAVE_ARCH_FIND_FIRST_ZERO_BIT
00230 L4_INLINE int
00231 l4util_find_first_zero_bit(const void * dest, l4_size_t size)
00232 {
00233     l4_mword_t dummy0, dummy1, dummy2, res;
00234
00235     if (!size)
00236         return 0;
00237
00238     __asm__ __volatile__
00239     (
00240         "repe;  scasl                \n\t"
00241         "je     lf                 \n\t"
00242         "xor    -4(%%edi),%%eax     \n\t"
00243         "sub    $4,%%edi            \n\t"
00244         "bsf    %%eax,%%edx         \n\t"
00245         "1:                                     \n\t"
00246         "sub    %%esi,%%edi        \n\t"
00247         "shl    $3,%%edi          \n\t"
00248         "add    %%edi,%%edx        \n\t"
00249         :
00250         "=a" (dummy0), "=c" (dummy1), "=d" (res), "=D" (dummy2)
00251         :
00252         "a" (~0UL), "c" ((size + 31) >> 5), "d" (0), "D" (dest), "S" (dest)
00253         :
00254         "cc", "memory");
00255
00256     return res;
00257 }
00258
00259 L4_END_DECLS

```

16.158 amd64/l4/util/cpu.h File Reference

CPU related functions.

```
#include <l4/sys/compiler.h>  
Include dependency graph for cpu.h:
```



Functions

- int [l4util_cpu_has_cpuid](#) (void)
Check whether the CPU supports the "cpuid" instruction.
- unsigned int [l4util_cpu_capabilities](#) (void)
Returns the CPU capabilities if the "cpuid" instruction is available.
- unsigned int [l4util_cpu_capabilities_nocheck](#) (void)
Returns the CPU capabilities.
- void **[l4util_cpu_cpuid](#)** (unsigned long mode, unsigned long *eax, unsigned long *ebx, unsigned long *ecx, unsigned long *edx)
Generic CPUID access function.

16.158.1 Detailed Description

CPU related functions.

Author

Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file [cpu.h](#).

16.159 cpu.h

[Go to the documentation of this file.](#)

```

00001
00006
00007 /*
00008  * (c) 2004-2009 Author(s)
00009  *     economic rights: Technische Universität Dresden (Germany)
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012
00013 #ifndef __L4_UTIL_CPU_H
00014 #define __L4_UTIL_CPU_H
00015
00016 #include <l4/sys/compiler.h>
00017
00018 L4_BEGIN_DECLS
00019
00025
00031 L4_INLINE int          l4util_cpu_has_cpuid(void);
00032
00039 L4_INLINE unsigned int l4util_cpu_capabilities(void);
00040
00046 L4_INLINE unsigned int l4util_cpu_capabilities_nocheck(void);
00047
00051 L4_INLINE void
00052 l4util_cpu_cpuid(unsigned long mode,
00053                 unsigned long *eax, unsigned long *ebx,
00054                 unsigned long *ecx, unsigned long *edx);
00055
00057 static inline void
00058 l4util_cpu_pause(void)
00059 {
00060     __asm__ __volatile__ ("rep; nop");
00061 }
00062
00063 L4_INLINE int
00064 l4util_cpu_has_cpuid(void)
00065 {
00066     return 1;
00067 }
00068
00069 L4_INLINE void
00070 l4util_cpu_cpuid(unsigned long mode,
00071                 unsigned long *eax, unsigned long *ebx,
00072                 unsigned long *ecx, unsigned long *edx)
00073 {
00074     asm volatile("cpuid"
00075                 : "=a" (*eax),
00076                   "=b" (*ebx),
00077                   "=c" (*ecx),
00078                   "=d" (*edx)
00079                 : "a" (mode)
00080                 );
00081 }
00082
00083 L4_INLINE unsigned int
00084 l4util_cpu_capabilities_nocheck(void)
00085 {
00086     unsigned long dummy, capability;
00087
00088     /* get CPU capabilities */
00089     l4util_cpu_cpuid(1, &dummy, &dummy, &dummy, &capability);
00090
00091     return capability;
00092 }
00093
00094 L4_INLINE unsigned int
00095 l4util_cpu_capabilities(void)
00096 {
00097     if (!l4util_cpu_has_cpuid())
00098         return 0; /* CPU has not cpuid instruction */
00099
00100     return l4util_cpu_capabilities_nocheck();
00101 }
00102
00103 L4_END_DECLS
00104
00105 #endif
00106

```

16.160 arm/l4/util/cpu.h File Reference

CPU related functions.

16.160.1 Detailed Description

CPU related functions.

Author

Adam Lackorzynski adam@os.inf.tu-dresden.de

Definition in file [cpu.h](#).

16.161 cpu.h

[Go to the documentation of this file.](#)

```

00001
00007
00008 /*
00009  * (c) 2004-2009 Author(s)
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013
00014 #ifndef __L4_UTIL_ARCH_ARM_CPU_H__
00015 #define __L4_UTIL_ARCH_ARM_CPU_H__
00016
00017 /* Nothing yet */
00018
00019 #endif /* __L4_UTIL_ARCH_ARM_CPU_H__ */

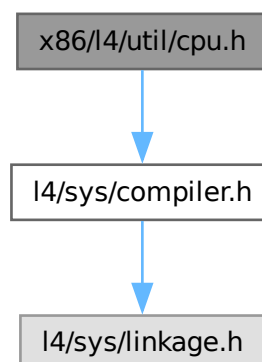
```

16.162 x86/l4/util/cpu.h File Reference

CPU related functions.

```
#include <l4/sys/compiler.h>
```

Include dependency graph for cpu.h:



Functions

- int [l4util_cpu_has_cpuid](#) (void)
Check whether the CPU supports the "cpuid" instruction.
- unsigned int [l4util_cpu_capabilities](#) (void)
Returns the CPU capabilities if the "cpuid" instruction is available.
- unsigned int [l4util_cpu_capabilities_noccheck](#) (void)
Returns the CPU capabilities.
- void [l4util_cpu_cpuid](#) (unsigned long mode, unsigned long *eax, unsigned long *ebx, unsigned long *ecx, unsigned long *edx)
Generic CPUID access function.

16.162.1 Detailed Description

CPU related functions.

Author

Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file [cpu.h](#).

16.163 cpu.h

[Go to the documentation of this file.](#)

```

00001
00006
00007 /*
00008  * (c) 2004-2009 Author(s)
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012
00013 #ifndef __L4_UTIL_CPU_H
00014 #define __L4_UTIL_CPU_H
00015
00016 #include <l4/sys/compiler.h>
00017
00018 L4_BEGIN_DECLS
00019
00025
00031 L4_INLINE int          l4util_cpu_has_cpuid(void);
00032
00039 L4_INLINE unsigned int l4util_cpu_capabilities(void);
00040
00046 L4_INLINE unsigned int l4util_cpu_capabilities_noccheck(void);
00047
00051 L4_INLINE void
00052 l4util_cpu_cpuid(unsigned long mode,
00053                  unsigned long *eax, unsigned long *ebx,
00054                  unsigned long *ecx, unsigned long *edx);
00055
00057 static inline void
00058 l4util_cpu_pause(void)
00059 {
00060     __asm__ __volatile__ ("rep; nop");
00061 }
00062
00063 L4_INLINE int
00064 l4util_cpu_has_cpuid(void)
00065 {
00066     unsigned long eax;
00067
00068     asm volatile("pushl %%ebx          \t\n"
00069                  "pushfl          \t\n"
```

```

00070         "popl %%eax    \t\n" /* get eflags */
00071         "movl %%eax, %%ebx \t\n" /* save it */
00072         "xorl $0x200000, %%eax \t\n" /* toggle ID bit */
00073         "pushl %%eax    \t\n"
00074         "popfl         \t\n" /* set again */
00075         "pushfl        \t\n"
00076         "popl %%eax    \t\n" /* get it again */
00077         "xorl %%ebx, %%eax \t\n"
00078         "pushl %%ebx    \t\n"
00079         "popfl         \t\n" /* restore saved flags */
00080         "popl %%ebx    \t\n"
00081         : "a" (eax)
00082         : /* no input */
00083         );
00084
00085     return eax & 0x200000;
00086 }
00087
00088 L4_INLINE void
00089 l4util_cpu_cpuid(unsigned long mode,
00090                 unsigned long *eax, unsigned long *ebx,
00091                 unsigned long *ecx, unsigned long *edx)
00092 {
00093     asm volatile("pushl %%ebx    \t\n"
00094                 "cpuid          \t\n"
00095                 "mov %%ebx, %%esi \t\n"
00096                 "popl %%ebx      \t\n"
00097                 : "=a" (*eax),
00098                   "=S" (*ebx),
00099                   "=c" (*ecx),
00100                   "=d" (*edx)
00101                 : "a" (mode));
00102 }
00103
00104 L4_INLINE unsigned int
00105 l4util_cpu_capabilities_nocheck(void)
00106 {
00107     unsigned long dummy, capability;
00108
00109     /* get CPU capabilities */
00110     l4util_cpu_cpuid(1, &dummy, &dummy, &dummy, &capability);
00111
00112     return capability;
00113 }
00114
00115 L4_INLINE unsigned int
00116 l4util_cpu_capabilities(void)
00117 {
00118     if (!l4util_cpu_has_cpuid())
00119         return 0; /* CPU has not cpuid instruction */
00120
00121     return l4util_cpu_capabilities_nocheck();
00122 }
00123
00124 L4_END_DECLS
00125
00126 #endif
00127

```

16.164 amd64/l4/util/mbi_argv.h File Reference

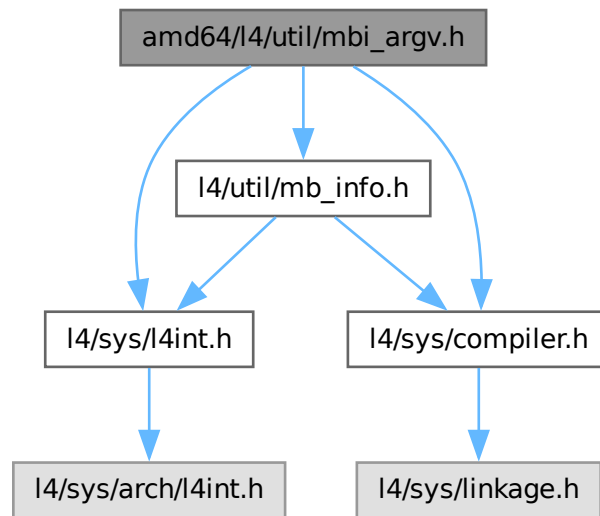
command line handling

```

#include <l4/sys/l4int.h>
#include <l4/util/mb_info.h>
#include <l4/sys/compiler.h>

```


Include dependency graph for mbi_argv.h:



16.164.1 Detailed Description

command line handling

Date

2003

Author

Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file [mbi_argv.h](#).

16.165 mbi_argv.h

[Go to the documentation of this file.](#)

```

00001
00007
00008 /*
00009  * (c) 2003-2009 Author(s)
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013
00014 #ifndef L4UTIL_MBI_ARGV
00015 #define L4UTIL_MBI_ARGV
00016
00017 #include <l4/sys/l4int.h>
00018 #include <l4/util/mb_info.h>

```

```

00019 #include <l4/sys/compiler.h>
00020
00021 L4_BEGIN_DECLS
00022
00023 L4_CV void l4util_mbi_to_argv(l4_mword_t flag, l4util_mb_info_t *mbi);
00024
00025 extern int l4util_argc;
00026 extern char *l4util_argv[];
00027
00028 L4_END_DECLS
00029
00030 #endif
00031

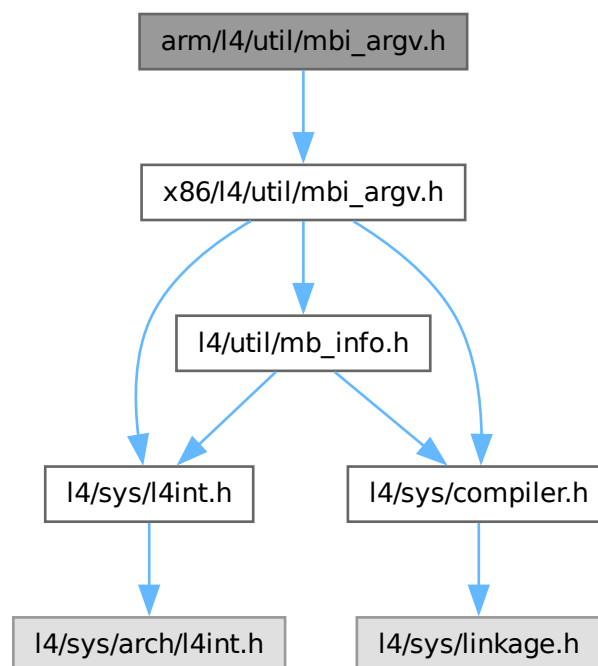
```

16.166 arm/l4/util/mbi_argv.h File Reference

Multiboot.

```
#include <x86/l4/util/mbi_argv.h>
```

Include dependency graph for mbi_argv.h:



16.166.1 Detailed Description

Multiboot.

Definition in file [mbi_argv.h](#).

16.167 mbi_argv.h

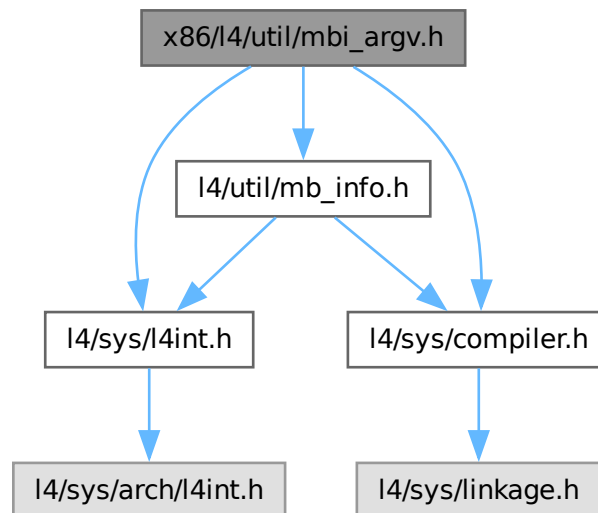
[Go to the documentation of this file.](#)

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  * License: see LICENSE.spdx (in this directory or the directories above)
00009  */
00010 /* If this persists, move it to a generic place */
00011 #include <x86/l4/util/mbi_argv.h>
```

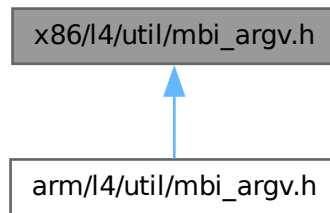
16.168 x86/l4/util/mbi_argv.h File Reference

command line handling

```
#include <l4/sys/l4int.h>
#include <l4/util/mb_info.h>
#include <l4/sys/compiler.h>
Include dependency graph for mbi_argv.h:
```



This graph shows which files directly or indirectly include this file:



16.168.1 Detailed Description

command line handling

Date

2003

Author

Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file [mbi_argv.h](#).

16.169 mbi_argv.h

[Go to the documentation of this file.](#)

```

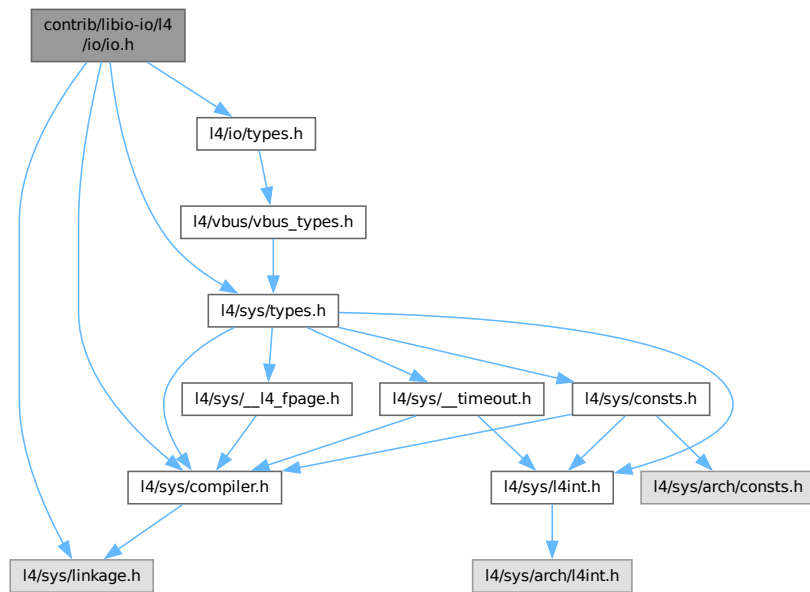
00001
00007
00008 /*
00009  * (c) 2003-2009 Author(s)
00010  *     economic rights: Technische Universität Dresden (Germany)
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013
00014 #ifndef L4UTIL_MBI_ARGV
00015 #define L4UTIL_MBI_ARGV
00016
00017 #include <l4/sys/l4int.h>
00018 #include <l4/util/mb_info.h>
00019 #include <l4/sys/compiler.h>
00020
00021 L4_BEGIN_DECLS
00022
00023 void l4util_mbi_to_argv(l4_mword_t flag, l4util_mb_info_t *mbi);
00024
00025 extern int l4util_argc;
00026 extern char *l4util_argv[];
00027
00028 L4_END_DECLS
00029
00030 #endif
00031

```

16.170 contrib/libio-io/l4/io/io.h File Reference

```
#include <l4/sys/compiler.h>
#include <l4/sys/types.h>
#include <l4/sys/linkage.h>
#include <l4/io/types.h>
```

Include dependency graph for io.h:



Functions

- [l4_cap_idx_t l4io_request_icu](#) (void)
Request the ICU object of the client.
- long [l4io_request_iomem](#) ([l4_addr_t](#) phys, unsigned long size, int flags, [l4_addr_t](#) *virt)
Request an IO memory region.
- long [l4io_request_iomem_region](#) ([l4_addr_t](#) phys, [l4_addr_t](#) virt, unsigned long size, int flags)
Request an IO memory region and map it to a specified region.
- long [l4io_release_iomem](#) ([l4_addr_t](#) virt, unsigned long size)
Release an IO memory region.
- long [l4io_request_ioport](#) (unsigned portnum, unsigned len)
Request an IO port region.
- long [l4io_release_ioport](#) (unsigned portnum, unsigned len)
Release an IO port region.
- [l4io_device_handle_t l4io_get_root_device](#) (void)
Get root device handle of the device bus.
- int [l4io_iterate_devices](#) ([l4io_device_handle_t](#) *devhandle, [l4io_device_t](#) *dev, [l4io_resource_handle_t](#) *reshandle)
Iterate over the device bus.
- int [l4io_lookup_device](#) (const char *devname, [l4io_device_handle_t](#) *dev_handle, [l4io_device_t](#) *dev, [l4io_resource_handle_t](#) *res_handle)

Find a device by name.

- int [l4io_lookup_resource](#) (l4io_device_handle_t devhandle, enum [l4io_resource_types_t](#) type, l4io_resource_handle_t *reshandle, [l4io_resource_t](#) *res)

Request a specific resource from a device description.

- [l4_addr_t](#) [l4io_request_resource_iomem](#) (l4io_device_handle_t devhandle, l4io_resource_handle_t *reshandle)

Request IO memory.

- void [l4io_request_all_ioports](#) (void(*res_cb)([l4vbus_resource_t](#) const *res))

Request all available IO-port resources.

- int [l4io_has_resource](#) (enum [l4io_resource_types_t](#) type, [l4vbus_paddr_t](#) start, [l4vbus_paddr_t](#) end)

Check if a resource is available.

16.170.1 Function Documentation

16.170.1.1 l4io_get_root_device()

```
l4io_device_handle_t l4io_get_root_device (
    void )    [inline]
```

Get root device handle of the device bus.

Returns

root device handle

Definition at line 257 of file [io.h](#).

16.170.1.2 l4io_iterate_devices()

```
int l4io_iterate_devices (
    l4io_device_handle_t * devhandle,
    l4io\_device\_t * dev,
    l4io_resource_handle_t * reshandle)
```

Iterate over the device bus.

Parameters

| | | |
|---------|------------------|---|
| in, out | <i>devhandle</i> | Device handle to start iterating at. The next device handle is returned here. |
| out | <i>dev</i> | Device information, may be NULL. |
| out | <i>reshandle</i> | Resource handle. |

Returns

0 on success, error code otherwise

References [L4_CV](#), and [L4_EXPORT](#).

16.170.1.3 l4io_request_all_ioports()

```
void l4io_request_all_ioports (
    void(* res_cb ) (l4vbus_resource_t const *res))
```

Request all available IO-port resources.

Parameters

| | |
|---------------------|--|
| <code>res_cb</code> | Callback function called for every port resource found, give NULL for no callback. |
|---------------------|--|

References [L4_CV](#), and [L4_EXPORT](#).

16.170.1.4 l4io_request_icu()

```
l4_cap_idx_t l4io_request_icu (
    void )
```

Request the ICU object of the client.

Returns

Client ICU object, an invalid capability selector is returned if no ICU is available.

References [L4_CV](#), and [L4_EXPORT](#).

16.171 io.h

[Go to the documentation of this file.](#)

```
00001
00004 /*
00005  * (c) 2008-2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00006  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00007  *      economic rights: Technische Universität Dresden (Germany)
00008  * License: see LICENSE.spdx (in this directory or the directories above)
00009  */
00010
00011
00012 #pragma once
00013
00014 #include <l4/sys/compiler.h>
00015 #include <l4/sys/types.h>
00016 #include <l4/sys/linkage.h>
00017 #include <l4/io/types.h>
00018
00019 L4_BEGIN_DECLS
00020
00024
00037 L4_CV long L4_EXPORT
00038 l4io_request_irq(int irqnum, l4_cap_idx_t irqcap);
00039
00046 L4_CV l4_cap_idx_t L4_EXPORT
00047 l4io_request_icu(void);
00048
00060 L4_CV long L4_EXPORT
00061 l4io_release_irq(int irqnum, l4_cap_idx_t irq_cap);
00062
00087 L4_CV long L4_EXPORT
00088 l4io_request_iomem(l4_addr_t phys, unsigned long size, int flags,
00089                   l4_addr_t *virt);
00090
00112 L4_CV long L4_EXPORT
00113 l4io_request_iomem_region(l4_addr_t phys, l4_addr_t virt,
```

```

00114             unsigned long size, int flags);
00115
00123 L4_CV long L4_EXPORT
00124 l4io_release_iomem(l4_addr_t virt, unsigned long size);
00125
00135 L4_CV long L4_EXPORT
00136 l4io_request_ioport(unsigned portnum, unsigned len);
00137
00147 L4_CV long L4_EXPORT
00148 l4io_release_ioport(unsigned portnum, unsigned len);
00149
00150
00151 /* ----- Device handling ----- */
00152
00158 L4_INLINE
00159 l4io_device_handle_t l4io_get_root_device(void);
00160
00171 L4_CV int L4_EXPORT
00172 l4io_iterate_devices(l4io_device_handle_t *devhandle,
00173                     l4io_device_t *dev, l4io_resource_handle_t *reshandle);
00174
00187 L4_CV int L4_EXPORT
00188 l4io_lookup_device(const char *devname,
00189                   l4io_device_handle_t *dev_handle,
00190                   l4io_device_t *dev, l4io_resource_handle_t *res_handle);
00191
00207 L4_CV int L4_EXPORT
00208 l4io_lookup_resource(l4io_device_handle_t devhandle,
00209                     enum l4io_resource_types_t type,
00210                     l4io_resource_handle_t *reshandle,
00211                     l4io_resource_t *res);
00212
00213
00214 /* ----- Convenience functions ----- */
00215
00228 L4_CV l4_addr_t L4_EXPORT
00229 l4io_request_resource_iomem(l4io_device_handle_t devhandle,
00230                             l4io_resource_handle_t *reshandle);
00231
00238 L4_CV void L4_EXPORT
00239 l4io_request_all_ioports(void (*res_cb)(l4vbus_resource_t const *res));
00240
00249 L4_CV int L4_EXPORT
00250 l4io_has_resource(enum l4io_resource_types_t type,
00251                  l4vbus_paddr_t start, l4vbus_paddr_t end);
00252
00253 /* ----- */
00254 /* Implementations */
00255
00256 L4_INLINE
00257 l4io_device_handle_t l4io_get_root_device(void)
00258 { return 0; }
00259
00260 L4_END_DECLS

```

16.172 l4/cxx/alloc.h File Reference

Alloc list.

Data Structures

- class [L4::Alloc_list](#)
A simple list-based allocator.

Namespaces

- namespace [L4](#)
L4 low-level kernel interface.

16.172.1 Detailed Description

Alloc list.

Definition in file [alloc.h](#).

16.173 alloc.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2004-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00007  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #pragma once
00013
00014 namespace L4 {
00015
00020     class Alloc_list
00021     {
00022     public:
00023         Alloc_list() : _free(0) {}
00024         Alloc_list(void *blk, unsigned long size) : _free(0)
00025         { free( blk, size); }
00026
00027         void free(void *blk, unsigned long size);
00028         void *alloc(unsigned long size);
00029
00030     private:
00031         struct Elem
00032         {
00033             Elem *next;
00034             unsigned long size;
00035         };
00036
00037         Elem *_free;
00038     };
00039 }
```

16.174 arith

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008
00009 #pragma once
00010
00011 namespace cxx { namespace arith {
00012
00027     template<typename N, typename D>
00028     constexpr N
00029     div_ceil(N const &n, D const &d)
00030     {
00031         // Since C++11 the "quotient is truncated towards zero (fractional part is
00032         // discarded)". Thus a negative quotient is already ceiled, whereas a
00033         // positive quotient is floored. Furthermore, since C++11 the sign of the
00034         // % operator is no longer implementation defined, thus we can use n % d to
00035         // detect if the quotient is positive (n % d >= 0) and was truncated (n % d !=
00036         // 0). In that case, we add one to round to the next largest integer.
00037         return n / d + (n % d > 0);
00038     }
00039
00047     template< unsigned long V >
00048     struct Ld
00049     {
00050         enum { value = Ld<V / 2>::value + 1 };
00051     };
00052 }
```

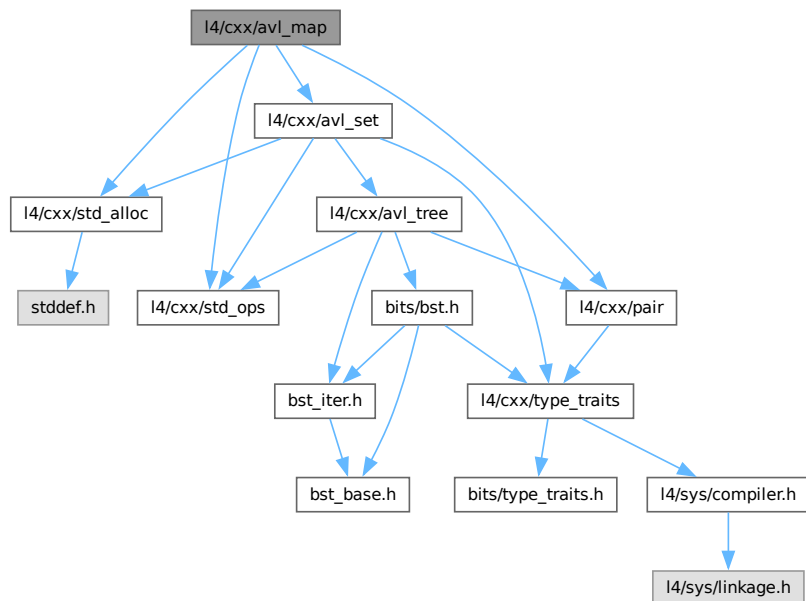
```
00051 };
00052
00053 template<>
00054 struct Ld<0>
00055 {
00056     enum { value = ~0UL };
00057 };
00058
00059 template<>
00060 struct Ld<1>
00061 {
00062     enum { value = 0 };
00063 };
00064
00072 constexpr unsigned
00073 log2u(unsigned val)
00074 {
00075     return 8 * sizeof(val) - __builtin_clz(val) - 1;
00076 }
00077
00079 constexpr unsigned
00080 log2u(unsigned long val)
00081 {
00082     return 8 * sizeof(val) - __builtin_clzl(val) - 1;
00083 }
00084
00086 constexpr unsigned
00087 log2u(unsigned long long val)
00088 {
00089     return 8 * sizeof(val) - __builtin_clzll(val) - 1;
00090 }
00091
00100 template<typename T>
00101 constexpr unsigned
00102 log2u_ceil(T val)
00103 {
00104     return val == 1 ? 0 : log2u(val - 1) + 1;
00105 }
00106
00107 }
```

16.175 l4/cxx/avl_map File Reference

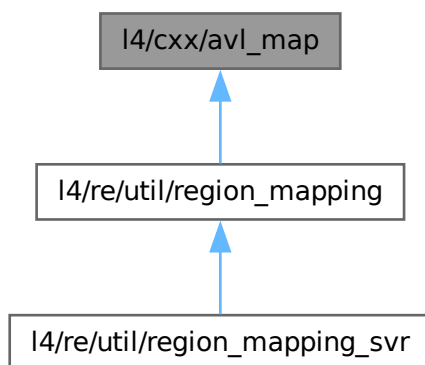
AVL map.

```
#include <l4/cxx/std_alloc>
#include <l4/cxx/std_ops>
#include <l4/cxx/pair>
#include <l4/cxx/avl_set>
```

Include dependency graph for avl_map:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct `cxx::Bits::Avl_map_get_key< KEY_TYPE >`
Key-getter for `Avl_map`.
- class `cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >`
AVL tree based associative container.

Namespaces

- namespace `cxx`
Our C++ library.
- namespace `cxx::Bits`
Internal helpers for the cxx package.

16.175.1 Detailed Description

AVL map.

Definition in file [avl_map](#).

16.176 avl_map

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008
00009 #pragma once
00010
00011 #include <l4/cxx/std_alloc>
00012 #include <l4/cxx/std_ops>
00013 #include <l4/cxx/pair>
00014 #include <l4/cxx/avl_set>
00015
00016 namespace cxx {
00017     namespace Bits {
00018
00019         template<typename KEY_TYPE>
00020         struct Avl_map_get_key
00021         {
00022             typedef KEY_TYPE Key_type;
00023             template<typename NODE>
00024             static Key_type const &key_of(NODE const *n)
00025             { return n->item.first; }
00026         };
00027
00028         template< typename KEY_TYPE, typename DATA_TYPE,
00029                 template<typename A> class COMPARE = Lt_functor,
00030                 template<typename B> class ALLOC = New_allocator >
00031         class Avl_map :
00032         public Bits::Base_avl_set<Pair<KEY_TYPE, DATA_TYPE>,
00033                                 COMPARE<KEY_TYPE>, ALLOC,
00034                                 Bits::Avl_map_get_key<KEY_TYPE> >
00035         {
00036         private:
00037             typedef Pair<KEY_TYPE, DATA_TYPE> Local_item_type;
00038             typedef Bits::Base_avl_set<Local_item_type, COMPARE<KEY_TYPE>, ALLOC,
00039                                     Bits::Avl_map_get_key<KEY_TYPE> > Base_type;
00040         public:
00041             typedef COMPARE<KEY_TYPE> Key_compare;
00042             typedef KEY_TYPE Key_type;
00043             typedef DATA_TYPE Data_type;
00044             typedef typename Base_type::Node Node;
00045             typedef typename Base_type::Node_allocator Node_allocator;
00046
00047             typedef typename Base_type::Iterator Iterator;
00048             typedef typename Base_type::Iterator iterator;
00049             typedef typename Base_type::Const_iterator Const_iterator;
00050             typedef typename Base_type::Const_iterator const_iterator;
00051             typedef typename Base_type::Rev_iterator Rev_iterator;
00052             typedef typename Base_type::Rev_iterator reverse_iterator;
00053             typedef typename Base_type::Const_rev_iterator Const_rev_iterator;
00054             typedef typename Base_type::Const_rev_iterator const_reverse_iterator;

```

```

00075
00080  Avl_map(Node_allocator const &alloc = Node_allocator())
00081      : Base_type(alloc)
00082  {}
00083
00099  cxx::Pair<Iterator, int> insert(Key_type const &key, Data_type const &data)
00100  { return Base_type::insert(Pair<Key_type, Data_type>(key, data)); }
00101
00116  template<typename... Args>
00117  cxx::Pair<Iterator, int> emplace(Args &&...args)
00118  { return Base_type::emplace(cxx::forward<Args>(args)...); }
00119
00125  Data_type const &operator [] (Key_type const &key) const
00126  { return this->find_node(key)->second; }
00127
00137  Data_type &operator [] (Key_type const &key)
00138  {
00139      Node n = this->find_node(key);
00140      if (n)
00141          return const_cast<Data_type&>(n->second);
00142      else
00143          return insert(key, Data_type()).first->second;
00144  }
00145 };
00146
00147 }
00148

```

16.177 l4/cxx/avl_set File Reference

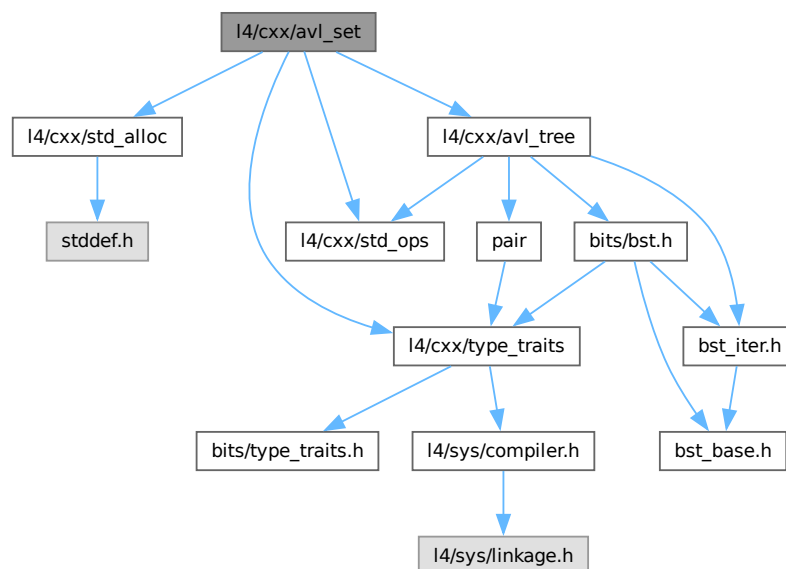
AVL set.

```

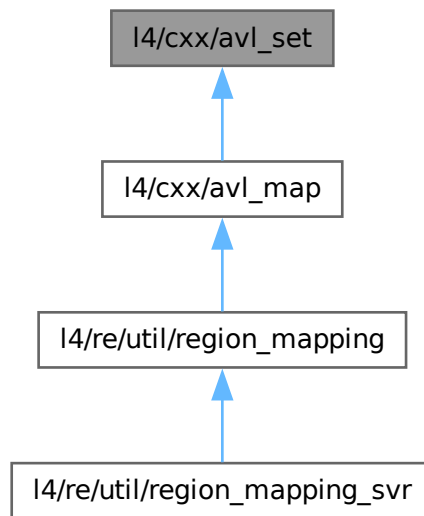
#include <l4/cxx/std_alloc>
#include <l4/cxx/std_ops>
#include <l4/cxx/type_traits>
#include <l4/cxx/avl_tree>

```

Include dependency graph for avl_set:



This graph shows which files directly or indirectly include this file:



Data Structures

- class [cxx::Bits::Avl_set_iter](#)< Node, Key, Node_op >
Generic iterator for the AVL-tree based set.
- struct [cxx::Bits::Avl_set_get_key](#)< KEY_TYPE >
Internal, key-getter for [Avl_set](#) nodes.
- class [cxx::Bits::Base_avl_set](#)< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >
AVL set with internally managed nodes.
- class [cxx::Bits::Base_avl_set](#)< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node
A smart pointer to a tree item.
- class [cxx::Avl_set](#)< ITEM_TYPE, COMPARE, ALLOC >
AVL set for simple comparable items.

Namespaces

- namespace [cxx](#)
Our C++ library.
- namespace [cxx::Bits](#)
Internal helpers for the cxx package.

16.177.1 Detailed Description

AVL set.

Definition in file [avl_set](#).

16.178 avl_set

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00008  *      Carsten Weinhold <weinhold@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013
00014 #pragma once
00015
00016 #include <l4/cxx/std_alloc>
00017 #include <l4/cxx/std_ops>
00018 #include <l4/cxx/type_traits>
00019 #include <l4/cxx/avl_tree>
00020
00021 struct Avl_set_tester;
00022
00023 namespace cxx {
00024 namespace Bits {
00025
00034 template<typename Node, typename Key, typename Node_op>
00035 class Avl_set_iter : public __Bst_iter_b<Node, Node_op>
00036 {
00037 private:
00039     typedef __Bst_iter_b<Node, Node_op> Base;
00040
00042     typedef typename Type_traits<Key>::Non_const_type Non_const_key;
00043
00045     typedef Avl_set_iter<Node, Non_const_key, Node_op> Non_const_iter;
00046
00047     using Base::_n;
00048     using Base::_r;
00049     using Base::inc;
00050
00051 public:
00053     Avl_set_iter() = default;
00054
00059     Avl_set_iter(Node const *t) : Base(t) {}
00060
00065     Avl_set_iter(Base const &o) : Base(o) {}
00066
00068     Avl_set_iter(Non_const_iter const &o)
00069     : Base(o) {}
00070
00072     Avl_set_iter &operator = (Non_const_iter const &o)
00073     { Base::operator = (o); return *this; }
00074
00080     Key &operator * () const { return const_cast<Node*>(_n)->item; }
00081
00087     Key *operator -> () const { return &const_cast<Node*>(_n)->item; }
00088
00092     Avl_set_iter &operator ++ () { inc(); return *this; }
00093
00097     Avl_set_iter operator ++ (int)
00098     { Avl_set_iter tmp = *this; inc(); return tmp; }
00099 };
00100
00102 template<typename KEY_TYPE>
00103 struct Avl_set_get_key
00104 {
00105     typedef KEY_TYPE Key_type;
00106     template<typename NODE>
00107     static Key_type const &key_of(NODE const *n)
00108     { return n->item; }
00109 };
00110
00124 template<typename ITEM_TYPE, class COMPARE,
00125         template<typename A> class ALLOC,
00126         typename GET_KEY>
00127 class Base_avl_set
00128 {
00129     friend struct ::Avl_set_tester;
00130
00131 public:
00132     enum
00133     {
00134         E_noent = 2,
00141         E_exist = 17,
00142         E_nomem = 12,
00143         E_inval = 22
00144     };

```

```

00145
00147     typedef ITEM_TYPE Item_type;
00148
00150     typedef GET_KEY Get_key;
00151
00153     typedef typename GET_KEY::Key_type Key_type;
00154
00156     typedef typename Type_traits<Item_type>::Const_type Const_item_type;
00157
00159     typedef COMPARE Item_compare;
00160
00161 private:
00163     class _Node : public Avl_tree_node
00164     {
00165     public:
00167         Item_type item;
00168
00169         _Node() = default;
00170
00171         _Node(Item_type const &item) : Avl_tree_node(), item(item) {}
00172
00173         template<typename ...ARGS>
00174         _Node(ARGS &&...args) : Avl_tree_node(), item(cxx::forward<ARGS>(args)...)
00175         {}
00176     };
00177
00178 public:
00182     class Node
00183     {
00184     private:
00185         friend class Base_avl_set<ITEM_TYPE, COMPARE, ALLOC, GET_KEY>;
00186
00187         _Node const *_n;
00188
00189         explicit Node(_Node const *n) : _n(n) {}
00190
00191     public:
00193         Node() : _n(0) {}
00194
00200         Item_type const &operator * () { return _n->item; }
00201
00207         Item_type const *operator -> () { return _n->item; }
00208
00214         bool valid() const { return _n; }
00215
00217         operator Item_type const * () { return _n ? _n->item : 0; }
00218     };
00219
00221     typedef ALLOC<_Node> Node_allocator;
00222
00223 private:
00224     typedef Avl_tree<_Node, GET_KEY, COMPARE> Tree;
00225
00226     Tree _tree;
00227
00229     Node_allocator _alloc;
00230
00231     typedef typename Tree::Fwd_iter_ops Fwd;
00232     typedef typename Tree::Rev_iter_ops Rev;
00233
00239     void deep_copy(Base_avl_set const &o)
00240     {
00241         for (Const_iterator i = o.begin(); i != o.end(); ++i)
00242             insert(*i);
00243     }
00244
00245 public:
00246     typedef typename Type_traits<Item_type>::Param_type Item_param_type;
00247
00249     typedef Avl_set_iter<_Node, Item_type, Fwd> Iterator;
00250     typedef Iterator iterator;
00252     typedef Avl_set_iter<_Node, Const_item_type, Fwd> Const_iterator;
00253     typedef Const_iterator const_iterator;
00255     typedef Avl_set_iter<_Node, Item_type, Rev> Rev_iterator;
00256     typedef Rev_iterator reverse_iterator;
00258     typedef Avl_set_iter<_Node, Const_item_type, Rev> Const_rev_iterator;
00259     typedef Const_rev_iterator const_reverse_iterator;
00260
00268     explicit Base_avl_set(Node_allocator const &alloc = Node_allocator())
00269         : _tree(), _alloc(alloc)
00270     {}
00271
00272     ~Base_avl_set()
00273     { clear(); }
00274
00283     Base_avl_set(Base_avl_set const &o,
00284                 Node_allocator const &alloc = Node_allocator())

```



```

00285     : _tree(), _alloc(alloc)
00286     { deep_copy(o); }
00287
00295 Base_avl_set &operator = (Base_avl_set const &o)
00296 {
00297     if (this != &o)
00298     {
00299         clear();
00300         deep_copy(o);
00301     }
00302
00303     return *this;
00304 }
00305
00323 cxx::Pair<Iterator, int> insert(Item_type const &item);
00324
00343 template<typename... Args>
00344 cxx::Pair<Iterator, int> emplace(Args&&... args);
00345
00349 void clear() noexcept
00350 {
00351     _tree.remove_all([this](_Node *n)
00352     {
00353         n->~_Node();
00354         _alloc.free(n);
00355     });
00356 }
00357
00366 int remove(Key_type const &item)
00367 {
00368     _Node *n = _tree.remove(item);
00369
00370     if (n)
00371     {
00372         n->~_Node();
00373         _alloc.free(n);
00374         return 0;
00375     }
00376
00377     return -E_noent;
00378 }
00379
00384 int erase(Key_type const &item)
00385 { return remove(item); }
00386
00395 Node find_node(Key_type const &item) const
00396 { return Node(_tree.find_node(item)); }
00397
00406 Node lower_bound_node(Key_type const &key) const
00407 { return Node(_tree.lower_bound_node(key)); }
00408
00409 Node lower_bound_node(Key_type &&key) const
00410 { return Node(_tree.lower_bound_node(key)); }
00411
00417 Const_iterator begin() const { return _tree.begin(); }
00418
00424 Const_iterator end() const { return _tree.end(); }
00425
00431 Iterator begin() { return _tree.begin(); }
00432
00438 Iterator end() { return _tree.end(); }
00439
00445 Const_rev_iterator rbegin() const { return _tree.rbegin(); }
00446
00452 Const_rev_iterator rend() const { return _tree.rend(); }
00453
00459 Rev_iterator rbegin() { return _tree.rbegin(); }
00460
00466 Rev_iterator rend() { return _tree.rend(); }
00467
00468 Const_iterator find(Key_type const &item) const
00469 { return _tree.find(item); }
00470
00471 #ifdef __DEBUG_L4_AVL
00472 bool rec_dump(bool print)
00473 {
00474     return _tree.rec_dump(print);
00475 }
00476 #endif
00477 };
00478
00479 //-----
00480 /* Implementation of AVL Tree */
00481
00482 /* Insert new _Node. */
00483 template<typename Item, class Compare, template<typename A> class Alloc, typename KEY_TYPE>
00484 Pair<typename Base_avl_set<Item, Compare, Alloc, KEY_TYPE>::Iterator, int>

```

```

00485 Base_avl_set<Item, Compare, Alloc, KEY_TYPE>::insert(Item const &item)
00486 {
00487     _Node *n = _alloc.alloc();
00488     if (!n)
00489         return cxx::pair(end(), -E_nomem);
00490
00491     new (n, Nothrow()) _Node(item);
00492     Pair<_Node *, bool> err = _tree.insert(n);
00493     if (!err.second)
00494     {
00495         n->~_Node();
00496         _alloc.free(n);
00497     }
00498
00499     return cxx::pair(Iterator(typename Tree::Iterator(err.first, err.first)), err.second ? 0 :
-E_exist);
00500 }
00501
00502 /* In-place insert new _Node. */
00503 template<typename Item, class Compare, template<typename A> class Alloc, typename KEY_TYPE>
00504 template<typename... Args>
00505 Pair<typename Base_avl_set<Item, Compare, Alloc, KEY_TYPE>::Iterator, int>
00506 Base_avl_set<Item, Compare, Alloc, KEY_TYPE>::emplace(Args&&... args)
00507 {
00508     _Node *n = _alloc.alloc();
00509     if (!n)
00510         return cxx::pair(end(), -E_nomem);
00511
00512     new (n, Nothrow()) _Node(cxx::forward<Args>(args)...);
00513     Pair<_Node *, bool> err = _tree.insert(n);
00514     if (!err.second)
00515     {
00516         n->~_Node();
00517         _alloc.free(n);
00518     }
00519
00520     return cxx::pair(Iterator(typename Tree::Iterator(err.first, err.first)), err.second ? 0 :
-E_exist);
00521 }
00522
00523 } // namespace Bits
00524
00525 template<typename ITEM_TYPE, template<typename T> class COMPARE = Lt_functor,
00526         template<typename A> class ALLOC = New_allocator>
00527 class Avl_set :
00528     public Bits::Base_avl_set<ITEM_TYPE, COMPARE<ITEM_TYPE>, ALLOC,
00529                             Bits::Avl_set_get_key<ITEM_TYPE>»
00530 {
00531 private:
00532     typedef Bits::Base_avl_set<ITEM_TYPE, COMPARE<ITEM_TYPE>, ALLOC,
00533                             Bits::Avl_set_get_key<ITEM_TYPE>» Base;
00534 public:
00535     typedef typename Base::Node_allocator Node_allocator;
00536     Avl_set() = default;
00537     Avl_set(Node_allocator const &alloc)
00538         : Base(alloc)
00539     {}
00540 };
00541
00542 } // namespace cxx

```

16.179 I4/cxx/avl_tree File Reference

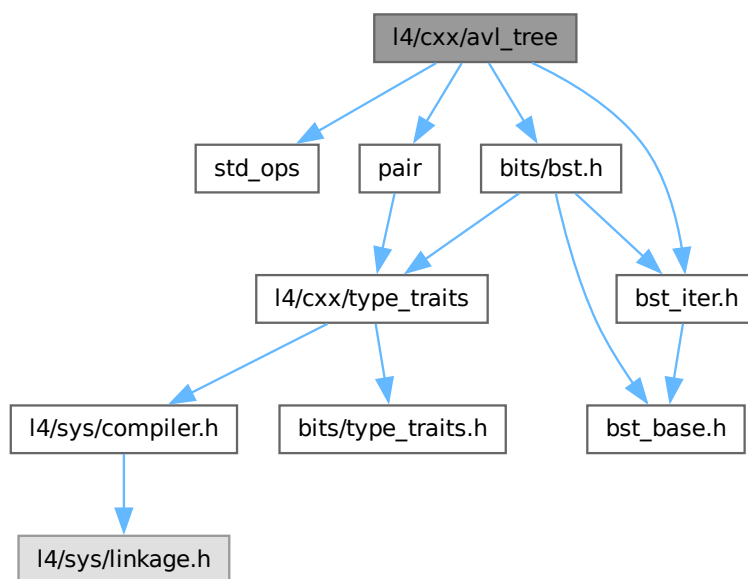
AVL tree.

```

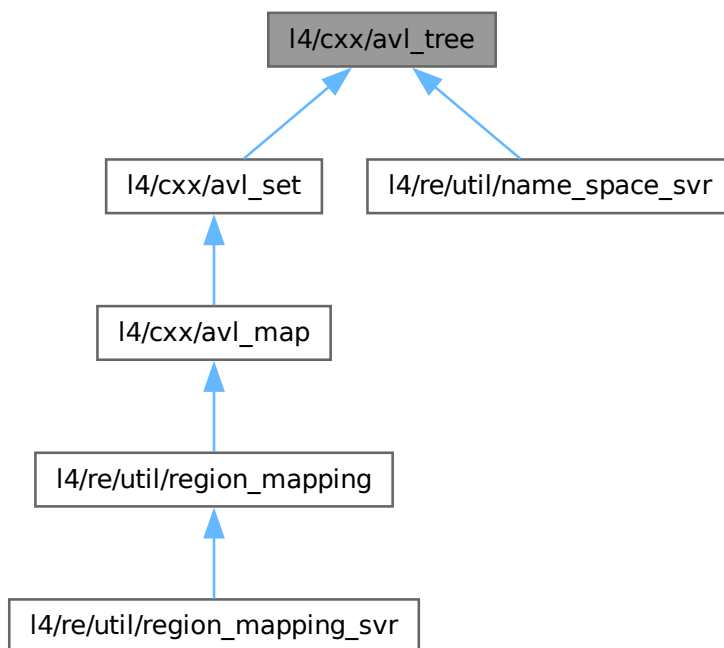
#include "std_ops"
#include "pair"
#include "bits/bst.h"
#include "bits/bst_iter.h"

```

Include dependency graph for avl_tree:



This graph shows which files directly or indirectly include this file:



Data Structures

- class [cxx::Avl_tree_node](#)
Node of an AVL tree.
- class [cxx::Avl_tree< Node, Get_key, Compare >](#)
A generic AVL tree.

Namespaces

- namespace [cxx](#)
Our C++ library.

16.179.1 Detailed Description

AVL tree.

Definition in file [avl_tree](#).

16.180 avl_tree

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00008  *      Carsten Weinhold <weinhold@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013
00014 #pragma once
00015
00016 #include "std_ops"
00017 #include "pair"
00018
00019 #include "bits/bst.h"
00020 #include "bits/bst_iter.h"
00021
00022 struct Avl_set_tester;
00023
00024 namespace cxx {
00025
00029 class Avl_tree_node : public Bits::Bst_node
00030 {
00031     friend struct ::Avl_set_tester;
00032
00033 private:
00034     template< typename Node, typename Get_key, typename Compare >
00035     friend class Avl_tree;
00036
00038     typedef Bits::Direction Bal;
00040     typedef Bits::Direction Dir;
00041
00042     // We are a final BST node, hide interior.
00044     using Bits::Bst_node::next;
00045     using Bits::Bst_node::next_p;
00046     using Bits::Bst_node::rotate;
00048
00050     Bal _balance;
00051
00052 protected:
00054     Avl_tree_node() = default;
00055
00056 private:
00057     Avl_tree_node(Avl_tree_node const &o) = delete;
00058     Avl_tree_node(Avl_tree_node &&o) = delete;
00059

```

```

00061     Avl_tree_node &operator = (Avl_tree_node const &o) = default;
00063     Avl_tree_node &operator = (Avl_tree_node &&o) = default;
00064
00066     explicit Avl_tree_node(bool) : Bits::Bst_node(true), _balance(Dir::N) {}
00067
00069     static Bits::Bst_node *rotate2(Bst_node **t, Bal idir, Bal pre);
00070
00072     bool balanced() const { return _balance == Bal::N; }
00073
00075     Bal balance() const { return _balance; }
00076
00078     void balance(Bal b) { _balance = b; }
00079 };
00080
00081
00098 template< typename Node, typename Get_key,
00099           typename Compare = Lt_functor<typename Get_key::Key_type> >
00100 class Avl_tree : public Bits::Bst<Node, Get_key, Compare>
00101 {
00102 private:
00103     typedef Bits::Bst<Node, Get_key, Compare> Bst;
00104
00106     using Bst::_head;
00107
00109     using Bst::k;
00110
00112     typedef typename Avl_tree_node::Bal Bal;
00114     typedef typename Avl_tree_node::Bal Dir;
00115
00116     Avl_tree(Avl_tree const &o) = delete;
00117     Avl_tree &operator = (Avl_tree const &o) = delete;
00118     Avl_tree(Avl_tree &&o) = delete;
00119     Avl_tree &operator = (Avl_tree &&o) = delete;
00120
00121 public:
00123     typedef typename Bst::Key_type Key_type;
00124     typedef typename Bst::Key_param_type Key_param_type;
00126
00127     // Grab iterator types from Bst
00130     typedef typename Bst::Iterator Iterator;
00132     typedef typename Bst::Const_iterator Const_iterator;
00134     typedef typename Bst::Rev_iterator Rev_iterator;
00136     typedef typename Bst::Const_rev_iterator Const_rev_iterator;
00138
00146     Pair<Node *, bool> insert(Node *new_node);
00147
00154     Node *remove(Key_param_type key);
00158     Node *erase(Key_param_type key) { return remove(key); }
00159
00161     Avl_tree() = default;
00162
00164     ~Avl_tree() noexcept
00165     {
00166         this->remove_all([](Node *){});
00167     }
00168
00169 #ifdef __DEBUG_L4_AVL
00170     bool rec_dump(Avl_tree_node *n, int depth, int *dp, bool print, char pfx);
00171     bool rec_dump(bool print)
00172     {
00173         int dp=0;
00174         return rec_dump(static_cast<Avl_tree_node *>(_head), 0, &dp, print, '+');
00175     }
00176 #endif
00177 };
00178
00179
00180 //-----
00181 /* IMPLEMENTATION: Bits::__Bst_iter_b */
00182
00183
00184 inline
00185 Bits::Bst_node *
00186 Avl_tree_node::rotate2(Bst_node **t, Bal idir, Bal pre)
00187 {
00188     typedef Bits::Bst_node N;
00189     typedef Avl_tree_node A;
00190     N *tmp[2] = { *t, N::next(*t, idir) };
00191     *t = N::next(tmp[1], !idir);
00192     A *n = static_cast<A*>(*t);
00193
00194     N::next(tmp[0], idir, N::next(n, !idir));
00195     N::next(tmp[1], !idir, N::next(n, idir));
00196     N::next(n, !idir, tmp[0]);
00197     N::next(n, idir, tmp[1]);
00198
00199     n->balance(Bal::N);

```

```

00200
00201     if (pre == Bal::N)
00202     {
00203         static_cast<A*>(tmp[0])->balance(Bal::N);
00204         static_cast<A*>(tmp[1])->balance(Bal::N);
00205         return nullptr;
00206     }
00207
00208     static_cast<A*>(tmp[pre != idir])->balance(!pre);
00209     static_cast<A*>(tmp[pre == idir])->balance(Bal::N);
00210
00211     return N::next(tmp[pre == idir], !pre);
00212 }
00213
00214 //-----
00215 /* Implementation of AVL Tree */
00216
00217 /* Insert new _Node. */
00218 template< typename Node, typename Get_key, class Compare>
00219 Pair<Node *, bool>
00220 Avl_tree<Node, Get_key, Compare>::insert(Node *new_node)
00221 {
00222     typedef Avl_tree_node A;
00223     typedef Bits::Bst_node N;
00224     N **t = &_head; /* search variable */
00225     N **s = &_head; /* node where rebalancing may occur */
00226     Key_param_type new_key = Get_key::key_of(new_node);
00227
00228     // search insertion point
00229     for (N *p; (p = *t);)
00230     {
00231         Dir b = this->dir(new_key, p);
00232         if (b == Dir::N)
00233             return pair(static_cast<Node*>(p), false);
00234
00235         if (!static_cast<A const *>(p)->balanced())
00236             s = t;
00237
00238         t = A::next_p(p, b);
00239     }
00240
00241     *static_cast<A*>(new_node) = A(true);
00242     *t = new_node;
00243
00244     N *n = *s;
00245     A *a = static_cast<A*>(n);
00246     if (!a->balanced())
00247     {
00248         A::Bal b(this->greater(new_key, n));
00249         if (a->balance() != b)
00250         {
00251             // ok we got in balance the shorter subtree go higher
00252             a->balance(Bal::N);
00253             // propagate the new balance down to the new node
00254             n = A::next(n, b);
00255         }
00256         else if (b == Bal(this->greater(new_key, A::next(n, b))))
00257         {
00258             // left-left or right-right case -> single rotation
00259             A::rotate(s, b);
00260             a->balance(Bal::N);
00261             static_cast<A*>(*s)->balance(Bal::N);
00262             n = A::next(*s, b);
00263         }
00264         else
00265         {
00266             // need a double rotation
00267             n = A::next(A::next(n, b), !b);
00268             n = A::rotate2(s, b, n == new_node ? Bal::N : Bal(this->greater(new_key, n)));
00269         }
00270     }
00271
00272     for (A::Bal b; n && n != new_node; static_cast<A*>(n)->balance(b), n = A::next(n, b))
00273         b = Bal(this->greater(new_key, n));
00274
00275     return pair(new_node, true);
00276 }
00277
00278
00279 /* remove an element */
00280 template< typename Node, typename Get_key, class Compare>
00281 inline
00282 Node *Avl_tree<Node, Get_key, Compare>::remove(Key_param_type key)
00283 {
00284     typedef Avl_tree_node A;
00285     typedef Bits::Bst_node N;
00286     N **q = &_head; /* search variable */

```

```

00287     N **s = &_head; /* last ('deepest') node on the search path to q
00288                     * with balance 0, at this place the rebalancing
00289                     * stops in any case */
00290     N **t = nullptr;
00291     Dir dir;
00292
00293     // find target node and rebalancing entry
00294     for (N *n; (n = *q); q = A::next_p(n, dir))
00295     {
00296         dir = Dir(this->greater(key, n));
00297         if (dir == Dir::L && !this->greater(k(n), key))
00298             /* found node */
00299             t = q;
00300
00301         if (!A::next(n, dir))
00302             break;
00303
00304         A const *a = static_cast<A const *>(n);
00305         if (a->balanced() || (a->balance() == !dir && A::next<A>(n, !dir)->balanced()))
00306             s = q;
00307     }
00308
00309     // nothing found
00310     if (!t)
00311         return nullptr;
00312
00313     A *i = static_cast<A*>(*t);
00314
00315     for (N *n; (n = *s); s = A::next_p(n, dir))
00316     {
00317         dir = Dir(this->greater(key, n));
00318
00319         if (!A::next(n, dir))
00320             break;
00321
00322         A *a = static_cast<A*>(n);
00323         // got one out of balance
00324         if (a->balanced())
00325             a->balance(!dir);
00326         else if (a->balance() == dir)
00327             a->balance(Bal::N);
00328         else
00329         {
00330             // we need rotations to get in balance
00331             Bal b = A::next<A>(n, !dir)->balance();
00332             if (b == dir)
00333                 A::rotate2(s, !dir, A::next<A>(A::next(n, !dir), dir)->balance());
00334             else
00335             {
00336                 A::rotate(s, !dir);
00337                 if (b != Bal::N)
00338                 {
00339                     a->balance(Bal::N);
00340                     static_cast<A*>(*s)->balance(Bal::N);
00341                 }
00342                 else
00343                 {
00344                     a->balance(!dir);
00345                     static_cast<A*>(*s)->balance(dir);
00346                 }
00347             }
00348             if (n == i)
00349                 t = A::next_p(*s, dir);
00350         }
00351     }
00352
00353     A *n = static_cast<A*>(*q);
00354     *t = n;
00355     *q = A::next(n, !dir);
00356     *n = *i;
00357
00358     return static_cast<Node*>(i);
00359 }
00360
00361 #ifdef __DEBUG_L4_AVL
00362 template< typename Node, typename Get_key, class Compare>
00363 bool Avl_tree<Node, Get_key, Compare>::rec_dump(Avl_tree_node *n, int depth, int *dp, bool print, char
00364 pfx)
00365 {
00366     typedef Avl_tree_node A;
00367
00368     if (!n)
00369         return true;
00370
00371     int dpx[2] = {depth, depth};
00372     bool res = true;
00373

```

```

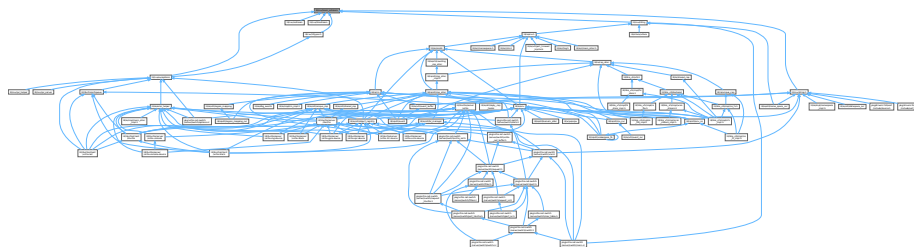
00373     res = rec_dump(A::next<A>(n, Dir::R), depth + 1, dpx + 1, print, '/');
00374
00375     if (print)
00376     {
00377         fprintf(stderr, "%2d: [%8p] b=%1d: ", depth, n, (int)n->balance().d);
00378
00379         for (int i = 0; i < depth; ++i)
00380             std::cerr << "    ";
00381
00382         std::cerr << pfx << (static_cast<Node*>(n)->item) << std::endl;
00383     }
00384
00385     res = res & rec_dump(A::next<A>(n, Dir::L), depth + 1, dpx, print, '\\');
00386
00387     int b = dpx[1] - dpx[0];
00388
00389     if (b < 0)
00390         *dp = dpx[0];
00391     else
00392         *dp = dpx[1];
00393
00394     Bal x = n->balance();
00395     if ((b < -1 || b > 1) ||
00396         (b == 0 && x != Bal::N) ||
00397         (b == -1 && x != Bal::L) ||
00398         (b == 1 && x != Bal::R))
00399     {
00400         if (print)
00401             fprintf(stderr, "%2d: [%8p] b=%1d: balance error %d\n", depth, n, (int)n->balance().d, b);
00402         return false;
00403     }
00404     return res;
00405 }
00406 #endif
00407
00408 }
00409

```

16.181 I4/cxx/basic_ostream File Reference

Basic IO stream.

This graph shows which files directly or indirectly include this file:



Data Structures

- class [L4::IOModifier](#)
Modifier class for the IO stream.

Namespaces

- namespace [L4](#)
L4 low-level kernel interface.

Variables

- `IOModifier` const `L4::hex`
Modifies the stream to print numbers as hexadecimal values.
- `IOModifier` const `L4::dec`
Modifies the stream to print numbers as decimal values.

16.181.1 Detailed Description

Basic IO stream.

Definition in file [basic_ostream](#).

16.182 basic_ostream

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #pragma once
00013
00014 namespace L4 {
00015
00022     class IOModifier
00023     {
00024     public:
00025         IOModifier(int x) : mod(x) {}
00026         bool operator == (IOModifier o) { return mod == o.mod; }
00027         bool operator != (IOModifier o) { return mod != o.mod; }
00028         int mod;
00029     };
00030
00035     class IOBackend
00036     {
00037     public:
00038         typedef int Mode;
00039
00040     protected:
00041         friend class BasicOStream;
00042
00043         IOBackend()
00044         : int_mode(10)
00045         {}
00046
00047         virtual ~IOBackend() {}
00048
00049         virtual void write(char const *str, unsigned len) = 0;
00050
00051     private:
00052         void write(IOModifier m);
00053         void write(long long int c, int len);
00054         void write(long long unsigned c, int len);
00055         void write(long long unsigned c, unsigned char base = 10,
00056                   unsigned char len = 0, char pad = ' ');
00057
00058         Mode mode() const
00059         { return int_mode; }
00060
00061         void mode(Mode m)
00062         { int_mode = m; }
00063
00064         int int_mode;
00065     };
00066
00071     class BasicOStream
00072     {
00073     public:
00074         BasicOStream(IOBackend *b)

```

```

00075     : iob(b)
00076     {}
00077
00078     void write(char const *str, unsigned len)
00079     {
00080         if (iob)
00081             iob->write(str, len);
00082     }
00083
00084     void write(long long int c, int len)
00085     {
00086         if (iob)
00087             iob->write(c, len);
00088     }
00089
00090     void write(long long unsigned c, unsigned char base = 10,
00091                unsigned char len = 0, char pad = ' ')
00092     {
00093         if (iob)
00094             iob->write(c, base, len, pad);
00095     }
00096
00097     void write(long long unsigned c, int len)
00098     {
00099         if (iob)
00100             iob->write(c, len);
00101     }
00102
00103     void write(IOModifier m)
00104     {
00105         if (iob)
00106             iob->write(m);
00107     }
00108
00109     IOBackend::Mode be_mode() const
00110     {
00111         if (iob)
00112             return iob->mode();
00113         return 0;
00114     }
00115
00116     void be_mode(IOBackend::Mode m)
00117     {
00118         if (iob)
00119             iob->mode(m);
00120     }
00121
00122 private:
00123     IOBackend *iob;
00124 };
00125
00126 class IONumFmt
00127 {
00128 public:
00129     IONumFmt(unsigned long long n, unsigned char base = 10,
00130              unsigned char len = 0, char pad = ' ')
00131         : n(n), base(base), len(len), pad(pad)
00132     {}
00133
00134     BasicOStream &print(BasicOStream &o) const;
00135
00136 private:
00137     unsigned long long n;
00138     unsigned char base, len;
00139     char pad;
00140 };
00141
00142 inline IONumFmt n_hex(unsigned long long n) { return IONumFmt(n, 16); }
00143
00144 extern IOModifier const hex;
00145
00146 extern IOModifier const dec;
00147
00148 inline
00149 BasicOStream &IONumFmt::print(BasicOStream &o) const
00150 {
00151     o.write(n, base, len, pad);
00152     return o;
00153 }
00154
00155 // Implementation
00156
00157 inline
00158 L4::BasicOStream &
00159 operator « (L4::BasicOStream &s, char const * const str)

```

```

00172 {
00173     if (!str)
00174     {
00175         s.write("(NULL)", 6);
00176         return s;
00177     }
00178
00179     unsigned l = 0;
00180     for (; str[l] != 0; l++)
00181     ;
00182     s.write(str, l);
00183     return s;
00184 }
00185
00186 inline
00187 L4::BasicOStream &
00188 operator « (L4::BasicOStream &s, signed short u)
00189 {
00190     s.write(static_cast<long long signed>(u), -1);
00191     return s;
00192 }
00193
00194 inline
00195 L4::BasicOStream &
00196 operator « (L4::BasicOStream &s, signed u)
00197 {
00198     s.write(static_cast<long long signed>(u), -1);
00199     return s;
00200 }
00201
00202 inline
00203 L4::BasicOStream &
00204 operator « (L4::BasicOStream &s, signed long u)
00205 {
00206     s.write(static_cast<long long signed>(u), -1);
00207     return s;
00208 }
00209
00210 inline
00211 L4::BasicOStream &
00212 operator « (L4::BasicOStream &s, signed long long u)
00213 {
00214     s.write(u, -1);
00215     return s;
00216 }
00217
00218 inline
00219 L4::BasicOStream &
00220 operator « (L4::BasicOStream &s, unsigned short u)
00221 {
00222     s.write(static_cast<long long unsigned>(u), -1);
00223     return s;
00224 }
00225
00226 inline
00227 L4::BasicOStream &
00228 operator « (L4::BasicOStream &s, unsigned u)
00229 {
00230     s.write(static_cast<long long unsigned>(u), -1);
00231     return s;
00232 }
00233
00234 inline
00235 L4::BasicOStream &
00236 operator « (L4::BasicOStream &s, unsigned long u)
00237 {
00238     s.write(static_cast<long long unsigned>(u), -1);
00239     return s;
00240 }
00241
00242 inline
00243 L4::BasicOStream &
00244 operator « (L4::BasicOStream &s, unsigned long long u)
00245 {
00246     s.write(u, -1);
00247     return s;
00248 }
00249
00250 inline
00251 L4::BasicOStream &
00252 operator « (L4::BasicOStream &s, void const *u)
00253 {
00254     long unsigned x = reinterpret_cast<long unsigned>(u);
00255     L4::IOBackend::Mode mode = s.be_mode();
00256     s.write(L4::hex);
00257     s.write(static_cast<long long unsigned>(x), -1);
00258     s.be_mode(mode);

```

```

00259     return s;
00260 }
00261
00262 inline
00263 L4::BasicOStream &
00264 operator « (L4::BasicOStream &s, L4::IOModifier m)
00265 {
00266     s.write(m);
00267     return s;
00268 }
00269
00270 inline
00271 L4::BasicOStream &
00272 operator « (L4::BasicOStream &s, char c)
00273 {
00274     s.write(&c, 1);
00275     return s;
00276 }
00277
00278 inline
00279 L4::BasicOStream &
00280 operator « (L4::BasicOStream &o, L4::IONumFmt const &n)
00281 { return n.print(o); }

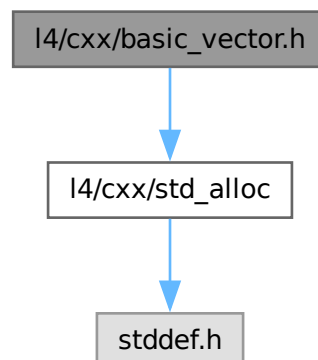
```

16.183 l4/cxx/basic_vector.h File Reference

Basic vector.

```
#include <l4/cxx/std_alloc>
```

Include dependency graph for basic_vector.h:



Namespaces

- namespace [cxx](#)
Our C++ library.

16.183.1 Detailed Description

Basic vector.

Definition in file [basic_vector.h](#).

16.184 basic_vector.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00007  *      economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011 #pragma once
00012
00013 #include <14/cxx/std_alloc>
00014
00015 namespace cxx {
00016
00017 template< typename T >
00018 class Basic_vector
00019 {
00020 public:
00021     Basic_vector(T *array, unsigned long capacity)
00022     : _array(array), _capacity(capacity)
00023     {
00024         for (unsigned long i = 0; i < capacity; ++i)
00025             new (&_array[i]) T();
00026     }
00027
00028 private:
00029     T *_array;
00030     unsigned long _capacity;
00031 };
00032
00033 };

```

16.185 bitfield

```

00001 // vi:set ft=c++: -- Mode: C++ --
00002 /*
00003  * (c) 2012 Alexander Warg <warg@os.inf.tu-dresden.de>,
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008
00009 #pragma once
00010
00011 #include "type_list"
00012
00013 namespace cxx {
00014
00015 template<typename T, unsigned LSB, unsigned MSB>
00016 class Bitfield
00017 {
00018 private:
00019     typedef remove_reference_t<T> Base_type;
00020
00021     static_assert(MSB >= LSB, "boundary mismatch in bit-field definition");
00022     static_assert(MSB < sizeof(Base_type) * 8, "MSB outside of bit-field type");
00023     static_assert(LSB < sizeof(Base_type) * 8, "LSB outside of bit-field type");
00024
00025     template<unsigned BITS> struct Best_type
00026     {
00027         template< typename TY > struct Cmp { enum { value = (BITS <= sizeof(TY)*8) }; };
00028         typedef cxx::type_list<
00029             unsigned char,
00030             unsigned short,
00031             unsigned int,
00032             unsigned long,
00033             unsigned long long
00034         > Unsigned_types;
00035         typedef cxx::find_type_t<Unsigned_types, Cmp> Type;
00036     };
00037
00038 public:
00039     enum
00040     {
00041         Bits = MSB + 1 - LSB,
00042         Lsb = LSB,
00043         Msb = MSB,
00044     };

```

```

00058
00060 static constexpr Base_type Low_mask
00061     = Base_type(~0ULL) » (sizeof(Base_type) * 8 - Bits);
00063 static constexpr Base_type Mask = Low_mask « Lsb;
00064
00071 typedef typename Best_type<Bits>::Type Bits_type;
00072
00079 typedef typename Best_type<Bits + Lsb>::Type Shift_type;
00080
00081 private:
00082     static_assert(sizeof(Bits_type)*8 >= Bits, "error finding the type to store the bits");
00083     static_assert(sizeof(Shift_type)*8 >= Bits + Lsb, "error finding the type to keep the shifted
bits");
00084     static_assert(sizeof(Bits_type) <= sizeof(Base_type), "size mismatch for Bits_type");
00085     static_assert(sizeof(Shift_type) <= sizeof(Base_type), "size mismatch for Shift_type");
00086     static_assert(sizeof(Bits_type) <= sizeof(Shift_type), "size mismatch for Shift_type and
Bits_type");
00087
00088 public:
00096 static constexpr Bits_type get(Shift_type val)
00097 { return (val » Lsb) & Low_mask; }
00098
00109 static constexpr Base_type get_unshifted(Shift_type val)
00110 { return val & Mask; }
00111
00124 static constexpr Base_type set_dirty(Base_type dest, Shift_type val)
00125 {
00126     //assert (! (val & ~Low_mask));
00127     return (dest & ~Mask) | (val « Lsb);
00128 }
00129
00144 static constexpr Base_type set_unshifted_dirty(Base_type dest, Shift_type val)
00145 {
00146     //assert (! (val & ~Mask));
00147     return (dest & ~Mask) | val;
00148 }
00149
00158 static Base_type set(Base_type dest, Bits_type val)
00159 { return set_dirty(dest, val & Low_mask); }
00160
00170 static Base_type set_unshifted(Base_type dest, Shift_type val)
00171 { return set_unshifted_dirty(dest, val & Mask); }
00172
00184 static constexpr Base_type val_dirty(Shift_type val) { return val « Lsb; }
00185
00193 static constexpr Base_type val(Bits_type val) { return val_dirty(val & Low_mask); }
00194
00203 static constexpr Base_type val_unshifted(Shift_type val) { return val & Mask; }
00204
00206 template< typename TT >
00207 class Value_base
00208 {
00209 private:
00210     TT v;
00211
00212 public:
00213     constexpr Value_base(TT t) : v(t) {}
00214     constexpr Bits_type get() const { return Bitfield::get(v); }
00215     constexpr Base_type get_unshifted() const { return Bitfield::get_unshifted(v); }
00216
00217     void set(Bits_type val) { v = Bitfield::set(v, val); }
00218     void set_dirty(Bits_type val) { v = Bitfield::set_dirty(v, val); }
00219     void set_unshifted(Shift_type val) { v = Bitfield::set_unshifted(v, val); }
00220     void set_unshifted_dirty(Shift_type val) { v = Bitfield::set_unshifted_dirty(v, val); }
00221 };
00222
00224 template< typename TT >
00225 class Value : public Value_base<TT>
00226 {
00227 public:
00228     constexpr Value(TT t) : Value_base<TT>(t) {}
00229     constexpr operator Bits_type () const { return this->get(); }
00230     constexpr Value &operator = (Bits_type val) { this->set(val); return *this; }
00231     constexpr Value &operator = (Value const &val)
00232     { this->set(val.get()); return *this; }
00233     Value(Value const &) = default;
00234 };
00235
00237 template< typename TT >
00238 class Value_unshifted : public Value_base<TT>
00239 {
00240 public:
00241     constexpr Value_unshifted(TT t) : Value_base<TT>(t) {}
00242     constexpr operator Shift_type () const { return this->get_unshifted(); }
00243     constexpr Value_unshifted &operator = (Shift_type val) { this->set_unshifted(val); return *this; }
00244     constexpr Value_unshifted &operator = (Value_unshifted const &val)
00245     { this->set_unshifted(val.get_unshifted()); return *this; }

```

```

00246     Value_unshifted(Value_unshifted const &) = default;
00247 };
00248
00250 typedef Value<Base_type &> Ref;
00252 typedef Value<Base_type volatile &> Ref_unshifted_volatile;
00254 typedef Value<Base_type const> Val;
00255
00257 typedef Value_unshifted<Base_type &> Ref_unshifted;
00259 typedef Value_unshifted<Base_type volatile &> Ref_unshifted_volatile;
00261 typedef Value_unshifted<Base_type const> Val_unshifted;
00262 };
00263
00264 #define CXX_BITFIELD_MEMBER(LSB, MSB, name, data_member) \
00265 \
00266 \
00267     typedef cxx::Bitfield<decltype(data_member), LSB, MSB> name ## _bfm_t; \
00268 \
00269     constexpr typename name ## _bfm_t::Val name() const { return data_member; } \
00270     typename name ## _bfm_t::Val name() const volatile { return data_member; } \
00271 \
00272     constexpr typename name ## _bfm_t::Ref name() { return data_member; } \
00273     typename name ## _bfm_t::Ref_unshifted_volatile name() volatile { return data_member; } \
00274
00275
00276 #define CXX_BITFIELD_MEMBER_RO(LSB, MSB, name, data_member) \
00277 \
00278 \
00279     typedef cxx::Bitfield<decltype(data_member), LSB, MSB> name ## _bfm_t; \
00280 \
00281     constexpr typename name ## _bfm_t::Val name() const { return data_member; } \
00282     typename name ## _bfm_t::Val name() const volatile { return data_member; } \
00283
00284
00285 #define CXX_BITFIELD_MEMBER_UNSHIFTED(LSB, MSB, name, data_member) \
00286 \
00287 \
00288     typedef cxx::Bitfield<decltype(data_member), LSB, MSB> name ## _bfm_t; \
00289 \
00290     constexpr typename name ## _bfm_t::Val_unshifted name() const { return data_member; } \
00291     typename name ## _bfm_t::Val_unshifted name() const volatile { return data_member; } \
00292 \
00293     constexpr typename name ## _bfm_t::Ref_unshifted name() { return data_member; } \
00294     typename name ## _bfm_t::Ref_unshifted_volatile name() volatile { return data_member; } \
00295
00296
00297 #define CXX_BITFIELD_MEMBER_UNSHIFTED_RO(LSB, MSB, name, data_member) \
00298 \
00299 \
00300     typedef cxx::Bitfield<decltype(data_member), LSB, MSB> name ## _bfm_t; \
00301 \
00302     constexpr typename name ## _bfm_t::Val_unshifted name() const { return data_member; } \
00303     typename name ## _bfm_t::Val_unshifted name() const volatile { return data_member; } \
00304
00305 }

```

16.186 bitmap

```

00001 // vi:set ft=c++: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2014 Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *     economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008
00009 #pragma once
00010
00011 namespace cxx {
00012
00018 class Bitmap_base
00019 {
00020 protected:
00024     typedef unsigned long word_type;
00025
00026     enum
00027     {
00028         W_bits = sizeof(word_type) * 8,
00029         C_bits = 8,
00030     };
00031
00035     word_type *_bits;
00036
00044     static unsigned word_index(unsigned bit) { return bit / W_bits; }

```

```

00045
00053 static unsigned bit_index(unsigned bit) { return bit % W_bits; }
00054
00058 class Bit
00059 {
00060     Bitmap_base *_bm;
00061     long _bit;
00062
00063 public:
00064     Bit(Bitmap_base *bm, long bit) : _bm(bm), _bit(bit) {}
00065     Bit &operator = (bool val) { _bm->bit(_bit, val); return *this; }
00066     operator bool () const { return _bm->bit(_bit); }
00067 };
00068
00069 public:
00070     explicit Bitmap_base(void *bits) noexcept : _bits(reinterpret_cast<word_type *>(bits)) {}
00071
00073 static long words(long bits) noexcept { return (bits + W_bits - 1) / W_bits; }
00074 static long bit_buffer_bytes(long bits) noexcept
00075 { return words(bits) * W_bits / 8; }
00076
00078 template< long BITS >
00079 class Word
00080 {
00081 public:
00082     typedef unsigned long Type;
00083     enum
00084     {
00085         Size = (BITS + W_bits - 1) / W_bits
00086     };
00087 };
00088
00090 static long chars(long bits) noexcept
00091 { return (bits + C_bits - 1) / C_bits; }
00092
00094 template< long BITS >
00095 class Char
00096 {
00097 public:
00098     typedef unsigned char Type;
00099     enum
00100     {
00101         Size = (BITS + C_bits - 1) / C_bits
00102     };
00103 };
00104
00111 void bit(long bit, bool on) noexcept;
00112
00118 void clear_bit(long bit) noexcept;
00119
00127 void atomic_clear_bit(long bit) noexcept;
00128
00136 word_type atomic_get_and_clear(long bit) noexcept;
00137
00143 void set_bit(long bit) noexcept;
00144
00152 void atomic_set_bit(long bit) noexcept;
00153
00161 word_type atomic_get_and_set(long bit) noexcept;
00162
00171 word_type bit(long bit) const noexcept;
00172
00181 word_type operator [] (long bit) const noexcept
00182 { return this->bit(bit); }
00183
00191 Bit operator [] (long bit) noexcept
00192 { return Bit(this, bit); }
00193
00205 long scan_zero(long max_bit, long start_bit = 0) const noexcept;
00206
00207 void *bit_buffer() const noexcept { return _bits; }
00208
00209 protected:
00210     static int _bzl(unsigned long w) noexcept;
00211 };
00212
00213
00219 template<int BITS>
00220 class Bitmap : public Bitmap_base
00221 {
00222 private:
00223     char _bits[Bitmap_base::Char<BITS>::Size];
00224
00225 public:
00227     Bitmap() noexcept : Bitmap_base(_bits) {}
00228     Bitmap(Bitmap<BITS> const &o) noexcept : Bitmap_base(_bits)
00229     { __builtin_memcpy(_bits, o._bits, sizeof(_bits)); }

```



```

00242     long scan_zero(long start_bit = 0) const noexcept;
00243
00244     void clear_all()
00245     { __builtin_memset(_bits, 0, sizeof(_bits)); }
00246 };
00247
00248
00249 inline
00250 void
00251 Bitmap_base::bit(long bit, bool on) noexcept
00252 {
00253     long idx = word_index(bit);
00254     long b   = bit_index(bit);
00255     _bits[idx] = (_bits[idx] & ~(1UL << b)) | (static_cast<unsigned long>(on) << b);
00256 }
00257
00258 inline
00259 void
00260 Bitmap_base::clear_bit(long bit) noexcept
00261 {
00262     long idx = word_index(bit);
00263     long b   = bit_index(bit);
00264     _bits[idx] &= ~(1UL << b);
00265 }
00266
00267 inline
00268 void
00269 Bitmap_base::atomic_clear_bit(long bit) noexcept
00270 {
00271     long idx = word_index(bit);
00272     long b   = bit_index(bit);
00273     word_type mask = 1UL << b;
00274     __atomic_and_fetch(&_bits[idx], ~mask, __ATOMIC_RELAXED);
00275 }
00276
00277 inline
00278 Bitmap_base::word_type
00279 Bitmap_base::atomic_get_and_clear(long bit) noexcept
00280 {
00281     long idx = word_index(bit);
00282     long b   = bit_index(bit);
00283     word_type mask = 1UL << b;
00284     return __atomic_fetch_and(&_bits[idx], ~mask, __ATOMIC_RELAXED) & mask;
00285 }
00286
00287 inline
00288 void
00289 Bitmap_base::set_bit(long bit) noexcept
00290 {
00291     long idx = word_index(bit);
00292     long b   = bit_index(bit);
00293     _bits[idx] |= (1UL << b);
00294 }
00295
00296 inline
00297 void
00298 Bitmap_base::atomic_set_bit(long bit) noexcept
00299 {
00300     long idx = word_index(bit);
00301     long b   = bit_index(bit);
00302     word_type mask = 1UL << b;
00303     __atomic_or_fetch(&_bits[idx], mask, __ATOMIC_RELAXED);
00304 }
00305
00306 inline
00307 Bitmap_base::word_type
00308 Bitmap_base::atomic_get_and_set(long bit) noexcept
00309 {
00310     long idx = word_index(bit);
00311     long b   = bit_index(bit);
00312     word_type mask = 1UL << b;
00313     return __atomic_fetch_or(&_bits[idx], mask, __ATOMIC_RELAXED) & mask;
00314 }
00315
00316 inline
00317 Bitmap_base::word_type
00318 Bitmap_base::bit(long bit) const noexcept
00319 {
00320     long idx = word_index(bit);
00321     long b   = bit_index(bit);
00322     return _bits[idx] & (1UL << b);
00323 }
00324
00325 inline
00326 int
00327 Bitmap_base::bzl(unsigned long w) noexcept
00328 {

```

```

00329     for (int i = 0; i < W_bits; ++i, w >>= 1)
00330     {
00331         if ((w & 1) == 0)
00332             return i;
00333     }
00334     return -1;
00335 }
00336
00337 inline
00338 long
00339 Bitmap_base::scan_zero(long max_bit, long start_bit) const noexcept
00340 {
00341     if (!(operator [] (start_bit)))
00342         return start_bit;
00343
00344     long idx = word_index(start_bit);
00345
00346     max_bit -= start_bit & ~(W_bits - 1);
00347
00348     for (; max_bit > 0; max_bit -= W_bits, ++idx)
00349     {
00350         if (_bits[idx] == 0)
00351             return idx * W_bits;
00352
00353         if (_bits[idx] != ~0UL)
00354         {
00355             long zbit = _bzl(_bits[idx]);
00356             return zbit < max_bit ? idx * W_bits + zbit : -1;
00357         }
00358     }
00359
00360     return -1;
00361 }
00362
00363 template<int BITS> inline
00364 long
00365 Bitmap<BITS>::scan_zero(long start_bit) const noexcept
00366 {
00367     return Bitmap_base::scan_zero(BITS, start_bit);
00368 }
00369
00370 };

```

16.187 I4/cxx/bits/bst.h File Reference

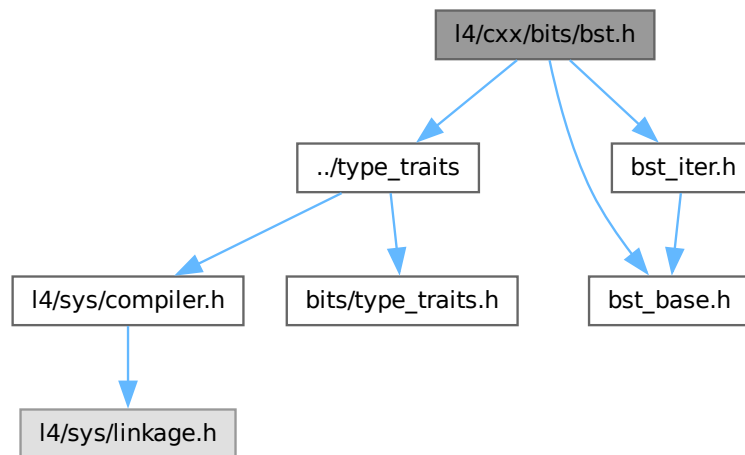
AVL tree.

```

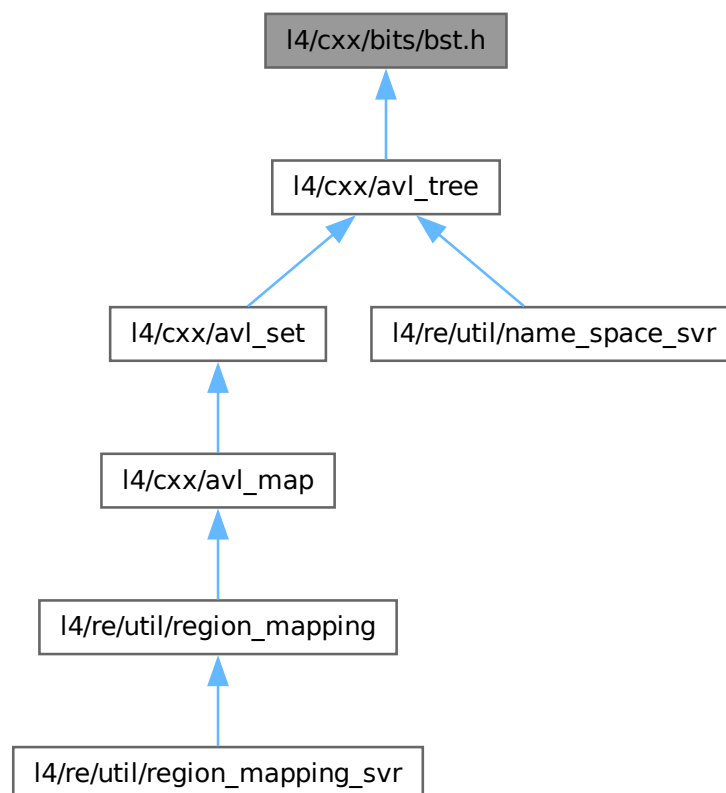
#include "../type_traits"
#include "bst_base.h"
#include "bst_iter.h"

```

Include dependency graph for bst.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- class [cxx::Bits::Bst< Node, Get_key, Compare >](#)
Basic binary search tree (BST).

Namespaces

- namespace [cxx](#)
Our C++ library.
- namespace [cxx::Bits](#)
Internal helpers for the cxx package.

16.187.1 Detailed Description

AVL tree.

Definition in file [bst.h](#).

16.188 bst.h

[Go to the documentation of this file.](#)

```

00001 // vi:ft=cpp
00002 /*
00003  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00004  *      Carsten Weinhold <weinhold@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009 #pragma once
00010
00011 #include "../type_traits"
00012 #include "bst_base.h"
00013 #include "bst_iter.h"
00014
00015 struct Avl_set_tester;
00016
00017 namespace cxx { namespace Bits {
00018
00019 template< typename Node, typename Get_key, typename Compare >
00020 class Bst
00021 {
00022     friend struct ::Avl_set_tester;
00023 private:
00024     typedef Direction Dir;
00025
00026     struct Fwd
00027     {
00028         static Node *child(Node const *n, Direction d)
00029         { return Bst_node::next<Node>(n, d); }
00030
00031         static bool cmp(Node const *l, Node const *r)
00032         { return Compare()(Get_key::key_of(l), Get_key::key_of(r)); }
00033     };
00034
00035     struct Rev
00036     {
00037         static Node *child(Node const *n, Direction d)
00038         { return Bst_node::next<Node>(n, !d); }
00039
00040         static bool cmp(Node const *l, Node const *r)
00041         { return Compare()(Get_key::key_of(r), Get_key::key_of(l)); }
00042     };
00043
00044 public:
00045     typedef typename Get_key::Key_type Key_type;

```

```

00062     typedef typename Type_traits<Key_type>::Param_type Key_param_type;
00063
00065     typedef Fwd Fwd_iter_ops;
00067     typedef Rev Rev_iter_ops;
00068
00070
00072     typedef __Bst_iter<Node, Node, Fwd> Iterator;
00074     typedef __Bst_iter<Node, Node const, Fwd> Const_iterator;
00076     typedef __Bst_iter<Node, Node, Rev> Rev_iterator;
00078     typedef __Bst_iter<Node, Node const, Rev> Const_rev_iterator;
00080
00081 protected:
00090
00092     Bst_node *_head;
00093
00095     Bst() : _head(nullptr) {}
00096
00098     Node *head() const { return static_cast<Node*>(_head); }
00099
00101     static Key_type k(Bst_node const *n)
00102     { return Get_key::key_of(static_cast<Node const *>(n)); }
00103
00112     static Dir dir(Key_param_type l, Key_param_type r)
00113     {
00114         Compare cmp;
00115         Dir d(cmp(r, l));
00116         if (d == Direction::L && !cmp(l, r))
00117             return Direction::N;
00118         return d;
00119     }
00120
00129     static Dir dir(Key_param_type l, Bst_node const *r)
00130     { return dir(l, k(r)); }
00131
00133     static bool greater(Key_param_type l, Key_param_type r)
00134     { return Compare()(r, l); }
00135
00137     static bool greater(Key_param_type l, Bst_node const *r)
00138     { return greater(l, k(r)); }
00139
00141     static bool greater(Bst_node const *l, Bst_node const *r)
00142     { return greater(k(l), k(r)); }
00144
00151     template<typename FUNC>
00152     static void remove_tree(Bst_node *head, FUNC &&callback)
00153     {
00154         if (Bst_node *n = Bst_node::next(head, Dir::L))
00155             remove_tree(n, callback);
00156
00157         if (Bst_node *n = Bst_node::next(head, Dir::R))
00158             remove_tree(n, callback);
00159
00160         callback(static_cast<Node *>(head));
00161     }
00162
00163 public:
00164
00173     Const_iterator begin() const { return Const_iterator(head()); }
00178     Const_iterator end() const { return Const_iterator(); }
00179
00184     Iterator begin() { return Iterator(head()); }
00189     Iterator end() { return Iterator(); }
00190
00195     Const_rev_iterator rbegin() const { return Const_rev_iterator(head()); }
00200     Const_rev_iterator rend() const { return Const_rev_iterator(); }
00201
00206     Rev_iterator rbegin() { return Rev_iterator(head()); }
00211     Rev_iterator rend() { return Rev_iterator(); }
00213
00214
00219
00225     Node *find_node(Key_param_type key) const;
00226
00233     Node *lower_bound_node(Key_param_type key) const;
00234
00241     Const_iterator find(Key_param_type key) const;
00242
00251     template<typename FUNC>
00252     void remove_all(FUNC &&callback)
00253     {
00254         if (!_head)
00255             return;
00256
00257         Bst_node *head = _head;
00258         _head = nullptr;
00259         remove_tree(head, cxx::forward<FUNC>(callback));
00260     }

```

```

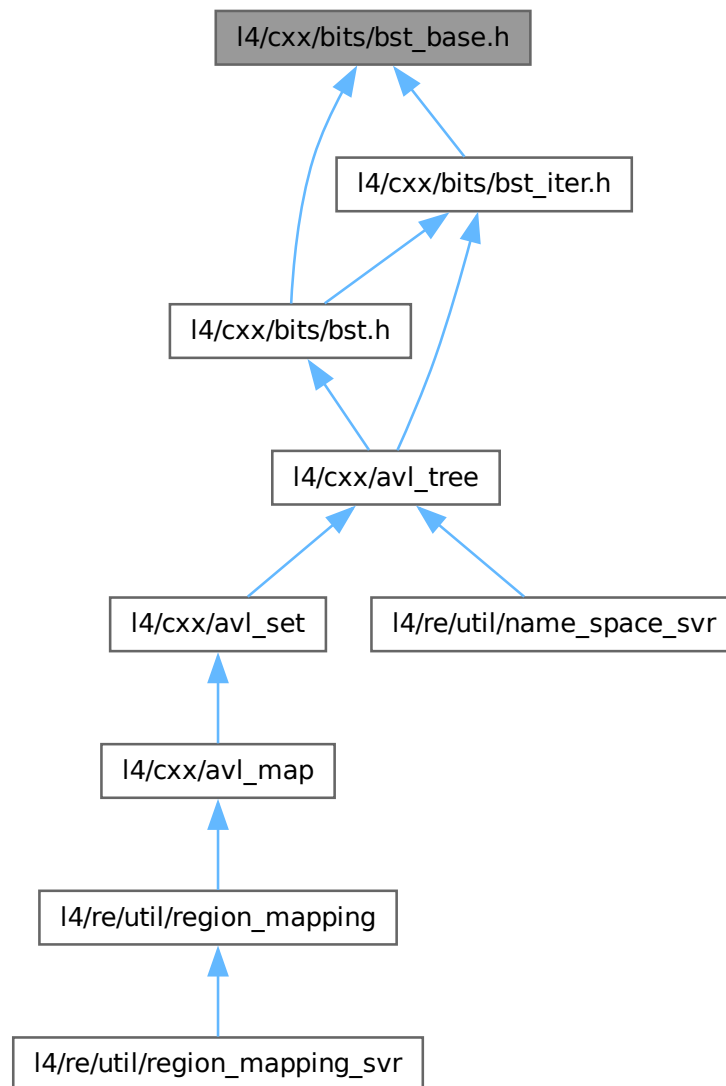
00261
00262
00264 };
00265
00266 /* find an element */
00267 template< typename Node, typename Get_key, class Compare>
00268 inline
00269 Node *
00270 Bst<Node, Get_key, Compare>::find_node(Key_param_type key) const
00271 {
00272     Dir d;
00273
00274     for (Bst_node *q = _head; q; q = Bst_node::next(q, d))
00275     {
00276         d = dir(key, q);
00277         if (d == Dir::N)
00278             return static_cast<Node*>(q);
00279     }
00280     return nullptr;
00281 }
00282
00283 template< typename Node, typename Get_key, class Compare>
00284 inline
00285 Node *
00286 Bst<Node, Get_key, Compare>::lower_bound_node(Key_param_type key) const
00287 {
00288     Dir d;
00289     Bst_node *r = nullptr;
00290
00291     for (Bst_node *q = _head; q; q = Bst_node::next(q, d))
00292     {
00293         d = dir(key, q);
00294         if (d == Dir::L)
00295             r = q; // found a node greater than key
00296         else if (d == Dir::N)
00297             return static_cast<Node*>(q);
00298     }
00299     return static_cast<Node*>(r);
00300 }
00301
00302 /* find an element */
00303 template< typename Node, typename Get_key, class Compare>
00304 inline
00305 typename Bst<Node, Get_key, Compare>::Const_iterator
00306 Bst<Node, Get_key, Compare>::find(Key_param_type key) const
00307 {
00308     Bst_node *q = _head;
00309     Bst_node *r = q;
00310
00311     for (Dir d; q; q = Bst_node::next(q, d))
00312     {
00313         d = dir(key, q);
00314         if (d == Dir::N)
00315             return Iterator(static_cast<Node*>(q), static_cast<Node *>(r));
00316
00317         if (d != Dir::L && q == r)
00318             r = Bst_node::next(q, d);
00319     }
00320     return Iterator();
00321 }
00322
00323 }}

```

16.189 I4/cxx/bits/bst_base.h File Reference

AVL tree.

This graph shows which files directly or indirectly include this file:



Data Structures

- struct [cxx::Bits::Direction](#)
The direction to go in a binary search tree.
- class [cxx::Bits::Bst_node](#)
Basic type of a node in a binary search tree (BST).

Namespaces

- namespace [cxx](#)
Our C++ library.
- namespace [cxx::Bits](#)
Internal helpers for the cxx package.

16.189.1 Detailed Description

AVL tree.

Definition in file [bst_base.h](#).

16.190 bst_base.h

[Go to the documentation of this file.](#)

```

00001 // vi:ft=cpp
00006 /*
00007  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00008  *          Carsten Weinhold <weinhold@os.inf.tu-dresden.de>
00009  *          economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013
00014 #pragma once
00015
00016 /*
00017  * This file contains very basic bits for implementing binary search trees
00018  */
00019 namespace cxx {
00023 namespace Bits {
00024
00028 struct Direction
00029 {
00031     enum Direction_e
00032     {
00033         L = 0,
00034         R = 1,
00035         N = 2
00036     };
00037     unsigned char d;
00038
00040     Direction() = default;
00041
00043     Direction(Direction_e d) : d(d) {}
00044
00046     explicit Direction(bool b) : d(Direction_e(b)) /*d(b ? R : L)*/ {}
00047
00052     Direction operator ! () const { return Direction(!d); }
00053
00055
00057     bool operator == (Direction_e o) const { return d == o; }
00059     bool operator != (Direction_e o) const { return d != o; }
00061     bool operator == (Direction o) const { return d == o.d; }
00063     bool operator != (Direction o) const { return d != o.d; }
00065 };
00066
00070 class Bst_node
00071 {
00072     // all BSTs are friends
00073     template< typename Node, typename Get_key, typename Compare >
00074     friend class Bst;
00075
00076 protected:
00085
00087     static Bst_node *next(Bst_node const *p, Direction d)
00088     { return p->_c[d.d]; }
00089
00091     static void next(Bst_node *p, Direction d, Bst_node *n)
00092     { p->_c[d.d] = n; }
00093
00095     static Bst_node **next_p(Bst_node *p, Direction d)
00096     { return &p->_c[d.d]; }
00097
00099     template< typename Node > static
00100     Node *next(Bst_node const *p, Direction d)
00101     { return static_cast<Node *>(p->_c[d.d]); }
00102
00104     static void rotate(Bst_node **t, Direction idir);
00106
00107 private:
00108     Bst_node *_c[2];
00109
00110 protected:

```



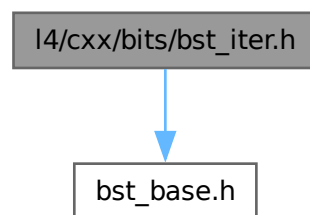
```
00112  Bst_node() {}
00113
00115  explicit Bst_node(bool) { _c[0] = _c[1] = 0; }
00116 };
00117
00118 inline
00119 void
00120 Bst_node::rotate(Bst_node **t, Direction idir)
00121 {
00122     Bst_node *tmp = *t;
00123     *t = next(tmp, idir);
00124     next(tmp, idir, next(*t, !idir));
00125     next(*t, !idir, tmp);
00126 }
00127
00128 }
```

16.191 I4/cxx/bits/bst_iter.h File Reference

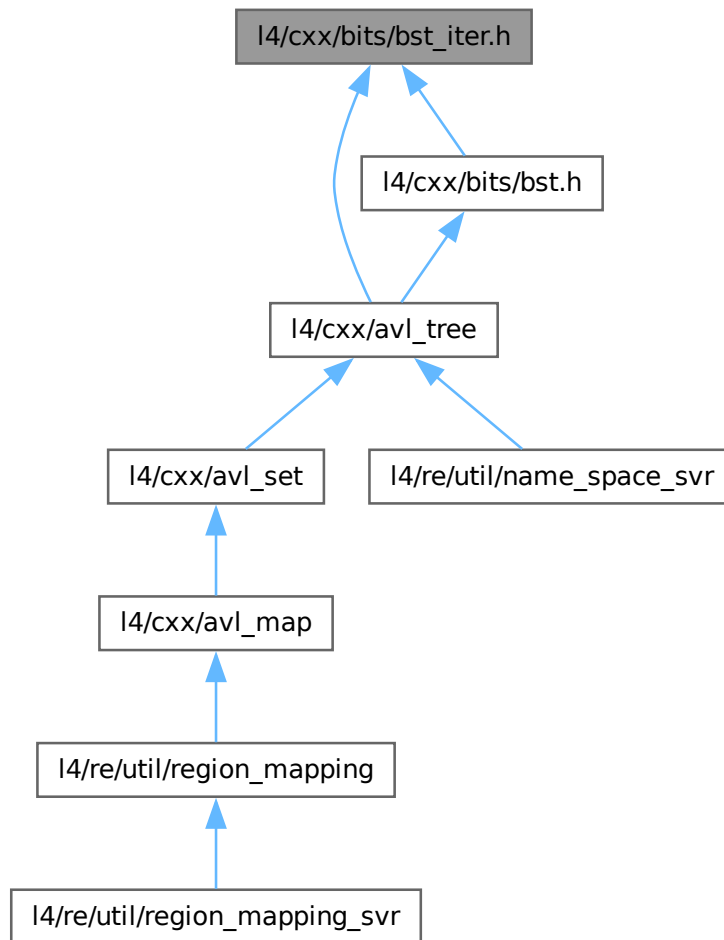
AVL tree.

```
#include "bst_base.h"
```

Include dependency graph for bst_iter.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace `cxx`
Our C++ library.
- namespace `cxx::Bits`
Internal helpers for the cxx package.

16.191.1 Detailed Description

AVL tree.

Definition in file `bst_iter.h`.

16.192 bst_iter.h

[Go to the documentation of this file.](#)

```

00001 // vi:ft=cpp
00002 /*
00003  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00004  *      Carsten Weinhold <weinhold@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009
00010 #pragma once
00011
00012 #include "bst_base.h"
00013
00014 namespace cxx { namespace Bits {
00015
00016     template< typename Node, typename Node_op >
00017     class __Bst_iter_b
00018     {
00019     protected:
00020         typedef Direction Dir;
00021         Node const *_n;
00022         Node const *_r;
00023
00024         __Bst_iter_b() : _n(0), _r(0) {}
00025
00026         __Bst_iter_b(Node const *t)
00027             : _n(t), _r(_n)
00028             { _downmost(); }
00029
00030         __Bst_iter_b(Node const *t, Node const *r)
00031             : _n(t), _r(r)
00032             {}
00033
00034         inline void _downmost();
00035
00036         inline void inc();
00037
00038     public:
00039         bool operator == (__Bst_iter_b const &o) const { return _n == o._n; }
00040         bool operator != (__Bst_iter_b const &o) const { return _n != o._n; }
00041     };
00042
00043     template< typename Node, typename Node_type, typename Node_op >
00044     class __Bst_iter : public __Bst_iter_b<Node, Node_op>
00045     {
00046     private:
00047         typedef __Bst_iter_b<Node, Node_op> Base;
00048         using Base::_n;
00049         using Base::_r;
00050         using Base::inc;
00051
00052     public:
00053         __Bst_iter() {}
00054
00055         __Bst_iter(Node const *t) : Base(t) {}
00056         __Bst_iter(Node const *t, Node const *r) : Base(t, r) {}
00057
00058         // template<typename Key2>
00059         __Bst_iter(Base const &o) : Base(o) {}
00060
00061         Node_type &operator * () const { return *const_cast<Node *>(_n); }
00062         Node_type *operator -> () const { return const_cast<Node *>(_n); }
00063         __Bst_iter &operator ++ () { inc(); return *this; }
00064         __Bst_iter &operator ++ (int)
00065         { __Bst_iter tmp = *this; inc(); return tmp; }
00066     };
00067
00068 //-----
00069 /* IMPLEMENTATION: __Bst_iter_b */
00070
00071     template< typename Node, typename Node_op>
00072     inline
00073     void __Bst_iter_b<Node, Node_op>::_downmost()
00074     {
00075         while (_n)
00076         {
00077             Node *n = Node_op::child(_n, Dir::L);
00078             if (n)
00079                 _n = n;
00080             else
00081

```

```

00134     return;
00135     }
00136 }
00137
00138 template< typename Node, typename Node_op>
00139 void __Bst_iter_b<Node, Node_op>::inc()
00140 {
00141     if (!_n)
00142         return;
00143
00144     if (_n == _r)
00145     {
00146         _r = _n = Node_op::child(_r, Dir::R);
00147         _downmost();
00148         return;
00149     }
00150
00151     if (Node_op::child(_n, Dir::R))
00152     {
00153         _n = Node_op::child(_n, Dir::R);
00154         _downmost();
00155         return;
00156     }
00157
00158     Node const *q = _r;
00159     Node const *p = _r;
00160     while (1)
00161     {
00162         if (Node_op::cmp(_n, q))
00163         {
00164             p = q;
00165             q = Node_op::child(q, Dir::L);
00166         }
00167         else if (_n == q || Node_op::child(q, Dir::R) == _n)
00168         {
00169             _n = p;
00170             return;
00171         }
00172         else
00173             q = Node_op::child(q, Dir::R);
00174     }
00175 }
00176
00177 }}

```

16.193 list_basics.h

```

00001 /*
00002  * (c) 2011 Alexander Warg <warg@os.inf.tu-dresden.de>
00003  *     economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00008 #pragma once
00009
00010 namespace cxx { namespace Bits {
00011
00012     template< typename T >
00013     class List_iterator_end_ptr
00014     {
00015     private:
00016         template< typename U > friend class Basic_list;
00017         static void * const _end;
00018     };
00019
00020     template< typename T >
00021     void * const List_iterator_end_ptr<T>::_end = nullptr;
00022
00023     template< typename VALUE_T, typename TYPE >
00024     struct Basic_list_policy
00025     {
00026         typedef VALUE_T *Value_type;
00027         typedef VALUE_T const *Const_value_type;
00028         typedef TYPE **Type;
00029         typedef TYPE *Const_type;
00030         typedef TYPE *Head_type;
00031         typedef TYPE Item_type;
00032
00033         static Type next(Type c) { return &(*c)->_n; }
00034         static Const_type next(Const_type c) { return c->_n; }
00035     };
00036

```

```

00038 template< typename POLICY >
00039 class Basic_list
00040 {
00041     Basic_list(Basic_list const &) = delete;
00042     void operator = (Basic_list const &) = delete;
00043
00044 public:
00045     typedef typename POLICY::Value_type Value_type;
00046     typedef typename POLICY::Const_value_type Const_value_type;
00047
00048     class Iterator
00049     {
00050     private:
00051         typedef typename POLICY::Type Internal_type;
00052
00053     public:
00054         typedef typename POLICY::Value_type value_type;
00055         typedef typename POLICY::Value_type Value_type;
00056
00057         Value_type operator * () const { return static_cast<Value_type>(*_c); }
00058         Value_type operator -> () const { return static_cast<Value_type>(*_c); }
00059         Iterator operator ++ () { _c = POLICY::next(_c); return *this; }
00060
00061         bool operator == (Iterator const &o) const { return *_c == *o._c; }
00062         bool operator != (Iterator const &o) const { return !operator == (o); }
00063
00064         Iterator() : _c(__end()) {}
00065
00066     private:
00067         friend class Basic_list;
00068         static Internal_type __end()
00069         {
00070             union X { Internal_type l; void * const *v; } z;
00071             z.v = &Bits::List_iterator_end_ptr<void>::_end;
00072             return z.l;
00073         }
00074
00075         explicit Iterator(Internal_type i) : _c(i) {}
00076
00077         Internal_type _c;
00078     };
00079
00080     class Const_iterator
00081     {
00082     private:
00083         typedef typename POLICY::Const_type Internal_type;
00084
00085     public:
00086         typedef typename POLICY::Value_type value_type;
00087         typedef typename POLICY::Value_type Value_type;
00088
00089         Value_type operator * () const { return static_cast<Value_type>(_c); }
00090         Value_type operator -> () const { return static_cast<Value_type>(_c); }
00091         Const_iterator operator ++ () { _c = POLICY::next(_c); return *this; }
00092
00093         friend bool operator == (Const_iterator const &lhs, Const_iterator const &rhs)
00094         { return lhs._c == rhs._c; }
00095         friend bool operator != (Const_iterator const &lhs, Const_iterator const &rhs)
00096         { return lhs._c != rhs._c; }
00097
00098         Const_iterator() {}
00099         Const_iterator(Iterator const &o) : _c(*o) {}
00100
00101     private:
00102         friend class Basic_list;
00103
00104         explicit Const_iterator(Internal_type i) : _c(i) {}
00105
00106         Internal_type _c;
00107     };
00108
00109     // BSS allocation
00110     explicit Basic_list(bool) {}
00111     Basic_list() : _f(nullptr) {}
00112
00113     Basic_list(Basic_list &&o) : _f(o._f)
00114     {
00115         o.clear();
00116     }
00117
00118     Basic_list &operator = (Basic_list &&o)
00119     {
00120         if (&o != this)
00121         {
00122             _f = o._f;
00123             o.clear();
00124         }
00125     }

```

```

00125
00126     return *this;
00127 }
00128
00130 bool empty() const { return !_f; }
00132 Value_type front() const { return static_cast<Value_type>(_f); }
00133
00139 void clear() { _f = nullptr; }
00140
00142 Iterator begin() { return Iterator(&_f); }
00144 Const_iterator begin() const { return Const_iterator(_f); }
00152 static Const_iterator iter(Const_value_type c) { return Const_iterator(c); }
00154 Const_iterator end() const { return Const_iterator(nullptr); }
00156 Iterator end() { return Iterator(); }
00157
00158 protected:
00159     static typename POLICY::Type __get_internal(Iterator const &i) { return i._c; }
00160     static Iterator __iter(typename POLICY::Type c) { return Iterator(c); }
00161
00163     typename POLICY::Head_type _f;
00164 };
00165
00166 }}
00167

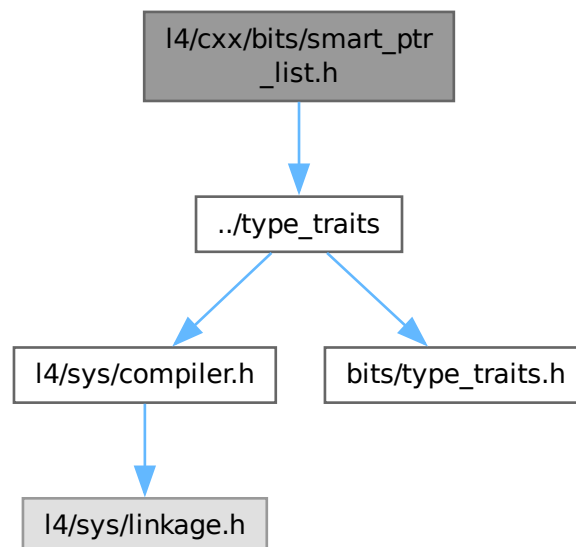
```

16.194 I4/cxx/bits/smart_ptr_list.h File Reference

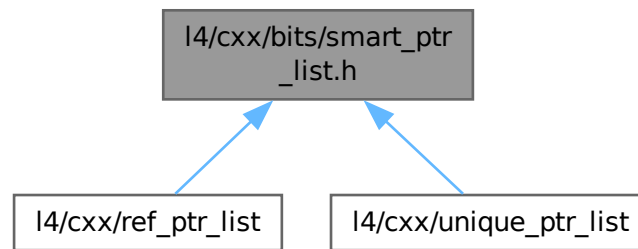
Implementation of a list of smart-pointer-managed objects.

```
#include "../type_traits"
```

Include dependency graph for smart_ptr_list.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- class `cxx::Bits::Smart_ptr_list_item< T, STORE_T >`
List item for an arbitrary item in a `Smart_ptr_list`.
- class `cxx::Bits::Smart_ptr_list< ITEM >`
List of smart-pointer-managed objects.

Namespaces

- namespace `cxx`
Our C++ library.
- namespace `cxx::Bits`
Internal helpers for the `cxx` package.

16.194.1 Detailed Description

Implementation of a list of smart-pointer-managed objects.

Definition in file [smart_ptr_list.h](#).

16.195 smart_ptr_list.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * Copyright (C) 2018, 2022, 2024 Kernkonzept GmbH.
00007  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011 #pragma once
00012
00013 #include "../type_traits"
00014
00015 namespace cxx { namespace Bits {
00016

```

```

00026 template <typename T, typename STORE_T>
00027 class Smart_ptr_list_item
00028 {
00029     using Value_type = T;
00030     using Storage_type = STORE_T;
00031
00032     template<typename U> friend class Smart_ptr_list;
00033     Storage_type _n;
00034 };
00035
00045 template <typename ITEM>
00046 class Smart_ptr_list
00047 {
00048     using Value_type = typename ITEM::Value_type;
00049     using Next_type = typename ITEM::Storage_type;
00050
00051 public:
00052     class Iterator
00053     {
00054     public:
00055         Iterator() : _c(nullptr) {}
00056
00057         Value_type *operator * () const { return _c; }
00058         Value_type *operator -> () const { return _c; }
00059
00060         Iterator operator ++ ()
00061         {
00062             _c = _c->n.get();
00063             return *this;
00064         }
00065
00066         bool operator == (Iterator const &o) const { return _c == o._c; }
00067         bool operator != (Iterator const &o) const { return !operator == (o); }
00068
00069     private:
00070         friend class Smart_ptr_list;
00071
00072         explicit Iterator(Value_type *i) : _c(i) {}
00073
00074         Value_type *_c;
00075     };
00076
00077     class Const_iterator
00078     {
00079     public:
00080         Const_iterator() : _c(nullptr) {}
00081
00082         Value_type const *operator * () const { return _c; }
00083         Value_type const *operator -> () const { return _c; }
00084
00085         Const_iterator operator ++ ()
00086         {
00087             _c = _c->n.get();
00088             return *this;
00089         }
00090
00091         bool operator == (Const_iterator const &o) const { return _c == o._c; }
00092         bool operator != (Const_iterator const &o) const { return !operator == (o); }
00093
00094     private:
00095         friend class Smart_ptr_list;
00096
00097         explicit Const_iterator(Value_type const *i) : _c(i) {}
00098
00099         Value_type const *_c;
00100     };
00101
00102     Smart_ptr_list() : _b(&_f) {}
00103
00105     void push_front(Next_type &&e)
00106     {
00107         e->n = cxx::move(this->_f);
00108         this->_f = cxx::move(e);
00109
00110         if (_b == &_f)
00111             _b = &(_f->n);
00112     }
00113
00115     void push_front(Next_type const &e)
00116     {
00117         e->n = cxx::move(this->_f);
00118         this->_f = e;
00119
00120         if (_b == &_f)
00121             _b = &(_f->n);
00122     }
00123

```



```

00125 void push_back(Next_type &&e)
00126 {
00127     *_b = cxx::move(e);
00128     _b = &((*_b)->_n);
00129 }
00130
00132 void push_back(Next_type const &e)
00133 {
00134     *_b = e;
00135     _b = &((*_b)->_n);
00136 }
00137
00139 Value_type *front() const
00140 { return _f.get(); }
00141
00149 Next_type pop_front()
00150 {
00151     Next_type ret = cxx::move(_f);
00152
00153     if (ret)
00154         _f = cxx::move(ret->_n);
00155
00156     if (!_f)
00157         _b = &_f;
00158
00159     return ret;
00160 }
00161
00163 bool empty() const
00164 { return !_f; }
00165
00166 Iterator begin() { return Iterator(_f.get()); }
00167 Iterator end() { return Iterator(); }
00168
00169 Const_iterator begin() const { return Const_iterator(_f.get()); }
00170 Const_iterator end() const { return Const_iterator(); }
00171
00172 Const_iterator cbegin() const { return const_iterator(_f.get()); }
00173 Const_iterator cend() const { return Const_iterator(); }
00174
00175 private:
00176     Next_type _f;
00177     Next_type *_b;
00178 };
00179
00180 } }

```

16.196 type_traits.h

```

00001 // vi:ft=cpp
00002 /*
00003  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00004  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009
00010 #pragma once
00011
00012 namespace cxx {
00013
00014     class Null_type;
00015
00016     template<bool flag, typename T, typename F>
00017     struct Select { using Type = T; };
00018
00019     template<typename T, typename F>
00020     struct Select<false, T, F> { using Type = F; };
00021
00022     template<typename T, typename U>
00023     class Conversion
00024     {
00025     public:
00026         using S = char;
00027
00028         class B { char dummy[2]; };
00029
00030         static S test(U);
00031         static B test(...);
00032         static T make_T();
00033     };
00034
00035     static constexpr bool exists = sizeof(test(make_T())) == sizeof(S);

```

```

00035     static constexpr bool two_way = exists && Conversion<U, T>::exists;
00036     static constexpr bool exists_2_way = two_way;
00037     static constexpr bool same_type = false;
00038 };
00039
00040 template<>
00041 class Conversion<void, void>
00042 {
00043 public:
00044     static constexpr bool exists = true;
00045     static constexpr bool two_way = true;
00046     static constexpr bool exists_2_way = two_way;
00047     static constexpr bool same_type = true;
00048 };
00049
00050 template<typename T>
00051 class Conversion<T, T>
00052 {
00053 public:
00054     static constexpr bool exists = true;
00055     static constexpr bool two_way = true;
00056     static constexpr bool exists_2_way = two_way;
00057     static constexpr bool same_type = true;
00058 };
00059
00060 template<typename T>
00061 class Conversion<void, T>
00062 {
00063 public:
00064     static constexpr bool exists = false;
00065     static constexpr bool two_way = false;
00066     static constexpr bool exists_2_way = two_way;
00067     static constexpr bool same_type = false;
00068 };
00069
00070 template<typename T>
00071 class Conversion<T, void>
00072 {
00073 public:
00074     static constexpr bool exists = false;
00075     static constexpr bool two_way = false;
00076     static constexpr bool exists_2_way = two_way;
00077     static constexpr bool same_type = false;
00078 };
00079
00080 namespace TT
00081 {
00082     template<typename U>
00083     struct Pointer_traits
00084     {
00085         using Pointee = Null_type;
00086         static constexpr bool value = false;
00087     };
00088
00089     template<typename U>
00090     struct Pointer_traits<U *>
00091     {
00092         using Pointee = U;
00093         static constexpr bool value = true;
00094     };
00095
00096     template<typename U>
00097     struct Ref_traits
00098     {
00099         using Referee = U;
00100         static constexpr bool value = false;
00101     };
00102
00103     template<typename U>
00104     struct Ref_traits<U &>
00105     {
00106         using Referee = U;
00107         static constexpr bool value = true;
00108     };
00109
00110     template<typename U>
00111     struct Add_ref { using Type = U &; };
00112
00113     template<typename U>
00114     struct Add_ref<U &> { using Type = U; };
00115
00116     template<typename U>
00117     struct PMF_traits
00118     { static constexpr bool value = false; };
00119
00120     template<typename U, typename F>
00121     struct PMF_traits<U F:: *>

```

```

00122     { static constexpr bool value = true; };
00123
00124     template<typename U>
00125     struct Is_unsigned
00126     { static constexpr bool value = false; };
00127
00128     template<>
00129     struct Is_unsigned<unsigned>
00130     { static constexpr bool value = true; };
00131
00132     template<>
00133     struct Is_unsigned<unsigned char>
00134     { static constexpr bool value = true; };
00135
00136     template<>
00137     struct Is_unsigned<unsigned short>
00138     { static constexpr bool value = true; };
00139
00140     template<>
00141     struct Is_unsigned<unsigned long>
00142     { static constexpr bool value = true; };
00143
00144     template<>
00145     struct Is_unsigned<unsigned long long>
00146     { static constexpr bool value = true; };
00147
00148     template<typename U>
00149     struct Is_signed
00150     { static constexpr bool value = false; };
00151
00152     template<>
00153     struct Is_signed<signed>
00154     { static constexpr bool value = true; };
00155
00156     template<>
00157     struct Is_signed<signed char>
00158     { static constexpr bool value = true; };
00159
00160     template<>
00161     struct Is_signed<signed short>
00162     { static constexpr bool value = true; };
00163
00164     template<>
00165     struct Is_signed<signed long>
00166     { static constexpr bool value = true; };
00167
00168     template<>
00169     struct Is_signed<signed long long>
00170     { static constexpr bool value = true; };
00171
00172     template<typename U>
00173     struct Is_int
00174     { static constexpr bool value = false; };
00175
00176     template<>
00177     struct Is_int<char>
00178     { static constexpr bool value = true; };
00179
00180     template<>
00181     struct Is_int<bool>
00182     { static constexpr bool value = true; };
00183
00184     template<>
00185     struct Is_int<wchar_t>
00186     { static constexpr bool value = true; };
00187
00188     template<typename U>
00189     struct Is_float
00190     { static constexpr bool value = false; };
00191
00192     template<>
00193     struct Is_float<float>
00194     { static constexpr bool value = true; };
00195
00196     template<>
00197     struct Is_float<double>
00198     { static constexpr bool value = true; };
00199
00200     template<>
00201     struct Is_float<long double>
00202     { static constexpr bool value = true; };
00203
00204     template<typename T>
00205     struct Const_traits
00206     {
00207         using Type = T;
00208         using Const_type = const T;

```

```

00209     static constexpr bool value = false;
00210 };
00211
00212 template<typename T>
00213 struct Const_traits<const T>
00214 {
00215     using Type = T;
00216     using Const_type = const T;
00217     static constexpr bool value = true;
00218 };
00219 };
00220
00221 template<typename T>
00222 struct Type_traits
00223 {
00224     static constexpr bool is_unsigned = TT::Is_unsigned<T>::value;
00225     static constexpr bool is_signed = TT::Is_signed<T>::value;
00226
00227     static constexpr bool is_int = TT::Is_int<T>::value;
00228     static constexpr bool is_float = TT::Is_float<T>::value;
00229
00230     static constexpr bool is_pointer = TT::Pointer_traits<T>::value;
00231     static constexpr bool is_pointer_to_member = TT::PMF_traits<T>::value;
00232     static constexpr bool is_reference = TT::Ref_traits<T>::value;
00233
00234     static constexpr bool is_scalar = is_unsigned || is_signed || is_int
00235         || is_pointer || is_pointer_to_member || is_reference;
00236
00237     static constexpr bool is_fundamental = is_unsigned || is_signed || is_float
00238         || Conversion<T, void>::same_type;
00239
00240     static constexpr bool is_const = TT::Const_traits<T>::value;
00241
00253     static constexpr unsigned long align(unsigned long address)
00254     { return (address + alignof(T) - 1UL) & ~(alignof(T) - 1UL); }
00255
00256     using Param_type = typename Select<is_scalar, T, typename TT::Add_ref<typename
TT::Const_traits<T>::Const_type>::Type>::Type;
00257
00258     using Pointee_type = typename TT::Pointer_traits<T>::Pointee;
00259     using Referee_type = typename TT::Ref_traits<T>::Referee;
00260     using Non_const_type = typename TT::Const_traits<T>::Type;
00261     using Const_type = typename TT::Const_traits<T>::Const_type;
00262 };
00263
00264 };

```

16.197 dlist

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2011 Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *     economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008
00009 #pragma once
00010
00011 namespace cxx {
00012
00013 class D_list_item
00014 {
00015 public:
00016     constexpr D_list_item() : _dli_next(nullptr), _dli_prev(nullptr) {}
00017
00018     D_list_item(D_list_item const &) = delete;
00019     void operator = (D_list_item const &) = delete;
00020
00021 private:
00022     friend struct D_list_item_policy;
00023
00024     D_list_item *_dli_next, *_dli_prev;
00025 };
00026
00027 struct D_list_item_policy
00028 {
00029     typedef D_list_item Item;
00030     static D_list_item *&prev(D_list_item *e) { return e->_dli_prev; }
00031     static D_list_item *&next(D_list_item *e) { return e->_dli_next; }
00032     static D_list_item *&prev(D_list_item const *e) { return e->_dli_prev; }
00033     static D_list_item *&next(D_list_item const *e) { return e->_dli_next; }
00034 };

```

```

00035
00036 template< typename T >
00037 struct Sd_list_head_policy
00038 {
00039     typedef T *Head_type;
00040     static T *head(Head_type h) { return h; }
00041     static void set_head(Head_type &h, T *v) { h = v; }
00042 };
00043
00044 template<
00045     typename T,
00046     typename C = D_list_item_policy
00047 >
00048 class D_list_cyclic
00049 {
00050 protected:
00051     template< typename VALUE, typename ITEM >
00052     class __Iterator
00053     {
00054     public:
00055         typedef VALUE *Value_type;
00056         typedef VALUE *value_type;
00057
00058         __Iterator() {}
00059
00060         bool operator == (__Iterator const &o) const
00061         { return _c == o._c; }
00062
00063         bool operator != (__Iterator const &o) const
00064         { return _c != o._c; }
00065
00066         __Iterator &operator ++ ()
00067         {
00068             _c = C::next(_c);
00069             return *this;
00070         }
00071
00072         __Iterator &operator -- ()
00073         {
00074             _c = C::prev(_c);
00075             return *this;
00076         }
00077
00078         Value_type operator * () const { return static_cast<Value_type>(_c); }
00079         Value_type operator -> () const { return static_cast<Value_type>(_c); }
00080
00081     protected:
00082         friend class D_list_cyclic;
00083
00084         explicit __Iterator(ITEM *s) : _c(s) {}
00085
00086         ITEM *_c;
00087     };
00088
00089     public:
00090         typedef T *Value_type;
00091         typedef T *value_type;
00092         typedef __Iterator<T, typename C::Item> Iterator;
00093         typedef Iterator Const_iterator;
00094
00095         static void remove(T *e)
00096         {
00097             C::next(C::prev(e)) = C::next(e);
00098             C::prev(C::next(e)) = C::prev(e);
00099             C::next(e) = nullptr;
00100         }
00101
00102         static Iterator erase(Iterator const &e)
00103         {
00104             typename C::Item *n = C::next(*e);
00105             remove(*e);
00106             return __iter(n);
00107         }
00108
00109         static Iterator iter(T const *e) { return Iterator(const_cast<T*>(e)); }
00110
00111         static bool in_list(T const *e) { return C::next(const_cast<T*>(e)); }
00112         static bool has_sibling(T const *e) { return C::next(const_cast<T*>(e)) != e; }
00113
00114         static Iterator insert_after(T *e, Iterator const &pos)
00115         {
00116             C::prev(e) = pos._c;
00117             C::next(e) = C::next(pos._c);
00118             C::prev(C::next(pos._c)) = e;
00119             C::next(pos._c) = e;
00120             return pos;
00121         }

```

```

00122
00123 static Iterator insert_before(T *e, Iterator const &pos)
00124 {
00125     C::next(e) = pos._c;
00126     C::prev(e) = C::prev(pos._c);
00127     C::next(C::prev(pos._c)) = e;
00128     C::prev(pos._c) = e;
00129     return pos;
00130 }
00131
00132 protected:
00133 static void self_insert(typename C::Item *e)
00134 { C::next(e) = C::prev(e) = e; }
00135
00136 static void remove_last(T *e)
00137 { C::next(e) = nullptr; }
00138
00139 static void splice_heads(Const_iterator pos, typename C::Item *other_list)
00140 {
00141     typename C::Item *ins_next = pos._c;
00142     typename C::Item *ins_prev = C::prev(pos._c);
00143     typename C::Item *other_head = C::next(other_list);
00144     typename C::Item *other_tail = C::prev(other_list);
00145
00146     C::next(ins_prev) = other_head;
00147     C::prev(other_head) = ins_prev;
00148     C::prev(ins_next) = other_tail;
00149     C::next(other_tail) = ins_next;
00150 }
00151
00152 static Iterator __iter(typename C::Item *e) { return Iterator(e); }
00153 };
00154
00155 template<
00156     typename T,
00157     typename C = D_list_item_policy,
00158     typename H = Sd_list_head_policy<T>,
00159     bool BSS = false
00160 >
00161 class Sd_list : public D_list_cyclic<T, C>
00162 {
00163 private:
00164     typedef D_list_cyclic<T, C> Base;
00165 public:
00166     class Iterator : public Base::Iterator
00167     {
00168     public:
00169         Iterator &operator ++ ()
00170         {
00171             if (this->_c)
00172                 Base::Iterator::operator ++ ();
00173
00174             if (this->_c == _h)
00175                 this->_c = nullptr;
00176
00177             return *this;
00178         }
00179
00180         Iterator &operator -- () = delete;
00181     private:
00182         friend class Sd_list;
00183         explicit Iterator(T *h) : Base::Iterator(h), _h(h) {}
00184         typename C::Item *_h;
00185     };
00186
00187     class R_iterator : public Base::Iterator
00188     {
00189     public:
00190         R_iterator &operator ++ ()
00191         {
00192             if (this->_c)
00193                 Base::Iterator::operator -- ();
00194
00195             if (this->_c == _h)
00196                 this->_c = nullptr;
00197
00198             return *this;
00199         }
00200
00201         R_iterator &operator -- () = delete;
00202     private:
00203         friend class Sd_list;
00204     };

```

```

00215     explicit R_iterator(T *h) : Base::Iterator(h), _h(h) {}
00216     typename C::Item *_h;
00217 };
00218
00219 //typedef typename Base::Iterator Iterator;
00220 enum Pos
00221 { Back, Front };
00222
00223 Sd_list()
00224 {
00225     if (!BSS)
00226         H::set_head(_f, nullptr);
00227 }
00228
00229 bool empty() const { return !H::head(_f); }
00230 T *front() const { return H::head(_f); }
00231
00232 void remove(T *e)
00233 {
00234     T *h = H::head(_f);
00235     if (e == C::next(e)) // must be the last
00236     {
00237         Base::remove_last(e);
00238         H::set_head(_f, nullptr);
00239         return;
00240     }
00241
00242     if (e == H::head(_f))
00243         H::set_head(_f, static_cast<T*>(C::next(h)));
00244
00245     Base::remove(e);
00246 }
00247
00248 Iterator erase(Iterator const &e)
00249 {
00250     Iterator next = e;
00251     ++next;
00252
00253     remove(*e);
00254     return next;
00255 }
00256
00257 void push(T *e, Pos pos)
00258 {
00259     T *h = H::head(_f);
00260     if (!h)
00261     {
00262         Base::self_insert(e);
00263         H::set_head(_f, e);
00264     }
00265     else
00266     {
00267         Base::insert_before(e, this->iter(h));
00268         if (pos == Front)
00269             H::set_head(_f, e);
00270     }
00271 }
00272
00273 void push_back(T *e) { push(e, Back); }
00274 void push_front(T *e) { push(e, Front); }
00275 void rotate_to(T *h) { H::set_head(_f, h); }
00276
00277 typename H::Head_type const &head() const & { return _f; }
00278 typename H::Head_type const &head() const && = delete;
00279 typename H::Head_type &head() & { return _f; }
00280
00281 Iterator begin() { return Iterator(H::head(_f)); }
00282 Iterator end() { return Iterator(nullptr); }
00283
00284 R_iterator rbegin()
00285 {
00286     if (head())
00287         return R_iterator(static_cast<T*>(C::prev(H::head(_f))));
00288     return R_iterator(nullptr);
00289 }
00290 R_iterator rend() { return R_iterator(nullptr); }
00291
00292 private:
00293     Sd_list(Sd_list const &);
00294     void operator = (Sd_list const &);
00295
00296     typename H::Head_type _f;
00297 };
00298
00299 template<
00300     typename T,
00301     typename C = D_list_item_policy,

```

```

00302     bool BSS = false
00303 >
00304 class D_list : public D_list_cyclic<T, C>
00305 {
00306 private:
00307     typedef D_list_cyclic<T, C> Base;
00308     typedef typename C::Item Internal_type;
00309
00310 public:
00311     enum Pos
00312     { Back, Front };
00313
00314     typedef typename Base::Iterator Iterator;
00315     typedef typename Base::Const_iterator Const_iterator;
00316     typedef T* value_type;
00317     typedef T* Value_type;
00318
00319     D_list() { this->self_insert(&_h); }
00320     ~D_list() { clear(); }
00321
00322     D_list(D_list &o)
00323     {
00324         if (o.empty())
00325         {
00326             this->self_insert(&_h);
00327         }
00328         else
00329         {
00330             Internal_type *p = C::prev(&o._h);
00331             Internal_type *n = C::next(&o._h);
00332             C::prev(&_h) = p;
00333             C::next(&_h) = n;
00334             C::next(p) = &_h;
00335             C::prev(n) = &_h;
00336             o.self_insert(&o._h);
00337         }
00338     }
00339
00340     D_list &operator=(D_list &o)
00341     {
00342         if (&o == this)
00343             return *this;
00344
00345         clear();
00346
00347         if (!o.empty())
00348         {
00349             Internal_type *p = C::prev(&o._h);
00350             Internal_type *n = C::next(&o._h);
00351             C::prev(&_h) = p;
00352             C::next(&_h) = n;
00353             C::next(p) = &_h;
00354             C::prev(n) = &_h;
00355             o.self_insert(&o._h);
00356         }
00357     }
00358
00359     D_list(D_list const &) = delete;
00360     void operator = (D_list const &) = delete;
00361
00362     void splice(Const_iterator pos, D_list &&other)
00363     {
00364         if (other.empty())
00365             return;
00366
00367         Base::splice_heads(pos, &other._h);
00368         other.self_insert(&other._h);
00369     }
00370
00371     bool empty() const { return C::next(&_h) == &_h; }
00372
00373     static void remove(T *e) { Base::remove(e); }
00374     Iterator erase(Iterator const &e) { return Base::erase(e); }
00375
00376     void clear()
00377     {
00378         // Just clear the _dli_next pointers of all elements. It is the indicator
00379         // that an element is not on a list.
00380         Internal_type *i = C::next(&_h);
00381         while (i != &_h)
00382         {
00383             Internal_type *d = i;
00384             i = C::next(i);
00385             C::next(d) = nullptr;
00386         }
00387
00388         this->self_insert(&_h);

```



```

00389     }
00390
00391     void push(T *e, Pos pos)
00392     {
00393         if (pos == Front)
00394             Base::insert_after(e, end());
00395         else
00396             Base::insert_before(e, end());
00397     }
00398
00399     void push_back(T *e) { push(e, Back); }
00400     void push_front(T *e) { push(e, Front); }
00401
00402     T *pop_back()
00403     {
00404         T *ret = *(end()--);
00405         remove(ret);
00406         return ret;
00407     }
00408
00409     T *pop_front()
00410     {
00411         T *ret = *begin();
00412         remove(ret);
00413         return ret;
00414     }
00415
00416     Iterator begin() const { return this->__iter(C::next(const_cast<Internal_type *>(&_h))); }
00417     Iterator end() const { return this->__iter(const_cast<Internal_type *>(&_h)); }
00418
00419     bool has_sibling(T const *e)
00420     {
00421         return C::next(const_cast<T*>(e)) != &_h
00422             || C::prev(const_cast<T*>(e)) != &_h;
00423     }
00424
00425 private:
00426     Internal_type _h;
00427 };
00428
00429 }
00430
00431

```

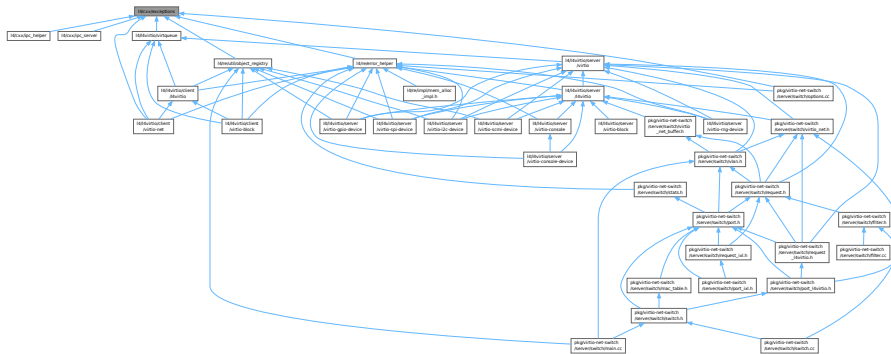
16.198 elide_dtor

```

00001 // vi:set ft=c++: -- Mode: C++ --
00002 /*
00003  * Copyright (C) 2026 Kernkonzept GmbH.
00004  * Author(s): Jan Klötzke <jan.kloetzke@kernkonzept.com>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008
00009 #pragma once
00010
00011 #include <l4/cxx/type_traits>
00012 #include <stddef.h>
00013
00014 namespace cxx {
00015
00016     template< typename T >
00017     class Elide_dtor
00018     {
00019     private:
00020         // Small helper struct so that we don't have to define a global operator new.
00021         struct X : T
00022         {
00023             constexpr void *operator new (size_t, void *p) noexcept { return p; }
00024             constexpr void operator delete (void *) {}
00025
00026             constexpr X() = default;
00027
00028             template<typename ...Args>
00029             constexpr X(Args && ...a) : T(cxx::forward<Args>(a)...) {}
00030
00031             constexpr X(T const &o) : T(o) {}
00032
00033             constexpr X(T &&o) : T(cxx::move(o)) {}
00034         };
00035
00036     public:
00037         template< typename ...Args >
00038         explicit constexpr Elide_dtor(Args && ...args)

```


This graph shows which files directly or indirectly include this file:



Data Structures

- class [L4::Exception_tracer](#)
Back-trace support for exceptions.
- class [L4::Base_exception](#)
Base class for all exceptions, thrown by the [L4Re](#) framework.
- class [L4::Runtime_error](#)
Exception for an abstract runtime error.
- class [L4::Out_of_memory](#)
Exception signalling insufficient memory.
- class [L4::Element_already_exists](#)
Exception for duplicate element insertions.
- class [L4::Unknown_error](#)
Exception for an unknown condition.
- class [L4::Element_not_found](#)
Exception for a failed lookup (element not found).
- class [L4::Invalid_capability](#)
Indicates that an invalid object was invoked.
- class [L4::Com_error](#)
Error conditions during IPC.
- class [L4::Bounds_error](#)
Access out of bounds.

Namespaces

- namespace [L4](#)
[L4](#) low-level kernel interface.

Macros

- `#define L4_CXX_EXCEPTION_BACKTRACE 20`
Number of instruction pointers in backtrace.

16.199.1 Detailed Description

Base exceptions.

Definition in file [exceptions](#).

16.200 exceptions

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014 #pragma once
00015
00016 #include <l4/cxx/l4types.h>
00017 #include <l4/cxx/basic_ostream>
00018 #include <l4/sys/err.h>
00019 #include <l4/sys/capability>
00020
00021
00027
00028 #ifndef L4_CXX_NO_EXCEPTION_BACKTRACE
00029 # define L4_CXX_EXCEPTION_BACKTRACE 20
00030 #endif
00031
00032 #if defined(L4_CXX_EXCEPTION_BACKTRACE)
00033 #include <l4/util/backtrace.h>
00034 #endif
00035
00037 namespace L4
00038 {
00051     class Exception_tracer
00052     {
00053     #if defined(L4_CXX_EXCEPTION_BACKTRACE)
00054     private:
00055         void *_pc_array[L4_CXX_EXCEPTION_BACKTRACE];
00056         int _frame_cnt;
00057
00058     protected:
00062     #if defined(__PIC__)
00063         Exception_tracer() noexcept : _frame_cnt(0) {}
00064     #else
00065         Exception_tracer() noexcept
00066             : _frame_cnt(l4util_backtrace(_pc_array, L4_CXX_EXCEPTION_BACKTRACE)) {}
00067     #endif
00068
00069     public:
00073         void const *const *_pc_array() const noexcept { return _pc_array; }
00077         int frame_count() const noexcept { return _frame_cnt; }
00078     #else
00079     protected:
00083         Exception_tracer() noexcept {}
00084
00085     public:
00089         void const *const *_pc_array() const noexcept { return 0; }
00093         int frame_count() const noexcept { return 0; }
00094     #endif
00095     };
00096
00105     class Base_exception : public Exception_tracer
00106     {
00107     protected:
00109         Base_exception() noexcept {}
00110
00111     public:
00115         virtual char const *str() const noexcept = 0;
00116
00118         virtual ~Base_exception() noexcept {}
00119     };
00120
00128     class Runtime_error : public Base_exception
00129     {
00130     private:

```

```

00131     long _errno;
00132     char _extra[80];
00133
00134 public:
00141     explicit Runtime_error(long err_no, char const *extra = 0) noexcept
00142     : _errno(err_no)
00143     {
00144         if (!extra)
00145             _extra[0] = 0;
00146         else
00147         {
00148             unsigned i = 0;
00149             for (; i < sizeof(_extra) && extra[i]; ++i)
00150                 _extra[i] = extra[i];
00151             _extra[i < sizeof(_extra) ? i : sizeof(_extra) - 1] = 0;
00152         }
00153     }
00154     char const *str() const noexcept override
00155     { return l4sys_errtostr(_errno); }
00156
00162     char const *extra_str() const { return _extra; }
00163     ~Runtime_error() noexcept {}
00164
00170     long err_no() const noexcept { return _errno; }
00171 };
00172
00177 class Out_of_memory : public Runtime_error
00178 {
00179 public:
00181     explicit Out_of_memory(char const *extra = "") noexcept
00182     : Runtime_error(-L4_ENOMEM, extra) {}
00184     ~Out_of_memory() noexcept {}
00185 };
00186
00187
00192 class Element_already_exists : public Runtime_error
00193 {
00194 public:
00195     explicit Element_already_exists(char const *e = "") noexcept
00196     : Runtime_error(-L4_EEXIST, e) {}
00197     ~Element_already_exists() noexcept {}
00198 };
00199
00208 class Unknown_error : public Base_exception
00209 {
00210 public:
00211     Unknown_error() noexcept {}
00212     char const *str() const noexcept override { return "unknown error"; }
00213     ~Unknown_error() noexcept {}
00214 };
00215
00220 class Element_not_found : public Runtime_error
00221 {
00222 public:
00223     explicit Element_not_found(char const *e = "") noexcept
00224     : Runtime_error(-L4_ENOENT, e) {}
00225 };
00226
00234 class Invalid_capability : public Base_exception
00235 {
00236 private:
00237     Cap<void> const _o;
00238
00239 public:
00241     explicit Invalid_capability(Cap<void> const &o) noexcept : _o(o) {}
00245     template< typename T>
00246     explicit Invalid_capability(Cap<T> const &o) noexcept : _o(o.cap()) {}
00247     char const *str() const noexcept override { return "invalid object"; }
00248
00253     Cap<void> const &cap() const noexcept { return _o; }
00254     ~Invalid_capability() noexcept {}
00255 };
00256
00263 class Com_error : public Runtime_error
00264 {
00265 public:
00270     explicit Com_error(long err) noexcept : Runtime_error(err) {}
00271
00272     ~Com_error() noexcept {}
00273 };
00274
00278 class Bounds_error : public Runtime_error
00279 {
00280 public:
00281     explicit Bounds_error(char const *e = "") noexcept
00282     : Runtime_error(-L4_ERANGE, e) {}
00283     ~Bounds_error() noexcept {}

```

```

00284     };
00286 };
00287
00288 inline
00289 L4::BasicOStream &
00290 operator « (L4::BasicOStream &o, L4::Base_exception const &e)
00291 {
00292     o « "Exception: " « e.str() « ".\n";
00293     if (e.frame_count())
00294     {
00295         o « "Backtrace:\n";
00296         for (int i = 0; i < e.frame_count(); ++i)
00297             o « "    " « L4::n_hex(l4_addr_t(e.pc_array()[i])) « '\n';
00298     }
00299
00300     return o;
00301 }
00302
00303 inline
00304 L4::BasicOStream &
00305 operator « (L4::BasicOStream &o, L4::Runtime_error const &e)
00306 {
00307     o « "Exception: " « e.str();
00308     if (e.extra_str() && e.extra_str()[0])
00309         o « ": " « e.extra_str();
00310     o « ".\n";
00311     if (e.frame_count())
00312     {
00313         o « "Backtrace:\n";
00314         for (int i = 0; i < e.frame_count(); ++i)
00315             o « "    " « L4::n_hex(l4_addr_t(e.pc_array()[i])) « '\n';
00316     }
00317
00318     return o;
00319 }

```

16.201 hlist

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2011 Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *     economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008
00009 #pragma once
00010
00011 #include "bits/list_basics.h"
00012 #include "type_traits"
00013
00014 namespace cxx {
00015
00021 template<typename ELEM_TYPE>
00022 class H_list_item_t
00023 {
00024 public:
00030     H_list_item_t() : _n(0), _pn(0) {}
00037     ~H_list_item_t() noexcept { l_remove(); }
00038
00039 private:
00040     H_list_item_t(H_list_item_t const &) = delete;
00041
00042     template<typename T, typename P> friend class H_list;
00043     template<typename T, typename X> friend struct Bits::Basic_list_policy;
00044
00045     void l_remove() noexcept
00046     {
00047         if (!_pn)
00048             return;
00049
00050         *_pn = _n;
00051         if (_n)
00052             _n->_pn = _pn;
00053
00054         _pn = nullptr;
00055     }
00056
00057     H_list_item_t *_n, **_pn;
00058 };
00059
00061 typedef H_list_item_t<void> H_list_item;
00062

```

```

00068 template< typename T, typename POLICY = Bits::Basic_list_policy< T, H_list_item> >
00069 class H_list : public Bits::Basic_list<POLICY>
00070 {
00071 private:
00072     typedef typename POLICY::Item_type Item;
00073     typedef Bits::Basic_list<POLICY> Base;
00074     H_list(H_list const &);
00075     void operator = (H_list const &);
00076
00077 public:
00078     typedef typename Base::Iterator Iterator;
00079
00080     // BSS allocation
00081     explicit H_list(bool x) : Base(x) {}
00082     H_list() : Base() {}
00083
00093     static Iterator iter(T *c) { return Base::__iter(c->Item::_pn); }
00094
00096     static bool in_list(T const *e) { return e->Item::_pn; }
00097
00099     void add(T *e)
00100     {
00101         if (this->_f)
00102             this->_f->_pn = &e->Item::_n;
00103         e->Item::_n = this->_f;
00104         e->Item::_pn = &this->_f;
00105         this->_f = static_cast<Item*>(e);
00106     }
00107
00109     void push_front(T *e) { add(e); }
00110
00116     T *pop_front()
00117     {
00118         T *r = this->front();
00119         remove(r);
00120         return r;
00121     }
00122
00133     Iterator insert(T *e, Iterator const &pred)
00134     {
00135         Item **x = &this->_f;
00136         if (pred != Base::end())
00137             x = &(*pred)->_n;
00138
00139         e->Item::_n = *x;
00140
00141         if (*x)
00142             (*x)->_pn = &(e->Item::_n);
00143
00144         e->Item::_pn = x;
00145         *x = static_cast<Item*>(e);
00146         return iter(e);
00147     }
00148
00160     static Iterator insert_after(T *e, Iterator const &pred)
00161     {
00162         Item **x = &(*pred)->_n;
00163         e->Item::_n = *x;
00164
00165         if (*x)
00166             (*x)->_pn = &(e->Item::_n);
00167
00168         e->Item::_pn = x;
00169         *x = static_cast<Item*>(e);
00170         return iter(e);
00171     }
00172
00180     static void insert_before(T *e, Iterator const &succ)
00181     {
00182         Item **x = Base::__get_internal(succ);
00183
00184         e->Item::_n = *x;
00185         e->Item::_pn = x;
00186
00187         if (*x)
00188             (*x)->_pn = &e->Item::_n;
00189
00190         *x = static_cast<Item*>(e);
00191     }
00192
00204     static void replace(T *p, T *e)
00205     {
00206         e->Item::_n = p->Item::_n;
00207         e->Item::_pn = p->Item::_pn;
00208         *(p->Item::_pn) = static_cast<Item*>(e);
00209         if (e->Item::_n)
00210             e->Item::_n->_pn = &(e->Item::_n);

```

```

00211
00212     p->Item::_pn = nullptr;
00213 }
00214
00220 static void remove(T *e)
00221 { e->Item::l_remove(); }
00222
00236 static Iterator erase(Iterator const &e)
00237 { e->Item::l_remove(); return e; }
00238 };
00239
00247 template< typename T >
00248 struct H_list_t : H_list<T, Bits::Basic_list_policy< T, H_list_item_t<T> > >
00249 {
00250     H_list_t() = default;
00251     H_list_t(bool b)
00252     : H_list<T, Bits::Basic_list_policy< T, H_list_item_t<T> > >(b)
00253     {};
00254 };
00255
00256 template< typename T >
00257 class H_list_bss : public H_list<T>
00258 {
00259 public:
00260     H_list_bss() : H_list<T>(true) {}
00261 };
00262
00263 template< typename T >
00264 class H_list_t_bss : public H_list_t<T>
00265 {
00266 public:
00267     H_list_t_bss() : H_list_t<T>(true) {}
00268 };
00269
00270
00271 }

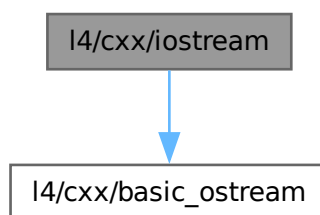
```

16.202 I4/cxx/iostream File Reference

IO Stream.

```
#include <l4/cxx/basic_ostream>
```

Include dependency graph for iostream:



Namespaces

- namespace [L4](#)
L4 low-level kernel interface.

Variables

- `BasicOStream L4::cout`
Standard output stream.
- `BasicOStream L4::cerr`
Standard error stream.

16.202.1 Detailed Description

IO Stream.

Definition in file [iostream](#).

16.203 iostream

[Go to the documentation of this file.](#)

```
00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008  * (c) 2004-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *          Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010  *          economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014 #pragma once
00015
00016 #include <l4/cxx/basic_ostream>
00017
00018 namespace L4 {
00019
00023     extern BasicOStream cout;
00024
00028     extern BasicOStream cerr;
00029
00030     extern void iostream_init();
00031
00032     static void __attribute__((used, constructor)) __iostream_init()
00033     { iostream_init(); }
00034 };
```

16.204 l4/cxx/ipc_helper File Reference

IPC helper.

```
#include <l4/cxx/exceptions>
#include <l4/sys/utcb.h>
```

- namespace **L4**
L4 low-level kernel interface.

- void [L4::throw_ipc_exception](#) (L4::Cap< void > const &o, l4_msgtag_t const &err, l4_utcb_t *utcb)
Throw an [L4 IPC error](#) as exception.
- void [L4::throw_ipc_exception](#) (void const *o, l4_msgtag_t const &err, l4_utcb_t *utcb)
Throw an [L4 IPC error](#) as exception.

IPC helper.

Definition in file [ipc_helper](#).

16.205 ipc_helper

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013 #pragma once
00014
00015 #include <l4/cxx/exceptions>
00016 #include <l4/sys/utcb.h>
00017
00021
00022 namespace L4
00023 {
00024 #ifdef __EXCEPTIONS
00033     inline void
00034     throw_ipc_exception([[maybe_unused]] L4::Cap<void> const &o,
00035                         l4_msgtag_t const &err, l4_utcb_t *utcb)
00036     {
00037         if (err.has_error())
00038             throw (L4::Com_error(l4_error_u(err, utcb)));
00039     }
00040
00049     inline void
00050     throw_ipc_exception(void const *o, l4_msgtag_t const &err,
00051                         l4_utcb_t *utcb)
00052     { throw_ipc_exception(L4::Cap<void>(o), err, utcb); }
00053 #endif
00054
00055 }

```

16.206 l4/cxx/ipc_stream File Reference

IPC stream.

```

#include <l4/sys/ipc.h>
#include <l4/sys/capability>
#include <l4/sys/cxx/ipc_types>
#include <l4/sys/cxx/ipc_varg>
#include <l4/cxx/type_traits>
#include <l4/cxx/minmax>

```


Namespaces

- namespace [L4](#)
L4 low-level kernel interface.
- namespace [L4::lpc](#)
IPC related functionality.

Functions

- `template<typename T>`
`Internal::Buf_cp_out< T > L4::lpc::buf_cp_out (T const *v, unsigned long size)`
Insert an array into an [lpc::Ostream](#).
- `template<typename T>`
`Internal::Buf_cp_in< T > L4::lpc::buf_cp_in (T *v, unsigned long &size)`
Extract an array from an [lpc::Istream](#).
- `template<typename T>`
`Str_cp_in< T > L4::lpc::str_cp_in (T *v, unsigned long &size)`
Create a [Str_cp_in](#) for the given values.
- `template<typename T>`
`Msg_ptr< T > L4::lpc::msg_ptr (T *&p)`
Create an [Msg_ptr](#) to adjust the given pointer.
- `template<typename T>`
`Internal::Buf_in< T > L4::lpc::buf_in (T *&v, unsigned long &size)`
Return a pointer to stream array data.
- `L4::lpc::Istream & operator>> (L4::lpc::Istream &s, bool &v)`
*Extract one element of type *T* from the stream *s*.*
- `L4::lpc::Istream & operator>> (L4::lpc::Istream &s, l4_msgtag_t &v)`
*Extract the *L4* message tag from the stream *s*.*
- `template<typename T>`
`L4::lpc::Istream & operator>> (L4::lpc::Istream &s, L4::lpc::Internal::Buf_in< T > const &v)`
*Extract an array of *T* elements from the stream *s*.*
- `template<typename T>`
`L4::lpc::Istream & operator>> (L4::lpc::Istream &s, L4::lpc::Msg_ptr< T > const &v)`
*Extract an element of type *T* from the stream *s*.*
- `template<typename T>`
`L4::lpc::Istream & operator>> (L4::lpc::Istream &s, L4::lpc::Internal::Buf_cp_in< T > const &v)`
*Extract an array of *T* elements from the stream *s*.*
- `template<typename T>`
`L4::lpc::Istream & operator>> (L4::lpc::Istream &s, L4::lpc::Str_cp_in< T > const &v)`
Extract a zero-terminated string from the stream.
- `L4::lpc::Ostream & operator<< (L4::lpc::Ostream &s, bool v)`
*Insert an element to type *T* into the stream *s*.*
- `L4::lpc::Ostream & operator<< (L4::lpc::Ostream &s, l4_msgtag_t const &v)`
*Insert the *L4* message tag into the stream *s*.*
- `template<typename T>`
`L4::lpc::Ostream & operator<< (L4::lpc::Ostream &s, L4::lpc::Internal::Buf_cp_out< T > const &v)`
*Insert an array with elements of type *T* into the stream *s*.*
- `L4::lpc::Ostream & operator<< (L4::lpc::Ostream &s, char const *v)`
*Insert a zero terminated character string into the stream *s*.*
- `template<typename T>`
`T L4::lpc::read (Istream &s)`
Read a value out of a stream.

16.206.1 Detailed Description

IPC stream.

Definition in file [ipc_stream](#).

16.206.2 Function Documentation

16.206.2.1 `operator<<()` [1/4]

```
L4::Ipc::Ostream & operator<< (  
    L4::Ipc::Ostream & s,  
    bool v) [inline]
```

Insert an element to type `T` into the stream `s`.

Parameters

| | |
|----------------|---|
| <code>s</code> | The stream to insert the element <code>v</code> . |
| <code>v</code> | The element to insert. |

Returns

The stream `s`.

Definition at line [1197](#) of file [ipc_stream](#).

References [L4::Ipc::Ostream::put\(\)](#).

Here is the call graph for this function:



16.206.2.2 operator<<() [2/4]

```
L4::Ipc::Ostream & operator<< (
    L4::Ipc::Ostream & s,
    char const * v) [inline]
```

Insert a zero terminated character string into the stream *s*.

Parameters

| | |
|----------|--|
| <i>s</i> | The stream to insert the string <i>v</i> . |
| <i>v</i> | The string to insert. |

Returns

The stream *s*.

This operator produces basically the same content as the array insertion, however the length of the array is calculated using `strlen(v) + 1`. The string is copied into the message including the trailing zero.

Definition at line 1269 of file `ipc_stream`.

References `L4::Ipc::Ostream::put()`.

Here is the call graph for this function:

**16.206.2.3 operator<<() [3/4]**

```
template<typename T>
L4::Ipc::Ostream & operator<< (
    L4::Ipc::Ostream & s,
    L4::Ipc::Internal::Buf_cp_out< T > const & v) [inline]
```

Insert an array with elements of type *T* into the stream *s*.

Parameters

| | |
|----------|--|
| <i>s</i> | The stream to insert the array <i>v</i> . |
| <i>v</i> | The array to insert (see <code>Ipc::Buf_cp_out()</code>). |

Returns

The stream *s*.

Definition at line 1248 of file [ipc_stream](#).

References [L4::Ipc::Ostream::put\(\)](#).

Here is the call graph for this function:

**16.206.2.4 operator<<() [4/4]**

```

L4::Ipc::Ostream & operator<< (
    L4::Ipc::Ostream & s,
    l4_msgtag_t const & v) [inline]
  
```

Insert the [L4](#) message tag into the stream *s*.

Parameters

| | |
|----------|---|
| <i>s</i> | The stream to insert the tag <i>v</i> . |
| <i>v</i> | The L4 message tag to insert. |

Returns

The stream *s*.

Note

Only one message tag can be inserted into a stream. Multiple insertions simply overwrite previous insertions.

Definition at line 1232 of file [ipc_stream](#).

References [L4::Ipc::Ostream::tag\(\)](#).

Here is the call graph for this function:



16.206.2.5 operator>>() [1/6]

```
L4::Ipc::Istream & operator>> (
    L4::Ipc::Istream & s,
    bool & v) [inline]
```

Extract one element of type T from the stream s.

Parameters

| | | |
|-----|---|-----------------------------|
| | s | The stream to extract from. |
| out | v | Extracted value. |

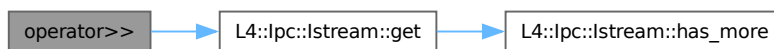
Returns

The stream s.

Definition at line 1044 of file [ipc_stream](#).

References [L4::Ipc::Istream::get\(\)](#).

Here is the call graph for this function:

**16.206.2.6 operator>>>() [2/6]**

```
template<typename T>
L4::Ipc::Istream & operator>>> (
    L4::Ipc::Istream & s,
    L4::Ipc::Internal::Buf_cp_in< T > const & v) [inline]
```

Extract an array of T elements from the stream s.

Parameters

| | | |
|-----|---|--|
| | s | The stream to extract from. |
| out | v | Buffer description to copy the array to (Ipc::Buf_cp_out()). |

Returns

The stream *s*.

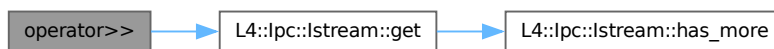
This operator does a copy out of the data into the given buffer.

See `lpc::Buf_in`, `lpc::Buf_cp_in`, and `lpc::Buf_cp_out`.

Definition at line 1151 of file `ipc_stream`.

References [L4::lpc::Istream::get\(\)](#).

Here is the call graph for this function:

**16.206.2.7 operator>>() [3/6]**

```

template<typename T>
L4::Ipc::Istream & operator>> (
    L4::Ipc::Istream & s,
    L4::Ipc::Internal::Buf_in< T > const & v) [inline]
  
```

Extract an array of *T* elements from the stream *s*.

Parameters

| | | |
|-----|----------|---|
| | <i>s</i> | The stream to extract from. |
| out | <i>v</i> | Pointer to the extracted array (<code>ipc_buf_in()</code>). |

Returns

The stream *s*.

This operator actually does not copy out the data in the array, but returns a pointer into the message buffer itself. This means that the data is only valid as long as there is no new data inserted into the stream.

Note

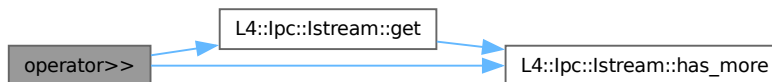
If array does not fit into transmitted words size will be set to zero. Client has to implement check against zero.

See `lpc::Buf_in`, `lpc::Buf_cp_in`, and `lpc::Buf_cp_out`.

Definition at line 1103 of file `ipc_stream`.

References `L4::lpc::Istream::get()`, and `L4::lpc::Istream::has_more()`.

Here is the call graph for this function:

**16.206.2.8 operator>>() [4/6]**

```

template<typename T>
L4::Ipc::Istream & operator>> (
    L4::Ipc::Istream & s,
    L4::Ipc::Msg_ptr< T > const & v) [inline]
  
```

Extract an element of type `T` from the stream `s`.

Parameters

| | | |
|------------------|----------------|-----------------------------------|
| | <code>s</code> | The stream to extract from. |
| <code>out</code> | <code>v</code> | Pointer to the extracted element. |

Returns

The stream `s`.

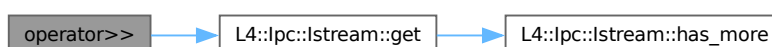
This operator actually does not copy out the data, but returns a pointer into the message buffer itself. This means that the data is only valid as long as there is no new data inserted into the stream.

See `Msg_ptr`.

Definition at line 1130 of file `ipc_stream`.

References `L4::lpc::Istream::get()`.

Here is the call graph for this function:



16.206.2.9 operator>>() [5/6]

```
template<typename T>
L4::Ipc::Istream & operator>> (
    L4::Ipc::Istream & s,
    L4::Ipc::Str_cp_in< T > const & v) [inline]
```

Extract a zero-terminated string from the stream.

Parameters

| | | |
|-----|----------|--|
| | <i>s</i> | The stream to extract from. |
| out | <i>v</i> | Buffer description to copy the array to (lpc::Str_cp_out()). |

Returns

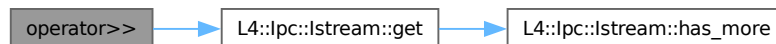
The stream *s*.

This operator does a copy out of the data into the given buffer.

Definition at line 1172 of file [ipc_stream](#).

References [L4::Ipc::Istream::get\(\)](#).

Here is the call graph for this function:

**16.206.2.10 operator>>() [6/6]**

```
L4::Ipc::Istream & operator>> (
    L4::Ipc::Istream & s,
    l4_msgtag_t & v) [inline]
```

Extract the [L4](#) message tag from the stream *s*.

Parameters

| | | |
|-----|----------|-----------------------------|
| | <i>s</i> | The stream to extract from. |
| out | <i>v</i> | The extracted tag. |

Returns

The stream `s`.

Definition at line 1078 of file `ipc_stream`.

References `L4::ipc::Istream::tag()`.

Here is the call graph for this function:

**16.207 ipc_stream**

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *                Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *                Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010  *                economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014 #pragma once
00015
00016 #include <l4/sys/ipc.h>
00017 #include <l4/sys/capability>
00018 #include <l4/sys/cxx/ipc_types>
00019 #include <l4/sys/cxx/ipc_varg>
00020 #include <l4/cxx/type_traits>
00021 #include <l4/cxx/minmax>
00022
00023 namespace L4 {
00024 namespace Ipc {
00025
00026 class Ostream;
00027 class Istream;
00028
00029 namespace Internal {
00047 template< typename T >
00048 class Buf_cp_out
00049 {
00050 public:
00057   Buf_cp_out(T const *v, unsigned long size) : _v(v), _s(size) {}
00058
00066   unsigned long size() const { return _s; }
00067
00075   T const *buf() const { return _v; }
00076
00077 private:
00078   friend class Ostream;
00079   T const *_v;
00080   unsigned long _s;
00081 };
00082 }
00083
00099 template< typename T >
00100 Internal::Buf_cp_out<T> buf_cp_out(T const *v, unsigned long size)
00101 { return Internal::Buf_cp_out<T>(v, size); }
00102
00103
00104 namespace Internal {
  
```

```

00117 template< typename T >
00118 class Buf_cp_in
00119 {
00120 public:
00129   Buf_cp_in(T *v, unsigned long &size) : _v(v), _s(&size) {}
00130
00131   unsigned long &size() const { return *_s; }
00132   T *buf() const { return _v; }
00133
00134 private:
00135   friend class Istream;
00136   T *_v;
00137   unsigned long *_s;
00138 };
00139 }
00140
00158 template< typename T >
00159 Internal::Buf_cp_in<T> buf_cp_in(T *v, unsigned long &size)
00160 { return Internal::Buf_cp_in<T>(v, size); }
00161
00177 template< typename T >
00178 class Str_cp_in
00179 {
00180 public:
00189   Str_cp_in(T *v, unsigned long &size) : _v(v), _s(&size) {}
00190
00191   unsigned long &size() const { return *_s; }
00192   T *buf() const { return _v; }
00193
00194 private:
00195   friend class Istream;
00196   T *_v;
00197   unsigned long *_s;
00198 };
00199
00212 template< typename T >
00213 Str_cp_in<T> str_cp_in(T *v, unsigned long &size)
00214 { return Str_cp_in<T>(v, size); }
00215
00228 template< typename T >
00229 class Msg_ptr
00230 {
00231 private:
00232   T **_p;
00233 public:
00240   explicit Msg_ptr(T *&p) : _p(&p) {}
00241   void set(T *p) const { *_p = p; }
00242 };
00243
00251 template< typename T >
00252 Msg_ptr<T> msg_ptr(T *&p)
00253 { return Msg_ptr<T>(p); }
00254
00255 namespace Internal {
00270 template< typename T >
00271 class Buf_in
00272 {
00273 public:
00280   Buf_in(T *&v, unsigned long &size) : _v(&v), _s(&size) {}
00281
00282   void set_size(unsigned long s) const { *_s = s; }
00283   T *&buf() const { return *_v; }
00284
00285 private:
00286   friend class Istream;
00287   T **_v;
00288   unsigned long *_s;
00289 };
00290 }
00291
00309 template< typename T >
00310 Internal::Buf_in<T> buf_in(T *&v, unsigned long &size)
00311 { return Internal::Buf_in<T>(v, size); }
00312
00313 namespace Utcstream_check
00314 {
00315   static bool check_utcb_data_offset(unsigned sz)
00316   { return sz > sizeof(l4_umword_t) * L4_UTCB_GENERIC_DATA_SIZE; }
00317 }
00318
00319
00334 class Istream
00335 {
00336 public:
00348   Istream(l4_utcb_t *utcb)
00349   : _tag(), _utcb(utcb),

```

```

00350     _current_msg(reinterpret_cast<char*>(l4_utcb_mr_u(utcb)->mr)),
00351     _pos(0), _current_buf(0)
00352 {}
00353
00358 void reset()
00359 {
00360     _pos = 0;
00361     _current_buf = 0;
00362     _current_msg = reinterpret_cast<char*>(l4_utcb_mr_u(utcb)->mr);
00363 }
00364
00368 template< typename T >
00369 bool has_more(unsigned long count = 1)
00370 {
00371     auto const max_bytes = L4_UTCB_GENERIC_DATA_SIZE * sizeof(l4_umword_t);
00372     unsigned apos = cxx::Type_traits<T>::align(_pos);
00373     return (count <= max_bytes / sizeof(T))
00374         && (apos + (sizeof(T) * count)
00375             <= _tag.words() * sizeof(l4_umword_t));
00376 }
00377
00382
00393 template< typename T >
00394 unsigned long get(T *buf, unsigned long elems)
00395 {
00396     if (L4_UNLIKELY(!has_more<T>(elems)))
00397         return 0;
00398
00399     unsigned long size = elems * sizeof(T);
00400     _pos = cxx::Type_traits<T>::align(_pos);
00401
00402     __builtin_memcpy(buf, _current_msg + _pos, size);
00403     _pos += size;
00404     return elems;
00405 }
00406
00407
00413 template< typename T >
00414 void skip(unsigned long elems)
00415 {
00416     if (L4_UNLIKELY(!has_more<T>(elems)))
00417         return;
00418
00419     unsigned long size = elems * sizeof(T);
00420     _pos = cxx::Type_traits<T>::align(_pos);
00421     _pos += size;
00422 }
00423
00438 template< typename T >
00439 unsigned long get(Msg_ptr<T> const &buf, unsigned long elems = 1)
00440 {
00441     if (L4_UNLIKELY(!has_more<T>(elems)))
00442         return 0;
00443
00444     unsigned long size = elems * sizeof(T);
00445     _pos = cxx::Type_traits<T>::align(_pos);
00446
00447     buf.set(reinterpret_cast<T*>(_current_msg + _pos));
00448     _pos += size;
00449     return elems;
00450 }
00451
00452
00463 template< typename T >
00464 bool get(T &v)
00465 {
00466     if (L4_UNLIKELY(!has_more<T>()))
00467     {
00468         v = T();
00469         return false;
00470     }
00471
00472     _pos = cxx::Type_traits<T>::align(_pos);
00473     v = *(reinterpret_cast<T*>(_current_msg + _pos));
00474     _pos += sizeof(T);
00475     return true;
00476 }
00477
00478
00479 bool get(Ipc::Varg *va)
00480 {
00481     Ipc::Varg::Tag t;
00482     if (!has_more<Ipc::Varg::Tag>())
00483     {
00484         va->tag(0);
00485         return 0;
00486     }

```

```

00487     get(t);
00488     va->tag(t);
00489     char const *d;
00490     get(msg_ptr(d), va->length());
00491     va->data(d);
00492
00493     return 1;
00494 }
00495
00505 l4_msgtag_t tag() const { return _tag; }
00506
00507
00517 l4_msgtag_t &tag() { return _tag; }
00518
00520
00525 inline bool put(Rcv_fpage const &);
00526
00531 inline bool put(Small_buf const &);
00532
00533
00538
00548 inline l4_msgtag_t wait(l4_umword_t *src)
00549 { return wait(src, L4_IPC_NEVER); }
00550
00561 inline l4_msgtag_t wait(l4_umword_t *src, l4_timeout_t timeout);
00562
00572 inline l4_msgtag_t receive(l4_cap_idx_t src)
00573 { return receive(src, L4_IPC_NEVER); }
00574
00575 inline l4_msgtag_t receive(l4_cap_idx_t src, l4_timeout_t timeout);
00576
00577
00581 inline l4_utcb_t *utcb() const { return _utcb; }
00582
00583 protected:
00584     l4_msgtag_t _tag;
00585     l4_utcb_t *_utcb;
00586     char *_current_msg;
00587     unsigned _pos;
00588     unsigned char _current_buf;
00589 };
00590
00591 class Istream_copy : public Istream
00592 {
00593 private:
00594     l4_msg_regs_t _mrs;
00595
00596 public:
00597     Istream_copy(Istream const &o) : Istream(o), _mrs(*l4_utcb_mr_u(o.utcb()))
00598     {
00599         // do some reverse mr to utcb trickery
00600         _utcb = reinterpret_cast<l4_utcb_t *>
00601             (reinterpret_cast<l4_addr_t>(&_mrs)
00602              - reinterpret_cast<l4_addr_t>(l4_utcb_mr_u(nullptr)));
00603         _current_msg = reinterpret_cast<char*>(l4_utcb_mr_u(_utcb)->mr);
00604     }
00605
00606 };
00607
00623 class Ostream
00624 {
00625 public:
00629     Ostream(l4_utcb_t *utcb)
00630     : _tag(), _utcb(utcb),
00631       _current_msg(reinterpret_cast<char *>(l4_utcb_mr_u(_utcb)->mr)),
00632       _pos(0), _current_item(0)
00633     {}
00634
00638     void reset()
00639     {
00640         _pos = 0;
00641         _current_item = 0;
00642         _current_msg = reinterpret_cast<char*>(l4_utcb_mr_u(_utcb)->mr);
00643     }
00644
00652
00659 template< typename T >
00660 bool put(T *buf, unsigned long size)
00661 {
00662     size *= sizeof(T);
00663     _pos = cxx::Type_traits<T>::align(_pos);
00664     if (Utc_stream_check::check_utcb_data_offset(_pos + size))
00665         return false;
00666
00667     __builtin_memcpy(_current_msg + _pos, buf, size);
00668     _pos += size;
00669     return true;
00670 }

```



```

00671
00672 template< typename T >
00673 bool put(T const &v)
00674 {
00675     _pos = cxx::Type_traits<T>::align(_pos);
00676     if (Utc_stream_check::check_utcb_data_offset(_pos + sizeof(T)))
00677         return false;
00678
00679     *(reinterpret_cast<T*>(_current_msg + _pos)) = v;
00680     _pos += sizeof(T);
00681     return true;
00682 }
00683
00684 int put(Varg const &va)
00685 {
00686     put(va.tag());
00687     put(va.data(), va.length());
00688
00689     return 0;
00690 }
00691
00692 template< typename T >
00693 int put(Varg_t<T> const &va)
00694 { return put(static_cast<Varg const &>(va)); }
00695
00696 l4_msgtag_t tag() const { return _tag; }
00697
00698 l4_msgtag_t &tag() { return _tag; }
00699
00700 inline bool put_snd_item(Snd_fpage const &);
00701
00702 inline l4_msgtag_t send(l4_cap_idx_t dst, long proto = 0, unsigned flags = 0);
00703
00704 inline l4_utcb_t *utcb() const { return _utcb; }
00705 #if 0
00706 unsigned long tell() const
00707 {
00708     unsigned w = l4_bytes_to_mwords(_pos) - _current_item * 2;
00709     _tag = l4_msgtag(0, w, _current_item, 0);
00710 }
00711 #endif
00712 public:
00713     l4_msgtag_t prepare_ipc(long proto = 0, unsigned flags = 0)
00714     {
00715         unsigned w = l4_bytes_to_mwords(_pos) - _current_item * 2;
00716         return l4_msgtag(proto, w, _current_item, flags);
00717     }
00718
00719 // XXX: this is a hack for <l4/sys/cxx/ipc_server> adaption
00720 void set_ipc_params(l4_msgtag_t tag)
00721 {
00722     _pos = (tag.words() + tag.items() * 2) * sizeof(l4_umword_t);
00723     _current_item = tag.items();
00724 }
00725 protected:
00726     l4_msgtag_t _tag;
00727     l4_utcb_t * _utcb;
00728     char *_current_msg;
00729     unsigned _pos;
00730     unsigned char _current_item;
00731 };
00732
00733 class Iostream : public Istream, public Ostream
00734 {
00735 public:
00736     explicit Iostream(l4_utcb_t *utcb)
00737     : Istream(utcb), Ostream(utcb)
00738     {}
00739
00740 // disambiguate those functions
00741     l4_msgtag_t tag() const { return Istream::tag(); }
00742     l4_msgtag_t &tag() { return Istream::tag(); }
00743     l4_utcb_t *utcb() const { return Istream::utcb(); }
00744
00745     void reset()
00746     {
00747         Istream::reset();
00748         Ostream::reset();
00749     }
00750
00751
00752

```

```

00829
00830 using Istream::get;
00831 using Istream::put;
00832 using Ostream::put;
00833
00835
00840
00856 inline l4_msgtag_t call(l4_cap_idx_t dst, l4_timeout_t timeout, long proto = 0);
00857 inline l4_msgtag_t call(l4_cap_idx_t dst, long proto = 0);
00858
00874 inline l4_msgtag_t reply_and_wait(l4_umword_t *src_dst, long proto = 0)
00875 { return reply_and_wait(src_dst, L4_IPC_SEND_TIMEOUT_0, proto); }
00876
00877 inline l4_msgtag_t send_and_wait(l4_cap_idx_t dest, l4_umword_t *src,
00878                                 long proto = 0)
00879 { return send_and_wait(dest, src, L4_IPC_SEND_TIMEOUT_0, proto); }
00880
00897 inline l4_msgtag_t reply_and_wait(l4_umword_t *src_dst,
00898                                 l4_timeout_t timeout, long proto = 0);
00899 inline l4_msgtag_t send_and_wait(l4_cap_idx_t dest, l4_umword_t *src,
00900                                 l4_timeout_t timeout, long proto = 0);
00901 inline l4_msgtag_t reply(l4_timeout_t timeout, long proto = 0);
00902 inline l4_msgtag_t reply(long proto = 0)
00903 { return reply(L4_IPC_SEND_TIMEOUT_0, proto); }
00904
00906 };
00907
00908
00909 inline bool
00910 Ostream::put_snd_item(Snd_fpage const &v)
00911 {
00912     typedef Snd_fpage T;
00913     _pos = cxx::Type_traits<Snd_fpage>::align(_pos);
00914     if (Utc_b_stream_check::check_utcb_data_offset(_pos + sizeof(T))
00915         return false;
00916
00917     *(reinterpret_cast<T*>(_current_msg + _pos)) = v;
00918     _pos += sizeof(T);
00919     ++_current_item;
00920     return true;
00921 }
00922
00923
00924 inline bool
00925 Istream::put(Rcv_fpage const &item)
00926 {
00927     unsigned words = item.forward_mappings() ? 3 : 2;
00928     if (_current_buf >= L4_UTCB_GENERIC_BUFFERS_SIZE - words - 1)
00929         return false;
00930
00931     l4_utcb_br_u(_utcb)->bdr &= ~L4_BDR_OFFSET_MASK;
00932
00933     l4_umword_t *buf
00934         = reinterpret_cast<l4_umword_t *>(&l4_utcb_br_u(_utcb)->br[_current_buf]);
00935     *buf++ = item.base_x();
00936     *buf++ = item.data();
00937     if (item.forward_mappings())
00938         *buf++ = item.rcv_task();
00939     _current_buf += words;
00940     return true;
00941 }
00942
00943
00944 inline bool
00945 Istream::put(Small_buf const &item)
00946 {
00947     if (_current_buf >= L4_UTCB_GENERIC_BUFFERS_SIZE - 2)
00948         return false;
00949
00950     l4_utcb_br_u(_utcb)->bdr &= ~L4_BDR_OFFSET_MASK;
00951
00952     reinterpret_cast<Small_buf*>(&l4_utcb_br_u(_utcb)->br[_current_buf]) = item;
00953     _current_buf += 1;
00954     return true;
00955 }
00956
00957
00958 inline l4_msgtag_t
00959 Ostream::send(l4_cap_idx_t dst, long proto, unsigned flags)
00960 {
00961     l4_msgtag_t tag = prepare_ipc(proto, L4_MSGTAG_FLAGS & flags);
00962     return l4_ipc_send(dst, _utcb, tag, L4_IPC_NEVER);
00963 }
00964
00965 inline l4_msgtag_t
00966 Iostream::call(l4_cap_idx_t dst, l4_timeout_t timeout, long label)
00967 {

```

```

00968     l4_msgtag_t tag = prepare_ipc(label);
00969     tag = l4_ipc_call(dst, Ostream::_utcb, tag, timeout);
00970     Istream::tag() = tag;
00971     Istream::_pos = 0;
00972     return tag;
00973 }
00974
00975 inline l4_msgtag_t
00976 Iostream::call(l4_cap_idx_t dst, long label)
00977 { return call(dst, L4_IPC_NEVER, label); }
00978
00979
00980 inline l4_msgtag_t
00981 Iostream::reply_and_wait(l4_umword_t *src_dst, l4_timeout_t timeout, long proto)
00982 {
00983     l4_msgtag_t tag = prepare_ipc(proto);
00984     tag = l4_ipc_reply_and_wait(Ostream::_utcb, tag, src_dst, timeout);
00985     Istream::tag() = tag;
00986     Istream::_pos = 0;
00987     return tag;
00988 }
00989
00990
00991 inline l4_msgtag_t
00992 Iostream::send_and_wait(l4_cap_idx_t dest, l4_umword_t *src,
00993                        l4_timeout_t timeout, long proto)
00994 {
00995     l4_msgtag_t tag = prepare_ipc(proto);
00996     tag = l4_ipc_send_and_wait(dest, Ostream::_utcb, tag, src, timeout);
00997     Istream::tag() = tag;
00998     Istream::_pos = 0;
00999     return tag;
01000 }
01001
01002 inline l4_msgtag_t
01003 Iostream::reply(l4_timeout_t timeout, long proto)
01004 {
01005     l4_msgtag_t tag = prepare_ipc(proto);
01006     tag = l4_ipc_send(L4_INVALID_CAP | L4_SYSF_REPLY, Ostream::_utcb, tag, timeout);
01007     Istream::tag() = tag;
01008     Istream::_pos = 0;
01009     return tag;
01010 }
01011
01012 inline l4_msgtag_t
01013 Istream::wait(l4_umword_t *src, l4_timeout_t timeout)
01014 {
01015     l4_msgtag_t res;
01016     res = l4_ipc_wait(_utcb, src, timeout);
01017     tag() = res;
01018     _pos = 0;
01019     return res;
01020 }
01021
01022
01023 inline l4_msgtag_t
01024 Istream::receive(l4_cap_idx_t src, l4_timeout_t timeout)
01025 {
01026     l4_msgtag_t res;
01027     res = l4_ipc_receive(src, _utcb, timeout);
01028     tag() = res;
01029     _pos = 0;
01030     return res;
01031 }
01032
01033 } // namespace Ipc
01034 } // namespace L4
01035
01044 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, bool &v) { s.get(v); return s; }
01045 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, int &v) { s.get(v); return s; }
01046 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, long int &v) { s.get(v); return s; }
01047 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, long long int &v) { s.get(v); return s; }
01048 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, unsigned int &v) { s.get(v); return s; }
01049 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, unsigned long int &v) { s.get(v); return s; }
01050 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, unsigned long long int &v) { s.get(v);
    return s; }
01051 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, short int &v) { s.get(v); return s; }
01052 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, unsigned short int &v) { s.get(v); return s; }
01053 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, char &v) { s.get(v); return s; }
01054 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, unsigned char &v) { s.get(v); return s; }
01055 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, signed char &v) { s.get(v); return s; }
01056 inline L4::Ipc::Istream &operator « (L4::Ipc::Istream &s, L4::Ipc::Rcv_fpage const &v) { s.put(v);
    return s; }
01057 inline L4::Ipc::Istream &operator « (L4::Ipc::Istream &s, L4::Ipc::Small_buf const &v) { s.put(v);
    return s; }

```

```

01058 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, L4::Ipc::Snd_fpage &v)
01059 {
01060     l4_umword_t b, d;
01061     s » b » d;
01062     v = L4::Ipc::Snd_fpage(b, d);
01063     return s;
01064 }
01065 inline L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, L4::Ipc::Varg &v)
01066 { s.get(&v); return s; }
01067
01068
01077 inline
01078 L4::Ipc::Istream &operator » (L4::Ipc::Istream &s, l4_msgtag_t &v)
01079 {
01080     v = s.tag();
01081     return s;
01082 }
01083
01101 template< typename T >
01102 inline
01103 L4::Ipc::Istream &operator » (L4::Ipc::Istream &s,
01104                               L4::Ipc::Internal::Buf_in<T> const &v)
01105 {
01106     unsigned long si;
01107     if (s.get(si) && s.has_more<T>(si))
01108         v.set_size(s.get(L4::Ipc::Msg_ptr<T>(v.buf()), si));
01109     else
01110         v.set_size(0);
01111     return s;
01112 }
01113
01128 template< typename T >
01129 inline
01130 L4::Ipc::Istream &operator » (L4::Ipc::Istream &s,
01131                               L4::Ipc::Msg_ptr<T> const &v)
01132 {
01133     s.get(v);
01134     return s;
01135 }
01136
01149 template< typename T >
01150 inline
01151 L4::Ipc::Istream &operator » (L4::Ipc::Istream &s,
01152                               L4::Ipc::Internal::Buf_cp_in<T> const &v)
01153 {
01154     unsigned long sz;
01155     s.get(sz);
01156     v.size() = s.get(v.buf(), cxx::min(v.size(), sz));
01157     return s;
01158 }
01159
01170 template< typename T >
01171 inline
01172 L4::Ipc::Istream &operator » (L4::Ipc::Istream &s,
01173                               L4::Ipc::Str_cp_in<T> const &v)
01174 {
01175     unsigned long sz;
01176     s.get(sz);
01177     unsigned long rsz = s.get(v.buf(), cxx::min(v.size(), sz));
01178     if (rsz < v.size() && v.buf()[rsz - 1])
01179         ++rsz; // add the zero termination behind the received data
01180
01181     if (rsz != 0)
01182         v.buf()[rsz - 1] = 0;
01183
01184     v.size() = rsz;
01185     return s;
01186 }
01187
01188
01197 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, bool v) { s.put(v); return s; }
01198 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, int v) { s.put(v); return s; }
01199 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, long int v) { s.put(v); return s; }
01200 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, long long int v) { s.put(v); return s; }
01201 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, unsigned int v) { s.put(v); return s; }
01202 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, unsigned long int v) { s.put(v); return s; }
01203 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, unsigned long long int v) { s.put(v); return
s; }
01204 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, short int v) { s.put(v); return s; }
01205 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, unsigned short int v) { s.put(v); return s;
}
01206 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, char v) { s.put(v); return s; }
01207 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, unsigned char v) { s.put(v); return s; }
01208 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, signed char v) { s.put(v); return s; }
01209 inline L4::Ipc::Ostream &operator « (L4::Ipc::Ostream &s, L4::Ipc::Snd_fpage const &v) {
s.put_snd_item(v); return s; }
01210 template< typename T >

```

```

01211 inline L4::Ipc::Ostream &operator << (L4::Ipc::Ostream &s, L4::Cap<T> const &v)
01212 { s << L4::Ipc::Snd_fpage(v.fpage()); return s; }
01213
01214 inline L4::Ipc::Ostream &operator << (L4::Ipc::Ostream &s, L4::Ipc::Varg const &v)
01215 { s.put(v); return s; }
01216 template< typename T >
01217 inline L4::Ipc::Ostream &operator << (L4::Ipc::Ostream &s, L4::Ipc::Varg_t<T> const &v)
01218 { s.put(v); return s; }
01219
01231 inline
01232 L4::Ipc::Ostream &operator << (L4::Ipc::Ostream &s, l4_msgtag_t const &v)
01233 {
01234     s.tag() = v;
01235     return s;
01236 }
01237
01246 template< typename T >
01247 inline
01248 L4::Ipc::Ostream &operator << (L4::Ipc::Ostream &s,
01249                               L4::Ipc::Internal::Buf_cp_out<T> const &v)
01250 {
01251     s.put(v.size());
01252     s.put(v.buf(), v.size());
01253     return s;
01254 }
01255
01268 inline
01269 L4::Ipc::Ostream &operator << (L4::Ipc::Ostream &s, char const *v)
01270 {
01271     unsigned long l = __builtin_strlen(v) + 1;
01272     s.put(l);
01273     s.put(v, l);
01274     return s;
01275 }
01276
01277 namespace L4 { namespace Ipc {
01287 template< typename T >
01288 inline
01289 T read(Istream &s) { T t; s >> t; return t; }
01290
01291 } // namespace Ipc
01292 } // namespace L4

```

16.208 ipc_timeout_queue

```

00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2014 Steffen Liebergeld <steffen.liebergeld@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include <l4/cxx/hlist>
00010 #include <l4/sys/cxx/ipc_server_loop>
00011
00012 namespace L4 { namespace Ipc_svr {
00013
00018 class Timeout : public cxx::H_list_item
00019 {
00020     friend class Timeout_queue;
00021 public:
00023     Timeout() : _timeout(0) {}
00024
00026     virtual ~Timeout() = 0;
00027
00034     virtual void expired() = 0;
00035
00042     l4_kernel_clock_t timeout() const
00043     { return _timeout; }
00044
00045 private:
00046     l4_kernel_clock_t _timeout;
00047 };
00048
00049 inline Timeout::~~Timeout() {}
00050
00055 class Timeout_queue
00056 {
00057 public:
00059     typedef L4::Ipc_svr::Timeout Timeout;
00060
00065     l4_kernel_clock_t next_timeout() const

```

```

00066 {
00067     if (auto e = _timeouts.front())
00068         return e->timeout();
00069     return 0;
00070 }
00071 }
00072
00081 bool timeout_expired(l4_kernel_clock_t now) const
00082 {
00083     l4_kernel_clock_t next = next_timeout();
00084     return (next != 0) && (next <= now);
00085 }
00086
00091 void handle_expired_timeouts(l4_kernel_clock_t now)
00092 {
00093     while (!_timeouts.empty())
00094     {
00095         Queue::Iterator top = _timeouts.begin();
00096         if ((*top)->_timeout > now)
00097             return;
00098
00099         Timeout *t = *top;
00100         top = _timeouts.erase(top);
00101         t->expired();
00102     }
00103 }
00104
00111 void add(Timeout *timeout, l4_kernel_clock_t time)
00112 {
00113     timeout->_timeout = time;
00114     Queue::Iterator i = _timeouts.begin();
00115     while (i != _timeouts.end() && (*i)->timeout() < time)
00116         ++i;
00117     _timeouts.insert_before(timeout, i);
00118 }
00119
00126 void remove(Timeout *timeout)
00127 {
00128     _timeouts.remove(timeout);
00129 }
00130
00131 private:
00132     typedef cxx::H_list<Timeout> Queue;
00133     Queue _timeouts;
00134 };
00135
00149 template< typename HOOKS, typename BR_MAN = Br_manager_no_buffers >
00150 class Timeout_queue_hooks : public BR_MAN
00151 {
00152     l4_kernel_clock_t _now()
00153     { return static_cast<HOOKS*>(this)->now(); }
00154
00155     unsigned _timeout_br()
00156     { return this->first_free_br(); }
00157
00158 public:
00160     l4_timeout_t timeout()
00161     {
00162         l4_kernel_clock_t t = queue.next_timeout();
00163         if (t)
00164             return l4_timeout(L4_IPC_TIMEOUT_0, l4_timeout_abs(t, _timeout_br()));
00165         return L4_IPC_SEND_TIMEOUT_0;
00166     }
00167
00169     void setup_wait(l4_utcb_t *utcb, L4::Ipc_svr::Reply_mode mode)
00170     {
00171         // we must handle the timer only when called after a possible reply
00172         // otherwise we probably destroy the reply message.
00173         if (mode == L4::Ipc_svr::Reply_separate)
00174         {
00175             l4_kernel_clock_t now = _now();
00176             if (queue.timeout_expired(now))
00177                 queue.handle_expired_timeouts(now);
00178         }
00179
00180         BR_MAN::setup_wait(utcb, mode);
00181     }
00182
00184     L4::Ipc_svr::Reply_mode before_reply(l4_msgtag_t, l4_utcb_t *)
00185     {
00186         // split up reply and wait when a timeout has expired
00187         if (queue.timeout_expired(_now()))
00188             return L4::Ipc_svr::Reply_separate;
00189         return L4::Ipc_svr::Reply_compound;
00190     }
00191 }

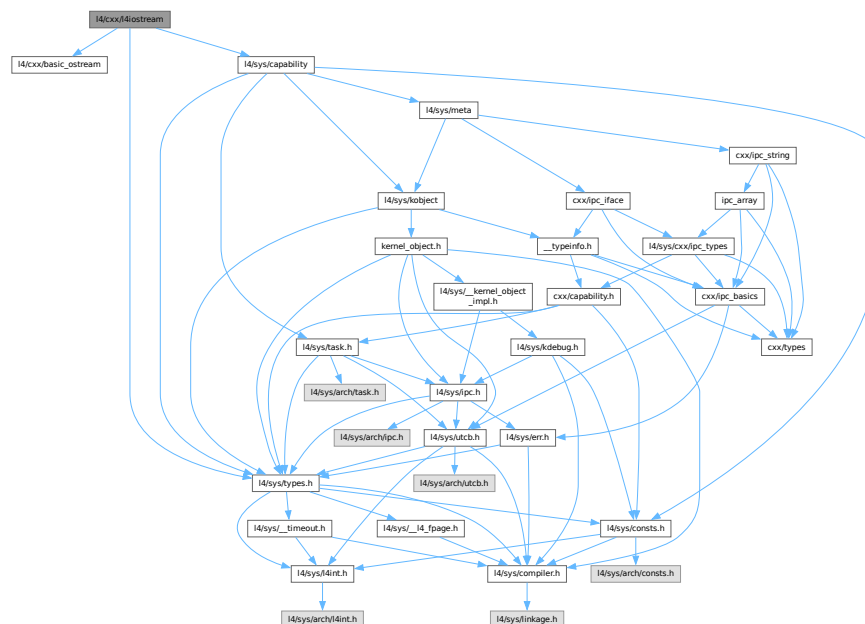
```

```
00202 int add_timeout(Timeout *timeout, l4_kernel_clock_t time) override
00203 {
00204     queue.add(timeout, time);
00205     return 0;
00206 }
00207
00215 int remove_timeout(Timeout *timeout) override
00216 {
00217     queue.remove(timeout);
00218     return 0;
00219 }
00220
00221 Timeout_queue queue;
00222 };
00223
00224 }}
```

16.209 `l4/cxx/l4iostream` File Reference

L4 IO stream.

```
#include <l4/cxx/basic_ostream>
#include <l4/sys/types.h>
#include <l4/sys/capability>
Include dependency graph for l4iostream:
```



16.209.1 Detailed Description

L4 IO stream.

Definition in file [l4iostream.](#)

16.210 l4iostream

[Go to the documentation of this file.](#)

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008  * (c) 2004-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013 #pragma once
00014
00015 #include <l4/cxx/basic_ostream>
00016 #include <l4/sys/types.h>
00017 #include <l4/sys/capability>
00018
00019 inline
00020 L4::BasicOStream &operator « (L4::BasicOStream &o, l4_msgtag_t const &tag)
00021 {
00022     L4::IOBackend::Mode m = o.be_mode();
00023     o « "[l=" « L4::dec « tag.label() « "; w=" « tag.words() « "; i="
00024       « tag.items() « "];";
00025     o.be_mode(m);
00026     return o;
00027 }
00028
00029 template<typename T>
00030 inline
00031 L4::BasicOStream &operator « (L4::BasicOStream &o, L4::Cap<T> const &cap)
00032 {
00033     o « "[C:" « L4::n_hex(cap.cap()) « "];";
00034     return o;
00035 }

```

16.211 l4/cxx/l4types.h File Reference

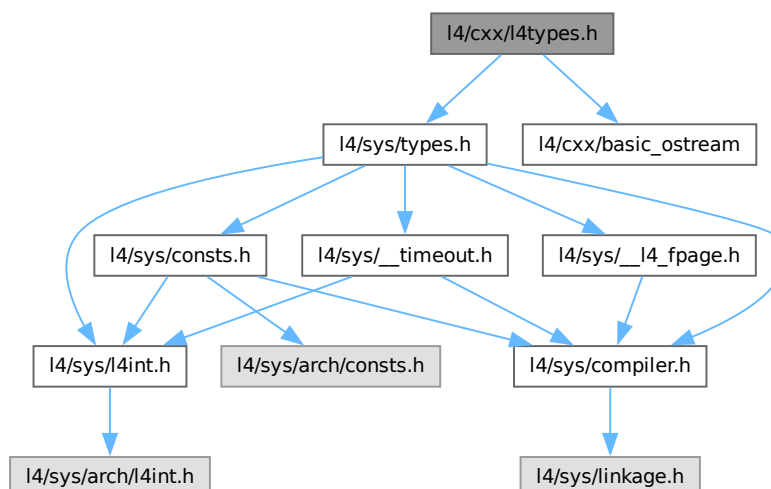
[L4 Types.](#)

```

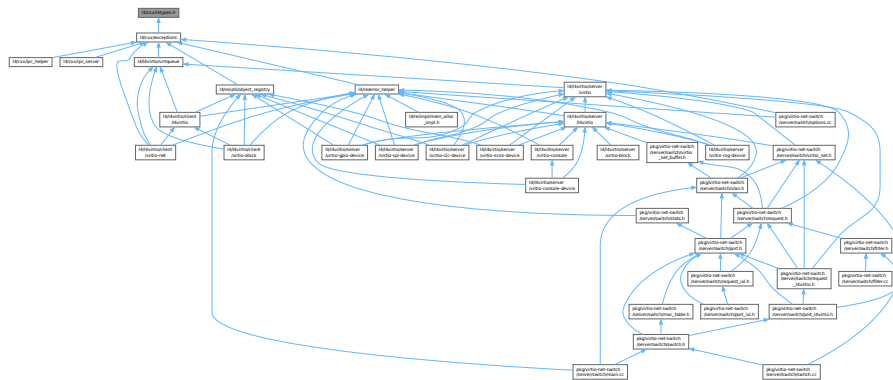
#include <l4/sys/types.h>
#include <l4/cxx/basic_ostream>

```

Include dependency graph for l4types.h:



This graph shows which files directly or indirectly include this file:



16.211.1 Detailed Description

L4 Types.

Definition in file [l4types.h](#).

16.212 l4types.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2004-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011 #pragma once
00012
00013 #include <l4/sys/types.h>
00014 #include <l4/cxx/basic_ostream>

```

16.213 list

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *     economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008
00009 #pragma once
00010
00011 #include <l4/cxx/type_traits>
00012 #include <l4/cxx/std_alloc>
00013 #include <l4/cxx/std_ops>
00014
00015 namespace cxx {
00016 /*
00017  * Classes: List_item, List<D, Alloc>
00018  */
00019
00020 class List_item
00021 {
00022 public:
00023     class Iter

```

```

00035 {
00036 public:
00037     Iter(List_item *c, List_item *f) noexcept : _c(c), _f(f) {}
00038     Iter(List_item *f = 0) noexcept : _c(f), _f(f) {}
00039
00040     List_item *operator * () const noexcept { return _c; }
00041     List_item *operator -> () const noexcept { return _c; }
00042     Iter &operator ++ () noexcept
00043     {
00044         if (!_f)
00045             _c = 0;
00046         else
00047             _c = _c->get_next_item();
00048
00049         if (_c == _f)
00050             _c = 0;
00051
00052         return *this;
00053     }
00054
00055     Iter operator ++ (int) noexcept
00056     { Iter o = *this; operator ++ (); return o; }
00057
00058     Iter &operator -- () noexcept
00059     {
00060         if (!_f)
00061             _c = 0;
00062         else
00063             _c = _c->get_prev_item();
00064
00065         if (_c == _f)
00066             _c = 0;
00067
00068         return *this;
00069     }
00070
00071     Iter operator -- (int) noexcept
00072     { Iter o = *this; operator -- (); return o; }
00073
00075     List_item *remove_me() noexcept
00076     {
00077         if (!_c)
00078             return 0;
00079
00080         List_item *l = _c;
00081         operator ++ ();
00082         l->remove_me();
00083
00084         if (_f == l)
00085             _f = _c;
00086
00087         return l;
00088     }
00089
00090 private:
00091     List_item *_c, *_f;
00092 };
00093
00107 template< typename T, bool Poly = false>
00108 class T_iter : public Iter
00109 {
00110 private:
00111     static bool const P = !Conversion<const T*, const List_item *>::exists
00112         || Poly;
00113
00114     static List_item *cast_to_li(T *i) noexcept
00115     {
00116         if constexpr (P)
00117             return dynamic_cast<List_item*>(i);
00118         else
00119             return i;
00120     }
00121
00122     static T *cast_to_type(List_item *i) noexcept
00123     {
00124         if constexpr (P)
00125             return dynamic_cast<T*>(i);
00126         else
00127             return static_cast<T*>(i);
00128     }
00129
00130 public:
00131
00132     template< typename O >
00133     explicit T_iter(T_iter<O> const &o) noexcept
00134         : Iter(o) { dynamic_cast<T*>(*o); }
00135

```

```

00136     //TIter(CListItem *f) : Iter(f) {}
00137     T_iter(T *f = 0) noexcept : Iter(cast_to_li(f)) {}
00138     T_iter(T *c, T *f) noexcept
00139     : Iter(cast_to_li(c), cast_to_li(f))
00140     {}
00141
00142     inline T *operator * () const noexcept
00143     { return cast_to_type(Iter::operator * ()); }
00144     inline T *operator -> () const noexcept
00145     { return operator * (); }
00146
00147     T_iter<T, Poly> operator ++ (int) noexcept
00148     { T_iter<T, Poly> o = *this; Iter::operator ++ (); return o; }
00149     T_iter<T, Poly> operator -- (int) noexcept
00150     { T_iter<T, Poly> o = *this; Iter::operator -- (); return o; }
00151     T_iter<T, Poly> &operator ++ () noexcept
00152     { Iter::operator ++ (); return *this; }
00153     T_iter<T, Poly> &operator -- () noexcept
00154     { Iter::operator -- (); return *this; }
00155     inline T *remove_me() noexcept;
00156 };
00157
00158 List_item() noexcept : _n(this), _p(this) {}
00159
00160 protected:
00161     List_item(List_item const &) noexcept : _n(this), _p(this) {}
00162
00163 public:
00164     List_item *get_prev_item() const noexcept { return _p; }
00165
00166     List_item *get_next_item() const noexcept { return _n; }
00167
00171     void insert_prev_item(List_item *p) noexcept
00172     {
00173         p->_p->_n = this;
00174         List_item *pr = p->_p;
00175         p->_p = _p;
00176         _p->_n = p;
00177         _p = pr;
00178     }
00179
00181     void insert_next_item(List_item *p) noexcept
00182     {
00183         p->_p->_n = _n;
00184         p->_p = this;
00185         _n->_p = p;
00186         _n = p;
00187     }
00188
00190     void remove_me() noexcept
00191     {
00192         if (_p != this)
00193         {
00194             _p->_n = _n;
00195             _n->_p = _p;
00196         }
00197         _p = _n = this;
00198     }
00199
00208     template< typename C, typename N >
00209     static inline C *push_back(C *head, N *p) noexcept;
00210
00219     template< typename C, typename N >
00220     static inline C *push_front(C *head, N *p) noexcept;
00221
00230     template< typename C, typename N >
00231     static inline C *remove(C *head, N *p) noexcept;
00232
00233 private:
00234     List_item *_n, *_p;
00235 };
00236
00237
00238 /* IMPLEMENTATION -----*/
00239 template< typename C, typename N >
00240 C *List_item::push_back(C *h, N *p) noexcept
00241 {
00242     if (!p)
00243         return h;
00244     if (!h)
00245         return p;
00246     h->insert_prev_item(p);
00247     return h;
00248 }
00249
00250 template< typename C, typename N >
00251 C *List_item::push_front(C *h, N *p) noexcept

```

```

00252 {
00253     if (!p)
00254         return h;
00255     if (h)
00256         h->insert_prev_item(p);
00257     return p;
00258 }
00259
00260 template< typename C, typename N >
00261 C *List_item::remove(C *h, N *p) noexcept
00262 {
00263     if (!p)
00264         return h;
00265     if (!h)
00266         return 0;
00267     if (h == p)
00268     {
00269         if (p == p->_n)
00270             h = 0;
00271         else
00272             h = static_cast<C*>(p->_n);
00273     }
00274     p->remove_me();
00275
00276     return h;
00277 }
00278
00279 template< typename T, bool Poly >
00280 inline
00281 T *List_item::T_iter<T, Poly>::remove_me() noexcept
00282 { return cast_to_type(Iter::remove_me()); }
00283
00284
00285 template< typename T >
00286 class T_list_item : public List_item
00287 {
00288 public:
00289     T *next() const { return static_cast<T*>(List_item::get_next_item()); }
00290     T *prev() const { return static_cast<T*>(List_item::get_prev_item()); }
00291 };
00292
00293
00294 template< typename LI >
00295 class L_list
00296 {
00297 private:
00298     LI *_h;
00299
00300 public:
00301     L_list() : _h(0) {}
00302
00303     void push_front(LI *e) { _h = LI::push_front(_h, e); }
00304     void push_back(LI *e) { _h = LI::push_back(_h, e); }
00305     void insert_before(LI *e, LI *p)
00306     {
00307         p->insert_prev_item(e);
00308         if (_h == p)
00309             _h = e;
00310     }
00311     void insert_after(LI *e, LI *p) { p->insert_next_item(e); }
00312
00313     void remove(LI *e)
00314     { _h = LI::remove(_h, e); }
00315
00316     LI *head() const { return _h; }
00317 };
00318
00319
00325 template< typename D, template<typename A> class Alloc = New_allocator >
00326 class List
00327 {
00328 private:
00329     class E : public List_item
00330     {
00331     public:
00332         E(D const &d) noexcept : data(d) {}
00333         D data;
00334     };
00335
00336 public:
00337     class Node : private E
00338     {};
00339
00340     typedef Alloc<Node> Node_alloc;
00341
00346     class Iter
00347     {

```

```

00348 private:
00349     List_item::T_iter<E> _i;
00350
00351 public:
00352     Iter(E *e) noexcept : _i(e) {}
00353
00354     D &operator * () const noexcept { return (*_i)->data; }
00355     D &operator -> () const noexcept { return (*_i)->data; }
00356
00357     Iter operator ++ (int) noexcept
00358     { Iter o = *this; operator ++ (); return o; }
00359     Iter operator -- (int) noexcept
00360     { Iter o = *this; operator -- (); return o; }
00361     Iter &operator ++ () noexcept { ++_i; return *this; }
00362     Iter &operator -- () noexcept { --_i; return *this; }
00363
00365     operator E* () const noexcept { return *_i; }
00366 };
00367
00368 List(Alloc<Node> const &a = Alloc<Node>()) noexcept : _h(0), _l(0), _a(a) {}
00369
00371 void push_back(D const &d) noexcept
00372 {
00373     void *n = _a.alloc();
00374     if (!n) return;
00375     _h = E::push_back(_h, new (n) E(d));
00376     ++_l;
00377 }
00378
00380 void push_front(D const &d) noexcept
00381 {
00382     void *n = _a.alloc();
00383     if (!n) return;
00384     _h = E::push_front(_h, new (n) E(d));
00385     ++_l;
00386 }
00387
00389 void remove(Iter const &i) noexcept
00390 { E *e = i; _h = E::remove(_h, e); --_l; _a.free(e); }
00391
00393 unsigned long size() const noexcept { return _l; }
00394
00396 D const &operator [] (unsigned long idx) const noexcept
00397 { Iter i = _h; for (; idx && *i; ++i, --idx) { } return *i; }
00398
00400 D &operator [] (unsigned long idx) noexcept
00401 { Iter i = _h; for (; idx && *i; ++i, --idx) { } return *i; }
00402
00404 Iter items() noexcept { return Iter(_h); }
00405
00406 private:
00407     E *_h;
00408     unsigned _l;
00409     Alloc<Node> _a;
00410 };
00411
00412
00413 };
00414

```

16.214 list_alloc

```

00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00004  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009
00010 #pragma once
00011
00012 #include <l4/cxx/arith>
00013 #include <l4/cxx/minmax>
00014 #include <l4/cxx/type_traits>
00015 #include <l4/sys/consts.h>
00016
00017 namespace cxx {
00018
00022 class List_alloc
00023 {
00024 private:

```

```

00025     friend class List_alloc_sanity_guard;
00026
00027     struct Mem_block
00028     {
00029         Mem_block *next;
00030         unsigned long size;
00031     };
00032
00033     Mem_block *_first;
00034
00035     inline void check_overlap(void *, unsigned long) const;
00036     inline void sanity_check_list(char const *, char const *) const;
00037     inline void merge();
00038
00039 public:
00040
00041     List_alloc() : _first(0) {}
00042
00043     inline void free(void *block, unsigned long size, bool initial_free = false);
00044
00045     inline void *alloc(unsigned long size, unsigned long align,
00046         unsigned long lower = 0, unsigned long upper = ~0UL);
00047
00048     inline void *alloc_max(unsigned long min, unsigned long *max,
00049         unsigned long align, unsigned granularity,
00050         unsigned long lower = 0, unsigned long upper = ~0UL);
00051
00052     inline unsigned long avail() const;
00053
00054     template <typename DBG>
00055     void dump_free_list(DBG &out) const;
00056 };
00057
00058 #if !defined (CXX_LIST_ALLOC_SANITY)
00059 class List_alloc_sanity_guard
00060 {
00061 public:
00062     List_alloc_sanity_guard(List_alloc const *, char const *)
00063     {}
00064
00065 };
00066
00067 void
00068 List_alloc::check_overlap(void *, unsigned long) const
00069 {}
00070
00071 void
00072 List_alloc::sanity_check_list(char const *, char const *) const
00073 {}
00074
00075 #else
00076 class List_alloc_sanity_guard
00077 {
00078 private:
00079     List_alloc const *a;
00080     char const *func;
00081
00082 public:
00083     List_alloc_sanity_guard(List_alloc const *a, char const *func)
00084         : a(a), func(func)
00085     { a->sanity_check_list(func, "entry"); }
00086
00087     ~List_alloc_sanity_guard()
00088     { a->sanity_check_list(func, "exit"); }
00089 };
00090
00091 void
00092 List_alloc::check_overlap(void *b, unsigned long s) const
00093 {
00094     unsigned long const mb_align = (1UL << arith::Ld<sizeof(Mem_block)>::value) - 1;
00095     if ((unsigned long)b & mb_align)
00096     {
00097         L4::cerr << "List_alloc(FATAL): trying to free unaligned memory: "
00098             << b << " align=" << arith::Ld<sizeof(Mem_block)>::value << "\n";
00099     }
00100
00101     Mem_block const *c = _first;
00102     for (; c ; c = c->next)
00103     {
00104         unsigned long x_s = (unsigned long)b;
00105         unsigned long x_e = x_s + s;
00106         unsigned long b_s = (unsigned long)c;
00107         unsigned long b_e = b_s + c->size;
00108
00109         if ((x_s >= b_s && x_s < b_e)

```

```

00169         || (x_e > b_s && x_e <= b_e)
00170         || (b_s >= x_s && b_s < x_e)
00171         || (b_e > x_s && b_e <= x_e))
00172     {
00173         L4::cerr << "List_alloc(FATAL): trying to free memory that "
00174                 "is already free: \n ["
00175                 << (void*)x_s << '-' << (void*)x_e << "] overlaps ["
00176                 << (void*)b_s << '-' << (void*)b_e << "]\n";
00177     }
00178 }
00179 }
00180
00181 void
00182 List_alloc::sanity_check_list(char const *func, char const *info) const
00183 {
00184     Mem_block const *c = _first;
00185     for (; c ; c = c->next)
00186     {
00187         if (c->next)
00188         {
00189             if (c >= c->next)
00190             {
00191                 L4::cerr << "List_alloc(FATAL): " << func << ' (' << info
00192                         << "): list order violation\n";
00193             }
00194
00195             if (((unsigned long)c) + c->size > (unsigned long)c->next)
00196             {
00197                 L4::cerr << "List_alloc(FATAL): " << func << ' (' << info
00198                         << "): list order violation\n";
00199             }
00200         }
00201     }
00202 }
00203
00204 #endif
00205
00206 void
00207 List_alloc::merge()
00208 {
00209     List_alloc_sanity_guard __attribute__((unused)) guard(this, __func__);
00210     Mem_block *c = _first;
00211     while (c && c->next)
00212     {
00213         unsigned long f_start = reinterpret_cast<unsigned long>(c);
00214         unsigned long f_end   = f_start + c->size;
00215         unsigned long n_start = reinterpret_cast<unsigned long>(c->next);
00216
00217         if (f_end == n_start)
00218         {
00219             c->size += c->next->size;
00220             c->next = c->next->next;
00221             continue;
00222         }
00223
00224         c = c->next;
00225     }
00226 }
00227
00228 void
00229 List_alloc::free(void *block, unsigned long size, bool initial_free)
00230 {
00231     List_alloc_sanity_guard __attribute__((unused)) guard(this, __func__);
00232
00233     unsigned long const mb_align = (1UL << arith::Ld<sizeof(Mem_block)>::value) - 1;
00234
00235     if (initial_free)
00236     {
00237         // enforce alignment constraint on initial memory
00238         unsigned long nblock = (reinterpret_cast<unsigned long>(block) + mb_align)
00239                               & ~mb_align;
00240         size = (size - (nblock - reinterpret_cast<unsigned long>(block)))
00241              & ~mb_align;
00242         block = reinterpret_cast<void*>(nblock);
00243     }
00244     else
00245         // blow up size to the minimum aligned size
00246         size = (size + mb_align) & ~mb_align;
00247
00248     check_overlap(block, size);
00249
00250     Mem_block **c = &_first;
00251     Mem_block *next = 0;
00252
00253     if (*c)
00254     {
00255         while (*c && *c < block)

```

```

00256         c = &(*c)->next;
00257
00258     next = *c;
00259 }
00260
00261 *c = reinterpret_cast<Mem_block*>(block);
00262
00263 (*c)->next = next;
00264 (*c)->size = size;
00265
00266 merge();
00267 }
00268
00269 void *
00270 List_alloc::alloc_max(unsigned long min, unsigned long *max, unsigned long align,
00271                      unsigned granularity, unsigned long lower,
00272                      unsigned long upper)
00273 {
00274     List_alloc_sanity_guard __attribute__((unused)) guard(this, __func__);
00275
00276     unsigned char const mb_bits = arith::Ld<sizeof(Mem_block)>::value;
00277     unsigned long const mb_align = (1UL << mb_bits) - 1;
00278
00279     // blow minimum up to at least the minimum aligned size of a Mem_block
00280     min = l4_round_size(min, mb_bits);
00281     // truncate maximum to at least the size of a Mem_block
00282     *max = l4_trunc_size(*max, mb_bits);
00283     // truncate maximum size according to granularity
00284     *max = *max & ~(granularity - 1UL);
00285
00286     if (min > *max)
00287         return 0;
00288
00289     unsigned long almask = align ? (align - 1UL) : 0;
00290
00291     // minimum alignment is given by the size of a Mem_block
00292     if (almask < mb_align)
00293         almask = mb_align;
00294
00295     Mem_block **c = &_first;
00296     Mem_block **fit = 0;
00297     unsigned long max_fit = 0;
00298     unsigned long a_lower = (lower + almask) & ~almask;
00299
00300     for (; *c; c = &(*c)->next)
00301     {
00302         // address of free memory block
00303         unsigned long n_start = reinterpret_cast<unsigned long>(*c);
00304
00305         // block too small, next
00306         // XXX: maybe we can skip this and just do the test below
00307         if ((*c)->size < min)
00308             continue;
00309
00310         // block outside region, next
00311         if (upper < n_start || a_lower > n_start + (*c)->size)
00312             continue;
00313
00314         // aligned start address within the free block
00315         unsigned long a_start = (n_start + almask) & ~almask;
00316
00317         // check if aligned start address is behind the block, next
00318         if (a_start - n_start >= (*c)->size)
00319             continue;
00320
00321         a_start = a_start < a_lower ? a_lower : a_start;
00322
00323         // end address would overflow, next
00324         if (min > ~0UL - a_start)
00325             continue;
00326
00327         // block outside region, next
00328         if (a_start + min - 1UL > upper)
00329             continue;
00330
00331         // remaining size after subtracting the padding for the alignment
00332         unsigned long r_size = (*c)->size - a_start + n_start;
00333
00334         // upper limit can limit maximum size
00335         if (a_start + r_size - 1UL > upper)
00336             r_size = upper - a_start + 1UL;
00337
00338         // round down according to granularity
00339         r_size &= ~(granularity - 1UL);
00340
00341         // block too small
00342         if (r_size < min)

```



```

00343         continue;
00344
00345         if (r_size >= *max)
00346         {
00347             fit = c;
00348             max_fit = *max;
00349             break;
00350         }
00351
00352         if (r_size > max_fit)
00353         {
00354             max_fit = r_size;
00355             fit = c;
00356         }
00357     }
00358
00359     if (fit)
00360     {
00361         unsigned long n_start = reinterpret_cast<unsigned long>(*fit);
00362         unsigned long a_lower = (lower + almask) & ~almask;
00363         unsigned long a_start = (n_start + almask) & ~almask;
00364         a_start = a_start < a_lower ? a_lower : a_start;
00365         unsigned long r_size = (*fit)->size - a_start + n_start;
00366
00367         if (a_start > n_start)
00368         {
00369             (*fit)->size -= r_size;
00370             fit = &(*fit)->next;
00371         }
00372         else
00373             *fit = (*fit)->next;
00374
00375         *max = max_fit;
00376         if (r_size == max_fit)
00377             return reinterpret_cast<void *>(a_start);
00378
00379         Mem_block *m = reinterpret_cast<Mem_block*>(a_start + max_fit);
00380         m->next = *fit;
00381         m->size = r_size - max_fit;
00382         *fit = m;
00383         return reinterpret_cast<void *>(a_start);
00384     }
00385
00386     return 0;
00387 }
00388
00389 void *
00390 List_alloc::alloc(unsigned long size, unsigned long align, unsigned long lower,
00391                  unsigned long upper)
00392 {
00393     List_alloc_sanity_guard __attribute__((unused)) guard(this, __func__);
00394
00395     unsigned long const mb_align
00396         = (1UL << arith::Ld<sizeof(Mem_block)>::value) - 1;
00397
00398     // blow up size to the minimum aligned size
00399     size = (size + mb_align) & ~mb_align;
00400
00401     unsigned long almask = align ? (align - 1UL) : 0;
00402
00403     // minimum alignment is given by the size of a Mem_block
00404     if (almask < mb_align)
00405         almask = mb_align;
00406
00407     Mem_block **c = &_first;
00408     unsigned long a_lower = (lower + almask) & ~almask;
00409
00410     for (; *c; c=&(*c)->next)
00411     {
00412         // address of free memory block
00413         unsigned long n_start = reinterpret_cast<unsigned long>(*c);
00414
00415         // block too small, next
00416         // XXX: maybe we can skip this and just do the test below
00417         if ((*c)->size < size)
00418             continue;
00419
00420         // block outside region, next
00421         if (upper < n_start || a_lower > n_start + (*c)->size)
00422             continue;
00423
00424         // aligned start address within the free block
00425         unsigned long a_start = (n_start + almask) & ~almask;
00426
00427         // block too small after alignment, next
00428         if (a_start - n_start >= (*c)->size)
00429             continue;

```

```

00430
00431     a_start = a_start < a_lower ? a_lower : a_start;
00432
00433     // end address would overflow, next
00434     if (size > ~0UL - a_start)
00435         continue;
00436
00437     // block outside region, next
00438     if (a_start + size - 1UL > upper)
00439         continue;
00440
00441     // remaining size after subtracting the padding
00442     // for the alignment
00443     unsigned long r_size = (*c)->size - a_start + n_start;
00444
00445     // block too small
00446     if (r_size < size)
00447         continue;
00448
00449     if (a_start > n_start)
00450     {
00451         // have free space before the allocated block
00452         // shrink the block and set c to the next pointer of that
00453         // block
00454         (*c)->size -= r_size;
00455         c = &(*c)->next;
00456     }
00457     else
00458         // drop the block, c remains the next pointer of the
00459         // previous block
00460         *c = (*c)->next;
00461
00462     // allocated the whole remaining space
00463     if (r_size == size)
00464         return reinterpret_cast<void*>(a_start);
00465
00466     // add a new free block behind the allocated block
00467     Mem_block *m = reinterpret_cast<Mem_block*>(a_start + size);
00468     m->next = *c;
00469     m->size = r_size - size;
00470     *c = m;
00471     return reinterpret_cast<void *>(a_start);
00472 }
00473
00474 return 0;
00475 }
00476
00477 unsigned long
00478 List_alloc::avail() const
00479 {
00480     List_alloc_sanity_guard __attribute__((unused)) guard(this, __FUNCTION__);
00481     Mem_block const *c = _first;
00482     unsigned long a = 0;
00483     while (c)
00484     {
00485         a += c->size;
00486         c = c->next;
00487     }
00488     return a;
00489 }
00490
00491 template <typename DBG>
00492 void
00493 List_alloc::dump_free_list(DBG &out) const
00494 {
00495     Mem_block const *c = _first;
00496     while (c)
00497     {
00498         static constexpr char const *const unitstr[4] =
00499             { "Byte", "KiB", "MiB", "GiB" };
00500
00501         unsigned sz = c->size;
00502         unsigned i;
00503         for (i = 0; i < cxx::array_size(unitstr) && sz > 8 « 10; ++i)
00504             sz »= 10;
00505
00506         out.printf("%12p - %12p (%u %s)\n",
00507             c, reinterpret_cast<char const *>(c) + c->size - 1, sz, unitstr[i]);
00508
00509         c = c->next;
00510     }
00511 }
00512 }
00513
00514 }

```

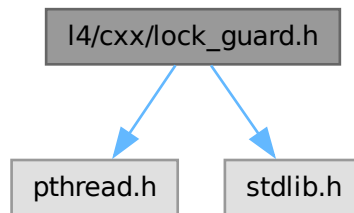
16.215 l4/cxx/lock_guard.h File Reference

Lock guard implementation.

```
#include <pthread.h>
```

```
#include <stdlib.h>
```

Include dependency graph for lock_guard.h:



Data Structures

- class [L4::Lock_guard](#)

Basic lock guard implementation that prevents forgotten unlocks on exit paths from a method or a block of code.

Namespaces

- namespace [L4](#)

[L4](#) low-level kernel interface.

16.215.1 Detailed Description

Lock guard implementation.

Definition in file [lock_guard.h](#).

16.216 lock_guard.h

[Go to the documentation of this file.](#)

```
00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * Copyright (C) 2025 Kernkonzept GmbH.
00004  * Author(s): Martin Decky <martin.decky@kernkonzept.com>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008
00013
00014 #pragma once
00015
00016 #include <pthread.h>
```

```

00017 #include <stdlib.h>
00018
00019 namespace L4 {
00020
00044 class Lock_guard
00045 {
00046 public:
00047     Lock_guard() = delete;
00048     Lock_guard(const Lock_guard &) = delete;
00049     Lock_guard &operator=(const Lock_guard &) = delete;
00050
00059     explicit Lock_guard(pthread_mutex_t &lock) : _lock(&lock)
00060     {
00061         _status = pthread_mutex_lock(_lock);
00062     }
00063
00071     Lock_guard(Lock_guard &&guard) : _lock(guard._lock), _status(guard._status)
00072     {
00073         guard.release();
00074     }
00075
00090     Lock_guard &operator=(Lock_guard &&guard)
00091     {
00092         // Unlock the currently associated mutex (if any).
00093         reset();
00094
00095         // Move the state from the other guard.
00096         _lock = guard._lock;
00097         _status = guard._status;
00098
00099         // Release the mutex from the other guard.
00100         guard.release();
00101
00102         return *this;
00103     }
00104
00110     int status() const
00111     {
00112         return _status;
00113     }
00114
00126     ~Lock_guard()
00127     {
00128         reset();
00129     }
00130
00131 private:
00137     void release()
00138     {
00139         _lock = nullptr;
00140     }
00141
00150     void reset()
00151     {
00152         // No mutex might be associated with this lock guard only if the mutex has
00153         // been moved to a different lock guard.
00154         if (_lock)
00155         {
00156             _status = pthread_mutex_unlock(_lock);
00157             release();
00158         }
00159     }
00160
00161     pthread_mutex_t *_lock;
00162     int _status;
00163 };
00164
00165 } // namespace L4

```

16.217 I4/cxx/main_thread File Reference

Main thread.


```

00019  * the GNU General Public License. This exception does not however
00020  * invalidate any other reasons why the executable file might be covered by
00021  * the GNU General Public License.
00022  */
00023
00024 #ifndef L4_CXX_MAIN_THREAD_H__
00025 #define L4_CXX_MAIN_THREAD_H__
00026
00027 #include <l4/cxx/thread>
00028
00029 namespace cxx {
00030     class MainThread : public Thread
00031     {
00032     public:
00033         MainThread() : Thread(true)
00034         {}
00035     };
00036 };
00037
00038 #endif /* L4_CXX_MAIN_THREAD_H__ */

```

16.219 minmax

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include "type_traits"
00011
00012 namespace cxx
00013 {
00014     // trivial, used to terminate the variadic recursion
00015     template<typename A>
00016     constexpr A const &
00017     min(A const &a)
00018     { return a; }
00019
00020     template<typename A, typename ...ARGS>
00021     constexpr A const &
00022     min(A const &a1, A const &a2, ARGS const &...a)
00023     {
00024         return min((a1 <= a2) ? a1 : a2, a...);
00025     }
00026
00027     template<typename A, typename ...ARGS>
00028     constexpr A const &
00029     min(cxx::identity_t<A> const &a1,
00030         cxx::identity_t<A> const &a2,
00031         ARGS const &...a)
00032     {
00033         return min<A>((a1 <= a2) ? a1 : a2, a...);
00034     }
00035
00036     // trivial, used to terminate the variadic recursion
00037     template<typename A>
00038     constexpr A const &
00039     max(A const &a)
00040     { return a; }
00041
00042     template<typename A, typename ...ARGS>
00043     constexpr A const &
00044     max(A const &a1, A const &a2, ARGS const &...a)
00045     { return max((a1 >= a2) ? a1 : a2, a...); }
00046
00047     template<typename A, typename ...ARGS>
00048     constexpr A const &
00049     max(cxx::identity_t<A> const &a1,
00050         cxx::identity_t<A> const &a2,
00051         ARGS const &...a)
00052     {
00053         return max<A>((a1 >= a2) ? a1 : a2, a...);
00054     }
00055
00056     template<typename T1>
00057     inline
00058     T1 clamp(T1 v, T1 lo, T1 hi)
00059     { return min(hi, max(lo, v)); }
00060 }

```

16.220 numeric

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * Copyright (C) 2025 Kernkonzept GmbH.
00004  * Author(s): Martin Decky <martin.decky@kernkonzept.com>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008
00009 #pragma once
00010
00011 #include <l4/cxx/type_traits>
00012
00013 namespace cxx {
00014
00033 template <typename T>
00034 constexpr T gcd(T a, T b)
00035 {
00036     static_assert(Type_traits<T>::is_unsigned, "Type must be unsigned");
00037
00038     while (b != 0)
00039     {
00040         T remainder = a % b;
00041         a = b;
00042         b = remainder;
00043     }
00044
00045     return a;
00046 }
00047
00067 template <typename T>
00068 constexpr T lcm(T a, T b)
00069 {
00070     static_assert(Type_traits<T>::is_unsigned, "Type must be unsigned");
00071
00072     if (a == 0 || b == 0)
00073         return 0;
00074
00075     return (a / gcd(a, b)) * b;
00076 }
00077
00078 }

```

16.221 observer

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2010 Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *     economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include <l4/cxx/hlist>
00011
00012 namespace cxx {
00013
00014 class Observer : public H_list_item
00015 {
00016 public:
00017     virtual void notify() = 0;
00018 };
00019
00020 class Notifier : public H_list<Observer>
00021 {
00022 public:
00023     void notify()
00024     {
00025         for (Iterator i = begin(); i != end(); ++i)
00026             i->notify();
00027     }
00028 };
00029
00030 }
00031
00032

```


Namespaces

- namespace `cxx`
Our C++ library.

16.222.1 Detailed Description

Pair implementation.

Definition in file `pair`.

16.223 pair

[Go to the documentation of this file.](#)

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include <l4/cxx/type_traits>
00011
00012 namespace cxx {
00013
00014 template< typename First, typename Second >
00015 struct Pair
00016 {
00017     typedef First First_type;
00018     typedef Second Second_type;
00019
00020     First first;
00021     Second second;
00022
00023     template<typename A1, typename A2>
00024     Pair(A1 &&first, A2 &&second)
00025         : first(cxx::forward<A1>(first)), second(cxx::forward<A2>(second)) {}
00026
00027     template<typename A1>
00028     Pair(A1 &&first)
00029         : first(cxx::forward<A1>(first)), second() {}
00030
00031     Pair() = default;
00032 };
00033
00034 template< typename F, typename S >
00035 Pair<F,S> pair(F const &f, S const &s)
00036 { return cxx::Pair<F,S>(f,s); }
00037
00038 template< typename Cmp, typename Typ >
00039 class Pair_first_compare
00040 {
00041 private:
00042     Cmp const &_cmp;
00043
00044 public:
00045     Pair_first_compare(Cmp const &cmp = Cmp()) : _cmp(cmp) {}
00046
00047     bool operator () (Typ const &l, Typ const &r) const
00048     { return _cmp(l.first,r.first); }
00049 };
00050
00051 template< typename OS, typename A, typename B >
00052 inline
00053 OS &operator << (OS &os, cxx::Pair<A,B> const &p)
00054 {
00055     os << p.first << ' ' << p.second;
00056     return os;
00057 }
00058
00059 }
```

16.224 ref_ptr

```

00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *     economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008
00009 #pragma once
00010
00011 #include "type_traits"
00012 #include <stddef.h>
00013 #include <14/sys/compiler.h>
00014
00015 namespace cxx {
00016
00017 template< typename T >
00018 struct Default_ref_counter
00019 {
00020     void h_drop_ref(T *p) noexcept
00021     {
00022         if (p->remove_ref() == 0)
00023             delete p;
00024     }
00025
00026     void h_take_ref(T *p) noexcept
00027     {
00028         p->add_ref();
00029     }
00030 };
00031
00032 struct Ref_ptr_base
00033 {
00034     enum Default_value
00035     { Nil = 0 };
00036 };
00037
00038 template<typename T, template< typename X > class CNT = Default_ref_counter>
00039 class Weak_ptr;
00040
00066 template <
00067     typename T = void,
00068     template< typename X > class CNT = Default_ref_counter
00069 >
00070 class Ref_ptr : public Ref_ptr_base, private CNT<T>
00071 {
00072 private:
00073     typedef decltype(nullptr) Null_type;
00074     typedef Weak_ptr<T, CNT> Wp;
00075
00076 public:
00077     Ref_ptr() noexcept : _p(0) {}
00078
00080     Ref_ptr(Ref_ptr_base::Default_value v)
00081     : _p(reinterpret_cast<T*>(static_cast<unsigned long>(v))) {}
00082
00088     Ref_ptr(Wp const &o) noexcept : _p(o.ptr())
00089     { __take_ref(); }
00090
00092     Ref_ptr(decltype(nullptr) n) noexcept : _p(n) {}
00093
00100     template<typename X>
00101     explicit Ref_ptr(X *o) noexcept : _p(o)
00102     { __take_ref(); }
00103
00114     Ref_ptr(T *o, [[maybe_unused]] bool d) noexcept : _p(o) { }
00115
00121     T *get() const noexcept
00122     {
00123         return _p;
00124     }
00125
00127     T *ptr() const noexcept
00128     {
00129         return _p;
00130     }
00131
00138     T *release() noexcept
00139     {
00140         T *p = _p;
00141         _p = 0;
00142         return p;
00143     }
00144

```

```

00145 ~Ref_ptr() noexcept
00146 { __drop_ref(); }
00147
00148 template<typename OT>
00149 Ref_ptr(Ref_ptr<OT, CNT> const &o) noexcept
00150 {
00151     _p = o.ptr();
00152     __take_ref();
00153 }
00154
00155 Ref_ptr(Ref_ptr<T> const &o) noexcept
00156 {
00157     _p = o._p;
00158     __take_ref();
00159 }
00160
00161 template< typename OT >
00162 void operator = (Ref_ptr<OT> const &o) noexcept
00163 {
00164     __drop_ref();
00165     _p = o.ptr();
00166     __take_ref();
00167 }
00168
00169 void operator = (Ref_ptr<T> const &o) noexcept
00170 {
00171     if (&o == this)
00172         return;
00173
00174     __drop_ref();
00175     _p = o._p;
00176     __take_ref();
00177 }
00178
00179 void operator = (Null_type) noexcept
00180 {
00181     __drop_ref();
00182     _p = 0;
00183 }
00184
00185 template<typename OT>
00186 Ref_ptr(Ref_ptr<OT, CNT> &&o) noexcept
00187 { _p = o.release(); }
00188
00189 Ref_ptr(Ref_ptr<T> &&o) noexcept
00190 { _p = o.release(); }
00191
00192 template< typename OT >
00193 void operator = (Ref_ptr<OT> &&o) noexcept
00194 {
00195     __drop_ref();
00196     _p = o.release();
00197 }
00198
00199 void operator = (Ref_ptr<T> &&o) noexcept
00200 {
00201     if (&o == this)
00202         return;
00203
00204     __drop_ref();
00205     _p = o.release();
00206 }
00207
00208 [[nodiscard]] explicit operator bool () const noexcept { return _p; }
00209
00210 T *operator -> () const noexcept
00211 { return _p; }
00212
00213 [[nodiscard]] bool operator == (Ref_ptr const &o) const noexcept
00214 { return _p == o._p; }
00215
00216 [[nodiscard]] bool operator != (Ref_ptr const &o) const noexcept
00217 { return _p != o._p; }
00218
00219 [[nodiscard]] bool operator < (Ref_ptr const &o) const noexcept
00220 { return _p < o._p; }
00221
00222 [[nodiscard]] bool operator <= (Ref_ptr const &o) const noexcept
00223 { return _p <= o._p; }
00224
00225 [[nodiscard]] bool operator > (Ref_ptr const &o) const noexcept
00226 { return _p > o._p; }
00227
00228 [[nodiscard]] bool operator >= (Ref_ptr const &o) const noexcept
00229 { return _p >= o._p; }
00230
00231 [[nodiscard]] bool operator == (T const *o) const noexcept

```

```

00232     { return _p == o; }
00233
00234     [[nodiscard]] bool operator < (T const *o) const noexcept
00235     { return _p < o; }
00236
00237     [[nodiscard]] bool operator <= (T const *o) const noexcept
00238     { return _p <= o; }
00239
00240     [[nodiscard]] bool operator > (T const *o) const noexcept
00241     { return _p > o; }
00242
00243     [[nodiscard]] bool operator >= (T const *o) const noexcept
00244     { return _p >= o; }
00245
00246 private:
00247     void __drop_ref() noexcept
00248     {
00249         if (_p)
00250             static_cast<CNT<T>*>(this)->h_drop_ref(_p);
00251     }
00252
00253     void __take_ref() noexcept
00254     {
00255         if (_p)
00256             static_cast<CNT<T>*>(this)->h_take_ref(_p);
00257     }
00258
00259     T *_p;
00260 };
00261
00262
00263 template<typename T, template< typename X > class CNT>
00264 class Weak_ptr
00265 {
00266 private:
00267     struct Null_type;
00268     typedef Ref_ptr<T, CNT> Rp;
00269
00270 public:
00271     Weak_ptr() = default;
00272     Weak_ptr(decltype(nullptr)) : _p(nullptr) {}
00273     // backwards 0 ctor
00274     explicit Weak_ptr(int x) noexcept
00275     L4_DEPRECATED("Use initialization from 'nullptr'")
00276     : _p(nullptr)
00277     { if (x != 0) __builtin_trap(); }
00278
00279     Weak_ptr(Rp const &o) noexcept : _p(o.ptr()) {}
00280     explicit Weak_ptr(T *o) noexcept : _p(o) {}
00281
00282     template<typename OT>
00283     Weak_ptr(Weak_ptr<OT, CNT> const &o) noexcept : _p(o.ptr()) {}
00284
00285     Weak_ptr(Weak_ptr<T, CNT> const &o) noexcept : _p(o._p) {}
00286
00287     Weak_ptr<T, CNT> &operator = (const Weak_ptr<T, CNT> &o) = default;
00288
00289     T *get() const noexcept { return _p; }
00290     T *ptr() const noexcept { return _p; }
00291
00292     T *operator -> () const noexcept { return _p; }
00293     operator Null_type const * () const noexcept
00294     { return reinterpret_cast<Null_type const*>(_p); }
00295
00296 private:
00297     T *_p;
00298 };
00299
00300 template<typename OT, typename T> inline
00301 Ref_ptr<OT> ref_ptr_static_cast(Ref_ptr<T> const &o)
00302 { return ref_ptr(static_cast<OT*>(o.ptr())); }
00303
00304 template< typename T >
00305 inline Ref_ptr<T> ref_ptr(T *t)
00306 { return Ref_ptr<T>(t); }
00307
00308 template< typename T >
00309 inline Weak_ptr<T> weak_ptr(T *t)
00310 { return Weak_ptr<T>(t); }
00311
00312
00313 class Ref_obj
00314 {
00315 private:
00316     mutable int _ref_cnt;
00317
00318 public:

```

```

00319   Ref_obj() : _ref_cnt(0) {}
00320   void add_ref() const noexcept { ++_ref_cnt; }
00321   int remove_ref() const noexcept { return --_ref_cnt; }
00322 };
00323
00324 template< typename T, typename... Args >
00325 Ref_ptr<T>
00326 make_ref_obj(Args &&... args)
00327 { return cxx::Ref_ptr<T>(new T(cxx::forward<Args>(args)...)); }
00328
00329 template<typename T, typename U>
00330 Ref_ptr<T>
00331 dynamic_pointer_cast(Ref_ptr<U> const &p) noexcept
00332 {
00333     // our constructor from a naked pointer increments the counter
00334     return Ref_ptr<T>(dynamic_cast<T *>(p.get()));
00335 }
00336
00337 template<typename T, typename U>
00338 Ref_ptr<T>
00339 static_pointer_cast(Ref_ptr<U> const &p) noexcept
00340 {
00341     // our constructor from a naked pointer increments the counter
00342     return Ref_ptr<T>(static_cast<T *>(p.get()));
00343 }
00344
00345 }

```

16.225 l4/cxx/ref_ptr_list File Reference

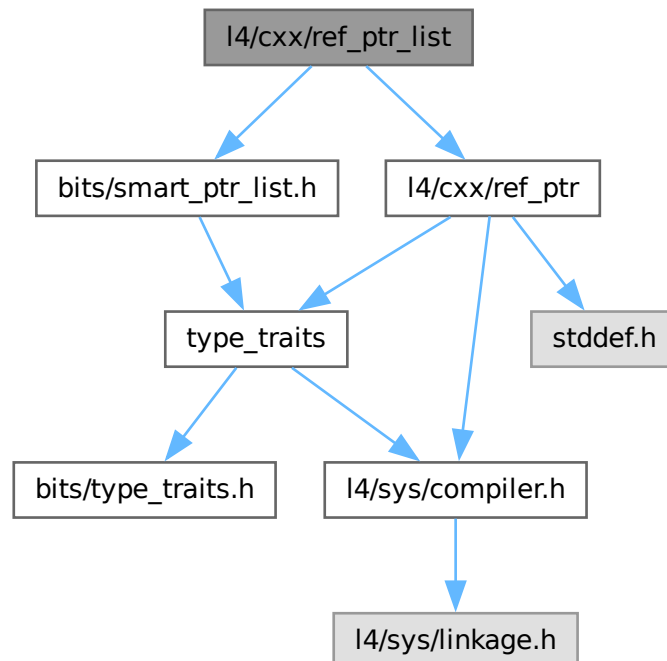
Implementation of a list of ref-ptr-managed objects.

```

#include <l4/cxx/ref_ptr>
#include "bits/smart_ptr_list.h"

```

Include dependency graph for ref_ptr_list:



Data Structures

- struct [cxx::Ref_obj_list_item](#)< T >

Item for list linked via [cxx::Ref_ptr](#) with default reference counting.

Namespaces

- namespace [cxx](#)

Our C++ library.

Typedefs

- template<typename T>
using [cxx::Ref_ptr_list_item](#) = [Bits::Smart_ptr_list_item](#)<T, [cxx::Ref_ptr](#)<T> >
Item for list linked with [cxx::Ref_ptr](#).
- template<typename T>
using [cxx::Ref_ptr_list](#) = [Bits::Smart_ptr_list](#)<[Ref_ptr_list_item](#)<T> >
Single-linked list where elements are connected via a [cxx::Ref_ptr](#).

16.225.1 Detailed Description

Implementation of a list of ref-ptr-managed objects.

Definition in file [ref_ptr_list](#).

16.226 [ref_ptr_list](#)

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * Copyright (C) 2018, 2022, 2024 Kernkonzept GmbH.
00008  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #pragma once
00013
00014 #include <l4/cxx/ref_ptr>
00015
00016 #include "bits/smart_ptr_list.h"
00017
00018 namespace cxx {
00019
00021 template <typename T>
00022 using Ref_ptr_list_item = Bits::Smart_ptr_list_item<T, cxx::Ref_ptr<T> >;
00023
00025 template <typename T>
00026 struct Ref_obj_list_item : public Ref_ptr_list_item<T>, public cxx::Ref_obj {};
00027
00031 template <typename T>
00032 using Ref_ptr_list = Bits::Smart_ptr_list<Ref_ptr_list_item<T> >;
00033
00034 }
```

16.227 result

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002
00003 #pragma once
00004
00005 #include <assert.h>
00006 #include <stdlib.h>
00007
00008 #include "type_traits"
00009 #include "utils"
00010
00011 namespace cxx {
00012
00019 class Error
00020 {
00021     int _err;
00022
00023 public:
00024     constexpr explicit Error(int err) noexcept : _err(err) {}
00025     constexpr Error(Error const &) noexcept = default;
00026     constexpr Error(Error &&) noexcept = default;
00027
00028     constexpr int error() const noexcept
00029     { return _err; }
00030 };
00031
00038 template <typename T>
00039 class Result
00040 {
00041     union
00042     {
00043         T _res;
00044         int _err;
00045     };
00046     bool _has_result;
00047
00048 public:
00055     Result() = delete;
00056
00057     Result(Error err) noexcept
00058     : _err(err.error()), _has_result(false)
00059     {}
00060
00061     template<typename... Args>
00062     explicit Result(in_place_t, Args&&... args)
00063     noexcept (noexcept(T(cxx::forward<Args>(args)...)))
00064     : _res(cxx::forward<Args>(args)...), _has_result(true)
00065     {}
00066
00067     Result(T &&val)
00068     noexcept (noexcept(T(cxx::move(val))))
00069     : _res(cxx::move(val)), _has_result(true)
00070     {}
00071
00072     Result(Result const &o)
00073     noexcept (noexcept(T(o._res)))
00074     : _has_result(o._has_result)
00075     {
00076         if (_has_result)
00077             new (&_res) T(o._res);
00078         else
00079             _err = o._err;
00080     }
00081
00082     Result(Result &&o)
00083     noexcept (noexcept(T(cxx::move(o._res))))
00084     : _has_result(o._has_result)
00085     {
00086         if (_has_result)
00087             new (&_res) T(cxx::move(o._res));
00088         else
00089             _err = o._err;
00090     }
00091
00092     ~Result()
00093     {
00094         if (_has_result)
00095             _res.~T();
00096     }
00097
00098     Result &operator=(Result const &o)
00099     noexcept (noexcept(T(o._res)))
00100     {
00101         if (_has_result)
00102             _res.~T();

```

```

00103
00104     _has_result = o._has_result;
00105     if (_has_result)
00106         new (&_res) T(o._res);
00107     else
00108         _err = o._err;
00109
00110     return *this;
00111 }
00112
00113 Result &operator=(Result &o)
00114 {
00115     if (_has_result)
00116     {
00117         if (o._has_result)
00118             _res = cxx::move(o._res);
00119         else
00120         {
00121             _res.~T();
00122             _has_result = false;
00123             _err = o._err;
00124         }
00125     }
00126     else
00127     {
00128         if (o._has_result)
00129         {
00130             new (&_res) T(cxx::move(o._res));
00131             _has_result = true;
00132         }
00133         else
00134             _err = o._err;
00135     }
00136
00137     return *this;
00138 }
00139
00140 Result &operator=(Error err) noexcept
00141 {
00142     if (_has_result)
00143         _res.~T();
00144
00145     _has_result = false;
00146     _err = err.error();
00147
00148     return *this;
00149 }
00150
00151 Result &operator=(T &val)
00152 {
00153     if (_has_result)
00154         _res = cxx::move(val);
00155     else
00156     {
00157         new (&_res) T(cxx::move(val));
00158         _has_result = true;
00159     }
00160
00161     return *this;
00162 }
00163
00164 explicit operator bool() const noexcept
00165 { return _has_result; }
00166
00167 int error() const noexcept
00168 { return _has_result ? 0 : _err; }
00169
00170 T const &result() const & noexcept
00171 {
00172     assert(_has_result);
00173     if (!_has_result) [[unlikely]]
00174         abort();
00175
00176     return _res;
00177 }
00178
00179 T&& result() && noexcept
00180 {
00181     assert(_has_result);
00182     if (!_has_result) [[unlikely]]
00183         abort();
00184
00185     return cxx::move(_res);
00186 }
00187 };
00188
00189 }

```


16.228 slab_alloc

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009
00010 #pragma once
00011
00012 #include <l4/cxx/std_alloc>
00013 #include <l4/cxx/hlist>
00014 #include <l4/sys/consts.h>
00015
00016 namespace cxx {
00017
00029 template< int Obj_size, int Slab_size = L4_PAGESIZE,
00030          int Max_free = 2, template<typename A> class Alloc = New_allocator >
00031 class Base_slab
00032 {
00033 private:
00034     struct Free_o
00035     {
00036         Free_o *next;
00037     };
00038
00039 protected:
00040     struct Slab_i;
00041
00042 private:
00043     struct Slab_head : public H_list_item
00044     {
00045         unsigned num_free;
00046         Free_o *free;
00047         Base_slab<Obj_size, Slab_size, Max_free, Alloc> *cache;
00048
00049         inline Slab_head() noexcept : num_free(0), free(0), cache(0)
00050         {}
00051     };
00052
00053 // In an empty or partially filled slab, each free object stores a pointer to
00054 // the next free object. Thus, the size of an object needs to be at least the
00055 // size of a pointer.
00056 static_assert(Obj_size >= sizeof(void *),
00057               "Object size must be at least the size of a pointer.");
00058 static_assert(Obj_size <= Slab_size - sizeof(Slab_head),
00059               "Object_size exceeds slab capability.");
00060
00061 public:
00062     enum
00063     {
00064         object_size      = Obj_size,
00065         slab_size        = Slab_size,
00066         objects_per_slab = (Slab_size - sizeof(Slab_head)) / object_size,
00067         max_free_slabs   = Max_free,
00068     };
00069
00070 protected:
00071     struct Slab_store
00072     {
00073         char _o[slab_size - sizeof(Slab_head)];
00074         Free_o *object(unsigned obj) noexcept
00075         { return reinterpret_cast<Free_o*>(_o + object_size * obj); }
00076     };
00077
00078     struct Slab_i : public Slab_store, public Slab_head
00079     {};
00080
00081 public:
00082     typedef Alloc<Slab_i> Slab_alloc;
00083
00084     typedef void Obj_type;
00085
00086 private:
00087     Slab_alloc _alloc;
00088     unsigned _num_free;
00089     unsigned _num_slabs;
00090     H_list<Slab_i> _full_slabs;
00091     H_list<Slab_i> _partial_slabs;
00092     H_list<Slab_i> _empty_slabs;
00093
00094     void add_slab(Slab_i *s) noexcept
00095     {

```

```

00112     s->num_free = objects_per_slab;
00113     s->cache = this;
00114
00115     //L4::cerr << "Slab: " << this << "->add_slab(" << s << ", size="
00116     // << slab_size << "):" << " f=" << s->object(0) << '\n';
00117
00118     // initialize free list
00119     Free_o *f = s->free = s->object(0);
00120     for (unsigned i = 1; i < objects_per_slab; ++i)
00121     {
00122         f->next = s->object(i);
00123         f = f->next;
00124     }
00125     f->next = 0;
00126
00127     // insert slab into cache's list
00128     _empty_slabs.push_front(s);
00129     ++_num_slabs;
00130     ++_num_free;
00131 }
00132
00133 bool grow() noexcept
00134 {
00135     Slab_i *s = _alloc.alloc();
00136     if (!s)
00137         return false;
00138
00139     new (s, cxx::Nothrow()) Slab_i();
00140
00141     add_slab(s);
00142     return true;
00143 }
00144
00145 void shrink() noexcept
00146 {
00147     if (!_alloc.can_free)
00148         return;
00149
00150     while (!_empty_slabs.empty() && _num_free > max_free_slabs)
00151     {
00152         Slab_i *s = _empty_slabs.front();
00153         _empty_slabs.remove(s);
00154         --_num_free;
00155         --_num_slabs;
00156         _alloc.free(s);
00157     }
00158 }
00159
00160 public:
00161 Base_slab(Slab_alloc const &alloc = Slab_alloc()) noexcept
00162 : _alloc(alloc), _num_free(0), _num_slabs(0), _full_slabs(),
00163   _partial_slabs(), _empty_slabs()
00164 {}
00165
00166 ~Base_slab() noexcept
00167 {
00168     while (!_empty_slabs.empty())
00169     {
00170         Slab_i *o = _empty_slabs.front();
00171         _empty_slabs.remove(o);
00172         _alloc.free(o);
00173     }
00174     while (!_partial_slabs.empty())
00175     {
00176         Slab_i *o = _partial_slabs.front();
00177         _partial_slabs.remove(o);
00178         _alloc.free(o);
00179     }
00180     while (!_full_slabs.empty())
00181     {
00182         Slab_i *o = _full_slabs.front();
00183         _full_slabs.remove(o);
00184         _alloc.free(o);
00185     }
00186 }
00187
00188 void *alloc() noexcept
00189 {
00190     H_list<Slab_i> *free = &_partial_slabs;
00191     if (free->empty())
00192         free = &_empty_slabs;
00193
00194     if (free->empty() && !grow())
00195         return 0;
00196
00197     Slab_i *s = free->front();
00198     Free_o *o = s->free;

```

```

00218     s->free = o->next;
00219
00220     if (free == &_empty_slabs)
00221     {
00222         _empty_slabs.remove(s);
00223         --_num_free;
00224     }
00225
00226     --(s->num_free);
00227
00228     if (!s->free)
00229     {
00230         _partial_slabs.remove(s);
00231         _full_slabs.push_front(s);
00232     }
00233     else if (free == &_empty_slabs)
00234         _partial_slabs.push_front(s);
00235
00236     //L4::cerr << this << "->alloc(): " << o << ", of " << s << '\n';
00237
00238     return o;
00239 }
00240
00246 void free(void *_o) noexcept
00247 {
00248     if (!_o)
00249         return;
00250
00251     unsigned long addr = reinterpret_cast<unsigned long>(_o);
00252
00253     // find out the slab the object is in
00254     addr = (addr / slab_size) * slab_size;
00255     Slab_i *s = reinterpret_cast<Slab_i*>(addr);
00256
00257     if (s->cache != this)
00258         return;
00259
00260     Free_o *o = reinterpret_cast<Free_o*>(_o);
00261
00262     o->next = s->free;
00263     s->free = o;
00264
00265     bool was_full = false;
00266
00267     if (!s->num_free)
00268     {
00269         _full_slabs.remove(s);
00270         was_full = true;
00271     }
00272
00273     ++(s->num_free);
00274
00275     if (s->num_free == objects_per_slab)
00276     {
00277         if (!was_full)
00278             _partial_slabs.remove(s);
00279
00280         _empty_slabs.push_front(s);
00281         ++_num_free;
00282         if (_num_free > max_free_slabs)
00283             shrink();
00284
00285         was_full = false;
00286     }
00287     else if (was_full)
00288         _partial_slabs.push_front(s);
00289
00290     //L4::cerr << this << "->free(" << _o << "): of " << s << '\n';
00291 }
00292
00299 unsigned total_objects() const noexcept
00300 { return _num_slabs * objects_per_slab; }
00301
00308 unsigned free_objects() const noexcept
00309 {
00310     unsigned count = 0;
00311
00312     /* count partial slabs first */
00313     for (typename H_list<Slab_i>::Const_iterator s = _partial_slabs.begin();
00314          s != _partial_slabs.end(); ++s)
00315         count += s->num_free;
00316
00317     /* add empty slabs */
00318     count += _num_free * objects_per_slab;
00319
00320     return count;
00321 }

```

```

00322 };
00323
00333 template<typename Type, int Slab_size = L4_PAGESIZE,
00334         int Max_free = 2, template<typename A> class Alloc = New_allocator >
00335 class Slab : public Base_slab<sizeof(Type), Slab_size, Max_free, Alloc>
00336 {
00337 private:
00338     typedef Base_slab<sizeof(Type), Slab_size, Max_free, Alloc> Base_type;
00339 public:
00340
00341     typedef Type Obj_type;
00342
00343     Slab(typename Base_type::Slab_alloc const &alloc
00344         = typename Base_type::Slab_alloc()) noexcept
00345         : Base_slab<sizeof(Type), Slab_size, Max_free, Alloc>(alloc) {}
00346
00347
00355     Type *alloc() noexcept
00356     {
00357         return reinterpret_cast<Type *>(Base_type::alloc());
00358     }
00359
00366     void free(Type *o) noexcept
00367     { Base_slab<sizeof(Type), Slab_size, Max_free, Alloc>::free(o); }
00368 };
00369
00370
00386 template< int Obj_size, int Slab_size = L4_PAGESIZE,
00387         int Max_free = 2, template<typename A> class Alloc = New_allocator >
00388 class Base_slab_static
00389 {
00390 private:
00391     typedef Base_slab<Obj_size, Slab_size, Max_free, Alloc> _A;
00392     static _A _a;
00393 public:
00394     typedef void Obj_type;
00395     enum
00396     {
00398         object_size      = Obj_size,
00400         slab_size        = Slab_size,
00402         objects_per_slab = _A::objects_per_slab,
00404         max_free_slabs   = Max_free,
00405     };
00406
00412     void *alloc() noexcept { return _a.alloc(); }
00413
00420     void free(void *p) noexcept { _a.free(p); }
00421
00430     unsigned total_objects() const noexcept { return _a.total_objects(); }
00431
00440     unsigned free_objects() const noexcept { return _a.free_objects(); }
00441 };
00442
00443
00444 template< int _O, int _S, int _M, template<typename A> class Alloc >
00445 typename Base_slab_static<_O, _S, _M, Alloc>::_A
00446     Base_slab_static<_O, _S, _M, Alloc>::_a;
00447
00463 template<typename Type, int Slab_size = L4_PAGESIZE,
00464         int Max_free = 2, template<typename A> class Alloc = New_allocator >
00465 class Slab_static
00466 : public Base_slab_static<sizeof(Type), Slab_size, Max_free, Alloc>
00467 {
00468 public:
00469
00470     typedef Type Obj_type;
00478     Type *alloc() noexcept
00479     {
00480         return reinterpret_cast<Type *>(
00481             Base_slab_static<sizeof(Type), Slab_size, Max_free, Alloc>::alloc());
00482     }
00483 };
00484
00485 }

```

16.229 slist

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2011 Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *     economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)

```

```

00007  */
00008
00009 #pragma once
00010
00011 #include "bits/list_basics.h"
00012
00013 namespace cxx {
00014
00015 class S_list_item
00016 {
00017 public:
00018     S_list_item() : _n(nullptr) {}
00019     // BSS allocation
00020     explicit S_list_item(bool) {}
00021
00022 private:
00023     template<typename T, typename P> friend class S_list;
00024     template<typename T, typename P> friend class S_list_tail;
00025     template<typename T, typename X> friend struct Bits::Basic_list_policy;
00026
00027     S_list_item(S_list_item const &);
00028     void operator = (S_list_item const &);
00029
00030     S_list_item *_n;
00031 };
00032
00033 template< typename T, typename POLICY = Bits::Basic_list_policy< T, S_list_item > >
00040 class S_list : public Bits::Basic_list<POLICY>
00041 {
00042     S_list(S_list const &) = delete;
00043     void operator = (S_list const &) = delete;
00044 private:
00045     typedef typename Bits::Basic_list<POLICY> Base;
00046 public:
00047     typedef typename Base::Iterator Iterator;
00048
00049     S_list(S_list &&o) : Base(static_cast<Base&&>(o)) {}
00050
00051     S_list &operator = (S_list &&o)
00052     {
00053         if (&o != this)
00054             Base::operator = (static_cast<Base&&>(o));
00055         return *this;
00056     }
00057
00058     // BSS allocation
00059     explicit S_list(bool x) : Base(x) {}
00060
00061     S_list() : Base() {}
00062
00063     void add(T *e)
00064     {
00065         e->_n = this->_f;
00066         this->_f = e;
00067     }
00068
00069     template< typename CAS >
00070     void add(T *e, CAS const &c)
00071     {
00072         do
00073         {
00074             e->_n = this->_f;
00075         }
00076         while (!c(&this->_f, e->_n, e));
00077     }
00078
00079     void push_front(T *e) { add(e); }
00080
00081     T *pop_front()
00082     {
00083         T *r = this->front();
00084         if (this->_f)
00085             this->_f = this->_f->_n;
00086         return r;
00087     }
00088
00089     void insert(T *e, Iterator const &pred)
00090     {
00091         S_list_item *p = *pred;
00092         e->_n = p->_n;
00093         p->_n = e;
00094     }
00095
00096     static void insert_before(T *e, Iterator const &succ)

```

```

00107 {
00108     S_list_item **x = Base::__get_internal(succ);
00109
00110     e->_n = *x;
00111     *x = e;
00112 }
00113
00114 static void replace(Iterator const &p, T*e)
00115 {
00116     S_list_item **x = Base::__get_internal(p);
00117     e->_n = (*x)->_n;
00118     *x = e;
00119 }
00120
00121 static Iterator erase(Iterator const &e)
00122 {
00123     S_list_item **x = Base::__get_internal(e);
00124     *x = (*x)->_n;
00125     return e;
00126 }
00127
00128 };
00129
00130
00131 template< typename T >
00132 class S_list_bss : public S_list<T>
00133 {
00134 public:
00135     S_list_bss() : S_list<T>(true) {}
00136 };
00137
00138 template< typename T, typename POLICY = Bits::Basic_list_policy< T, S_list_item > >
00139 class S_list_tail : public S_list<T, POLICY>
00140 {
00141 private:
00142     typedef S_list<T, POLICY> Base;
00143     void add(T *e) = delete;
00144
00145 public:
00146     using Iterator = typename Base::Iterator;
00147     S_list_tail() : Base(), _tail(&this->_f) {}
00148
00149     S_list_tail(S_list_tail &t)
00150     : Base(static_cast<Base&&>(t)), _tail(Base::empty() ? &this->_f : t._tail)
00151     {
00152         t._tail = &t._f; // reset the source tail
00153     }
00154
00155     S_list_tail &operator = (S_list_tail &t)
00156     {
00157         if (&t != this)
00158         {
00159             Base::operator = (static_cast<Base &&>(t));
00160             // The Basic_list operator= does not clear the tail, so we can use it
00161             _tail = Base::empty() ? &this->_f : t._tail;
00162             t._tail = &t._f; // reset the source tail
00163         }
00164
00165         return *this;
00166     }
00167
00168     void push_front(T *e)
00169     {
00170         if (Base::empty())
00171             _tail = &e->_n;
00172
00173         Base::push_front(e);
00174     }
00175
00176     void push_back(T *e)
00177     {
00178         e->_n = nullptr;
00179         *_tail = e;
00180         _tail = &e->_n;
00181     }
00182
00183     void clear()
00184     {
00185         Base::clear();
00186         _tail = &this->_f;
00187     }
00188
00189     void append(S_list_tail &o)
00190     {
00191         T *x = o.front();
00192         *_tail = x;
00193         if (x)

```

```

00194     _tail = o._tail;
00195     o.clear();
00196 }
00197
00198 T *pop_front()
00199 {
00200     T *t = Base::pop_front();
00201     if (t && Base::empty())
00202         _tail = &this->_f;
00203     return t;
00204 }
00205
00206 private:
00207     static void insert(T *e, Iterator const &pred);
00208     static void insert_before(T *e, Iterator const &succ);
00209     static void replace(Iterator const &p, T*e);
00210     static Iterator erase(Iterator const &e);
00211
00212 private:
00213     S_list_item **_tail;
00214 };
00215
00216 }

```

16.230 static_container

```

00001 // vi:set ft=c++ -- Mode: C++ --
00002 /*
00003  * Copyright (C) 2012-2013 Technische Universität Dresden.
00004  * Copyright (C) 2016-2017, 2020, 2023-2024 Kernkonzept GmbH.
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008
00009 #pragma once
00010
00011 #include <l4/cxx/type_traits>
00012 #include <stddef.h>
00013
00014 namespace cxx {
00015
00016 template< typename T >
00017 class Static_container
00018 {
00019 private:
00020     struct X : T
00021     {
00022         void *operator new (size_t, void *p) noexcept { return p; }
00023         void operator delete (void *) {}
00024         X() = default;
00025         template<typename ...Args>
00026         X(Args && ...a) : T(cxx::forward<Args>(a)...) {}
00027     };
00028
00029 public:
00030     void operator = (Static_container const &) = delete;
00031     Static_container(Static_container const &) = delete;
00032     Static_container() = default;
00033
00034     T *get() { return reinterpret_cast<X*>(_s); }
00035     T *operator -> () { return get(); }
00036     T &operator * () { return *get(); }
00037     operator T* () { return get(); }
00038
00039     void construct()
00040     { new (reinterpret_cast<void*>(_s)) X; }
00041
00042     template< typename ...Args >
00043     void construct(Args && ...args)
00044     { new (reinterpret_cast<void*>(_s)) X(cxx::forward<Args>(args)...) ; }
00045
00046 private:
00047     char _s[sizeof(X)] __attribute__((aligned(__alignof(X)))));
00048 };
00049
00050 }
00051
00052

```

16.231 static_vector

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002
00003 #pragma once
00004
00005 #include "type_traits"
00006
00007 namespace cxx {
00008
00015 template<typename T, typename IDX = unsigned>
00016 class static_vector
00017 {
00018 private:
00019     template<typename X, typename IDX2> friend class static_vector;
00020     T *_v;
00021     IDX _l = 0;
00022
00023 public:
00024     typedef T value_type;
00025     typedef IDX index_type;
00026
00027     static_vector() = default;
00028     static_vector(value_type *v, index_type length) : _v(v), _l(length) {}
00029
00030     template<typename Z,
00031             typename = enable_if_t<is_same<remove_extent_t<Z>, T>::value>
00032     constexpr static_vector(Z &v) : _v(v), _l(array_size(v))
00033     {}
00034
00036     template<typename X,
00037             typename = enable_if_t<is_convertible_v<X, T>>
00038     static_vector(static_vector<X, IDX> const &o) : _v(o._v), _l(o._l) {}
00039
00040     index_type size() const { return _l; }
00041     bool empty() const { return _l == 0; }
00042
00043     value_type &operator [] (index_type idx) { return _v[idx]; }
00044     value_type const &operator [] (index_type idx) const { return _v[idx]; }
00045
00046     value_type *begin() { return _v; }
00047     value_type *end() { return _v + _l; }
00048     value_type const *begin() const { return _v; }
00049     value_type const *end() const { return _v + _l; }
00050     value_type const *cbegin() const { return _v; }
00051     value_type const *cend() const { return _v + _l; }
00052
00054     index_type index(value_type const *o) const { return o - _v; }
00055     index_type index(value_type const &o) const { return &o - _v; }
00056 };
00057
00058 }

```

16.232 std_alloc

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009
00010 #pragma once
00011
00012 #include <stddef.h>
00013 namespace cxx {
00019 class Nothrow {};
00020 }
00021
00028 inline void *operator new (size_t, void *mem, cxx::Nothrow const &) noexcept
00029 { return mem; }
00030
00035 void *operator new (size_t, cxx::Nothrow const &) noexcept;
00036
00042 void operator delete (void *, cxx::Nothrow const &) noexcept;
00043
00044 namespace cxx {
00045
00046
00055 template< typename _Type >

```



```

00056 class New_allocator
00057 {
00058 public:
00059     enum { can_free = true };
00060
00061     New_allocator() noexcept {}
00062     New_allocator(New_allocator const &) noexcept {}
00063
00064     ~New_allocator() noexcept {}
00065
00066     _Type *alloc() noexcept
00067     { return static_cast<_Type*> (::operator new(sizeof (_Type), cxx::Nothrow())); }
00068
00069     void free(_Type *t) noexcept
00070     { ::operator delete(t, cxx::Nothrow()); }
00071 };
00072
00073 }
00074

```

16.233 std_ops

```

00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00004  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009
00010 #pragma once
00011
00012 namespace cxx {
00013
00014 template< typename Obj >
00015 struct Lt_functor
00016 {
00017     bool operator () (Obj const &l, Obj const &r) const
00018     { return l < r; }
00019 };
00020
00021 };
00022
00023
00024 };
00025

```

16.234 string

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008
00009 #pragma once
00010
00011 #include <limits>
00012 #include <string>
00013 #include <string_view>
00014
00015 namespace cxx {
00016
00017 class String
00018 {
00019 public:
00020     typedef char const *Index;
00021
00022     String(char const *s) noexcept : _start(s), _len(strlen(s)) {}
00023     String(char const *s, unsigned long len) noexcept : _start(s), _len(len) {}
00024
00025     String(char const *s, char const *e) noexcept : _start(s), _len(e - s) {}
00026
00027     String() : _start(0), _len(0) {}
00028
00029     Index start() const { return _start; }
00030     Index end() const { return _start + _len; }
00031

```

```

00058 int len() const { return _len; }
00059
00061 void start(char const *s) { _start = s; }
00063 void len(unsigned long len) { _len = len; }
00065 bool empty() const { return !_len; }
00066
00068 String head(Index end) const
00069 {
00070     if (end < _start)
00071         return String();
00072
00073     if (eof(end))
00074         return *this;
00075
00076     return String(_start, end - _start);
00077 }
00078
00080 String head(unsigned long end) const
00081 { return head(start() + end); }
00082
00084 String substr(unsigned long idx, unsigned long len = ~0UL) const
00085 {
00086     if (idx >= _len)
00087         return String(end(), 0UL);
00088
00089     return String(_start + idx, cxx::min(len, _len - idx));
00090 }
00091
00093 String substr(char const *start, unsigned long len = 0) const
00094 {
00095     if (start >= _start && !eof(start))
00096     {
00097         unsigned long nlen = _start + _len - start;
00098         if (len != 0)
00099             nlen = cxx::min(nlen, len);
00100         return String(start, nlen);
00101     }
00102
00103     return String(end(), 0UL);
00104 }
00105
00107 template< typename F >
00108 char const *find_match(F &&match) const
00109 {
00110     String::Index s = _start;
00111     while (1)
00112     {
00113         if (eof(s))
00114             return s;
00115
00116         if (match(*s))
00117             return s;
00118
00119         ++s;
00120     }
00121 }
00122
00124 char const *find(char const *c) const
00125 { return find(c, start()); }
00126
00128 char const *find(int c) const
00129 { return find(c, start()); }
00130
00132 char const *rfind(char const *c) const
00133 {
00134     if (!_len)
00135         return end();
00136
00137     char const *p = end();
00138     --p;
00139     while (p >= _start)
00140     {
00141         if (*p == *c)
00142             return p;
00143         --p;
00144     }
00145     return end();
00146 }
00147
00148
00155 Index starts_with(cxx::String const &c) const
00156 {
00157     unsigned long i;
00158     for (i = 0; i < c._len && i < _len; ++i)
00159         if (_start[i] != c[i])
00160             return 0;
00161     return i == c._len ? start() + i : 0;

```

```

00162     }
00163
00165 char const *find(int c, char const *s) const
00166 {
00167     if (s < _start)
00168         return end();
00169
00170     while (1)
00171     {
00172         if (eof(s))
00173             return s;
00174
00175         if (*s == c)
00176             return s;
00177
00178         ++s;
00179     }
00180 }
00181
00191 char const *find(char const *c, char const *s) const
00192 {
00193     if (s < _start)
00194         return end();
00195
00196     while (1)
00197     {
00198         if (eof(s))
00199             return s;
00200
00201         for (char const *x = c; *x; ++x)
00202             if (*s == *x)
00203                 return s;
00204
00205         ++s;
00206     }
00207 }
00208
00210 char const &operator [] (unsigned long idx) const { return _start[idx]; }
00212 char const &operator [] (int idx) const { return _start[idx]; }
00214 char const &operator [] (Index idx) const { return *idx; }
00215
00217 bool eof(char const *s) const { return s >= _start + _len || !*s; }
00218
00227 template<typename INT>
00228 int from_dec(INT *v) const
00229 {
00230     *v = 0;
00231     Index c;
00232     for (c = start(); !eof(c); ++c)
00233     {
00234         unsigned char n;
00235         if (*c >= '0' && *c <= '9')
00236             n = *c - '0';
00237         else
00238             return c - start();
00239
00240         *v *= 10;
00241         *v += n;
00242     }
00243     return c - start();
00244 }
00245
00256 template<typename INT>
00257 int from_hex(INT *v) const
00258 {
00259     *v = 0;
00260     unsigned shift = 0;
00261     Index c;
00262     for (c = start(); !eof(c); ++c)
00263     {
00264         shift += 4;
00265         if (shift > sizeof(INT) * 8)
00266             return -1;
00267         unsigned char n;
00268         if (*c >= '0' && *c <= '9')
00269             n = *c - '0';
00270         else if (*c >= 'A' && *c <= 'F')
00271             n = *c - 'A' + 10;
00272         else if (*c >= 'a' && *c <= 'f')
00273             n = *c - 'a' + 10;
00274         else
00275             return c - start();
00276
00277         *v <<= 4;
00278         *v |= n;
00279     }
00280     return c - start();

```


Data Structures

- class [L4::String](#)
A null-terminated string container class.

Namespaces

- namespace [L4](#)
[L4](#) low-level kernel interface.

16.235.1 Detailed Description

String.

Definition in file [string.h](#).

16.236 string.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2004-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00007  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #pragma once
00013
00014 #include <l4/cxx/basic_ostream>
00015
00016 namespace L4 {
00017
00022     class String
00023     {
00024     public:
00025         String(char const *str = "") : _str(str)
00026         {}
00027
00028         unsigned length() const
00029         {
00030             unsigned l;
00031             for (l = 0; _str[l]; l++)
00032                 ;
00033             return l;
00034         }
00035
00036         char const *p_str() const { return _str; }
00037
00038     private:
00039         char const *_str;
00040     };
00041 }
00042
00043 inline
00044 L4::BasicOStream &operator << (L4::BasicOStream &o, L4::String const &s)
00045 {
00046     o << s.p_str();
00047     return o;
00048 }

```

16.237 type_list

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 #pragma once
00003
00004 /*
00005  * (c) 2012 Alexander Warg <warg@os.inf.tu-dresden.de>,
00006  *     economic rights: Technische Universität Dresden (Germany)
00007  *
00008  * License: see LICENSE.spdx (in this directory or the directories above)
00009  */
00010
00011
00012 #include "type_traits"
00013
00014 namespace cxx {
00015
00016 template< typename ...T >
00017 struct type_list;
00018
00019 template<>
00020 struct type_list<>
00021 {
00022     typedef false_type head;
00023     typedef false_type tail;
00024 };
00025
00026 template<typename HEAD, typename ...TAIL>
00027 struct type_list<HEAD, TAIL...>
00028 {
00029     typedef HEAD head;
00030     typedef type_list<TAIL...> tail;
00031 };
00032
00033 template<typename TYPELIST, template <typename T> class PREDICATE>
00034 struct find_type;
00035
00036 template<template <typename T> class PREDICATE>
00037 struct find_type<type_list<>, PREDICATE>
00038 {
00039     typedef false_type type;
00040 };
00041
00042 template<typename TYPELIST, template <typename T> class PREDICATE>
00043 struct find_type
00044 {
00045     typedef typename conditional<PREDICATE<typename TYPELIST::head>::value,
00046                               typename TYPELIST::head,
00047                               typename find_type<typename TYPELIST::tail, PREDICATE>::type>::type
00048     type;
00049 };
00050
00051 template<typename TYPELIST, template <typename T> class PREDICATE>
00052 using find_type_t = typename find_type<TYPELIST, PREDICATE>::type;
00053 }

```

16.238 type_traits

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002
00003 /*
00004  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00005  *     Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00006  *     economic rights: Technische Universität Dresden (Germany)
00007  *
00008  * License: see LICENSE.spdx (in this directory or the directories above)
00009  */
00010
00011
00012 #pragma once
00013
00014 #pragma GCC system_header
00015
00016 #include <14/sys/compiler.h>
00017 #include "bits/type_traits.h"
00018
00019 namespace cxx {
00020
00021 template< typename T, T V >
00022 struct integral_constant
00023 {

```

```

00024     static T const value = V;
00025     typedef T value_type;
00026     typedef integral_constant<T, V> type;
00027 };
00028
00029 typedef integral_constant<bool, true> true_type;
00030 typedef integral_constant<bool, false> false_type;
00031
00032 template<typename...>
00033 using void_t = void;
00034
00035 template< typename T > struct remove_reference;
00036
00037 template< typename T > struct identity { typedef T type; };
00038 template< typename T > using identity_t = typename identity<T>::type;
00039
00040 template< typename T1, typename T2 > struct is_same;
00041
00042 template< typename T > struct remove_const;
00043
00044 template< typename T > struct remove_volatile;
00045
00046 template< typename T > struct remove_cv;
00047
00048 template< typename T > struct remove_pointer;
00049
00050 template< typename T > struct remove_extent;
00051
00052 template< typename T > struct remove_all_extents;
00053
00054
00055
00056 template< typename, typename >
00057 struct is_same : false_type {};
00058
00059 template< typename T >
00060 struct is_same<T, T> : true_type {};
00061
00062 template< typename T1, typename T2 >
00063 inline constexpr bool is_same_v = is_same<T1, T2>::value;
00064
00065 template< typename T >
00066 struct remove_reference { typedef T type; };
00067
00068 template< typename T >
00069 struct remove_reference<T &> { typedef T type; };
00070
00071 template< typename T >
00072 struct remove_reference<T &&> { typedef T type; };
00073
00074 template< typename T >
00075 using remove_reference_t = typename remove_reference<T>::type;
00076
00077 template< typename T > struct remove_const { typedef T type; };
00078 template< typename T > struct remove_const<T const> { typedef T type; };
00079 template< typename T > using remove_const_t = typename remove_const<T>::type;
00080
00081 template< typename T > struct remove_volatile { typedef T type; };
00082 template< typename T > struct remove_volatile<T volatile> { typedef T type; };
00083 template< typename T > using remove_volatile_t = typename remove_volatile<T>::type;
00084
00085 template< typename T >
00086 struct remove_cv { typedef remove_const_t<remove_volatile_t<T> type; };
00087
00088 template< typename T >
00089 using remove_cv_t = typename remove_cv<T>::type;
00090
00091 template<class T>
00092 struct remove_cvref { using type = remove_cv_t<remove_reference_t<T> }; };
00093
00094 template< typename T >
00095 using remove_cvref_t = typename remove_cvref<T>::type;
00096
00097 template< typename T, typename >
00098 struct __remove_pointer_h { typedef T type; };
00099
00100 template< typename T, typename I >
00101 struct __remove_pointer_h<T, I*> { typedef I type; };
00102
00103 template< typename T >
00104 struct remove_pointer : __remove_pointer_h<T, remove_cv_t<T> {};
00105
00106 template< typename T >
00107 using remove_pointer_t = typename remove_pointer<T>::type;
00108
00109
00110 template< typename T >

```

```

00111 struct remove_extent { typedef T type; };
00112
00113 template< typename T >
00114 struct remove_extent<T[]> { typedef T type; };
00115
00116 template< typename T, unsigned long N >
00117 struct remove_extent<T[N]> { typedef T type; };
00118
00119 template< typename T >
00120 using remove_extent_t = typename remove_extent<T>::type;
00121
00122
00123 template< typename T >
00124 struct remove_all_extents { typedef T type; };
00125
00126 template< typename T >
00127 struct remove_all_extents<T[]> { typedef typename remove_all_extents<T>::type type; };
00128
00129 template< typename T, unsigned long N >
00130 struct remove_all_extents<T[N]> { typedef typename remove_all_extents<T>::type type; };
00131
00132 template< typename T >
00133 using remove_all_extents_t = typename remove_all_extents<T>::type;
00134
00135 template< typename T >
00136 constexpr T &&
00137 forward(cxx::remove_reference_t<T> &t)
00138 { return static_cast<T &&>(t); }
00139
00140 template< typename T >
00141 constexpr T &&
00142 forward(cxx::remove_reference_t<T> &&t)
00143 { return static_cast<T &&>(t); }
00144
00145 template< typename T >
00146 constexpr cxx::remove_reference_t<T> &&
00147 move(T &&t) { return static_cast<cxx::remove_reference_t<T> &&>(t); }
00148
00149 template< bool, typename T = void >
00150 struct enable_if {};
00151
00152 template< typename T >
00153 struct enable_if<true, T> { typedef T type; };
00154
00155 template< bool C, typename T = void >
00156 using enable_if_t = typename enable_if<C, T>::type;
00157
00158 template< typename T >
00159 struct is_const : false_type {};
00160
00161 template< typename T >
00162 struct is_const<T const> : true_type {};
00163
00164 template< typename T >
00165 inline constexpr bool is_const_v = is_const<T>::value;
00166
00167 template< typename T >
00168 struct is_volatile : false_type {};
00169
00170 template< typename T >
00171 struct is_volatile<T volatile> : true_type {};
00172
00173 template< typename T >
00174 inline constexpr bool is_volatile_v = is_volatile<T>::value;
00175
00176 template< typename T >
00177 struct is_pointer : false_type {};
00178
00179 template< typename T >
00180 struct is_pointer<T *> : true_type {};
00181
00182 template< typename T >
00183 inline constexpr bool is_pointer_v = is_pointer<T>::value;
00184
00185 template<class T>
00186 inline constexpr bool is_null_pointer_v = is_same_v<decltype(nullptr), remove_cv_t<T>;
00187
00188 template< typename T >
00189 struct is_reference : false_type {};
00190
00191 template< typename T >
00192 struct is_reference<T &> : true_type {};
00193
00194 template< typename T >
00195 struct is_reference<T &&> : true_type {};
00196
00197 template< typename T >

```



```

00198 inline constexpr bool is_reference_v = is_reference<T>::value;
00199
00200 template< bool, typename, typename >
00201 struct conditional;
00202
00203 template< bool C, typename T_TRUE, typename T_FALSE >
00204 struct conditional { typedef T_TRUE type; };
00205
00206 template< typename T_TRUE, typename T_FALSE >
00207 struct conditional< false, T_TRUE, T_FALSE > { typedef T_FALSE type; };
00208
00209 template< bool C, typename T_TRUE, typename T_FALSE >
00210 using conditional_t = typename conditional<C, T_TRUE, T_FALSE>::type;
00211
00212 template<typename T>
00213 struct is_enum : integral_constant<bool, __is_enum(T)> {};
00214
00215 template< typename T >
00216 inline constexpr bool is_enum_v = is_enum<T>::value;
00217
00218 template<typename T>
00219 struct is_polymorphic : cxx::integral_constant<bool, __is_polymorphic(T)> {};
00220
00221 template< typename T > struct is_integral : false_type {};
00222
00223 template<> struct is_integral<bool> : true_type {};
00224
00225 template<> struct is_integral<char> : true_type {};
00226 template<> struct is_integral<signed char> : true_type {};
00227 template<> struct is_integral<unsigned char> : true_type {};
00228 template<> struct is_integral<short> : true_type {};
00229 template<> struct is_integral<unsigned short> : true_type {};
00230 template<> struct is_integral<int> : true_type {};
00231 template<> struct is_integral<unsigned int> : true_type {};
00232 template<> struct is_integral<long> : true_type {};
00233 template<> struct is_integral<unsigned long> : true_type {};
00234 template<> struct is_integral<long long> : true_type {};
00235 template<> struct is_integral<unsigned long long> : true_type {};
00236
00237 template< typename T >
00238 inline constexpr bool is_integral_v = is_integral<T>::value;
00239
00240 template< typename T, bool = is_integral_v<T> || is_enum_v<T> >
00241 struct __is_signed_helper : integral_constant<bool, static_cast<bool>(T(-1) < T(0))> {};
00242
00243 template< typename T >
00244 struct __is_signed_helper<T, false> : integral_constant<bool, false> {};
00245
00246 template< typename T >
00247 struct is_signed : __is_signed_helper<T> {};
00248
00249 template< typename T >
00250 inline constexpr bool is_signed_v = is_signed<T>::value;
00251
00252
00253 template< typename >
00254 struct is_array : false_type {};
00255
00256 template< typename T >
00257 struct is_array<T[]> : true_type {};
00258
00259 template< typename T, unsigned long N >
00260 struct is_array<T[N]> : true_type {};
00261
00262 template< typename T >
00263 inline constexpr bool is_array_v = is_array<T>::value;
00264
00265 template< typename T, unsigned N >
00266 constexpr unsigned array_size(T const (&)[N]) { return N; }
00267
00268 template< int SIZE, bool SIGN = false, bool = true > struct int_type_for_size;
00269
00270 template<> struct int_type_for_size<sizeof(char), true, true>
00271 { typedef signed char type; };
00272
00273 template<> struct int_type_for_size<sizeof(char), false, true>
00274 { typedef unsigned char type; };
00275
00276 template<> struct int_type_for_size<sizeof(short), true, (sizeof(short) > sizeof(char))>
00277 { typedef short type; };
00278
00279 template<> struct int_type_for_size<sizeof(short), false, (sizeof(short) > sizeof(char))>
00280 { typedef unsigned short type; };
00281
00282 template<> struct int_type_for_size<sizeof(int), true, (sizeof(int) > sizeof(short))>
00283 { typedef int type; };
00284

```

```

00285 template<> struct int_type_for_size<sizeof(int), false, (sizeof(int) > sizeof(short))>
00286 { typedef unsigned int type; };
00287
00288 template<> struct int_type_for_size<sizeof(long), true, (sizeof(long) > sizeof(int))>
00289 { typedef long type; };
00290
00291 template<> struct int_type_for_size<sizeof(long), false, (sizeof(long) > sizeof(int))>
00292 { typedef unsigned long type; };
00293
00294 template<> struct int_type_for_size<sizeof(long long), true, (sizeof(long long) > sizeof(long))>
00295 { typedef long long type; };
00296
00297 template<> struct int_type_for_size<sizeof(long long), false, (sizeof(long long) > sizeof(long))>
00298 { typedef unsigned long long type; };
00299
00300 template< int SIZE, bool SIGN = false>
00301 using int_type_for_size_t = typename int_type_for_size<SIZE, SIGN>::type;
00302
00303 template< typename T, class Enable = void > struct underlying_type {};
00304
00305 template< typename T >
00306 struct underlying_type<T, typename enable_if<is_enum_v<T>::type >
00307 {
00308     typedef int_type_for_size_t<sizeof(T), is_signed_v<T> type;
00309 };
00310
00311 template< typename T >
00312 using underlying_type_t = typename underlying_type<T>::type;
00313
00314 template< typename T > struct make_signed;
00315 template<> struct make_signed<char> { typedef signed char type; };
00316 template<> struct make_signed<unsigned char> { typedef signed char type; };
00317 template<> struct make_signed<signed char> { typedef signed char type; };
00318 template<> struct make_signed<unsigned int> { typedef signed int type; };
00319 template<> struct make_signed<signed int> { typedef signed int type; };
00320 template<> struct make_signed<unsigned long int> { typedef signed long int type; };
00321 template<> struct make_signed<signed long int> { typedef signed long int type; };
00322 template<> struct make_signed<unsigned long long int> { typedef signed long long int type; };
00323 template<> struct make_signed<signed long long int> { typedef signed long long int type; };
00324 template< typename T > using make_signed_t = typename make_signed<T>::type;
00325
00326 template< typename T > struct make_unsigned;
00327 template<> struct make_unsigned<char> { typedef unsigned char type; };
00328 template<> struct make_unsigned<unsigned char> { typedef unsigned char type; };
00329 template<> struct make_unsigned<signed char> { typedef unsigned char type; };
00330 template<> struct make_unsigned<unsigned int> { typedef unsigned int type; };
00331 template<> struct make_unsigned<signed int> { typedef unsigned int type; };
00332 template<> struct make_unsigned<unsigned long int> { typedef unsigned long int type; };
00333 template<> struct make_unsigned<signed long int> { typedef unsigned long int type; };
00334 template<> struct make_unsigned<unsigned long long int> { typedef unsigned long long int type; };
00335 template<> struct make_unsigned<signed long long int> { typedef unsigned long long int type; };
00336 template< typename T > using make_unsigned_t = typename make_unsigned<T>::type;
00337
00338
00339 template<typename From, typename To>
00340 struct is_convertible
00341 {
00342 private:
00343     struct _true { char x[2]; };
00344     struct _false {};
00345
00346     static _true _helper(To const *);
00347     static _false _helper(...);
00348 public:
00349     enum
00350     {
00351         value = sizeof(_true) == sizeof(_helper(static_cast<From*>(0)))
00352             ? true : false
00353     };
00354
00355     typedef bool value_type;
00356 };
00357
00358 template<typename From, typename To>
00359 inline constexpr bool is_convertible_v = is_convertible<From, To>::value;
00360
00361 template< typename T >
00362 struct is_empty : integral_constant<bool, __is_empty(T)> {};
00363
00364 template< typename T >
00365 inline constexpr bool is_empty_v = is_empty<T>::value;
00366
00367
00368 #if L4_HAS_BUILTIN(__is_function)
00369     template < typename T >
00370     struct is_function : integral_constant<bool, __is_function(T)> {};
00371 #else

```

```

00372     template < typename T >
00373     struct is_function : integral_constant<bool,      !is_reference_v<T>
00374                                     && !is_const_v<const T> {}> {};
00375 #endif
00376
00377 template< typename T >
00378 inline constexpr bool is_function_v = is_function<T>::value;
00379
00380 }
00381

```

16.239 unique_ptr

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * Copyright (C) 2013      Technische Universität Dresden.
00004  * Copyright (C) 2014-2017, 2020, 2023-2024 Kernkonzept GmbH.
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008
00009 #pragma once
00010
00011 #include "type_traits"
00012
00013 namespace cxx
00014 {
00015
00016     template< typename T >
00017     struct default_delete
00018     {
00019         default_delete() {}
00020
00021         template< typename U >
00022         default_delete(default_delete<U> const &) {}
00023
00024         void operator () (T *p) const
00025         { delete p; }
00026     };
00027
00028     template< typename T >
00029     struct default_delete<T[]>
00030     {
00031         default_delete() {}
00032
00033         void operator () (T *p)
00034         { delete [] p; }
00035     };
00036
00037     template< typename T, typename C >
00038     struct unique_ptr_index_op {};
00039
00040     template< typename T, typename C >
00041     struct unique_ptr_index_op<T[], C>
00042     {
00043         typedef T &reference;
00044         reference operator [] (int idx) const
00045         { return static_cast<C const *>(this)->get()[idx]; }
00046     };
00047
00048     template< typename T, typename T_Del = default_delete<T> >
00049     class unique_ptr : public unique_ptr_index_op<T, unique_ptr<T, T_Del> >
00050     {
00051     private:
00052         struct _unspec;
00053         typedef _unspec* _unspec_ptr_type;
00054
00055     public:
00056         typedef cxx::remove_extent_t<T> element_type;
00057         typedef element_type *pointer;
00058         typedef element_type &reference;
00059         typedef T_Del deleter_type;
00060
00061         unique_ptr() : _ptr(pointer()) {}
00062
00063         explicit unique_ptr(pointer p) : _ptr(p) {}
00064
00065         unique_ptr(unique_ptr &&o) : _ptr(o.release()) {}
00066
00067         ~unique_ptr() { reset(); }
00068
00069         unique_ptr &operator = (unique_ptr &&o)
00070         {

```

```

00071     reset(o.release());
00072     return *this;
00073 }
00074
00075 unique_ptr &operator = (_unspec_ptr_type)
00076 {
00077     reset();
00078     return *this;
00079 }
00080
00081 element_type &operator * () const { return *get(); }
00082 pointer operator -> () const { return get(); }
00083
00084 pointer get() const { return _ptr; }
00085
00086 operator _unspec_ptr_type () const
00087 { return reinterpret_cast<_unspec_ptr_type>(get()); }
00088
00089 pointer release()
00090 {
00091     pointer r = _ptr;
00092     _ptr = 0;
00093     return r;
00094 }
00095
00096 void reset(pointer p = pointer())
00097 {
00098     if (p != get())
00099     {
00100         deleter_type()(get());
00101         _ptr = p;
00102     }
00103 }
00104
00105 unique_ptr(unique_ptr const &) = delete;
00106 unique_ptr &operator = (unique_ptr const &) = delete;
00107
00108 private:
00109     pointer _ptr;
00110 };
00111
00112 template< typename T >
00113 unique_ptr<T>
00114 make_unique_ptr(T *p)
00115 { return unique_ptr<T>(p); }
00116
00117 template< typename T >
00118 cxx::enable_if_t< cxx::is_array<T>::value, unique_ptr<T> >
00119 make_unique(unsigned long size)
00120 { return cxx::unique_ptr<T>(new cxx::remove_extent_t<T>[size]()); }
00121
00122 template< typename T, typename... Args >
00123 cxx::enable_if_t< !cxx::is_array<T>::value, unique_ptr<T> >
00124 make_unique(Args &&... args)
00125 { return cxx::unique_ptr<T>(new T(cxx::forward<Args>(args)...)); }
00126
00127 }

```

16.240 l4/cxx/unique_ptr_list File Reference

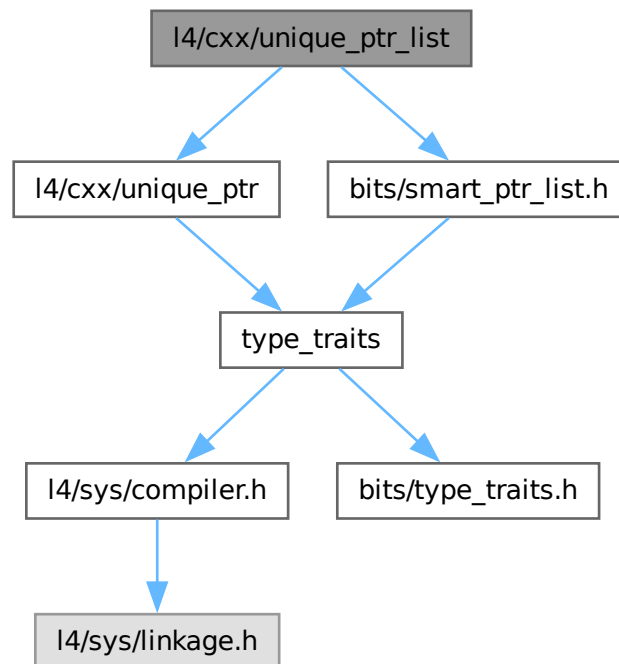
Implementation of a list of unique-ptr-managed objects.

```

#include <l4/cxx/unique_ptr>
#include "bits/smart_ptr_list.h"

```

Include dependency graph for `unique_ptr_list`:



Namespaces

- namespace `cxx`
Our C++ library.

Typedefs

- `template<typename T>`
using `cxx::Unique_ptr_list_item` = `Bits::Smart_ptr_list_item<T, cxx::unique_ptr<T> >`
Item for list linked with `cxx::unique_ptr`.
- `template<typename T>`
using `cxx::Unique_ptr_list` = `Bits::Smart_ptr_list<Unique_ptr_list_item<T> >`
Single-linked list where elements are connected with a `cxx::unique_ptr`.

16.240.1 Detailed Description

Implementation of a list of unique-ptr-managed objects.

Definition in file `unique_ptr_list`.

16.241 unique_ptr_list

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * Copyright (C) 2018-2019, 2022, 2024 Kernkonzept GmbH.
00004  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include <l4/cxx/unique_ptr>
00011
00012 #include "bits/smart_ptr_list.h"
00013
00014 namespace cxx {
00015
00016 template <typename T>
00017 using Unique_ptr_list_item = Bits::Smart_ptr_list_item<T, cxx::unique_ptr<T> >;
00018
00019 template <typename T>
00020 using Unique_ptr_list = Bits::Smart_ptr_list<Unique_ptr_list_item<T> >;
00021
00022 }
00023

```

16.242 utils

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * Copyright (C) 2013 Technische Universität Dresden.
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 namespace cxx {
00010
00011 template< typename T > inline
00012 T access_once(T const *a)
00013 {
00014     #if 1
00015         __asm__ __volatile__ ( "" : "=m"(*const_cast<T*>(a));
00016                                T tmp = *a;
00017         __asm__ __volatile__ ( "" : "=m"(*const_cast<T*>(a));
00018                                return tmp;
00019     #else
00020         return *static_cast<T const volatile *>(a);
00021     #endif
00022 }
00023
00024 template< typename T, typename VAL > inline
00025 void write_now(T *a, VAL &&val)
00026 {
00027     __asm__ __volatile__ ( "" : "=m"(*a);
00028                            *a = val;
00029     __asm__ __volatile__ ( "" : : "m"(*a));
00030 }
00031
00032 struct in_place_t { explicit in_place_t() = default; };
00033
00034 inline constexpr in_place_t in_place {};
00035
00036 }
00037

```

16.243 weak_ref

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * Copyright (C) 2015, 2017, 2024 Kernkonzept GmbH.
00004  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00005  *
00006  * Alexander Warg <alexander.warg@kernkonzept.com>
00007  */
00008

```

```

00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009  #pragma once
00010
00011  #include "hlist"
00012
00013  namespace cxx {
00014
00024  class Weak_ref_base : public H_list_item_t<Weak_ref_base>
00025  {
00026  protected:
00027      Weak_ref_base(void const *ptr = nullptr) : _obj(ptr) {}
00028      void reset_hard() { _obj = nullptr; }
00029      void const *_obj;
00030
00031  public:
00038      struct List : H_list_t<Weak_ref_base>
00039      {
00040          void reset()
00041          {
00042              while (!empty())
00043                  pop_front()->reset_hard();
00044          }
00045
00046          ~List()
00047          { reset(); }
00048      };
00049
00050      explicit operator bool () const
00051      { return _obj ? true : false; }
00052  };
00053
00054  template <typename T>
00095  class Weak_ref : public Weak_ref_base
00096  {
00097  public:
00098      T *get() const
00099      { return reinterpret_cast<T*>(const_cast<void *>(_obj)); }
00100
00101      T *reset(T *n)
00102      {
00103          T *r = get();
00104          if (r)
00105              r->remove_weak_ref(this);
00106
00107          _obj = n;
00108          if (n)
00109              n->add_weak_ref(this);
00110
00111          return r;
00112      }
00113
00114      Weak_ref(T *s = nullptr) : Weak_ref_base(s)
00115      {
00116          if (s)
00117              s->add_weak_ref(this);
00118      }
00119
00120      ~Weak_ref() { reset(0); }
00121
00122      void operator = (T *n)
00123      { reset(n); }
00124
00125      Weak_ref(Weak_ref const &o) : Weak_ref_base(o._obj)
00126      {
00127          if (T *x = get())
00128              x->add_weak_ref(this);
00129      }
00130
00131      Weak_ref &operator = (Weak_ref const &o)
00132      {
00133          if (&o == this)
00134              return *this;
00135
00136          reset(o.get());
00137          return *this;
00138      }
00139
00140      T &operator * () const { return get(); }
00141      T *operator -> () const { return get(); }
00142  };
00143
00144  class Weak_ref_obj
00145  {
00146  protected:
00147      template <typename T> friend class Weak_ref;

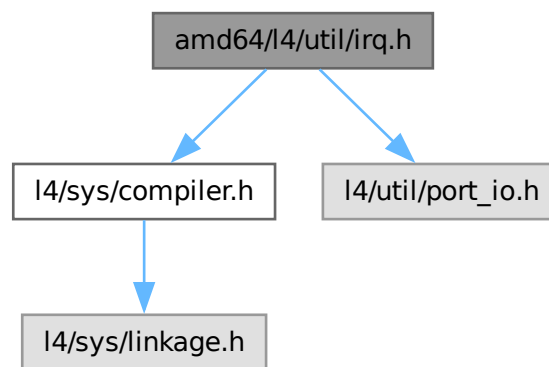
```

```
00148     mutable Weak_ref_base::List weak_references;
00149
00150     void add_weak_ref(Weak_ref_base *ref) const
00151     { weak_references.push_front(ref); }
00152
00153     void remove_weak_ref(Weak_ref_base *ref) const
00154     { weak_references.remove(ref); }
00155 };
00156
00157 }
```

16.244 amd64/l4/util/irq.h File Reference

some PIC and hardware interrupt related functions

```
#include <l4/sys/compiler.h>
#include <l4/util/port_io.h>
Include dependency graph for irq.h:
```



16.244.1 Detailed Description

some PIC and hardware interrupt related functions

Date

2003

Author

Jork Loeser jork.loeser@inf.tu-dresden.de Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file [irq.h](#).

16.245 irq.h

[Go to the documentation of this file.](#)

```

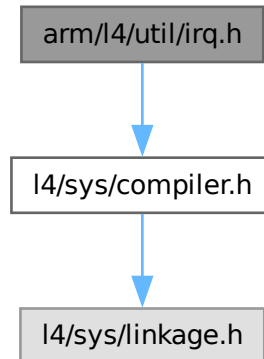
00001
00008
00009 /*
00010  * (c) 2003-2009 Author(s)
00011  *     economic rights: Technische Universität Dresden (Germany)
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014
00015 #ifndef __L4_IRQ_H__
00016 #define __L4_IRQ_H__
00017
00018 #include <l4/sys/compiler.h>
00019 #include <l4/util/port_io.h>
00020
00021 L4_BEGIN_DECLS
00022
00027
00030 static inline void
00031 l4util_cli (void)
00032 {
00033     __asm__ __volatile__ ("cli" : : : "memory");
00034 }
00035
00038 static inline void
00039 l4util_sti (void)
00040 {
00041     __asm__ __volatile__ ("sti" : : : "memory");
00042 }
00043
00047 static inline void
00048 l4util_flags_save (l4_umword_t *flags)
00049 {
00050     __asm__ __volatile__ ("pushf ; popq %0" : "=g" (*flags) : : "memory");
00051 }
00052
00055 static inline void
00056 l4util_flags_restore (l4_umword_t *flags)
00057 {
00058     __asm__ __volatile__ ("pushq %0 ; popf" : : "g" (*flags) : "memory");
00059 }
00061
00062 L4_END_DECLS
00063
00064 #endif

```

16.246 arm/l4/util/irq.h File Reference

ARM specific implementation of irq functions.

```
#include <l4/sys/compiler.h>
Include dependency graph for irq.h:
```



16.246.1 Detailed Description

ARM specific implementation of irq functions.

Do not use.

Definition in file [irq.h](#).

16.247 irq.h

[Go to the documentation of this file.](#)

```

00001
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *               Alexander Warg <warg@os.inf.tu-dresden.de>,
00010  *               Frank Mehnert <fm3@os.inf.tu-dresden.de>
00011  *               economic rights: Technische Universität Dresden (Germany)
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014 #ifndef __L4UTIL__ARCH_ARCH__IRQ_H__
00015 #define __L4UTIL__ARCH_ARCH__IRQ_H__
00016
00017 #ifdef __GNUC__
00018
00019 #include <l4/sys/compiler.h>
00020
00021 L4_BEGIN_DECLS
00022
00023 L4_INLINE void l4util_cli (void);
00024 L4_INLINE void l4util_sti (void);
00025 L4_INLINE void l4util_flags_save(l4_umword_t *flags);
00026 L4_INLINE void l4util_flags_restore(l4_umword_t *flags);
00027
00028 L4_INLINE
00029 void
00030 l4util_cli(void)
00031 {
00032     extern void __do_not_use_l4util_cli(void);
00033     __do_not_use_l4util_cli();
00034 }
00035

```

```

00036
00037 L4_INLINE
00038 void
00039 l4util_sti(void)
00040 {
00041     extern void __do_not_use_l4util_sti(void);
00042     __do_not_use_l4util_sti();
00043 }
00044
00045
00046 L4_INLINE
00047 void
00048 l4util_flags_save(l4_umword_t *flags)
00049 {
00050     (void)flags;
00051     extern void __do_not_use_l4util_flags_save(void);
00052     __do_not_use_l4util_flags_save();
00053 }
00054
00055 L4_INLINE
00056 void
00057 l4util_flags_restore(l4_umword_t *flags)
00058 {
00059     (void)flags;
00060     extern void __do_not_use_l4util_flags_restore(void);
00061     __do_not_use_l4util_flags_restore();
00062 }
00063
00064 L4_END_DECLS
00065
00066 #endif // __GNUC__
00067
00068 #endif /* ! __L4UTIL__ARCH_ARCH__IRQ_H__ */

```

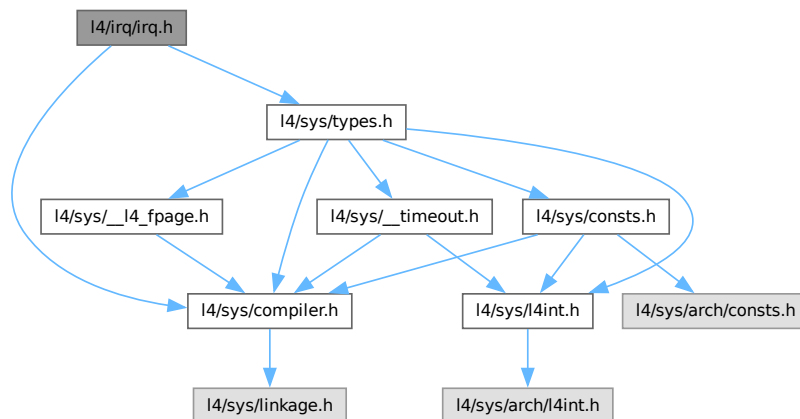
16.248 l4/irq/irq.h File Reference

IRQ handling routines.

```
#include <l4/sys/compiler.h>
```

```
#include <l4/sys/types.h>
```

Include dependency graph for irq.h:



Functions

- `l4irq_t * l4irq_attach` (int irqnum)
Attach/connect to IRQ.

- `l4irq_t * l4irq_attach_ft` (int irqnum, unsigned mode)
Attach/connect to IRQ using given type.
- `l4irq_t * l4irq_attach_thread` (int irqnum, `l4_cap_idx_t` to_thread)
Attach/connect to IRQ.
- `l4irq_t * l4irq_attach_thread_ft` (int irqnum, `l4_cap_idx_t` to_thread, unsigned mode)
Attach/connect to IRQ using given type.
- `long l4irq_wait` (`l4irq_t *irq`)
Wait for specified IRQ.
- `long l4irq_unmask_and_wait_any` (`l4irq_t *unmask_irq`, `l4irq_t **ret_irq`)
Unmask a specific IRQ and wait for any attached IRQ.
- `long l4irq_wait_any` (`l4irq_t **irq`)
Wait for any attached IRQ.
- `long l4irq_unmask` (`l4irq_t *irq`)
Unmask a specific IRQ.
- `long l4irq_detach` (`l4irq_t *irq`)
Detach from IRQ.
- `l4irq_t * l4irq_request` (int irqnum, void(*isr_handler)(void *), void *isr_data, int irq_thread_prio, unsigned mode)
Attach asynchronous ISR handler to IRQ.
- `long l4irq_release` (`l4irq_t *irq`)
Release asynchronous ISR handler and free resources.
- `l4irq_t * l4irq_attach_cap` (`l4_cap_idx_t` irqcap)
Attach/connect to IRQ.
- `l4irq_t * l4irq_attach_cap_ft` (`l4_cap_idx_t` irqcap, unsigned mode)
Attach/connect to IRQ using given type.
- `l4irq_t * l4irq_attach_thread_cap` (`l4_cap_idx_t` irqcap, `l4_cap_idx_t` to_thread)
Attach/connect to IRQ.
- `l4irq_t * l4irq_attach_thread_cap_ft` (`l4_cap_idx_t` irqcap, `l4_cap_idx_t` to_thread, unsigned mode)
Attach/connect to IRQ using given type.
- `l4irq_t * l4irq_request_cap` (`l4_cap_idx_t` irqcap, void(*isr_handler)(void *), void *isr_data, int irq_thread_prio, unsigned mode)
Attach asynchronous ISR handler to IRQ.

16.248.1 Detailed Description

IRQ handling routines.

Definition in file [irq.h](#).

16.249 irq.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Henning Schild <hschild@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * This file is part of TUD:OS and distributed under the terms of the
00011  * GNU General Public License 2.
00012  * Please see the COPYING-GPL-2 file for details.
00013  */
00014 #pragma once

```

```

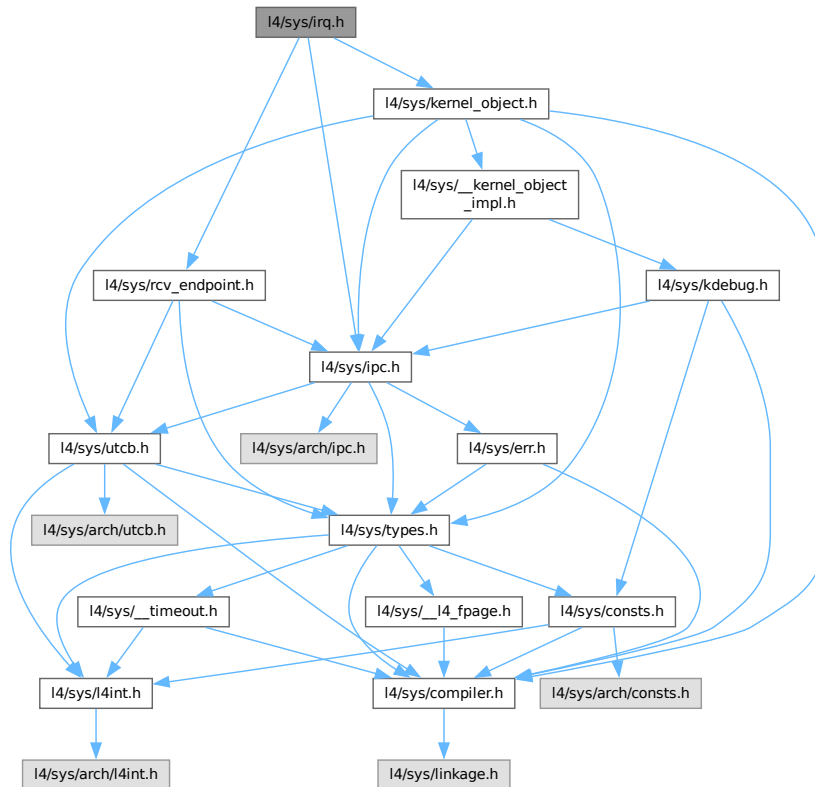
00015
00016 #include <l4/sys/compiler.h>
00017 #include <l4/sys/types.h>
00018
00019 __BEGIN_DECLS
00020
00027
00028 struct l4irq_t;
00029 typedef struct l4irq_t l4irq_t;
00030
00041 L4_CV l4irq_t *
00042 l4irq_attach(int irqnum);
00043
00055 L4_CV l4irq_t *
00056 l4irq_attach_ft(int irqnum, unsigned mode);
00057
00068 L4_CV l4irq_t *
00069 l4irq_attach_thread(int irqnum, l4_cap_idx_t to_thread);
00070
00082 L4_CV l4irq_t *
00083 l4irq_attach_thread_ft(int irqnum, l4_cap_idx_t to_thread,
00084                        unsigned mode);
00085
00093 L4_CV long
00094 l4irq_wait(l4irq_t *irq);
00095
00105 L4_CV long
00106 l4irq_unmask_and_wait_any(l4irq_t *unmask_irq, l4irq_t **ret_irq);
00107
00115 L4_CV long
00116 l4irq_wait_any(l4irq_t **irq);
00117
00128 L4_CV long
00129 l4irq_unmask(l4irq_t *irq);
00130
00138 L4_CV long
00139 l4irq_detach(l4irq_t *irq);
00140
00141
00142
00143 /*****
00152
00165 L4_CV l4irq_t *
00166 l4irq_request(int irqnum, void (*isr_handler)(void *), void *isr_data,
00167              int irq_thread_prio, unsigned mode);
00168
00176 L4_CV long
00177 l4irq_release(l4irq_t *irq);
00178
00179
00180
00181 /*****
00182 /*****
00183
00188
00199 L4_CV l4irq_t *
00200 l4irq_attach_cap(l4_cap_idx_t irqcap);
00201
00213 L4_CV l4irq_t *
00214 l4irq_attach_cap_ft(l4_cap_idx_t irqcap, unsigned mode);
00215
00226 L4_CV l4irq_t *
00227 l4irq_attach_thread_cap(l4_cap_idx_t irqcap, l4_cap_idx_t to_thread);
00228
00240 L4_CV l4irq_t *
00241 l4irq_attach_thread_cap_ft(l4_cap_idx_t irqcap, l4_cap_idx_t to_thread,
00242                             unsigned mode);
00243
00244 /*****
00252
00265 L4_CV l4irq_t *
00266 l4irq_request_cap(l4_cap_idx_t irqcap,
00267                  void (*isr_handler)(void *), void *isr_data,
00268                  int irq_thread_prio, unsigned mode);
00269
00270 __END_DECLS

```

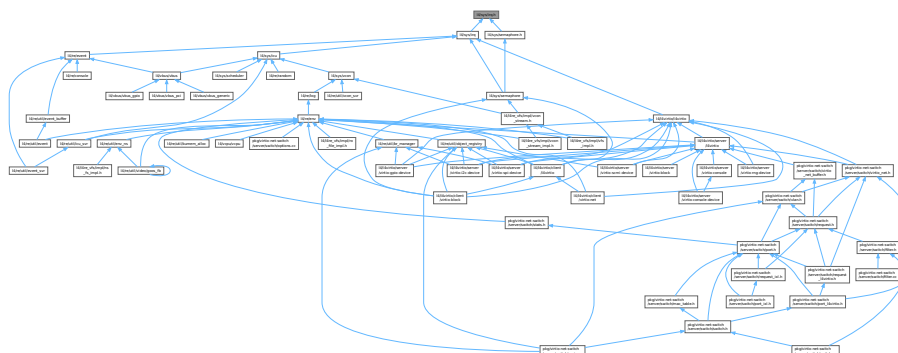
16.250 l4/sys/irq.h File Reference

C Irq interface.

```
#include <linux/sys/kernel_object.h>
#include <linux/sys/ipc.h>
#include <linux/sys/rcv_endpoint.h>
Include dependency graph for irq.h:
```



This graph shows which files directly or indirectly include this file:



Functions

- `l4_msgtag_t l4_irq_detach (l4_cap_idx_t irq) L4_NOTHROW`
Detach from an interrupt source.
- `l4_msgtag_t l4_irq_detach_u (l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW`

- Detach from this interrupt.*
- `l4_msgtag_t l4_irq_bind_vcpu (l4_cap_idx_t irq, l4_cap_idx_t thread, l4_umword_t cfg) L4_NOTHROW`
- Bind a thread to this Irq for vCPU interrupt forwarding.*
- `l4_msgtag_t l4_irq_bind_vcpu_u (l4_cap_idx_t irq, l4_cap_idx_t thread, l4_umword_t cfg, l4_utcb_t *utcb) L4_NOTHROW`
- Bind a thread to this Irq for vCPU interrupt forwarding.*
- `l4_msgtag_t l4_irq_trigger (l4_cap_idx_t irq) L4_NOTHROW`
- Trigger an IRQ.*
- `l4_msgtag_t l4_irq_trigger_u (l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW`
- Trigger the object.*
- `l4_msgtag_t l4_irq_receive (l4_cap_idx_t irq, l4_timeout_t to) L4_NOTHROW`
- Unmask and wait for specified IRQ.*
- `l4_msgtag_t l4_irq_receive_u (l4_cap_idx_t irq, l4_timeout_t timeout, l4_utcb_t *utcb) L4_NOTHROW`
- Unmask and wait for this IRQ.*
- `l4_msgtag_t l4_irq_wait (l4_cap_idx_t irq, l4_umword_t *label, l4_timeout_t to) L4_NOTHROW`
- Unmask IRQ and wait for any message.*
- `l4_msgtag_t l4_irq_wait_u (l4_cap_idx_t irq, l4_umword_t *label, l4_timeout_t timeout, l4_utcb_t *utcb) L4_NOTHROW`
- Unmask IRQ and (open) wait for any message.*
- `l4_msgtag_t l4_irq_unmask (l4_cap_idx_t irq) L4_NOTHROW`
- Unmask IRQ.*
- `l4_msgtag_t l4_irq_unmask_u (l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW`
- Unmask this IRQ.*

16.250.1 Detailed Description

C Irq interface.

Definition in file [irq.h](#).

16.251 irq.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *           Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *           Björn Döbel <doebel@os.inf.tu-dresden.de>
00010  *           economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014 #pragma once
00015
00016 #include <l4/sys/kernel_object.h>
00017 #include <l4/sys/ipc.h>
00018 #include <l4/sys/rcv_endpoint.h>
00019
00046
00062 L4_INLINE l4_msgtag_t
00063 l4_irq_detach(l4_cap_idx_t irq) L4_NOTHROW;
00064
00071 L4_INLINE l4_msgtag_t
00072 l4_irq_detach_u(l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW;
00073
00074
00109 L4_INLINE l4_msgtag_t
00110 l4_irq_bind_vcpu(l4_cap_idx_t irq, l4_cap_idx_t thread,
00111                 l4_umword_t cfg) L4_NOTHROW;

```

```

00112
00119 L4_INLINE l4_msgtag_t
00120 l4_irq_bind_vcpu_u(l4_cap_idx_t irq, l4_cap_idx_t thread, l4_umword_t cfg,
00121                  l4_utcb_t *utcb) L4_NOTHROW;
00122
00123
00139 L4_INLINE l4_msgtag_t
00140 l4_irq_trigger(l4_cap_idx_t irq) L4_NOTHROW;
00141
00148 L4_INLINE l4_msgtag_t
00149 l4_irq_trigger_u(l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW;
00150
00160 L4_INLINE l4_msgtag_t
00161 l4_irq_receive(l4_cap_idx_t irq, l4_timeout_t to) L4_NOTHROW;
00162
00169 L4_INLINE l4_msgtag_t
00170 l4_irq_receive_u(l4_cap_idx_t irq, l4_timeout_t timeout, l4_utcb_t *utcb) L4_NOTHROW;
00171
00182 L4_INLINE l4_msgtag_t
00183 l4_irq_wait(l4_cap_idx_t irq, l4_umword_t *label,
00184            l4_timeout_t to) L4_NOTHROW;
00185
00192 L4_INLINE l4_msgtag_t
00193 l4_irq_wait_u(l4_cap_idx_t irq, l4_umword_t *label,
00194              l4_timeout_t timeout, l4_utcb_t *utcb) L4_NOTHROW;
00195
00206 L4_INLINE l4_msgtag_t
00207 l4_irq_unmask(l4_cap_idx_t irq) L4_NOTHROW;
00208
00215 L4_INLINE l4_msgtag_t
00216 l4_irq_unmask_u(l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW;
00217
00221 enum l4_irq_sender_op
00222 {
00223     L4_IRQ_SENDER_OP_RESERVED1 = 0, // Ex ATTACH
00224     L4_IRQ_SENDER_OP_DETACH    = 1,
00225     L4_IRQ_SENDER_OP_BIND_VCPU = 2,
00226 };
00227
00231 enum l4_irq_op
00232 {
00233     L4_IRQ_OP_TRIGGER    = 2,
00234     L4_IRQ_OP_EOI       = 4
00235 };
00236
00237 /*****
00238  * Implementations
00239  */
00240
00241 L4_INLINE l4_msgtag_t
00242 l4_irq_detach_u(l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW
00243 {
00244     l4_utcb_mr_u(utcb)->mr[0] = L4_IRQ_SENDER_OP_DETACH;
00245     return l4_ipc_call(irq, utcb, l4_msgtag(L4_PROTO_IRQ_SENDER, 1, 0, 0),
00246                      L4_IPC_NEVER);
00247 }
00248
00249 L4_INLINE l4_msgtag_t
00250 l4_irq_bind_vcpu_u(l4_cap_idx_t irq, l4_cap_idx_t thread, l4_umword_t cfg,
00251                  l4_utcb_t *utcb) L4_NOTHROW
00252 {
00253     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00254     m->mr[0] = L4_IRQ_SENDER_OP_BIND_VCPU;
00255     m->mr[1] = cfg;
00256     m->mr[2] = l4_map_obj_control(0, 0);
00257     m->mr[3] = l4_obj_fpage(thread, 0, L4_CAP_FPAGE_RWS).raw;
00258     return l4_ipc_call(irq, utcb, l4_msgtag(L4_PROTO_IRQ_SENDER, 2, 1, 0),
00259                      L4_IPC_NEVER);
00260 }
00261
00262 L4_INLINE l4_msgtag_t
00263 l4_irq_trigger_u(l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW
00264 {
00265     return l4_ipc_send(irq, utcb, l4_msgtag(L4_PROTO_IRQ, 0, 0, 0),
00266                      L4_IPC_BOTH_TIMEOUT_0);
00267 }
00268
00269 L4_INLINE l4_msgtag_t
00270 l4_irq_receive_u(l4_cap_idx_t irq, l4_timeout_t to, l4_utcb_t *utcb) L4_NOTHROW
00271 {
00272     l4_utcb_mr_u(utcb)->mr[0] = L4_IRQ_OP_EOI;
00273     return l4_ipc_call(irq, utcb, l4_msgtag(L4_PROTO_IRQ, 1, 0, 0), to);
00274 }
00275
00276 L4_INLINE l4_msgtag_t
00277 l4_irq_wait_u(l4_cap_idx_t irq, l4_umword_t *label,
00278              l4_timeout_t to, l4_utcb_t *utcb) L4_NOTHROW

```



```

00279 {
00280     l4_utcb_mr_u(utcb)->mr[0] = L4_IRQ_OP_EOI;
00281     return l4_ipc_send_and_wait(irq, utcb, l4_msgtag(L4_PROTO_IRQ, 1, 0, 0),
00282                                label, to);
00283 }
00284
00285 L4_INLINE l4_msgtag_t
00286 l4_irq_unmask_u(l4_cap_idx_t irq, l4_utcb_t *utcb) L4_NOTHROW
00287 {
00288     l4_utcb_mr_u(utcb)->mr[0] = L4_IRQ_OP_EOI;
00289     return l4_ipc_send(irq, utcb, l4_msgtag(L4_PROTO_IRQ, 1, 0, 0), L4_IPC_NEVER);
00290 }
00291
00292
00293 L4_INLINE l4_msgtag_t
00294 l4_irq_detach(l4_cap_idx_t irq) L4_NOTHROW
00295 {
00296     return l4_irq_detach_u(irq, l4_utcb());
00297 }
00298
00299 L4_INLINE l4_msgtag_t
00300 l4_irq_bind_vcpu(l4_cap_idx_t irq, l4_cap_idx_t thread,
00301                 l4_umword_t cfg) L4_NOTHROW
00302 {
00303     return l4_irq_bind_vcpu_u(irq, thread, cfg, l4_utcb());
00304 }
00305
00306 L4_INLINE l4_msgtag_t
00307 l4_irq_trigger(l4_cap_idx_t irq) L4_NOTHROW
00308 {
00309     return l4_irq_trigger_u(irq, l4_utcb());
00310 }
00311
00312 L4_INLINE l4_msgtag_t
00313 l4_irq_receive(l4_cap_idx_t irq, l4_timeout_t to) L4_NOTHROW
00314 {
00315     return l4_irq_receive_u(irq, to, l4_utcb());
00316 }
00317
00318 L4_INLINE l4_msgtag_t
00319 l4_irq_wait(l4_cap_idx_t irq, l4_umword_t *label,
00320            l4_timeout_t to) L4_NOTHROW
00321 {
00322     return l4_irq_wait_u(irq, label, to, l4_utcb());
00323 }
00324
00325 L4_INLINE l4_msgtag_t
00326 l4_irq_unmask(l4_cap_idx_t irq) L4_NOTHROW
00327 {
00328     return l4_irq_unmask_u(irq, l4_utcb());
00329 }
00330

```

16.252 x86/I4/util/irq.h File Reference

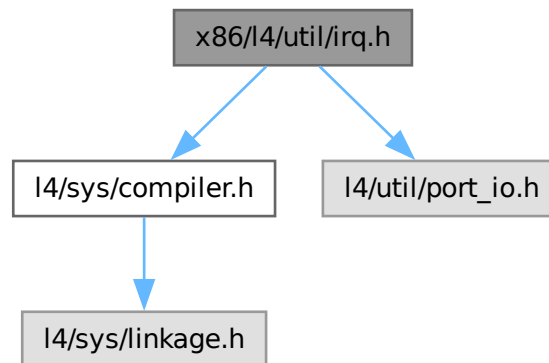
some PIC and hardware interrupt related functions

```

#include <l4/sys/compiler.h>
#include <l4/util/port_io.h>

```

Include dependency graph for irq.h:



16.252.1 Detailed Description

some PIC and hardware interrupt related functions

Date

2003

Author

Jork Loeser jork.loeser@inf.tu-dresden.de Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file [irq.h](#).

16.253 irq.h

[Go to the documentation of this file.](#)

```

00001
00008
00009 /*
00010  * (c) 2003-2009 Author(s)
00011  *     economic rights: Technische Universität Dresden (Germany)
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014
00015 #ifndef __L4_IRQ_H__
00016 #define __L4_IRQ_H__
00017
00018 #include <l4/sys/compiler.h>
00019 #include <l4/util/port_io.h>
00020
00021 L4_BEGIN_DECLS
00022
00027
00030 static inline void
00031 l4util_cli (void)
00032 {

```

```

00033  __asm__ __volatile__ ("cli" : : : "memory");
00034 }
00035
00038 static inline void
00039 l4util_sti (void)
00040 {
00041  __asm__ __volatile__ ("sti" : : : "memory");
00042 }
00043
00047 static inline void
00048 l4util_flags_save (l4_umword_t *flags)
00049 {
00050  __asm__ __volatile__ ("pushfl ; popl %0" : "=g" (*flags) : : "memory");
00051 }
00052
00055 static inline void
00056 l4util_flags_restore (l4_umword_t *flags)
00057 {
00058  __asm__ __volatile__ ("pushl %0 ; popfl" : : "g" (*flags) : "memory");
00059 }
00061
00062 L4_END_DECLS
00063
00064 #endif

```

16.254 backend

```

00001 // vi:set ft=c++: -- Mode: C++ --
00002 /*
00003  * (c) 2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009 #pragma once
00010
00011 #include <l4/l4re_vfs/vfs.h>
00012 #include <l4/crtn/initpriorities.h>
00013
00014 extern "C" void l4re_vfs_select_poll_notify(void);
00015
00016 namespace L4Re { namespace Vfs {
00017
00019 extern L4Re::Vfs::Ops *vfs_ops asm ("l4re_env_posix_vfs_ops");
00020
00021 class Mount_tree;
00022
00030 class Be_file : public File
00031 {
00032 public:
00033  void *operator new (size_t size) noexcept
00034  { return vfs_ops->malloc(size); }
00035
00036  void *operator new (size_t, void *m) noexcept
00037  { return m; }
00038
00039  void operator delete (void *m)
00040  { vfs_ops->free(m); }
00041
00042  // used in close, to unlock all locks of a file (as POSIX says)
00043  int unlock_all_locks() noexcept override
00044  { return 0; }
00045
00046  // for mmap
00047  L4::Cap<L4Re::Dataspace> data_space() noexcept override
00048  { return L4::Cap<L4Re::Dataspace>::Invalid; }
00049
00051 ssize_t readv(const struct iovec*, int) noexcept override
00052 { return -EINVAL; }
00053
00055 ssize_t writev(const struct iovec*, int) noexcept override
00056 { return -EINVAL; }
00057
00059 ssize_t pwritev(const struct iovec*, int, off64_t) noexcept override
00060 { return -EINVAL; }
00061
00063 ssize_t preadv(const struct iovec*, int, off64_t) noexcept override
00064 { return -EINVAL; }
00065
00067 off64_t lseek(off64_t, int) noexcept override
00068 { return -ESPIPE; }
00069

```

```

00071 int ftruncate(off64_t) noexcept override
00072 { return -EINVAL; }
00073
00075 int fsync() const noexcept override
00076 { return -EINVAL; }
00077
00079 int fdatsync() const noexcept override
00080 { return -EINVAL; }
00081
00083 int ioctl(unsigned long, va_list) noexcept override
00084 { return -EINVAL; }
00085
00086 int fstat(struct stat64 *) const noexcept override
00087 { return -EINVAL; }
00088
00090 int fchmod(mode_t) noexcept override
00091 { return -EINVAL; }
00092
00094 int get_status_flags() const noexcept override
00095 { return 0; }
00096
00098 int set_status_flags(long) noexcept override
00099 { return 0; }
00100
00102 int get_lock(struct flock64 *) noexcept override
00103 { return -ENOLCK; }
00104
00106 int set_lock(struct flock64 *, bool) noexcept override
00107 { return -ENOLCK; }
00108
00110 int faccessat(const char *, int, int) noexcept override
00111 { return -ENOTDIR; }
00112
00114 int fchmodat(const char *, mode_t, int) noexcept override
00115 { return -ENOTDIR; }
00116
00118 int utime(const struct utimbuf *) noexcept override
00119 { return -EROFS; }
00120
00122 int utimes(const struct timeval [2]) noexcept override
00123 { return -EROFS; }
00124
00126 int utimensat(const char *, const struct timespec [2], int) noexcept override
00127 { return -EROFS; }
00128
00130 int mkdir(const char *, mode_t) noexcept override
00131 { return -ENOTDIR; }
00132
00134 int unlink(const char *) noexcept override
00135 { return -ENOTDIR; }
00136
00138 int rename(const char *, const char *) noexcept override
00139 { return -ENOTDIR; }
00140
00142 int link(const char *, const char *) noexcept override
00143 { return -ENOTDIR; }
00144
00146 int symlink(const char *, const char *) noexcept override
00147 { return -EPERM; }
00148
00150 int rmdir(const char *) noexcept override
00151 { return -ENOTDIR; }
00152
00154 ssize_t readlink(char *, size_t) override
00155 { return -EINVAL; }
00156
00157 ssize_t getdents(char *, size_t) noexcept override
00158 { return -ENOTDIR; }
00159
00160
00161
00162 // Socket interface
00163 int bind(sockaddr const *, socklen_t) noexcept override
00164 { return -ENOTSOCK; }
00165
00166 int connect(sockaddr const *, socklen_t) noexcept override
00167 { return -ENOTSOCK; }
00168
00169 ssize_t send(void const *, size_t, int) noexcept override
00170 { return -ENOTSOCK; }
00171
00172 ssize_t recv(void *, size_t, int) noexcept override
00173 { return -ENOTSOCK; }
00174
00175 ssize_t sendto(void const *, size_t, int, sockaddr const *, socklen_t) noexcept
00176 override
00177 { return -ENOTSOCK; }

```

```

00178
00179 ssize_t recvfrom(void *, size_t, int, sockaddr *, socklen_t *) noexcept override
00180 { return -ENOTSOCK; }
00181
00182 ssize_t sendmsg(msghdr const *, int) noexcept override
00183 { return -ENOTSOCK; }
00184
00185 ssize_t recvmsg(msghdr *, int) noexcept override
00186 { return -ENOTSOCK; }
00187
00188 int getsockopt(int, int, void *, socklen_t *) noexcept override
00189 { return -ENOTSOCK; }
00190
00191 int setsockopt(int, int, void const *, socklen_t) noexcept override
00192 { return -ENOTSOCK; }
00193
00194 int listen(int) noexcept override
00195 { return -ENOTSOCK; }
00196
00197 int accept(sockaddr *, socklen_t *) noexcept override
00198 { return -ENOTSOCK; }
00199
00200 int shutdown(int) noexcept override
00201 { return -ENOTSOCK; }
00202
00203 int getsockname(sockaddr *, socklen_t *) noexcept override
00204 { return -ENOTSOCK; }
00205
00206 int getpeername(sockaddr *, socklen_t *) noexcept override
00207 { return -ENOTSOCK; }
00208
00217 bool check_ready(Ready_type) noexcept override
00218 { return false; }
00219
00220 ~Be_file() noexcept = 0;
00221
00222 private:
00224 int get_entry(const char *, int, mode_t, cxx::Ref_ptr<File> *) noexcept override
00225 { return -ENOTDIR; }
00226
00227 protected:
00228 const char *get_mount(const char *path, cxx::Ref_ptr<File> *dir) noexcept;
00229 };
00230
00231 inline
00232 Be_file::~Be_file() noexcept {}
00233
00234 class Be_file_pos : public Be_file
00235 {
00236 public:
00237 Be_file_pos() noexcept : Be_file(), _pos(0) {}
00238
00239 virtual off64_t size() const noexcept = 0;
00240
00241 ssize_t readv(const struct iovec *v, int iovcnt) noexcept override
00242 {
00243     ssize_t r = preadv(v, iovcnt, _pos);
00244     if (r > 0)
00245         _pos += r;
00246     return r;
00247 }
00248
00249 ssize_t writev(const struct iovec *v, int iovcnt) noexcept override
00250 {
00251     ssize_t r = pwritev(v, iovcnt, _pos);
00252     if (r > 0)
00253         _pos += r;
00254     return r;
00255 }
00256
00257 ssize_t preadv(const struct iovec *v, int iovcnt, off64_t offset) noexcept override = 0;
00258 ssize_t pwritev(const struct iovec *v, int iovcnt, off64_t offset) noexcept override = 0;
00259
00260 off64_t lseek(off64_t offset, int whence) noexcept override
00261 {
00262     off64_t r;
00263     switch (whence)
00264     {
00265     case SEEK_SET: r = offset; break;
00266     case SEEK_CUR: r = _pos + offset; break;
00267     case SEEK_END: r = size() + offset; break;
00268     default: return -EINVAL;
00269     };
00270
00271     if (r < 0)
00272         return -EINVAL;
00273

```

```

00274     _pos = r;
00275     return _pos;
00276 }
00277
00278 ~Be_file_pos() noexcept = 0;
00279
00280 protected:
00281     off64_t pos() const noexcept { return _pos; }
00282
00283 private:
00284     off64_t _pos;
00285 };
00286
00287 inline Be_file_pos::~Be_file_pos() noexcept {}
00288
00289 class Be_file_stream : public Be_file
00290 {
00291 public:
00292     ssize_t preadv(const struct iovec *v, int iovcnt, off64_t) noexcept override
00293     { return readv(v, iovcnt); }
00294
00295     ssize_t pwritev(const struct iovec *v, int iovcnt, off64_t) noexcept override
00296     { return writev(v, iovcnt); }
00297
00298     ~Be_file_stream() noexcept = 0;
00299 };
00300
00301 inline Be_file_stream::~Be_file_stream() noexcept {}
00302
00303 class Be_file_system : public File_system
00304 {
00305 private:
00306     char const *_fstype;
00307
00308 public:
00309     explicit Be_file_system(char const *_fstype) noexcept
00310     : File_system(), _fstype(_fstype)
00311     {
00312         vfs_ops->register_file_system(this);
00313     }
00314
00315     ~Be_file_system() noexcept
00316     {
00317         vfs_ops->unregister_file_system(this);
00318     }
00319
00320     char const *_type() const noexcept override { return _fstype; }
00321 };
00322
00323 /* Make sure filesystems can register before the constructor of libmount
00324  * runs */
00325 #define L4RE_VFS_FILE_SYSTEM_ATTRIBUTE \
00326     __attribute__((init_priority(INIT_PRIO_LATE)))
00327
00328 }
00329
00330 }

```

16.255 default_ops_impl.h

```

00001 // vi:ft=cpp
00002 /*
00003  * Copyright (C) 2016, 2023-2024 Kernkonzept GmbH.
00004  * Author(s): Alexander Warg <alexander.warg@kernkonzept.com>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #include <l4/cxx/static_container>
00009
00010 namespace {
00011 struct Vfs_init
00012 {
00013     // The Static_containers are used to prevent automatic destruction during
00014     // program shutdown. At least the `vfs` object must never be destructed
00015     // because any later attempt to do any kind of file-descriptor access in
00016     // the program would crash, and we could not be sure that the destructor
00017     // would really be executed after each possible operation using files or file
00018     // descriptors.
00019     cxx::Static_container<Vfs> vfs;
00020
00021     // The Static_containers below are just for providing ordering. The factories
00022     // must be initialized after the `vfs` object.
00023     cxx::Static_container<L4Re::Vfs::File_factory_t<L4Re::Dataspace, L4Re::Core::Ro_file> > ro_file;

```

```

00024   cxx::Static_container<L4Re::Vfs::File_factory_t<L4Re::Namespace, L4Re::Core::Ns_dir> > ns_dir;
00025   cxx::Static_container<L4Re::Vfs::File_factory_t<L4::Vcon, L4Re::Core::Vcon_stream> > vcon_stream;
00026
00027   Vfs_init()
00028   {
00029       vfs.construct();
00030       __rtld_l4re_env_posix_vfs_ops = vfs;
00031       ns_dir.construct();
00032       auto ns_ptr = cxx::ref_ptr(ns_dir.get());
00033       vfs->register_file_factory(ns_ptr);
00034       ns_ptr.release(); // prevent deletion of static object
00035
00036       ro_file.construct();
00037       auto ro_ptr = cxx::ref_ptr(ro_file.get());
00038       vfs->register_file_factory(ro_ptr);
00039       ro_ptr.release(); // prevent deletion of static object
00040
00041       vcon_stream.construct();
00042       auto vcon_ptr = cxx::ref_ptr(vcon_stream.get());
00043       vfs->register_file_factory(vcon_ptr);
00044       vcon_ptr.release(); // prevent deletion of static object
00045   }
00046 };
00047
00048 static Vfs_init __vfs_init __attribute__((init_priority(INIT_PRIO_VFS_INIT)));
00049
00050 };

```

16.256 fd_store.h

```

00001 /*
00002  * (c) 2010 Alexander Warg <warg@os.inf.tu-dresden.de>
00003  *     economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include <l4/l4re_vfs/vfs.h>
00010
00011 namespace L4Re { namespace Core {
00012
00013     using cxx::Ref_ptr;
00014
00015     class Fd_store
00016     {
00017     public:
00018         enum { MAX_FILES = 50 };
00019
00020         Fd_store() noexcept : _fd_hint(0) {}
00021
00022         int alloc() noexcept;
00023         void free(int fd) noexcept;
00024         bool check_fd(int fd) noexcept;
00025         Ref_ptr<L4Re::Vfs::File> get(int fd) noexcept;
00026         void set(int fd, Ref_ptr<L4Re::Vfs::File> const &f) noexcept;
00027
00028     private:
00029         int _fd_hint;
00030         Ref_ptr<L4Re::Vfs::File> _files[MAX_FILES];
00031     };
00032
00033     inline
00034     bool
00035     Fd_store::check_fd(int fd) noexcept
00036     {
00037         return fd >= 0 && fd < MAX_FILES;
00038     }
00039
00040     inline
00041     Ref_ptr<L4Re::Vfs::File>
00042     Fd_store::get(int fd) noexcept
00043     {
00044         if (check_fd(fd))
00045             return _files[fd];
00046
00047         return Ref_ptr<>::Nil;
00048     }
00049
00050     inline
00051     void
00052     Fd_store::set(int fd, Ref_ptr<L4Re::Vfs::File> const &f) noexcept
00053     {

```

```

00054     _files[fd] = f;
00055 }
00056
00057 }}
```

16.257 fd_store_impl.h

```

00001 /*
00002  * (c) 2010 Alexander Warg <warg@os.inf.tu-dresden.de>
00003  *     economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #include "fd_store.h"
00008
00009 namespace L4Re { namespace Core {
00010
00011     int
00012     Fd_store::alloc() noexcept
00013     {
00014         for (int i = _fd_hint; i < MAX_FILES; ++i)
00015         {
00016             if (!_files[i])
00017             {
00018                 _fd_hint = i + 1;
00019                 return i;
00020             }
00021         }
00022
00023         return -1;
00024     }
00025
00026     void
00027     Fd_store::free(int fd) noexcept
00028     {
00029         _files[fd] = 0;
00030         if (fd < _fd_hint)
00031             _fd_hint = fd;
00032     }
00033
00034 }}
00035
```

16.258 ns_fs.h

```

00001 /*
00002  * (c) 2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *     Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *     economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include <l4/l4re_vfs/backend>
00011 #include <l4/sys/capability>
00012 #include <l4/re/namespace>
00013 #include <l4/re/unique_cap>
00014
00015 namespace L4Re { namespace Core {
00016
00017     using cxx::Ref_ptr;
00018
00019     class Env_dir : public L4Re::Vfs::Be_file
00020     {
00021     public:
00022         explicit Env_dir(L4Re::Env const *env)
00023             : _env(env), _current_cap_entry(env->initial_caps())
00024         {}
00025
00026         ssize_t readv(const struct iovec*, int) noexcept override { return -EISDIR; }
00027         ssize_t writev(const struct iovec*, int) noexcept override { return -EISDIR; }
00028         ssize_t preadv(const struct iovec*, int, off64_t) noexcept override { return -EISDIR; }
00029         ssize_t pwritev(const struct iovec*, int, off64_t) noexcept override { return -EISDIR; }
00030         int fstat(struct stat64 *) const noexcept override;
00031         int faccessat(const char *path, int mode, int flags) noexcept override;
00032         int get_entry(const char *path, int flags, mode_t mode,
00033                     Ref_ptr<L4Re::Vfs::File> *) noexcept override;
00034     };
00035
```



```

00034     ssize_t getdents(char *, size_t) noexcept override;
00035
00036     ~Env_dir() noexcept {}
00037
00038 private:
00039     int get_ds(const char *path, L4Re::Unique_cap<L4Re::Dataspace> *ds) noexcept;
00040     bool check_type(Env::Cap_entry const *e, long protocol) noexcept;
00041
00042     L4Re::Env const *_env;
00043     Env::Cap_entry const *_current_cap_entry;
00044 };
00045
00046 class Ns_dir : public L4Re::Vfs::Be_file
00047 {
00048 public:
00049     explicit Ns_dir(L4::Cap<L4Re::Namespace> ns)
00050         : _ns(ns), _current_dir_pos(0)
00051     {}
00052
00053     ssize_t readv(const struct iovec*, int) noexcept override { return -EISDIR; }
00054     ssize_t writev(const struct iovec*, int) noexcept override { return -EISDIR; }
00055     ssize_t preadv(const struct iovec*, int, off64_t) noexcept override { return -EISDIR; }
00056     ssize_t pwritev(const struct iovec*, int, off64_t) noexcept override { return -EISDIR; }
00057     int fstat(struct stat64 *) const noexcept override;
00058     int faccessat(const char *path, int mode, int flags) noexcept override;
00059     int get_entry(const char *path, int flags, mode_t mode,
00060                  Ref_ptr<L4Re::Vfs::File> *) noexcept override;
00061     ssize_t getdents(char *, size_t) noexcept override;
00062
00063     ~Ns_dir() noexcept {}
00064
00065 private:
00066     int get_ds(const char *path, L4Re::Unique_cap<L4Re::Dataspace> *ds) noexcept;
00067
00068     L4::Cap<L4Re::Namespace> _ns;
00069     size_t _current_dir_pos;
00070 };
00071
00072 {}

```

16.259 ns_fs_impl.h

```

00001 /*
00002  * (c) 2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #include "ns_fs.h"
00009
00010 #include <l4/re/dataspace>
00011 #include <l4/re/util/env_ns>
00012 #include <l4/re/unique_cap>
00013 #include <dirent.h>
00014
00015 namespace L4Re { namespace Core {
00016
00017     static
00018     Ref_ptr<L4Re::Vfs::File>
00019     cap_to_vfs_object(L4::Cap<void> o, int *err)
00020     {
00021         L4::Cap<L4::Meta> m = L4::cap_reinterpret_cast<L4::Meta>(o);
00022         long proto = 0;
00023         char name_buf[256];
00024         L4::Ipc::String<char> name(sizeof(name_buf), name_buf);
00025         l4_ret_t r = l4_error(m->interface(0, &proto, &name));
00026         *err = -ENOPROTOPT;
00027         if (r < 0)
00028             // could not get type of object so bail out
00029             return Ref_ptr<L4Re::Vfs::File>();
00030
00031         *err = -EPROTO;
00032         Ref_ptr<L4Re::Vfs::File_factory> factory;
00033
00034         if (proto != 0)
00035             factory = L4Re::Vfs::vfs_ops->get_file_factory(proto);
00036
00037         if (!factory)
00038             factory = L4Re::Vfs::vfs_ops->get_file_factory(name.data);
00039
00040         if (!factory)
00041             return Ref_ptr<L4Re::Vfs::File>();
00042     }
00043 } }

```

```

00042
00043     *err = -ENOMEM;
00044     return factory->create(o);
00045 }
00046
00047
00048 int
00049 Ns_dir::get_ds(const char *path, L4Re::Unique_cap<L4Re::Dataspace> *ds) noexcept
00050 {
00051     auto file = L4Re::make_unique_cap<L4Re::Dataspace>(L4Re::virt_cap_alloc);
00052
00053     if (!file.is_valid())
00054         return -ENOMEM;
00055
00056     int err = _ns->query(path, file.get());
00057
00058     if (err < 0)
00059         return -ENOENT;
00060
00061     *ds = cxx::move(file);
00062     return err;
00063 }
00064
00065 int
00066 Ns_dir::get_entry(const char *path, int /*flags*/, mode_t /*mode*/,
00067                  Ref_ptr<L4Re::Vfs::File> *f) noexcept
00068 {
00069     if (!*path)
00070     {
00071         *f = cxx::ref_ptr(this);
00072         return 0;
00073     }
00074
00075     L4Re::Unique_cap<Dataspace> file;
00076     int err = get_ds(path, &file);
00077
00078     if (err < 0)
00079         return -ENOENT;
00080
00081     cxx::Ref_ptr<L4Re::Vfs::File> fi = cap_to_vfs_object(file.get(), &err);
00082     if (!fi)
00083         return err;
00084
00085     file.release();
00086     *f = cxx::move(fi);
00087     return 0;
00088 }
00089
00090 int
00091 Ns_dir::faccessat(const char *path, int mode, int /*flags*/) noexcept
00092 {
00093     auto tmpcap = L4Re::make_unique_cap<void>(L4Re::virt_cap_alloc);
00094
00095     if (!tmpcap.is_valid())
00096         return -ENOMEM;
00097
00098     if (_ns->query(path, tmpcap.get()))
00099         return -ENOENT;
00100
00101     if (mode & W_OK)
00102         return -EACCES;
00103
00104     return 0;
00105 }
00106
00107 int
00108 Ns_dir::fstat(struct stat64 *b) const noexcept
00109 {
00110     b->st_dev = 1;
00111     b->st_ino = 1;
00112     b->st_mode = S_IRWXU | S_IFDIR;
00113     b->st_nlink = 0;
00114     b->st_uid = 0;
00115     b->st_gid = 0;
00116     b->st_rdev = 0;
00117     b->st_size = 0;
00118     b->st_blksize = 0;
00119     b->st_blocks = 0;
00120     b->st_atime = 0;
00121     b->st_mtime = 0;
00122     b->st_ctime = 0;
00123     return 0;
00124 }
00125
00126 ssize_t
00127 Ns_dir::getdents(char *buf, size_t dest_sz) noexcept
00128 {

```

```

00129 struct dirent64 *dest = reinterpret_cast<struct dirent64 *>(buf);
00130 ssize_t ret = 0;
00131 L4_addr_t infoaddr;
00132 size_t infosz;
00133
00134 L4Re::Unique_cap<Dataspace> dirinfofile;
00135 int err = get_ds(".dirinfo", &dirinfofile);
00136 if (err)
00137     return 0;
00138
00139 infosz = dirinfofile->size();
00140 if (infosz <= 0)
00141     return 0;
00142
00143 infoaddr = L4_PAGESIZE;
00144 err = L4Re::Env::env()->rm()->attach(&infoaddr, infosz,
00145                                     Rm::F::Search_addr | Rm::F::R,
00146                                     dirinfofile.get(), 0);
00147 if (err < 0)
00148     return 0;
00149
00150 char *p = reinterpret_cast<char *>(infoaddr) + _current_dir_pos;
00151 char *end = reinterpret_cast<char *>(infoaddr) + infosz;
00152
00153 char *current_dirinfo_entry = p;
00154 while (dest && p < end)
00155 {
00156     // parse lines of dirinfofile
00157     long len = 0;
00158     for (; p < end && *p >= '0' && *p <= '9'; ++p)
00159     {
00160         len *= 10;
00161         len += *p - '0';
00162     }
00163
00164     if (len == 0)
00165         break;
00166
00167     if (p == end)
00168         break;
00169
00170     if (*p != ':')
00171         break;
00172     p++; // skip colon
00173
00174     if (p + len >= end)
00175         break;
00176
00177     unsigned l = len + 1;
00178     if (l > sizeof(dest->d_name))
00179         l = sizeof(dest->d_name);
00180
00181     unsigned n = offsetof(struct dirent64, d_name) + 1;
00182     n = (n + sizeof(long) - 1) & ~(sizeof(long) - 1);
00183
00184     if (n > dest_sz)
00185         break;
00186
00187     dest->d_ino = 1;
00188     dest->d_off = 0;
00189     memcpy(dest->d_name, p, l - 1);
00190     dest->d_name[l - 1] = 0;
00191     dest->d_reclen = n;
00192     dest->d_type = DT_UNKNOWN;
00193     ret += n;
00194     dest_sz -= n;
00195
00196     // next entry
00197     dest = reinterpret_cast<struct dirent64 *>
00198         (reinterpret_cast<unsigned long>(dest) + n);
00199
00200     // next infodirfile line
00201     p += len;
00202     while (p < end && *p && (*p == '\\n' || *p == '\\r'))
00203         p++;
00204
00205     current_dirinfo_entry = p;
00206 }
00207
00208 _current_dir_pos = current_dirinfo_entry - reinterpret_cast<char *>(infoaddr);
00209
00210 if (!ret) // hack since we should only reset this at open times
00211     _current_dir_pos = 0;
00212
00213 L4Re::Env::env()->rm()->detach(infoaddr, 0);
00214
00215 return ret;

```

```

00216 }
00217
00218 int
00219 Env_dir::get_ds(const char *path, L4Re::Unique_cap<L4Re::Dataspace> *ds) noexcept
00220 {
00221     while (*path == '/')
00222         ++path;
00223     Vfs::Path p(path);
00224     Vfs::Path first = p.strip_first();
00225
00226     if (first.empty())
00227         return -ENOENT;
00228
00229     L4::Cap<L4Re::Namespace>
00230     c = _env->get_cap<L4Re::Namespace>(first.path(), first.length());
00231
00232     if (!c.is_valid())
00233         return -ENOENT;
00234
00235     if (p.empty())
00236     {
00237         *ds = L4Re::Unique_cap<L4Re::Dataspace>(L4::cap_reinterpret_cast<L4Re::Dataspace>(c));
00238         return 0;
00239     }
00240
00241     auto file = L4Re::make_unique_cap<L4Re::Dataspace>(L4Re::virt_cap_alloc);
00242
00243     if (!file.is_valid())
00244         return -ENOMEM;
00245
00246     int err = c->query(p.path(), p.length(), file.get());
00247
00248     if (err < 0)
00249         return -ENOENT;
00250
00251     *ds = cxx::move(file);
00252     return err;
00253 }
00254
00255 int
00256 Env_dir::get_entry(const char *path, int /*flags*/, mode_t /*mode*/,
00257                   Ref_ptr<L4Re::Vfs::File> *f) noexcept
00258 {
00259     if (!*path)
00260     {
00261         *f = cxx::ref_ptr(this);
00262         return 0;
00263     }
00264
00265     L4Re::Unique_cap<Dataspace> file;
00266     int err = get_ds(path, &file);
00267
00268     if (err < 0)
00269         return -ENOENT;
00270
00271     cxx::Ref_ptr<L4Re::Vfs::File> fi = cap_to_vfs_object(file.get(), &err);
00272     if (!fi)
00273         return err;
00274
00275     file.release();
00276     *f = cxx::move(fi);
00277     return 0;
00278 }
00279
00280 int
00281 Env_dir::faccessat(const char *path, int mode, int /*flags*/) noexcept
00282 {
00283     while (*path == '/')
00284         ++path;
00285     Vfs::Path p(path);
00286     Vfs::Path first = p.strip_first();
00287
00288     if (first.empty())
00289         return -ENOENT;
00290
00291     L4::Cap<L4Re::Namespace>
00292     c = _env->get_cap<L4Re::Namespace>(first.path(), first.length());
00293
00294     if (!c.is_valid())
00295         return -ENOENT;
00296
00297     if (p.empty())
00298     {
00299         if (mode & W_OK)
00300             return -EACCES;
00301
00302         return 0;

```

```

00303     }
00304
00305     auto tmpcap = L4Re::make_unique_cap<void>(L4Re::virt_cap_alloc);
00306
00307     if (!tmpcap.is_valid())
00308         return -ENOMEM;
00309
00310     if (c->query(p.path(), p.length(), tmpcap.get()))
00311         return -ENOENT;
00312
00313     if (mode & W_OK)
00314         return -EACCES;
00315
00316     return 0;
00317 }
00318
00319 bool
00320 Env_dir::check_type(Env::Cap_entry const *e, long protocol) noexcept
00321 {
00322     L4::Cap<L4::Meta> m(e->cap);
00323     return m->supports(protocol).label();
00324 }
00325
00326 int
00327 Env_dir::fstat(struct stat64 *b) const noexcept
00328 {
00329     b->st_dev = 1;
00330     b->st_ino = 1;
00331     b->st_mode = S_IRWXU | S_IFDIR;
00332     b->st_nlink = 0;
00333     b->st_uid = 0;
00334     b->st_gid = 0;
00335     b->st_rdev = 0;
00336     b->st_size = 0;
00337     b->st_blksize = 0;
00338     b->st_blocks = 0;
00339     b->st_atime = 0;
00340     b->st_mtime = 0;
00341     b->st_ctime = 0;
00342     return 0;
00343 }
00344
00345 ssize_t
00346 Env_dir::getdents(char *buf, size_t sz) noexcept
00347 {
00348     struct dirent64 *d = reinterpret_cast<struct dirent64 *>(buf);
00349     ssize_t ret = 0;
00350
00351     while (d
00352            && _current_cap_entry
00353            && _current_cap_entry->flags != ~0UL)
00354     {
00355         unsigned l = strlen(_current_cap_entry->name) + 1;
00356         if (l > sizeof(d->d_name))
00357             l = sizeof(d->d_name);
00358
00359         unsigned n = offsetof(struct dirent64, d_name) + 1;
00360         n = (n + sizeof(long) - 1) & ~(sizeof(long) - 1);
00361
00362         if (n <= sz)
00363         {
00364             d->d_ino = 1;
00365             d->d_off = 0;
00366             memcpy(d->d_name, _current_cap_entry->name, l);
00367             d->d_name[l - 1] = 0;
00368             d->d_reclen = n;
00369             if (check_type(_current_cap_entry, L4Re::Namespace::Protocol))
00370                 d->d_type = DT_DIR;
00371             else if (check_type(_current_cap_entry, L4Re::Dataspace::Protocol))
00372                 d->d_type = DT_REG;
00373             else
00374                 d->d_type = DT_UNKNOWN;
00375             ret += n;
00376             sz -= n;
00377             d = reinterpret_cast<struct dirent64 *>
00378                 (reinterpret_cast<unsigned long*>(d) + n);
00379             _current_cap_entry++;
00380         }
00381         else
00382             return ret;
00383     }
00384
00385     // bit of a hack because we should only (re)set this when opening the dir
00386     if (!ret)
00387         _current_cap_entry = _env->initial_caps();
00388
00389     return ret;

```

```
00390 }
00391
00392 }}
```

16.260 ro_file.h

```
00001 /*
00002  * (c) 2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include <l4/l4re_vfs/backend>
00011
00012 namespace L4Re { namespace Core {
00013
00014 class Ro_file : public L4Re::Vfs::Be_file_pos
00015 {
00016 private:
00017     L4::Cap<L4Re::Dataspace> _ds;
00018     off64_t _size;
00019     char const *_addr;
00020
00021 public:
00022     explicit Ro_file(L4::Cap<L4Re::Dataspace> ds) noexcept
00023         : Be_file_pos(), _ds(ds), _addr(0)
00024     {
00025         _size = _ds->size();
00026     }
00027
00028     L4::Cap<L4Re::Dataspace> data_space() noexcept override { return _ds; }
00029
00030     int fstat(struct stat64 *buf) const noexcept override;
00031
00032     int ioctl(unsigned long, va_list) noexcept override;
00033
00034     off64_t size() const noexcept override { return _size; }
00035
00036     int get_status_flags() const noexcept override
00037     { return O_RDONLY; }
00038
00039     int set_status_flags(long) noexcept override
00040     { return 0; }
00041
00052     bool check_ready(Ready_type rt) noexcept override
00053     { return rt == Read; }
00054
00055     ~Ro_file() noexcept;
00056
00057 private:
00058     ssize_t read_single(const struct iovec*, off64_t) noexcept;
00059     ssize_t preadv(const struct iovec *, int, off64_t) noexcept override;
00060     ssize_t pwritev(const struct iovec *, int , off64_t) noexcept override;
00061 };
00062
00063
00064 }}
```

16.261 ro_file_impl.h

```
00001 /*
00002  * (c) 2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008
00009 #include "ro_file.h"
00010
00011 #include <sys/ioctl.h>
00012
00013 #include <l4/re/env>
00014
00015 namespace L4Re { namespace Core {
```

```

00016
00017 Ro_file::~~Ro_file() noexcept
00018 {
00019     if (_addr)
00020         L4Re::Env::env()->rm()->detach(l4_addr_t(_addr), 0);
00021
00022     L4Re::virt_cap_alloc->release(_ds);
00023 }
00024
00025 int
00026 Ro_file::fstat(struct stat64 *buf) const noexcept
00027 {
00028     static int fake = 0;
00029
00030     memset(buf, 0, sizeof(*buf));
00031     buf->st_size = _size;
00032     buf->st_mode = S_IFREG | 0644;
00033     buf->st_dev = _ds.cap();
00034     buf->st_ino = ++fake;
00035     buf->st_blksize = L4_PAGE_SIZE;
00036     buf->st_blocks = l4_round_page(_size);
00037     return 0;
00038 }
00039
00040 ssize_t
00041 Ro_file::read_single(const struct iovec *vec, off64_t pos) noexcept
00042 {
00043     off64_t l = vec->iiov_len;
00044     if (_size - pos < l)
00045         l = _size - pos;
00046
00047     if (l > 0)
00048     {
00049         Vfs_config::memcpy(vec->iiov_base, _addr + pos, l);
00050         return l;
00051     }
00052
00053     return 0;
00054 }
00055
00056 ssize_t
00057 Ro_file::preadv(const struct iovec *vec, int cnt, off64_t offset) noexcept
00058 {
00059     if (!_addr)
00060     {
00061         void const *file = reinterpret_cast<void*>(L4_PAGE_SIZE);
00062         long err = L4Re::Env::env()->rm()->attach(&file, _size,
00063             Rm::F::Search_addr | Rm::F::R,
00064             _ds, 0);
00065
00066         if (err < 0)
00067             return err;
00068
00069         _addr = static_cast<char const *>(file);
00070     }
00071
00072     ssize_t l = 0;
00073
00074     while (cnt > 0)
00075     {
00076         ssize_t r = read_single(vec, offset);
00077         offset += r;
00078         l += r;
00079
00080         if (static_cast<size_t>(r) < vec->iiov_len)
00081             return l;
00082
00083         ++vec;
00084         --cnt;
00085     }
00086     return l;
00087 }
00088
00089 ssize_t
00090 Ro_file::pwritev(const struct iovec *, int, off64_t) noexcept
00091 {
00092     return -EROFS;
00093 }
00094
00095 int
00096 Ro_file::ioctl(unsigned long v, va_list args) noexcept
00097 {
00098     switch (v)
00099     {
00100         case FIONREAD: // return amount of data still available
00101             int *available = va_arg(args, int *);
00102             *available = _size - pos();

```

```

00103         return 0;
00104     };
00105     return -ENOTTY;
00106 }
00107
00108 }}

```

16.262 vcon_stream.h

```

00001 /*
00002  * (c) 2010 Alexander Warg <warg@os.inf.tu-dresden.de>
00003  *     economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include <l4/sys/capability>
00010 #include <l4/sys/vcon>
00011 #include <l4/sys/semaphore>
00012
00013 #include <l4/l4re_vfs/backend>
00014
00015 namespace L4Re { namespace Core {
00016
00017     class Vcon_stream : public L4Re::Vfs::Be_file_stream
00018     {
00019     private:
00020         L4::Cap<L4::Vcon> _s;
00021         L4::Cap<L4::Semaphore> _irq;
00022         unsigned _irq_bound;
00023
00024     public:
00025         explicit Vcon_stream(L4::Cap<L4::Vcon> s) noexcept;
00026
00027         ssize_t readv(const struct iovec*, int iovcnt) noexcept override;
00028         ssize_t writev(const struct iovec*, int iovcnt) noexcept override;
00029         int fstat(struct stat64 *buf) const noexcept override;
00030         int get_status_flags() const noexcept override { return O_RDWR; }
00031         int set_status_flags(long) noexcept override { return 0; }
00032         int ioctl(unsigned long request, va_list args) noexcept override;
00033
00034         ~Vcon_stream() noexcept {}
00035         void operator delete (void *) {}
00036     };
00037
00038 }}

```

16.263 vcon_stream_impl.h

```

00001 /*
00002  * (c) 2010 Alexander Warg <warg@os.inf.tu-dresden.de>
00003  *     economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00008 #include <l4/re/env>
00009 #include <l4/sys/factory>
00010 #include <l4/cxx/minmax>
00011
00012 #include "vcon_stream.h"
00013
00014 #include <limits.h>
00015 #include <termios.h>
00016 #include <unistd.h>
00017 #include <sys/ioctl.h>
00018 #include <sys/ttydefaults.h>
00019
00020 namespace L4Re { namespace Core {
00021     Vcon_stream::Vcon_stream(L4::Cap<L4::Vcon> s) noexcept
00022     : Be_file_stream(),
00023       _s(s), _irq(L4Re::virt_cap_alloc->alloc<L4::Semaphore>()), _irq_bound(false)
00024     {
00025         // [[maybe_unused]] int res =
00026         l4_error(L4Re::Env::env()->factory()->create(_irq));
00027         // (void)res; // handle errors!
00028     }

```



```

00029
00030 ssize_t
00031 Vcon_stream::readv(const struct iovec *iovec, int iovcnt) noexcept
00032 {
00033     if (iovcnt < 0)
00034         return -EINVAL;
00035
00036     if (!_irq_bound)
00037     {
00038         bool was_bound = __atomic_exchange_n(&_irq_bound, true, __ATOMIC_SEQ_CST);
00039         if (!was_bound)
00040             if (l4_error(_s->bind(0, _irq)) < 0)
00041                 return -EIO;
00042     }
00043
00044     ssize_t bytes = 0;
00045     for (; iovcnt > 0; --iovcnt, ++iovec)
00046     {
00047         size_t len = cxx::min<size_t>(iovec->iov_len, SSIZE_MAX - bytes);
00048         if (len == 0)
00049             continue;
00050
00051         char *buf = static_cast<char *>(iovec->iov_base);
00052
00053         while (1)
00054         {
00055             size_t l = cxx::min<size_t>(L4_VCON_READ_SIZE, len);
00056             int ret = _s->read(buf, l);
00057
00058             if (ret > static_cast<int>(l))
00059                 ret = l;
00060
00061             if (ret < 0)
00062                 return ret;
00063             else if (ret == 0)
00064             {
00065                 if (bytes)
00066                     return bytes;
00067
00068                 ret = _s->read(buf, l);
00069                 if (ret < 0)
00070                     return ret;
00071                 else if (ret == 0)
00072                 {
00073                     _irq->down();
00074                     continue;
00075                 }
00076             }
00077
00078             bytes += ret;
00079             len -= ret;
00080             buf += ret;
00081
00082             if (len == 0)
00083                 break;
00084         }
00085     }
00086
00087     return bytes;
00088 }
00089
00090 ssize_t
00091 Vcon_stream::writev(const struct iovec *iovec, int iovcnt) noexcept
00092 {
00093     l4_msg_regs_t store;
00094     l4_msg_regs_t *mr = l4_utcb_mr();
00095
00096     if (iovcnt < 0)
00097         return -EINVAL;
00098
00099     Vfs_config::memcpy(&store, mr, sizeof(store));
00100
00101     ssize_t written = 0;
00102     while (iovcnt)
00103     {
00104         size_t sl = cxx::min<size_t>(iovec->iov_len, SSIZE_MAX - written);
00105         char const *b = static_cast<char const *>(iovec->iov_base);
00106
00107         for (; sl > L4_VCON_WRITE_SIZE;
00108             ; sl -= L4_VCON_WRITE_SIZE, b += L4_VCON_WRITE_SIZE,
00109             written += L4_VCON_WRITE_SIZE)
00110             _s->send(b, L4_VCON_WRITE_SIZE);
00111
00112         _s->send(b, sl);
00113
00114         written += sl;
00115     }

```

```

00116         ++iovec;
00117         --iovcnt;
00118     }
00119     Vfs_config::memcpy(mr, &store, sizeof(store));
00120     return written;
00121 }
00122
00123 int
00124 Vcon_stream::fstat(struct stat64 *buf) const noexcept
00125 {
00126     buf->st_size = 0;
00127     buf->st_mode = 0666;
00128     buf->st_dev = _s.cap();
00129     buf->st_ino = 0;
00130     return 0;
00131 }
00132
00133 int
00134 Vcon_stream::ioctl(unsigned long request, va_list args) noexcept
00135 {
00136     switch (request) {
00137     case TCGETS:
00138     {
00139         //vt100_tcgetattr(term, (struct termios *)argp);
00140
00141         struct termios *t = va_arg(args, struct termios *);
00142
00143         l4_vcon_attr_t l4a;
00144         if (!l4_error(_s->get_attr(&l4a)))
00145         {
00146             t->c_iflag = l4a.i_flags;
00147             t->c_oflag = l4a.o_flags; // output flags
00148             t->c_cflag = 0; // control flags
00149             t->c_lflag = l4a.l_flags; // local flags
00150         }
00151         else
00152             t->c_iflag = t->c_oflag = t->c_cflag = t->c_lflag = 0;
00153         #if 0
00154             //t->c_lflag |= ECHO; // if term->echo
00155             t->c_lflag |= ICANON; // if term->term_mode == VT100MODE_COOKED
00156         #endif
00157
00158         t->c_cc[VEOF] = CEOF;
00159         t->c_cc[VEOL] = _POSIX_VDISABLE;
00160         t->c_cc[VEOL2] = _POSIX_VDISABLE;
00161         t->c_cc[VERASE] = CERASE;
00162         t->c_cc[VWERASE] = CWERASE;
00163         t->c_cc[VKILL] = CKILL;
00164         t->c_cc[VREPRINT] = CREPRINT;
00165         t->c_cc[VINTR] = CINTR;
00166         t->c_cc[VQUIT] = _POSIX_VDISABLE;
00167         t->c_cc[VSUSP] = CSUSP;
00168         t->c_cc[VSTART] = CSTART;
00169         t->c_cc[VSTOP] = CSTOP;
00170         t->c_cc[VLNEXT] = CLNEXT;
00171         t->c_cc[VDISCARD] = CDISCARD;
00172         t->c_cc[VMIN] = CMIN;
00173         t->c_cc[VTIME] = 0;
00174
00175     }
00176
00177     return 0;
00178
00179     case TCSETS:
00180     case TCSETSW:
00181     case TCSETSF:
00182     {
00183         //vt100_tcsetattr(term, (struct termios *)argp);
00184         struct termios const *t = va_arg(args, struct termios const *);
00185
00186         // XXX: well, we're cheating, get this from the other side!
00187
00188         l4_vcon_attr_t l4a;
00189         l4a.i_flags = t->c_iflag;
00190         l4a.o_flags = t->c_oflag; // output flags
00191         l4a.l_flags = t->c_lflag; // local flags
00192         _s->set_attr(&l4a);
00193     }
00194     return 0;
00195
00196     default:
00197         break;
00198     };
00199     return -ENOTTY;
00200 }
00201
00202 }

```

16.264 vfs_impl.h

```

00001 /*
00002  * (c) 2008-2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00004  *      Björn Döbel <doebel@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009
00010 #include "fd_store.h"
00011 #include "vcon_stream.h"
00012 #include "ns_fs.h"
00013
00014 #include <l4/bid_config.h>
00015 #include <l4/re/env>
00016 #include <l4/re/rm>
00017 #include <l4/re/dataspace>
00018 #include <l4/sys/assert.h>
00019 #include <l4/cxx/hlist>
00020 #include <l4/cxx/pair>
00021 #include <l4/cxx/std_alloc>
00022
00023 #include <l4/l4re_vfs/backend>
00024 #include <l4/re/shared_cap>
00025
00026 #include <unistd.h>
00027 #include <stdarg.h>
00028 #include <errno.h>
00029 #include <sys/uio.h>
00030 #include <sys/mman.h>
00031
00032 #if 0
00033 #include <l4/sys/kdebug.h>
00034 static int debug_mmap = 1;
00035 #define DEBUG_LOG(level, dbg...) do { if (level) dbg } while (0)
00036 #else
00037 #define DEBUG_LOG(level, dbg...) do { } while (0)
00038 #endif
00039
00040 #define USE_BIG_ANON_DS
00041
00042 using L4Re::Rm;
00043
00044 namespace {
00045 using cxx::Ref_ptr;
00046
00047 class Fd_store : public L4Re::Core::Fd_store
00048 {
00049 public:
00050     Fd_store() noexcept;
00051 };
00052
00053 // for internal Vcon_streams we want to have a placement new operator, so
00054 // inherit and add one
00055 class Std_stream : public L4Re::Core::Vcon_stream
00056 {
00057 public:
00058     Std_stream(L4::Cap<L4::Vcon> c) : L4Re::Core::Vcon_stream(c) {}
00059 };
00060
00061 Fd_store::Fd_store() noexcept
00062 {
00063     // use this strange way to prevent deletion of the stdio object
00064     // this depends on Fd_store to being a singleton !!!
00065     static char m[sizeof(Std_stream)] __attribute__((aligned(sizeof(long))));
00066     if (auto log = L4Re::Env::env()->log())
00067     {
00068         Std_stream *s = new (m) Std_stream(log);
00069         set(0, cxx::ref_ptr(s)); // stdin
00070         set(1, cxx::ref_ptr(s)); // stdout
00071         set(2, cxx::ref_ptr(s)); // stderr
00072
00073         // make sure that we never delete the static io stream thing
00074         s->add_ref();
00075     }
00076 }
00077
00078 class Root_mount_tree : public L4Re::Vfs::Mount_tree
00079 {
00080 public:
00081     Root_mount_tree() : L4Re::Vfs::Mount_tree(0) {}
00082     void operator delete (void *) {}
00083 };

```

```

00090
00091 class Vfs : public L4Re::Vfs::Ops
00092 {
00093 private:
00094     bool _early_oom;
00095
00096 public:
00097     Vfs()
00098     : _early_oom(true), _root_mount(), _root(L4Re::Env::env())
00099     {
00100         _root_mount.add_ref();
00101         _root.add_ref();
00102         _root_mount.mount(cxx::ref_ptr(&_root));
00103         _cwd = cxx::ref_ptr(&_root);
00104
00105 #if 0
00106         Ref_ptr<L4Re::Vfs::File> rom;
00107         _root.openat("rom", 0, 0, &rom);
00108
00109         _root_mount.create_tree("lib/foo", rom);
00110
00111         _root.openat("lib", 0, 0, &_cwd);
00112
00113 #endif
00114     }
00115
00116     int alloc_fd(Ref_ptr<L4Re::Vfs::File> const &f) noexcept override;
00117     Ref_ptr<L4Re::Vfs::File> free_fd(int fd) noexcept override;
00118     Ref_ptr<L4Re::Vfs::File> get_root() noexcept override;
00119     Ref_ptr<L4Re::Vfs::File> get_cwd() noexcept override;
00120     void set_cwd(Ref_ptr<L4Re::Vfs::File> const &dir) noexcept override;
00121     Ref_ptr<L4Re::Vfs::File> get_file(int fd) noexcept override;
00122     cxx::Pair<Ref_ptr<L4Re::Vfs::File>, int>
00123         set_fd(int fd, Ref_ptr<L4Re::Vfs::File> const &f = Ref_ptr<>::Nil) noexcept
00124         override;
00125
00126     int mmap2(void *start, size_t len, int prot, int flags, int fd,
00127               off_t offset, void **ptr) noexcept override;
00128
00129     int munmap(void *start, size_t len) noexcept override;
00130     int mremap(void *old, size_t old_sz, size_t new_sz, int flags,
00131                void **new_addr) noexcept override;
00132     int mprotect(const void *a, size_t sz, int prot) noexcept override;
00133     int msync(void *addr, size_t len, int flags) noexcept override;
00134     int madvise(void *addr, size_t len, int advice) noexcept override;
00135
00136     int register_file_system(L4Re::Vfs::File_system *f) noexcept override;
00137     int unregister_file_system(L4Re::Vfs::File_system *f) noexcept override;
00138     L4Re::Vfs::File_system *get_file_system(char const *fstype) noexcept override;
00139     L4Re::Vfs::File_system_list file_system_list() noexcept override;
00140
00141     int register_file_factory(cxx::Ref_ptr<L4Re::Vfs::File_factory> f) noexcept override;
00142     int unregister_file_factory(cxx::Ref_ptr<L4Re::Vfs::File_factory> f) noexcept override;
00143     Ref_ptr<L4Re::Vfs::File_factory> get_file_factory(int proto) noexcept override;
00144     Ref_ptr<L4Re::Vfs::File_factory> get_file_factory(char const *proto_name) noexcept override;
00145     int mount(char const *path, cxx::Ref_ptr<L4Re::Vfs::File> const &dir) noexcept override;
00146
00147     void operator delete (void *) {}
00148
00149     void *malloc(size_t size) noexcept override { return Vfs_config::malloc(size); }
00150     void free(void *m) noexcept override { Vfs_config::free(m); }
00151
00152 private:
00153     Root_mount_tree _root_mount;
00154     L4Re::Core::Env_dir _root;
00155     Ref_ptr<L4Re::Vfs::File> _cwd;
00156     Fd_store fds;
00157
00158     L4Re::Vfs::File_system *_fs_registry;
00159
00160     struct File_factory_item : cxx::H_list_item_t<File_factory_item>
00161     {
00162         cxx::Ref_ptr<L4Re::Vfs::File_factory> f;
00163         explicit File_factory_item(cxx::Ref_ptr<L4Re::Vfs::File_factory> const &f)
00164             : f(f) {}
00165
00166         File_factory_item() = default;
00167         File_factory_item(File_factory_item const &) = delete;
00168         File_factory_item &operator = (File_factory_item const &) = delete;
00169     };
00170
00171     cxx::H_list_t<File_factory_item> _file_factories;
00172
00173     l4_addr_t _anon_offset;
00174     L4Re::Shared_cap<L4Re::Dataspace> _anon_ds;
00175
00176     int alloc_ds(unsigned long size, L4Re::Shared_cap<L4Re::Dataspace> *ds);

```

```

00177 int alloc_anon_mem(l4_umword_t size, L4Re::Shared_cap<L4Re::Dataspace> *ds,
00178                   l4_addr_t *offset);
00179
00180 void align_mmap_start_and_length(void **start, size_t *length);
00181 int munmap_regions(void *start, size_t len);
00182
00183 L4Re::Vfs::File_system *find_fs_from_type(char const *fstype) noexcept;
00184 };
00185
00186 static inline bool strequal(char const *a, char const *b)
00187 {
00188     for (; *a && *a == *b; ++a, ++b)
00189         ;
00190     return *a == *b;
00191 }
00192
00193 int
00194 Vfs::register_file_system(L4Re::Vfs::File_system *f) noexcept
00195 {
00196     using L4Re::Vfs::File_system;
00197
00198     if (!f)
00199         return -EINVAL;
00200
00201     for (File_system *c = _fs_registry; c; c = c->next())
00202         if (strequal(c->type(), f->type()))
00203             return -EEXIST;
00204
00205     f->next(_fs_registry);
00206     _fs_registry = f;
00207
00208     return 0;
00209 }
00210
00211 int
00212 Vfs::unregister_file_system(L4Re::Vfs::File_system *f) noexcept
00213 {
00214     using L4Re::Vfs::File_system;
00215
00216     if (!f)
00217         return -EINVAL;
00218
00219     File_system **p = &_fs_registry;
00220
00221     for (; *p; p = &(*p)->next())
00222         if (*p == f)
00223         {
00224             *p = f->next();
00225             f->next() = 0;
00226             return 0;
00227         }
00228
00229     return -ENOENT;
00230 }
00231
00232 L4Re::Vfs::File_system *
00233 Vfs::find_fs_from_type(char const *fstype) noexcept
00234 {
00235     L4Re::Vfs::File_system_list fsl(_fs_registry);
00236     for (L4Re::Vfs::File_system_list::Iterator c = fsl.begin();
00237          c != fsl.end(); ++c)
00238         if (strequal(c->type(), fstype))
00239             return *c;
00240     return 0;
00241 }
00242
00243 L4Re::Vfs::File_system_list
00244 Vfs::file_system_list() noexcept
00245 {
00246     return L4Re::Vfs::File_system_list(_fs_registry);
00247 }
00248
00249 L4Re::Vfs::File_system *
00250 Vfs::get_file_system(char const *fstype) noexcept
00251 {
00252     L4Re::Vfs::File_system *fs;
00253     if ((fs = find_fs_from_type(fstype)))
00254         return fs;
00255
00256     // Try to load a file system module dynamically
00257     int res = Vfs_config::load_module(fstype);
00258     if (res < 0)
00259         return 0;
00260
00261     // Try again
00262     return find_fs_from_type(fstype);
00263 }

```

```

00264
00265 int
00266 Vfs::register_file_factory(cxx::Ref_ptr<L4Re::Vfs::File_factory> f) noexcept
00267 {
00268     if (!f)
00269         return -EINVAL;
00270
00271     void *x = this->malloc(sizeof(File_factory_item));
00272     if (!x)
00273         return -ENOMEM;
00274
00275     auto ff = new (x, cxx::Nothrow()) File_factory_item(f);
00276     _file_factories.push_front(ff);
00277     return 0;
00278 }
00279
00280 int
00281 Vfs::unregister_file_factory(cxx::Ref_ptr<L4Re::Vfs::File_factory> f) noexcept
00282 {
00283     for (auto p: _file_factories)
00284     {
00285         if (p->f == f)
00286         {
00287             _file_factories.remove(p);
00288             p->~File_factory_item();
00289             this->free(p);
00290             return 0;
00291         }
00292     }
00293     return -ENOENT;
00294 }
00295
00296 Ref_ptr<L4Re::Vfs::File_factory>
00297 Vfs::get_file_factory(int proto) noexcept
00298 {
00299     for (auto p: _file_factories)
00300         if (p->f->proto() == proto)
00301             return p->f;
00302
00303     return Ref_ptr<L4Re::Vfs::File_factory>();
00304 }
00305
00306 Ref_ptr<L4Re::Vfs::File_factory>
00307 Vfs::get_file_factory(char const *proto_name) noexcept
00308 {
00309     for (auto p: _file_factories)
00310     {
00311         auto n = p->f->proto_name();
00312         if (n)
00313         {
00314             char const *a = n;
00315             char const *b = proto_name;
00316             for (; *a && *b && *a == *b; ++a, ++b)
00317                 ;
00318
00319             if ((*a == 0) && (*b == 0))
00320                 return p->f;
00321         }
00322     }
00323
00324     return Ref_ptr<L4Re::Vfs::File_factory>();
00325 }
00326
00327 int
00328 Vfs::alloc_fd(Ref_ptr<L4Re::Vfs::File> const &f) noexcept
00329 {
00330     int fd = fds.alloc();
00331     if (fd < 0)
00332         return -EMFILE;
00333
00334     if (f)
00335         fds.set(fd, f);
00336
00337     return fd;
00338 }
00339
00340 Ref_ptr<L4Re::Vfs::File>
00341 Vfs::free_fd(int fd) noexcept
00342 {
00343     Ref_ptr<L4Re::Vfs::File> f = fds.get(fd);
00344
00345     if (!f)
00346         return Ref_ptr<>::Nil;
00347
00348     fds.free(fd);
00349     return f;
00350 }

```

```

00351
00352
00353 Ref_ptr<L4Re::Vfs::File>
00354 Vfs::get_root() noexcept
00355 {
00356     return cxx::ref_ptr(&_root);
00357 }
00358
00359 Ref_ptr<L4Re::Vfs::File>
00360 Vfs::get_cwd() noexcept
00361 {
00362     return _cwd;
00363 }
00364
00365 void
00366 Vfs::set_cwd(Ref_ptr<L4Re::Vfs::File> const &dir) noexcept
00367 {
00368     // FIXME: check for is dir
00369     if (dir)
00370         _cwd = dir;
00371 }
00372
00373 Ref_ptr<L4Re::Vfs::File>
00374 Vfs::get_file(int fd) noexcept
00375 {
00376     return fds.get(fd);
00377 }
00378
00379 cxx::Pair<Ref_ptr<L4Re::Vfs::File>, int>
00380 Vfs::set_fd(int fd, Ref_ptr<L4Re::Vfs::File> const &f) noexcept
00381 {
00382     if (!fds.check_fd(fd))
00383         return cxx::pair(Ref_ptr<L4Re::Vfs::File>(Ref_ptr<>::Nil), EBADF);
00384
00385     Ref_ptr<L4Re::Vfs::File> old = fds.get(fd);
00386     fds.set(fd, f);
00387     return cxx::pair(old, 0);
00388 }
00389
00390
00391 #define GET_FILE_DBG(fd, err) \
00392     Ref_ptr<L4Re::Vfs::File> fi = fds.get(fd); \
00393     if (!fi) \
00394     { \
00395         return -err; \
00396     }
00397
00398 #define GET_FILE(fd, err) \
00399     Ref_ptr<L4Re::Vfs::File> fi = fds.get(fd); \
00400     if (!fi) \
00401         return -err;
00402
00403 void
00404 Vfs::align_mmap_start_and_length(void **start, size_t *length)
00405 {
00406     l4_addr_t const s = reinterpret_cast<l4_addr_t>(*start);
00407     size_t const o = s & (L4_PAGESIZE - 1);
00408
00409     *start = reinterpret_cast<void*>(l4_trunc_page(s));
00410     *length = l4_round_page(*length + o);
00411 }
00412
00413 int
00414 Vfs::munmap_regions(void *start, size_t len)
00415 {
00416     using namespace L4;
00417     using namespace L4Re;
00418
00419     int err;
00420     Cap<Dataspace> ds;
00421     Cap<Rm> r = Env::env()->rm();
00422
00423     if (l4_addr_t(start) & (L4_PAGESIZE - 1))
00424         return -EINVAL;
00425
00426     align_mmap_start_and_length(&start, &len);
00427
00428     while (1)
00429     {
00430         DEBUG_LOG(debug_mmap, {
00431             l4_kd_outstring("DETACH: start = 0x");
00432             l4_kd_outhex32(l4_addr_t(start));
00433             l4_kd_outstring(" len = 0x");
00434             l4_kd_outhex32(len);
00435             l4_kd_outstring("\n");
00436         });
00437         err = r->detach(l4_addr_t(start), len, &ds, This_task);

```

```

00438     if (err < 0)
00439         return err;
00440
00441     switch (err & Rm::Detach_result_mask)
00442     {
00443     case Rm::Split_ds:
00444         if (ds.is_valid())
00445             L4Re::virt_cap_alloc->take(ds);
00446         return 0;
00447     case Rm::Detached_ds:
00448         if (ds.is_valid())
00449             L4Re::virt_cap_alloc->release(ds);
00450         break;
00451     default:
00452         break;
00453     }
00454
00455     if (!(err & Rm::Detach_again))
00456         return 0;
00457 }
00458 }
00459
00460 int
00461 Vfs::munmap(void *start, size_t len) L4_NOTHROW
00462 {
00463     using namespace L4;
00464     using namespace L4Re;
00465
00466     int err = 0;
00467     Cap<Rm> r = Env::env()->rm();
00468
00469     // Fields for obtaining a list of areas for the calling process
00470     long area_cnt = -1;           // No. of areas in this process
00471     Rm::Area const *area_array;
00472     bool matches_area = false;    // true if unmap parameters match an area
00473
00474     // First check if there are any areas matching the munmap request. Those
00475     // might have been created by an mmap call using PROT_NONE as protection
00476     // modifier.
00477
00478     area_cnt = r->get_areas((l4_addr_t) start, &area_array);
00479
00480     // It is enough to check for the very first entry, since get_areas will
00481     // only return areas with a starting address equal or greater to <start>.
00482     // However, we intend to unmap at most the area starting exactly at
00483     // <start>.
00484     if (area_cnt > 0)
00485     {
00486         size_t area_size = area_array[0].end - area_array[0].start + 1;
00487
00488         // Only free the area if the munmap parameters describe it exactly.
00489         if (area_array[0].start == (l4_addr_t) start && area_size == len)
00490         {
00491             r->free_area((l4_addr_t) start);
00492             matches_area = true;
00493         }
00494     }
00495
00496     // After clearing possible area reservations from PROT_NONE mappings, clear
00497     // any regions in the address range specified. Note that errors shall be
00498     // suppressed if an area was freed but no regions were found.
00499     err = munmap_regions(start, len);
00500     if (err == -ENOENT && matches_area)
00501         return 0;
00502
00503     return err;
00504 }
00505
00506 int
00507 Vfs::alloc_ds(unsigned long size, L4Re::Shared_cap<L4Re::Dataspace> *ds)
00508 {
00509     *ds = L4Re::make_shared_cap<L4Re::Dataspace>(L4Re::virt_cap_alloc);
00510
00511     if (!ds->is_valid())
00512         return -ENOMEM;
00513
00514     int err;
00515     if ((err = Vfs_config::allocator()->alloc(size, ds->get())) < 0)
00516         return err;
00517
00518     DEBUG_LOG(debug_mmap, {
00519         l4_kd_outstring("ANON DS ALLOCATED: size=");
00520         l4_kd_outhex32(size);
00521         l4_kd_outstring(" cap = 0x");
00522         l4_kd_outhex32(ds->cap());
00523         l4_kd_outstring("\n");
00524     });

```



```

00525
00526     return 0;
00527 }
00528
00529 int
00530 Vfs::alloc_anon_mem(l4_umword_t size, L4Re::Shared_cap<L4Re::Dataspace> *ds,
00531                    l4_addr_t *offset)
00532 {
00533     #if !defined(CONFIG_MMU)
00534         // Small values for !MMU systems. These platforms do not have much memory
00535         // typically and the memory must be instantly allocated.
00536         enum
00537         {
00538             ANON_MEM_DS_POOL_SIZE = 256UL < 10, // size of a pool dataspace used for anon memory
00539             ANON_MEM_MAX_SIZE      = 32UL < 10, // chunk size that will be allocate a dataspace
00540         };
00541     #elif defined(USE_BIG_ANON_DS)
00542         enum
00543         {
00544             ANON_MEM_DS_POOL_SIZE = 256UL < 20, // size of a pool dataspace used for anon memory
00545             ANON_MEM_MAX_SIZE      = 32UL < 20, // chunk size that will be allocate a dataspace
00546         };
00547     #else
00548         enum
00549         {
00550             ANON_MEM_DS_POOL_SIZE = 256UL < 20, // size of a pool dataspace used for anon memory
00551             ANON_MEM_MAX_SIZE      = 0UL < 20,  // chunk size that will be allocate a dataspace
00552         };
00553     #endif
00554
00555     if (size >= ANON_MEM_MAX_SIZE)
00556     {
00557         int err;
00558         if ((err = alloc_ds(size, ds)) < 0)
00559             return err;
00560
00561         *offset = 0;
00562
00563         if (!_early_oom)
00564             return err;
00565
00566         return (*ds)->allocate(0, size);
00567     }
00568
00569     if (!_anon_ds.is_valid() || _anon_offset + size >= ANON_MEM_DS_POOL_SIZE)
00570     {
00571         int err;
00572         if ((err = alloc_ds(ANON_MEM_DS_POOL_SIZE, ds)) < 0)
00573             return err;
00574
00575         _anon_offset = 0;
00576         _anon_ds = *ds;
00577     }
00578     else
00579         *ds = _anon_ds;
00580
00581     if (_early_oom)
00582     {
00583         if (int err = (*ds)->allocate(_anon_offset, size))
00584             return err;
00585     }
00586
00587     *offset = _anon_offset;
00588     _anon_offset += size;
00589     return 0;
00590 }
00591
00592 int
00593 Vfs::mmap2(void *start, size_t len, int prot, int flags, int fd, off_t page4k_offset,
00594            void **resp_ptr) L4_NOTHROW
00595 {
00596     DEBUG_LOG(debug_mmap, {
00597         l4_kd_outstring("MMAP params: ");
00598         l4_kd_outstring("start = 0x");
00599         l4_kd_outhex32(l4_addr_t(start));
00600         l4_kd_outstring(", len = 0x");
00601         l4_kd_outhex32(len);
00602         l4_kd_outstring(", prot = 0x");
00603         l4_kd_outhex32(prot);
00604         l4_kd_outstring(", flags = 0x");
00605         l4_kd_outhex32(flags);
00606         l4_kd_outstring(", offset = 0x");
00607         l4_kd_outhex32(page4k_offset);
00608         l4_kd_outstring("\n");
00609     });
00610
00611     using namespace L4Re;

```

```

00612     off64_t offset = l4_trunc_page(page4k_offset « 12);
00613
00614     if (flags & MAP_FIXED)
00615         if (l4_addr_t(start) & (L4_PAGESIZE - 1))
00616             return -EINVAL;
00617
00618     align_mmap_start_and_length(&start, &len);
00619
00620     // special code to just reserve an area of the virtual address space
00621     // Same behavior should be exposed when mapping with PROT_NONE. Mind that
00622     // PROT_NONE can only be specified exclusively, since it is defined to 0x0.
00623     if ((flags & 0x1000000) || (prot == PROT_NONE))
00624     {
00625         int err;
00626         L4::Cap<Rm> r = Env::env()->rm();
00627         l4_addr_t area = reinterpret_cast<l4_addr_t>(start);
00628         err = r->reserve_area(&area, len, L4Re::Rm::F::Search_addr);
00629         if (err < 0)
00630             return err;
00631
00632         *resp_ptr = reinterpret_cast<void*>(area);
00633
00634         DEBUG_LOG(debug_mmap, {
00635             l4_kd_outstring("  MMAP reserved area: 0x");
00636             l4_kd_outhex32(area);
00637             l4_kd_outstring("  length= 0x");
00638             l4_kd_outhex32(len);
00639             l4_kd_outstring("\n");
00640         });
00641
00642         return 0;
00643     }
00644
00645     L4Re::Shared_cap<L4Re::Dataspace> ds;
00646     l4_addr_t anon_offset = 0;
00647     L4Re::Rm::Flags rm_flags(0);
00648
00649     if (flags & (MAP_ANONYMOUS | MAP_PRIVATE))
00650     {
00651         rm_flags |= L4Re::Rm::F::Detach_free;
00652
00653         int err = alloc_anon_mem(len, &ds, &anon_offset);
00654         if (err)
00655             return err;
00656
00657         DEBUG_LOG(debug_mmap, {
00658             l4_kd_outstring("  USE ANON MEM: 0x");
00659             l4_kd_outhex32(ds.cap());
00660             l4_kd_outstring("  offs = 0x");
00661             l4_kd_outhex32(anon_offset);
00662             l4_kd_outstring("\n");
00663         });
00664     }
00665
00666     char const *region_name = "[unknown]";
00667     l4_addr_t file_offset = 0;
00668     if (!(flags & MAP_ANONYMOUS))
00669     {
00670         Ref_ptr<L4Re::Vfs::File> fi = fds.get(fd);
00671         if (!fi)
00672             return -EBADF;
00673
00674         region_name = fi->path();
00675
00676         L4::Cap<L4Re::Dataspace> fds = fi->data_space();
00677
00678         if (!fds.is_valid())
00679             return -EINVAL;
00680
00681         if (len + offset > l4_round_page(fds->size()))
00682             return -EINVAL;
00683
00684         if (flags & MAP_PRIVATE)
00685         {
00686             DEBUG_LOG(debug_mmap, l4_kd_outstring("COW\n"));
00687             int err = ds->copy_in(anon_offset, fds, offset, len);
00688             file_offset = offset;
00689             if (err == -L4_EINVAL)
00690             {
00691                 L4::Cap<Rm> r = Env::env()->rm();
00692                 Rm::Unique_region<char*> src;
00693                 Rm::Unique_region<char*> dst;
00694                 err = r->attach(&src, len,
00695                     L4Re::Rm::F::Search_addr | L4Re::Rm::F::R,
00696                     fds, offset);
00697                 if (err < 0)
00698                     return err;

```

```

00699
00700         err = r->attach(&dst, len,
00701                        L4Re::Rm::F::Search_addr | L4Re::Rm::F::RW,
00702                        ds.get(), anon_offset);
00703         if (err < 0)
00704             return err;
00705
00706         memcpy(dst.get(), src.get(), len);
00707
00708         region_name = "[mmap-private]";
00709         file_offset = (unsigned long)dst.get();
00710     }
00711     else if (err)
00712         return err;
00713
00714     offset = anon_offset;
00715 }
00716 else
00717 {
00718     L4Re::virt_cap_alloc->take(fds);
00719     ds = L4Re::Shared_cap<L4Re::Dataspace>(fds, L4Re::virt_cap_alloc);
00720 }
00721 }
00722 else
00723 {
00724     offset = anon_offset;
00725     region_name = "[anon]";
00726     file_offset = offset;
00727 }
00728
00729
00730 if (!(flags & MAP_FIXED) && start == 0)
00731     start = reinterpret_cast<void*>(L4_PAGESIZE);
00732
00733 char *data = static_cast<char *>(start);
00734 L4::Cap<Rm> r = Env::env()->rm();
00735 l4_addr_t overmap_area = L4_INVALID_ADDR;
00736
00737 int err;
00738 if (flags & MAP_FIXED)
00739 {
00740     overmap_area = l4_addr_t(start);
00741
00742     err = r->reserve_area(&overmap_area, len);
00743     if (err < 0)
00744         overmap_area = L4_INVALID_ADDR;
00745
00746     rm_flags |= Rm::F::In_area;
00747
00748     // Make sure to remove old mappings residing at the respective address
00749     // range. If none exists, we are fine as well, allowing us to ignore
00750     // ENOENT here.
00751     err = munmap_regions(start, len);
00752     if (err && err != -ENOENT)
00753         return err;
00754 }
00755
00756 if (!(flags & MAP_FIXED))
00757     rm_flags |= Rm::F::Search_addr;
00758 if (prot & PROT_READ)
00759     rm_flags |= Rm::F::R;
00760 if (prot & PROT_WRITE)
00761     rm_flags |= Rm::F::W;
00762 if (prot & PROT_EXEC)
00763     rm_flags |= Rm::F::X;
00764
00765 err = r->attach(&data, len, rm_flags,
00766               L4::Ipc::make_cap(ds.get(), (prot & PROT_WRITE)
00767                                ? L4_CAP_FPAGE_RW
00768                                : L4_CAP_FPAGE_RO),
00769               offset, L4_PAGESHIFT, L4::Cap<L4::Task>::Invalid,
00770               region_name, file_offset);
00771
00772 DEBUG_LOG(debug_mmap, {
00773     l4_kd_outstring("  MAPPED: 0x");
00774     l4_kd_outhex32(ds.cap());
00775     l4_kd_outstring("  addr: 0x");
00776     l4_kd_outhex32(l4_addr_t(data));
00777     l4_kd_outstring("  bytes: 0x");
00778     l4_kd_outhex32(len);
00779     l4_kd_outstring("  offset: 0x");
00780     l4_kd_outhex32(offset);
00781     l4_kd_outstring("  err = ");
00782     l4_kd_outdec(err);
00783     l4_kd_outstring("\n");
00784 });
00785

```

```

00786
00787     if (overmap_area != L4_INVALID_ADDR)
00788         r->free_area(overmap_area);
00789
00790     if (err < 0)
00791         return err;
00792
00793     l4_assert (!(start && !data));
00794
00795     // release ownership of the attached DS
00796     ds.release();
00797     *resptr = data;
00798
00799     return 0;
00800 }
00801
00802 namespace {
00803     class Auto_area
00804     {
00805     public:
00806         L4::Cap<L4Re::Rm> r;
00807         l4_addr_t a;
00808
00809         explicit Auto_area(L4::Cap<L4Re::Rm> r, l4_addr_t a = L4_INVALID_ADDR)
00810             : r(r), a(a) {}
00811
00812         int reserve(l4_addr_t _a, l4_size_t sz, L4Re::Rm::Flags flags)
00813         {
00814             free();
00815             a = _a;
00816             int e = r->reserve_area(&a, sz, flags);
00817             if (e)
00818                 a = L4_INVALID_ADDR;
00819             return e;
00820         }
00821
00822         void free()
00823         {
00824             if (is_valid())
00825             {
00826                 r->free_area(a);
00827                 a = L4_INVALID_ADDR;
00828             }
00829         }
00830
00831         bool is_valid() const { return a != L4_INVALID_ADDR; }
00832
00833         ~Auto_area() { free(); }
00834     };
00835 }
00836
00837 int
00838 Vfs::mremap(void *old_addr, size_t old_size, size_t new_size, int flags,
00839             void **new_addr) L4_NOTHROW
00840 {
00841     using namespace L4Re;
00842
00843     DEBUG_LOG(debug_mmap, {
00844         l4_kd_outstring("Mremap: addr = 0x");
00845         l4_kd_outhex32((l4_umword_t)old_addr);
00846         l4_kd_outstring(" old_size = 0x");
00847         l4_kd_outhex32(old_size);
00848         l4_kd_outstring(" new_size = 0x");
00849         l4_kd_outhex32(new_size);
00850         l4_kd_outstring("\n");
00851     });
00852
00853     if (flags & MREMAP_FIXED && !(flags & MREMAP_MAYMOVE))
00854         return -EINVAL;
00855
00856     l4_addr_t oa = l4_trunc_page(reinterpret_cast<l4_addr_t>(old_addr));
00857     if (oa != reinterpret_cast<l4_addr_t>(old_addr))
00858         return -EINVAL;
00859
00860     bool const fixed = flags & MREMAP_FIXED;
00861     bool const maymove = flags & MREMAP_MAYMOVE;
00862
00863     L4::Cap<Rm> r = Env::env()->rm();
00864
00865     // sanitize input parameters to multiples of pages
00866     old_size = l4_round_page(old_size);
00867     new_size = l4_round_page(new_size);
00868
00869     if (!fixed)
00870     {
00871         if (new_size < old_size)
00872             {

```

```

00873         *new_addr = old_addr;
00874         return munmap(reinterpret_cast<void*>(oa + new_size),
00875                        old_size - new_size);
00876     }
00877
00878     if (new_size == old_size)
00879     {
00880         *new_addr = old_addr;
00881         return 0;
00882     }
00883 }
00884
00885 Auto_area old_area(r);
00886 int err = old_area.reserve(oa, old_size, L4Re::Rm::Flags(0));
00887 if (err < 0)
00888     return -EINVAL;
00889
00890 l4_addr_t pad_addr;
00891 Auto_area new_area(r);
00892 if (fixed)
00893 {
00894     l4_addr_t na = l4_trunc_page(reinterpret_cast<l4_addr_t>(*new_addr));
00895     if (na != reinterpret_cast<l4_addr_t>(*new_addr))
00896         return -EINVAL;
00897
00898     // check if the current virtual memory area can be expanded
00899     int err = new_area.reserve(na, new_size, L4Re::Rm::Flags(0));
00900     if (err < 0)
00901         return err;
00902
00903     pad_addr = na;
00904     // unmap all stuff and remap ours ....
00905 }
00906 else
00907 {
00908     l4_addr_t ta = oa + old_size;
00909     unsigned long ts = new_size - old_size;
00910     // check if the current virtual memory area can be expanded
00911     long err = new_area.reserve(ta, ts, L4Re::Rm::Flags(0));
00912     if (!maymove && err)
00913         return -ENOMEM;
00914
00915     L4Re::Rm::Offset toffs;
00916     L4Re::Rm::Flags tflags;
00917     L4::Cap<L4Re::Dataspace> tds;
00918
00919     err = r->find(&ta, &ts, &toffs, &tflags, &tds);
00920
00921     // there is enough space to expand the mapping in place
00922     if (err == -ENOENT || (err == 0 && (tflags & Rm::F::In_area)))
00923     {
00924         old_area.free(); // pad at the original address
00925         pad_addr = oa + old_size;
00926         *new_addr = old_addr;
00927     }
00928     else if (!maymove)
00929         return -ENOMEM;
00930     else
00931     {
00932         // search for a new area to remap
00933         err = new_area.reserve(0, new_size, Rm::F::Search_addr);
00934         if (err < 0)
00935             return -ENOMEM;
00936
00937         pad_addr = new_area.a + old_size;
00938         *new_addr = reinterpret_cast<void *>(new_area.a);
00939     }
00940 }
00941
00942 if (old_area.is_valid())
00943 {
00944     unsigned long size = old_size;
00945
00946     l4_addr_t a = old_area.a;
00947     unsigned long s = 1;
00948     L4Re::Rm::Offset o;
00949     L4Re::Rm::Flags f;
00950     L4::Cap<L4Re::Dataspace> ds;
00951
00952     while (r->find(&a, &s, &o, &f, &ds) >= 0 && !(f & Rm::F::In_area))
00953     {
00954         if (a < old_area.a)
00955         {
00956             auto d = old_area.a - a;
00957             a = old_area.a;
00958             s -= d;
00959             o += d;

```

```

00960     }
00961
00962     if (a + s > old_area.a + old_size)
00963         s = old_area.a + old_size - a;
00964
00965     l4_addr_t x = a - old_area.a + new_area.a;
00966
00967     int err = r->attach(&x, s, Rm::F::In_area | f,
00968                       L4::Ipc::make_cap(ds, f.cap_rights(), o));
00969     if (err < 0)
00970         return err;
00971
00972     // count the new attached ds reference
00973     L4Re::virt_cap_alloc->take(ds);
00974
00975     err = r->detach(a, s, &ds, This_task,
00976                   Rm::Detach_exact | Rm::Detach_keep);
00977     if (err < 0)
00978         return err;
00979
00980     switch (err & Rm::Detach_result_mask)
00981     {
00982     case Rm::Split_ds:
00983         // add a reference as we split up a mapping
00984         if (ds.is_valid())
00985             L4Re::virt_cap_alloc->take(ds);
00986         break;
00987     case Rm::Detached_ds:
00988         if (ds.is_valid())
00989             L4Re::virt_cap_alloc->release(ds);
00990         break;
00991     default:
00992         break;
00993     }
00994
00995     if (size <= s)
00996         break;
00997     a += s;
00998     size -= s;
00999     s = 1;
01000 }
01001
01002 old_area.free();
01003 }
01004
01005 if (old_size < new_size)
01006 {
01007     l4_addr_t const pad_sz = new_size - old_size;
01008     l4_addr_t toffs;
01009     L4Re::Shared_cap<L4Re::Dataspace> tds;
01010     int err = alloc_anon_mem(pad_sz, &tds, &toffs);
01011     if (err)
01012         return err;
01013
01014     // FIXME: must get the protection rights from the old
01015     // mapping and use the same here, for now just use RWX
01016     err = r->attach(&pad_addr, pad_sz,
01017                   Rm::F::In_area | Rm::F::Detach_free | Rm::F::RWX,
01018                   L4::Ipc::make_cap_rw(tds.get()), toffs);
01019     if (err < 0)
01020         return err;
01021
01022     // release ownership of tds, the region map is now the new owner
01023     tds.release();
01024 }
01025
01026 return 0;
01027 }
01028
01029 int
01030 Vfs::mprotect(const void * /* a */, size_t /* sz */, int prot) L4_NOTHROW
01031 {
01032     return (prot & PROT_WRITE) ? -ENOSYS : 0;
01033 }
01034
01035 int
01036 Vfs::msync(void *, size_t, int) L4_NOTHROW
01037 { return 0; }
01038
01039 int
01040 Vfs::madvise(void *, size_t, int) L4_NOTHROW
01041 { return 0; }
01042
01043 }
01044
01045 L4Re::Vfs::Ops *_rtld_l4re_env_posix_vfs_ops;
01046 extern void *l4re_env_posix_vfs_ops __attribute__((alias("__rtld_l4re_env_posix_vfs_ops")));

```

```

        visibility("default")));
01047
01048 namespace {
01049     class Real_mount_tree : public L4Re::Vfs::Mount_tree
01050     {
01051     public:
01052         explicit Real_mount_tree(char *n) : Mount_tree(n) {}
01053
01054         void *operator new (size_t size)
01055         { return __rtld_l4re_env_posix_vfs_ops->malloc(size); }
01056
01057         void operator delete (void *mem)
01058         { __rtld_l4re_env_posix_vfs_ops->free(mem); }
01059     };
01060 }
01061
01062 int
01063 Vfs::mount(char const *path, cxx::Ref_ptr<L4Re::Vfs::File> const &dir) noexcept
01064 {
01065     using L4Re::Vfs::File;
01066     using L4Re::Vfs::Mount_tree;
01067     using L4Re::Vfs::Path;
01068
01069     cxx::Ref_ptr<Mount_tree> root = get_root()->mount_tree();
01070     if (!root)
01071         return -EINVAL;
01072
01073     cxx::Ref_ptr<Mount_tree> base;
01074     Path p = root->lookup(Path(path), &base);
01075
01076     while (!p.empty())
01077     {
01078         Path f = p.strip_first();
01079
01080         if (f.empty())
01081             return -EEXIST;
01082
01083         char *name = __rtld_l4re_env_posix_vfs_ops->strndup(f.path(), f.length());
01084         if (!name)
01085             return -ENOMEM;
01086
01087         auto nt = cxx::make_ref_obj<Real_mount_tree>(name);
01088         if (!nt)
01089         {
01090             __rtld_l4re_env_posix_vfs_ops->free(name);
01091             return -ENOMEM;
01092         }
01093
01094         base->add_child_node(nt);
01095         base = nt;
01096
01097         if (p.empty())
01098         {
01099             nt->mount(dir);
01100             return 0;
01101         }
01102     }
01103
01104     return -EINVAL;
01105 }
01106
01107 #undef DEBUG_LOG
01108 #undef GET_FILE_DBG
01109 #undef GET_FILE

```

16.265 vfs.h

```

00001 /*
00002  * (c) 2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include <14/sys/compiler.h>
00011
00012 #include <unistd.h>
00013 #include <stdarg.h>
00014 #include <fcntl.h>
00015 #include <sys/stat.h>
00016 #include <sys/time.h>

```

```

00017 #include <sys/mman.h>
00018 #include <sys/socket.h>
00019 #include <utime.h>
00020 #include <errno.h>
00021
00022 #ifndef AT_FDCWD
00023 # define AT_FDCWD -100
00024 #endif
00025
00026 #ifdef __cplusplus
00027
00028 #include <l4/sys/capability>
00029 #include <l4/re/cap_alloc>
00030 #include <l4/re/dataspace>
00031 #include <l4/cxx/pair>
00032 #include <l4/cxx/ref_ptr>
00033
00034 namespace L4Re {
00038 namespace Vfs {
00039
00040 class Mount_tree;
00041 class File;
00042
00052 class Generic_file
00053 {
00054 public:
00060 enum Ready_type : unsigned
00061 {
00062     Read = 0,
00063     Write,
00064     Exception
00065 };
00066
00067 virtual ~Generic_file() noexcept = 0;
00068
00080 virtual int unlock_all_locks() noexcept = 0;
00081
00090 virtual int fstat(struct stat64 *buf) const noexcept = 0;
00091
00097 virtual int fchmod(mode_t) noexcept = 0;
00098
00108 virtual int get_status_flags() const noexcept = 0;
00109
00125 virtual int set_status_flags(long flags) noexcept = 0;
00126
00127 virtual int utime(const struct utimbuf *) noexcept = 0;
00128 virtual int utimes(const struct timeval [2]) noexcept = 0;
00129 virtual ssize_t readlink(char *, size_t) = 0;
00130
00146 virtual bool check_ready(Ready_type rt) noexcept = 0;
00147 };
00148
00149 inline
00150 Generic_file::~Generic_file() noexcept
00151 {}
00152
00160 class Directory
00161 {
00162 public:
00163     virtual ~Directory() noexcept = 0;
00164
00178 virtual int faccessat(const char *path, int mode, int flags) noexcept = 0;
00179
00192 virtual int mkdir(const char *path, mode_t mode) noexcept = 0;
00193
00204 virtual int unlink(const char *path) noexcept = 0;
00205
00219 virtual int rename(const char *src_path, const char *dst_path) noexcept = 0;
00220
00234 virtual int link(const char *src_path, const char *dst_path) noexcept = 0;
00235
00248 virtual int symlink(const char *src_path, const char *dst_path) noexcept = 0;
00249
00260 virtual int rmdir(const char *path) noexcept = 0;
00261 virtual int openat(const char *path, int flags, mode_t mode,
00262                  cxx::Ref_ptr<File> *f) noexcept = 0;
00263
00264 virtual ssize_t getdents(char *buf, size_t sizebytes) noexcept = 0;
00265
00266 virtual int fchmodat(const char *pathname,
00267                     mode_t mode, int flags) noexcept = 0;
00268
00269 virtual int utimensat(const char *pathname,
00270                      const struct timespec times[2], int flags) noexcept = 0;
00271
00275 virtual int get_entry(const char *, int, mode_t, cxx::Ref_ptr<File> *) noexcept = 0;
00276 };

```



```

00277
00278 inline
00279 Directory::~Directory() noexcept
00280 {}
00281
00287 class Regular_file
00288 {
00289 public:
00290     virtual ~Regular_file() noexcept = 0;
00291
00302     virtual L4::Cap<L4Re::Dataspace> data_space() noexcept = 0;
00303
00313     virtual ssize_t readv(const struct iovec*, int iovcnt) noexcept = 0;
00314
00325     virtual ssize_t writev(const struct iovec*, int iovcnt) noexcept = 0;
00326
00327     virtual ssize_t preadv(const struct iovec *iov, int iovcnt, off64_t offset) noexcept = 0;
00328     virtual ssize_t pwritev(const struct iovec *iov, int iovcnt, off64_t offset) noexcept = 0;
00329
00337     virtual off64_t lseek(off64_t, int) noexcept = 0;
00338
00339
00347     virtual int ftruncate(off64_t pos) noexcept = 0;
00348
00354     virtual int fsync() const noexcept = 0;
00355
00361     virtual int fdatsync() const noexcept = 0;
00362
00372     virtual int get_lock(struct flock64 *lock) noexcept = 0;
00373
00382     virtual int set_lock(struct flock64 *lock, bool wait) noexcept = 0;
00383 };
00384
00385 inline
00386 Regular_file::~Regular_file() noexcept
00387 {}
00388
00389 class Socket
00390 {
00391 public:
00392     virtual ~Socket() noexcept = 0;
00393     virtual int bind(sockaddr const *, socklen_t) noexcept = 0;
00394     virtual int connect(sockaddr const *, socklen_t) noexcept = 0;
00395     virtual ssize_t send(void const *, size_t, int) noexcept = 0;
00396     virtual ssize_t recv(void *, size_t, int) noexcept = 0;
00397     virtual ssize_t sendto(void const *, size_t, int, sockaddr const *, socklen_t) noexcept = 0;
00398     virtual ssize_t recvfrom(void *, size_t, int, sockaddr *, socklen_t *) noexcept = 0;
00399     virtual ssize_t sendmsg(msghdr const *, int) noexcept = 0;
00400     virtual ssize_t recvmsg(msghdr *, int) noexcept = 0;
00401     virtual int getsockopt(int level, int opt, void *, socklen_t *) noexcept = 0;
00402     virtual int setsockopt(int level, int opt, void const *, socklen_t) noexcept = 0;
00403     virtual int listen(int) noexcept = 0;
00404     virtual int accept(sockaddr *addr, socklen_t *) noexcept = 0;
00405     virtual int shutdown(int) noexcept = 0;
00406
00407     virtual int getsockname(sockaddr *, socklen_t *) noexcept = 0;
00408     virtual int getpeername(sockaddr *, socklen_t *) noexcept = 0;
00409 };
00410
00411 inline
00412 Socket::~Socket() noexcept
00413 {}
00414
00420 class Special_file
00421 {
00422 public:
00423     virtual ~Special_file() noexcept = 0;
00424
00435     virtual int ioctl(unsigned long cmd, va_list args) noexcept = 0;
00436 };
00437
00438 inline
00439 Special_file::~Special_file() noexcept
00440 {}
00441
00455 class File :
00456     public Generic_file,
00457     public Regular_file,
00458     public Directory,
00459     public Special_file,
00460     public Socket
00461 {
00462     friend class Mount_tree;
00463
00464 private:
00465     void operator = (File const &);
00466

```

```

00467 protected:
00468     File() noexcept : _ref_cnt(0) {}
00469     File(File const &)
00470     : Generic_file(), Regular_file(), Directory(), Special_file(), _ref_cnt(0)
00471     {}
00472
00473 public:
00474
00475     const char *get_mount(const char *path, cxx::Ref_ptr<File> *dir,
00476                           cxx::Ref_ptr<Mount_tree> *mt = 0) noexcept;
00477
00478     int openat(const char *path, int flags, mode_t mode,
00479               cxx::Ref_ptr<File> *f) noexcept override;
00480
00481     void add_ref() noexcept { ++_ref_cnt; }
00482     int remove_ref() noexcept { return --_ref_cnt; }
00483
00484     virtual ~File() noexcept = 0;
00485
00486     cxx::Ref_ptr<Mount_tree> mount_tree() const noexcept
00487     { return _mount_tree; }
00488
00489     char const *path() const noexcept { return _path; }
00490
00491 private:
00492     int _ref_cnt;
00493     cxx::Ref_ptr<Mount_tree> _mount_tree;
00494     char _path[80] = "";
00495 };
00496
00497 inline
00498 File::~File() noexcept
00499 {}
00500
00501 class Path
00502 {
00503 private:
00504     char const *_p;
00505     unsigned _l;
00506
00507 public:
00508     Path() noexcept : _p(0), _l(0) {}
00509
00510     explicit Path(char const *p) noexcept : _p(p)
00511     { for (_l = 0; *p; ++p, ++_l) ; }
00512
00513     Path(char const *p, unsigned l) noexcept : _p(p), _l(l)
00514     {}
00515
00516     static bool __is_sep(char s) noexcept;
00517
00518     Path cmp_path(char const *prefix) const noexcept;
00519
00520     struct Invalid_ptr;
00521     operator Invalid_ptr const * () const
00522     { return reinterpret_cast<Invalid_ptr const *>(_p); }
00523
00524     unsigned length() const { return _l; }
00525     char const *path() const { return _p; }
00526
00527     bool empty() const { return _l == 0; }
00528
00529     bool is_sep(unsigned offset) const { return __is_sep(_p[offset]); }
00530
00531     bool strip_sep()
00532     {
00533         bool s = false;
00534         for (; __is_sep(*_p) && _l; ++_p, --_l)
00535             s = true;
00536         return s;
00537     }
00538
00539     Path first() const
00540     {
00541         unsigned i;
00542         for (i = 0; i < _l && !is_sep(i); ++i)
00543             ;
00544         return Path(_p, i);
00545     }
00546
00547     Path strip_first()
00548     {
00549         Path r = first();
00550         _p += r.length();
00551         _l -= r.length();
00552         strip_sep();
00553     }

```

```

00554     return r;
00555 }
00556
00557 };
00558
00559
00560 class Mount_tree
00561 {
00562 public:
00563     explicit Mount_tree(char *n) noexcept;
00564
00565     Path lookup(Path const &path, cxx::Ref_ptr<Mount_tree> *mt,
00566                 cxx::Ref_ptr<Mount_tree> *mp = 0) noexcept;
00567
00568     Path find(Path const &p, cxx::Ref_ptr<Mount_tree> *t) noexcept;
00569
00570     cxx::Ref_ptr<File> mount() const
00571     { return _mount; }
00572
00573     void mount(cxx::Ref_ptr<File> const &m)
00574     {
00575         m->_mount_tree = cxx::ref_ptr(this);
00576         _mount = m;
00577     }
00578
00579     static int create_tree(cxx::Ref_ptr<Mount_tree> const &root,
00580                             char const *path,
00581                             cxx::Ref_ptr<File> const &dir) noexcept;
00582
00583     void add_child_node(cxx::Ref_ptr<Mount_tree> const &cld);
00584
00585     virtual ~Mount_tree() noexcept = 0;
00586
00587     void add_ref() noexcept { ++_ref_cnt; }
00588     int remove_ref() noexcept { return --_ref_cnt; }
00589
00590 private:
00591     friend class Real_mount_tree;
00592
00593     int _ref_cnt;
00594     char *_name;
00595     cxx::Ref_ptr<Mount_tree> _cld;
00596     cxx::Ref_ptr<Mount_tree> _sib;
00597     cxx::Ref_ptr<File> _mount;
00598 };
00599
00600 inline
00601 Mount_tree::~Mount_tree() noexcept
00602 {}
00603
00604 inline bool
00605 Path::__is_sep(char s) noexcept
00606 { return s == '/'; }
00607
00608 inline Path
00609 Path::cmp_path(char const *n) const noexcept
00610 {
00611     char const *p = _p;
00612     for (; *p && !__is_sep(*p) && *n; ++p, ++n)
00613         if (*p != *n)
00614             return Path();
00615     if (*n || (*p && !__is_sep(*p)))
00616         return Path();
00617     return Path(p, _l - (p - _p));
00618 }
00619
00620 inline
00621 Mount_tree::Mount_tree(char *n) noexcept
00622 : _ref_cnt(0), _name(n)
00623 {}
00624
00625 inline Path
00626 Mount_tree::find(Path const &p, cxx::Ref_ptr<Mount_tree> *t) noexcept
00627 {
00628     if (!_cld)
00629         return Path();
00630     for (cxx::Ref_ptr<Mount_tree> x = _cld; x; x = x->_sib)
00631     {
00632         Path const r = p.cmp_path(x->_name);
00633         if (r)
00634         {
00635             *t = x;
00636             return r;
00637         }
00638     }
00639 }

```

```

00647     }
00648     }
00649
00650     return Path();
00651 }
00652
00653 inline Path
00654 Mount_tree::lookup(Path const &path, cxx::Ref_ptr<Mount_tree> *mt,
00655                    cxx::Ref_ptr<Mount_tree> *mp) noexcept
00656 {
00657     cxx::Ref_ptr<Mount_tree> x(this);
00658     Path p = path;
00659
00660     if (p.first().cmp_path("."))
00661         p.strip_first();
00662
00663     Path last_mp = p;
00664
00665     if (mp)
00666         *mp = x;;
00667
00668     while (1)
00669     {
00670         Path r = x->find(p, &x);
00671
00672         if (!r)
00673         {
00674             if (mp)
00675                 return last_mp;
00676
00677             if (mt)
00678                 *mt = x;
00679
00680             return p;
00681         }
00682
00683         r.strip_sep();
00684
00685         if (mp && x->_mount)
00686         {
00687             last_mp = r;
00688             *mp = x;
00689         }
00690
00691         if (r.empty())
00692         {
00693             if (mt)
00694                 *mt = x;
00695
00696             if (mp)
00697                 return last_mp;
00698             else
00699                 return r;
00700         }
00701
00702         p = r;
00703     }
00704 }
00705
00706 inline
00707 void
00708 Mount_tree::add_child_node(cxx::Ref_ptr<Mount_tree> const &cld)
00709 {
00710     cld->_sib = _cld;
00711     _cld = cld;
00712 }
00713
00714 inline
00715 const char *
00716 File::get_mount(const char *path, cxx::Ref_ptr<File> *dir,
00717                cxx::Ref_ptr<Mount_tree> *mt) noexcept
00718 {
00719     if (!_mount_tree)
00720     {
00721         *dir = cxx::ref_ptr(this);
00722         return path;
00723     }
00724
00725     cxx::Ref_ptr<Mount_tree> mp;
00726     Path p = _mount_tree->lookup(Path(path), mt, &mp);
00727     if (mp->mount())
00728     {
00729         *dir = mp->mount();
00730         return p.path();
00731     }
00732     else
00733     {

```

```

00734     *dir = cxx::ref_ptr(this);
00735     return path;
00736 }
00737 }
00738
00739 inline int
00740 File::openat(const char *path, int flags, mode_t mode,
00741             cxx::Ref_ptr<File> *f) noexcept
00742 {
00743     cxx::Ref_ptr<File> dir;
00744     cxx::Ref_ptr<Mount_tree> mt;
00745     char const *m_path = get_mount(path, &dir, &mt);
00746
00747     int res = dir->get_entry(m_path, flags, mode, f);
00748
00749     if (res < 0)
00750         return res;
00751
00752     if (!(*f)->_mount_tree && mt)
00753         (*f)->_mount_tree = mt;
00754
00755     // Debugging {
00756     size_t i = 0;
00757     for (; i < sizeof((*f)->_path) - 1 && path[i]; ++i)
00758         (*f)->_path[i] = path[i];
00759     (*f)->_path[i] = '\0';
00760     // } Debugging
00761
00762     return res;
00763 }
00764
00773 class Mman
00774 {
00775 public:
00777     virtual int mmap2(void *start, size_t len, int prot, int flags, int fd,
00778                     off_t offset, void **ptr) noexcept = 0;
00779
00781     virtual int munmap(void *start, size_t len) noexcept = 0;
00782
00784     virtual int mremap(void *old, size_t old_sz, size_t new_sz, int flags,
00785                     void **new_addr) noexcept = 0;
00786
00788     virtual int mprotect(const void *a, size_t sz, int prot) noexcept = 0;
00789
00791     virtual int msync(void *addr, size_t len, int flags) noexcept = 0;
00792
00794     virtual int madvise(void *addr, size_t len, int advice) noexcept = 0;
00795
00796     virtual ~Mman() noexcept = 0;
00797 };
00798
00799 inline
00800 Mman::~Mman() noexcept {}
00801
00802 class File_factory
00803 {
00804 private:
00805     int _ref_cnt = 0;
00806     int _proto = 0;
00807     char const *_proto_name = 0;
00808
00809     template<typename T> friend struct cxx::Default_ref_counter;
00810     void add_ref() noexcept { ++_ref_cnt; }
00811     int remove_ref() noexcept { return --_ref_cnt; }
00812
00813 public:
00814     explicit File_factory(int proto) : _proto(proto) {}
00815     explicit File_factory(char const *proto_name) : _proto_name(proto_name) {}
00816     File_factory(int proto, char const *proto_name)
00817         : _proto(proto), _proto_name(proto_name)
00818     {}
00819
00820     File_factory(File_factory const &) = delete;
00821     File_factory &operator = (File_factory const &) = delete;
00822
00823     char const *proto_name() const { return _proto_name; }
00824     int proto() const { return _proto; }
00825
00826     virtual ~File_factory() noexcept = 0;
00827     virtual cxx::Ref_ptr<File> create(L4::Cap<void> file) = 0;
00828 };
00829
00830 inline File_factory::~File_factory() noexcept {}
00831
00832 template<typename IFACE, typename IMPL>
00833 class File_factory_t : public File_factory
00834 {

```

```

00835 public:
00836     File_factory_t()
00837     : File_factory(IFACE::Protocol, L4::kobject_typeid<IFACE>()->name())
00838     {}
00839
00840     cxx::Ref_ptr<File> create(L4::Cap<void> file) override
00841     { return cxx::make_ref_obj<IMPL>(L4::cap_cast<IFACE>(file)); }
00842 };
00843
00857 class File_system
00858 {
00859 protected:
00860     File_system *_next;
00861
00862 public:
00863     File_system() noexcept : _next(0) {}
00869     virtual char const *type() const noexcept = 0;
00870
00887     virtual int mount(char const *source, unsigned long mountflags,
00888                      void const *data, cxx::Ref_ptr<File> *dir) noexcept = 0;
00889
00890     virtual ~File_system() noexcept = 0;
00891
00896     File_system *next() const noexcept { return _next; }
00897     File_system *&next() noexcept { return _next; }
00898     void next(File_system *n) noexcept { _next = n; }
00899 };
00900
00901 inline
00902 File_system::~File_system() noexcept
00903 {}
00904
00905 class File_system_list
00906 {
00907 public:
00908     class Iterator
00909     {
00910     public:
00911         explicit constexpr Iterator(File_system *c = nullptr) : _c(c) {}
00912
00913         Iterator &operator++()
00914         {
00915             if (_c)
00916                 _c = _c->next();
00917             return *this;
00918         }
00919
00920         bool operator==(Iterator const &other) const { return _c == other._c; }
00921         bool operator!=(Iterator const &other) const { return _c != other._c; }
00922         File_system *operator*() const { return _c; }
00923         File_system *operator->() const { return _c; }
00924
00925     private:
00926         File_system *_c;
00927     };
00928
00929     File_system_list(File_system *head) : _head(head) {}
00930
00931     constexpr Iterator begin() const
00932     { return Iterator(_head); }
00933
00934     constexpr Iterator end() const
00935     { return Iterator(); }
00936
00937 private:
00938     File_system *_head;
00939 };
00940
00946 class Fs
00947 {
00948 public:
00954     virtual cxx::Ref_ptr<File> get_file(int fd) noexcept = 0;
00955
00957     virtual cxx::Ref_ptr<File> get_root() noexcept = 0;
00958
00960     virtual cxx::Ref_ptr<File> get_cwd() noexcept { return get_root(); }
00961
00963     virtual void set_cwd(cxx::Ref_ptr<File> const &) noexcept {}
00964
00970     virtual int alloc_fd(cxx::Ref_ptr<File> const &f = cxx::Ref_ptr<>::Nil) noexcept = 0;
00971
00982     virtual cxx::Pair<cxx::Ref_ptr<File>, int>
00983         set_fd(int fd, cxx::Ref_ptr<File> const &f = cxx::Ref_ptr<>::Nil) noexcept = 0;
00984
00990     virtual cxx::Ref_ptr<File> free_fd(int fd) noexcept = 0;
00991
00999     virtual int mount(char const *path, cxx::Ref_ptr<File> const &dir) noexcept = 0;

```

```

01000
01008     virtual int register_file_system(File_system *f) noexcept = 0;
01009
01017     virtual int unregister_file_system(File_system *f) noexcept = 0;
01018
01026     virtual File_system *get_file_system(char const *fstype) noexcept = 0;
01027
01036     virtual File_system_list file_system_list() noexcept = 0;
01037
01041     int mount(char const *source, char const *target,
01042               char const *fstype, unsigned long mountflags,
01043               void const *data) noexcept;
01044
01045     virtual int register_file_factory(cxx::Ref_ptr<File_factory> f) noexcept = 0;
01046     virtual int unregister_file_factory(cxx::Ref_ptr<File_factory> f) noexcept = 0;
01047     virtual cxx::Ref_ptr<File_factory> get_file_factory(int proto) noexcept = 0;
01048     virtual cxx::Ref_ptr<File_factory> get_file_factory(char const *proto_name) noexcept = 0;
01049
01050     virtual ~Fs() = 0;
01051
01052 private:
01053     int mount_one(char const *source, char const *target,
01054                   File_system *fs, unsigned long mountflags,
01055                   void const *data) noexcept;
01056 };
01057
01058 inline int
01059 Fs::mount_one(char const *source, char const *target,
01060               File_system *fs, unsigned long mountflags,
01061               void const *data) noexcept
01062 {
01063     cxx::Ref_ptr<File> dir;
01064     int res = fs->mount(source, mountflags, data, &dir);
01065
01066     if (res < 0)
01067         return res;
01068
01069     return mount(target, dir);
01070 }
01071
01072 inline int
01073 Fs::mount(char const *source, char const *target,
01074           char const *fstype, unsigned long mountflags,
01075           void const *data) noexcept
01076 {
01077     if (fstype[0] == 'a'
01078         && fstype[1] == 'u'
01079         && fstype[2] == 't'
01080         && fstype[3] == 'o'
01081         && fstype[4] == 0)
01082     {
01083         File_system_list fsl = file_system_list();
01084         for (File_system_list::Iterator c = fsl.begin(); c != fsl.end(); ++c)
01085             if (mount_one(source, target, *c, mountflags, data) == 0)
01086                 return 0;
01087
01088         return -ENODEV;
01089     }
01090
01091     File_system *fs = get_file_system(fstype);
01092
01093     if (!fs)
01094         return -ENODEV;
01095
01096     return mount_one(source, target, fs, mountflags, data);
01097 }
01098
01099 inline
01100 Fs::~Fs()
01101 {}
01102
01109 class Ops : public Mman, public Fs
01110 {
01111 public:
01112     virtual void *malloc(size_t bytes) noexcept = 0;
01113     virtual void free(void *mem) noexcept = 0;
01114     virtual ~Ops() noexcept = 0;
01115
01116     char *strndup(char const *str, unsigned l) noexcept
01117     {
01118         unsigned len;
01119         for (len = 0; str[len] && len < l; ++len)
01120             ;
01121
01122         if (len == 0)
01123             return nullptr;
01124     }

```

```

01125     ++len;
01126
01127     char *b = static_cast<char *>(this->malloc(len));
01128     if (b == nullptr)
01129         return nullptr;
01130
01131     char *r = b;
01132     for (; len - 1 > 0 && *str; --len, ++b, ++str)
01133         *b = *str;
01134
01135     *b = 0;
01136     return r;
01137 }
01138
01139 };
01140
01141 inline
01142 Ops::~Ops() noexcept
01143 {}
01144
01145 }}
01146
01147 #endif
01148

```

16.266 virtio-net

```

00001 // vi:ft=cpp
00002 /* SPDX-License-Identifier: MIT */
00003 /*
00004  * Copyright (C) 2022, 2024-2025 Kernkonzept GmbH.
00005  * Author(s): Stephan Gerhold <stephan.gerhold@kernkonzept.com>
00006  */
00007 #pragma once
00008
00009 #include <cstring>
00010 #include <functional>
00011
00012 #include <l4/cxx/exceptions>
00013 #include <l4/cxx/minmax>
00014 #include <l4/re/dataspace>
00015 #include <l4/re/env>
00016 #include <l4/re/error_helper>
00017 #include <l4/re/util/unique_cap>
00018 #include <l4/sys/consts.h>
00019
00020 #include <l4/l4virtio/client/l4virtio>
00021 #include <l4/l4virtio/l4virtio>
00022 #include <l4/l4virtio/virtio_net.h>
00023 #include <l4/l4virtio/virtqueue>
00024
00025 namespace L4virtio { namespace Driver {
00026
00030 class Virtio_net_device : public L4virtio::Driver::Device
00031 {
00032 public:
00037     struct Packet
00038     {
00039         l4virtio_net_header_t hdr;
00040         l4_uint8_t data[1500 + 14]; /* MTU + Ethernet header */
00041     };
00042
00047     int rx_queue_size() const
00048     { return max_queue_size(0); }
00049
00054     int tx_queue_size() const
00055     { return max_queue_size(1); }
00056
00070     void setup_device(L4::Cap<L4virtio::Device> srvcap,
00071                     L4::Cap<L4Re::Dataspace>
00072                     queue_ds = L4::Cap<L4Re::Dataspace>::Invalid)
00073     {
00074         // Contact device.
00075         driver_connect(srvcap, false);
00076
00077         if (_config->device != L4VIRTIO_ID_NET)
00078             L4Re::chksys(-L4_ENODEV, "Device is not a network device.");
00079
00080         if (_config->num_queues < 2)
00081             L4Re::chksys(-L4_EINVAL, "Invalid number of queues reported.");
00082
00083         auto rxqsz = rx_queue_size();
00084         auto txqsz = tx_queue_size();

```



```

00085
00086 // Allocate memory for RX/TX queue and RX/TX packet buffers
00087 auto rxqoff = 0;
00088 auto txqoff = l4_round_size(rxqoff + rxqsz * _rxq.total_size(rxqsz),
00089                             L4virtio::Virtqueue::Desc_align);
00090 auto rxpktoff = l4_round_size(txqoff + txqsz * _txq.total_size(txqsz),
00091                               L4virtio::Virtqueue::Desc_align);
00092 auto txpktoff = rxpktoff + rxqsz * sizeof(Packet);
00093 auto totalsz = txpktoff + txqsz * sizeof(Packet);
00094
00095 auto *e = L4Re::Env::env();
00096
00097 if (queue_ds)
00098     _queue_ds = L4Re::Util::Unique_cap<L4Re::Dataspace>(queue_ds);
00099 else
00100 {
00101     _queue_ds = L4Re::chkcap(L4Re::Util::make_unique_cap<L4Re::Dataspace>(),
00102                             "Allocate queue dataspace capability");
00103     L4Re::chksys(e->mem_alloc()->alloc(totalsz, _queue_ds.get(),
00104                                         L4Re::Mem_alloc::Continuous
00105                                         | L4Re::Mem_alloc::Pinned),
00106                 "Allocate memory for virtio structures");
00107 }
00108
00109 L4Re::chksys(e->rm()->attach(&_queue_region, totalsz,
00110                             L4Re::Rm::F::Search_addr | L4Re::Rm::F::RW,
00111                             L4::Ipc::make_cap_rw(_queue_ds.get(), 0,
00112                             L4_PAGESHIFT),
00113                             "Attach dataspace for virtio structures");
00114
00115 l4_uint64_t devaddr;
00116 L4Re::chksys(register_ds(_queue_ds.get(), 0, totalsz, &devaddr),
00117               "Register queue dataspace with device");
00118
00119 _rxq.init_queue(rxqsz, _queue_region.get() + rxqoff);
00120 _txq.init_queue(txqsz, _queue_region.get() + txqoff);
00121
00122 config_queue(0, rxqsz, devaddr + rxqoff,
00123              devaddr + rxqoff + _rxq.avail_offset(),
00124              devaddr + rxqoff + _rxq.used_offset());
00125 config_queue(1, txqsz, devaddr + txqoff,
00126              devaddr + txqoff + _txq.avail_offset(),
00127              devaddr + txqoff + _txq.used_offset());
00128
00129 _rxpkts = reinterpret_cast<Packet*>(_queue_region.get() + rxpktoff);
00130 _txpkts = reinterpret_cast<Packet*>(_queue_region.get() + txpktoff);
00131
00132 // Prepare descriptors to save work later
00133 for (l4_uint16_t descno = 0; descno < rxqsz; ++descno)
00134 {
00135     auto &desc = _rxq.desc(descno);
00136     desc.addr = L4virtio::Ptr<void>(devaddr + rxpktoff +
00137                                     descno * sizeof(Packet));
00138     desc.len = sizeof(Packet);
00139     desc.flags.write() = 1;
00140 }
00141 for (l4_uint16_t descno = 0; descno < txqsz; ++descno)
00142 {
00143     auto &desc = _txq.desc(descno);
00144     desc.addr = L4virtio::Ptr<void>(devaddr + txpktoff +
00145                                     descno * sizeof(Packet));
00146     desc.len = sizeof(Packet);
00147 }
00148
00149 // Setup notification IRQ
00150 _driver_notification_irq =
00151     L4Re::chkcap(L4Re::Util::make_unique_cap<L4::Irq>(),
00152                 "Allocate notification capability");
00153
00154 L4Re::chksys(l4_error(e->factory()->create(_driver_notification_irq.get()),
00155                                             "Create irq for notifications from device");
00156
00157 L4Re::chksys(_device->bind(0, _driver_notification_irq.get()),
00158             "Bind driver notification interrupt");
00159
00160 // Finish handshake with device
00161 l4virtio_set_feature(_config->driver_features_map,
00162                     L4VIRTIO_FEATURE_VERSION_1);
00163 l4virtio_set_feature(_config->driver_features_map, L4VIRTIO_NET_F_MAC);
00164 driver_acknowledge();
00165 }
00166
00170 l4virtio_net_config_t const &device_config() const
00171 {
00172     return *_config->device_config<l4virtio_net_config_t>();
00173 }
00174

```

```

00181 int bind_rx_notification_irq(L4::Cap<L4::Thread> thread, l4_umword_t label)
00182 {
00183     return l4_error(_driver_notification_irq->bind_thread(thread, label));
00184 }
00185
00192 Packet &rx_pkt(l4_uint16_t descno)
00193 {
00194     if (descno >= _rxq.num())
00195         throw L4::Bounds_error("Invalid used descriptor number in RX queue");
00196     return _rxpkts[descno];
00197 }
00198
00214 l4_uint16_t wait_rx(l4_uint32_t *len = nullptr)
00215 {
00216     l4_uint16_t descno;
00217     // Wait until used descriptor becomes available.
00218     for (;;)
00219     {
00220         descno = _rxq.find_next_used(len);
00221         if (descno != Virtqueue::Eoq)
00222             break;
00223
00224         L4Re::chksys(_driver_notification_irq->receive(), "Wait for RX");
00225     }
00226
00227     if (len)
00228         // Ensure that the length provided by the device in wait_for_next_used()
00229         // is not larger than the buffer and subtract the length of the header.
00230         *len = cxx::min(*len - sizeof(_rxpkts[0].hdr), sizeof(_rxpkts[0].data));
00231     return descno;
00232 }
00233
00242 void finish_rx(l4_uint16_t descno)
00243 {
00244     _rxq.free_descriptor(descno, descno);
00245 }
00246
00250 void queue_rx()
00251 {
00252     l4_uint16_t descno;
00253     while ((descno = _rxq.alloc_descriptor()) != Virtqueue::Eoq)
00254         _rxq.enqueue_descriptor(descno);
00255     notify(_rxq);
00256 }
00257
00272 bool tx(std::function<l4_uint32_t(Packet&)> prepare)
00273 {
00274     auto descno = _txq.alloc_descriptor();
00275     if (descno == Virtqueue::Eoq)
00276     {
00277         // Try again after cleaning old descriptors that have already been used
00278         free_used_tx_descriptors();
00279         descno = _txq.alloc_descriptor();
00280         if (descno == Virtqueue::Eoq)
00281             return false;
00282     }
00283
00284     auto &pkt = _txpkts[descno];
00285     auto &desc = _txq.desc(descno);
00286     desc.len = sizeof(pkt.hdr) + prepare(pkt);
00287     send(_txq, descno);
00288     return true;
00289 }
00290
00291 private:
00292 void free_used_tx_descriptors()
00293 {
00294     l4_uint16_t used;
00295     while ((used = _txq.find_next_used()) != Virtqueue::Eoq)
00296     {
00297         if (used >= _txq.num())
00298             throw L4::Bounds_error("Invalid used descriptor number in TX queue");
00299         _txq.free_descriptor(used, used);
00300     }
00301 }
00302
00303 private:
00304 L4Re::Util::Unique_cap<L4Re::Dataspace> _queue_ds;
00305 L4Re::Rm::Unique_region<l4_uint8_t *> _queue_region;
00306 L4Re::Util::Unique_cap<L4::Irq> _driver_notification_irq;
00307 L4virtio::Driver::Virtqueue _rxq, _txq;
00308 Packet *_rxpkts, *_txpkts;
00309 };
00310
00311 } }

```

16.267 l4virtio

```

00001 // vi:ft=cpp
00002 /* SPDX-License-Identifier: MIT */
00003 /*
00004  * Copyright (C) 2015-2020, 2022, 2024 Kernkonzept GmbH.
00005  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00006  *
00007  */
00008 #pragma once
00009
00010 #include <l4/sys/factory>
00011 #include <l4/sys/semaphore>
00012 #include <l4/re/dataspace>
00013 #include <l4/re/env>
00014 #include <l4/re/util/unique_cap>
00015 #include <l4/re/util/object_registry>
00016 #include <l4/re/error_helper>
00017
00018 #include <l4/util/atomic.h>
00019 #include <l4/util/bitops.h>
00020 #include <l4/l4virtio/l4virtio>
00021 #include <l4/l4virtio/virtqueue>
00022 #include <l4/sys/consts.h>
00023
00024 #include <cstring>
00025
00026 namespace L4virtio { namespace Driver {
00027
00031 class Device
00032 {
00033 public:
00056 void driver_connect(L4::Cap<L4virtio::Device> srvcap, bool manage_notify = true)
00057 {
00058     _device = srvcap;
00059
00060     _next_devaddr = L4_SUPERPAGESIZE;
00061
00062     auto *e = L4Re::Env::env();
00063
00064     // Set up the virtio configuration page.
00065
00066     _config_cap = L4Re::chkcap(L4Re::Util::make_unique_cap<L4Re::Dataspace>(),
00067                                "Allocate config dataspace capability");
00068
00069     l4_addr_t ds_offset;
00070     L4Re::chksys(_device->device_config(_config_cap.get(), &ds_offset),
00071                  "Request virtio config page");
00072
00073     if (ds_offset & ~L4_PAGEMASK)
00074         L4Re::chksys(-L4_EINVAL, "Virtio config page is page aligned.");
00075
00076     L4Re::chksys(e->rm()->attach(&_config, L4_PAGESIZE,
00077                                L4Re::Rm::F::Search_addr | L4Re::Rm::F::RW,
00078                                L4::Ipc::make_cap_rw(_config_cap.get(), ds_offset,
00079                                L4_PAGESHIFT),
00080                                "Attach config dataspace");
00081
00082     if (memcmp(&_config->magic, "virt", 4) != 0)
00083         L4Re::chksys(-L4_ENODEV, "Device config has wrong magic value");
00084
00085     if (_config->version != 2)
00086         L4Re::chksys(-L4_ENODEV, "Invalid virtio version, must be 2");
00087
00088     _device->set_status(0); // reset
00089     int status = L4VIRTIO_STATUS_ACKNOWLEDGE;
00090     _device->set_status(status);
00091
00092     status |= L4VIRTIO_STATUS_DRIVER;
00093     _device->set_status(status);
00094
00095     if (_config->fail_state())
00096         L4Re::chksys(-L4_EIO, "Device failure during initialisation.");
00097
00098     // Set up the interrupt used to notify the device about events.
00099     // (only supporting one interrupt with index 0 at the moment)
00100
00101     _host_irq = L4Re::chkcap(L4Re::Util::make_unique_cap<L4::Irq>(),
00102                              "Allocate host IRQ capability");
00103
00104     L4Re::chksys(_device->device_notification_irq(0, _host_irq.get()),
00105                  "Request device notification interrupt.");
00106
00107     // Set up the interrupt to get notifications from the device.
00108     // (only supporting one interrupt with index 0 at the moment)
00109     if (manage_notify)

```

```

00110     {
00111         _driver_notification =
00112             L4Re::chkcap(L4Re::Util::make_unique_cap<L4::Semaphore>(),
00113                 "Allocate notification capability");
00114
00115         L4Re::chksys(l4_error(e->factory()->create(_driver_notification.get()),
00116             "Create semaphore for notifications from device");
00117
00118         L4Re::chksys(_device->bind(0, _driver_notification.get()),
00119             "Bind driver notification interrupt");
00120     }
00121 }
00122
00129 int bind_notification_irq(unsigned index, L4::Cap<L4::Triggerable> irq) const
00130 { return l4_error(_device->bind(index, irq)); }
00131
00133 bool fail_state() const { return _config->fail_state(); }
00134
00145 bool feature_negotiated(unsigned int feat) const
00146 { return l4virtio_get_feature(_config->driver_features_map, feat); }
00147
00156 int driver_acknowledge()
00157 {
00158     if (!l4virtio_get_feature(_config->dev_features_map,
00159         L4VIRTIO_FEATURE_VERSION_1))
00160         L4Re::chksys(-L4_ENODEV,
00161             "Require Virtio 1.0 device; Legacy device not supported.");
00162
00163     _config->driver_features_map[0] &= _config->dev_features_map[0];
00164     _config->driver_features_map[1] &= _config->dev_features_map[1];
00165
00166     _device->set_status(_config->status | L4VIRTIO_STATUS_FEATURES_OK);
00167
00168     if (!(_config->status & L4VIRTIO_STATUS_FEATURES_OK))
00169         L4Re::chksys(-L4_EINVAL, "Negotiation of device features.");
00170
00171     _device->set_status(_config->status | L4VIRTIO_STATUS_DRIVER_OK);
00172
00173     if (_config->fail_state())
00174         return -L4_EIO;
00175
00176     return L4_EOK;
00177 }
00178
00196 int register_ds(L4::Cap<L4Re::Dataspace> ds, l4_umword_t offset,
00197     l4_umword_t size, l4_uint64_t *devaddr)
00198 {
00199     *devaddr = next_device_address(size);
00200     return _device->register_ds(L4::Ipc::make_cap_rw(ds), *devaddr, offset, size);
00201 }
00202
00212 int config_queue(int num, unsigned size, l4_uint64_t desc_addr,
00213     l4_uint64_t avail_addr, l4_uint64_t used_addr)
00214 {
00215     auto *queueconf = &_config->queues()[num];
00216     queueconf->num = size;
00217     queueconf->desc_addr = desc_addr;
00218     queueconf->avail_addr = avail_addr;
00219     queueconf->used_addr = used_addr;
00220     queueconf->ready = 1;
00221
00222     return _device->config_queue(num);
00223 }
00224
00230 int max_queue_size(int num) const
00231 {
00232     return _config->queues()[num].num_max;
00233 }
00234
00247 int send_and_wait(Virtqueue &queue, l4_uint16_t descno)
00248 {
00249     send(queue, descno);
00250
00251     // wait for a reply, we assume that no other
00252     // request will get in the way.
00253     auto head = wait_for_next_used(queue);
00254
00255     if (head < 0)
00256         return head;
00257
00258     return (head == descno) ? L4_EOK : -L4_EINVAL;
00259 }
00260
00268 int wait(int index) const
00269 {
00270     if (index != 0)
00271         return -L4_EEXIST;

```

```

00272
00273     return l4_ipc_error(_driver_notification->down(), l4_utcb());
00274 }
00275
00291 int wait_for_next_used(Virtqueue &queue, l4_uint32_t *len = nullptr) const
00292 {
00293     while (true)
00294     {
00295         int err = wait(0);
00296
00297         if (err < 0)
00298             return err;
00299
00300         auto head = queue.find_next_used(len);
00301         if (head != Virtqueue::Eoq) // spurious interrupt?
00302             return head;
00303     }
00304 }
00305
00312 void send(Virtqueue &queue, l4_uint16_t descno)
00313 {
00314     queue.enqueue_descriptor(descno);
00315     notify(queue);
00316 }
00317
00318 void notify(Virtqueue &queue)
00319 {
00320     if (!queue.no_notify_host())
00321         _host_irq->trigger();
00322 }
00323
00324 private:
00335 l4_uint64_t next_device_address(l4_umword_t size)
00336 {
00337     l4_umword_t ret;
00338     size = l4_round_page(size);
00339     do
00340     {
00341         ret = _next_devaddr;
00342         if (l4_umword_t(~0) - ret < size)
00343             L4Re::chksys(-L4_ENOMEM, "Out of device address space.");
00344     }
00345     while (!l4util_cmpxchg(&_next_devaddr, ret, ret + size));
00346
00347     return ret;
00348 }
00349
00350 protected:
00351     L4::Cap<L4virtio::Device> _device;
00352     L4Re::Rm::Unique_region<L4virtio::Device::Config_hdr *> _config;
00353     l4_umword_t _next_devaddr;
00354     L4Re::Util::Unique_cap<L4::Semaphore> _driver_notification;
00355
00356 private:
00357     L4Re::Util::Unique_cap<L4::Irq> _host_irq;
00358     L4Re::Util::Unique_cap<L4Re::Dataspace> _config_cap;
00359 };
00360
00361 } }

```

16.268 l4virtio

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /* SPDX-License-Identifier: MIT */
00003 /*
00004  * Copyright (C) 2013-2024 Kernkonzept GmbH.
00005  * Author(s): Alexander Warg <alexander.warg@kernkonzept.com>
00006  *             Matthias Lange <matthias.lange@kernkonzept.com>
00007  *
00008  */
00009 #pragma once
00010
00011 #include "virtio.h"
00012 #include <l4/sys/capability>
00013 #include <l4/sys/cxx/ipc_client>
00014 #include <l4/re/dataspace>
00015 #include <l4/sys/irq>
00016 #include <l4/cxx/utls>
00017
00018 namespace L4virtio {
00039 class Device :
00040     public L4::Kobject_t<Device, L4::Icu, L4VIRTIO_PROTOCOL,
00041         L4::Type_info::Demand_t<1> >

```

```

00042 {
00043 public:
00044 typedef l4virtio_config_queue_t Config_queue;
00045 struct Config_hdr : l4virtio_config_hdr_t
00046 {
00047     Config_queue *queues() const
00048     { return l4virtio_config_queues(this); }
00049
00050     template <typename T>
00051     T *device_config() const
00052     {
00053         return static_cast<T*>(l4virtio_device_config(this));
00054     }
00055
00056     int config_queue(unsigned num, L4::Cap<L4::Triggerable> out_notify,
00057                     L4::Cap<L4::Triggerable> in_notify,
00058                     l4_timeout_s to = L4_IPC_TIMEOUT_NEVER)
00059     {
00060         return send_cmd(L4VIRTIO_CMD_CFG_QUEUE | num,
00061                         out_notify, in_notify, to);
00062     }
00063
00064     int notify_queue(unsigned num, L4::Cap<L4::Triggerable> out_notify,
00065                     L4::Cap<L4::Triggerable> in_notify,
00066                     l4_timeout_s to = L4_IPC_TIMEOUT_NEVER)
00067     {
00068         return send_cmd(L4VIRTIO_CMD_NOTIFY_QUEUE | num,
00069                         out_notify, in_notify, to);
00070     }
00071
00072     bool fail_state() const
00073     {
00074         auto cfg_status = cxx::access_once(&status);
00075         return cfg_status
00076             & (L4VIRTIO_STATUS_FAILED | L4VIRTIO_STATUS_DEVICE_NEEDS_RESET);
00077     }
00078
00079     int set_status(unsigned new_status, L4::Cap<L4::Triggerable> out_notify,
00080                  L4::Cap<L4::Triggerable> in_notify,
00081                  l4_timeout_s to = L4_IPC_TIMEOUT_NEVER)
00082     {
00083         return send_cmd(L4VIRTIO_CMD_SET_STATUS | new_status,
00084                         out_notify, in_notify, to);
00085     }
00086
00087     int cfg_changed(unsigned reg, L4::Cap<L4::Triggerable> out_notify,
00088                    L4::Cap<L4::Triggerable> in_notify,
00089                    l4_timeout_s to = L4_IPC_TIMEOUT_NEVER)
00090     {
00091         return send_cmd(L4VIRTIO_CMD_CFG_CHANGED | reg,
00092                         out_notify, in_notify, to);
00093     }
00094
00095     int send_cmd(unsigned command, L4::Cap<L4::Triggerable> out_notify,
00096                 L4::Cap<L4::Triggerable> in_notify,
00097                 l4_timeout_s to = L4_IPC_TIMEOUT_NEVER)
00098     {
00099         cxx::write_now(&cmd, command);
00100
00101         if (out_notify)
00102             out_notify->trigger();
00103
00104         auto utcb = l4_utcb();
00105         auto ipc_to = l4_timeout(L4_IPC_TIMEOUT_0, to);
00106
00107         do
00108         {
00109             if (in_notify)
00110                 if (l4_ipc_error(l4_ipc_receive(in_notify.cap(), utcb, ipc_to),
00111                                 utcb) == L4_IPC_RETIMEOUT)
00112                     break;
00113         }
00114         while (cxx::access_once(&cmd));
00115
00116         return cxx::access_once(&cmd) ? -L4_EBUSY : L4_EOK;
00117     }
00118 };
00119
00120 L4_INLINE_RPC_OP(L4VIRTIO_OP_SET_STATUS, long,
00121                 set_status, (unsigned status));
00122
00123 L4_INLINE_RPC_OP(L4VIRTIO_OP_CONFIG_QUEUE, long,
00124                 config_queue, (unsigned queue));
00125
00126 L4_INLINE_RPC_OP(L4VIRTIO_OP_REGISTER_DS, long,
00127                 register_ds, (L4::Ipc::Cap<L4Re::Dataspace> ds_cap,
00128                               l4_uint64_t base, l4_umword_t offset,

```

```

00184         l4_umword_t size));
00185
00194     L4_INLINE_RPC_OP(L4VIRTIO_OP_DEVICE_CONFIG, long, device_config,
00195                     (L4::Ipc::Out<L4::Cap<L4Re::Dataspace> > config_ds,
00196                      l4_addr_t *ds_offset));
00197
00216     L4_INLINE_RPC_OP(L4VIRTIO_OP_GET_DEVICE_IRQ, long, device_notification_irq,
00217                     (unsigned index, L4::Ipc::Out<L4::Cap<L4::Triggerable> > irq));
00218
00219
00220     typedef L4::Typeid::Rpc<set_status_t, config_queue_t, register_ds_t,
00221                             device_config_t, device_notification_irq_t>
00222         Rpc;
00223 };
00224
00225 }

```

16.269 l4virtio

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /* SPDX-License-Identifier: MIT */
00003 /*
00004  * Copyright (C) 2014-2025 Kernkonzept GmbH.
00005  * Author(s): Alexander Warg <alexander.warg@kernkonzept.com>
00006  *            Manuel von Oltersdorff-Kalettka <manuel.kalettka@kernkonzept.com>
00007  *
00008  */
00009 #pragma once
00010
00011 #include <algorithm>
00012 #include <limits.h>
00013 #include <memory>
00014 #include <vector>
00015
00016 #include <l4/re/dataspace>
00017 #include <l4/re/util/debug>
00018 #include <l4/re/env>
00019 #include <l4/re/error_helper>
00020 #include <l4/re/rm>
00021 #include <l4/re/util/cap_alloc>
00022 #include <l4/re/util/shared_cap>
00023 #include <l4/re/util/unique_cap>
00024
00025 #include <l4/sys/types.h>
00026 #include <l4/re/util/meta>
00027
00028 #include <l4/cxx/bitfield>
00029 #include <l4/cxx/utls>
00030 #include <l4/cxx/unique_ptr>
00031
00032 #include <l4/sys/cxx/ipc_legacy>
00033
00034 #include "../l4virtio"
00035 #include "virtio"
00036
00040 namespace L4virtio {
00041 namespace Svr {
00042
00052 class Dev_config
00053 {
00054 public:
00055     typedef Dev_status Status;
00056     typedef Dev_features Features;
00057
00058 private:
00059     typedef L4Re::Rm::Unique_region< l4virtio_config_hdr_t*> Cfg_region;
00060     typedef L4Re::Util::Shared_cap<L4Re::Dataspace> Cfg_cap;
00061
00062     l4_uint32_t _vendor, _device, _qoffset, _nqueues;
00063     l4_uint32_t _host_features[sizeof(l4virtio_config_hdr_t::dev_features_map)
00064                                   / sizeof(l4_uint32_t)];
00065     Cfg_cap _ds;
00066     Cfg_region _config;
00067     l4_addr_t _ds_offset = 0;
00068
00069     Status _status{0}; // status shadow, can be trusted by the device model
00070
00071     static l4_uint32_t align(l4_uint32_t x)
00072     { return (x + 0xfU) & ~0xfU; }
00073
00074     void attach_n_init_cfg(Cfg_cap const &cfg, l4_addr_t offset)
00075     {
00076         L4Re::chksys(L4Re::Env::env()->rm()->attach(&_config, L4_PAGESIZE,

```

```

00077                                     L4Re::Rm::F::Search_addr | L4Re::Rm::F::RW,
00078                                     L4::Ipc::make_cap_rw(cfg.get()),
00079                                     offset),
00080     "Attach config space to local address space.");
00081
00082     _config->generation = 0;
00083     memset(_config->driver_features_map, 0, sizeof(_config->driver_features_map));
00084     memset(_host_features, 0, sizeof(_host_features));
00085     set_host_feature(L4VIRTIO_FEATURE_VERSION_1);
00086     reset_hdr();
00087
00088     _ds = cfg;
00089     _ds_offset = offset;
00090 }
00091
00092 protected:
00093     void volatile *get_priv_config() const
00094     {
00095         return l4virtio_device_config(_config.get());
00096     }
00097
00098 public:
00099
00100     Dev_config(l4_uint32_t vendor, l4_uint32_t device,
00101               unsigned cfg_size, l4_uint32_t num_queues = 0)
00102     : _vendor(vendor), _device(device),
00103       _qoffset(0x100 + align(cfg_size)),
00104       _nqueues(num_queues)
00105     {
00106         using L4Re::Dataspace;
00107         using L4Re::chkcapp;
00108         using L4Re::chksys;
00109
00110         if (sizeof(l4virtio_config_queue_t) * _nqueues + _qoffset > L4_PAGESIZE)
00111         {
00112             // too many queues does not fit into our page
00113             _qoffset = 0;
00114             _nqueues = 0;
00115         }
00116
00117         auto cfg = chkcapp(L4Re::Util::make_shared_cap<Dataspace>());
00118         chksys(L4Re::Env::env()->mem_alloc()->alloc(L4_PAGESIZE, cfg.get()));
00119
00120         attach_n_init_cfg(cfg, 0);
00121     }
00122
00123     Dev_config(Cfg_cap const &cfg, l4_addr_t cfg_offset,
00124               l4_uint32_t vendor, l4_uint32_t device,
00125               unsigned cfg_size, l4_uint32_t num_queues = 0)
00126     : _vendor(vendor), _device(device),
00127       _qoffset(0x100 + align(cfg_size)),
00128       _nqueues(num_queues)
00129     {
00130         if (sizeof(l4virtio_config_queue_t) * _nqueues + _qoffset > L4_PAGESIZE)
00131         {
00132             // too many queues does not fit into our page
00133             _qoffset = 0;
00134             _nqueues = 0;
00135         }
00136
00137         attach_n_init_cfg(cfg, cfg_offset);
00138     }
00139
00140     void set_host_feature(unsigned feature)
00141     { l4virtio_set_feature(_host_features, feature); }
00142
00143     void clear_host_feature(unsigned feature)
00144     { l4virtio_clear_feature(_host_features, feature); }
00145
00146     bool get_host_feature(unsigned feature)
00147     { return l4virtio_get_feature(_host_features, feature); }
00148
00149     bool get_guest_feature(unsigned feature)
00150     { return l4virtio_get_feature(_config->driver_features_map, feature); }
00151
00152     l4_uint32_t &host_features(unsigned idx)
00153     { return _host_features[idx]; }
00154
00155     l4_uint32_t host_features(unsigned idx) const
00156     { return _host_features[idx]; }
00157
00158     void set_device_notify_index(unsigned idx)
00159     { _config->cfg_device_notify_index = idx; }
00160
00161     l4_uint32_t num_queues() const
00162     { return _nqueues; }
00163

```



```

00207  l4_uint32_t guest_features(unsigned idx) const
00208  { return _config->driver_features_map[idx]; }
00209
00221  l4_uint32_t negotiated_features(unsigned idx) const
00222  { return _config->driver_features_map[idx] & _host_features[idx]; }
00223
00231  Status status() const { return _status; }
00232
00239  l4_uint32_t get_cmd() const
00240  {
00241      return hdr()->cmd;
00242  }
00243
00250  void reset_cmd()
00251  {
00252      const_cast<l4_uint32_t volatile &>(hdr()->cmd) = 0;
00253  }
00254
00262  void set_status(Status status)
00263  {
00264      _status = status;
00265      const_cast<l4_uint32_t volatile &>(hdr()->status) = status.raw;
00266  }
00267
00274  void add_irq_status(l4_uint32_t status)
00275  {
00276      const_cast<l4_uint32_t volatile &>(hdr()->irq_status) |= status;
00277  }
00278
00285  void set_device_needs_reset()
00286  {
00287      _status.device_needs_reset() = 1;
00288      const_cast<l4_uint32_t volatile &>(hdr()->status) = _status.raw;
00289  }
00290
00295  bool change_queue_config(l4_uint32_t num_queues)
00296  {
00297      if (sizeof(l4virtio_config_queue_t) * num_queues + _qoffset > L4_PAGESIZE)
00298          // too many queues does not fit into our page
00299          return false;
00300
00301      _nqueues = num_queues;
00302      reset_hdr(true);
00303      return true;
00304  }
00305
00312  l4virtio_config_queue_t volatile const *qconfig(unsigned index) const
00313  {
00314      if (L4_UNLIKELY(_qoffset < sizeof (l4virtio_config_hdr_t)))
00315          return 0;
00316
00317      if (L4_UNLIKELY(index >= _nqueues))
00318          return 0;
00319
00320      return reinterpret_cast<l4virtio_config_queue_t const *>
00321          (reinterpret_cast<char *>(_config.get()) + _qoffset) + index;
00322  }
00323
00327  void reset_hdr(bool inc_generation = false) const
00328  {
00329      _config->magic = L4VIRTIO_MAGIC;
00330      _config->version = 2;
00331      _config->device = _device;
00332      _config->vendor = _vendor;
00333      _config->status = 0;
00334      _config->irq_status = 0;
00335      _config->num_queues = _nqueues;
00336      _config->queues_offset = _qoffset;
00337
00338      memcpy(_config->dev_features_map, _host_features,
00339             sizeof(_config->dev_features_map));
00340      wmb();
00341      if (inc_generation)
00342          ++_config->generation;
00343  }
00344
00345
00356  bool reset_queue(unsigned index, unsigned num_max,
00357                  bool inc_generation = false,
00358                  unsigned device_notify_index = 0) const
00359  {
00360      l4virtio_config_queue_t volatile *qc;
00361      // this function is allowed to write to the device config
00362      qc = const_cast<l4virtio_config_queue_t volatile *>(qconfig(index));
00363      if (L4_UNLIKELY(qc == 0))
00364          return false;
00365  }

```

```

00366     qc->num_max = num_max;
00367     qc->num = 0;
00368     qc->ready = 0;
00369     qc->device_notify_index = device_notify_index;
00370     wmb();
00371     if (inc_generation)
00372         ++_config->generation;
00373
00374     return true;
00375 }
00376
00381 l4virtio_config_hdr_t const volatile *hdr() const
00382 { return _config.get(); }
00383
00388 L4::Cap<L4Re::Dataspace> ds() const { return _ds.get(); }
00389
00394 l4_addr_t ds_offset() const
00395 { return _ds_offset; }
00396 };
00397
00398
00399 template<typename PRIV_CONFIG>
00400 class Dev_config_t : public Dev_config
00401 {
00402 public:
00403     typedef PRIV_CONFIG Priv_config;
00404
00405     Dev_config_t(l4_uint32_t vendor, l4_uint32_t device,
00406                 l4_uint32_t num_queues = 0)
00407         : Dev_config(vendor, device, sizeof(PRIV_CONFIG), num_queues)
00408     {}
00409
00410     Dev_config_t(L4Re::Util::Shared_cap<L4Re::Dataspace> const &cfg,
00411                 l4_addr_t cfg_offset, l4_uint32_t vendor, l4_uint32_t device,
00412                 l4_uint32_t num_queues = 0)
00413         : Dev_config(cfg, cfg_offset, vendor, device, sizeof(PRIV_CONFIG),
00414                     num_queues)
00415     {}
00416
00417     Priv_config volatile *priv_config() const
00418     {
00419         return static_cast<Priv_config volatile *>(get_priv_config());
00420     }
00421 };
00422
00423 struct No_custom_data {};
00424
00425 template <typename DATA>
00426 class Driver_mem_region_t : public DATA
00427 {
00428 public:
00429     struct Flags
00430     {
00431         Flags() = default;
00432         explicit Flags(l4_uint32_t raw) : raw(raw) {}
00433
00434         l4_uint32_t raw;
00435         CXX_BITFIELD_MEMBER(0, 0, rw, raw);
00436     };
00437
00438 private:
00439     typedef L4Re::Util::Unique_cap<L4Re::Dataspace> Ds_cap;
00440
00441     l4_uint64_t _drv_base;
00442     l4_uint64_t _trans_offset;
00443     l4_umword_t _size;
00444     Flags _flags;
00445
00446     Ds_cap _ds;
00447     l4_addr_t _ds_offset;
00448
00449     L4Re::Rm::Unique_region<l4_addr_t> _local_base;
00450
00451     template<typename T>
00452     T _local(l4_uint64_t addr) const
00453     {
00454         return reinterpret_cast<T>(addr - _trans_offset);
00455     }
00456
00457 public:
00458     Driver_mem_region_t() : _size(0) {}
00459
00460     Driver_mem_region_t(l4_uint64_t drv_base, l4_umword_t size,
00461                         l4_addr_t offset, Ds_cap &&ds)
00462         : _drv_base(l4_trunc_page(drv_base)), _size(0), _flags(0),
00463           _ds_offset(l4_trunc_page(offset))

```

```

00515 {
00516     using L4Re::chksys;
00517     using L4Re::Env;
00518
00519     L4Re::Dataspace::Stats ds_info = L4Re::Dataspace::Stats();
00520     // Sometimes we work with dataspace that do not implement all dataspace
00521     // methods and return an error instead. An example of such a dataspace is
00522     // io's Vi::System_bus. We detect this case when the info method returns
00523     // -L4_ENOSYS and simply assume the dataspace is good for us.
00524     long err = ds->info(&ds_info);
00525     if (err >= 0)
00526     {
00527         l4_addr_t ds_size = l4_round_page(ds_info.size);
00528
00529         if (ds_size < L4_PAGESIZE)
00530             chksys(-L4_EINVAL, "DS too small");
00531
00532         if (_ds_offset >= ds_size)
00533             chksys(-L4_ERANGE, "offset larger than DS size");
00534
00535         size = l4_round_page(size);
00536         if (size > ds_size)
00537             chksys(-L4_EINVAL, "size larger than DS size");
00538
00539         if (_ds_offset > ds_size - size)
00540             chksys(-L4_EINVAL, "invalid offset or size");
00541
00542         // overflow check
00543         if ((ULONG_MAX - size) < _drv_base)
00544             chksys(-L4_EINVAL, "invalid size");
00545
00546         _flags.rw() = (ds_info.flags & L4Re::Dataspace::F::W).raw != 0;
00547     }
00548     else if (err == -L4_ENOSYS)
00549     {
00550         _flags.rw() = true;
00551     }
00552     else
00553     {
00554         chksys(err, "getting data-space infos");
00555     }
00556
00557     auto f = L4Re::Rm::F::Search_addr | L4Re::Rm::F::R;
00558     if (_flags.rw())
00559         f |= L4Re::Rm::F::W;
00560
00561     // use a big alignment to save PT/TLB entries and kernel memory resources!
00562     chksys(Env::env()->rm()->attach(&_local_base, size, f,
00563                                     L4::Ipc::make_cap(ds.get(), _flags.rw()
00564                                                         ? L4_CAP_FPAGE_RW
00565                                                         : L4_CAP_FPAGE_RO),
00566                                     _ds_offset, L4_SUPERPAGESHIFT));
00567
00568     _size = size;
00569     _ds = cxx::move(ds);
00570     _trans_offset = _drv_base - _local_base.get();
00571 }
00572
00573 bool is_writable() const { return _flags.rw(); }
00574
00575 Flags flags() const { return _flags; }
00576
00577 bool empty() const
00578 { return _size == 0; }
00579
00580 l4_uint64_t drv_base() const { return _drv_base; }
00581
00582 l4_addr_t local_base() const { return _local_base.get(); }
00583
00584 l4_umword_t size() const { return _size; }
00585
00586 l4_addr_t ds_offset() const { return _ds_offset; }
00587
00588 L4::Cap<L4Re::Dataspace> ds() const { return _ds.get(); }
00589
00590 bool contains(l4_uint64_t base, l4_umword_t size) const
00591 {
00592     if (base < _drv_base)
00593         return false;
00594
00595     if (base > _drv_base + _size - 1)
00596         return false;
00597
00598     if (size > _size)
00599         return false;
00600
00601     if (base - _drv_base > _size - size)

```

```

00617         return false;
00618
00619         return true;
00620     }
00621
00622     template<typename T>
00623     T *local(Ptr<T> p) const
00624     { return _local<T*>(p.get()); }
00625 };
00626
00627 typedef Driver_mem_region_t<No_custom_data> Driver_mem_region;
00628
00629 template <typename DATA>
00630 class Driver_mem_list_t
00631 {
00632 public:
00633     typedef Driver_mem_region_t<DATA> Mem_region;
00634
00635 private:
00636     cxx::unique_ptr<Mem_region[]> _l;
00637     unsigned _max;
00638     unsigned _free;
00639
00640 public:
00641     typedef L4Re::Util::Unique_cap<L4Re::Dataspace> Ds_cap;
00642
00643     Driver_mem_list_t() : _max(0), _free(0) {}
00644
00645     void init(unsigned max)
00646     {
00647         _l = cxx::make_unique<Driver_mem_region_t<DATA>[]>(max);
00648         _max = max;
00649         _free = 0;
00650     }
00651
00652     bool full() const
00653     { return _free == _max; }
00654
00655     Mem_region const *add(l4_uint64_t drv_base, l4_umword_t size,
00656                          l4_addr_t offset, Ds_cap &ds)
00657     {
00658         if (full())
00659             L4Re::chksys(-L4_ENOMEM);
00660
00661         _l[_free++] = Mem_region(drv_base, size, offset, cxx::move(ds));
00662         return &_l[_free - 1];
00663     }
00664
00665     void remove(Mem_region const *r)
00666     {
00667         if (r < &_l[0] || r >= &_l[_free])
00668             L4Re::chksys(-L4_ERANGE);
00669
00670         unsigned idx = r - &_l[0];
00671
00672         for (unsigned i = idx + 1; i < _free - 1; ++i)
00673             _l[i] = cxx::move(_l[i + 1]);
00674
00675         _l[--_free] = Mem_region();
00676     }
00677
00678     Mem_region *find(l4_uint64_t base, l4_umword_t size) const
00679     {
00680         return _find(base, size);
00681     }
00682
00683     void load_desc(Virtqueue::Desc const &desc, Request_processor const *p,
00684                   Virtqueue::Desc const **table) const
00685     {
00686         Mem_region const *r = find(desc.addr.get(), desc.len);
00687         if (L4_UNLIKELY(!r))
00688             throw Bad_descriptor(p, Bad_descriptor::Bad_address);
00689
00690         *table = static_cast<Virtqueue::Desc const *>(r->local(desc.addr));
00691     }
00692
00693     void load_desc(Virtqueue::Desc const &desc, Request_processor const *p,
00694                   Mem_region const **data) const
00695     {
00696         Mem_region const *r = find(desc.addr.get(), desc.len);
00697         if (L4_UNLIKELY(!r))
00698             throw Bad_descriptor(p, Bad_descriptor::Bad_address);
00699
00700         *data = r;
00701     }
00702
00703     template<typename ARG>

```

```

00777 void load_desc(Virtqueue::Desc const &desc, Request_processor const *p,
00778               ARG *data) const
00779 {
00780     Mem_region *r = find(desc.addr.get(), desc.len);
00781     if (L4_UNLIKELY(!r))
00782         throw Bad_descriptor(p, Bad_descriptor::Bad_address);
00783     *data = ARG(r, desc, p);
00784 }
00785
00786 Mem_region *begin() { return &_l[0]; }
00787 Mem_region const *begin() const { return &_l[0]; }
00788
00789 Mem_region *end() { return &_l[_free]; }
00790 Mem_region const *end() const { return &_l[_free]; }
00791
00792 private:
00793 Mem_region *_find(l4_uint64_t base, l4_umword_t size) const
00794 {
00795     for (unsigned i = 0; i < _free; ++i)
00796         if (_l[i].contains(base, size))
00797             return &_l[i];
00798     return 0;
00799 }
00800
00801 };
00802
00803 };
00804
00805 typedef Driver_mem_list_t<No_custom_data> Driver_mem_list;
00806
00807 template<typename DATA>
00808 class Device_t
00809 {
00810 public:
00811     typedef Driver_mem_list_t<DATA> Mem_list;
00812
00813 protected:
00814     Mem_list _mem_info;
00815
00816 private:
00817     Dev_config *_device_config;
00818
00819     using Ds_vector = std::vector<L4::Cap<L4Re::Dataspace>>;
00820     std::shared_ptr<Ds_vector const> _trusted_ds_caps;
00821
00822     bool _trusted_ds_validation_enabled = false;
00823
00824 public:
00825     L4_RPC_LEGACY_DISPATCH(L4virtio::Device);
00826     template<typename IOS> int virtio_dispatch(unsigned r, IOS &ios)
00827     { return dispatch(r, ios); }
00828
00829     virtual void reset() = 0;
00830
00831     virtual bool check_features()
00832     { return true; }
00833
00834     virtual bool check_queues() = 0;
00835
00836     virtual int reconfig_queue(unsigned idx) = 0;
00837
00838     virtual void cfg_changed(unsigned /* reg */) {};
00839
00840     virtual void register_single_driver_irq()
00841     { L4Re::chksys(-L4_ENOSYS, "Legacy single IRQ interface not implemented."); }
00842
00843     virtual void trigger_driver_config_irq() = 0;
00844
00845     virtual L4::Cap<L4::Irq> device_notify_irq() const
00846     {
00847         L4Re::chksys(-L4_ENOSYS, "Legacy single IRQ interface not implemented.");
00848         return L4::Cap<L4::Irq>();
00849     }
00850
00851     virtual void register_driver_irq(unsigned idx)
00852     {
00853         if (idx != 0)
00854             L4Re::chksys(-L4_ENOSYS, "Multi IRQ interface not implemented.");
00855         register_single_driver_irq();
00856     }
00857
00858     virtual L4::Cap<L4::Irq> device_notify_irq(unsigned idx)
00859     {
00860         if (idx != 0)
00861             L4Re::chksys(-L4_ENOSYS, "Multi IRQ interface not implemented.");
00862     }
00863
00864
00865
00866
00867
00868
00869
00870
00871

```

```

00892     return device_notify_irq();
00893 }
00894
00896 virtual unsigned num_events_supported() const
00897 { return 1; }
00898
00899 virtual L4::Ipc_svr::Server_iface *server_iface() const = 0;
00900
00901 // Make it possible to react to changes in the list of memory regions
00902 virtual void check_mem_list() {};
00903
00907 Device_t(Dev_config *dev_config)
00908 : _device_config(dev_config)
00909 {}
00910
00914 Mem_list const *mem_info() const
00915 { return &_amp;_mem_info; };
00916
00917 long op_set_status(L4virtio::Device::Rights, unsigned status)
00918 { return _set_status(status); }
00919
00920 long op_config_queue(L4virtio::Device::Rights, unsigned queue)
00921 {
00922     Dev_config::Status status = _device_config->status();
00923     if (status.fail_state() || !status.acked() || !status.driver())
00924         return -L4_EIO;
00925
00926     return reconfig_queue(queue);
00927 }
00928
00929 long op_register_ds(L4virtio::Device::Rights,
00930                    L4::Ipc::Snd_fpage ds_cap_fp, l4_uint64_t ds_base,
00931                    l4_umword_t offset, l4_umword_t sz)
00932 {
00933     L4Re::Util::Dbg()
00934         .printf("Registering dataspace from 0x%llx with %lu KiB, offset 0x%lx\n",
00935                ds_base, sz » 10, offset);
00936
00937     _check_n_init_shm(ds_cap_fp, ds_base, sz, offset);
00938
00939     return 0;
00940 }
00941
00942 long op_device_config(L4virtio::Device::Rights,
00943                      L4::Ipc::Cap<L4Re::Dataspace> &config_ds,
00944                      l4_addr_t &ds_offset)
00945 {
00946     L4Re::Util::Dbg()
00947         .printf("register client: config DS: %lx\n", _device_config->ds().cap());
00948
00949     config_ds = L4::Ipc::make_cap(_device_config->ds(), L4_CAP_FPAGE_RW);
00950     ds_offset = _device_config->ds_offset();
00951     return 0;
00952 }
00953
00954 long op_device_notification_irq(L4virtio::Device::Rights,
00955                                unsigned idx,
00956                                L4::Ipc::Cap<L4::Triggerable> &irq)
00957 {
00958     auto cap = device_notify_irq(idx);
00959
00960     if (!cap.is_valid())
00961         return -L4_EINVAL;
00962
00963     irq = L4::Ipc::make_cap(cap, L4_CAP_FPAGE_RO);
00964     return L4_EOK;
00965 }
00966
00967 int op_bind(L4::Icu::Rights, l4_umword_t idx, L4::Ipc::Snd_fpage irq_cap_fp)
00968 {
00969     if (idx >= num_events_supported())
00970         return -L4_ERANGE;
00971
00972     if (!irq_cap_fp.cap_received())
00973         return -L4_EINVAL;
00974
00975     register_driver_irq(idx);
00976
00977     return L4_EOK;
00978 }
00979
00980 int op_unbind(L4::Icu::Rights, l4_umword_t, L4::Ipc::Snd_fpage)
00981 {
00982     return -L4_ENOSYS;
00983 }
00984
00985 int op_info(L4::Icu::Rights, L4::Icu::_Info &info)

```

```

00986 {
00987     info.features = 0;
00988     info.nr_irqs = num_events_supported();
00989     info.nr_msis = 0;
00990
00991     return L4_EOK;
00992 }
00993
00994 int op_msi_info(L4::Icu::Rights, l4_umword_t, l4_uint64_t, l4_icu_msi_info_t &)
00995 { return -L4_ENOSYS; }
00996
00997 int op_mask(L4::Icu::Rights, l4_umword_t)
00998 { return -L4_ENOSYS; }
00999
01000 int op_unmask(L4::Icu::Rights, l4_umword_t)
01001 { return -L4_ENOREPLY; }
01002
01003 int op_set_mode(L4::Icu::Rights, l4_umword_t, l4_umword_t)
01004 { return -L4_ENOSYS; }
01005
01019 void reset_queue_config(unsigned idx, unsigned num_max,
01020                        bool inc_generation = false,
01021                        unsigned device_notify_index = 0)
01022 {
01023     _device_config->reset_queue(idx, num_max, inc_generation,
01024                                device_notify_index);
01025 }
01026
01031 void init_mem_info(unsigned num)
01032 {
01033     _mem_info.init(num);
01034 }
01035
01043 void device_error()
01044 {
01045     reset();
01046     _device_config->set_device_needs_reset();
01047
01048     // the device MUST NOT notify the driver before DRIVER_OK.
01049     if (_device_config->status().driver_ok())
01050         trigger_driver_config_irq();
01051 }
01052
01066 bool setup_queue(Virtqueue *q, unsigned qn, unsigned num_max)
01067 {
01068     l4virtio_config_queue_t volatile const *qc;
01069     qc = _device_config->qconfig(qn);
01070     if (L4_UNLIKELY(qc == 0))
01071         return false;
01072
01073     if (!qc->ready)
01074     {
01075         q->disable();
01076         return true;
01077     }
01078
01079     // read to local variables before check
01080     l4_uint32_t num = qc->num;
01081     l4_uint64_t desc = qc->desc_addr;
01082     l4_uint64_t avail = qc->avail_addr;
01083     l4_uint64_t used = qc->used_addr;
01084
01085     if (0)
01086         printf("%p: setup queue: num=0x%x max_num=0x%x desc=0x%llx avail=0x%llx used=0x%llx\n",
01087                this, num, num_max, desc, avail, used);
01088
01089     if (!num || num > num_max)
01090         return false;
01091
01092     // num must be power of two
01093     if (num & (num - 1))
01094         return false;
01095
01096     if (desc & 0xf)
01097         return false;
01098
01099     if (avail & 0x1)
01100         return false;
01101
01102     if (used & 0x3)
01103         return false;
01104
01105     auto const *desc_info = _mem_info.find(desc, Virtqueue::desc_size(num));
01106     if (L4_UNLIKELY(!desc_info))
01107         return false;
01108
01109     auto const *avail_info = _mem_info.find(avail, Virtqueue::avail_size(num));

```

```

01110     if (L4_UNLIKELY(!avail_info))
01111         return false;
01112
01113     auto const *used_info = _mem_info.find(used, Virtqueue::used_size(num));
01114     if (L4_UNLIKELY(!used_info || !used_info->is_writable()))
01115         return false;
01116
01117     L4Re::Util::Dbg()
01118         .printf("shm=[%llx-%llx] local=[%lx-%lx] desc=[%llx-%llx] (%p-%p)\n",
01119             desc_info->drv_base(), desc_info->drv_base() + desc_info->size() - 1,
01120             desc_info->local_base(),
01121             desc_info->local_base() + desc_info->size() - 1,
01122             desc, desc + Virtqueue::desc_size(num),
01123             desc_info->local(Ptr<char>(desc)),
01124             desc_info->local(Ptr<char>(desc)) + Virtqueue::desc_size(num));
01125
01126     L4Re::Util::Dbg()
01127         .printf("shm=[%llx-%llx] local=[%lx-%lx] avail=[%llx-%llx] (%p-%p)\n",
01128             avail_info->drv_base(), avail_info->drv_base() + avail_info->size() - 1,
01129             avail_info->local_base(),
01130             avail_info->local_base() + avail_info->size() - 1,
01131             avail, avail + Virtqueue::avail_size(num),
01132             avail_info->local(Ptr<char>(avail)),
01133             avail_info->local(Ptr<char>(avail)) + Virtqueue::avail_size(num));
01134
01135     L4Re::Util::Dbg()
01136         .printf("shm=[%llx-%llx] local=[%lx-%lx] used=[%llx-%llx] (%p-%p)\n",
01137             used_info->drv_base(), used_info->drv_base() + used_info->size() - 1,
01138             used_info->local_base(),
01139             used_info->local_base() + used_info->size() - 1,
01140             used, used + Virtqueue::used_size(num),
01141             used_info->local(Ptr<char>(used)),
01142             used_info->local(Ptr<char>(used)) + Virtqueue::used_size(num));
01143
01144     q->setup(num, desc_info->local(Ptr<void>(desc)),
01145         avail_info->local(Ptr<void>(avail)),
01146         used_info->local(Ptr<void>(used)));
01147     return true;
01148 }
01149
01150 void check_n_init_shm(L4Re::Util::Unique_cap<L4Re::Dataspace> &&shm,
01151     l4_uint64_t base, l4_umword_t size, l4_addr_t offset)
01152 {
01153     if (_mem_info.full())
01154         L4Re::chksys(-L4_ENOMEM);
01155
01156     auto const *i = _mem_info.add(base, size, offset, cxx::move(shm));
01157     L4Re::Util::Dbg()
01158         .printf("PORT[%p]: DMA guest [%llx-%llx] local [%lx-%lx] offset %lx\n",
01159             this, i->drv_base(), i->drv_base() + i->size() - 1,
01160             i->local_base(),
01161             i->local_base() + i->size() - 1,
01162             i->ds_offset());
01163 }
01164
01173 bool handle_mem_cmd_write()
01174 {
01175     l4_uint32_t cmd = _device_config->get_cmd();
01176     if (L4_LIKELY(!(cmd & L4VIRTIO_CMD_MASK)))
01177         return false;
01178
01179     switch (cmd & L4VIRTIO_CMD_MASK)
01180     {
01181     case L4VIRTIO_CMD_SET_STATUS:
01182         _set_status(cmd & ~L4VIRTIO_CMD_MASK);
01183         break;
01184
01185     case L4VIRTIO_CMD_CFG_QUEUE:
01186         reconfig_queue(cmd & ~L4VIRTIO_CMD_MASK);
01187         break;
01188
01189     case L4VIRTIO_CMD_CFG_CHANGED:
01190         cfg_changed(cmd & ~L4VIRTIO_CMD_MASK);
01191         break;
01192
01193     default:
01194         // unknown command
01195         break;
01196     }
01197
01198     _device_config->reset_cmd();
01199
01200     return true;
01201 }
01202
01206 void enable_trusted_ds_validation()
01207 {

```



```

01208     _trusted_ds_validation_enabled = true;
01209 }
01210
01216 void
01217 add_trusted_daspaces(std::shared_ptr<Ds_vector> ds)
01218 {
01219     _trusted_ds_caps = ds;
01220 }
01221
01222
01223 private:
01235 long validate_ds(L4::Cap<L4Re::Dataspace> ds)
01236 {
01237     if (!_trusted_ds_caps)
01238         return -L4_EINVAL;
01239     if (std::any_of(_trusted_ds_caps->cbegin(), _trusted_ds_caps->cend(),
01240         [&ds] (L4::Cap<L4Re::Dataspace> cap)
01241         {
01242             return L4Re::Env::env()->task()
01243                 ->cap_equal(ds, cap).label() == 1;
01244         })
01245     )
01246     {
01247         return L4_EOK;
01248     }
01249     return -L4_EINVAL;
01250 }
01251
01252 void _check_n_init_shm(L4::Ipc::Snd_fpage shm_cap_fp,
01253     l4_uint64_t base, l4_umword_t size, l4_addr_t offset)
01254 {
01255     if (!shm_cap_fp.cap_received())
01256         L4Re::chksys(-L4_EINVAL);
01257
01258     L4Re::Util::Unique_cap<L4Re::Dataspace> ds(
01259         L4Re::chkcap(server_iface()->template_rcv_cap<L4Re::Dataspace>(0)));
01260     L4Re::chksys(server_iface()->realloc_rcv_cap(0));
01261
01262     if (_trusted_ds_validation_enabled)
01263         L4Re::chksys(validate_ds(ds.get()), "Validating the dataspace.");
01264
01265     check_n_init_shm(cxx::move(ds), base, size, offset);
01266
01267     check_mem_list();
01268 }
01269
01270 bool check_features_internal()
01271 {
01272     static_assert(sizeof(l4virtio_config_hdr_t::driver_features_map)
01273         == sizeof(l4virtio_config_hdr_t::dev_features_map),
01274         "Driver and device feature maps must be of the same size");
01275
01276     // From the Virtio 1.0 specification 6.1 Driver Requirements and 6.2 Device
01277     // Requirements: A driver MUST accept VIRTIO_F_VERSION_1 if it is offered.
01278     // A device MUST offer VIRTIO_F_VERSION_1. A device MAY fail to operate
01279     // further if VIRTIO_F_VERSION_1 is not accepted.
01280     //
01281     // The L4virtio implementation does not support legacy interfaces so we
01282     // fail here if the Virtio 1.0 feature was not accepted.
01283     if (!_device_config->get_guest_feature(L4VIRTIO_FEATURE_VERSION_1))
01284         return false;
01285
01286     for (auto i = 0u;
01287         i < sizeof(l4virtio_config_hdr_t::driver_features_map)
01288         / sizeof(l4virtio_config_hdr_t::driver_features_map[0]);
01289         i++)
01290     {
01291         // Driver must not accept features that were not offered by device
01292         if (_device_config->guest_features(i)
01293             & ~_device_config->host_features(i))
01294             return false;
01295     }
01296     return check_features();
01297 }
01298
01300 void check_and_update_status(Dev_config::Status status)
01301 {
01302     // snapshot of current status
01303     Dev_config::Status current_status = _device_config->status();
01304
01305     // handle reset
01306     if (!status.raw)
01307     {
01308         _device_config->set_status(status);
01309         return;
01310     }
01311

```

```

01332 // Do no further processing in case of driver or device failure. If FAILED
01333 // or DEVICE_NEEDS_RESET are set only these fail_state bits will be set in
01334 // addition to the current status bits already set.
01335 if (current_status.fail_state() || status.fail_state())
01336 {
01337     if (current_status.fail_state() != status.fail_state())
01338     {
01339         current_status.fail_state() =
01340             current_status.fail_state() | status.fail_state();
01341         _device_config->set_status(current_status);
01342     }
01343     return;
01344 }
01345
01346 // Enforce init sequence ACKNOWLEDGE, DRIVER, FEATURES_OK, DRIVER_OK.
01347 // We do not enforce that only one additional new bit is set per call.
01348 if ((!status.acked() && status.driver())
01349     || (!status.driver() && status.features_ok())
01350     || (!status.features_ok() && status.driver_ok()))
01351 {
01352     current_status.device_needs_reset() = 1;
01353     _device_config->set_status(current_status);
01354     return;
01355 }
01356
01357 // only check feature compatibility before DRIVER_OK is set
01358 if (status.features_ok() && !status.driver_ok()
01359     && !check_features_internal())
01360     status.features_ok() = 0;
01361
01362 // Note that if FEATURES_OK and DRIVER_OK are both updated to being set
01363 // at the same time the above check_features_internal() is skipped; this is
01364 // considered undefined behaviour but it is not prevented.
01365 if (status.running() && !check_queues())
01366 {
01367     current_status.device_needs_reset() = 1;
01368     _device_config->set_status(current_status);
01369     return;
01370 }
01371
01372 _device_config->set_status(status);
01373 }
01374
01375 int _set_status(unsigned new_status)
01376 {
01377     if (new_status == 0)
01378     {
01379         L4Re::Util::Dbg().printf("Resetting device\n");
01380         reset();
01381         _device_config->reset_hdr(true);
01382     }
01383
01384     Dev_config::Status status(new_status);
01385     check_and_update_status(status);
01386
01387     return 0;
01388 }
01389
01390 };
01391
01392 typedef Device_t<No_custom_data> Device;
01393
01394 } // namespace Svr
01395
01396 }
01397
01398 }

```

16.270 virtio

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /* SPDX-License-Identifier: MIT */
00003 /*
00004  * Copyright (C) 2014-2020, 2023-2025 Kernkonzept GmbH.
00005  * Author(s): Alexander Warg <alexander.warg@kernkonzept.com>
00006  *
00007  */
00008 #pragma once
00009
00010 #include <l4/sys/types.h>
00011 #include <l4/cxx/bitfield>
00012 #include <l4/cxx/minmax>
00013 #include <l4/cxx/utils>
00014
00015 #include <limits.h>

```

```

00016 #include <string.h>
00017 #include <stdio.h>
00018
00019 #include "../virtqueue"
00020
00026 namespace L4virtio {
00027 namespace Svr {
00028
00032 struct Dev_status
00033 {
00034     unsigned char raw;
00035     Dev_status() = default;
00036
00038     explicit Dev_status(l4_uint32_t v) : raw(v) {}
00039
00040     CXX_BITFIELD_MEMBER(0, 0, acked, raw);
00041     CXX_BITFIELD_MEMBER(1, 1, driver, raw);
00042     CXX_BITFIELD_MEMBER(2, 2, driver_ok, raw);
00043     CXX_BITFIELD_MEMBER(3, 3, features_ok, raw);
00044     CXX_BITFIELD_MEMBER(6, 7, fail_state, raw);
00045     CXX_BITFIELD_MEMBER(6, 6, device_needs_reset, raw);
00046     CXX_BITFIELD_MEMBER(7, 7, failed, raw);
00047
00057     bool running() const
00058     {
00059         return (raw == 0xf);
00060     }
00061 };
00062
00066 struct Dev_features
00067 {
00068     l4_uint32_t raw;
00069     Dev_features() = default;
00070
00072     explicit Dev_features(l4_uint32_t v) : raw(v) {}
00073
00074     CXX_BITFIELD_MEMBER(28, 28, ring_indirect_desc, raw);
00075     CXX_BITFIELD_MEMBER(29, 29, ring_event_idx, raw);
00076 };
00077
00078
00087 class Virtqueue : public L4virtio::Virtqueue
00088 {
00089 public:
00093     class Head_desc
00094     {
00095     friend class Virtqueue;
00096 private:
00097     Virtqueue::Desc const *_d;
00098     Head_desc(Virtqueue *r, unsigned i) : _d(r->desc(i)) {}
00099
00100 public:
00102     Head_desc() : _d(0) {}
00103
00105     bool valid() const { return _d; }
00106
00108     explicit operator bool () const
00109     { return valid(); }
00110
00112     Desc const *desc() const
00113     { return _d; }
00114 };
00115
00116 struct Request : Head_desc
00117 {
00118     Virtqueue *ring = nullptr;
00119     Request() = default;
00120 private:
00121     friend class Virtqueue;
00122     Request(Virtqueue *r, unsigned i) : Head_desc(r, i), ring(r) {}
00123 };
00124
00125
00136 Request next_avail()
00137 {
00138     if (L4_LIKELY(_current_avail != _avail->idx))
00139     {
00140         rmb();
00141         unsigned head = _current_avail & _idx_mask;
00142         ++_current_avail;
00143         return Request(this, _avail->ring[head]);
00144     }
00145     return Request();
00146 }
00147
00160 void rewind_avail(Head_desc const &d)
00161 {

```

```

00162     unsigned head_idx = d._d - _desc;
00163     // Calculate the distance between _current_avail and head_idx, taking into
00164     // account that _current_avail might have wrapped around with respect to
00165     // _idx_mask in the meantime.
00166     _current_avail -= (_current_avail - head_idx) & _idx_mask;
00167 }
00168
00175 bool desc_avail() const
00176 {
00177     return _current_avail != _avail->idx;
00178 }
00179
00190 void consumed(Head_desc const &r, l4_uint32_t len = 0)
00191 {
00192     l4_uint16_t i = _used->idx & _idx_mask;
00193     _used->ring[i] = Used_elem(r._d - _desc, len);
00194     wmb();
00195     ++_used->idx;
00196 }
00197
00212 template<typename ITER>
00213 void consumed(ITER const &begin, ITER const &end)
00214 {
00215     l4_uint16_t added = 0;
00216     l4_uint16_t idx = _used->idx;
00217
00218     for (auto elem = begin ; elem != end; ++elem, ++added)
00219         _used->ring[(idx + added) & _idx_mask]
00220             = Used_elem(elem->first._d - _desc, elem->second);
00221
00222     wmb();
00223     _used->idx += added;
00224 }
00225
00239 template<typename QUEUE_OBSERVER>
00240 void finish(Head_desc &d, QUEUE_OBSERVER *o, l4_uint32_t len = 0)
00241 {
00242     consumed(d, len);
00243     o->notify_queue(this);
00244     d._d = 0;
00245 }
00246
00261 template<typename ITER, typename QUEUE_OBSERVER>
00262 void finish(ITER const &begin, ITER const &end, QUEUE_OBSERVER *o)
00263 {
00264     consumed(begin, end);
00265     o->notify_queue(this);
00266 }
00267
00273 void disable_notify()
00274 {
00275     if (L4_LIKELY(ready()))
00276         _used->flags.no_notify() = 1;
00277 }
00278
00284 void enable_notify()
00285 {
00286     if (L4_LIKELY(ready()))
00287         _used->flags.no_notify() = 0;
00288 }
00289
00298 Desc const *desc(unsigned idx) const
00299 { return _desc + idx; }
00300
00301 };
00302
00306 struct Data_buffer
00307 {
00308     char *pos;
00309     l4_uint32_t left;
00310
00311     Data_buffer() = default;
00312
00322 template<typename T>
00323 explicit Data_buffer(T *p)
00324 : pos(reinterpret_cast<char *>(p)), left(sizeof(T))
00325 {}
00326
00336 template<typename T>
00337 void set(T *p)
00338 {
00339     pos = reinterpret_cast<char *>(p);
00340     left = sizeof(T);
00341 }
00342
00354 l4_uint32_t copy_to(Data_buffer *dst, l4_uint32_t max = UINT_MAX)
00355 {

```

```

00356     unsigned long bytes = cxx::min(cxx::min(left, dst->left), max);
00357     memcpy(dst->pos, pos, bytes);
00358     left -= bytes;
00359     pos += bytes;
00360     dst->left -= bytes;
00361     dst->pos += bytes;
00362     return bytes;
00363 }
00364
00375 l4_uint32_t skip(l4_uint32_t bytes)
00376 {
00377     unsigned long b = cxx::min(left, bytes);
00378     left -= b;
00379     pos += b;
00380     return b;
00381 }
00382
00388 bool done() const
00389 { return left == 0; }
00390 };
00391
00392 class Request_processor;
00393
00397 struct Bad_descriptor
00398 {
00400     enum Error
00401     {
00402         Bad_address,
00403         Bad_rights,
00404         Bad_flags,
00405         Bad_next,
00406         Bad_size
00407     };
00408
00410     Request_processor const *proc;
00411
00412     // The error code
00413     Error error;
00414
00421     Bad_descriptor(Request_processor const *proc, Error e)
00422     : proc(proc), error(e)
00423     {}
00424
00430     char const *message() const
00431     {
00432         static char const *const err[] =
00433         {
00434             [Bad_address] = "Descriptor address cannot be translated",
00435             [Bad_rights] = "Insufficient memory access rights",
00436             [Bad_flags] = "Invalid descriptor flags",
00437             [Bad_next] = "The descriptor's `next` index is invalid",
00438             [Bad_size] = "Invalid size of the memory block"
00439         };
00440
00441         if (error >= (sizeof(err) / sizeof(err[0])) || !err[error])
00442             return "Unknown error";
00443
00444         return err[error];
00445     }
00446 };
00447
00448
00472 class Request_processor
00473 {
00474 private:
00476     Virtqueue::Desc const *_table;
00477
00479     Virtqueue::Desc _current;
00480
00482     l4_uint16_t _num;
00483
00484 public:
00500     template<typename DESC_MAN, typename ...ARGS>
00501     void start(DESC_MAN *dm, Virtqueue *ring, Virtqueue::Head_desc const &request, ARGS... args)
00502     {
00503         _current = cxx::access_once(request.desc());
00504
00505         if (_current.flags.indirect())
00506         {
00507             dm->load_desc(_current, this, &_table);
00508             _num = _current.len / sizeof(Virtqueue::Desc);
00509             if (L4_UNLIKELY(!_num))
00510                 throw Bad_descriptor(this, Bad_descriptor::Bad_size);
00511             _current = cxx::access_once(_table);
00512         }
00513     }
00514     else

```

```

00515     {
00516         _table = ring->desc(0);
00517         _num = ring->num();
00518     }
00519
00520     dm->load_desc(_current, this, cxx::forward<ARGS>(args)...);
00521 }
00522
00533 template<typename DESC_MAN, typename ...ARGS>
00534 Virtqueue::Request const &start(DESC_MAN *dm, Virtqueue::Request const &request, ARGS... args)
00535 {
00536     start(dm, request.ring, request, cxx::forward<ARGS>(args)...);
00537     return request;
00538 }
00539
00545 Virtqueue::Desc::Flags current_flags() const
00546 { return _current.flags; }
00547
00553 bool has_more() const
00554 { return _current.flags.next(); }
00555
00569 template<typename DESC_MAN, typename ...ARGS>
00570 bool next(DESC_MAN *dm, ARGS... args)
00571 {
00572     if (!_current.flags.next())
00573         return false;
00574
00575     if (L4_UNLIKELY(_current.next >= _num))
00576         throw Bad_descriptor(this, Bad_descriptor::Bad_next);
00577
00578     _current = cxx::access_once(_table + _current.next);
00579
00580     if (0) // we ignore this for performance reasons
00581         if (L4_UNLIKELY(_current.flags.indirect()))
00582             throw Bad_descriptor(this, Bad_descriptor::Bad_flags);
00583
00584     // must throw an exception in case of a bad descriptor
00585     dm->load_desc(_current, this, cxx::forward<ARGS>(args)...);
00586     return true;
00587 }
00588 };
00589
00590 }
00591 }

```

16.271 virtio-block

```

00001 // vi:ft=cpp
00002 /* SPDX-License-Identifier: MIT */
00003 /*
00004  * Copyright (C) 2015-2022, 2024 Kernkonzept GmbH.
00005  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00006  *           Manuel von Oltersdorff-Kalettka <manuel.kalettka@kernkonzept.com>
00007  */
00008
00009 #pragma once
00010
00011 #include <l4/sys/factory>
00012 #include <l4/sys/semaphore>
00013 #include <l4/re/dataspace>
00014 #include <l4/re/env>
00015 #include <l4/re/util/unique_cap>
00016 #include <l4/re/util/object_registry>
00017 #include <l4/re/error_helper>
00018
00019 #include <l4/util/atomic.h>
00020 #include <l4/util/bitops.h>
00021 #include <l4/l4virtio/client/l4virtio>
00022 #include <l4/l4virtio/l4virtio>
00023 #include <l4/l4virtio/virtqueue>
00024 #include <l4/l4virtio/virtio_block.h>
00025 #include <l4/sys/consts.h>
00026
00027 #include <cstring>
00028 #include <vector>
00029 #include <functional>
00030
00031 namespace L4virtio { namespace Driver {
00032
00036 class Block_device : public Device
00037 {
00038 public:
00039     typedef std::function<void(unsigned char)> Callback;

```

```

00040
00041 private:
00042     enum { Header_size = sizeof(l4virtio_block_header_t) };
00043
00044     struct Request
00045     {
00046         l4_uint16_t tail;
00047         Callback callback;
00048
00049         Request() : tail(Virtqueue::Eq), callback(0) {}
00050     };
00051
00052 public:
00053     class Handle
00054     {
00055     public:
00056         friend Block_device;
00057         l4_uint16_t head;
00058
00059         explicit Handle(l4_uint16_t descno) : head(descno) {}
00060
00061     public:
00062         Handle() : head(Virtqueue::Eq) {}
00063         bool valid() const { return head != Virtqueue::Eq; }
00064     };
00065
00066 void setup_device(L4::Cap<L4virtio::Device> srvcap, l4_size_t usermem,
00067                 void **userdata, Ptr<void> &user_devaddr,
00068                 L4::Cap<L4Re::Dataspace> qds = L4::Cap<L4Re::Dataspace>(),
00069                 l4_uint32_t fmask0 = -1U, l4_uint32_t fmask1 = -1U)
00070 {
00071     // Contact device.
00072     driver_connect(srvcap);
00073
00074     if (_config->device != L4VIRTIO_ID_BLOCK)
00075         L4Re::chksys(-L4_ENODEV, "Device is not a block device.");
00076
00077     if (_config->num_queues != 1)
00078         L4Re::chksys(-L4_EINVAL, "Invalid number of queues reported.");
00079
00080     // Memory is shared in one large dataspace which contains queues,
00081     // space for header/status and additional user-defined memory.
00082     unsigned queuesz = max_queue_size(0);
00083     l4_size_t totalsz = l4_round_page(usermem);
00084
00085     l4_uint64_t const header_offset =
00086         l4_round_size(_queue.total_size(queuesz),
00087                       l4util_bsr(aligned(l4virtio_block_header_t)));
00088     l4_uint64_t const status_offset = header_offset + queuesz * Header_size;
00089     l4_uint64_t const usermem_offset = l4_round_page(status_offset + queuesz);
00090
00091     // reserve space for one header/status per descriptor
00092     // TODO Should be reduced to 1/3 but this way no freelist is needed.
00093     totalsz += usermem_offset;
00094
00095     auto *e = L4Re::Env::env();
00096     if (!qds.is_valid())
00097     {
00098         _ds = L4Re::chkcap(L4Re::Util::make_unique_cap<L4Re::Dataspace>(),
00099                           "Allocate queue dataspace capability");
00100         L4Re::chksys(e->mem_alloc()->alloc(totalsz, _ds.get(),
00101                                             L4Re::Mem_alloc::Continuous
00102                                             | L4Re::Mem_alloc::Pinned),
00103                       "Allocate memory for virtio structures");
00104         _queue_ds = _ds.get();
00105     }
00106     else
00107     {
00108         if (qds->size() < totalsz)
00109             L4Re::chksys(-L4_EINVAL, "External queue dataspace too small.");
00110         _queue_ds = qds;
00111     }
00112
00113     // Now sort out which region goes where in the dataspace.
00114     L4Re::chksys(e->rm()->attach(&_queue_region, totalsz,
00115                                 L4Re::Rm::F::Search_addr | L4Re::Rm::F::RW,
00116                                 L4::Ipc::make_cap_rw(_queue_ds), 0,
00117                                 L4_PAGESHIFT),
00118                   "Attach dataspace for virtio structures");
00119
00120     l4_uint64_t devaddr;
00121     L4Re::chksys(register_ds(_queue_ds, 0, totalsz, &devaddr),
00122                   "Register queue dataspace with device");
00123
00124     _queue.init_queue(queuesz, _queue_region.get());
00125
00126     config_queue(0, queuesz, devaddr, devaddr + _queue.avail_offset(),
00127                 devaddr + _queue.used_offset());
00128
00129

```

```

00154     _header_addr = devaddr + header_offset;
00155     _headers = reinterpret_cast<l4virtio_block_header_t *>(_queue_region.get()
00156                                                         + header_offset);
00157
00158
00159     _status_addr = devaddr + status_offset;
00160     _status = _queue_region.get() + status_offset;
00161
00162     user_devaddr = Ptr<void>(devaddr + usermem_offset);
00163     if (userdata)
00164         *userdata = _queue_region.get() + usermem_offset;
00165
00166     // setup the callback mechanism
00167     _pending.assign(queueesz, Request());
00168
00169     // Finish handshake with device.
00170     _config->driver_features_map[0] = fmask0;
00171     _config->driver_features_map[1] = fmask1;
00172     driver_acknowledge();
00173 }
00174
00175 l4virtio_block_config_t const &device_config() const
00176 {
00177     return *_config->device_config<l4virtio_block_config_t>();
00178 }
00179
00180 Handle start_request(l4_uint64_t sector, l4_uint32_t type,
00181                    Callback callback)
00182 {
00183     l4_uint16_t descno = _queue.alloc_descriptor();
00184     if (descno == Virtqueue::Eoq)
00185         return Handle(Virtqueue::Eoq);
00186
00187     L4virtio::Virtqueue::Desc &desc = _queue.desc(descno);
00188     Request &req = _pending[descno];
00189
00190     // setup the header
00191     l4virtio_block_header_t &head = _headers[descno];
00192     head.type = type;
00193     head.ioprio = 0;
00194     head.sector = sector;
00195
00196     // and put it in the descriptor
00197     desc.addr = Ptr<void>(_header_addr + descno * Header_size);
00198     desc.len = Header_size;
00199     desc.flags.raw = 0; // no write, no indirect
00200
00201     req.tail = descno;
00202     req.callback = callback;
00203
00204     return Handle(descno);
00205 }
00206
00207 int add_block(Handle handle, Ptr<void> addr, l4_uint32_t size)
00208 {
00209     l4_uint16_t descno = _queue.alloc_descriptor();
00210     if (descno == Virtqueue::Eoq)
00211         return -L4_EAGAIN;
00212
00213     Request &req = _pending[handle.head];
00214     L4virtio::Virtqueue::Desc &desc = _queue.desc(descno);
00215     L4virtio::Virtqueue::Desc &prev = _queue.desc(req.tail);
00216
00217     prev.next = descno;
00218     prev.flags.next() = true;
00219
00220     desc.addr = addr;
00221     desc.len = size;
00222     desc.flags.raw = 0;
00223     if (_headers[handle.head].type > 0) // write or flush request
00224         desc.flags.write() = true;
00225
00226     req.tail = descno;
00227
00228     return L4_EOK;
00229 }
00230
00231 int send_request(Handle handle)
00232 {
00233     // add the status bit
00234     auto descno = _queue.alloc_descriptor();
00235     if (descno == Virtqueue::Eoq)
00236         return -L4_EAGAIN;
00237
00238     Request &req = _pending[handle.head];
00239     L4virtio::Virtqueue::Desc &desc = _queue.desc(descno);
00240     L4virtio::Virtqueue::Desc &prev = _queue.desc(req.tail);

```



```

00275
00276     prev.next = descno;
00277     prev.flags.next() = true;
00278
00279     desc.addr = Ptr<void>(_status_addr + descno);
00280     desc.len = 1;
00281     desc.flags.raw = 0;
00282     desc.flags.write() = true;
00283
00284     req.tail = descno;
00285
00286     send(_queue, handle.head);
00287
00288     return L4_EOK;
00289 }
00290
00306 int process_request(Handle handle)
00307 {
00308     // add the status bit
00309     auto descno = _queue.alloc_descriptor();
00310     if (descno == Virtqueue::Eoq)
00311         return -L4_EAGAIN;
00312
00313     L4virtio::Virtqueue::Desc &desc = _queue.desc(descno);
00314     L4virtio::Virtqueue::Desc &prev = _queue.desc(_pending[handle.head].tail);
00315
00316     prev.next = descno;
00317     prev.flags.next() = true;
00318
00319     desc.addr = Ptr<void>(_status_addr + descno);
00320     desc.len = 1;
00321     desc.flags.raw = 0;
00322     desc.flags.write() = true;
00323
00324     _pending[handle.head].tail = descno;
00325
00326     int ret = send_and_wait(_queue, handle.head);
00327     unsigned char status = _status[descno];
00328     free_request(handle);
00329
00330     if (ret < 0)
00331         return ret;
00332
00333     switch (status)
00334     {
00335     case L4VIRTIO_BLOCK_S_OK: return L4_EOK;
00336     case L4VIRTIO_BLOCK_S_IOERR: return -L4_EIO;
00337     case L4VIRTIO_BLOCK_S_UNSUPP: return -L4_ENOSYS;
00338     }
00339
00340     return -L4_EINVAL;
00341 }
00342
00343 void free_request(Handle handle)
00344 {
00345     if (handle.head != Virtqueue::Eoq
00346         && _pending[handle.head].tail != Virtqueue::Eoq)
00347         _queue.free_descriptor(handle.head, _pending[handle.head].tail);
00348     _pending[handle.head].tail = Virtqueue::Eoq;
00349 }
00350
00357 void process_used_queue()
00358 {
00359     for (l4_uint16_t descno = _queue.find_next_used();
00360          descno != Virtqueue::Eoq;
00361          descno = _queue.find_next_used()
00362          )
00363     {
00364         if (descno >= _queue.num() || _pending[descno].tail == Virtqueue::Eoq)
00365             L4Re::chksys(-L4_ENOSYS, "Bad descriptor number");
00366
00367         unsigned char status = _status[descno];
00368         free_request(Handle(descno));
00369
00370         if (_pending[descno].callback)
00371             _pending[descno].callback(status);
00372     }
00373 }
00374
00375 protected:
00376     L4Re::Util::Unique_cap<L4Re::Dataspace> _ds;
00377     L4Re::Cap<L4Re::Dataspace> _queue_ds;
00378
00379 private:
00380     L4Re::Rm::Unique_region<unsigned char *> _queue_region;
00381     l4virtio_block_header_t *_headers;
00382     unsigned char *_status;

```

```

00383     l4_uint64_t _header_addr;
00384     l4_uint64_t _status_addr;
00385     Virtqueue _queue;
00386     std::vector<Request> _pending;
00387 };
00388
00389 } }

```

16.272 virtio-block

```

00001 // vi:ft=cpp
00002 /* SPDX-License-Identifier: MIT */
00003 /*
00004  * Copyright (C) 2017-2021, 2024-2025 Kernkonzept GmbH.
00005  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00006  */
00007 */
00008 #pragma once
00009
00010 #include <l4/cxx/unique_ptr>
00011 #include <l4/re/util/unique_cap>
00012
00013 #include <climits>
00014
00015 #include <l4/l4virtio/virtio.h>
00016 #include <l4/l4virtio/virtio_block.h>
00017 #include <l4/l4virtio/server/l4virtio>
00018 #include <l4/sys/cxx/ipc_epiface>
00019
00020 namespace L4virtio { namespace Svr {
00021
00022     template <typename Ds_data> class Block_dev_base;
00023
00027     template <typename Ds_data>
00028     class Block_request
00029     {
00030     public:
00031         friend class Block_dev_base<Ds_data>;
00032         enum { Header_size = sizeof(l4virtio_block_header_t) };
00033
00034         struct Data_block
00035         {
00036             Driver_mem_region_t<Ds_data> *mem;
00037             void *addr;
00038             l4_uint32_t len;
00039
00040             Data_block() = default;
00041
00042             Data_block(Driver_mem_region_t<Ds_data> *m, Virtqueue::Desc const &desc,
00043                       Request_processor const *)
00044             : mem(m), addr(m->local(desc.addr)), len(desc.len)
00045             {}
00046         };
00047
00048         unsigned data_size() const
00049         {
00050             Request_processor rp;
00051             Data_block data;
00052
00053             rp.start(_mem_list, _request, &data);
00054
00055             unsigned total = data.len;
00056
00057             try
00058             {
00059                 while (rp.has_more())
00060                 {
00061                     rp.next(_mem_list, &data);
00062                     total += data.len;
00063                 }
00064             }
00065             catch (Bad_descriptor const &e)
00066             {
00067                 // need to convert the exception because e contains a raw pointer to rp
00068                 throw L4::Runtime_error(-L4_EIO, "bad virtio descriptor");
00069             }
00070
00071             if (total < Header_size + 1)
00072                 throw L4::Runtime_error(-L4_EIO, "virtio request too short");
00073
00074             return total - Header_size - 1;
00075         }
00076     };
00077
00078     namespace {
00079         struct Data_block {
00080             Driver_mem_region_t<Ds_data> *mem;
00081             void *addr;
00082             l4_uint32_t len;
00083
00084             Data_block() = default;
00085
00086             Data_block(Driver_mem_region_t<Ds_data> *m, Virtqueue::Desc const &desc,
00087                       Request_processor const *)
00088             : mem(m), addr(m->local(desc.addr)), len(desc.len)
00089             {}
00090         };
00091     }
00092
00093     unsigned data_size() const
00094     {
00095         Request_processor rp;
00096         Data_block data;
00097
00098         rp.start(_mem_list, _request, &data);
00099
00100         unsigned total = data.len;
00101
00102         try
00103         {
00104             while (rp.has_more())
00105             {
00106                 rp.next(_mem_list, &data);
00107                 total += data.len;
00108             }
00109         }
00110         catch (Bad_descriptor const &e)
00111         {
00112             // need to convert the exception because e contains a raw pointer to rp
00113             throw L4::Runtime_error(-L4_EIO, "bad virtio descriptor");
00114         }
00115
00116         if (total < Header_size + 1)
00117             throw L4::Runtime_error(-L4_EIO, "virtio request too short");
00118
00119         return total - Header_size - 1;
00120     }
00121 } }

```

```

00090     }
00091
00095     bool has_more()
00096     {
00097         // peek into the remaining data
00098         while (_data.len == 0 && _rp.has_more())
00099             _rp.next(_mem_list, &_data);
00100
00101         // there always must be one byte left for status
00102         return (_data.len > 1 || _rp.has_more());
00103     }
00104
00113     Data_block next_block()
00114     {
00115         Data_block out;
00116
00117         if (_data.len == 0)
00118         {
00119             if (!_rp.has_more())
00120                 throw L4::Runtime_error(-L4_EEXIST,
00121                                         "No more data blocks in virtio request");
00122
00123             if (_todo_blocks == 0)
00124                 throw Bad_descriptor(&_rp, Bad_descriptor::Bad_size);
00125             --_todo_blocks;
00126
00127             _rp.next(_mem_list, &_data);
00128         }
00129
00130         if (_data.len > _max_block_size)
00131             throw Bad_descriptor(&_rp, Bad_descriptor::Bad_size);
00132
00133         out = _data;
00134
00135         if (!_rp.has_more())
00136         {
00137             --(out.len);
00138             _data.len = 1;
00139             _data.addr = static_cast<char *>(_data.addr) + out.len;
00140         }
00141         else
00142             _data.len = 0; // is consumed
00143
00144         return out;
00145     }
00146
00148     l4virtio_block_header_t const &header() const
00149     { return _header; }
00150
00151 private:
00152     Block_request(Virtqueue::Request req, Driver_mem_list_t<Ds_data> *mem_list,
00153                  unsigned max_blocks, l4_uint32_t max_block_size)
00154     : _mem_list(mem_list),
00155       _request(req),
00156       _todo_blocks(max_blocks),
00157       _max_block_size(max_block_size)
00158     {
00159         // read header which should be in the first block
00160         _rp.start(mem_list, _request, &_data);
00161         --_todo_blocks;
00162
00163         if (_data.len < Header_size)
00164             throw Bad_descriptor(&_rp, Bad_descriptor::Bad_size);
00165
00166         _header = *(static_cast<l4virtio_block_header_t *>(_data.addr));
00167         _data.addr = static_cast<char *>(_data.addr) + Header_size;
00168         _data.len -= Header_size;
00169
00170         // if there is no space for status bit we cannot really recover
00171         if (!_rp.has_more() && _data.len == 0)
00172             throw Bad_descriptor(&_rp, Bad_descriptor::Bad_size);
00173     }
00174
00175     int release_request(Virtqueue *queue, l4_uint8_t status, unsigned sz)
00176     {
00177         // write back status
00178         // If there was an error on the way or the status byte is in its
00179         // own block, fast-forward to the last block.
00180         if (_rp.has_more())
00181         {
00182             while (_rp.next(_mem_list, &_data) && _todo_blocks > 0)
00183                 --_todo_blocks;
00184
00185             if (_todo_blocks > 0 && _data.len > 0)
00186                 *(static_cast<l4_uint8_t *>(_data.addr) + _data.len - 1) = status;
00187             else

```

```

00189         return -L4_EIO; // too many data blocks
00190     }
00191     else if (_data.len > 0)
00192         *(static_cast<l4_uint8_t *>(_data.addr)) = status;
00193     else
00194         return -L4_EIO; // no space for final status byte
00195
00196     // now release the head
00197     queue->consumed(_request, sz);
00198
00199     return L4_EOK;
00200 }
00201
00202 Driver_mem_list_t<Ds_data> *_mem_list;
00203 l4virtio_block_header_t _header;
00204 Request_processor _rp;
00205 Data_block _data;
00206
00207 Virtqueue::Request _request;
00208 unsigned _todo_blocks;
00209 l4_uint32_t _max_block_size;
00210 };
00211
00212 struct Block_features : public Dev_config::Features
00213 {
00214     Block_features() = default;
00215     Block_features(l4_uint32_t raw) : Dev_config::Features(raw) {}
00216
00217     CXX_BITFIELD_MEMBER( 1, 1, size_max, raw);
00218     CXX_BITFIELD_MEMBER( 2, 2, seg_max, raw);
00219     CXX_BITFIELD_MEMBER( 4, 4, geometry, raw);
00220     CXX_BITFIELD_MEMBER( 5, 5, ro, raw);
00221     CXX_BITFIELD_MEMBER( 6, 6, blk_size, raw);
00222     CXX_BITFIELD_MEMBER( 9, 9, flush, raw);
00223     CXX_BITFIELD_MEMBER(10, 10, topology, raw);
00224     CXX_BITFIELD_MEMBER(11, 11, config_wce, raw);
00225     CXX_BITFIELD_MEMBER(12, 12, mq, raw);
00226     CXX_BITFIELD_MEMBER(13, 13, discard, raw);
00227     CXX_BITFIELD_MEMBER(14, 14, write_zeroes, raw);
00228 };
00229
00230 template <typename Ds_data>
00231 class Block_dev_base : public L4virtio::Svr::Device_t<Ds_data>
00232 {
00233 private:
00234     L4Re::Util::Unique_cap<L4::Irq> _kick_guest_irq;
00235     Virtqueue _queue;
00236     unsigned _vq_max;
00237     l4_uint32_t _max_block_size = UINT_MAX;
00238     Dev_config_t<l4virtio_block_config_t> _dev_config;
00239
00240 public:
00241     typedef Block_request<Ds_data> Request;
00242
00243 protected:
00244     Block_features negotiated_features() const
00245     { return _dev_config.negotiated_features(0); }
00246
00247     Block_features device_features() const
00248     { return _dev_config.host_features(0); }
00249
00250     void set_device_features(Block_features df)
00251     { _dev_config.host_features(0) = df.raw; }
00252
00253     void set_size_max(l4_uint32_t sz)
00254     {
00255         _dev_config.priv_config()->size_max = sz;
00256         Block_features df = device_features();
00257         df.size_max() = true;
00258         set_device_features(df);
00259
00260         _max_block_size = sz;
00261     }
00262
00263     void set_seg_max(l4_uint32_t sz)
00264     {
00265         _dev_config.priv_config()->seg_max = sz;
00266         Block_features df = device_features();
00267         df.seg_max() = true;
00268         set_device_features(df);
00269     }
00270
00271     void set_geometry(l4_uint16_t cylinders, l4_uint8_t heads, l4_uint8_t sectors)
00272     {
00273         l4virtio_block_config_t volatile *pc = _dev_config.priv_config();
00274         pc->geometry.cylinders = cylinders;

```

```

00319     pc->geometry.heads = heads;
00320     pc->geometry.sectors = sectors;
00321     Block_features df = device_features();
00322     df.geometry() = true;
00323     set_device_features(df);
00324 }
00325
00332 void set_blk_size(l4_uint32_t sz)
00333 {
00334     _dev_config.priv_config()->blk_size = sz;
00335     Block_features df = device_features();
00336     df.blk_size() = true;
00337     set_device_features(df);
00338 }
00339
00348 void set_topology(l4_uint8_t physical_block_exp,
00349                  l4_uint8_t alignment_offset,
00350                  l4_uint32_t min_io_size,
00351                  l4_uint32_t opt_io_size)
00352 {
00353     l4virtio_block_config_t volatile *pc = _dev_config.priv_config();
00354     pc->topology.physical_block_exp = physical_block_exp;
00355     pc->topology.alignment_offset = alignment_offset;
00356     pc->topology.min_io_size = min_io_size;
00357     pc->topology.opt_io_size = opt_io_size;
00358     Block_features df = device_features();
00359     df.topology() = true;
00360     set_device_features(df);
00361 }
00362
00364 void set_flush()
00365 {
00366     Block_features df = device_features();
00367     df.flush() = true;
00368     set_device_features(df);
00369 }
00370
00375 void set_config_wce(l4_uint8_t writeback)
00376 {
00377     l4virtio_block_config_t volatile *pc = _dev_config.priv_config();
00378     pc->writeback = writeback;
00379     Block_features df = device_features();
00380     df.config_wce() = true;
00381     set_device_features(df);
00382 }
00383
00388 l4_uint8_t get_writeback()
00389 {
00390     l4virtio_block_config_t volatile *pc = _dev_config.priv_config();
00391     return pc->writeback;
00392 }
00393
00402 void set_discard(l4_uint32_t max_discard_sectors, l4_uint32_t max_discard_seg,
00403                 l4_uint32_t discard_sector_alignment)
00404 {
00405     l4virtio_block_config_t volatile *pc = _dev_config.priv_config();
00406     pc->max_discard_sectors = max_discard_sectors;
00407     pc->max_discard_seg = max_discard_seg;
00408     pc->discard_sector_alignment = discard_sector_alignment;
00409     Block_features df = device_features();
00410     df.discard() = true;
00411     set_device_features(df);
00412 }
00413
00422 void set_write_zeroes(l4_uint32_t max_write_zeroes_sectors,
00423                      l4_uint32_t max_write_zeroes_seg,
00424                      l4_uint8_t write_zeroes_may_unmap)
00425 {
00426     l4virtio_block_config_t volatile *pc = _dev_config.priv_config();
00427     pc->max_write_zeroes_sectors = max_write_zeroes_sectors;
00428     pc->max_write_zeroes_seg = max_write_zeroes_seg;
00429     pc->write_zeroes_may_unmap = write_zeroes_may_unmap;
00430     Block_features df = device_features();
00431     df.write_zeroes() = true;
00432     set_device_features(df);
00433 }
00434
00435 public:
00444 Block_dev_base(l4_uint32_t vendor, unsigned queue_size, l4_uint64_t capacity,
00445               bool read_only)
00446 : L4virtio::Svr::Device_t<Ds_data>(&_dev_config),
00447   _vq_max(queue_size),
00448   _dev_config(vendor, L4VIRTIO_ID_BLOCK, 1)
00449 {
00450     this->reset_queue_config(0, queue_size);
00451
00452     Block_features df(0);

```

```

00453     df.ring_indirect_desc() = true;
00454     df.ro() = read_only;
00455     set_device_features(df);
00456
00457     _dev_config.set_host_feature(L4VIRTIO_FEATURE_VERSION_1);
00458
00459     _dev_config.priv_config()->capacity = capacity;
00460 }
00461
00465 virtual void reset_device() = 0;
00466
00470 virtual bool queue_stopped() = 0;
00471
00483 void finalize_request(cxx::unique_ptr<Request> req, unsigned sz,
00484                      l4_uint8_t status = L4VIRTIO_BLOCK_S_OK)
00485 {
00486     if (_dev_config.status().fail_state() || !_queue.ready())
00487         return;
00488
00489     if (req->release_request(&_queue, status, sz) < 0)
00490         this->device_error();
00491
00492     if (!_queue.no_notify_guest())
00493         return;
00494
00495     _dev_config.add_irq_status(L4VIRTIO_IRQ_STATUS_VRING);
00496     _kick_guest_irq->trigger();
00497
00498     // Request can be dropped here.
00499 }
00500
00501 int reconfig_queue(unsigned idx) override
00502 {
00503     if (idx == 0 && this->setup_queue(&_queue, 0, _vq_max))
00504         return 0;
00505
00506     return -L4_EINVAL;
00507 }
00508
00509 void reset() override
00510 {
00511     _queue.disable();
00512     _dev_config.reset_queue(0, _vq_max);
00513     _dev_config.reset_hdr();
00514     reset_device();
00515 }
00516
00517 protected:
00518 bool check_for_new_requests()
00519 {
00520     if (!_queue.ready() || queue_stopped())
00521         return false;
00522
00523     if (_dev_config.status().fail_state())
00524         return false;
00525
00526     return _queue.desc_avail();
00527 }
00528
00530 cxx::unique_ptr<Request> get_request()
00531 {
00532     cxx::unique_ptr<Request> req;
00533
00534     if (!_queue.ready() || queue_stopped())
00535         return req;
00536
00537     if (_dev_config.status().fail_state())
00538         return req;
00539
00540     auto r = _queue.next_avail();
00541     if (!r)
00542         return req;
00543
00544     try
00545     {
00546         cxx::unique_ptr<Request> cur{
00547             new Request(r, &(this->_mem_info), _vq_max, _max_block_size)};
00548         req = cxx::move(cur);
00549     }
00550
00551     catch (Bad_descriptor const &e)
00552     {
00553         this->device_error();
00554         return req;
00555     }
00556
00557     return req;

```

```

00558     }
00559
00560 private:
00561     void register_single_driver_irq() override
00562     {
00563         _kick_guest_irq = L4Re::Util::Unique_cap<L4::Irq>(
00564             L4Re::chkcap(this->server_iface()->template rcv_cap<L4::Irq>(0)));
00565
00566         L4Re::chksys(this->server_iface()->realloc_rcv_cap(0));
00567     }
00568
00569     void trigger_driver_config_irq() override
00570     {
00571         _dev_config.add_irq_status(L4VIRTIO_IRQ_STATUS_CONFIG);
00572         _kick_guest_irq->trigger();
00573     }
00574
00575     bool check_queues() override
00576     {
00577         if (!_queue.ready())
00578         {
00579             reset();
00580             return false;
00581         }
00582
00583         return true;
00584     }
00585 };
00586
00587 template <typename Ds_data>
00588 struct Block_dev
00589 : Block_dev_base<Ds_data>,
00590   L4::Epiface_t<Block_dev<Ds_data>, L4virtio::Device>
00591 {
00592 private:
00593     class Irq_object : public L4::Irqep_t<Irq_object>
00594     {
00595     public:
00596         Irq_object(Block_dev<Ds_data> *parent) : _parent(parent) {}
00597
00598         void handle_irq()
00599         {
00600             _parent->kick();
00601         }
00602
00603     private:
00604         Block_dev<Ds_data> *_parent;
00605     };
00606     Irq_object _irq_handler;
00607
00608 protected:
00609     L4::Epiface *irq_iface()
00610     { return &_amp;_irq_handler; }
00611
00612 public:
00613     Block_dev(l4_uint32_t vendor, unsigned queue_size, l4_uint64_t capacity,
00614              bool read_only)
00615     : Block_dev_base<Ds_data>(vendor, queue_size, capacity, read_only),
00616       _irq_handler(this)
00617     {}
00618
00629     L4::Cap<void> register_obj(L4::Registry_iface *registry,
00630                               char const *service = 0)
00631     {
00632         L4Re::chkcap(registry->register_irq_obj(this->irq_iface()));
00633         L4::Cap<void> ret;
00634         if (service)
00635             ret = registry->register_obj(this, service);
00636         else
00637             ret = registry->register_obj(this);
00638         L4Re::chkcap(ret);
00639
00640         return ret;
00641     }
00642
00643     L4::Cap<void> register_obj(L4::Registry_iface *registry,
00644                               L4::Cap<L4::Rcv_endpoint> ep)
00645     {
00646         L4Re::chkcap(registry->register_irq_obj(this->irq_iface()));
00647
00648         return L4Re::chkcap(registry->register_obj(this, ep));
00649     }
00650
00651     typedef Block_request<Ds_data> Request;
00652     virtual bool process_request(cxx::unique_ptr<Request> &&req) = 0;
00653
00654 protected:

```

```

00669 L4::Ipc_svr::Server_iface *server_iface() const override
00670 {
00671     return this->L4::Epiface::server_iface();
00672 }
00673
00674 void kick()
00675 {
00676     for (;;)
00677     {
00678         auto req = this->get_request();
00679         if (!req)
00680             return;
00681         if (!this->process_request(cx::move(req)))
00682             return;
00683     }
00684 }
00685
00686 private:
00687 L4::Cap<L4::Irq> device_notify_irq() const override
00688 {
00689     return L4::cap_cast<L4::Irq>(_irq_handler.obj_cap());
00690 }
00691 };
00692
00693 }

```

16.273 virtio-console

```

00001 // vi:ft=cpp
00002 /* SPDX-License-Identifier: MIT */
00003 /*
00004  * Copyright (C) 2019-2024 Kernkonzept GmbH.
00005  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00006  *           Phillip Raffeck <phillip.raffeck@kernkonzept.com>
00007  *           Steffen Liebergeld <steffen.liebergeld@kernkonzept.com>
00008  *           Jan Klötzke <jan.kloetzke@kernkonzept.com>
00009  */
00010 #pragma once
00011
00012 #include <l4/l4virtio/server/l4virtio>
00013 #include <l4/re/error_helper>
00014
00015 namespace L4virtio { namespace Svr { namespace Console {
00016
00017     struct Features : Dev_config::Features
00018     {
00019     Features() = default;
00020     explicit Features(l4_uint32_t raw) : Dev_config::Features(raw) {}
00021     CXX_BITFIELD_MEMBER(0, 0, console_size, raw);
00022     CXX_BITFIELD_MEMBER(1, 1, console_multiport, raw);
00023     CXX_BITFIELD_MEMBER(2, 2, emerg_write, raw);
00024     };
00025
00026     struct Control_message
00027     {
00028     enum Events
00029     {
00030         Device_ready = 0,
00031         Device_add = 1,
00032         Device_remove = 2,
00033         Port_ready = 3,
00034         Console_port = 4,
00035         Resize = 5,
00036         Port_open = 6,
00037         Port_name = 7,
00038     };
00039
00040     l4_uint32_t id;
00041     l4_uint16_t event;
00042     l4_uint16_t value;
00043
00044     Control_message() {}
00045     Control_message(l4_uint32_t i, l4_uint16_t e, l4_uint16_t v)
00046     : id(i), event(e), value(v) {}
00047     };
00048
00049     struct Control_request
00050     {
00051     Control_message *msg;
00052     l4_uint32_t len;
00053     Driver_mem_region *mem;
00054     };

```



```

00074
00109 struct Port
00110 {
00114     enum Port_status
00115     {
00117         Port_disabled = 0,
00119         Port_added,
00121         Port_ready,
00123         Port_open,
00125         Port_failed,
00127         Port_num_states,
00128     };
00129
00131     enum { Control_queue_size = 0x10 };
00132
00133     Virtqueue tx;
00134     Virtqueue rx;
00135     Port_status status;
00136     Port_status reported_status;
00137     unsigned vq_max;
00138
00139     Port() : status(Port_disabled), vq_max(Control_queue_size) {}
00140     Port(Port const &) = delete;
00141     Port &operator = (Port const &) = delete;
00142
00143     virtual ~Port() = default;
00144
00146     bool is_open() const
00147     { return status == Port_open; }
00148
00150     virtual void reset()
00151     {
00152         status = Port_disabled;
00153         reported_status = Port_disabled;
00154     }
00155
00157     bool queues_ready() const
00158     { return tx.ready() && rx.ready(); }
00159
00161     bool rx_ready() const
00162     { return is_open() && rx.ready(); }
00163
00165     bool tx_ready() const
00166     { return is_open() && tx.ready(); }
00167
00169     struct Transition {
00170         l4_int16_t event;
00171         l4_uint16_t value;
00172         Port_status next;
00173     };
00174
00194     static constexpr Transition
00195     state_transitions[Port_num_states][Port_num_states] =
00196     {
00197         /* reported          current          */
00198
00199         /* Port_disabled */ /* Port_disabled */ {{ -1, 0, Port_disabled },
00200             /* Port_added   */ { Control_message::Device_add, 0,
00201                 Port_added },
00202             /* Port_ready   */ { Control_message::Device_add, 0,
00203                 Port_ready },
00204             /* Port_open    */ { Control_message::Device_add, 0,
00205                 Port_ready },
00206             /* Port_failed  */ { Control_message::Device_add, 0,
00207                 Port_failed }},
00208
00209         /* Port_added      */ /* Port_disabled */ {{ Control_message::Device_remove,
00210             0, Port_disabled },
00211             /* Port_added   */ { -1, 0, Port_added },
00212             /* Port_ready   */ { -1, 0, Port_ready },
00213             /* Port_open    */ { Control_message::Port_open, 1,
00214                 Port_open },
00215             /* Port_failed  */ { -1, 0, Port_failed }},
00216
00217         /* Port_ready      */ /* Port_disabled */ {{ Control_message::Device_remove,
00218             0, Port_disabled },
00219             /* Port_added   */ { -1, 0, Port_added },
00220             /* Port_ready   */ { -1, 0, Port_ready },
00221             /* Port_open    */ { Control_message::Port_open, 1,
00222                 Port_open },
00223             /* Port_failed  */ { -1, 0, Port_failed }},
00224
00225         /* Port_open       */ /* Port_disabled */ {{ Control_message::Port_open, 0,
00226             Port_ready },
00227             /* Port_added   */ { Control_message::Port_open, 0,
00228                 Port_added },
00229             /* Port_ready   */ { Control_message::Port_open, 0,

```

```

00230                                     Port_ready },
00231                                     /* Port_open      */ { -1, 0, Port_open },
00232                                     /* Port_failed    */ { Control_message::Port_open, 0,
00233                                     Port_ready }},
00234
00235     /* Port_failed */ /* Port_disabled */ {{ Control_message::Device_remove,
00236     0, Port_disabled },
00237     /* Port_added */ { -1, 0, Port_added },
00238     /* Port_ready */ { -1, 0, Port_ready },
00239     /* Port_open */ { Control_message::Port_open, 1,
00240     Port_open },
00241     /* Port_failed */ { -1, 0, Port_failed }},
00242 };
00243 };
00244
00267 class Virtio_con : public L4virtio::Svr::Device
00268 {
00269     enum Virtqueue_names
00270     {
00271         Ctrl_rx = 2,
00272         Ctrl_tx = 3,
00273     };
00274
00275     struct Serial_config_space
00276     {
00277         l4_uint16_t cols;
00278         l4_uint16_t rows;
00279         l4_uint32_t max_nr_ports;
00280         l4_uint32_t emerg_wr;
00281     } __attribute__((packed));
00282
00283 public:
00293     explicit Virtio_con(unsigned max_ports, bool enable_multiport)
00294     : L4virtio::Svr::Device(&_dev_config),
00295       _num_ports(enable_multiport ? max_ports : 1),
00296       _dev_config(L4VIRTIO_VENDOR_KK, L4VIRTIO_ID_CONSOLE,
00297         enable_multiport ? max_ports * 2 + 2 : 2)
00298     {
00299         if (_num_ports < 1)
00300             L4Re::chksys(-L4_EINVAL, "At least one port is required.");
00301
00302         Features hf(0);
00303
00304         hf.console_multiport() = enable_multiport;
00305
00306         _dev_config.host_features(0) = hf.raw;
00307
00308         if (enable_multiport)
00309             _dev_config.priv_config()->max_nr_ports = _num_ports;
00310         _dev_config.reset_hdr();
00311     }
00312
00313     void reset_queue_configs()
00314     {
00315         for (unsigned q = 0; q < _dev_config.num_queues(); ++q)
00316             reset_queue_config(q, max_queue_size(q));
00317     }
00318
00319     int reconfig_queue(unsigned index) override
00320     {
00321         if (index >= _dev_config.num_queues())
00322             return -L4_ERANGE;
00323
00324         if (setup_queue(get_queue(index), index, max_queue_size(index)))
00325             return 0;
00326
00327         return -L4_EINVAL;
00328     }
00329
00334     bool multiport_enabled() const
00335     {
00336         return _negotiated_features.console_multiport()
00337             && _dev_config.num_queues() > Ctrl_rx;
00338     }
00339
00340     bool ctrl_queue_ready() const
00341     { return _ctrl_port.is_open(); }
00342
00343     bool check_features(void) override
00344     {
00345         _negotiated_features = Features(_dev_config.negotiated_features(0));
00346         return true;
00347     }
00348
00349     bool check_queues() override
00350     {
00351         // NOTE

```

```

00352 // The VIRTIO specification states:
00353 // "The port 0 receive and transmit queues always exist"
00354 // The linux driver however does not setup port 0 if the multiport feature
00355 // is negotiated.
00356 // We just go along with the linux driver and do not expect port 0 to be up,
00357 // if the multiport feature is negotiated.
00358
00359 if (multiport_enabled())
00360     // If MULTIPORT was negotiated, ctrl queues should be set up.
00361     return _ctrl_port.queues_ready();
00362
00363 // If MULTIPORT was not negotiated, port 0 should be set up.
00364 port(0)->status = Port::Port_open;
00365 return port(0)->queues_ready();
00366 }
00367
00379 int port_add(unsigned idx)
00380 {
00381     Port *p = port(idx);
00382
00383     if (p->status != Port::Port_disabled)
00384         return -L4_EPERM;
00385
00386     p->status = Port::Port_added;
00387     port_report_status(idx);
00388
00389     return L4_EOK;
00390 }
00391
00403 int port_remove(unsigned idx)
00404 {
00405     Port *p = port(idx);
00406
00407     if (p->status == Port::Port_disabled)
00408         return -L4_EPERM;
00409
00410     p->status = Port::Port_disabled;
00411     port_report_status(idx);
00412
00413     return L4_EOK;
00414 }
00415
00428 int port_open(unsigned idx, bool open)
00429 {
00430     Port *p = port(idx);
00431
00432     if ((open && p->status != Port::Port_ready)
00433         || (!open && p->status != Port::Port_open))
00434         return -L4_EPERM;
00435
00436     p->status = open ? Port::Port_open : Port::Port_ready;
00437     port_report_status(idx);
00438
00439     return L4_EOK;
00440 }
00441
00455 int port_name(unsigned idx, char const *name)
00456 {
00457     Port *p = port(idx);
00458
00459     if (p->status == Port::Port_disabled)
00460         return -L4_EPERM;
00461
00462     return send_control_message(idx, Control_message::Port_name, 0, name);
00463 }
00464
00487 int send_control_message(l4_uint32_t idx, l4_uint16_t event,
00488                        l4_uint16_t value = 0, const char *name = 0)
00489 {
00490     if (!ctrl_queue_ready())
00491         return -L4_ENODEV;
00492
00493     Virtqueue *q = &_ctrl_port.rx;
00494     if (!q->ready())
00495         return -L4_ENODEV;
00496
00497     Virtqueue::Request r = q->next_avail();
00498     if (!r)
00499         return -L4_EBUSY;
00500
00501     Request_processor rp;
00502     Control_request req;
00503     rp.start(this, r, &req);
00504
00505     if (req.len < sizeof(Control_message))
00506         return -L4_ENOMEM;
00507

```

```

00508     Control_message msg(idx, event, value);
00509
00510     memcpy(req.msg, &msg, sizeof(msg));
00511
00512     if (event == Control_message::Port_name && name)
00513     {
00514         size_t name_len = cxx::min(req.len - sizeof(msg), strlen(name));
00515         memcpy(reinterpret_cast<char*>(req.msg) + sizeof(msg), name, name_len);
00516         q->finish(r, this, sizeof(msg) + name_len);
00517     }
00518     else
00519         q->finish(r, this, sizeof(msg));
00520
00521     return L4_EOK;
00522 }
00523
00536 int handle_control_message()
00537 {
00538     // Report port state transitions if that failed in the past...
00539     if (_report_port_state)
00540     {
00541         _report_port_state = false;
00542
00543         for (unsigned i = 0; i < _num_ports; ++i)
00544             if (!port_report_status(i))
00545                 _report_port_state = true;
00546     }
00547
00548     Virtqueue *q = &ctrl_port.tx;
00549     if (!q->ready())
00550         return -L4_ENODEV;
00551
00552     int ret = L4_EOK;
00553     Virtqueue::Request r;
00554     while ((r = q->next_avail()))
00555     {
00556         Request_processor rp;
00557         Control_request req;
00558
00559         rp.start(this, r, &req);
00560
00561         Control_message msg;
00562         if (req.len < sizeof(msg))
00563         {
00564             // Just ignore malformed input.
00565             q->finish(r, this);
00566             ret = -L4_EINVAL;
00567             continue;
00568         }
00569
00570         memcpy(&msg, req.msg, sizeof(msg));
00571         q->finish(r, this);
00572
00573         if (_ctrl_port.status == Port::Port_disabled)
00574         {
00575             // When the control queue is disabled, only device ready is accepted.
00576             if (msg.event == Control_message::Device_ready)
00577             {
00578                 if (msg.value)
00579                     _ctrl_port.status = Port::Port_open;
00580             }
00581
00582             process_device_ready(msg.value);
00583             continue;
00584         }
00585
00586         if (!ctrl_queue_ready())
00587             continue;
00588
00589         // Ignore invalid port ids
00590         if (msg.id >= max_ports())
00591             break;
00592
00593         switch (msg.event)
00594         {
00595             case Control_message::Port_ready:
00596                 process_port_ready(msg.id, msg.value);
00597                 break;
00598             case Control_message::Port_open:
00599                 process_port_open(msg.id, msg.value);
00600                 break;
00601             default:
00602                 ret = -L4_EINVAL;
00603                 break;
00604         }
00605     }
00606

```

```

00607     return ret;
00608 }
00609
00611 void load_desc(L4virtio::Virtqueue::Desc const &desc,
00612               Request_processor const *proc,
00613               L4virtio::Virtqueue::Desc const **table)
00614 {
00615     this->_mem_info.load_desc(desc, proc, table);
00616 }
00617
00619 void load_desc(L4virtio::Virtqueue::Desc const &desc,
00620               Request_processor const *proc,
00621               Control_request *data)
00622 {
00623     auto *region = this->_mem_info.find(desc.addr.get(), desc.len);
00624     if (L4_UNLIKELY(!region))
00625         throw Bad_descriptor(proc, Bad_descriptor::Bad_address);
00626
00627     data->msg = reinterpret_cast<Control_message *>(region->local(desc.addr));
00628     data->len = desc.len;
00629     data->mem = region;
00630 }
00631
00632 void reset() override
00633 {
00634     for (unsigned p = 0; p < _num_ports; ++p)
00635         port(p)->reset();
00636
00637     _ctrl_port.reset();
00638     reset_queue_configs();
00639     _dev_config.reset_hdr();
00640     _negotiated_features = Features(0);
00641     _report_port_state = false;
00642
00643     reset_device();
00644 }
00645
00652 virtual void reset_device() {}
00653
00663 virtual void notify_queue(Virtqueue *queue) = 0;
00664
00672 virtual Port *port(unsigned port) = 0;
00673 virtual Port const *port(unsigned port) const = 0;
00674
00685 virtual void process_device_ready(l4_uint16_t value) = 0;
00686
00698 virtual void process_port_ready(l4_uint32_t id, l4_uint16_t value)
00699 {
00700     Port *p = port(id);
00701
00702     switch (p->status)
00703     {
00704     case Port::Port_added:
00705     case Port::Port_ready:
00706         p->status = value ? Port::Port_ready : Port::Port_failed;
00707         break;
00708     case Port::Port_open:
00709         if (!value)
00710             p->status = Port::Port_failed;
00711         break;
00712     default:
00713         // invalid state for PORT_READY message
00714         break;
00715     }
00716 }
00717
00728 virtual void process_port_open(l4_uint32_t id, l4_uint16_t value) = 0;
00729
00730 unsigned max_ports() const
00731 { return _num_ports; }
00732
00733 private:
00734 bool is_control_queue(unsigned q) const
00735 { return q == Ctrl_rx || q == Ctrl_tx; }
00736
00737 unsigned queue_to_port(unsigned q) const
00738 { return (q == 0 || q == 1) ? 0 : (q / 2) - 1; }
00739
00748 unsigned max_queue_size(unsigned q) const
00749 {
00750     if (is_control_queue(q))
00751         return _ctrl_port.vq_max;
00752
00753     return port(queue_to_port(q))->vq_max;
00754 }
00755
00764 Virtqueue *get_queue(unsigned q)

```

```

00765 {
00766     Port *p;
00767     if (is_control_queue(q))
00768         p = &_ctrl_port;
00769     else
00770         p = port(queue_to_port(q));
00771
00772     if (q & 1)
00773         return &p->tx;
00774     else
00775         return &p->rx;
00776 }
00777
00778 bool port_report_status(unsigned idx)
00779 {
00780     Port *p = port(idx);
00781     while (p->status != p->reported_status)
00782     {
00783         auto const &trans
00784             = Port::state_transitions[p->reported_status][p->status];
00785
00786         if (trans.event >= 0
00787             && send_control_message(idx, trans.event, trans.value) < 0)
00788         {
00789             _report_port_state = true;
00790             return false;
00791         }
00792
00793         p->reported_status = trans.next;
00794     }
00795
00796     return true;
00797 }
00798
00799 unsigned _num_ports;
00800 bool _report_port_state = false;
00801
00802 protected:
00803     Dev_config_t<Serial_config_space> _dev_config;
00804     Port _ctrl_port;
00805     Features _negotiated_features{0};
00806 };
00807
00808 }}} // name space

```

16.274 virtio-console-device

```

00001 // vi:ft=cpp
00002 /* SPDX-License-Identifier: MIT */
00003 /*
00004  * Copyright (C) 2019-2024 Kernkonzept GmbH.
00005  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00006  *            Phillip Raffeck <phillip.raffeck@kernkonzept.com>
00007  *            Steffen Liebergeld <steffen.liebergeld@kernkonzept.com>
00008  *            Jan Klötzke <jan.kloetzke@kernkonzept.com>
00009  */
00010 #pragma once
00011
00012 #include <l4/cxx/bitmap>
00013 #include <l4/cxx/static_vector>
00014 #include <l4/l4virtio/server/l4virtio>
00015 #include <l4/l4virtio/server/virtio-console>
00016 #include <l4/re/error_helper>
00017
00018 namespace L4virtio { namespace Svr { namespace Console {
00019
00020 struct Device_port : public Port
00021 {
00022     struct Buffer : Data_buffer
00023     {
00024         Buffer() = default;
00025         Buffer(Driver_mem_region const *r,
00026              Virtqueue::Desc const &d,
00027              Request_processor const *)
00028         {
00029             pos = static_cast<char *>(r->local(d.addr));
00030             left = d.len;
00031         }
00032     };
00033
00034     Request_processor rp;
00035     Virtqueue::Request request;
00036     Buffer src;

```

```

00043
00044     bool poll_in_req = true;
00045     bool poll_out_req = true;
00046
00047     void reset() override
00048     {
00049         Port::reset();
00050         request = Virtqueue::Request();
00051         poll_in_req = true;
00052         poll_out_req = true;
00053     }
00054 };
00055
00118 class Device
00119 : public Virtio_con
00120 {
00121     class Irq_object : public L4::Irqep_t<Irq_object>
00122     {
00123     public:
00124         Irq_object(Device *parent) : _parent(parent) {}
00125
00126         void handle_irq() { _parent->kick(); }
00127
00128     private:
00129         Device *_parent;
00130     };
00131
00132 protected:
00133     L4::Epiface *irq_iface()
00134     { return &_irq_handler; }
00135
00136 public:
00145     explicit Device(unsigned vq_max)
00146     : Virtio_con(1, false),
00147       _irq_handler(this),
00148       _ports(cxx::make_unique<Device_port[]>(1))
00149     {
00150         _ports[0].vq_max = vq_max;
00151         reset_queue_configs();
00152     }
00153
00163     explicit Device(unsigned vq_max, unsigned ports)
00164     : Virtio_con(ports, true),
00165       _irq_handler(this),
00166       _ports(cxx::make_unique<Device_port[]>(ports))
00167     {
00168         for (unsigned i = 0; i < ports; ++i)
00169             _ports[i].vq_max = vq_max;
00170         reset_queue_configs();
00171     }
00172
00182     explicit Device(cxx::static_vector<unsigned> const &vq_max_nums)
00183     : Virtio_con(vq_max_nums.size(), true),
00184       _irq_handler(this),
00185       _ports(cxx::make_unique<Device_port[]>(max_ports()))
00186     {
00187         for (unsigned i = 0; i < vq_max_nums.size(); ++i)
00188             _ports[i].vq_max = vq_max_nums[i];
00189         reset_queue_configs();
00190     }
00191
00192     void register_single_driver_irq() override
00193     {
00194         _kick_driver_irq = L4Re::Util::Unique_cap<L4::Irq>(
00195             L4Re::chkcapi(server_iface()->rcv_cap<L4::Irq>(0)));
00196         L4Re::chksys(server_iface()->realloc_rcv_cap(0));
00197     }
00198
00199     L4::Cap<L4::Irq> device_notify_irq() const override
00200     { return _irq_handler.obj_cap(); }
00201
00202     void notify_queue(Virtqueue *queue) override
00203     {
00204         if (queue->no_notify_guest())
00205             return;
00206
00207         _dev_config.add_irq_status(L4VIRTIO_IRQ_STATUS_VRING);
00208         _kick_driver_irq->trigger();
00209     }
00210
00214     virtual void rx_data_available(unsigned port) = 0;
00215
00219     virtual void tx_space_available(unsigned port) = 0;
00220
00224     virtual bool queues_stopped()
00225     { return false; }
00226

```

```

00227 void trigger_driver_config_irq() override
00228 {
00229     _dev_config.add_irq_status(L4VIRTIO_IRQ_STATUS_CONFIG);
00230     _kick_driver_irq->trigger();
00231 }
00232
00233 void kick()
00234 {
00235     if (queues_stopped())
00236         return;
00237
00238     // We're not interested in logging any errors, just ignore return value.
00239     handle_control_message();
00240
00241     for (unsigned i = 0; i < max_ports(); ++i)
00242     {
00243         auto &p = _ports[i];
00244         if (p.poll_in_req && p.tx_ready() && p.tx.desc_avail())
00245         {
00246             p.poll_in_req = false;
00247             rx_data_available(i);
00248         }
00249
00250         if (p.poll_out_req && p.rx_ready() && p.rx.desc_avail())
00251         {
00252             p.poll_out_req = false;
00253             tx_space_available(i);
00254         }
00255     }
00256 }
00257
00272 unsigned port_read(char *buf, unsigned len, unsigned port = 0)
00273 {
00274     unsigned total = 0;
00275     Device_port &p = _ports[port];
00276     Virtqueue *q = &p.tx;
00277
00278     Data_buffer dst;
00279     dst.pos = buf;
00280     dst.left = len;
00281
00282     while (dst.left)
00283     {
00284         try
00285         {
00286             // Make sure we have a valid request where we can read data from
00287             if (!p.request.valid())
00288             {
00289                 p.request = p.tx_ready() ? q->next_avail()
00290                     : Virtqueue::Request();
00291                 if (!p.request.valid())
00292                     break;
00293
00294                 p.rp.start(mem_info(), p.request, &p.src);
00295             }
00296
00297             total += p.src.copy_to(&dst);
00298
00299             // We might have eaten up the current descriptor. Move to the next
00300             // if this is the case. At the end of the descriptor chain we have
00301             // to retire the current request altogether.
00302             if (!p.src.left)
00303             {
00304                 if (!p.rp.next(mem_info(), &p.src))
00305                 {
00306                     q->finish(p.request, this);
00307                     p.request = Virtqueue::Request();
00308                 }
00309             }
00310         }
00311         catch (Bad_descriptor const &)
00312         {
00313             q->finish(p.request, this);
00314             p.request = Virtqueue::Request();
00315             device_error();
00316             break;
00317         }
00318     }
00319
00320     if (total < len)
00321         p.poll_in_req = true;
00322
00323     return total;
00324 }
00325
00341 unsigned port_write(char const *buf, unsigned len, unsigned port = 0)
00342 {

```



```

00343     unsigned total = 0;
00344     Device_port &p = _ports[port];
00345     Virtqueue *q = &p.rx;
00346
00347     Data_buffer src;
00348     src.pos = const_cast<char*>(buf);
00349     src.left = len;
00350
00351     Request_processor rp;
00352     while (src.left)
00353     {
00354         auto r = p.rx_ready() ? q->next_avail() : Virtqueue::Request();
00355         if (!r.valid())
00356             break;
00357
00358         l4_uint32_t chunk = 0;
00359         try
00360         {
00361             Device_port::Buffer dst;
00362             rp.start(mem_info(), r, &dst);
00363
00364             for (;;)
00365             {
00366                 chunk += src.copy_to(&dst);
00367                 if (!src.left)
00368                     break;
00369                 if (!rp.next(mem_info(), &dst))
00370                     break;
00371             }
00372         }
00373         catch (Bad_descriptor const &)
00374         {
00375             device_error();
00376         }
00377
00378         q->finish(r, this, chunk);
00379         total += chunk;
00380     }
00381
00382     if (total < len)
00383         p.poll_out_req = true;
00384
00385     return total;
00386 }
00387
00399 void process_device_ready(l4_uint16_t value) override
00400 {
00401     if (!value)
00402         return;
00403
00404     for (unsigned i = 0; i < max_ports(); ++i)
00405         port_add(i);
00406 }
00407
00422 void process_port_ready(l4_uint32_t id, l4_uint16_t value) override
00423 {
00424     Virtio_con::process_port_ready(id, value);
00425
00426     Port *p = port(id);
00427     if (p->status == Port::Port_failed)
00428         port_remove(id);
00429     else if (p->status == Port::Port_ready)
00430         port_open(id, true);
00431 }
00432
00443 void process_port_open(l4_uint32_t id, l4_uint16_t value) override
00444 {
00445     static_cast<void>(id);
00446     static_cast<void>(value);
00447 }
00448
00449 protected:
00450 Port* port(unsigned idx) override
00451 {
00452     return &_ports[idx];
00453 }
00454
00455 Port const *port(unsigned idx) const override
00456 {
00457     return &_ports[idx];
00458 }
00459
00460 private:
00461 Irq_object _irq_handler;
00462 cxx::unique_ptr<Device_port[]> _ports;
00463 L4Re::Util::Unique_cap<L4::Irq> _kick_driver_irq;
00464 };

```

```
00465
00466 }}} // name space
```

16.275 virtio-gpio-device

```
00001 // vi:set ft=c++: -- Mode: C++ --
00002 /*
00003  * Copyright (C) 2025 Kernkonzept GmbH.
00004  * Author(s): Christian Pöttsch <christian.poetzsch@kernkonzept.com>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008
00009 #pragma once
00010
00011 #include <l4/re/error_helper>
00012 #include <l4/sys/cxx/ipc_epiface>
00013
00014 #include <l4/l4virtio/server/virtio>
00015 #include <l4/l4virtio/server/l4virtio>
00016 #include <l4/l4virtio/l4virtio>
00017
00018 #include <l4/re/error_helper>
00019 #include <l4/re/util/object_registry>
00020 #include <l4/re/util/br_manager>
00021 #include <l4/sys/cxx/ipc_epiface>
00022 #include <l4/cxx/pair>
00023
00024 #include <vector>
00025 #include <memory>
00026
00027 namespace L4virtio {
00028 namespace Svr {
00029
00030 /* GPIO message status types */
00031 enum : l4_uint8_t
00032 {
00033     Gpio_status_ok = 0x0,
00034     Gpio_status_err = 0x1
00035 };
00036
00037 /* GPIO message types */
00038 enum : l4_uint8_t
00039 {
00040     Gpio_msg_get_line_names = 0x1,
00041     Gpio_msg_get_direction = 0x2,
00042     Gpio_msg_set_direction = 0x3,
00043     Gpio_msg_get_value = 0x4,
00044     Gpio_msg_set_value = 0x5,
00045     Gpio_msg_set_irq_type = 0x6
00046 };
00047
00048 /* GPIO value types */
00049 enum : l4_uint8_t
00050 {
00051     Gpio_low = 0x0,
00052     Gpio_high = 0x1
00053 };
00054
00055 /* GPIO direction types */
00056 enum : l4_uint8_t
00057 {
00058     Gpio_direction_none = 0x0,
00059     Gpio_direction_out = 0x1,
00060     Gpio_direction_in = 0x2
00061 };
00062
00063 /* GPIO interrupt types */
00064 enum : l4_uint8_t
00065 {
00066     Gpio_irq_type_none = 0x0,
00067     Gpio_irq_type_edge_rising = 0x1,
00068     Gpio_irq_type_edge_falling = 0x2,
00069     Gpio_irq_type_edge_both = 0x3,
00070     Gpio_irq_type_level_high = 0x4,
00071     Gpio_irq_type_level_low = 0x8
00072 };
00073
00074 /* GPIO interrupt status types */
00075 enum : l4_uint8_t
00076 {
00077     Gpio_irq_status_invalid = 0x0,
00078     Gpio_irq_status_valid = 0x1
```

```

00079 };
00080
00081 struct Gpio_request
00082 {
00083     l4_uint16_t type;
00084     l4_uint16_t gpio;
00085     l4_uint32_t value;
00086 };
00087 static_assert(sizeof(Gpio_request) == 8,
00088     "Gpio_request contains padding bytes.");
00089
00090 struct Gpio_response
00091 {
00092     l4_uint8_t status;
00093     l4_uint8_t value;
00094 };
00095 static_assert(sizeof(Gpio_response) == 2,
00096     "Gpio_response contains padding bytes.");
00097
00098 struct Gpio_irq_request
00099 {
00100     l4_uint16_t gpio;
00101 };
00102 static_assert(sizeof(Gpio_irq_request) == 2,
00103     "Gpio_irq_request contains padding bytes.");
00104
00105 struct Gpio_irq_response
00106 {
00107     l4_uint8_t status;
00108 };
00109 static_assert(sizeof(Gpio_irq_response) == 1,
00110     "Gpio_irq_response contains padding bytes.");
00111
00112 struct Gpio_request_msg
00113 {
00114     struct Gpio_request *in_hdr = nullptr;
00115     struct Gpio_response *out_hdr = nullptr;
00116 };
00117
00118 struct Gpio_irq_request_msg
00119 {
00120     struct Gpio_irq_request *in_hdr = nullptr;
00121     struct Gpio_irq_response *out_hdr = nullptr;
00122 };
00123
00140 template <typename Request_handler,
00141     typename Epiface = L4virtio::Device>
00142 class Virtio_gpio : public L4virtio::Svr::Device,
00143     public L4::Epiface_t<Virtio_gpio<Request_handler,
00144         Epiface>,
00145         Epiface>
00146 {
00147 private:
00148     enum
00149     {
00150         queue_size = 128,
00151     };
00152
00153 public:
00154     using Gpio_request_handler = Request_handler;
00155
00166     struct Irq_handler
00167     {
00168         Irq_handler(Virtio_gpio *gpio, L4virtio::Svr::Virtqueue *q,
00169             L4virtio::Svr::Virtqueue::Head_desc const &head,
00170             l4_uint8_t *status)
00171             : _gpio(gpio), _q(q), _head(head), _status(status)
00172         {}
00173
00174         void handle_irq()
00175         {
00176             *_status = Gpio_irq_status_valid;
00177             _q->finish(_head, _gpio, sizeof(Gpio_irq_response));
00178         }
00179
00180         void cancel()
00181         {
00182             *_status = Gpio_irq_status_invalid;
00183             _q->finish(_head, _gpio, sizeof(Gpio_irq_response));
00184         }
00185
00186 private:
00187     Virtio_gpio *_gpio;
00188     L4virtio::Svr::Virtqueue *_q;
00189     L4virtio::Svr::Virtqueue::Head_desc _head;
00190     l4_uint8_t *_status;
00191 };

```

```

00192
00198 struct Host_irq : L4::Irqep_t<Host_irq>
00199 {
00200     explicit Host_irq(Virtio_gpio *gpio)
00201     : L4::Irqep_t<Host_irq>(), _gpio(gpio) {}
00202
00203     void handle_irq()
00204     { _gpio->handle_queue(); }
00205
00206 private:
00207     Virtio_gpio *_gpio;
00208 };
00209
00213 struct Request_processor : L4virtio::Svr::Request_processor
00214 {
00215     Request_processor(L4virtio::Svr::Virtqueue *q, Gpio_request_handler *hdlr,
00216                     Virtio_gpio *gpio)
00217     : _q(q), _req_handler(hdlr), _gpio(gpio), _head(), _req()
00218     {}
00219
00220 protected:
00221     bool init_queue()
00222     {
00223         auto r = _q->next_avail();
00224
00225         if (L4_UNLIKELY(!r))
00226             return false;
00227
00228         _head = start(_gpio->mem_info(), r, &_req);
00229
00230         return true;
00231     }
00232
00233 template <typename T>
00241 T get_request()
00242 {
00243     T req;
00244     req.in_hdr = reinterpret_cast<decltype(T::in_hdr)>(_req.pos);
00245
00246     // Need the next output buffer.
00247     if (!next(_gpio->mem_info(), &_req) || !current_flags().write())
00248         return req;
00249
00250     req.out_hdr = reinterpret_cast<decltype(T::out_hdr)>(_req.pos);
00251
00252     return req;
00253 }
00254
00255 struct Data_buffer : public L4virtio::Svr::Data_buffer
00256 {
00257     Data_buffer()
00258     {
00259         pos = nullptr;
00260         left = 0;
00261     }
00262     // This constructor is called from within start, so make it available.
00263     Data_buffer(L4virtio::Svr::Driver_mem_region const *r,
00264               L4virtio::Svr::Virtqueue::Desc const &d,
00265               L4virtio::Svr::Request_processor const *)
00266     {
00267         pos = static_cast<char *>(r->local(d.addr));
00268         left = d.len;
00269     }
00270
00271 };
00272
00273 L4virtio::Svr::Virtqueue *_q;
00274 Gpio_request_handler *_req_handler;
00275 Virtio_gpio *_gpio;
00276 L4virtio::Svr::Virtqueue::Head_desc _head;
00277 Data_buffer _req;
00278 };
00279
00280 // Handler for the gpio request queue
00281 struct Req_processor : Request_processor
00282 {
00283     using Request_processor::Request_processor;
00284
00285     void handle_request()
00286     {
00287         if (!this->_head)
00288             if (!this->init_queue())
00289                 return;
00290
00291         using Consumed_entry =
00292             cxx::Pair<L4virtio::Svr::Virtqueue::Head_desc, l4_uint32_t>;
00293         std::vector<Consumed_entry> consumed;

```

```

00294
00295     for (;;)
00296     {
00297         Gpio_request_msg req = this->template get_request<Gpio_request_msg>();
00298         if (!req.in_hdr || !req.out_hdr)
00299         {
00300             this->_gpio->device_error();
00301             break;
00302         }
00303
00304         // default response is error
00305         req.out_hdr->status = Gpio_status_err;
00306         switch (req.in_hdr->type)
00307         {
00308             case Gpio_msg_get_line_names:
00309                 // we don't support this
00310                 break;
00311             case Gpio_msg_get_direction:
00312             {
00313                 if (this->_req_handler->get_direction(req.in_hdr->gpio,
00314                                                     &req.out_hdr->value))
00315                     req.out_hdr->status = Gpio_status_ok;
00316                 break;
00317             }
00318             case Gpio_msg_set_direction:
00319             {
00320                 if (req.in_hdr->value == Gpio_direction_none ||
00321                     req.in_hdr->value == Gpio_direction_out ||
00322                     req.in_hdr->value == Gpio_direction_in)
00323                 {
00324                     if (this->_req_handler->set_direction(req.in_hdr->gpio,
00325                                                         req.in_hdr->value))
00326                         req.out_hdr->status = Gpio_status_ok;
00327                 }
00328                 break;
00329             }
00330             case Gpio_msg_get_value:
00331             {
00332                 if (this->_req_handler->get_value(req.in_hdr->gpio,
00333                                                  &req.out_hdr->value))
00334                     req.out_hdr->status = Gpio_status_ok;
00335                 break;
00336             }
00337             case Gpio_msg_set_value:
00338             {
00339                 if (req.in_hdr->value == Gpio_low ||
00340                     req.in_hdr->value == Gpio_high)
00341                 {
00342                     if (this->_req_handler->set_value(req.in_hdr->gpio,
00343                                                      req.in_hdr->value))
00344                         req.out_hdr->status = Gpio_status_ok;
00345                 }
00346                 break;
00347             }
00348             case Gpio_msg_set_irq_type:
00349             {
00350                 if (req.in_hdr->value == Gpio_irq_type_none ||
00351                     req.in_hdr->value == Gpio_irq_type_edge_rising ||
00352                     req.in_hdr->value == Gpio_irq_type_edge_falling ||
00353                     req.in_hdr->value == Gpio_irq_type_edge_both ||
00354                     req.in_hdr->value == Gpio_irq_type_level_high ||
00355                     req.in_hdr->value == Gpio_irq_type_level_low)
00356                 {
00357                     if (this->_req_handler->set_irq_type(req.in_hdr->gpio,
00358                                                         req.in_hdr->value))
00359                         req.out_hdr->status = Gpio_status_ok;
00360                 }
00361                 break;
00362             }
00363         }
00364
00365         // Save the descriptors which are done
00366         consumed.emplace_back(this->_head, sizeof(Gpio_response));
00367
00368         if (!this->init_queue())
00369             break;
00370     }
00371
00372     // Put all finished descriptors back into the used list and notify the
00373     // driver.
00374     this->_q->finish(consumed.begin(), consumed.end(), this->_gpio);
00375
00376     this->_head = Virtqueue::Head_desc();
00377 }
00378 };
00379
00380 // Handler for the gpio event queue

```

```

00381 struct Irq_req_processor : Request_processor
00382 {
00383     using Request_processor::Request_processor;
00384
00385     void handle_request()
00386     {
00387         if (!this->_head)
00388             if (!this->init_queue())
00389                 return;
00390
00391         for (;;)
00392         {
00393             // There is only one type of message in the event queue. This
00394             // basically arms (unmask) the irq.
00395             Gpio_irq_request_msg req = this->template get_request<Gpio_irq_request_msg>();
00396             if (!req.in_hdr || !req.out_hdr)
00397             {
00398                 this->_gpio->device_error();
00399                 break;
00400             }
00401
00402             // Save the virtio descriptor for this event in an extra Irq_handler
00403             // object. The descriptor will be returned to the client when the irq
00404             // is triggered or canceled.
00405             this->_req_handler->enable_irq(req.in_hdr->gpio,
00406                 std::make_shared<Irq_handler>(this->_gpio,
00407                     this->_q,
00408                     this->_head,
00409                     &req.out_hdr->status));
00410
00411             if (!this->init_queue())
00412                 break;
00413         }
00414
00415         this->_head = Virtqueue::Head_desc();
00416     }
00417 };
00418
00419 struct Features : public L4virtio::Svr::Dev_config::Features
00420 {
00421     Features() = default;
00422     Features(l4_uint32_t raw) : L4virtio::Svr::Dev_config::Features(raw) {}
00423
00424     CXX_BITFIELD_MEMBER(0, 0, gpio_f_irq, raw);
00425 };
00426
00427 struct Gpio_config_space
00428 {
00429     l4_uint16_t ngpio;
00430     l4_uint8_t padding[2];
00431     l4_uint32_t gpio_names_size;
00432 };
00433
00434 Virtio_gpio(Gpio_request_handler *hndlr,
00435             L4Re::Util::Object_registry *registry,
00436             l4_uint16_t ngpio)
00437 : L4virtio::Svr::Device(&_dev_config),
00438   _registry(registry),
00439   _dev_config(L4VIRTIO_VENDOR_KK, L4VIRTIO_ID_GPIO, 2),
00440   _host_irq(this),
00441   _req_processor(&_q[0], hndlr, this),
00442   _irq_req_processor(&_q[1], hndlr, this)
00443 {
00444     init_mem_info(2);
00445
00446     for (size_t i = 0; i < 2; i++)
00447     {
00448         reset_queue_config(i, queue_size);
00449         setup_queue(&_q[i], i, queue_size);
00450     }
00451
00452     registry->register_irq_obj(&_host_irq);
00453
00454     Features hf(0);
00455     hf.ring_indirect_desc() = true;
00456     hf.gpio_f_irq() = true;
00457     _dev_config.host_features(0) = hf.raw;
00458     _dev_config.set_host_feature(L4VIRTIO_FEATURE_VERSION_1);
00459
00460     // fill gpio config space
00461     _dev_config.priv_config()->ngpio = ngpio;
00462     _dev_config.priv_config()->gpio_names_size = 0; // not supported
00463
00464     _dev_config.reset_hdr();
00465 }
00466
00467 ~Virtio_gpio()

```

```

00468 { _registry->unregister_obj(&_host_irq); }
00469
00470 void notify_queue(L4virtio::Svr::Virtqueue *queue)
00471 {
00472     if (queue->no_notify_guest())
00473         return;
00474
00475     _dev_config.add_irq_status(L4VIRTIO_IRQ_STATUS_VRING);
00476     L4Re::chkipc(_notify_guest_irq->trigger(), "trigger guest irq");
00477 }
00478
00479 void handle_queue()
00480 {
00481     _req_processor.handle_request();
00482     _irq_req_processor.handle_request();
00483 }
00484
00485 void reset() override
00486 {}
00487
00488 bool check_queues() override
00489 { return true; }
00490
00491 int reconfig_queue(unsigned idx) override
00492 {
00493     if (idx >= sizeof(_q) / sizeof(_q[0]))
00494         return -L4_ERANGE;
00495
00496     return setup_queue(_q + idx, idx, queue_size);
00497 }
00498
00499 void trigger_driver_config_irq() override
00500 {
00501     _dev_config.add_irq_status(L4VIRTIO_IRQ_STATUS_CONFIG);
00502     _notify_guest_irq->trigger();
00503 }
00504
00505 L4::Ipc_svr::Server_iface *server_iface() const override
00506 { return L4::Epiface::server_iface(); }
00507
00508 long op_set_status(L4virtio::Device::Rights r, unsigned status)
00509 { return L4virtio::Svr::Device::op_set_status(r, status); }
00510
00511 long op_config_queue(L4virtio::Device::Rights r, unsigned queue)
00512 { return L4virtio::Svr::Device::op_config_queue(r, queue); }
00513
00514 long op_device_config(L4virtio::Device::Rights r,
00515                       L4::Ipc::Cap<L4Re::Dataspace> &config_ds,
00516                       L4_addr_t &ds_offset)
00517 { return L4virtio::Svr::Device::op_device_config(r, config_ds, ds_offset); }
00518
00519 L4::Cap<L4::Irq> device_notify_irq() const override
00520 { return L4::cap_cast<L4::Irq>(_host_irq.obj_cap()); }
00521
00522 void register_single_driver_irq() override
00523 {
00524     _notify_guest_irq = L4Re::chkcapi
00525         (server_iface()->template rcv_cap<L4::Irq>(0));
00526
00527     L4Re::chksys(server_iface()->realloc_rcv_cap(0));
00528 }
00529
00530 private:
00531     L4Re::Util::Object_registry *_registry;
00532     L4virtio::Svr::Dev_config_t<Gpio_config_space> _dev_config;
00533     Host_irq _host_irq;
00534     L4::Cap<L4::Irq> _notify_guest_irq;
00535     L4virtio::Svr::Virtqueue _q[2];
00536     Req_processor _req_processor;
00537     Irq_req_processor _irq_req_processor;
00538 };
00539
00540 } // namespace Svr
00541 } // namespace L4virtio

```

16.276 virtio-i2c-device

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * Copyright (C) 2024-2025 Kernkonzept GmbH.
00004  * Author(s): Martin Kuetzler <martin.kuetzler@kernkonzept.com>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)

```

```

00007  */
00008
00009 #pragma once
00010
00011 #include <l4/re/error_helper>
00012 #include <l4/sys/cxx/ipc_epiface>
00013
00014 #include <l4/l4virtio/server/virtio>
00015 #include <l4/l4virtio/server/l4virtio>
00016 #include <l4/l4virtio/l4virtio>
00017
00018 #include <l4/re/error_helper>
00019 #include <l4/re/util/object_registry>
00020 #include <l4/re/util/br_manager>
00021 #include <l4/sys/cxx/ipc_epiface>
00022
00023 #include <vector>
00024 #include <optional>
00025 #include <l4/cxx/pair>
00026
00027 namespace L4virtio {
00028 namespace Svr {
00029
00030 enum class I2c_transfer_result : l4_uint8_t
00031 {
00032     I2c_msg_ok = 0,
00033     I2c_msg_err = 1
00034 };
00035
00036 struct I2c_request_flags
00037 {
00038     l4_uint32_t raw;
00039
00040     CXX_BITFIELD_MEMBER(0, 0, fail_next, raw);
00041     CXX_BITFIELD_MEMBER(1, 1, m_rd, raw);
00042 };
00043 static_assert(sizeof(I2c_request_flags) == 4,
00044     "I2c_request_flags contains padding bytes.");
00045
00046 struct I2c_out_hdr
00047 {
00048     l4_uint16_t addr;
00049     l4_uint16_t padding;
00050     I2c_request_flags flags;
00051 };
00052 static_assert(sizeof(I2c_out_hdr) == 8, "I2c_out_hdr contains padding bytes.");
00053
00054 struct I2c_in_hdr
00055 {
00056     l4_uint8_t status;
00057 };
00058 static_assert(sizeof(I2c_in_hdr) == 1, "I2c_in_hdr contains padding bytes.");
00059
00060 struct I2c_req
00061 {
00062     struct I2c_out_hdr out_hdr;
00063     unsigned buf_len;
00064     l4_uint8_t* buf;
00065     struct I2c_in_hdr *in_hdr;
00066     L4virtio::Svr::Virtqueue::Head_desc head;
00067
00068     unsigned write_size;
00069
00070     void set_status(I2c_transfer_result status)
00071     {
00072         in_hdr->status = static_cast<l4_uint8_t>(status);
00073     }
00074 };
00075
00076 template <typename Request_handler,
00077     typename Epiface = L4virtio::Device>
00078 class Virtio_i2c : public L4virtio::Svr::Device,
00079     public L4::Epiface_t<Virtio_i2c<Request_handler,
00080         Epiface>,
00081         Epiface>
00082 {
00083 private:
00084     enum
00085     {
00086         Num_request_queues = 1,
00087         queue_size = 128,
00088     };
00089
00090 public:
00091     using I2c_request_handler = Request_handler;
00092
00093     class Host_irq : public L4::Irqep_t<Host_irq>

```



```

00110 {
00111 public:
00112     explicit Host_irq(Virtio_i2c *i2c) : L4::Irqep_t<Host_irq>(), _i2c(i2c) {}
00113
00114     void handle_irq()
00115     {
00116         _i2c->handle_queue();
00117     }
00118
00119 private:
00120     Virtio_i2c *_i2c;
00121 };
00122
00126 class Request_processor : public L4virtio::Svr::Request_processor
00127 {
00128 public:
00129
00130     struct Data_buffer : public L4virtio::Svr::Data_buffer
00131     {
00132         Data_buffer()
00133         {
00134             pos = nullptr;
00135             left = 0;
00136         }
00137         // This constructor is called from within start, so make it available.
00138         Data_buffer(L4virtio::Svr::Driver_mem_region const *r,
00139                     L4virtio::Svr::Virtqueue::Desc const &d,
00140                     L4virtio::Svr::Request_processor const *)
00141         {
00142             pos = static_cast<char *>(r->local(d.addr));
00143             left = d.len;
00144         }
00145     };
00146
00147     Request_processor(L4virtio::Svr::Virtqueue *q, I2c_request_handler *hndlr,
00148                      Virtio_i2c *i2c)
00149         : _q(q), _req_handler(hndlr), _i2c(i2c), _fail_next(false)
00150     {}
00151
00162     I2c_req get_request()
00163     {
00164         I2c_req request;
00165         if (_pending_req.has_value())
00166         {
00167             request = _pending_req.value();
00168             _pending_req.reset();
00169             return request;
00170         }
00171
00172         auto r = _q->next_avail();
00173         if (L4_UNLIKELY(!r))
00174         {
00175             request.head = Virtqueue::Head_desc();
00176             return request;
00177         }
00178
00179         Data_buffer req;
00180         request.head = start(_i2c->mem_info(), r, &req);
00181         memcpy(&request.out_hdr, req.pos, sizeof(I2c_out_hdr));
00182
00183         // number of bytes to be written in the answer.
00184         request.write_size = sizeof(I2c_in_hdr);
00185         request.buf_len = 0;
00186
00187         // 2nd part: either the optional buffer or the in_hdr
00188         if (next(_i2c->mem_info(), &req))
00189         {
00190             request.buf_len += req.left;
00191             request.buf = reinterpret_cast<l4_uint8_t *>(req.pos);
00192         }
00193
00194         // 3rd part: in_hdr
00195         if (next(_i2c->mem_info(), &req))
00196         {
00197             // 2nd part was indeed a buffer
00198             if (request.out_hdr.flags.m_rd())
00199                 request.write_size += request.buf_len;
00200
00201             // actual 3rd part
00202             request.in_hdr = reinterpret_cast<I2c_in_hdr *>(req.pos);
00203         }
00204         else
00205         {
00206             // no 3rd part, 2nd part is in_hdr;
00207             request.in_hdr = reinterpret_cast<I2c_in_hdr *>(request.buf);
00208         }

```

```

00209         request.buf = nullptr;
00210         request.buf_len = 0;
00211     }
00212
00213     return request;
00214 }
00215
00216 void handle_requests()
00217 {
00218     using Consumed_entry =
00219         cxx::Pair<L4virtio::Svr::Virtqueue::Head_desc, l4_uint32_t>;
00220     std::vector<Consumed_entry> consumed;
00221
00222     for (;;)
00223     {
00224         auto r = get_request();
00225         if (!r.head)
00226             break;
00227
00228         if (_fail_next)
00229         {
00230             r.set_status(I2c_transfer_result::I2c_msg_err);
00231             _fail_next = r.out_hdr.flags.fail_next();
00232         }
00233         else
00234         {
00235             std::optional<bool> ok;
00236             l4_uint16_t i2c_addr = r.out_hdr.addr >> 1;
00237             if (r.out_hdr.flags.m_rd())
00238                 ok = _req_handler->handle_read(i2c_addr, r.buf, r.buf_len);
00239             else
00240                 ok = _req_handler->handle_write(i2c_addr, r.buf, r.buf_len);
00241             if (ok.has_value())
00242             {
00243                 if (ok.value())
00244                 {
00245                     r.set_status(I2c_transfer_result::I2c_msg_ok);
00246                     _fail_next = false;
00247                 }
00248                 else
00249                 {
00250                     r.set_status(I2c_transfer_result::I2c_msg_err);
00251                     _fail_next = r.out_hdr.flags.fail_next();
00252                 }
00253             }
00254             else
00255             {
00256                 _pending_req = r;
00257                 break;
00258             }
00259         }
00260         consumed.emplace_back(r.head, r.write_size);
00261     }
00262
00263     if (!consumed.empty())
00264         _q->finish(consumed.begin(), consumed.end(), _i2c);
00265 }
00266
00267 private:
00268     L4virtio::Svr::Virtqueue *_q;
00269     I2c_request_handler *_req_handler;
00270     Virtio_i2c *_i2c;
00271     bool _fail_next;
00272     std::optional<I2c_req> _pending_req;
00273 };
00274
00275 struct Features : public L4virtio::Svr::Dev_config::Features
00276 {
00277     Features() = default;
00278     Features(l4_uint32_t raw) : L4virtio::Svr::Dev_config::Features(raw) {}
00279
00280     // This feature is mandatory. The driver is requested to abort communication
00281     // if this is not offered.
00282     CXX_BITFIELD_MEMBER(0, 0, zero_length_request, raw);
00283 };
00284
00285 Virtio_i2c(I2c_request_handler *hndlr, L4Re::Util::Object_registry *registry)
00286 : L4virtio::Svr::Device(&_dev_config),
00287   _dev_config(L4VIRTIO_VENDOR_KK, L4VIRTIO_ID_I2C, Num_request_queues),
00288   _req_handler(hndlr),
00289   _host_irq(this),
00290   _request_processor(&_q, hndlr, this)
00291 {
00292     init_mem_info(2);
00293     reset_queue_config(0, queue_size);
00294     setup_queue(&_q, 0, queue_size);
00295     registry->register_irq_obj(&_host_irq);

```

```

00296
00297     Features hf(0);
00298     hf.ring_indirect_desc() = true;
00299     hf.zero_length_request() = true;
00300     _dev_config.host_features(0) = hf.raw;
00301     _dev_config.set_host_feature(L4VIRTIO_FEATURE_VERSION_1);
00302     _dev_config.reset_hdr();
00303 }
00304
00305 void notify_queue(L4virtio::Svr::Virtqueue *)
00306 {
00307     if (_q.no_notify_guest())
00308         return;
00309
00310     _dev_config.add_irq_status(L4VIRTIO_IRQ_STATUS_VRING);
00311     L4Re::chkipc(_notify_guest_irq->trigger(), "trigger guest irq");
00312 }
00313
00314 void handle_queue()
00315 {
00316     _request_processor.handle_requests();
00317 }
00318
00319 void reset() override
00320 {
00321 }
00322
00323 bool check_queues() override
00324 {
00325     return true;
00326 }
00327
00328 int reconfig_queue(unsigned idx) override
00329 {
00330     if (idx != 0)
00331         return -L4_ERANGE;
00332
00333     setup_queue(&_q, 0, queue_size);
00334
00335     return L4_EOK;
00336 }
00337
00338 void trigger_driver_config_irq() override
00339 {
00340     _dev_config.add_irq_status(L4VIRTIO_IRQ_STATUS_CONFIG);
00341     _notify_guest_irq->trigger();
00342 }
00343
00344 L4::Ipc_svr::Server_iface *server_iface() const override
00345 {
00346     return L4::Epiface::server_iface();
00347 }
00348
00349 long op_set_status(L4virtio::Device::Rights r, unsigned status)
00350 {
00351     return L4virtio::Svr::Device::op_set_status(r, status);
00352 }
00353
00354 long op_config_queue(L4virtio::Device::Rights r, unsigned queue)
00355 {
00356     return L4virtio::Svr::Device::op_config_queue(r, queue);
00357 }
00358
00359 long op_device_config(L4virtio::Device::Rights r,
00360                      L4::Ipc::Cap<L4Re::Dataspace> &config_ds,
00361                      l4_addr_t &ds_offset)
00362 {
00363     return L4virtio::Svr::Device::op_device_config(r, config_ds, ds_offset);
00364 }
00365
00366 L4::Cap<L4::Irq> device_notify_irq() const override
00367 {
00368     return L4::cap_cast<L4::Irq>(_host_irq.obj_cap());
00369 }
00370
00371 void register_single_driver_irq() override
00372 {
00373     _notify_guest_irq = L4Re::chkcap
00374         (server_iface()->template rcv_cap<L4::Irq>(0));
00375
00376     L4Re::chksys(server_iface()->realloc_rcv_cap(0));
00377 }
00378
00379
00380 private:
00381     L4virtio::Svr::Dev_config_t<L4virtio::Svr::No_custom_data> _dev_config;
00382     I2c_request_handler *_req_handler;

```

```

00383     L4virtio::Svr::Virtqueue _q;
00384     Host_irq _host_irq;
00385     L4::Cap<L4::Irq> _notify_guest_irq;
00386     Request_processor _request_processor;
00387 };
00388
00389 } // namespace Svr
00390 } // namespace L4virtio

```

16.277 virtio-rng-device

```

00001 // vi:ft=c++: -*- Mode: C++ -*-
00002 /*
00003  * Copyright (C) 2024-2025 Kernkonzept GmbH.
00004  * Author(s): Martin Kuetzler <martin.kuetzler@kernkonzept.com>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008
00009 #pragma once
00010
00011 #include <l4/re/error_helper>
00012 #include <l4/sys/cxx/ipc_epiface>
00013
00014 #include <l4/l4virtio/server/virtio>
00015 #include <l4/l4virtio/server/l4virtio>
00016 #include <l4/l4virtio/l4virtio>
00017
00018 namespace L4virtio {
00019 namespace Svr {
00020
00021     template <typename Rnd_state, typename Epiface = L4virtio::Device>
00022     class Virtio_rng : public L4virtio::Svr::Device,
00023                       public L4::Epiface_t<Virtio_rng<Rnd_state>, Epiface>
00024     {
00025     private:
00026         enum
00027         {
00028             Num_request_queues = 1,
00029             queue_size = 128,
00030         };
00031
00032     public:
00033         using Random_state = Rnd_state;
00034
00035         class Host_irq : public L4::Irqep_t<Host_irq>
00036         {
00037         public:
00038             explicit Host_irq(Virtio_rng *rng) : L4::Irqep_t<Host_irq>(), _rng(rng) {}
00039
00040             void handle_irq()
00041             {
00042                 _rng->handle_queue();
00043             }
00044
00045     private:
00046         Virtio_rng *_rng;
00047     };
00048
00049     class Request_processor : public L4virtio::Svr::Request_processor
00050     {
00051     public:
00052         struct Data_buffer : public L4virtio::Svr::Data_buffer
00053         {
00054         {
00055             Data_buffer() = default;
00056             // This constructor is called from within start, so make it available.
00057             Data_buffer(L4virtio::Svr::Driver_mem_region const &r,
00058                        L4virtio::Svr::Virtqueue::Desc const &d,
00059                        L4virtio::Svr::Request_processor const *)
00060             {
00061                 pos = static_cast<char *>(r->local(d.addr));
00062                 left = d.len;
00063             }
00064         };
00065
00066         Request_processor(L4virtio::Svr::Virtqueue *q, Random_state *rnd,
00067                          Virtio_rng *rng)
00068             : _q(q), _rnd(rnd), _rng(rng), _head() {}
00069
00070         bool init_queue()
00071         {
00072             auto r = _q->next_avail();
00073

```

```

00092     if (L4_UNLIKELY(!r))
00093         return false;
00094
00095     _head = start(_rng->mem_info(), r, &_req);
00096
00097     return true;
00098 }
00099
00100 void handle_request()
00101 {
00102     if (!_head)
00103         if (!init_queue())
00104             return;
00105
00106     for (;;)
00107     {
00108         auto const pos = reinterpret_cast<unsigned char *>(_req.pos);
00109         L4Re::chksys(_rnd->get_random(_req.left, pos), "get random");
00110         _q->finish(_head, _rng, _req.left);
00111         if (!init_queue())
00112             break;
00113     }
00114     return;
00115 }
00116
00117 private:
00118     L4virtio::Svr::Virtqueue *_q;
00119     Random_state *_rnd;
00120     Virtio_rng *_rng;
00121     L4virtio::Svr::Virtqueue::Head_desc _head;
00122     Data_buffer _req;
00123 };
00124
00125 Virtio_rng(Random_state *rnd, L4::Registry_iface *registry)
00126 : L4virtio::Svr::Device(&_dev_config),
00127   _dev_config(L4VIRTIO_VENDOR_KK, L4VIRTIO_ID_RNG, Num_request_queues),
00128   _rnd(rnd),
00129   _host_irq(this),
00130   _request_processor(&_q, rnd, this)
00131 {
00132     init_mem_info(2);
00133     reset_queue_config(0, queue_size);
00134     setup_queue(&_q, 0, queue_size);
00135     registry->register_irq_obj(&_host_irq);
00136
00137     L4virtio::Svr::Dev_config::Features hf;
00138     hf.ring_indirect_desc() = true;
00139     _dev_config.host_features(0) = hf.raw;
00140     _dev_config.set_host_feature(L4VIRTIO_FEATURE_VERSION_1);
00141     _dev_config.reset_hdr();
00142 }
00143
00144 void notify_queue(L4virtio::Svr::Virtqueue *)
00145 {
00146     if (_q.no_notify_guest())
00147         return;
00148
00149     _dev_config.add_irq_status(L4VIRTIO_IRQ_STATUS_VRING);
00150     L4Re::chkipc(_notify_guest_irq->trigger(), "trigger guest irq");
00151 }
00152
00153 void handle_queue()
00154 {
00155     _request_processor.handle_request();
00156 }
00157
00158 void reset() override
00159 {
00160 }
00161
00162 bool check_queues() override
00163 {
00164     return true;
00165 }
00166
00167 int reconfig_queue(unsigned idx) override
00168 {
00169     if (idx != 0)
00170         return -L4_ERANGE;
00171
00172     setup_queue(&_q, 0, queue_size);
00173
00174     return L4_EOK;
00175 }
00176
00177 void trigger_driver_config_irq() override
00178 {

```

```

00179     _dev_config.add_irq_status(L4VIRTIO_IRQ_STATUS_CONFIG);
00180     _notify_guest_irq->trigger();
00181 }
00182
00183 L4::Ipc_svr::Server_iface *server_iface() const override
00184 {
00185     return L4::Epiface::server_iface();
00186 }
00187
00188 long op_set_status(L4virtio::Device::Rights r, unsigned status)
00189 {
00190     return L4virtio::Svr::Device::op_set_status(r, status);
00191 }
00192
00193 long op_config_queue(L4virtio::Device::Rights r, unsigned queue)
00194 {
00195     return L4virtio::Svr::Device::op_config_queue(r, queue);
00196 }
00197
00198 long op_device_config(L4virtio::Device::Rights r,
00199                      L4::Ipc::Cap<L4Re::Dataspace> &config_ds,
00200                      L4_addr_t &ds_offset)
00201 {
00202     return L4virtio::Svr::Device::op_device_config(r, config_ds, ds_offset);
00203 }
00204
00205 L4::Cap<L4::Irq> device_notify_irq() const override
00206 {
00207     return L4::cap_cast<L4::Irq>(_host_irq.obj_cap());
00208 }
00209
00210 void register_single_driver_irq() override
00211 {
00212     _notify_guest_irq = L4Re::chkcapi
00213         (server_iface()->template rcv_cap<L4::Irq>(0));
00214     L4Re::chksys(server_iface()->realloc_rcv_cap(0));
00215 }
00216
00217
00218
00219 private:
00220     L4virtio::Svr::Dev_config_t<L4virtio::Svr::No_custom_data>_dev_config;
00221     Random_state *_rnd;
00222     L4virtio::Svr::Virtqueue _q;
00223     Host_irq _host_irq;
00224     L4::Cap<L4::Irq> _notify_guest_irq;
00225     Request_processor _request_processor;
00226 };
00227
00228 } // namespace Svr
00229 } // namespace L4virtio

```

16.278 virtio-scmi-device

```

00001 // vi:ft=c++
00002 /* SPDX-License-Identifier: MIT */
00003 /*
00004  * Copyright (C) 2024 Kernkonzept GmbH.
00005  * Author(s): Christian Pötzsch <christian.poetzsch@kernkonzept.com>
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009 #pragma once
00010
00011 #include <l4/cxx/bitmap>
00012 #include <l4/l4virtio/l4virtio>
00013 #include <l4/l4virtio/server/l4virtio>
00014 #include <l4/l4virtio/server/virtio>
00015 #include <l4/re/util/object_registry>
00016
00017 #include <map>
00018 #include <vector>
00019
00020 namespace L4virtio { namespace Svr { namespace Scmi {
00021
00022     enum
00023     {
00024         Version = 0x20000
00025     };
00026
00027     enum
00028     {
00029         Success = 0,

```

```

00032 Not_supported = -1,
00033 Invalid_parameters = -2,
00034 Denied = -3,
00035 Not_found = -4,
00036 Out_of_range = -5,
00037 Busy = -6,
00038 Comms_error = -7,
00039 Generic_error = -8,
00040 Hardware_error = -9,
00041 Protocol_error = -10
00042 };
00043
00045 struct Scmi_hdr_t
00046 {
00047     14_uint32_t hdr_raw = 0;
00048     CXX_BITFIELD_MEMBER(18, 27, token, hdr_raw);
00049     CXX_BITFIELD_MEMBER(10, 17, protocol_id, hdr_raw);
00050     CXX_BITFIELD_MEMBER( 8,  9, message_type, hdr_raw);
00051     CXX_BITFIELD_MEMBER( 0,  7, message_id, hdr_raw);
00052 };
00053
00055 enum
00056 {
00057     Base_protocol = 0x10,
00058     Power_domain_management_protocol = 0x11,
00059     System_power_management_protocol = 0x12,
00060     Performance_domain_management_protocol = 0x13,
00061     Clock_management_protocol = 0x14,
00062     Sensor_management_protocol = 0x15,
00063     Reset_domain_management_protocol = 0x16,
00064     Voltage_domain_management_protocol = 0x17
00065 };
00066
00068 enum
00069 {
00070     Protocol_version = 0x0,
00071     Protocol_attributes = 0x1,
00072     Protocol_message_attributes = 0x2,
00073 };
00074
00076 enum
00077 {
00078     Base_discover_vendor = 0x3,
00079     Base_discover_sub_vendor = 0x4,
00080     Base_discover_implementation_version = 0x5,
00081     Base_discover_list_protocols = 0x6,
00082     Base_discover_agent = 0x7,
00083     Base_notify_errors = 0x8,
00084     Base_set_device_permissions = 0x9,
00085     Base_set_protocol_permissions = 0xa,
00086     Base_reset_agent_configuration = 0xb
00087 };
00088
00090 struct Base_attr_t
00091 {
00092     14_uint32_t attr_raw = 0;
00093     CXX_BITFIELD_MEMBER(8, 15, nagents, attr_raw);
00094     CXX_BITFIELD_MEMBER(0,  7, nprotos, attr_raw);
00095 };
00096
00098 enum
00099 {
00100     Performance_domain_attributes = 0x3,
00101     Performance_describe_levels = 0x4,
00102     Performance_limits_set = 0x5,
00103     Performance_limits_get = 0x6,
00104     Performance_level_set = 0x7,
00105     Performance_level_get = 0x8,
00106     Performance_notify_limits = 0x9,
00107     Performance_notify_level = 0xa,
00108     Performance_describe_fastchannel = 0xb,
00109 };
00110
00112 struct Performance_attr_t
00113 {
00114     14_uint32_t attr_raw = 0;
00115     CXX_BITFIELD_MEMBER(16, 16, power, attr_raw);
00116     CXX_BITFIELD_MEMBER( 0, 15, domains, attr_raw);
00117
00118     14_uint32_t stat_addr_low = 0;
00119     14_uint32_t stat_addr_high = 0;
00120     14_uint32_t stat_len = 0;
00121 };
00122
00124 struct Performance_domain_attr_t
00125 {
00126     14_uint32_t attr_raw = 0;

```

```

00127 CXX_BITFIELD_MEMBER(31, 31, set_limits, attr_raw);
00128 CXX_BITFIELD_MEMBER(30, 30, set_perf_level, attr_raw);
00129 CXX_BITFIELD_MEMBER(29, 29, perf_limits_change_notify, attr_raw);
00130 CXX_BITFIELD_MEMBER(28, 28, perf_level_change_notify, attr_raw);
00131 CXX_BITFIELD_MEMBER(27, 27, fast_channel, attr_raw);
00132
00133 l4_uint32_t rate_limit_raw = 0;
00134 CXX_BITFIELD_MEMBER( 0, 19, rate_limit, rate_limit_raw);
00135
00136 l4_uint32_t sustained_freq = 0;
00137 l4_uint32_t sustained_perf_level = 0;
00138 char name[16] = { 0 };
00139 };
00140
00142 struct Performance_describe_levels_n_t
00143 {
00144     l4_uint32_t num_levels_raw = 0;
00145     CXX_BITFIELD_MEMBER(16, 31, nremain_perf_levels, num_levels_raw);
00146     CXX_BITFIELD_MEMBER( 0, 11, nperf_levels, num_levels_raw);
00147 };
00148
00150 struct Performance_describe_level_t
00151 {
00152     l4_uint32_t perf_level = 0;
00153     l4_uint32_t power_cost = 0;
00154     l4_uint16_t trans_latency = 0;
00155     l4_uint16_t res0 = 0;
00156 };
00157
00158 template<typename OBSERV>
00159 struct Queue_worker : Request_processor
00160 {
00161     Queue_worker(OBSERV *o, Virtqueue *queue)
00162     : o(o), q(queue)
00163     {}
00164
00165     bool init_queue()
00166     {
00167         auto r = q->next_avail();
00168
00169         if (L4_UNLIKELY(!r))
00170             return false;
00171
00172         head = start(o->mem_info(), r, &req);
00173
00174         return true;
00175     }
00176
00177     bool next()
00178     { return Request_processor::next(o->mem_info(), &req); }
00179
00180     void finish(l4_uint32_t total)
00181     { q->finish(head, o, total); }
00182
00183     template<typename T>
00184     l4_ssize_t read(Data_buffer *buf, T *data, l4_size_t s = sizeof(T))
00185     {
00186         buf->pos = reinterpret_cast<char *>(data);
00187         buf->left = s;
00188         l4_size_t chunk = 0;
00189         for (;;)
00190         {
00191             chunk += req.copy_to(buf);
00192             if (req.done())
00193                 next();
00194             if (!buf->left)
00195                 break;
00196         }
00197         if (chunk != s)
00198             return -1;
00199         return chunk;
00200     }
00201
00202     template<typename T>
00203     l4_ssize_t write(Data_buffer *buf, T *data, l4_size_t s = sizeof(T))
00204     {
00205         buf->pos = reinterpret_cast<char *>(data);
00206         buf->left = s;
00207         l4_size_t chunk = 0;
00208         for (;;)
00209         {
00210             chunk += buf->copy_to(&req);
00211             if (req.done())
00212                 next();
00213             if (!buf->left)
00214                 break;
00215         }

```



```

00216     if (chunk != s)
00217         return -1;
00218     return chunk;
00219 }
00220
00221 l4_ssize_t handle_request()
00222 {
00223     try
00224     {
00225         if (!head && L4_UNLIKELY(!init_queue()))
00226             return 0;
00227
00228         for (;;)
00229         {
00230             l4_ssize_t total = 0;
00231             l4_ssize_t res = 0;
00232             Scmi_hdr_t hdr;
00233             Data_buffer buf = Data_buffer(&hdr);
00234             if ((res = read(&buf, &hdr)) < 0)
00235                 return res;
00236
00237             // Search/execute handler for given protocol
00238             auto proto = o->proto(hdr.protocol_id());
00239             if (proto)
00240             {
00241                 if ((res = proto->handle_request(hdr, buf, this)) < 0)
00242                     return res;
00243                 total += res;
00244             }
00245             else
00246             {
00247                 if ((res = write(&buf, &hdr)) < 0)
00248                     return res;
00249                 total += res;
00250
00251                 l4_int32_t status = Not_supported;
00252                 if ((res = write(&buf, &status)) < 0)
00253                     return res;
00254                 total += res;
00255             }
00256
00257             finish(total);
00258
00259             head = Virtqueue::Head_desc();
00260             if (L4_UNLIKELY(!init_queue()))
00261                 return 0;
00262         }
00263     }
00264     catch (L4virtio::Svr::Bad_descriptor const &e)
00265     {
00266         return e.error;
00267     }
00268     return 0;
00269 }
00270
00271 private:
00272 struct Buffer : Data_buffer
00273 {
00274     Buffer() = default;
00275     Buffer(L4virtio::Svr::Driver_mem_region const *r,
00276           Virtqueue::Desc const &d, Request_processor const *)
00277     {
00278         pos = static_cast<char *>(r->local(d.addr));
00279         left = d.len;
00280     }
00281 };
00282
00283 Virtqueue::Head_desc head;
00284 Buffer req;
00285
00286 OBSERV *o;
00287 Virtqueue *q;
00288 };
00289
00290 template<typename OBSERV>
00291 struct Proto
00292 {
00293     virtual l4_ssize_t handle_request(Scmi_hdr_t &hdr,
00294                                       Data_buffer &buf,
00295                                       Queue_worker<OBSERV> *qw) = 0;
00296 };
00297
00298 class Scmi_dev : public L4virtio::Svr::Device
00299 {
00300     struct Features : L4virtio::Svr::Dev_config::Features
00301     {
00302         Features() = default;

```

```

00341     Features(l4_uint32_t raw) : L4virtio::Svr::Dev_config::Features(raw) {}
00342 };
00343
00344 struct Host_irq : public L4::Irqp_t<Host_irq>
00345 {
00346     Scmi_dev *c;
00347     explicit Host_irq(Scmi_dev *c) : c(c) {}
00348     void handle_irq() { c->kick(); }
00349 };
00350
00351 enum
00352 {
00353     Queue_size = 0x10
00354 };
00355
00356 public:
00357     Scmi_dev(L4Re::Util::Object_registry *registry)
00358     : L4virtio::Svr::Device(&_dev_config),
00359       _dev_config(L4VIRTIO_VENDOR_KK, L4VIRTIO_ID_SCM, 1),
00360       _host_irq(this),
00361       _request_worker(this, &q[0])
00362     {
00363         init_mem_info(2);
00364
00365         L4Re::chkcap(registry->register_irq_obj(&_host_irq),
00366                     "Register irq object");
00367
00368         Features hf(0);
00369         hf.ring_indirect_desc() = true;
00370
00371         _dev_config.host_features(0) = hf.raw;
00372
00373         _dev_config.set_host_feature(L4VIRTIO_FEATURE_VERSION_1);
00374         _dev_config.reset_hdr();
00375
00376         reset();
00377     }
00378
00380 void add_proto(l4_uint32_t id, Proto<Scmi_dev> *proto)
00381 { _protos.insert({id, proto}); }
00382
00383 Proto<Scmi_dev> *proto(l4_uint32_t id) const
00384 {
00385     if (_protos.find(id) != _protos.end())
00386         return _protos.at(id);
00387     return nullptr;
00388 }
00389
00390 void notify_queue(L4virtio::Virtqueue *queue)
00391 {
00392     if (queue->no_notify_guest())
00393         return;
00394
00395     _dev_config.add_irq_status(L4VIRTIO_IRQ_STATUS_VRING);
00396     _kick_guest_irq->trigger();
00397 }
00398
00399 private:
00400     L4::Cap<L4::Irqp> device_notify_irq() const override
00401     { return L4::cap_cast<L4::Irqp>(_host_irq.obj_cap()); }
00402
00403 void register_single_driver_irq() override
00404 {
00405     _kick_guest_irq = L4Re::Util::Unique_cap<L4::Irqp>(
00406         L4Re::chkcap(server_iface()->template rcv_cap<L4::Irqp>(0)));
00407
00408     L4Re::chksys(server_iface()->realloc_rcv_cap(0));
00409 }
00410
00411 void kick()
00412 {
00413     if (_request_worker.handle_request() < 0)
00414         device_error();
00415 }
00416
00417 void reset() override
00418 {
00419     for (Virtqueue &q : _q)
00420         q.disable();
00421
00422     for (l4_uint32_t i = 0; i < _dev_config.num_queues(); i++)
00423         reset_queue_config(i, Queue_size);
00424 }
00425
00426 bool check_queues() override
00427 {
00428     return true;

```

```

00429     }
00430
00431     int reconfig_queue(unsigned idx) override
00432     {
00433         if (idx >= sizeof(_q) / sizeof(_q[0]))
00434             return -L4_ERANGE;
00435
00436         return setup_queue(_q + idx, idx, Queue_size);
00437     }
00438
00439     void trigger_driver_config_irq() override
00440     {
00441         _dev_config.add_irq_status(L4VIRTIO_IRQ_STATUS_CONFIG);
00442         _kick_guest_irq->trigger();
00443     }
00444
00445     L4virtio::Svr::Dev_config_t<L4virtio::Svr::No_custom_data> _dev_config;
00446     Host_irq _host_irq;
00447     L4Re::Util::Unique_cap<L4::Irq> _kick_guest_irq;
00448     Virtqueue _q[1];
00449     Queue_worker<Scmi_dev> _request_worker;
00450     std::map<l4_uint32_t, Proto<Scmi_dev> *> _protos;
00451 };
00452
00453 class Base_proto : public Proto<Scmi_dev>
00454 {
00455     virtual l4_int32_t fill_attr(Base_attr_t *attr) const = 0;
00456
00457     virtual std::vector<l4_uint32_t> prots() const = 0;
00458
00459     l4_ssize_t handle_request(Scmi_hdr_t &hdr, Data_buffer &buf,
00460                               Queue_worker<Scmi_dev> *qw) override
00461     {
00462         l4_ssize_t total = 0;
00463         l4_ssize_t res = 0;
00464         switch (hdr.message_id())
00465         {
00466             case Protocol_version:
00467             {
00468                 if ((res = qw->write(&buf, &hdr)) < 0)
00469                     return res;
00470                 total += res;
00471
00472                 struct
00473                 {
00474                     l4_int32_t status = Success;
00475                     l4_uint32_t version = Version;
00476                 } version;
00477                 if ((res = qw->write(&buf, &version)) < 0)
00478                     return res;
00479                 total += res;
00480                 break;
00481             }
00482             case Protocol_attributes:
00483             {
00484                 if ((res = qw->write(&buf, &hdr)) < 0)
00485                     return res;
00486                 total += res;
00487
00488                 Base_attr_t ba;
00489                 l4_int32_t status = fill_attr(&ba);
00490                 if ((res = qw->write(&buf, &status)) < 0)
00491                     return res;
00492                 total += res;
00493                 if (status == Success)
00494                 {
00495                     if ((res = qw->write(&buf, &ba)) < 0)
00496                         return res;
00497                     total += res;
00498                 }
00499                 break;
00500             }
00501             case Protocol_message_attributes:
00502             {
00503                 l4_uint32_t msg_id = 0;
00504                 if ((res = qw->read(&buf, &msg_id)) < 0)
00505                     return res;
00506
00507                 if ((res = qw->write(&buf, &hdr)) < 0)
00508                     return res;
00509                 total += res;
00510
00511                 struct
00512                 {
00513                     l4_int32_t status = Not_found;
00514                     l4_uint32_t attr = 0;
00515                 } attr;

```

```

00524         if (msg_id >= Protocol_version &&
00525             msg_id <= Base_discover_list_protocols)
00526             attr.status = Success;
00527
00528         if ((res = gw->write(&buf, &attr)) < 0)
00529             return res;
00530         total += res;
00531         break;
00532     }
00533     case Base_discover_vendor:
00534     {
00535         if ((res = gw->write(&buf, &hdr)) < 0)
00536             return res;
00537         total += res;
00538
00539         struct
00540         {
00541             l4_int32_t status = Success;
00542             l4_uint8_t vendor_identifiser[16] = { "L4Re" };
00543         } vendor;
00544         if ((res = gw->write(&buf, &vendor)) < 0)
00545             return res;
00546         total += res;
00547         break;
00548     }
00549     case Base_discover_sub_vendor:
00550     {
00551         if ((res = gw->write(&buf, &hdr)) < 0)
00552             return res;
00553         total += res;
00554
00555         struct
00556         {
00557             l4_int32_t status = Success;
00558             l4_uint8_t vendor_identifiser[16] = { "Scmi" };
00559         } vendor;
00560         if ((res = gw->write(&buf, &vendor)) < 0)
00561             return res;
00562         total += res;
00563         break;
00564     }
00565     case Base_discover_implementation_version:
00566     {
00567         if ((res = gw->write(&buf, &hdr)) < 0)
00568             return res;
00569         total += res;
00570
00571         struct
00572         {
00573             l4_int32_t status = Success;
00574             l4_uint32_t version = 1;
00575         } version;
00576         if ((res = gw->write(&buf, &version)) < 0)
00577             return res;
00578         total += res;
00579         break;
00580     }
00581     case Base_discover_list_protocols:
00582     {
00583         l4_uint32_t skip = 0;
00584         if ((res = gw->read(&buf, &skip)) < 0)
00585             return res;
00586
00587         if ((res = gw->write(&buf, &hdr)) < 0)
00588             return res;
00589         total += res;
00590
00591         auto p = prots();
00592         struct
00593         {
00594             l4_int32_t status = Success;
00595             l4_uint32_t num;
00596         } proto;
00597         proto.num = p.size();
00598         if ((res = gw->write(&buf, &proto)) < 0)
00599             return res;
00600         total += res;
00601
00602         // Array of uint32 where 4 protos are packed into one uint32. So
00603         // round up to 4 bytes and fill the array byte by byte.
00604         l4_uint8_t parr[(p.size() + 3) / 4 * 4] = { 0 };
00605         for (l4_size_t i = 0; i < p.size(); i++)
00606             parr[i] = p.at(i);
00607
00608         if ((res = gw->write(&buf, parr, sizeof(parr))) < 0)
00609             return res;
00610         total += res;

```

```

00611         break;
00612     }
00613     default:
00614     {
00615         if ((res = qw->write(&buf, &hdr)) < 0)
00616             return res;
00617         total += res;
00618
00619         l4_int32_t status = Not_supported;
00620         if ((res = qw->write(&buf, &status)) < 0)
00621             return res;
00622         total += res;
00623         break;
00624     }
00625 }
00626
00627 return total;
00628 }
00629 };
00630
00662 class Perf_proto : public Proto<Scmi_dev>
00663 {
00664     virtual l4_int32_t fill_attr(Performance_attr_t *attr) const = 0;
00665
00666     virtual l4_int32_t fill_domain_attr(l4_uint32_t domain_id,
00667                                         Performance_domain_attr_t *attr) const = 0;
00668
00669     virtual l4_int32_t fill_describe_levels_n(l4_uint32_t domain_id,
00670                                               l4_uint32_t level_idx,
00671                                               Performance_describe_levels_n_t *attr) const = 0;
00672
00673     virtual l4_int32_t fill_describe_levels(l4_uint32_t domain_id,
00674                                             l4_uint32_t level_idx,
00675                                             l4_uint32_t num,
00676                                             Performance_describe_level_t *attr) const = 0;
00677
00678     virtual l4_int32_t level_set(l4_uint32_t domain_id,
00679                                 l4_uint32_t perf_level) = 0;
00680
00681     virtual l4_int32_t level_get(l4_uint32_t domain_id,
00682                                 l4_uint32_t *perf_level) const = 0;
00683
00684     l4_ssize_t handle_request(Scmi_hdr_t &hdr, Data_buffer &buf,
00685                               Queue_worker<Scmi_dev> *qw) override
00686     {
00687         l4_ssize_t total = 0;
00688         l4_ssize_t res = 0;
00689         switch (hdr.message_id())
00690         {
00691             case Protocol_version:
00692             {
00693                 if ((res = qw->write(&buf, &hdr)) < 0)
00694                     return res;
00695                 total += res;
00696
00697                 struct
00698                 {
00699                     l4_int32_t status = Success;
00700                     l4_uint32_t version = Version;
00701                 } version;
00702                 if ((res = qw->write(&buf, &version)) < 0)
00703                     return res;
00704                 total += res;
00705                 break;
00706             }
00707             case Protocol_attributes:
00708             {
00709                 if ((res = qw->write(&buf, &hdr)) < 0)
00710                     return res;
00711                 total += res;
00712
00713                 Performance_attr_t pa;
00714                 l4_int32_t status = fill_attr(&pa);
00715                 if ((res = qw->write(&buf, &status)) < 0)
00716                     return res;
00717                 total += res;
00718                 if (status == Success)
00719                 {
00720                     if ((res = qw->write(&buf, &pa)) < 0)
00721                         return res;
00722                     total += res;
00723                 }
00724                 break;
00725             }
00726             case Protocol_message_attributes:
00727             {
00728                 l4_uint32_t msg_id = 0;

```

```

00738         if ((res = qw->read(&buf, &msg_id)) < 0)
00739             return res;
00740
00741         if ((res = qw->write(&buf, &hdr)) < 0)
00742             return res;
00743         total += res;
00744
00745         struct
00746         {
00747             l4_int32_t status = Not_found;
00748
00749             l4_uint32_t attr_raw = 0;
00750             CXX_BITFIELD_MEMBER(0, 0, fast_channel, attr_raw); // ignored
00751         } attr;
00752         if ((msg_id >= Protocol_version &&
00753             msg_id <= Performance_describe_levels) ||
00754             (msg_id >= Performance_level_set &&
00755             msg_id <= Performance_level_get))
00756             attr.status = Success;
00757
00758         if ((res = qw->write(&buf, &attr)) < 0)
00759             return res;
00760         total += res;
00761         break;
00762     }
00763     case Performance_domain_attributes:
00764     {
00765         l4_uint32_t domain_id = 0;
00766         if ((res = qw->read(&buf, &domain_id)) < 0)
00767             return res;
00768
00769         if ((res = qw->write(&buf, &hdr)) < 0)
00770             return res;
00771         total += res;
00772
00773         Performance_domain_attr_t attr;
00774         l4_int32_t status = fill_domain_attr(domain_id, &attr);
00775         if ((res = qw->write(&buf, &status)) < 0)
00776             return res;
00777         total += res;
00778         if (status == Success)
00779         {
00780             if ((res = qw->write(&buf, &attr)) < 0)
00781                 return res;
00782             total += res;
00783         }
00784         break;
00785     }
00786     case Performance_describe_levels:
00787     {
00788         struct
00789         {
00790             l4_uint32_t domain_id = 0;
00791             l4_uint32_t level_idx = 0;
00792         } param;
00793         if ((res = qw->read(&buf, &param)) < 0)
00794             return res;
00795
00796         if ((res = qw->write(&buf, &hdr)) < 0)
00797             return res;
00798         total += res;
00799
00800         // First figure out how many levels we support
00801         Performance_describe_levels_n_t attr;
00802         l4_int32_t status = fill_describe_levels_n(param.domain_id, param.level_idx,
00803                                                    &attr);
00804         if (status != Success)
00805         {
00806             // On error bail out early
00807             if ((res = qw->write(&buf, &status)) < 0)
00808                 return res;
00809             total += res;
00810             break;
00811         }
00812
00813         // Now fetch the actual levels
00814         Performance_describe_level_t attrl[attr.nperf_levels().get()];
00815         status = fill_describe_levels(param.domain_id, param.level_idx,
00816                                      attr.nperf_levels(), attrl);
00817         if ((res = qw->write(&buf, &status)) < 0)
00818             return res;
00819         total += res;
00820         if (status == Success)
00821         {
00822             // Write both answers to the client
00823             if ((res = qw->write(&buf, &attr)) < 0)
00824                 return res;

```

```

00825         total += res;
00826         if ((res = qw->write(&buf, attr1, sizeof(attr1))) < 0)
00827             return res;
00828         total += res;
00829     }
00830     break;
00831 }
00832 case Performance_level_set:
00833 {
00834     struct
00835     {
00836         l4_uint32_t domain_id;
00837         l4_uint32_t perf_level;
00838     } param;
00839     if ((res = qw->read(&buf, &param)) < 0)
00840         return res;
00841
00842     if ((res = qw->write(&buf, &hdr)) < 0)
00843         return res;
00844     total += res;
00845
00846     l4_int32_t status = level_set(param.domain_id, param.perf_level);
00847     if ((res = qw->write(&buf, &status)) < 0)
00848         return res;
00849     total += res;
00850     break;
00851 }
00852 case Performance_level_get:
00853 {
00854     l4_uint32_t domain_id = 0;
00855     if ((res = qw->read(&buf, &domain_id)) < 0)
00856         return res;
00857
00858     if ((res = qw->write(&buf, &hdr)) < 0)
00859         return res;
00860     total += res;
00861
00862     l4_uint32_t perf_level;
00863     l4_int32_t status = level_get(domain_id, &perf_level);
00864     if ((res = qw->write(&buf, &status)) < 0)
00865         return res;
00866     total += res;
00867     if (status == Success)
00868     {
00869         if ((res = qw->write(&buf, &perf_level)) < 0)
00870             return res;
00871         total += res;
00872     }
00873     break;
00874 }
00875 default:
00876 {
00877     if ((res = qw->write(&buf, &hdr)) < 0)
00878         return res;
00879     total += res;
00880
00881     l4_int32_t status = Not_supported;
00882     if ((res = qw->write(&buf, &status)) < 0)
00883         return res;
00884     total += res;
00885     break;
00886 }
00887 }
00888 return total;
00889 }
00890 };
00891
00892 } /* Scmi */ } /* Svr */ } /* L4virtio */

```

16.279 virtio-spi-device

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * Copyright (C) 2025 Kernkonzept GmbH.
00004  * Author(s): Martin Kuettler <martin.kuettler@kernkonzept.com>
00005  *            Jakub Jermar <jakub.jermar@kernkonzept.com>
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009
00010 #pragma once
00011
00012 #include <l4/re/error_helper>

```

```

00013 #include <l4/sys/cxx/ipc_epiface>
00014
00015 #include <l4/l4virtio/server/virtio>
00016 #include <l4/l4virtio/server/l4virtio>
00017 #include <l4/l4virtio/l4virtio>
00018
00019 #include <l4/re/error_helper>
00020 #include <l4/re/util/object_registry>
00021 #include <l4/re/util/br_manager>
00022 #include <l4/sys/cxx/ipc_epiface>
00023
00024 #include <vector>
00025 #include <l4/cxx/pair>
00026
00027 namespace L4virtio {
00028 namespace Svr {
00029
00030 struct Spi_config
00031 {
00032     l4_uint8_t cs_max_number;
00033     l4_uint8_t cs_change_supported;
00034     l4_uint8_t tx_nbits_supported;
00035     l4_uint8_t rx_nbits_supported;
00036     l4_uint32_t bits_per_word_mask;
00037     l4_uint32_t mode_func_supported;
00038     l4_uint32_t max_freq_hz;
00039     l4_uint32_t max_word_delay_ns;
00040     l4_uint32_t max_cs_setup_ns;
00041     l4_uint32_t max_cs_hold_ns;
00042     l4_uint32_t max_cs_inactive_ns;
00043 };
00044
00045 enum Spi_transfer_result: l4_uint8_t
00046 {
00047     Spi_trans_ok = 0,
00048     Spi_param_err = 1,
00049     Spi_trans_err = 2,
00050 };
00051
00052 struct Spi_transfer_head
00053 {
00054     l4_uint8_t chip_select_id;
00055     l4_uint8_t bits_per_word;
00056     l4_uint8_t cs_change;
00057     l4_uint8_t tx_nbits;
00058     l4_uint8_t rx_nbits;
00059     l4_uint8_t last_request; // XXX non-standard! aka reserved[0]
00060     l4_uint8_t reserved[2];
00061     l4_uint32_t mode;
00062     l4_uint32_t freq;
00063     l4_uint32_t word_delay_ns;
00064     l4_uint32_t cs_setup_ns;
00065     l4_uint32_t cs_delay_hold_ns;
00066     l4_uint32_t cs_change_delay_inactive_ns;
00067 };
00068 static_assert(sizeof(Spi_transfer_head) == 32,
00069     "Spi_transfer_head contains padding bytes.");
00070
00071 struct Spi_transfer_req
00072 {
00073     struct Spi_transfer_head head;
00074     l4_uint8_t *tx_buf = nullptr;
00075     l4_uint8_t *rx_buf = nullptr;
00076     Spi_transfer_result *result = nullptr;
00077
00078     unsigned tx_size = 0;
00079     unsigned rx_size = 0;
00080
00081     void set_result(Spi_transfer_result res)
00082     {
00083         *result = res;
00084     }
00085 };
00086
00100 template <typename Spi_request_handler, typename Epiface = L4virtio::Device>
00101 class Virtio_spi
00102 : public L4virtio::Svr::Device,
00103     public L4::Epiface_t<Virtio_spi<Spi_request_handler, Epiface>, Epiface>
00104 {
00105 private:
00106     enum
00107     {
00108         Num_request_queues = 1,
00109         Queue_size = 128,
00110     };
00111
00112 public:

```



```

00118 class Host_irq : public L4::Irqep_t<Host_irq>
00119 {
00120 public:
00121     explicit Host_irq(Virtio_spi *spi) : L4::Irqep_t<Host_irq>(), _spi(spi) {}
00122
00123     void handle_irq()
00124     {
00125         _spi->handle_queue();
00126     }
00127
00128 private:
00129     Virtio_spi *_spi;
00130 };
00131
00135 class Request_processor : public L4virtio::Svr::Request_processor
00136 {
00137 public:
00138
00139     struct Data_buffer : public L4virtio::Svr::Data_buffer
00140     {
00141         Data_buffer()
00142         {
00143             pos = nullptr;
00144             left = 0;
00145         }
00146         // This constructor is called from within the base start(), so make it
00147         // available.
00148         Data_buffer(L4virtio::Svr::Driver_mem_region const *r,
00149                     L4virtio::Svr::Virtqueue::Desc const &d,
00150                     L4virtio::Svr::Request_processor const *)
00151         {
00152             pos = static_cast<char *>(r->local(d.addr));
00153             left = d.len;
00154         }
00155     };
00156
00157     Request_processor(L4virtio::Svr::Virtqueue *q, Spi_request_handler *hndlr,
00158                       Virtio_spi *spi)
00159       : _q(q), _req_handler(hndlr), _spi(spi), _head(), _req()
00160     {}
00161
00162     bool init_queue()
00163     {
00164         {
00165             auto r = _q->next_avail();
00166
00167             if (L4_UNLIKELY(!r))
00168                 return false;
00169
00170             _head = start(_spi->mem_info(), r, &_req);
00171
00172             return true;
00173         }
00174
00185     Spi_transfer_req get_request()
00186     {
00187         Spi_transfer_req request;
00188         memcpy(&request.head, _req.pos, sizeof(Spi_transfer_head));
00189
00190         // Check for a TX and/or RX buffer
00191         if (!next(_spi->mem_info(), &_req))
00192         {
00193             L4Re::throw_error(-L4_EIO, "Virtio SPI request too short");
00194         }
00195
00196         if (current_flags().write())
00197         {
00198             // device-writable buffer
00199             request.rx_buf = reinterpret_cast<l4_uint8_t *>(_req.pos);
00200             request.rx_size = _req.left;
00201         }
00202         else
00203         {
00204             request.tx_buf = reinterpret_cast<l4_uint8_t *>(_req.pos);
00205             request.tx_size = _req.left;
00206         }
00207
00208         // Check for an RX buffer or request result buffer
00209         if (!next(_spi->mem_info(), &_req))
00210         {
00211             L4Re::throw_error(-L4_EIO, "Virtio SPI request too short");
00212         }
00213
00214         if (has_more())
00215         {
00216             // This must be the RX buffer
00217             if (request.rx_buf || !current_flags().write())

```

```

00218         {
00219             L4Re::throw_error(-L4_EIO, "Bad Virtio SPI request");
00220         }
00221
00222         request.rx_buf = reinterpret_cast<l4_uint8_t *>(_req.pos);
00223         request.rx_size = _req.left;
00224         next(_spi->mem_info(), &_req);
00225     }
00226
00227     request.result = reinterpret_cast<Spi_transfer_result *>(_req.pos);
00228
00229     return request;
00230 }
00231
00232 void handle_request()
00233 {
00234     if (!_head)
00235         if (!init_queue())
00236             return;
00237
00238     using Consumed_entry =
00239         cxx::Pair<L4virtio::Svr::Virtqueue::Head_desc, l4_uint32_t>;
00240     std::vector<Consumed_entry> consumed;
00241
00242     for (;;)
00243     {
00244         auto r = get_request();
00245         Spi_transfer_result res;
00246         if (r.tx_buf && r.rx_buf && (r.tx_size != r.rx_size))
00247             res = Spi_param_err;
00248         else
00249             res =
00250                 _req_handler->handle_transfer(r.head, r.tx_buf, r.rx_buf,
00251                                             r.tx_size ? r.tx_size : r.rx_size);
00252         r.set_result(res);
00253
00254         l4_uint32_t written = sizeof(Spi_transfer_result);
00255         if (res == Spi_trans_ok)
00256             written += r.rx_size;
00257
00258         consumed.emplace_back(_head, written);
00259         if (!init_queue())
00260             break;
00261     }
00262
00263     _q->finish(consumed.begin(), consumed.end(), _spi);
00264
00265     _head = Virtqueue::Head_desc();
00266 }
00267
00268 private:
00269     L4virtio::Svr::Virtqueue *_q;
00270     Spi_request_handler *_req_handler;
00271     Virtio_spi *_spi;
00272     L4virtio::Svr::Virtqueue::Head_desc _head;
00273     Data_buffer _req;
00274 };
00275
00276 Virtio_spi(Spi_request_handler *hdlr, L4Re::Util::Object_registry *registry)
00277 : L4virtio::Svr::Device(&_dev_config),
00278   _dev_config(L4VIRTIO_VENDOR_KK, L4VIRTIO_ID_SPI, Num_request_queues),
00279   _req_handler(hdlr),
00280   _host_irq(this),
00281   _request_processor(&_q, hdlr, this)
00282 {
00283     init_mem_info(2);
00284     reset_queue_config(0, Queue_size);
00285     setup_queue(&_q, 0, Queue_size);
00286     registry->register_irq_obj(&_host_irq);
00287
00288     Spi_config volatile *pc = _dev_config.priv_config();
00289
00290     pc->cs_max_number = hdlr->cs_max_number();
00291     pc->cs_change_supported = 0;
00292     pc->tx_nbits_supported = 0;
00293     pc->rx_nbits_supported = 0;
00294     pc->bits_per_word_mask = 0x80;
00295     pc->mode_func_supported = hdlr->mode_func_supported();
00296     pc->max_freq_hz = 0; // XXX: restrict wrt. controller
00297     pc->max_word_delay_ns = 0;
00298     pc->max_cs_setup_ns = 0;
00299     pc->max_cs_hold_ns = 0;
00300     pc->max_cs_inactive_ns = 0;
00301
00302     L4virtio::Svr::Dev_config::Features hf(0);
00303     _dev_config.host_features(0) = hf.raw;
00304     _dev_config.set_host_feature(L4VIRTIO_FEATURE_VERSION_1);

```

```

00305     _dev_config.reset_hdr();
00306 }
00307
00308 void notify_queue(L4virtio::Svr::Virtqueue *)
00309 {
00310     if (_q.no_notify_guest())
00311         return;
00312
00313     _dev_config.add_irq_status(L4VIRTIO_IRQ_STATUS_VRING);
00314     L4Re::chkipc(_notify_guest_irq->trigger(), "trigger guest irq");
00315 }
00316
00317 void handle_queue()
00318 {
00319     _request_processor.handle_request();
00320 }
00321
00322 void reset() override
00323 {
00324 }
00325
00326 bool check_queues() override
00327 {
00328     return true;
00329 }
00330
00331 int reconfig_queue(unsigned idx) override
00332 {
00333     if (idx != 0)
00334         return -L4_ERANGE;
00335
00336     setup_queue(&_q, 0, Queue_size);
00337
00338     return L4_EOK;
00339 }
00340
00341 void trigger_driver_config_irq() override
00342 {
00343     _dev_config.add_irq_status(L4VIRTIO_IRQ_STATUS_CONFIG);
00344     _notify_guest_irq->trigger();
00345 }
00346
00347 L4::Ipc_svr::Server_iface *server_iface() const override
00348 {
00349     return L4::Epiface::server_iface();
00350 }
00351
00352 long op_set_status(L4virtio::Device::Rights r, unsigned status)
00353 {
00354     return L4virtio::Svr::Device::op_set_status(r, status);
00355 }
00356
00357 long op_config_queue(L4virtio::Device::Rights r, unsigned queue)
00358 {
00359     return L4virtio::Svr::Device::op_config_queue(r, queue);
00360 }
00361
00362 long op_device_config(L4virtio::Device::Rights r,
00363                      L4::Ipc::Cap<L4Re::Dataspace> &config_ds,
00364                      l4_addr_t &ds_offset)
00365 {
00366     return L4virtio::Svr::Device::op_device_config(r, config_ds, ds_offset);
00367 }
00368
00369 L4::Cap<L4::Irq> device_notify_irq() const override
00370 {
00371     return L4::cap_cast<L4::Irq>(_host_irq.obj_cap());
00372 }
00373
00374 void register_single_driver_irq() override
00375 {
00376     _notify_guest_irq =
00377         L4Re::chkcip(server_iface()->template_rcv_cap<L4::Irq>(0));
00378
00379     L4Re::chksys(server_iface()->realloc_rcv_cap(0));
00380 }
00381
00382 private:
00383     L4virtio::Svr::Dev_config_t<Spi_config> _dev_config;
00384     Spi_request_handler *_req_handler;
00385     L4virtio::Svr::Virtqueue _q;
00386     Host_irq _host_irq;
00387     L4::Cap<L4::Irq> _notify_guest_irq;
00388     Request_processor _request_processor;
00389 };
00390
00391

```

```

00392 } // namespace Svr
00393 } // namespace L4virtio

```

16.280 virtio.h

```

00001 /* SPDX-License-Identifier: MIT */
00002 /*
00003  * Copyright (C) 2013-2022, 2024-2025 Kernkonzept GmbH.
00004  * Author(s): Alexander Warg <alexander.warg@kernkonzept.com>
00005  *           Matthias Lange <matthias.lange@kernkonzept.com>
00006  *
00007  */
00008 #pragma once
00009
00013
00027
00028 #include <l4/sys/compiler.h>
00029 #include <l4/sys/utcb.h>
00030 #include <l4/sys/ipc.h>
00031 #include <l4/sys/types.h>
00032
00034 enum L4_virtio_protocol
00035 {
00036     L4VIRTIO_PROTOCOL = 0,
00037 };
00038
00039 enum L4virtio_magic
00040 {
00041     L4VIRTIO_MAGIC = 0x74726976
00042 };
00043
00044 enum L4virtio_vendor
00045 {
00046     L4VIRTIO_VENDOR_KK = 0x44
00047 };
00048
00052 enum L4_virtio_opcodes
00053 {
00054     L4VIRTIO_OP_SET_STATUS      = 0,
00055     L4VIRTIO_OP_CONFIG_QUEUE    = 1,
00056     L4VIRTIO_OP_REGISTER_DS     = 3,
00057     L4VIRTIO_OP_DEVICE_CONFIG   = 4,
00058     L4VIRTIO_OP_GET_DEVICE_IRQ  = 5,
00059 };
00060
00062 enum L4virtio_device_ids
00063 {
00064     L4VIRTIO_ID_NET              = 1,
00065     L4VIRTIO_ID_BLOCK            = 2,
00066     L4VIRTIO_ID_CONSOLE          = 3,
00067     L4VIRTIO_ID_RNG              = 4,
00068     L4VIRTIO_ID_BALLOON          = 5,
00069     L4VIRTIO_ID_RPMMSG           = 7,
00070     L4VIRTIO_ID_SCSI             = 8,
00071     L4VIRTIO_ID_9P              = 9,
00072     L4VIRTIO_ID_RPROC_SERIAL     = 11,
00073     L4VIRTIO_ID_CAIF             = 12,
00074     L4VIRTIO_ID_GPU              = 16,
00075     L4VIRTIO_ID_INPUT            = 18,
00076     L4VIRTIO_ID_VSOCK            = 19,
00077     L4VIRTIO_ID_CRYPT            = 20,
00078     L4VIRTIO_ID_FS               = 26,
00079     L4VIRTIO_ID_SCSI             = 32,
00080     L4VIRTIO_ID_I2C              = 34,
00081     L4VIRTIO_ID_WATCHDOG         = 35,
00082     L4VIRTIO_ID_CAN              = 36,
00083     L4VIRTIO_ID_GPIO            = 41,
00084     L4VIRTIO_ID_SPI              = 45,
00085
00086     L4VIRTIO_ID_SOCKET           = 0x9999,
00087 };
00088
00090 enum L4virtio_device_status
00091 {
00092     L4VIRTIO_STATUS_ACKNOWLEDGE = 1,
00093     L4VIRTIO_STATUS_DRIVER      = 2,
00094     L4VIRTIO_STATUS_DRIVER_OK   = 4,
00095     L4VIRTIO_STATUS_FEATURES_OK = 8,
00096     L4VIRTIO_STATUS_DEVICE_NEEDS_RESET = 0x40,
00097     L4VIRTIO_STATUS_FAILED      = 0x80
00098 };
00099
00101 enum L4virtio_feature_bits

```

```

00102 {
00104     L4VIRTIO_FEATURE_VERSION_1 = 32,
00106     L4VIRTIO_FEATURE_CMD_CONFIG = 160
00107 };
00108
00113 enum L4_virtio_irq_status
00114 {
00115     L4VIRTIO_IRQ_STATUS_VRING = 1,
00116     L4VIRTIO_IRQ_STATUS_CONFIG = 2,
00117 };
00118
00122 enum L4_virtio_cmd
00123 {
00124     L4VIRTIO_CMD_NONE = 0x00000000,
00125     L4VIRTIO_CMD_SET_STATUS = 0x01000000,
00126     L4VIRTIO_CMD_CFG_QUEUE = 0x02000000,
00127     L4VIRTIO_CMD_CFG_CHANGED = 0x04000000,
00128     L4VIRTIO_CMD_NOTIFY_QUEUE = 0x08000000,
00129     L4VIRTIO_CMD_MASK = 0xff000000,
00130 };
00131
00135 typedef struct l4virtio_config_hdr_t
00136 {
00137     /* Virtio(0x00): device config */
00138     l4_uint32_t magic;
00139     l4_uint32_t version;
00140     l4_uint32_t device;
00141     l4_uint32_t vendor;
00142
00143     /* Virtio(0x10): device features */
00144     l4_uint32_t dev_features;
00145     l4_uint32_t dev_features_sel;
00146     l4_uint32_t _res1[2];
00147
00148     /* Virtio(0x20): driver features */
00149     l4_uint32_t driver_features;
00150     l4_uint32_t driver_features_sel;
00151
00152     /* L4Virtio(0x28): L4 queue */
00153     l4_uint32_t num_queues;
00154     l4_uint32_t queues_offset;
00155
00156     /* Virtio(0x30): queue status */
00157     l4_uint32_t queue_sel;
00158     l4_uint32_t queue_num_max;
00159     l4_uint32_t queue_num;
00160     l4_uint32_t _res3[2];
00161     l4_uint32_t queue_ready;
00162     l4_uint32_t _res4[2];
00163
00164     /* Virtio(0x50): queue notify */
00165     l4_uint32_t queue_notify;
00166     l4_uint32_t _res5[3];
00167
00168     /* Virtio(0x60): interrupt handling */
00169     l4_uint32_t irq_status;
00170     l4_uint32_t irq_ack;
00171     l4_uint32_t _res6[2];
00172
00173     /* Virtio(0x70): Device status register (read-only). The register must be
00174      * written using l4virtio_set_status(). */
00175     l4_uint32_t status;
00176
00177     /* L4Virtio(0x74): W: Event index to be used for config notifications (device to driver) */
00178     l4_uint32_t cfg_driver_notify_index;
00179     /* L4Virtio(0x78): R: Event index to be used for config notifications (driver to device) */
00180     l4_uint32_t cfg_device_notify_index;
00181
00182     /* L4Virtio(0x7c) L4 specific command register polled by the driver iff supported */
00183     l4_uint32_t cmd;
00184
00185     /* Virtio(0x80): queue descriptors */
00186     l4_uint64_t queue_desc;
00187     l4_uint32_t _res8[2];
00188     l4_uint64_t queue_avail;
00189     l4_uint32_t _res9[2];
00190     l4_uint64_t queue_used;
00191
00192     l4_uint32_t _res10[1];
00193
00194     /* Virtio(0xac): shared memory region */
00195     l4_uint32_t shm_sel;
00196     l4_uint64_t shm_len;
00197     l4_uint64_t shm_base;
00198
00199     /* L4Virtio(0xc0): use the unused space here for device and driver feature bitmaps */
00200     l4_uint32_t dev_features_map[6];

```

```

00201     l4_uint32_t _res1[2];
00202     l4_uint32_t driver_features_map[6];
00203     l4_uint32_t _res2[1];
00204
00205     /* Virtio(0xfc): config generation */
00206     l4_uint32_t generation;
00207 } l4virtio_config_hdr_t;
00208
00226 typedef struct l4virtio_config_queue_t
00227 {
00229     l4_uint16_t num_max;
00231     l4_uint16_t num;
00232
00234     l4_uint16_t ready;
00235
00237     l4_uint16_t driver_notify_index;
00238
00239     l4_uint64_t desc_addr;
00240     l4_uint64_t avail_addr;
00241     l4_uint64_t used_addr;
00242
00244     l4_uint16_t device_notify_index;
00245 } l4virtio_config_queue_t;
00246
00247 L4_BEGIN_DECLS
00248
00254 L4_INLINE l4virtio_config_queue_t *
00255 l4virtio_config_queues(l4virtio_config_hdr_t const *cfg)
00256 {
00257     return (l4virtio_config_queue_t *)(((l4_addr_t)cfg) + cfg->queues_offset);
00258 }
00259
00265 L4_INLINE void *
00266 l4virtio_device_config(l4virtio_config_hdr_t const *cfg)
00267 {
00268     return (void *)(((l4_addr_t)cfg) + 0x100);
00269 }
00270
00274 L4_INLINE void
00275 l4virtio_set_feature(l4_uint32_t *feature_map, unsigned feat)
00276 {
00277     unsigned idx = feat / 32;
00278
00279     if (idx < 8)
00280         feature_map[idx] |= 1UL << (feat % 32);
00281 }
00282
00286 L4_INLINE void
00287 l4virtio_clear_feature(l4_uint32_t *feature_map, unsigned feat)
00288 {
00289     unsigned idx = feat / 32;
00290
00291     if (idx < 8)
00292         feature_map[idx] &= ~(1UL << (feat % 32));
00293 }
00294
00298 L4_INLINE unsigned
00299 l4virtio_get_feature(l4_uint32_t *feature_map, unsigned feat)
00300 {
00301     unsigned idx = feat / 32;
00302
00303     if (idx >= 8)
00304         return 0;
00305
00306     return feature_map[idx] & (1UL << (feat % 32));
00307 }
00308
00314 L4_CV int
00315 l4virtio_set_status(l4_cap_idx_t cap, unsigned status) L4_NOTHROW;
00316
00322 L4_CV int
00323 l4virtio_config_queue(l4_cap_idx_t cap, unsigned queue) L4_NOTHROW;
00324
00330 L4_CV int
00331 l4virtio_register_ds(l4_cap_idx_t cap, l4_cap_idx_t ds_cap,
00332                     l4_uint64_t base, l4_umword_t offset,
00333                     l4_umword_t size) L4_NOTHROW;
00334
00340 L4_CV int
00341 l4virtio_device_config_ds(l4_cap_idx_t cap, l4_cap_idx_t config_ds,
00342                          l4_addr_t *ds_offset) L4_NOTHROW;
00343
00349 L4_CV int
00350 l4virtio_device_notification_irq(l4_cap_idx_t cap, unsigned index,
00351                                  l4_cap_idx_t irq) L4_NOTHROW;
00352
00353 L4_END_DECLS

```

00354

16.281 virtio_block.h

```

00001 /* SPDX-License-Identifier: MIT */
00002 /*
00003  * (c) 2014 Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00004  */
00005
00006 #pragma once
00007
00013
00014 #include <l4/sys/types.h>
00015
00019 enum L4virtio_block_operations
00020 {
00021     L4VIRTIO_BLOCK_T_IN      = 0,
00022     L4VIRTIO_BLOCK_T_OUT     = 1,
00023     L4VIRTIO_BLOCK_T_FLUSH   = 4,
00024     L4VIRTIO_BLOCK_T_GET_ID   = 8,
00025     L4VIRTIO_BLOCK_T_DISCARD = 11,
00026     L4VIRTIO_BLOCK_T_WRITE_ZEROES = 13,
00027 };
00028
00032 enum L4virtio_block_status
00033 {
00034     L4VIRTIO_BLOCK_S_OK      = 0,
00035     L4VIRTIO_BLOCK_S_IOERR   = 1,
00036     L4VIRTIO_BLOCK_S_UNSUPP  = 2
00037 };
00038
00042 typedef struct l4virtio_block_header_t
00043 {
00044     l4_uint32_t type;
00045     l4_uint32_t ioprio;
00046     l4_uint64_t sector;
00047 } l4virtio_block_header_t;
00048
00049 enum L4virtio_block_discard_flags_t
00050 {
00051     L4VIRTIO_BLOCK_DISCARD_F_UNMAP = 0x00000001UL,
00052     L4VIRTIO_BLOCK_DISCARD_F_RESERVED = 0xFFFFFFFFUL,
00053 };
00054
00058 typedef struct l4virtio_block_discard_t
00059 {
00060     l4_uint64_t sector;
00061     l4_uint32_t num_sectors;
00062     l4_uint32_t flags;
00063 } l4virtio_block_discard_t;
00064
00068 typedef struct l4virtio_block_config_t
00069 {
00070     l4_uint64_t capacity;
00071     l4_uint32_t size_max;
00072     l4_uint32_t seg_max;
00073     struct l4virtio_block_config_geometry_t
00074     {
00075         l4_uint16_t cylinders;
00076         l4_uint8_t heads;
00077         l4_uint8_t sectors;
00078     } geometry;
00079     l4_uint32_t blk_size;
00080     struct l4virtio_block_config_topology_t
00081     {
00082         l4_uint8_t physical_block_exp;
00083         l4_uint8_t alignment_offset;
00084         l4_uint16_t min_io_size;
00085         l4_uint32_t opt_io_size;
00086     } topology;
00087     l4_uint8_t writeback;
00088     l4_uint8_t unused0[1];
00089     l4_uint16_t num_queues;
00090     l4_uint32_t max_discard_sectors;
00091     l4_uint32_t max_discard_seg;
00092     l4_uint32_t discard_sector_alignment;
00093     l4_uint32_t max_write_zeroes_sectors;
00094     l4_uint32_t max_write_zeroes_seg;
00095     l4_uint8_t write_zeroes_may_unmap;
00096     l4_uint8_t unused1[3];
00097 } l4virtio_block_config_t;
00098
00102
00104 enum L4virtio_block_feature_bits

```

```

00105 {
00106     L4VIRTIO_BLOCK_F_SIZE_MAX = 1,
00107     L4VIRTIO_BLOCK_F_SEG_MAX = 2,
00108     L4VIRTIO_BLOCK_F_GEOMETRY = 4,
00109     L4VIRTIO_BLOCK_F_RO = 5,
00110     L4VIRTIO_BLOCK_F_BLK_SIZE = 6,
00111     L4VIRTIO_BLOCK_F_FLUSH = 9,
00112     L4VIRTIO_BLOCK_F_TOPOLOGY = 10,
00113     L4VIRTIO_BLOCK_F_CONFIG_WCE = 11,
00114     L4VIRTIO_BLOCK_F_DISCARD = 13,
00115     L4VIRTIO_BLOCK_F_WRITE_ZEROES = 14,
00116 };
00117

```

16.282 virtio_input.h

```

00001 /* SPDX-License-Identifier: MIT */
00002 /*
00003  * Copyright (C) 2019, 2022, 2024 Kernkonzept GmbH.
00004  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00005  */
00006 #pragma once
00007
00013
00014 #include <l4/sys/types.h>
00015
00019 enum L4virtio_input_config_select
00020 {
00021     L4VIRTIO_INPUT_CFG_UNSET = 0,
00022     L4VIRTIO_INPUT_CFG_ID_NAME = 1,
00023     L4VIRTIO_INPUT_CFG_ID_SERIAL = 2,
00024     L4VIRTIO_INPUT_CFG_ID_DEVIDS = 3,
00025     L4VIRTIO_INPUT_CFG_PROP_BITS = 0x10,
00026     L4VIRTIO_INPUT_CFG_EV_BITS = 0x11,
00027     L4VIRTIO_INPUT_CFG_ABS_INFO = 0x12
00028 };
00029
00033 typedef struct l4virtio_input_absinfo_t
00034 {
00035     l4_uint32_t min;
00036     l4_uint32_t max;
00037     l4_uint32_t fuzz;
00038     l4_uint32_t flat;
00039     l4_uint32_t res;
00040 } l4virtio_absinfo_t;
00041
00045 typedef struct l4virtio_input_devids_t
00046 {
00047     l4_uint16_t bustype;
00048     l4_uint16_t vendor;
00049     l4_uint16_t product;
00050     l4_uint16_t version;
00051 } l4virtio_input_devids_t;
00052
00056 typedef struct l4virtio_input_config_t
00057 {
00058     l4_uint8_t select;
00059     l4_uint8_t subsel;
00060     l4_uint8_t size;
00061     l4_uint8_t reserved[5];
00062     union
00063     {
00064         char string[128];
00065         l4_uint8_t bitmap[128];
00066         struct l4virtio_input_absinfo_t abs;
00067         struct l4virtio_input_devids_t ids;
00068     } u;
00069 } l4virtio_input_config_t;
00070
00074 typedef struct l4virtio_input_event_t
00075 {
00076     l4_uint16_t type;
00077     l4_uint16_t code;
00078     l4_uint32_t value;
00079 } l4virtio_input_event_t;
00080

```

16.283 virtqueue

```

00001 // vi:set ft=c++: -*- Mode: C++ -*-

```



```

00002  /* SPDX-License-Identifier: MIT */
00003  /*
00004   * (c) 2014 Alexander Warg <warg@os.inf.tu-dresden.de>
00005   */
00006
00007  #include <l4/re/util/debug>
00008  #include <l4/sys/types.h>
00009  #include <l4/sys/err.h>
00010  #include <l4/cxx/bitfield>
00011  #include <l4/cxx/exceptions>
00012  #include <cstdint>
00013
00014  #pragma once
00015
00016  namespace L4virtio {
00017
00018  #if defined(__ARM_ARCH) && __ARM_ARCH == 7
00019  static inline void wmb() { asm volatile ("dmb ishst" : : : "memory"); }
00020  static inline void rmb() { asm volatile ("dmb ish" : : : "memory"); }
00021  static inline void mb() { asm volatile ("dmb ish" : : : "memory"); }
00022  #elif defined(__ARM_ARCH) && __ARM_ARCH >= 8
00023  static inline void wmb() { asm volatile ("dmb ishst" : : : "memory"); }
00024  static inline void rmb() { asm volatile ("dmb ishld" : : : "memory"); }
00025  static inline void mb() { asm volatile ("dmb ish" : : : "memory"); }
00026  #elif defined(__mips__)
00027  static inline void wmb() { asm volatile ("sync" : : : "memory"); }
00028  static inline void rmb() { asm volatile ("sync" : : : "memory"); }
00029  static inline void mb() { asm volatile ("sync" : : : "memory"); }
00030  #elif defined(__amd64__) || defined(__i386__) || defined(__i686__)
00031  static inline void wmb() { asm volatile ("sfence" : : : "memory"); }
00032  static inline void rmb() { asm volatile ("lfence" : : : "memory"); }
00033  static inline void mb() { asm volatile ("mfence" : : : "memory"); }
00034  #elif defined(__riscv)
00035  static inline void wmb() { asm volatile ("fence ow, ow" : : : "memory"); }
00036  static inline void rmb() { asm volatile ("fence ir, ir" : : : "memory"); }
00037  static inline void mb() { asm volatile ("fence iorw, iorw" : : : "memory"); }
00038  #else
00039  #warning Missing proper memory write barrier
00040  static inline void wmb() { asm volatile ("": : : "memory"); }
00041  static inline void rmb() { asm volatile ("": : : "memory"); }
00042  static inline void mb() { asm volatile ("": : : "memory"); }
00043  #endif
00044
00045
00052  template< typename T >
00053  class Ptr
00054  {
00055  public:
00057      enum Invalid_type { Invalid };
00058
00059      Ptr() = default;
00060
00062      Ptr(Invalid_type) : _p(~0ULL) {}
00063
00065      explicit Ptr(l4_uint64_t vm_addr) : _p(vm_addr) {}
00066
00068      l4_uint64_t get() const { return _p; }
00069
00071      bool is_valid() const { return _p != ~0ULL; }
00072
00073  private:
00074      l4_uint64_t _p;
00075  };
00076
00077
00086  class Virtqueue
00087  {
00088  public:
00092      class Desc
00093      {
00094      public:
00098          struct Flags
00099          {
00100              l4_uint16_t raw;
00101              Flags() = default;
00102
00104              explicit Flags(l4_uint16_t v) : raw(v) {}
00105
00107              CXX_BITFIELD_MEMBER( 0, 0, next, raw);
00109              CXX_BITFIELD_MEMBER( 1, 1, write, raw);
00111              CXX_BITFIELD_MEMBER( 2, 2, indirect, raw);
00112          };
00113
00114          Ptr<void> addr;
00115          l4_uint32_t len;
00116          Flags flags;
00117          l4_uint16_t next;

```

```

00118
00122 void dump(unsigned idx) const
00123 {
00124     L4Re::Util::Dbg().printf("D[%04x]: %08llx (%x) f=%04x n=%04x\n",
00125                               idx, addr.get(),
00126                               len, static_cast<unsigned>(flags.raw),
00127                               static_cast<unsigned>(next));
00128 }
00129 };
00130
00134 class Avail
00135 {
00136 public:
00140     struct Flags
00141     {
00142         l4_uint16_t raw;
00143         Flags() = default;
00144
00146         explicit Flags(l4_uint16_t v) : raw(v) {}
00147
00149         CXX_BITFIELD_MEMBER( 0, 0, no_irq, raw);
00150     };
00151
00152     Flags flags;
00153     l4_uint16_t idx;
00154     l4_uint16_t ring[];
00155 };
00156
00160 struct Used_elem
00161 {
00162     Used_elem() = default;
00163
00171     Used_elem(l4_uint16_t id, l4_uint32_t len) : id(id), len(len) {}
00172     l4_uint32_t id;
00173     l4_uint32_t len;
00174 };
00175
00179 class Used
00180 {
00181 public:
00185     struct Flags
00186     {
00187         l4_uint16_t raw;
00188         Flags() = default;
00189
00191         explicit Flags(l4_uint16_t v) : raw(v) {}
00192
00194         CXX_BITFIELD_MEMBER( 0, 0, no_notify, raw);
00195     };
00196
00197     Flags flags;
00198     l4_uint16_t idx;
00199     Used_elem ring[];
00200 };
00201
00202 protected:
00203     Desc *_desc = nullptr;
00204     Avail *_avail = nullptr;
00205     Used *_used = nullptr;
00206
00208     l4_uint16_t _current_avail = 0;
00209
00214     l4_uint16_t _idx_mask = 0;
00215
00219     Virtqueue() = default;
00220
00221     Virtqueue(Virtqueue const &) = delete;
00222     ~Virtqueue() = default;
00223
00224 public:
00230 void disable()
00231 { _desc = 0; }
00232
00236 enum
00237 {
00238     Desc_align = 4, ///< Alignment of the descriptor table.
00239     Avail_align = 1, ///< Alignment of the available ring.
00240     Used_align = 2, ///< Alignment of the used ring.
00241 };
00242
00251 static unsigned long total_size(unsigned num)
00252 {
00253     static_assert(Desc_align >= Avail_align,
00254                   "virtqueue alignment assumptions broken");
00255     return l4_round_size(desc_size(num) + avail_size(num), Used_align)
00256           + used_size(num);
00257 }

```

```

00258
00267 static unsigned long desc_size(unsigned num)
00268 { return num * 16; }
00269
00275 static unsigned long desc_align()
00276 { return Desc_align; }
00277
00285 static unsigned long avail_size(unsigned num)
00286 { return 2 * num + 6; }
00287
00293 static unsigned long avail_align()
00294 { return Avail_align; }
00295
00304 static unsigned long used_size(unsigned num)
00305 { return 8 * num + 6; }
00306
00312 static unsigned long used_align()
00313 { return Used_align; }
00314
00320 unsigned long total_size() const
00321 {
00322     return (reinterpret_cast<char *>(_used) - reinterpret_cast<char *>(_desc))
00323           + used_size(num());
00324 }
00325
00329 unsigned long avail_offset() const
00330 { return reinterpret_cast<char *>(_avail) - reinterpret_cast<char *>(_desc); }
00331
00335 unsigned long used_offset() const
00336 { return reinterpret_cast<char *>(_used) - reinterpret_cast<char *>(_desc); }
00337
00355 void setup(unsigned num, void *desc, void *avail, void *used)
00356 {
00357     if (num > 0x10000)
00358         throw L4::Runtime_error(-L4_EINVAL, "Queue too large.");
00359
00360     _idx_mask = num - 1;
00361     _desc = static_cast<Desc*>(desc);
00362     _avail = static_cast<Avail*>(avail);
00363     _used = static_cast<Used*>(used);
00364
00365     _current_avail = 0;
00366
00367     L4Re::Util::Dbg().printf("VQ[%p]: num=%d d:%p a:%p u:%p\n",
00368                             this, num, _desc, _avail, _used);
00369 }
00370
00384 void setup_simple(unsigned num, void *ring)
00385 {
00386     l4_addr_t desc = reinterpret_cast<l4_addr_t>(ring);
00387     l4_addr_t avail = l4_round_size(desc + desc_size(num), Avail_align);
00388     void *used = reinterpret_cast<void *>(
00389         l4_round_size(avail + avail_size(num), Used_align));
00390     setup(num, ring, reinterpret_cast<void *>(avail), used);
00391 }
00392
00398 void dump(Desc const *d) const
00399 { d->dump(d - _desc); }
00400
00406 bool ready() const
00407 { return L4_LIKELY(_desc != 0); }
00408
00410 unsigned num() const
00411 { return _idx_mask + 1; }
00412
00420 bool no_notify_guest() const
00421 {
00422     mb(); // All queue updates must be visible before the flag can be read!
00423     return _avail->flags.no_irq();
00424 }
00425
00433 bool no_notify_host() const
00434 {
00435     return _used->flags.no_notify();
00436 }
00437
00443 void no_notify_host(bool value)
00444 {
00445     _used->flags.no_notify() = value;
00446 }
00447
00456 l4_uint16_t get_avail_idx() const { return _avail->idx; }
00457
00463 l4_uint16_t get_tail_avail_idx() const { return _current_avail; }
00464
00465 };
00466

```

```

00467 namespace Driver {
00468
00477 class Virtqueue : public L4virtio::Virtqueue
00478 {
00479 private:
00481     l4_uint16_t _next_free;
00482
00483 public:
00484     enum End_of_queue
00485     {
00486         // Indicates the end of the queue.
00487         Eoq = 0xFFFF
00488     };
00489
00490     Virtqueue() : _next_free(Eoq) {}
00491
00501     void initialize_rings(unsigned num)
00502     {
00503         _used->idx = 0;
00504         _avail->idx = 0;
00505
00506         // setup the freelist
00507         for (l4_uint16_t d = 0; d < num - 1; ++d)
00508             _desc[d].next = d + 1;
00509         _desc[num - 1].next = Eoq;
00510         _next_free = 0;
00511     }
00512
00529     void init_queue(unsigned num, void *desc, void *avail, void *used)
00530     {
00531         setup(num, desc, avail, used);
00532         initialize_rings(num);
00533     }
00534
00544     void init_queue(unsigned num, void *base)
00545     {
00546         setup_simple(num, base);
00547         initialize_rings(num);
00548     }
00549
00550
00565     l4_uint16_t alloc_descriptor()
00566     {
00567         l4_uint16_t idx = _next_free;
00568         if (idx == Eoq)
00569             return Eoq;
00570
00571         _next_free = _desc[idx].next;
00572
00573         return idx;
00574     }
00575
00581     void enqueue_descriptor(l4_uint16_t descno)
00582     {
00583         if (descno > _idx_mask)
00584             throw L4::Bounds_error();
00585
00586         _avail->ring[_avail->idx & _idx_mask] = descno; // _avail->idx expected to wrap
00587         wmb();
00588         ++_avail->idx;
00589     }
00590
00597     Desc &desc(l4_uint16_t descno)
00598     {
00599         if (descno > _idx_mask)
00600             throw L4::Bounds_error();
00601
00602         return _desc[descno];
00603     }
00604
00616     l4_uint16_t find_next_used(l4_uint32_t *len = nullptr)
00617     {
00618         if (_current_avail == _used->idx)
00619             return Eoq;
00620
00621         auto elem = _used->ring[_current_avail++ & _idx_mask];
00622
00623         if (len)
00624             *len = elem.len;
00625
00626         return elem.id;
00627     }
00628
00638     void free_descriptor(l4_uint16_t head, l4_uint16_t tail)
00639     {
00640         if (head > _idx_mask || tail > _idx_mask)
00641             throw L4::Bounds_error();

```

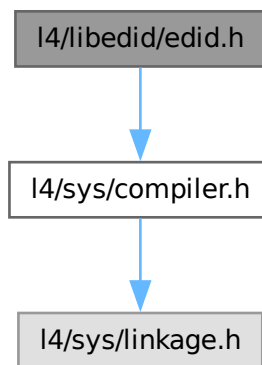
```

00642
00643     _desc[tail].next = _next_free;
00644     _next_free = head;
00645 }
00646 };
00647
00648 }
00649 } // namespace L4virtio

```

16.284 l4/libedid/edid.h File Reference

#include <l4/sys/compiler.h>
 Include dependency graph for edid.h:



Enumerations

- enum [Libedid_consts](#) { [Libedid_block_size](#) = 128 }
EDID constants.

Functions

- int [libedid_check_header](#) (const unsigned char *edid)
Check for valid EDID header.
- int [libedid_checksum](#) (const unsigned char *edid)
Calculates the EDID checksum.
- unsigned [libedid_version](#) (const unsigned char *edid)
Returns the EDID version number.
- unsigned [libedid_revision](#) (const unsigned char *edid)
Returns the EDID revision number.
- void [libedid_pnp_id](#) (const unsigned char *edid, unsigned char *id)
Extracts the display's PnP ID.
- void [libedid_preferred_resolution](#) (const unsigned char *edid, unsigned *w, unsigned *h)
Extract the display's preferred mode.
- unsigned [libedid_num_ext_blocks](#) (const unsigned char *edid)

Get the number of EDID extension blocks.

- unsigned `libedid_dump_standard_timings` (const unsigned char *edid)

Dump the standard timings to stdout.

- void `libedid_dump` (const unsigned char *edid)

Dump raw EDID data to stdout.

16.285 edid.h

[Go to the documentation of this file.](#)

```

00001
00004 /*
00005  * (c) 2014 Matthias Lange <matthias.lange@kernkonzept.com>
00006  *
00007  * This file is part of TUD:OS and distributed under the terms of the
00008  * GNU Lesser General Public License 2.1.
00009  * Please see the COPYING-LGPL-2.1 file for details.
00010  */
00011 #pragma once
00012
00013 #include <l4/sys/compiler.h>
00014
00019
00023 enum Libedid_consts
00024 {
00025     Libedid_block_size = 128,
00026 };
00027
00028 __BEGIN_DECLS
00029
00037 int libedid_check_header(const unsigned char *edid);
00038
00046 int libedid_checksum(const unsigned char *edid);
00047
00055 unsigned libedid_version(const unsigned char *edid);
00056
00064 unsigned libedid_revision(const unsigned char *edid);
00065
00072 void libedid_pnp_id(const unsigned char *edid, unsigned char *id);
00073
00081 void libedid_preferred_resolution(const unsigned char *edid,
00082                                   unsigned *w, unsigned *h);
00083
00091 unsigned libedid_num_ext_blocks(const unsigned char *edid);
00092
00100 unsigned libedid_dump_standard_timings(const unsigned char *edid);
00101
00107 void libedid_dump(const unsigned char *edid);
00108
00110
00111 __END_DECLS

```

16.286 l4/re/c/dataspace.h File Reference

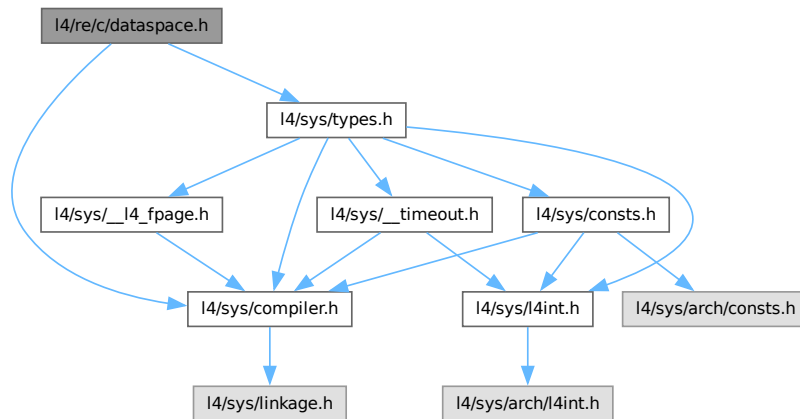
Data space C interface.

```

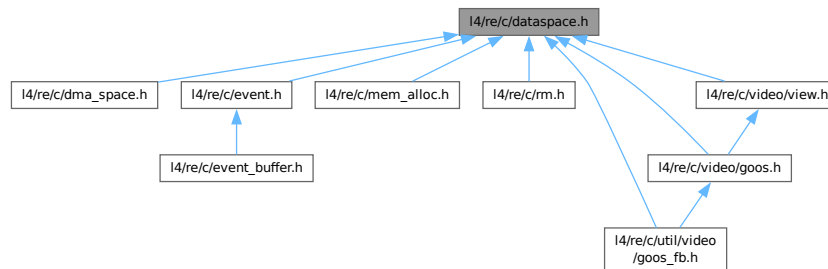
#include <l4/sys/compiler.h>
#include <l4/sys/types.h>

```

Include dependency graph for dataspace.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct `l4re_ds_stats_t`
Information about the data space.

Enumerations

- enum `l4re_ds_map_flags` { }
Flags to specify the memory mapping type of a request.

Functions

- `l4_ret_t l4re_ds_clear(l4re_ds_t ds, l4re_ds_offset_t offset, l4re_ds_size_t size)` `L4_NOTHROW`
Clear parts of a dataspace.
- `l4_ret_t l4re_ds_allocate(l4re_ds_t ds, l4re_ds_offset_t offset, l4re_ds_size_t size)` `L4_NOTHROW`
Allocate a range in the dataspace.

- `l4_ret_t l4re_ds_copy_in (l4re_ds_t ds, l4re_ds_offset_t dst_offs, l4re_ds_t src, l4re_ds_offset_t src_offs, l4re_ds_size_t size)` `L4_NOTHROW`
Copy contents from another dataspace.
- `l4re_ds_size_t l4re_ds_size (l4re_ds_t ds)` `L4_NOTHROW`
Get size of a dataspace.
- `l4re_ds_flags_t l4re_ds_flags (l4re_ds_t ds)` `L4_NOTHROW`
Get flags of the dataspace.
- `l4_ret_t l4re_ds_info (l4re_ds_t ds, l4re_ds_stats_t *stats)` `L4_NOTHROW`
Get information on the dataspace.
- `l4_ret_t l4re_ds_map_info (l4re_ds_t ds, l4_addr_t *start_addr, l4_addr_t *end_addr)` `L4_NOTHROW`
Get mapping range of dataspace.

Variables

- `L4_BEGIN_DECLS` typedef `l4_cap_idx_t l4re_ds_t`
Dataspace type.

16.286.1 Detailed Description

Data space C interface.

Definition in file [dataspace.h](#).

16.287 dataspace.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #pragma once
00013
00019
00020 #include <l4/sys/compiler.h>
00021 #include <l4/sys/types.h>
00022
00023 L4_BEGIN_DECLS
00024
00029 typedef l4_cap_idx_t l4re_ds_t;
00030 typedef l4_uint64_t l4re_ds_size_t;
00031 typedef l4_uint64_t l4re_ds_offset_t;
00032 typedef l4_uint64_t l4re_ds_map_addr_t;
00033 typedef unsigned long l4re_ds_flags_t;
00034
00039 typedef struct {
00040     l4re_ds_size_t size;
00041     l4re_ds_flags_t flags;
00042 } l4re_ds_stats_t;
00043
00048 enum l4re_ds_map_flags {
00049     L4RE_DS_F_R    = L4_FPAGE_RO,
00050     L4RE_DS_F_W    = L4_FPAGE_W,
00051     L4RE_DS_F_X    = L4_FPAGE_X,
00052     L4RE_DS_F_RW   = L4_FPAGE_RW,
00053     L4RE_DS_F_RX   = L4_FPAGE_RX,
00054     L4RE_DS_F_RWX  = L4_FPAGE_RWX,
00055
00056     L4RE_DS_F_RIGHTS_MASK = 0x0f,
00057

```



```

00058  L4RE_DS_F_NORMAL          = 0x00,
00059  L4RE_DS_F_CACHEABLE      = L4RE_DS_F_NORMAL,
00060  L4RE_DS_F_BUFFERABLE     = 0x10,
00061  L4RE_DS_F_UNCACHEABLE    = 0x20,
00062  L4RE_DS_F_CACHING_MASK   = 0x30,
00063  L4RE_DS_F_CACHING_SHIFT  = 4,
00064  };
00065
00071  L4_CV l4_ret_t
00072  l4re_ds_map(l4re_ds_t ds,
00073              l4re_ds_offset_t offset,
00074              l4re_ds_flags_t flags,
00075              l4re_ds_map_addr_t local_addr,
00076              l4re_ds_map_addr_t min_addr,
00077              l4re_ds_map_addr_t max_addr) L4_NOTHROW;
00078
00084  L4_CV l4_ret_t
00085  l4re_ds_map_region(l4re_ds_t ds,
00086                    l4re_ds_offset_t offset,
00087                    l4re_ds_flags_t flags,
00088                    l4re_ds_map_addr_t min_addr,
00089                    l4re_ds_map_addr_t max_addr) L4_NOTHROW;
00090
00097  L4_CV l4_ret_t
00098  l4re_ds_clear(l4re_ds_t ds, l4re_ds_offset_t offset,
00099               l4re_ds_size_t size) L4_NOTHROW;
00100
00107  L4_CV l4_ret_t
00108  l4re_ds_allocate(l4re_ds_t ds,
00109                  l4re_ds_offset_t offset,
00110                  l4re_ds_size_t size) L4_NOTHROW;
00111
00118  L4_CV l4_ret_t
00119  l4re_ds_copy_in(l4re_ds_t ds, l4re_ds_offset_t dst_offs,
00120                 l4re_ds_t src, l4re_ds_offset_t src_offs,
00121                 l4re_ds_size_t size) L4_NOTHROW;
00122
00129  L4_CV l4re_ds_size_t
00130  l4re_ds_size(l4re_ds_t ds) L4_NOTHROW;
00131
00138  L4_CV l4re_ds_flags_t
00139  l4re_ds_flags(l4re_ds_t ds) L4_NOTHROW;
00140
00147  L4_CV l4_ret_t
00148  l4re_ds_info(l4re_ds_t ds, l4re_ds_stats_t *stats) L4_NOTHROW;
00149
00156  L4_CV l4_ret_t
00157  l4re_ds_map_info(l4re_ds_t ds,
00158                  l4_addr_t *start_addr, l4_addr_t *end_addr) L4_NOTHROW;
00159
00160  L4_END_DECLS

```

16.288 l4/re/c/dma_space.h File Reference

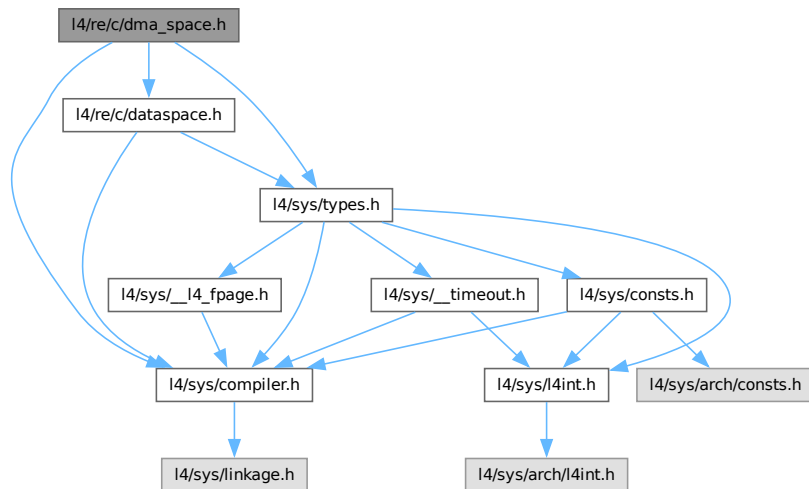
DMA space C interface.

```

#include <l4/sys/compiler.h>
#include <l4/sys/types.h>
#include <l4/re/c/dataspace.h>

```

Include dependency graph for dma_space.h:



Typedefs

- typedef `l4_cap_idx_t l4re_dma_space_t`
DMA space capability type.
- typedef `l4_uint64_t l4re_dma_space_dma_addr_t`
Data type for DMA addresses.

Enumerations

- enum `l4re_dma_space_direction` { `L4RE_DMA_SPACE_BIDIRECTIONAL` , `L4RE_DMA_SPACE_TO_DEVICE` , `L4RE_DMA_SPACE_FROM_DEVICE` , `L4RE_DMA_SPACE_NONE` }
Direction of the DMA transfers.
- enum `l4re_dma_space_space_attrbs` { `L4RE_DMA_SPACE_COHERENT` = 1 << 0 , `L4RE_DMA_SPACE_PHYS_SPACE` = 1 << 1 }
Attributes assigned to the DMA space when associated with a specific device.

Functions

- `l4_ret_t l4re_dma_space_map` (`l4re_dma_space_t` dma, `l4re_ds_t` src, `l4re_ds_offset_t` offset, `l4_size_t` *size, unsigned long attr, enum `l4re_dma_space_direction` dir, `l4re_dma_space_dma_addr_t` *dma_addr) `L4_NOTHROW`
Map the given part of this data space into the DMA address space.
- `l4_ret_t l4re_dma_space_unmap` (`l4re_dma_space_t` dma, `l4re_dma_space_dma_addr_t` dma_addr, `l4_size_t` size, unsigned long attr, enum `l4re_dma_space_direction` dir) `L4_NOTHROW`
Unmap the given part of this data space from the DMA address space.
- `l4_ret_t l4re_dma_space_associate` (`l4re_dma_space_t` dma, `l4_cap_idx_t` dma_task, unsigned long attr) `L4_NOTHROW`
Associate a (kernel) DMA space for a device to this Dma_space.
- `l4_ret_t l4re_dma_space_disassociate` (`l4re_dma_space_t` dma)
Disassociate the (kernel) DMA space from this Dma_space.

16.288.1 Detailed Description

DMA space C interface.

Definition in file [dma_space.h](#).

16.288.2 Enumeration Type Documentation

16.288.2.1 l4re_dma_space_direction

enum [l4re_dma_space_direction](#)

Direction of the DMA transfers.

Enumerator

| | |
|------------------------------|---------------------------------------|
| L4RE_DMA_SPACE_BIDIRECTIONAL | device reads and writes to the memory |
| L4RE_DMA_SPACE_TO_DEVICE | device reads the memory |
| L4RE_DMA_SPACE_FROM_DEVICE | device writes to the memory |
| L4RE_DMA_SPACE_NONE | device is coherently connected |

Definition at line 27 of file [dma_space.h](#).

16.288.2.2 l4re_dma_space_space_attrbs

enum [l4re_dma_space_space_attrbs](#)

Attributes assigned to the DMA space when associated with a specific device.

See also

[Space_attrbs](#)

Enumerator

| | |
|---------------------------|---|
| L4RE_DMA_SPACE_COHERENT | The device is connected coherently with the cache. This means that the <code>map()</code> and <code>unmap()</code> do not need to sync CPU caches before and after DMA. |
| L4RE_DMA_SPACE_PHYS_SPACE | The DMA space has no DMA task assigned and uses the CPUs physical memory. |

Definition at line 38 of file [dma_space.h](#).

16.289 dma_space.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00007  *
00008  * License: see LICENSE.spdx (in this directory or the directories above)
00009  */
00010 #pragma once
00011
00017
00018 #include <l4/sys/compiler.h>
00019 #include <l4/sys/types.h>
00020 #include <l4/re/c/dataspace.h>
00021
00022 L4_BEGIN_DECLS
00023
00027 enum l4re_dma_space_direction
00028 {
00029     L4RE_DMA_SPACE_BIDIRECTIONAL,
00030     L4RE_DMA_SPACE_TO_DEVICE,
00031     L4RE_DMA_SPACE_FROM_DEVICE,
00032     L4RE_DMA_SPACE_NONE
00033 };
00034
00038 enum l4re_dma_space_space_attrbs
00039 {
00040     L4RE_DMA_SPACE_COHERENT = 1 << 0,
00041     L4RE_DMA_SPACE_PHYS_SPACE = 1 << 1,
00042 };
00043
00049 typedef l4_cap_idx_t l4re_dma_space_t;
00050
00052 typedef l4_uint64_t l4re_dma_space_dma_addr_t;
00053
00060 L4_CV l4_ret_t
00061 l4re_dma_space_map(l4re_dma_space_t dma, l4re_ds_t src,
00062                   l4re_ds_offset_t offset,
00063                   l4_size_t * size, unsigned long attrs,
00064                   enum l4re_dma_space_direction dir,
00065                   l4re_dma_space_dma_addr_t *dma_addr) L4_NOTHROW;
00066
00067
00074 L4_CV l4_ret_t
00075 l4re_dma_space_unmap(l4re_dma_space_t dma, l4re_dma_space_dma_addr_t dma_addr,
00076                     l4_size_t size, unsigned long attrs,
00077                     enum l4re_dma_space_direction dir) L4_NOTHROW;
00078
00085 L4_CV l4_ret_t
00086 l4re_dma_space_associate(l4re_dma_space_t dma, l4_cap_idx_t dma_task,
00087                          unsigned long attr) L4_NOTHROW;
00088
00095 L4_CV l4_ret_t
00096 l4re_dma_space_disassociate(l4re_dma_space_t dma);
00097
00098
00099 L4_END_DECLS

```

16.290 l4/re/c/event.h File Reference

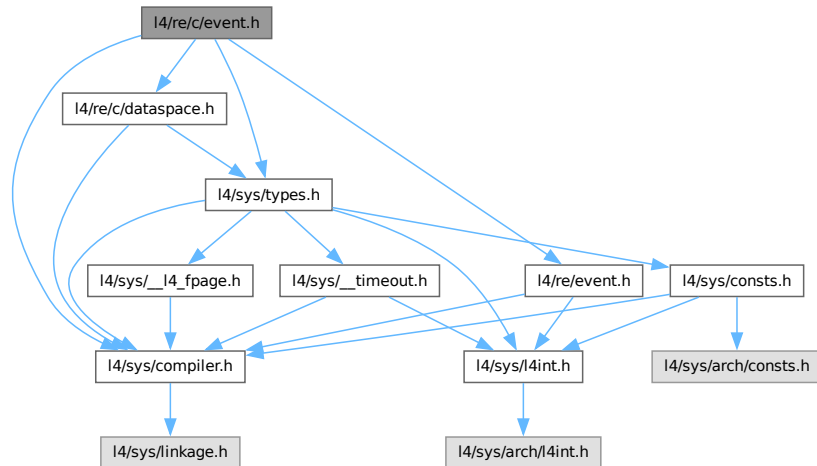
Event C interface.

```

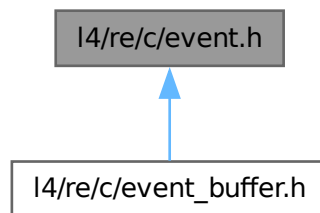
#include <l4/sys/compiler.h>
#include <l4/sys/types.h>
#include <l4/re/c/dataspace.h>
#include <l4/re/event.h>

```

Include dependency graph for event.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [l4re_event_t](#)
Event structure used in buffer.

Functions

- [l4_ret_t l4re_event_get_buffer](#) (const [l4_cap_idx_t](#) server, const [l4re_ds_t](#) ds) [L4_NOTHROW](#)
Get an event signal buffer.
- [l4_ret_t l4re_event_get_num_streams](#) (const [l4_cap_idx_t](#) server) [L4_NOTHROW](#)
Get number of streams.
- [l4_ret_t l4re_event_get_stream_info](#) (const [l4_cap_idx_t](#) server, int idx, [l4re_event_stream_info_t](#) *info) [L4_NOTHROW](#)
Get information on a stream.

- `l4_ret_t l4re_event_get_stream_info_for_id` (const `l4_cap_idx_t` server, `l4_umword_t` stream_id, `l4re_event_stream_info_t` *info) `L4_NOTHROW`
Get info for a stream given a stream id.
- `l4_ret_t l4re_event_get_axis_info` (const `l4_cap_idx_t` server, `l4_umword_t` id, unsigned naxes, unsigned const *axis, `l4re_event_absinfo_t` *info) `L4_NOTHROW`
Get Axis information for a stream.

16.290.1 Detailed Description

Event C interface.

Definition in file [event.h](#).

16.291 event.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #pragma once
00013
00019
00020 #include <l4/sys/compiler.h>
00021 #include <l4/sys/types.h>
00022 #include <l4/re/c/dataspace.h>
00023 #include <l4/re/event.h>
00024
00025 L4_BEGIN_DECLS
00026
00030 typedef struct
00031 {
00032     long long time;
00033     unsigned short type;
00034     unsigned short code;
00035     int value;
00036     l4_umword_t stream_id;
00037 } l4re_event_t;
00038
00050 L4_CV l4_ret_t
00051 l4re_event_get_buffer(const l4_cap_idx_t server,
00052                      const l4re_ds_t ds) L4_NOTHROW;
00053
00064 L4_CV l4_ret_t
00065 l4re_event_get_num_streams(const l4_cap_idx_t server) L4_NOTHROW;
00066
00079 L4_CV l4_ret_t
00080 l4re_event_get_stream_info(const l4_cap_idx_t server,
00081                           int idx, l4re_event_stream_info_t *info) L4_NOTHROW;
00082
00095 L4_CV l4_ret_t
00096 l4re_event_get_stream_info_for_id(const l4_cap_idx_t server,
00097                                  l4_umword_t stream_id,
00098                                  l4re_event_stream_info_t *info) L4_NOTHROW;
00099
00115 L4_CV l4_ret_t
00116 l4re_event_get_axis_info(const l4_cap_idx_t server, l4_umword_t id,
00117                          unsigned naxes, unsigned const *axis,
00118                          l4re_event_absinfo_t *info) L4_NOTHROW;
00119
00120 L4_END_DECLS

```

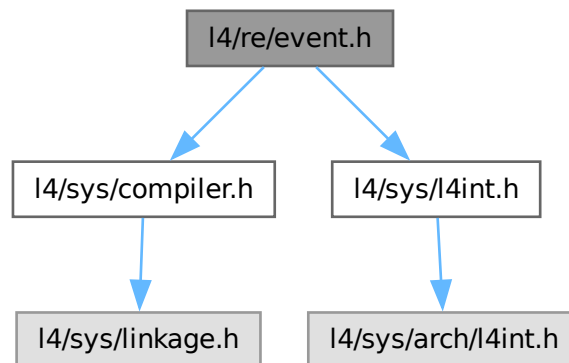
16.292 l4/re/event.h File Reference

Events.

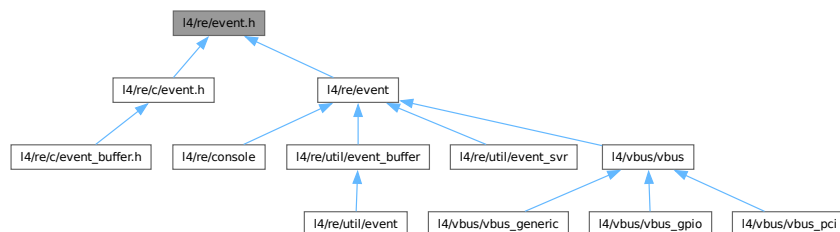
```
#include <l4/sys/compiler.h>
```

```
#include <l4/sys/l4int.h>
```

Include dependency graph for event.h:



This graph shows which files directly or indirectly include this file:



16.292.1 Detailed Description

Events.

Definition in file [event.h](#).

16.293 event.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00007  *      economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011 #pragma once
00012
00013 #include <l4/sys/compiler.h>
00014 #include <l4/sys/l4int.h>
00015
00016 typedef struct L4_EXPORT_TYPE l4re_event_stream_id_t
00017 {
00018     l4_uint16_t bustype;
00019     l4_uint16_t vendor;
00020     l4_uint16_t product;
00021     l4_uint16_t version;
00022 } l4re_event_stream_id_t;
00023
00024 typedef struct L4_EXPORT_TYPE l4re_event_absinfo_t
00025 {
00026     l4_int32_t value;
00027     l4_int32_t min;
00028     l4_int32_t max;
00029     l4_int32_t fuzz;
00030     l4_int32_t flat;
00031     l4_int32_t resolution;
00032 } l4re_event_absinfo_t;
00033
00034 enum l4re_event_stream_max_values_t
00035 {
00036     L4RE_EVENT_EV_MAX    = 0x1f,
00037     L4RE_EVENT_KEY_MAX   = 0x1ff,
00038     L4RE_EVENT_REL_MAX   = 0xf,
00039     L4RE_EVENT_ABS_MAX   = 0x3f,
00040     L4RE_EVENT_PROP_MAX  = 0x1f,
00041     L4RE_EVENT_SW_MAX    = 0xf, // should be >= L4RE_SW_MAX
00042 };
00043
00044 enum l4re_event_stream_props_t
00045 {
00046     L4RE_EVENT_STREAM_CALIBRATE = 0x001,
00047 };
00048
00049
00050 #define __UNUM_B(x) ((x+1) + sizeof(unsigned long)*8 - 1) / (sizeof(unsigned long)*8)
00051
00052 typedef struct L4_EXPORT_TYPE l4re_event_stream_info_t
00053 {
00054     l4_umword_t stream_id;
00055     char name[32];
00056     char phys[32];
00057     l4re_event_stream_id_t id;
00058
00059     unsigned long propbits[__UNUM_B(L4RE_EVENT_PROP_MAX)];
00060
00061     unsigned long evbits[__UNUM_B(L4RE_EVENT_EV_MAX)];
00062     unsigned long keybits[__UNUM_B(L4RE_EVENT_KEY_MAX)];
00063     unsigned long relbits[__UNUM_B(L4RE_EVENT_REL_MAX)];
00064     unsigned long absbits[__UNUM_B(L4RE_EVENT_ABS_MAX)];
00065     unsigned long swbits[__UNUM_B(L4RE_EVENT_SW_MAX)];
00066 } l4re_event_stream_info_t;
00067
00068
00069 typedef struct L4_EXPORT_TYPE l4re_event_stream_state_t
00070 {
00071     unsigned long keybits[__UNUM_B(L4RE_EVENT_KEY_MAX)];
00072     unsigned long swbits[__UNUM_B(L4RE_EVENT_SW_MAX)];
00073 } l4re_event_stream_state_t;
00074
00075 #undef __UNUM_B
00076

```

16.294 event_buffer.h

```

00001 #pragma once

```



```

00002  /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00004  *     economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008
00009 #include <l4/sys/compiler.h>
00010 #include <l4/sys/linkage.h>
00011 #include <l4/re/c/event.h>
00012
00013 L4_BEGIN_DECLS
00014
00015 typedef struct l4re_event_buffer_consumer_t
00016 {
00017     unsigned long _obj_buf[8];
00018 } l4re_event_buffer_consumer_t;
00019
00020 L4_CV void
00021 l4re_event_free(l4re_event_t *e) L4_NOTHROW;
00022
00023 L4_CV l4_ret_t
00024 l4re_event_buffer_attach(l4re_event_buffer_consumer_t *evbuf,
00025                          l4re_ds_t ds, l4_cap_idx_t rm) L4_NOTHROW;
00026
00027 L4_CV l4_ret_t
00028 l4re_event_buffer_detach(l4re_event_buffer_consumer_t *evbuf,
00029                          l4_cap_idx_t rm) L4_NOTHROW;
00030
00031 L4_CV l4re_event_t *
00032 l4re_event_buffer_next(l4re_event_buffer_consumer_t *evbuf) L4_NOTHROW;
00033
00034 typedef L4_CV void l4re_event_buffer_cb_t(l4re_event_t *ev, void *data);
00035
00036 L4_CV void
00037 l4re_event_buffer_consumer_foreach_available_event(l4re_event_buffer_consumer_t *evbuf,
00038                                                    void *data, l4re_event_buffer_cb_t *cb);
00039
00040
00041 L4_CV void
00042 l4re_event_buffer_consumer_process(l4re_event_buffer_consumer_t *evbuf,
00043                                   l4_cap_idx_t irq, l4_cap_idx_t thread, void *data,
00044                                   l4re_event_buffer_cb_t *cb);
00045
00046 L4_END_DECLS

```

16.295 l4/re/c/inhibitor.h File Reference

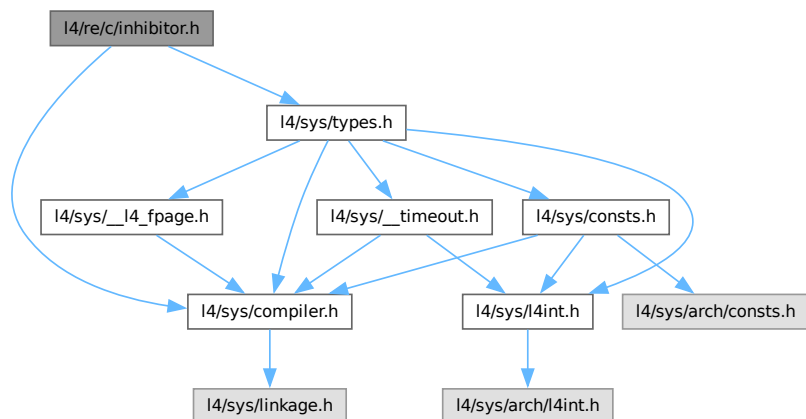
Inhibitor C interface.

```

#include <l4/sys/compiler.h>
#include <l4/sys/types.h>

```

Include dependency graph for inhibitor.h:



Functions

- [L4_BEGIN_DECLS l4_ret_t l4re_inhibitor_acquire](#) ([l4_cap_idx_t](#) cap, [l4_umword_t](#) id, char const *reason)
Acquire an inhibitor lock.
- [l4_ret_t l4re_inhibitor_release](#) ([l4_cap_idx_t](#) cap, [l4_umword_t](#) id)
Release an inhibitor lock.
- [l4_ret_t l4re_inhibitor_next_lock_info](#) ([l4_cap_idx_t](#) cap, char *name, unsigned len, [l4_mword_t](#) current_id)
Get information for the next available inhibitor lock.

16.295.1 Detailed Description

Inhibitor C interface.

Definition in file [inhibitor.h](#).

16.295.2 Function Documentation

16.295.2.1 l4re_inhibitor_acquire()

```
L4_BEGIN_DECLS l4_ret_t l4re_inhibitor_acquire (
    l4_cap_idx_t cap,
    l4_umword_t id,
    char const * reason)
```

Acquire an inhibitor lock.

Parameters

| | |
|---------------|--|
| <i>cap</i> | Capability for the Inhibitor object (see L4Re::Inhibitor) |
| <i>id</i> | ID of the inhibitor lock that shall be acquired. |
| <i>reason</i> | Reason why the inhibitor lock is acquired. (Used for informing the user or debugging.) |

Returns

0 for success, <0 on error.

See also

[L4Re::Inhibitor::acquire\(\)](#).

References [L4_CV](#), and [L4_EXPORT](#).

16.295.2.2 l4re_inhibitor_next_lock_info()

```
l4_ret_t l4re_inhibitor_next_lock_info (
    l4_cap_idx_t cap,
    char * name,
    unsigned len,
    l4_mword_t current_id)
```

Get information for the next available inhibitor lock.

Parameters

| | |
|-------------------|---|
| <i>cap</i> | Capability to the Inhibitor object (see L4Re::Inhibitor) |
| <i>name</i> | A pointer to a buffer for the name of the lock. |
| <i>len</i> | The length of the available buffer (usually L4Re::Inhibitor::Name_max is used). |
| <i>current_id</i> | The ID of the last available lock, use -1 to get the first lock. |

Return values

| | |
|-------------------|---|
| <i>>0</i> | The ID of the next available lock if there is one (in this case name shall contain the name of the inhibitor lock). |
| <i>-L4_ENODEV</i> | if there are no more locks. |
| <i><0</i> | Any other negative failure value. |

See also

[L4Re::Inhibitor::next_lock_info\(\)](#).

References [L4_END_DECLS](#).

16.295.2.3 l4re_inhibitor_release()

```
l4_ret_t l4re_inhibitor_release (
    l4_cap_idx_t cap,
    l4_umword_t id)
```

Release an inhibitor lock.

Parameters

| | |
|------------|---|
| <i>cap</i> | Capability for the Inhibitor object (see L4Re::Inhibitor). |
| <i>id</i> | ID of inhibitor that shall be released. |

Returns

0 for success, <0 on error.

See also

[L4Re::Inhibitor::release\(\)](#).

References [L4_CV](#), and [L4_EXPORT](#).

16.296 inhibitor.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * (c) 2014 Steffen Liebergeld <steffen.liebergeld@kernkonzept.com>
00003  *
00004  * License: see LICENSE.spdx (in this directory or the directories above)
00005  */
00006 #pragma once
00007
00013 #include <l4/sys/compiler.h>
00014 #include <l4/sys/types.h>
00015
00016 L4_BEGIN_DECLS
00017
00028 L4_CV l4_ret_t L4_EXPORT
00029 l4re_inhibitor_acquire(l4_cap_idx_t cap, l4_umword_t id,
00030                       char const *reason);
00031
00039 L4_CV l4_ret_t L4_EXPORT
00040 l4re_inhibitor_release(l4_cap_idx_t cap, l4_umword_t id);
00041
00057 L4_CV l4_ret_t L4_EXPORT
00058 l4re_inhibitor_next_lock_info(l4_cap_idx_t cap, char *name,
00059                               unsigned len, l4_mword_t current_id);
00060
00061 L4_END_DECLS
00062

```

16.297 I4/re/c/log.h File Reference

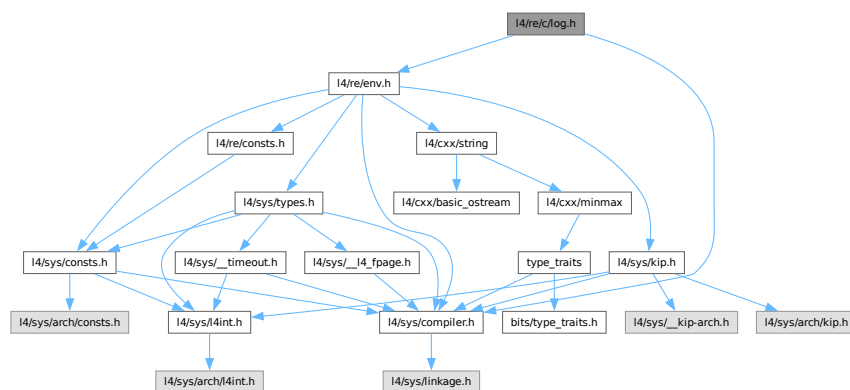
Log C interface.

```

#include <l4/re/env.h>
#include <l4/sys/compiler.h>

```

Include dependency graph for log.h:



Functions

- [L4_BEGIN_DECLS](#) void [l4re_log_print](#) (char const *string) [L4_NOTHROW](#)
Write a null terminated string to the default log.
- void [l4re_log_printn](#) (char const *string, int len) [L4_NOTHROW](#)
Write a string of a given length to the default log.
- void [l4re_log_print_srv](#) (const [l4_cap_idx_t](#) logcap, char const *string) [L4_NOTHROW](#)
Write a null terminated string to a log.
- void [l4re_log_printn_srv](#) (const [l4_cap_idx_t](#) logcap, char const *string, int len) [L4_NOTHROW](#)
Write a string of a given length to a log.

16.297.1 Detailed Description

Log C interface.

Definition in file [log.h](#).

16.298 log.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *      economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011 #pragma once
00012
00018
00019 #include <l4/re/env.h>
00020 #include <l4/sys/compiler.h>
00021
00022 L4_BEGIN_DECLS
00023
00032 L4_CV L4_INLINE void
00033 l4re_log_print(char const *string) L4_NOTHROW;
00034
00044 L4_CV L4_INLINE void
00045 l4re_log_printn(char const *string, int len) L4_NOTHROW;
00046
00047
00048
00049
00059 L4_CV void
00060 l4re_log_print_srv(const l4_cap_idx_t logcap,
00061                  char const *string) L4_NOTHROW;
00062
00073 L4_CV void
00074 l4re_log_printn_srv(const l4_cap_idx_t logcap,
00075                   char const *string, int len) L4_NOTHROW;
00076
00077
00078 /***** Implementations *****/
00079
00080 L4_CV L4_INLINE void
00081 l4re_log_print(char const *string) L4_NOTHROW
00082 {
00083     l4re_log_print_srv(l4re_global_env->log, string);
00084 }
00085
00086 L4_CV L4_INLINE void
00087 l4re_log_printn(char const *string, int len) L4_NOTHROW
00088 {
00089     l4re_log_printn_srv(l4re_global_env->log, string, len);
00090 }
00091
00092 L4_END_DECLS

```

16.299 l4/re/c/mem_alloc.h File Reference

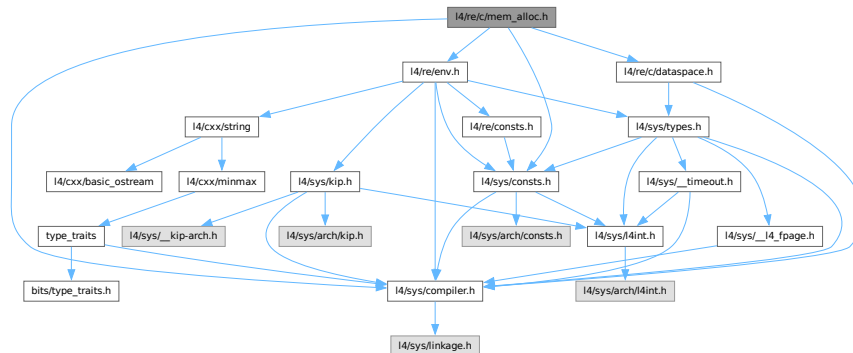
Memory allocator C interface.

```

#include <l4/re/env.h>
#include <l4/sys/compiler.h>
#include <l4/sys/consts.h>

```

```
#include <l4/re/c/dataspace.h>
Include dependency graph for mem_alloc.h:
```



Enumerations

- enum [l4re_ma_flags](#)
Flags for requesting memory at the memory allocator.

Functions

- [l4_ret_t l4re_ma_alloc](#) (long size, [l4re_ds_t](#) const mem, unsigned long flags) [L4_NOTHROW](#)
Allocate memory.
- [l4_ret_t l4re_ma_alloc_align](#) (long size, [l4re_ds_t](#) const mem, unsigned long flags, unsigned long align) [L4_NOTHROW](#)
Allocate memory.
- [l4_ret_t l4re_ma_alloc_align_srv](#) ([l4_cap_idx_t](#) srv, long size, [l4re_ds_t](#) const mem, unsigned long flags, unsigned long align) [L4_NOTHROW](#)
Allocate memory.

16.299.1 Detailed Description

Memory allocator C interface.

Definition in file [mem_alloc.h](#).

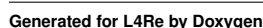
16.300 mem_alloc.h

[Go to the documentation of this file.](#)

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011 #pragma once
00012
00013 #include <l4/re/env.h>
```

16.301 l4/re/c/namespace.h File Reference

```
#include <l4/re/env.h>
#include <l4/sys/compiler.h>
Include dependency graph for namespace.h:
```



Enumerations

- enum [l4re_ns_register_flags](#)
Namespace register flags.

Functions

- [l4_ret_t l4re_ns_query_to_srv](#) ([l4re_namespace_t](#) srv, char const *name, [l4_cap_idx_t](#) const cap, int timeout)
[L4_NOTHROW](#)
Query the name space for the object named by name.
- [l4_ret_t l4re_ns_query_srv](#) ([l4re_namespace_t](#) srv, char const *name, [l4_cap_idx_t](#) const cap)
[L4_NOTHROW](#)
Query the name space for the object named by name.
- [l4_ret_t l4re_ns_register_obj_srv](#) ([l4re_namespace_t](#) srv, char const *name, [l4_cap_idx_t](#) const obj, unsigned flags) [L4_NOTHROW](#)
Register an object with a name.

Variables

- [L4_BEGIN_DECLS](#) typedef [l4_cap_idx_t l4re_namespace_t](#)
Namespace type.

16.301.1 Detailed Description

Namespace functions, C interface.

Definition in file [namespace.h](#).

16.302 namespace.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #pragma once
00013
00019
00020 #include <l4/re/env.h>
00021 #include <l4/sys/compiler.h>
00022
00029 enum l4re_ns_register_flags {
00030     L4RE_NS_REGISTER_RO = L4_FPAGE_RO,
00031     L4RE_NS_REGISTER_DIR = 0x10,
00032     L4RE_NS_REGISTER_RW = L4_FPAGE_RX,
00033     L4RE_NS_REGISTER_RWS = L4_FPAGE_RWX,
00034     L4RE_NS_REGISTER_S   = L4_FPAGE_W,
00035 };
00036
00037 L4_BEGIN_DECLS
00038
00043 typedef l4_cap_idx_t l4re_namespace_t;
00044
00045
00046

```



```

00055 L4_CV l4_ret_t
00056 l4re_ns_query_to_srv(l4re_namespace_t srv, char const *name,
00057                      l4_cap_idx_t const cap, int timeout) L4_NOTHROW;
00058
00075 L4_CV L4_INLINE l4_ret_t
00076 l4re_ns_query_srv(l4re_namespace_t srv, char const *name,
00077                  l4_cap_idx_t const cap) L4_NOTHROW;
00078
00086 L4_CV l4_ret_t
00087 l4re_ns_register_obj_srv(l4re_namespace_t srv, char const *name,
00088                         l4_cap_idx_t const obj, unsigned flags) L4_NOTHROW;
00089
00090
00091
00092 /***** Implementation *****/
00093
00094 L4_CV L4_INLINE l4_ret_t
00095 l4re_ns_query_srv(l4re_namespace_t srv, char const *name,
00096                  l4_cap_idx_t const cap) L4_NOTHROW
00097 {
00098     return l4re_ns_query_to_srv(srv, name, cap, 40000);
00099 }
00100
00101
00102 L4_END_DECLS

```

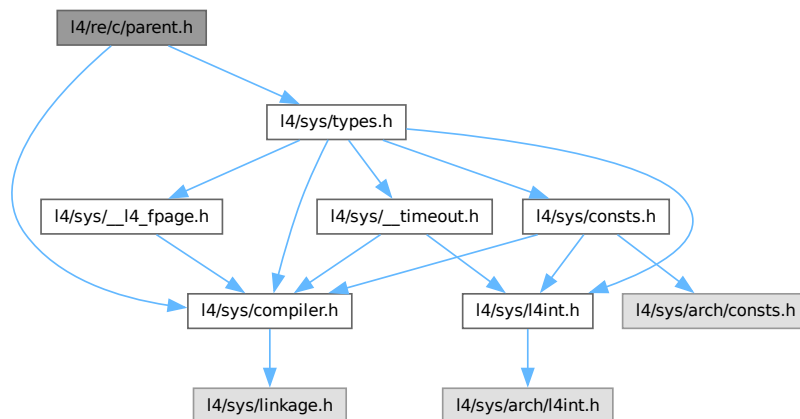
16.303 l4/re/c/parent.h File Reference

Parent C interface.

```
#include <l4/sys/compiler.h>
```

```
#include <l4/sys/types.h>
```

Include dependency graph for parent.h:



Functions

- [L4_BEGIN_DECLS](#) long [l4re_parent_signal](#) ([l4_cap_idx_t](#) parent, unsigned long sig, unsigned long val)
Send a signal using the parent protocol.

16.303.1 Detailed Description

Parent C interface.

Definition in file [parent.h](#).

16.303.2 Function Documentation

16.303.2.1 l4re_parent_signal()

```
L4_BEGIN_DECLS long l4re_parent_signal (  
    l4_cap_idx_t parent,  
    unsigned long sig,  
    unsigned long val)
```

Send a signal using the parent protocol.

Parameters

| | |
|---------------|---|
| <i>parent</i> | Capability index of parent to send signal to. |
| <i>sig</i> | Signal to send |
| <i>val</i> | Value of the signal |

Return values

| | |
|----|-----------|
| 0 | Success |
| <0 | IPC error |

See also

[L4Re::Parent::signal](#)

References [L4_END_DECLS](#).

16.304 parent.h

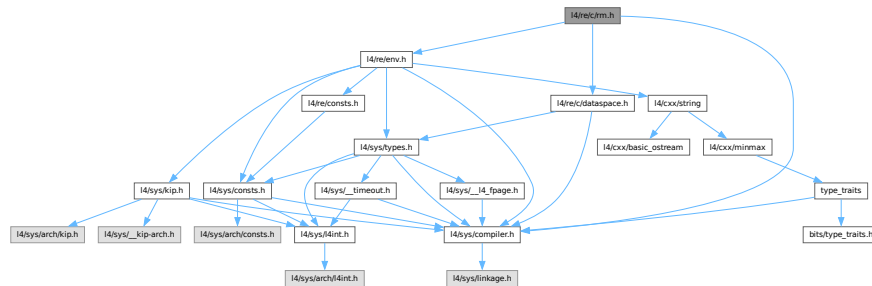
[Go to the documentation of this file.](#)

```
00001 /*  
00002  * Copyright (C) 2024-2025 Kernkonzept GmbH.  
00003  * Author(s): Marcus Haehnel <marcus.haehnel@kernkonzept.com>  
00004  *  
00005  * License: see LICENSE.spdx (in this directory or the directories above)  
00006  */  
00007  
00012  
00013 #pragma once  
00014  
00019  
00020 #include <l4/sys/compiler.h>  
00021 #include <l4/sys/types.h>  
00022  
00023 L4_BEGIN_DECLS  
00024  
00037 L4_CV long L4_EXPORT  
00038 l4re_parent_signal(l4_cap_idx_t parent, unsigned long sig, unsigned long val);  
00039  
00040 L4_END_DECLS
```

16.305 l4/re/c/rm.h File Reference

Region map interface, C interface.

```
#include <l4/re/env.h>
#include <l4/re/c/dataspace.h>
#include <l4/sys/compiler.h>
Include dependency graph for rm.h:
```



Enumerations

- enum `l4re_rm_flags_values` {
`L4RE_RM_F_R` = `L4RE_DS_F_R` , `L4RE_RM_F_W` = `L4RE_DS_F_W` , `L4RE_RM_F_X` = `L4RE_DS_F_X`
, `L4RE_RM_F_RX` = `L4RE_DS_F_RX` ,
`L4RE_RM_F_RW` = `L4RE_DS_F_RW` , `L4RE_RM_F_RWX` = `L4RE_DS_F_RWX` , `L4RE_RM_F_KERNEL`
= `0x100` , `L4RE_RM_F_DETACH_FREE` = `0x200` ,
`L4RE_RM_F_PAGER` = `0x400` , `L4RE_RM_F_RESERVED` = `0x800` , `L4RE_RM_CACHING_SHIFT` = `4` ,
`L4RE_RM_F_CACHING` = `L4RE_DS_F_CACHING_MASK` ,
`L4RE_RM_REGION_FLAGS` = `0xffff` , `L4RE_RM_F_CACHE_NORMAL` = `L4RE_DS_F_NORMAL` ,
`L4RE_RM_F_CACHE_BUFFERED` = `L4RE_DS_F_BUFFERABLE` , `L4RE_RM_F_CACHE_UNCACHED`
= `L4RE_DS_F_UNCACHEABLE` ,
`L4RE_RM_F_SEARCH_ADDR` = `0x020000` , `L4RE_RM_F_IN_AREA` = `0x040000` , `L4RE_RM_F_EAGER_MAP`
= `0x080000` , `L4RE_RM_F_NO_EAGER_MAP` = `0x100000` ,
`L4RE_RM_F_ATTACH_FLAGS` = `0x1f0000` }

Flags for region operations.

Functions

- int `l4re_rm_reserve_area` (`l4_addr_t` *start, unsigned long size, `l4re_rm_flags_t` flags, unsigned char align) `L4_NOTHROW`
Reserve the given area in the region map.
- int `l4re_rm_free_area` (`l4_addr_t` addr) `L4_NOTHROW`
Free an area from the region map.
- int `l4re_rm_attach` (void **start, unsigned long size, `l4re_rm_flags_t` flags, `l4re_ds_t` mem, `l4re_rm_offset_t` offs, unsigned char align) `L4_NOTHROW`
Attach a data space to a region.
- int `l4re_rm_attach_w_info` (void **start, unsigned long size, `l4re_rm_flags_t` flags, `l4re_ds_t` mem, `l4re_rm_offset_t` offs, unsigned char align, char const *name, `l4re_ds_offset_t` backing_offset) `L4_NOTHROW`
Attach a data space to a region.
- int `l4re_rm_detach` (void *addr) `L4_NOTHROW`
Detach and unmap a region from the address space in the current task.

- `int l4re_rm_detach_ds (void *addr, l4re_ds_t *ds) L4_NOTHROW`
Detach and unmap a region and return affected dataspace in the current task.
- `int l4re_rm_detach_unmap (l4_addr_t addr, l4_cap_idx_t task) L4_NOTHROW`
Detach and unmap in specified task.
- `int l4re_rm_detach_ds_unmap (void *addr, l4re_ds_t *ds, l4_cap_idx_t task) L4_NOTHROW`
Detach and unmap in specified task.
- `int l4re_rm_find (l4_addr_t *addr, unsigned long *size, l4re_rm_offset_t *offset, l4re_rm_flags_t *flags, l4re_ds_t *m) L4_NOTHROW`
Find a region given an address and size.
- `int l4re_rm_get_info (l4_addr_t addr, char *name, unsigned int len, l4re_rm_offset_t *backing_offset) L4_NOTHROW`
Return auxiliary information of a region.
- `void l4re_rm_show_lists (void) L4_NOTHROW`
Dump region map internal data structures.
- `int l4re_rm_reserve_area_srv (l4_cap_idx_t rm, l4_addr_t *start, unsigned long size, l4re_rm_flags_t flags, unsigned char align) L4_NOTHROW`
- `int l4re_rm_free_area_srv (l4_cap_idx_t rm, l4_addr_t addr) L4_NOTHROW`
- `int l4re_rm_attach_srv (l4_cap_idx_t rm, void **start, unsigned long size, l4re_rm_flags_t flags, l4re_ds_t mem, l4re_rm_offset_t offs, unsigned char align) L4_NOTHROW`
- `int l4re_rm_attach_w_info_srv (l4_cap_idx_t rm, void **start, unsigned long size, l4re_rm_flags_t flags, l4re_ds_t mem, l4re_rm_offset_t offs, unsigned char align, char const *name, l4re_ds_offset_t backing_offset) L4_NOTHROW`
- `int l4re_rm_detach_srv (l4_cap_idx_t rm, l4_addr_t addr, l4re_ds_t *ds, l4_cap_idx_t task) L4_NOTHROW`
- `int l4re_rm_find_srv (l4_cap_idx_t rm, l4_addr_t *addr, unsigned long *size, l4re_rm_offset_t *offset, l4re_rm_flags_t *flags, l4re_ds_t *m) L4_NOTHROW`
- `int l4re_rm_get_info_srv (l4_cap_idx_t rm, l4_addr_t addr, char *name, unsigned int len, l4re_rm_offset_t *backing_offset) L4_NOTHROW`
- `void l4re_rm_show_lists_srv (l4_cap_idx_t rm) L4_NOTHROW`
Dump region map internal data structures.

16.305.1 Detailed Description

Region map interface, C interface.

Definition in file [rm.h](#).

16.306 rm.h

[Go to the documentation of this file.](#)

```

00001
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009 #pragma once
00010
00011 #include <l4/re/env.h>
00012 #include <l4/re/c/dataspace.h>
00013 #include <l4/sys/compiler.h>
00014
00015 L4_BEGIN_DECLS
00016
00017 enum l4re_rm_flags_values {

```

```

00031 L4RE_RM_F_R      = L4RE_DS_F_R,
00032 L4RE_RM_F_W      = L4RE_DS_F_W,
00033 L4RE_RM_F_X      = L4RE_DS_F_X,
00034 L4RE_RM_F_RX     = L4RE_DS_F_RX,
00035 L4RE_RM_F_RW     = L4RE_DS_F_RW,
00036 L4RE_RM_F_RWX    = L4RE_DS_F_RWX,
00037
00038 L4RE_RM_F_KERNEL  = 0x100,
00039 L4RE_RM_F_DETACH_FREE = 0x200,
00040 L4RE_RM_F_PAGER   = 0x400,
00041 L4RE_RM_F_RESERVED = 0x800,
00042
00043 L4RE_RM_CACHING_SHIFT = 4,
00044
00046 L4RE_RM_F_CACHING      = L4RE_DS_F_CACHING_MASK,
00047
00048 L4RE_RM_REGION_FLAGS = 0xffff,
00049
00051 L4RE_RM_F_CACHE_NORMAL = L4RE_DS_F_NORMAL,
00052
00054 L4RE_RM_F_CACHE_BUFFERED = L4RE_DS_F_BUFFERABLE,
00055
00057 L4RE_RM_F_CACHE_UNCACHED = L4RE_DS_F_UNCACHEABLE,
00058
00059 L4RE_RM_F_SEARCH_ADDR = 0x020000,
00060 L4RE_RM_F_IN_AREA     = 0x040000,
00061 L4RE_RM_F_EAGER_MAP   = 0x080000,
00062 L4RE_RM_F_NO_EAGER_MAP = 0x100000,
00063 L4RE_RM_F_ATTACH_FLAGS = 0x1f0000,
00064 };
00065
00066 typedef l4_uint32_t l4re_rm_flags_t;
00067 typedef l4_uint64_t l4re_rm_offset_t;
00068
00077 L4_CV L4_INLINE int
00078 l4re_rm_reserve_area(l4_addr_t *start, unsigned long size,
00079                     l4re_rm_flags_t flags, unsigned char align) L4_NOTHROW;
00080
00089 L4_CV L4_INLINE int
00090 l4re_rm_free_area(l4_addr_t addr) L4_NOTHROW;
00091
00142 L4_CV L4_INLINE int
00143 l4re_rm_attach(void **start, unsigned long size, l4re_rm_flags_t flags,
00144               l4re_ds_t mem, l4re_rm_offset_t offs,
00145               unsigned char align) L4_NOTHROW;
00146
00200 L4_CV L4_INLINE int
00201 l4re_rm_attach_w_info(void **start, unsigned long size, l4re_rm_flags_t flags,
00202                      l4re_ds_t mem, l4re_rm_offset_t offs,
00203                      unsigned char align,
00204                      char const *name,
00205                      l4re_ds_offset_t backing_offset) L4_NOTHROW;
00206
00207
00225 L4_CV L4_INLINE int
00226 l4re_rm_detach(void *addr) L4_NOTHROW;
00227
00247 L4_CV L4_INLINE int
00248 l4re_rm_detach_ds(void *addr, l4re_ds_t *ds) L4_NOTHROW;
00249
00261 L4_CV L4_INLINE int
00262 l4re_rm_detach_unmap(l4_addr_t addr, l4_cap_idx_t task) L4_NOTHROW;
00263
00278 L4_CV L4_INLINE int
00279 l4re_rm_detach_ds_unmap(void *addr, l4re_ds_t *ds,
00280                        l4_cap_idx_t task) L4_NOTHROW;
00281
00282
00289 L4_CV L4_INLINE int
00290 l4re_rm_find(l4_addr_t *addr, unsigned long *size,
00291             l4re_rm_offset_t *offset,
00292             l4re_rm_flags_t *flags, l4re_ds_t *m) L4_NOTHROW;
00293
00301 L4_CV L4_INLINE int
00302 l4re_rm_get_info(l4_addr_t addr,
00303                 char *name, unsigned int len,
00304                 l4re_rm_offset_t *backing_offset) L4_NOTHROW;
00305
00306
00313 L4_CV L4_INLINE void
00314 l4re_rm_show_lists(void) L4_NOTHROW;
00315
00316
00317 /*
00318  * Variants of functions that also take a capability of the region map
00319  * service.
00320  */

```

```

00321
00322
00327 L4_CV int
00328 l4re_rm_reserve_area_srv(l4_cap_idx_t rm, l4_addr_t *start, unsigned long size,
00329                          l4re_rm_flags_t flags, unsigned char align) L4_NOTHROW;
00330
00335 L4_CV int
00336 l4re_rm_free_area_srv(l4_cap_idx_t rm, l4_addr_t addr) L4_NOTHROW;
00337
00342 L4_CV int
00343 l4re_rm_attach_srv(l4_cap_idx_t rm, void **start, unsigned long size,
00344                   l4re_rm_flags_t flags, l4re_ds_t mem,
00345                   l4re_rm_offset_t offs,
00346                   unsigned char align) L4_NOTHROW;
00347
00352 L4_CV int
00353 l4re_rm_attach_w_info_srv(l4_cap_idx_t rm, void **start, unsigned long size,
00354                          l4re_rm_flags_t flags, l4re_ds_t mem,
00355                          l4re_rm_offset_t offs,
00356                          unsigned char align,
00357                          char const *name,
00358                          l4re_ds_offset_t backing_offset) L4_NOTHROW;
00359
00360
00365 L4_CV int
00366 l4re_rm_detach_srv(l4_cap_idx_t rm, l4_addr_t addr,
00367                   l4re_ds_t *ds, l4_cap_idx_t task) L4_NOTHROW;
00368
00369
00374 L4_CV int
00375 l4re_rm_find_srv(l4_cap_idx_t rm, l4_addr_t *addr,
00376                 unsigned long *size, l4re_rm_offset_t *offset,
00377                 l4re_rm_flags_t *flags, l4re_ds_t *m) L4_NOTHROW;
00378
00379
00384 L4_CV int
00385 l4re_rm_get_info_srv(l4_cap_idx_t rm, l4_addr_t addr,
00386                    char *name, unsigned int len,
00387                    l4re_rm_offset_t *backing_offset) L4_NOTHROW;
00388
00393 L4_CV void
00394 l4re_rm_show_lists_srv(l4_cap_idx_t rm) L4_NOTHROW;
00395
00396
00397 /***** Implementations *****/
00398
00399 L4_CV L4_INLINE int
00400 l4re_rm_reserve_area(l4_addr_t *start, unsigned long size,
00401                    l4re_rm_flags_t flags, unsigned char align) L4_NOTHROW
00402 {
00403     return l4re_rm_reserve_area_srv(l4re_global_env->rm, start, size,
00404                                     flags, align);
00405 }
00406
00407 L4_CV L4_INLINE int
00408 l4re_rm_free_area(l4_addr_t addr) L4_NOTHROW
00409 {
00410     return l4re_rm_free_area_srv(l4re_global_env->rm, addr);
00411 }
00412
00413 L4_CV L4_INLINE int
00414 l4re_rm_attach(void **start, unsigned long size, l4re_rm_flags_t flags,
00415               l4re_ds_t mem, l4re_rm_offset_t offs,
00416               unsigned char align) L4_NOTHROW
00417 {
00418     return l4re_rm_attach_srv(l4re_global_env->rm, start, size,
00419                              flags, mem, offs, align);
00420 }
00421
00422 L4_CV L4_INLINE int
00423 l4re_rm_attach_w_info(void **start, unsigned long size, l4re_rm_flags_t flags,
00424                     l4re_ds_t mem, l4re_rm_offset_t offs,
00425                     unsigned char align,
00426                     char const *name,
00427                     l4re_ds_offset_t backing_offset) L4_NOTHROW
00428 {
00429     return l4re_rm_attach_w_info_srv(l4re_global_env->rm, start, size,
00430                                      flags, mem, offs, align,
00431                                      name, backing_offset);
00432 }
00433
00434
00435 L4_CV L4_INLINE int
00436 l4re_rm_detach(void *addr) L4_NOTHROW
00437 {
00438     return l4re_rm_detach_srv(l4re_global_env->rm,
00439                             (l4_addr_t)addr, 0, L4_BASE_TASK_CAP);
00439

```

```

00440 }
00441
00442 L4_CV L4_INLINE int
00443 l4re_rm_detach_unmap(l4_addr_t addr, l4_cap_idx_t task) L4_NOTHROW
00444 {
00445     return l4re_rm_detach_srv(l4re_global_env->rm, addr, 0, task);
00446 }
00447
00448 L4_CV L4_INLINE int
00449 l4re_rm_detach_ds(void *addr, l4re_ds_t *ds) L4_NOTHROW
00450 {
00451     return l4re_rm_detach_srv(l4re_global_env->rm, (l4_addr_t)addr,
00452                               ds, L4_BASE_TASK_CAP);
00453 }
00454
00455 L4_CV L4_INLINE int
00456 l4re_rm_detach_ds_unmap(void *addr, l4re_ds_t *ds, l4_cap_idx_t task) L4_NOTHROW
00457 {
00458     return l4re_rm_detach_srv(l4re_global_env->rm, (l4_addr_t)addr,
00459                               ds, task);
00460 }
00461
00462 L4_CV L4_INLINE int
00463 l4re_rm_find(l4_addr_t *addr, unsigned long *size,
00464              l4re_rm_offset_t *offset,
00465              l4re_rm_flags_t *flags, l4re_ds_t *m) L4_NOTHROW
00466 {
00467     return l4re_rm_find_srv(l4re_global_env->rm, addr, size, offset, flags, m);
00468 }
00469
00470 L4_CV L4_INLINE void
00471 l4re_rm_show_lists(void) L4_NOTHROW
00472 {
00473     l4re_rm_show_lists_srv(l4re_global_env->rm);
00474 }
00475
00476
00477
00478 L4_CV L4_INLINE int
00479 l4re_rm_get_info(l4_addr_t addr, char *name, unsigned int len,
00480                  l4re_rm_offset_t *backing_offset) L4_NOTHROW
00481 {
00482     return l4re_rm_get_info_srv(l4re_global_env->rm, addr, name, len,
00483                                 backing_offset);
00484 }
00485
00486 L4_END_DECLS

```

16.307 l4/re/c/util/cap_alloc.h File Reference

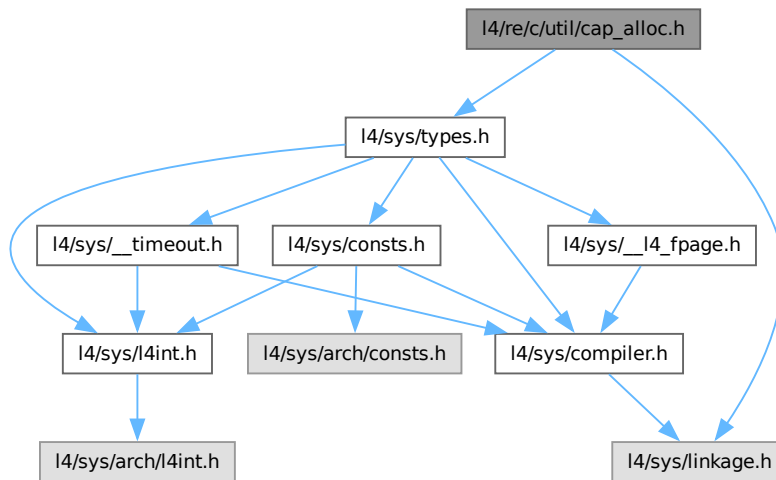
Capability allocator C interface.

```

#include <l4/sys/types.h>
#include <l4/sys/linkage.h>

```

Include dependency graph for `cap_alloc.h`:



Functions

- `L4_BEGIN_DECLS l4_cap_idx_t l4re_util_cap_alloc (void) L4_NOTHROW`
Get free capability index at capability allocator.
- `void l4re_util_cap_free (l4_cap_idx_t cap) L4_NOTHROW`
Return capability index to capability allocator.
- `void l4re_util_cap_free_um (l4_cap_idx_t cap) L4_NOTHROW`
Return capability index to capability allocator, and unmaps the object.
- `long l4re_util_cap_last (void) L4_NOTHROW`
Return last capability index the allocator can return.

16.307.1 Detailed Description

Capability allocator C interface.

Definition in file [cap_alloc.h](#).

16.308 cap_alloc.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #pragma once
00013
00019

```



```

00020 #include <l4/sys/types.h>
00021 #include <l4/sys/linkage.h>
00022
00023 L4_BEGIN_DECLS
00024
00029 L4_CV l4_cap_idx_t
00030 l4re_util_cap_alloc(void) L4_NOTHROW;
00031
00036 L4_CV void
00037 l4re_util_cap_free(l4_cap_idx_t cap) L4_NOTHROW;
00038
00044 L4_CV void
00045 l4re_util_cap_free_um(l4_cap_idx_t cap) L4_NOTHROW;
00046
00052 L4_CV long
00053 l4re_util_cap_last(void) L4_NOTHROW;
00054
00055 L4_END_DECLS

```

16.309 I4/re/c/util/kumem_alloc.h File Reference

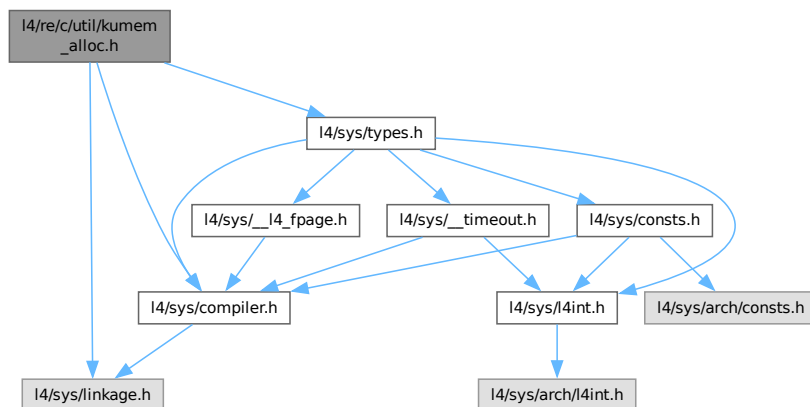
Kumem allocator utility C interface.

```
#include <l4/sys/compiler.h>
```

```
#include <l4/sys/types.h>
```

```
#include <l4/sys/linkage.h>
```

Include dependency graph for kumem_alloc.h:



Functions

- `L4_BEGIN_DECLS` `int l4re_util_kumem_alloc (l4_addr_t *mem, unsigned pages_order, l4_cap_idx_t task, l4_cap_idx_t rm) L4_NOTHROW`

Allocate state area.

16.309.1 Detailed Description

Kumem allocator utility C interface.

Definition in file [kumem_alloc.h](#).

16.309.2 Function Documentation

16.309.2.1 l4re_util_kumem_alloc()

```
L4_BEGIN_DECLS int l4re_util_kumem_alloc (
    l4_addr_t * mem,
    unsigned pages_order,
    l4_cap_idx_t task,
    l4_cap_idx_t rm)
```

Allocate state area.

Parameters

| | | |
|-----|--------------------|--|
| out | <i>mem</i> | Pointer to memory that has been allocated. |
| | <i>pages_order</i> | Size to allocate, in log2 pages. |
| | <i>task</i> | Task to use for allocation. |
| | <i>rm</i> | Region manager to use for allocation. |

Return values

| | |
|----|-----------------------|
| 0 | for success |
| <0 | error code on failure |

Note

The amount of kernel-user memory that can be allocated at once is limited by the used kernel implementation. The minimum allocatable amount is one page. A portable implementation should not depend on allocations greater than 16KiB to succeed.

Examples

[examples/sys/aliens/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

References [L4_END_DECLS](#), and [L4_NOTHROW](#).

16.310 kumem_alloc.h

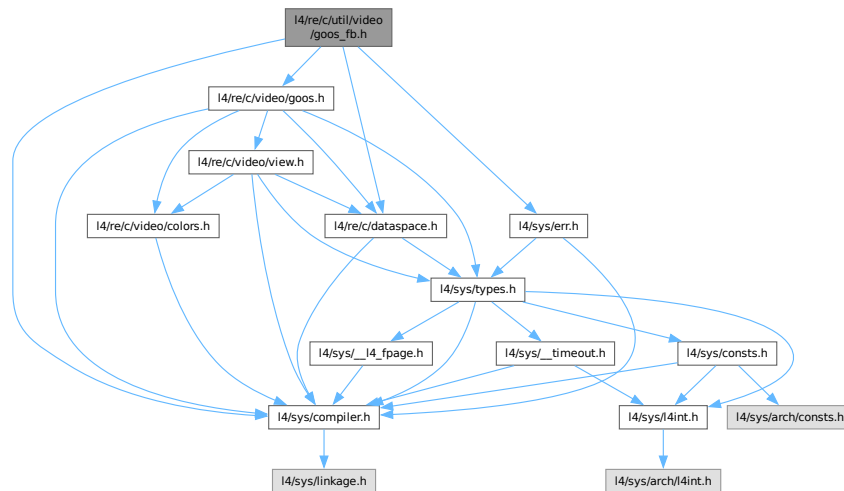
[Go to the documentation of this file.](#)

```
00001
00005 /*
00006  * (c) 2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #pragma once
00013
00019
00020 #include <l4/sys/compiler.h>
00021 #include <l4/sys/types.h>
00022 #include <l4/sys/linkage.h>
00023
00024 L4_BEGIN_DECLS
00025
00029 L4_CV int
00030 l4re_util_kumem_alloc(l4_addr_t *mem, unsigned pages_order,
00031                      l4_cap_idx_t task, l4_cap_idx_t rm) L4_NOTHROW;
00032
00033 L4_END_DECLS
```

16.311 l4/re/c/util/video/goos_fb.h File Reference

Framebuffer utility functionality.

```
#include <l4/sys/compiler.h>
#include <l4/re/c/video/goos.h>
#include <l4/sys/err.h>
#include <l4/re/c/dataspace.h>
Include dependency graph for goos_fb.h:
```



16.311.1 Detailed Description

Framebuffer utility functionality.

Definition in file [goos_fb.h](#).

16.312 goos_fb.h

[Go to the documentation of this file.](#)

```
00001
00005 /*
00006  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011 #pragma once
00012
00013 #include <l4/sys/compiler.h>
00014 #include <l4/re/c/video/goos.h>
00015 #include <l4/sys/err.h>
00016 #include <l4/re/c/dataspace.h>
00017
00018 L4_BEGIN_DECLS
00019
00020 typedef struct
00021 {
00022     unsigned long _obj_buf[6];
00023 } l4re_util_video_goos_fb_t;
```

```

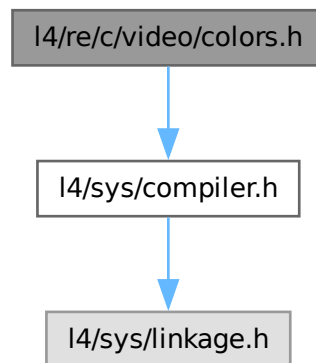
00024
00025 L4_CV int
00026 l4re_util_video_goos_fb_setup_name(l4re_util_video_goos_fb_t *goosfb,
00027                                     char const *name) L4_NOTHROW;
00028
00029 L4_CV void
00030 l4re_util_video_goos_fb_destroy(l4re_util_video_goos_fb_t *goosfb) L4_NOTHROW;
00031
00032 L4_CV int
00033 l4re_util_video_goos_fb_view_info(l4re_util_video_goos_fb_t *goosfb,
00034                                   l4re_video_view_info_t *info) L4_NOTHROW;
00035
00036 L4_CV void *
00037 l4re_util_video_goos_fb_attach_buffer(l4re_util_video_goos_fb_t *goosfb) L4_NOTHROW;
00038
00039 L4_CV int
00040 l4re_util_video_goos_fb_refresh(l4re_util_video_goos_fb_t *goosfb,
00041                                int x, int y, int w, int h) L4_NOTHROW;
00042
00043 L4_CV l4re_ds_t
00044 l4re_util_video_goos_fb_buffer(l4re_util_video_goos_fb_t *goosfb) L4_NOTHROW;
00045
00046 L4_CV l4_cap_idx_t
00047 l4re_util_video_goos_fb_goos(l4re_util_video_goos_fb_t *goosfb) L4_NOTHROW;
00048
00049 L4_END_DECLS

```

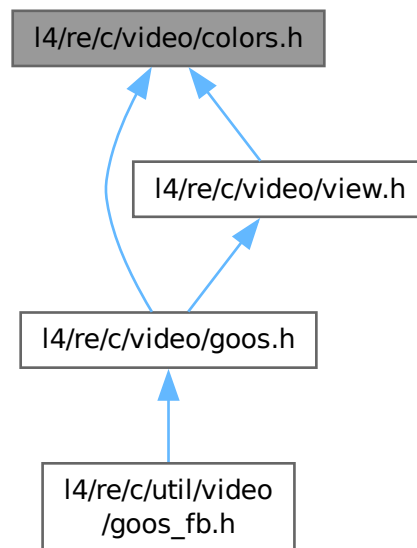
16.313 l4re/c/video/colors.h File Reference

#include <l4/sys/compiler.h>

Include dependency graph for colors.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct `l4re_video_color_component_t`
Color component structure.
- struct `l4re_video_pixel_info_t`
Pixel_info structure.

Typedefs

- typedef struct `l4re_video_color_component_t` **`l4re_video_color_component_t`**
Color component structure.
- typedef struct `l4re_video_pixel_info_t` **`l4re_video_pixel_info_t`**
Pixel_info structure.

16.313.1 Detailed Description

Note

The C interface of `L4Re::Video` does *NOT* reflect the full C++ interface on purpose. Use the C++ interface where possible.

Definition in file `colors.h`.

16.314 colors.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #pragma once
00013
00014 #include <l4/sys/compiler.h>
00015
00020 typedef struct l4re_video_color_component_t
00021 {
00022     unsigned char size;
00023     unsigned char shift;
00024 } __attribute__((packed)) l4re_video_color_component_t;
00025
00030 typedef struct l4re_video_pixel_info_t
00031 {
00032     l4re_video_color_component_t r, g, b, a;
00033     unsigned char bytes_per_pixel;
00034 } l4re_video_pixel_info_t;
00035
00036 L4_BEGIN_DECLS
00037
00038 L4_INLINE L4_CV int
00039 l4re_video_bits_per_pixel(l4re_video_pixel_info_t *p) L4_NOTHROW;
00040
00041 /* ***** */
00042 /* Implementations */
00043
00044 L4_INLINE L4_CV int
00045 l4re_video_bits_per_pixel(l4re_video_pixel_info_t *p) L4_NOTHROW
00046 {
00047     return p->r.size + p->b.size + p->g.size + p->a.size;
00048 }
00049
00050 L4_END_DECLS

```

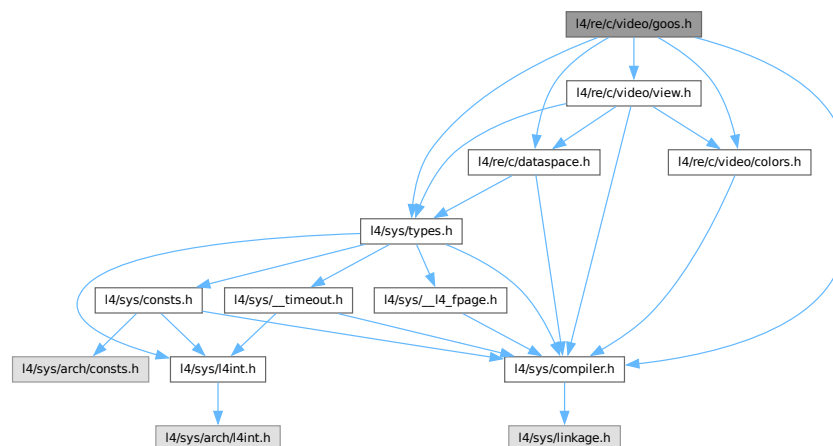
16.315 l4/re/c/video/goos.h File Reference

```

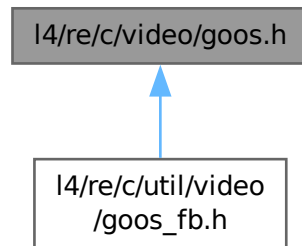
#include <l4/sys/compiler.h>
#include <l4/sys/types.h>
#include <l4/re/c/dataspace.h>
#include <l4/re/c/video/colors.h>
#include <l4/re/c/video/view.h>

```

Include dependency graph for goos.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [l4re_video_goos_info_t](#)
Goos information structure.

Typedefs

- typedef [l4_cap_idx_t](#) [l4re_video_goos_t](#)
Goos object type.

Enumerations

- enum [l4re_video_goos_info_flags_t](#) { [F_l4re_video_goos_auto_refresh](#) = 0x01 , [F_l4re_video_goos_pointer](#) = 0x02 , [F_l4re_video_goos_dynamic_views](#) = 0x04 , [F_l4re_video_goos_dynamic_buffers](#) = 0x08 }
- Flags of information on the goos.*

Functions

- [L4_BEGIN_DECLS](#) int [l4re_video_goos_info](#) ([l4re_video_goos_t](#) goos, [l4re_video_goos_info_t](#) *ginfo) [L4_NOTHROW](#)
Get information on a goos.
- int [l4re_video_goos_refresh](#) ([l4re_video_goos_t](#) goos, int x, int y, int w, int h) [L4_NOTHROW](#)
Flush a rectangle of pixels of the goos screen.
- int [l4re_video_goos_create_buffer](#) ([l4re_video_goos_t](#) goos, unsigned long size, [l4_cap_idx_t](#) buffer) [L4_NOTHROW](#)
Create a new buffer (memory buffer) for pixel data.
- int [l4re_video_goos_delete_buffer](#) ([l4re_video_goos_t](#) goos, unsigned idx) [L4_NOTHROW](#)
Delete a pixel buffer.
- int [l4re_video_goos_get_static_buffer](#) ([l4re_video_goos_t](#) goos, unsigned idx, [l4_cap_idx_t](#) buffer) [L4_NOTHROW](#)
Get the data-space capability of the static pixel buffer.
- int [l4re_video_goos_create_view](#) ([l4re_video_goos_t](#) goos, [l4re_video_view_t](#) *view) [L4_NOTHROW](#)
Create a new view (.
- int [l4re_video_goos_delete_view](#) ([l4re_video_goos_t](#) goos, [l4re_video_view_t](#) *view) [L4_NOTHROW](#)
Delete a view.
- int [l4re_video_goos_get_view](#) ([l4re_video_goos_t](#) goos, unsigned idx, [l4re_video_view_t](#) *view) [L4_NOTHROW](#)
Get a view for the given index.

16.315.1 Detailed Description

Note

The C interface of L4Re::Video does *NOT* reflect the full C++ interface on purpose. Use the C++ where possible.

Definition in file [goos.h](#).

16.316 goos.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #pragma once
00013
00014 #include <l4/sys/compiler.h>
00015 #include <l4/sys/types.h>
00016 #include <l4/re/c/dataspace.h>
00017 #include <l4/re/c/video/colors.h>
00018 #include <l4/re/c/video/view.h>
00019
00024
00029 enum l4re_video_goos_info_flags_t
00030 {
00031     F_l4re_video_goos_auto_refresh    = 0x01,
00032     F_l4re_video_goos_pointer        = 0x02,
00033     F_l4re_video_goos_dynamic_views  = 0x04,
00034     F_l4re_video_goos_dynamic_buffers = 0x08,
00035 };
00036
00041 typedef struct
00042 {
00043     unsigned long width;
00044     unsigned long height;
00045     unsigned flags;
00046     unsigned num_static_views;
00047     unsigned num_static_buffers;
00048     l4re_video_pixel_info_t pixel_info;
00049 } l4re_video_goos_info_t;
00050
00055 typedef l4_cap_idx_t l4re_video_goos_t;
00056
00057 L4_BEGIN_DECLS
00058
00070 L4_CV int
00071 l4re_video_goos_info(l4re_video_goos_t goos,
00072                     l4re_video_goos_info_t *ginfo) L4_NOTHROW;
00073
00083 L4_CV int
00084 l4re_video_goos_refresh(l4re_video_goos_t goos, int x, int y, int w,
00085                        int h) L4_NOTHROW;
00086
00098 L4_CV int
00099 l4re_video_goos_create_buffer(l4re_video_goos_t goos, unsigned long size,
00100                              l4_cap_idx_t buffer) L4_NOTHROW;
00101
00109 L4_CV int
00110 l4re_video_goos_delete_buffer(l4re_video_goos_t goos, unsigned idx) L4_NOTHROW;
00111
00122 L4_CV int
00123 l4re_video_goos_get_static_buffer(l4re_video_goos_t goos, unsigned idx,
00124                                   l4_cap_idx_t buffer) L4_NOTHROW;
00125
00132 L4_CV int
00133 l4re_video_goos_create_view(l4re_video_goos_t goos,
00134                             l4re_video_view_t *view) L4_NOTHROW;
00135
00143 L4_CV int
00144 l4re_video_goos_delete_view(l4re_video_goos_t goos,
00145                             l4re_video_view_t *view) L4_NOTHROW;

```



```

00146
00147
00159 L4_CV int
00160 l4re_video_goos_get_view(l4re_video_goos_t goos, unsigned idx,
00161                          l4re_video_view_t *view) L4_NOTHROW;
00162
00163 L4_END_DECLS

```

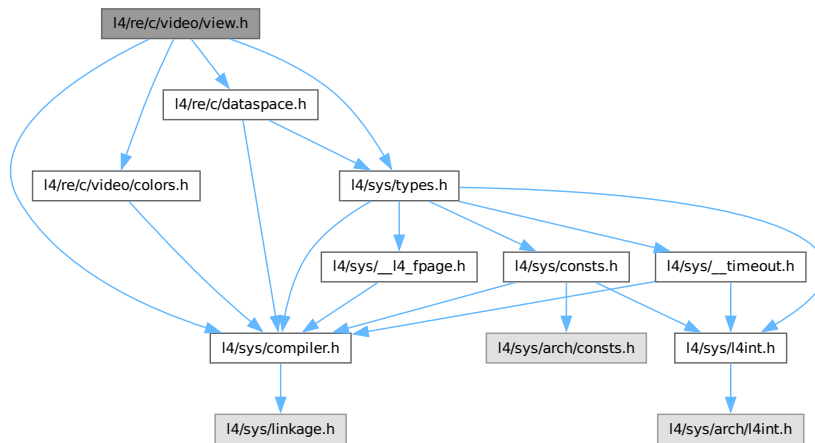
16.317 l4/re/c/video/view.h File Reference

```

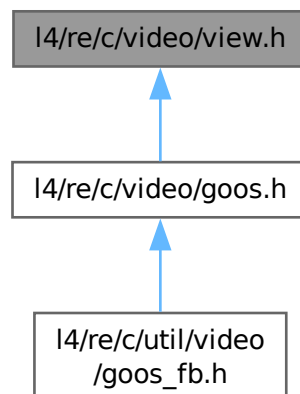
#include <l4/sys/compiler.h>
#include <l4/sys/types.h>
#include <l4/re/c/dataspace.h>
#include <l4/re/c/video/colors.h>

```

Include dependency graph for view.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [l4re_video_view_info_t](#)
View information structure.
- struct [l4re_video_view_t](#)
C representation of a goos view.

Typedefs

- typedef struct [l4re_video_view_info_t](#) **[l4re_video_view_info_t](#)**
View information structure.
- typedef struct [l4re_video_view_t](#) **[l4re_video_view_t](#)**
C representation of a goos view.

Enumerations

- enum [l4re_video_view_info_flags_t](#) {
[F_l4re_video_view_none](#) = 0x00 , [F_l4re_video_view_set_buffer](#) = 0x01 , [F_l4re_video_view_set_buffer_offset](#) = 0x02 , [F_l4re_video_view_set_bytes_per_line](#) = 0x04 ,
[F_l4re_video_view_set_pixel](#) = 0x08 , [F_l4re_video_view_set_position](#) = 0x10 , [F_l4re_video_view_dyn_allocated](#) = 0x20 , [F_l4re_video_view_set_background](#) = 0x40 ,
[F_l4re_video_view_set_flags](#) = 0x80 , **[F_l4re_video_view_fully_dynamic](#)** , [F_l4re_video_view_above](#) = 0x01000 , [F_l4re_video_view_flags_mask](#) = 0xff000 }
Flags of information on a view.

Functions

- [L4_BEGIN_DECLS](#) int [l4re_video_view_refresh](#) ([l4re_video_view_t](#) *view, int x, int y, int w, int h) [L4_NOTHROW](#)
Flush the given rectangle of pixels of the given view.
- int [l4re_video_view_get_info](#) ([l4re_video_view_t](#) *view, [l4re_video_view_info_t](#) *info) [L4_NOTHROW](#)
Retrieve information about the given view.
- int [l4re_video_view_set_info](#) ([l4re_video_view_t](#) *view, [l4re_video_view_info_t](#) *info) [L4_NOTHROW](#)
Set properties of the view.
- int [l4re_video_view_set_viewport](#) ([l4re_video_view_t](#) *view, int x, int y, int w, int h, unsigned long bofs) [L4_NOTHROW](#)
Set the viewport parameters of a view.
- int [l4re_video_view_stack](#) ([l4re_video_view_t](#) *view, [l4re_video_view_t](#) *pivot, int behind) [L4_NOTHROW](#)
Change the stacking order in the stack of visible views.

16.317.1 Detailed Description

Note

The C interface of `L4Re::Video` does *NOT* reflect the full C++ interface on purpose. Use the C++ where possible.

Definition in file [view.h](#).

16.318 view.h

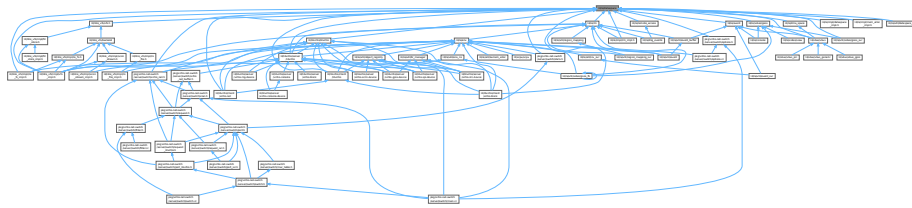
[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #pragma once
00013
00014 #include <l4/sys/compiler.h>
00015 #include <l4/sys/types.h>
00016 #include <l4/re/c/dataspace.h>
00017 #include <l4/re/c/video/colors.h>
00018
00023 enum l4re_video_view_info_flags_t
00024 {
00025     F_l4re_video_view_none           = 0x00,
00026     F_l4re_video_view_set_buffer     = 0x01,
00027     F_l4re_video_view_set_buffer_offset = 0x02,
00028     F_l4re_video_view_set_bytes_per_line = 0x04,
00029     F_l4re_video_view_set_pixel      = 0x08,
00030     F_l4re_video_view_set_position   = 0x10,
00031     F_l4re_video_view_dyn_allocated  = 0x20,
00032     F_l4re_video_view_set_background = 0x40,
00033     F_l4re_video_view_set_flags      = 0x80,
00034     F_l4re_video_view_fully_dynamic  = F_l4re_video_view_set_buffer
00035                                         | F_l4re_video_view_set_buffer_offset
00036                                         | F_l4re_video_view_set_bytes_per_line
00037                                         | F_l4re_video_view_set_pixel
00038                                         | F_l4re_video_view_set_position
00039                                         | F_l4re_video_view_dyn_allocated,
00040
00041     F_l4re_video_view_above          = 0x01000,
00042     F_l4re_video_view_flags_mask    = 0xff000,
00043 };
00044
00049 typedef struct l4re_video_view_info_t
00050 {
00051     unsigned          flags;
00052     unsigned          view_index;
00053     unsigned long     xpos, ypos, width, height;
00054     unsigned long     buffer_offset;
00055     unsigned long     bytes_per_line;
00056     l4re_video_pixel_info_t pixel_info;
00057     unsigned          buffer_index;
00058 } l4re_video_view_info_t;
00059
00060
00068 typedef struct l4re_video_view_t
00069 {
00070     l4_cap_idx_t goos;
00071     unsigned idx;
00072 } l4re_video_view_t;
00073
00074
00075 L4_BEGIN_DECLS
00076
00087 L4_CV int
00088 l4re_video_view_refresh(l4re_video_view_t *view, int x, int y, int w,
00089                        int h) L4_NOTHROW;
00090
00091
00098 L4_CV int
00099 l4re_video_view_get_info(l4re_video_view_t *view,
00100                          l4re_video_view_info_t *info) L4_NOTHROW;
00101
00113 L4_CV int
00114 l4re_video_view_set_info(l4re_video_view_t *view,
00115                           l4re_video_view_info_t *info) L4_NOTHROW;
00116
00133 L4_CV int
00134 l4re_video_view_set_viewport(l4re_video_view_t *view, int x, int y, int w,
00135                               int h, unsigned long bofs) L4_NOTHROW;
00136
00147 L4_CV int
00148 l4re_video_view_stack(l4re_video_view_t *view, l4re_video_view_t *pivot,
00149                        int behind) L4_NOTHROW;
00150
00151 L4_END_DECLS
00152

```


This graph shows which files directly or indirectly include this file:



Data Structures

- class [L4Re::Dataspace](#)
Interface for memory-like objects.
- struct [L4Re::Dataspace::F](#)
Dataspace flags definitions.
- struct [L4Re::Dataspace::Stats](#)
Information about the dataspace.

Namespaces

- namespace [L4Re](#)
L4Re C++ Interfaces.

16.320.1 Detailed Description

Dataspace interface.

Definition in file [dataspace](#).

16.321 dataspace

[Go to the documentation of this file.](#)

```
00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *                Alexander Warg <warg@os.inf.tu-dresden.de>,
00010  *                Björn Döbel <doebel@os.inf.tu-dresden.de>,
00011  *                Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00012  *                economic rights: Technische Universität Dresden (Germany)
00013  *
00014  * License: see LICENSE.spdx (in this directory or the directories above)
00015  */
00016
00017 #pragma once
00018
00019 #include <l4/bid_config.h>
00020 #include <l4/sys/types.h>
00021 #include <l4/sys/l4int.h>
00022 #include <l4/sys/capability>
00023 #include <l4/re/protocols.h>
00024 #include <l4/sys/cxx/ipc_types>
00025 #include <l4/sys/cxx/ipc_iface>
00026 #include <l4/sys/cxx/types>
00027
00028 namespace L4Re
```

```

00029 {
00030
00031     // MISSING:
00032     // * size support in map, mapped size in reply
00033
00050 class L4_EXPORT Dataspace :
00051     public L4::Kobject_t<Dataspace, L4::Kobject, L4RE_PROTO_DATASPACE,
00052         L4::Type_info::Demand_t<1> >
00053 {
00054 public:
00055
00057     struct F
00058     {
00059         enum
00060         {
00061             Caching_shift = 4,
00062         };
00063
00070         enum Flags
00071         {
00073             R = L4_FPAGE_RO,
00075             Ro = L4_FPAGE_RO,
00077             RW = L4_FPAGE_RW,
00079             W = L4_FPAGE_W,
00081             X = L4_FPAGE_X,
00083             RX = L4_FPAGE_RX,
00085             RWX = L4_FPAGE_RWX,
00087             Rights_mask = 0x0f,
00088
00091             Normal = 0x00,
00093             Cacheable = Normal,
00095             Bufferable = 0x10,
00097             Uncacheable = 0x20,
00099             Caching_mask = 0x30,
00100         };
00101
00102         friend void enum_bitops_enable(Flags);
00103     };
00104
00105     struct Flags : L4::Types::Flags_ops_t<Flags>
00106     {
00107         unsigned long raw;
00108
00109         Flags() = default;
00110         explicit constexpr Flags(unsigned long f) : raw(f) {}
00111         constexpr Flags(F::Flags f) : raw(f) {}
00112
00113         constexpr bool r() const { return raw & L4_FPAGE_RO; }
00114         constexpr bool w() const { return raw & L4_FPAGE_W; }
00115         constexpr bool x() const { return raw & L4_FPAGE_X; }
00116
00117         constexpr unsigned long fpage_rights() const
00118         { return raw & 0xf; }
00119     };
00120
00121     typedef l4_uint64_t Size;
00122     typedef l4_uint64_t Offset;
00123     typedef l4_uint64_t Map_addr;
00124
00128     struct Stats
00129     {
00130         Size size;
00131         Flags flags;
00132     };
00133
00158     l4_ret_t map(Offset offset, Flags flags, Map_addr local_addr,
00159                 Map_addr min_addr, Map_addr max_addr,
00160                 L4::Cap<L4::Task> dst = L4::Cap<L4::Task>::Invalid) const noexcept;
00161
00189     l4_ret_t map_region(Offset offset, Flags flags,
00190                        Map_addr min_addr, Map_addr max_addr,
00191                        L4::Cap<L4::Task> dst = L4::Cap<L4::Task>::Invalid) const noexcept;
00192
00210     L4_RPC(l4_ret_t, clear, (Offset offset, Size size));
00211
00231     L4_RPC(l4_ret_t, allocate, (Offset offset, Size size));
00232
00251     L4_RPC(l4_ret_t, copy_in, (Offset dst_offs, L4::Ipc::Cap<Dataspace> src,
00252                             Offset src_offs, Size size));
00253
00259     Size size() const noexcept;
00260
00269     Flags flags() const noexcept;
00270
00279     L4_RPC(l4_ret_t, info, (Stats *stats));
00280
00281     L4_RPC_NF(l4_ret_t, map, (Offset offset, Map_addr spot,

```

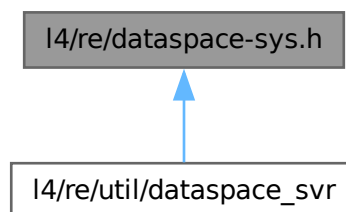
```

00282             Flags flags, L4::Ipc::Rcv_fpage r,
00283             L4::Ipc::Snd_fpage &fp));
00284
00304 #ifdef CONFIG_MMU
00305     L4_RPC_NF(long, map_info, (l4_addr_t *start_addr, l4_addr_t *end_addr));
00306     inline long map_info([[maybe_unused]] l4_addr_t *start_addr,
00307                          [[maybe_unused]] l4_addr_t *end_addr)
00308     { return 0; }
00309 #else
00310     L4_RPC(long, map_info, (l4_addr_t *start_addr, l4_addr_t *end_addr));
00311 #endif
00312
00313 private:
00314
00315     l4_ret_t __map(Offset offset, unsigned char *order, Flags flags,
00316                   Map_addr local_addr, L4::Cap<L4::Task> dst) const noexcept;
00317
00318 public:
00319     typedef L4::Typeid::Rpc<map_t, clear_t, info_t, copy_in_t,
00320                           allocate_t, map_info_t> Rpc;
00321 };
00322
00323 }
```

16.322 l4/re/dataspace-sys.h File Reference

Dataspace protocol definition.

This graph shows which files directly or indirectly include this file:



Namespaces

- namespace [L4Re](#)
L4Re C++ Interfaces.

Enumerations

- enum [L4Re::Dataspace_::Opcodes](#)
Data-space communication-protocol opcodes.

16.322.1 Detailed Description

Dataspace protocol definition.

Definition in file [dataspace-sys.h](#).

16.323 dataspace-sys.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #pragma once
00013
00014 namespace L4Re
00015 {
00016     namespace Dataspace_
00017     {
00023         enum Opcodes { Map, Clear, Stats, Copy, Take, Release, Allocate };
00024     };
00025 };
00026

```

16.324 dbg_events

```

00001 // vim:ft=cpp
00002
00003 #pragma once
00004
00005 #include <l4/sys/cxx/ipc_epiface>
00006 #include <l4/sys/utcb.h>
00007 #include <l4/re/remote_access>
00008 #include <l4/re/rm>
00009
00010 namespace L4Re {
00011     struct Dbg_events : L4::Kobject_t<Dbg_events, L4::Kobject, 0,
00012                                     L4::Type_info::Demand_t<2> >
00013     {
00014         L4_INLINE_RPC(int, request_backtrace, (l4_exc_regs_t regs,
00015                                             L4::Ipc::Cap<L4Re::Remote_access> raif,
00016                                             L4::Ipc::Cap<L4Re::Rm> rmif));
00017
00018         typedef L4::Typeid::Rpc<request_backtrace_t> Rpc;
00019     };
00020 } // L4Re

```

16.325 l4/re/dma_space File Reference

```

#include <l4/sys/types.h>
#include <l4/sys/l4int.h>
#include <l4/sys/capability>
#include <l4/re/dataspace>
#include <l4/re/protocols.h>
#include <l4/sys/cxx/types>
#include <l4/sys/cxx/ipc_types>
#include <l4/sys/cxx/ipc_iface>

```


16.326 dma_space

[Go to the documentation of this file.](#)

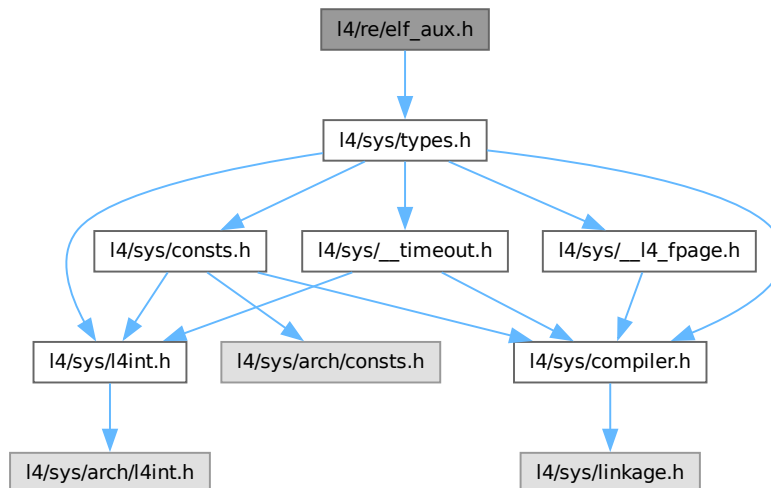
```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00006 /*
00007  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011
00012 #pragma once
00013
00014 #include <l4/sys/types.h>
00015 #include <l4/sys/l4int.h>
00016 #include <l4/sys/capability>
00017 #include <l4/re/dataspace>
00018 #include <l4/re/protocols.h>
00019 #include <l4/sys/cxx/types>
00020 #include <l4/sys/cxx/ipc_types>
00021 #include <l4/sys/cxx/ipc_iface>
00022
00023 namespace L4Re
00024 {
00025
00052 class Dma_space :
00053     public L4::Kobject_0t< Dma_space,
00054                          L4RE_PROTO_DMA_SPACE,
00055                          L4::Type_info::Demand_t<1> >
00056 {
00057 public:
00059     typedef l4_uint64_t Dma_addr;
00060
00064     enum Direction
00065     {
00066         Bidirectional,
00067         To_device,
00068         From_device,
00069         None
00070     };
00071
00076     enum Attribute
00077     {
00089         No_sync
00090     };
00091
00097     typedef L4::Types::Flags<Attribute> Attributes;
00098
00104     enum Space_attrib
00105     {
00112         Coherent,
00113
00118         Phys_space
00119     };
00120
00122     typedef L4::Types::Flags<Space_attrib> Space_attribs;
00123
00159     L4_INLINE_RPC(
00160         l4_ret_t, map, (L4::Ipc::Cap<L4Re::Dataspace> src,
00161                        L4Re::Dataspace::Offset offset,
00162                        L4::Ipc::In_out<l4_size_t *> size,
00163                        Attributes attrs, Direction dir,
00164                        Dma_addr *dma_addr));
00165
00176     L4_INLINE_RPC(
00177         l4_ret_t, unmap, (Dma_addr dma_addr,
00178                          l4_size_t size, Attributes attrs, Direction dir));
00179
00202     L4_INLINE_RPC(
00203         l4_ret_t, associate, (L4::Ipc::Opt<L4::Ipc::Cap<L4::Task> > dma_task,
00204                              Space_attribs attr),
00205         L4::Ipc::Call_t<L4_CAP_FPAGE_RW>);
00206
00218     L4_INLINE_RPC(
00219         l4_ret_t, disassociate, (),
00220         L4::Ipc::Call_t<L4_CAP_FPAGE_RW>);
00221
00222     typedef L4::Typeid::Rpcs<map_t, unmap_t, associate_t, disassociate_t> Rpcs;
00223 };
00224
00225 }
```

16.327 l4/re/elf_aux.h File Reference

Auxiliary information for binaries.

```
#include <l4/sys/types.h>
Include dependency graph for elf_aux.h:
```



Data Structures

- struct `l4re_elf_aux_t`
Generic header for each auxiliary vector element.
- struct `l4re_elf_aux_vma_t`
Auxiliary vector element for a reserved virtual memory area.
- struct `l4re_elf_aux_mword_t`
Auxiliary vector element for a single unsigned data word.

Macros

- `#define L4RE_ELF_AUX_ELEM const __attribute__((used, section(".ro4re_elf_aux"), aligned(sizeof(l4_umword_t))))`
Define an auxiliary vector element.
- `#define L4RE_ELF_AUX_ELEM_T(type, id, tag, val...)`
Define an auxiliary vector element.

Typedefs

- typedef struct `l4re_elf_aux_t` `l4re_elf_aux_t`
Generic header for each auxiliary vector element.
- typedef struct `l4re_elf_aux_vma_t` `l4re_elf_aux_vma_t`
Auxiliary vector element for a reserved virtual memory area.
- typedef struct `l4re_elf_aux_mword_t` `l4re_elf_aux_mword_t`
Auxiliary vector element for a single unsigned data word.

Enumerations

- enum {
[L4RE_ELF_AUX_T_NONE](#) = 0 , [L4RE_ELF_AUX_T_VMA](#) , [L4RE_ELF_AUX_T_STACK_SIZE](#) ,
[L4RE_ELF_AUX_T_STACK_ADDR](#) ,
[L4RE_ELF_AUX_T_KIP_ADDR](#) , [L4RE_ELF_AUX_T_EX_REGS_FLAGS](#) }

16.327.1 Detailed Description

Auxiliary information for binaries.

Definition in file [elf_aux.h](#).

16.328 elf_aux.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00007  *      economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011 #pragma once
00012
00013 #include <l4/sys/types.h>
00014
00015
00028
00041 #define L4RE_ELF_AUX_ELEM const __attribute__((used, section(".rol4re_elf_aux"),
    aligned(sizeof(l4_umword_t))))
00042
00056 #define L4RE_ELF_AUX_ELEM_T(type, id, tag, val...) \
    static L4RE_ELF_AUX_ELEM type id = {tag, sizeof(type), val}
00057
00058
00059 enum
00060 {
00064     L4RE\_ELF\_AUX\_T\_NONE = 0,
00065
00069     L4RE\_ELF\_AUX\_T\_VMA,
00070
00075     L4RE\_ELF\_AUX\_T\_STACK\_SIZE,
00076
00081     L4RE\_ELF\_AUX\_T\_STACK\_ADDR,
00082
00087     L4RE\_ELF\_AUX\_T\_KIP\_ADDR,
00088
00092     L4RE\_ELF\_AUX\_T\_EX\_REGS\_FLAGS,
00093 };
00094
00098 typedef struct l4re_elf_aux_t
00099 {
00100     l4_umword_t type;
00101     l4_umword_t length;
00102 } l4re_elf_aux_t;
00103
00107 typedef struct l4re_elf_aux_vma_t
00108 {
00109     l4_umword_t type;
00110     l4_umword_t length;
00111     l4_umword_t start;
00112     l4_umword_t end;
00113 } l4re_elf_aux_vma_t;
00114
00118 typedef struct l4re_elf_aux_mword_t
00119 {
00120     l4_umword_t type;
00121     l4_umword_t length;
00122     l4_umword_t value;
00123 } l4re_elf_aux_mword_t;
00124

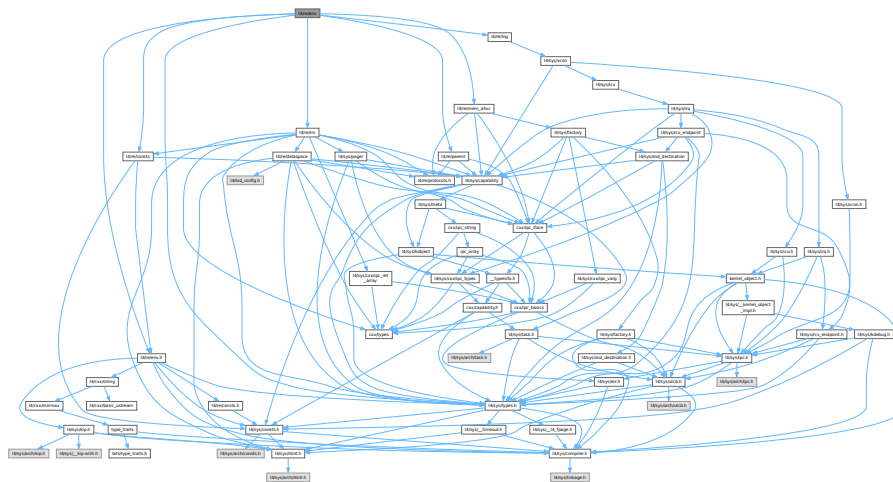
```

16.329 l4/re/env File Reference

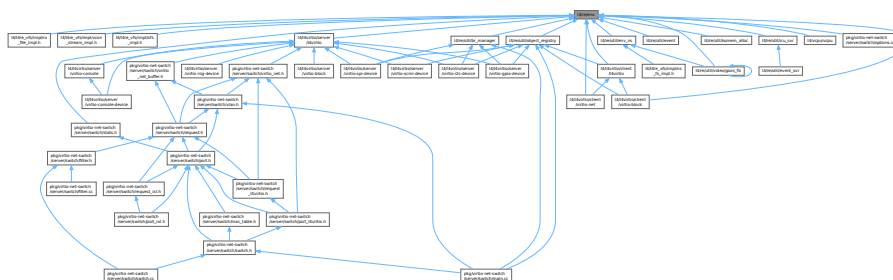
Environment interface.

```
#include <l4/sys/types.h>
#include <l4/re/rm>
#include <l4/re/parent>
#include <l4/re/mem_alloc>
#include <l4/re/log>
#include <l4/re/consts>
#include <l4/re/env.h>
```

Include dependency graph for env:



This graph shows which files directly or indirectly include this file:



Data Structures

- class [L4Re::Env](#)

C++ interface of the initial environment that is provided to an [L4](#) task.

Namespaces

- namespace [L4](#)
L4 low-level kernel interface.
- namespace [L4Re](#)
L4Re C++ Interfaces.

16.329.1 Detailed Description

Environment interface.

Definition in file [env](#).

16.330 env

[Go to the documentation of this file.](#)

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00003 /*
00008 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009 *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00010 *      Björn Döbel <doebel@os.inf.tu-dresden.de>
00011 *      economic rights: Technische Universität Dresden (Germany)
00012 *
00013 * License: see LICENSE.spdx (in this directory or the directories above)
00014 */
00015 #pragma once
00016
00017 #include <l4/sys/types.h>
00018
00019 #include <l4/re/rm>
00020 #include <l4/re/parent>
00021 #include <l4/re/mem_alloc>
00022 #include <l4/re/log>
00023 #include <l4/re/consts>
00024
00025 #include <l4/re/env.h>
00026
00027 namespace L4 {
00028     class Scheduler;
00029 }
00030
00031 namespace L4Re
00032 {
00033     class Itas;
00034     struct Dbg_events;
00035
00036     class L4_EXPORT Env
00037     {
00038     private:
00039         l4re_env_t _env;
00040     public:
00041
00042         typedef l4re_env_cap_entry_t Cap_entry;
00043
00044         static Env const *env() noexcept
00045         { return reinterpret_cast<Env*>(l4re_global_env); }
00046
00047         L4::Cap<Parent> parent() const noexcept
00048         { return L4::Cap<Parent>(_env.parent); }
00049         L4::Cap<Mem_alloc> mem_alloc() const noexcept
00050         { return L4::Cap<Mem_alloc>(_env.mem_alloc); }
00051         L4::Cap<L4::Factory> user_factory() const noexcept
00052         { return L4::Cap<L4::Factory>(_env.mem_alloc); }
00053         L4::Cap<Rm> rm() const noexcept
00054         { return L4::Cap<Rm>(_env.rm); }
00055         L4::Cap<Log> log() const noexcept
00056         { return L4::Cap<Log>(_env.log); }
00057         L4::Cap<L4::Thread> main_thread() const noexcept
00058         { return L4::Cap<L4::Thread>(_env.main_thread); }
00059         L4::Cap<L4::Task> task() const noexcept
00060         { return L4::Cap<L4::Task>(L4RE_THIS_TASK_CAP); }
00061         L4::Cap<L4::Factory> factory() const noexcept
00062         { return L4::Cap<L4::Factory>(_env.factory); }
00063         l4_cap_idx_t first_free_cap() const noexcept
00064         { return _env.first_free_cap; }
00065         l4_umword_t first_free_reply_cap() const noexcept
00066         { return _env.first_free_reply_cap; }
00067         l4_fpage_t utcb_area() const noexcept
00068         { return _env.utcb_area; }
00069         l4_addr_t first_free_utcb() const noexcept
00070         { return _env.first_free_utcb; }
00071
00072         Cap_entry const *initial_caps() const noexcept
00073         { return _env.caps; }
00074     };

```

```

00188
00197     Cap_entry const *get(char const *name, unsigned l) const noexcept
00198     { return l4re_env_get_cap_l(name, l, &_env); }
00199
00208     template< typename T >
00209     L4::Cap<T> get_cap(char const *name, unsigned l) const noexcept
00210     {
00211         if (Cap_entry const *e = get(name, l))
00212             return L4::Cap<T>(e->cap);
00213
00214         return L4::Cap<T>(-L4_ENOENT);
00215     }
00216
00223     template< typename T >
00224     L4::Cap<T> get_cap(char const *name) const noexcept
00225     { return get_cap<T>(name, __builtin_strlen(name)); }
00226
00231     void parent(L4::Cap<Parent> const &c) noexcept
00232     { _env.parent = c.cap(); }
00237     void mem_alloc(L4::Cap<Mem_alloc> const &c) noexcept
00238     { _env.mem_alloc = c.cap(); }
00243     void rm(L4::Cap<Rm> const &c) noexcept
00244     { _env.rm = c.cap(); }
00249     void log(L4::Cap<Log> const &c) noexcept
00250     { _env.log = c.cap(); }
00255     void main_thread(L4::Cap<L4::Thread> const &c) noexcept
00256     { _env.main_thread = c.cap(); }
00261     void factory(L4::Cap<L4::Factory> const &c) noexcept
00262     { _env.factory = c.cap(); }
00267     void first_free_cap(l4_cap_idx_t c) noexcept
00268     { _env.first_free_cap = c; }
00273     void first_free_reply_cap(l4_umword_t c) noexcept
00274     { _env.first_free_reply_cap = c; }
00279     void utcb_area(l4_fpage_t utcbs) noexcept
00280     { _env.utcb_area = utcbs; }
00285     void first_free_utcb(l4_addr_t u) noexcept
00286     { _env.first_free_utcb = u; }
00287
00293     L4::Cap<L4::Scheduler> scheduler() const noexcept
00294     { return L4::Cap<L4::Scheduler>(_env.scheduler); }
00295
00300     void scheduler(L4::Cap<L4::Scheduler> const &c) noexcept
00301     { _env.scheduler = c.cap(); }
00302
00312     L4::Cap<Itas> itas() const noexcept
00313     { return L4::Cap<Itas>(_env.itas); }
00314
00319     void itas(L4::Cap<Itas> const &c) noexcept
00320     { _env.itas = c.cap(); }
00321
00328     L4::Cap<Dbg_events> dbg_events() const noexcept
00329     { return L4::Cap<Dbg_events>(_env.dbg_events); }
00330
00338     void dbg_events(L4::Cap<Dbg_events> const &dbg_events) noexcept
00339     { _env.dbg_events = dbg_events.cap(); }
00340
00345     void initial_caps(Cap_entry *first) noexcept
00346     { _env.caps = first; }
00347 };
00348 };

```

16.331 l4/re/env.h File Reference

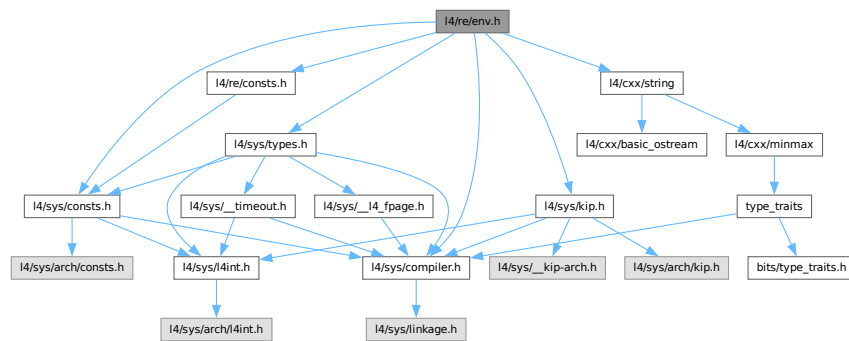
Environment interface.

```

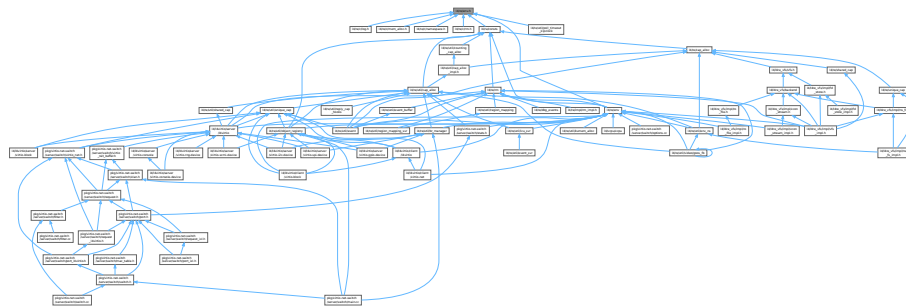
#include <l4/sys/consts.h>
#include <l4/sys/types.h>
#include <l4/sys/kip.h>
#include <l4/sys/compiler.h>
#include <l4/re/consts.h>
#include <l4/cxx/string>

```

Include dependency graph for env.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct `I4re_env_cap_entry_t`
Entry in the *L4Re* environment array for the named initial objects.
- struct `I4re_env_t`
Initial environment data structure.

Typedefs

- typedef struct `I4re_env_cap_entry_t` `I4re_env_cap_entry_t`
Entry in the *L4Re* environment array for the named initial objects.
- typedef struct `I4re_env_t` `I4re_env_t`
Initial environment data structure.

Functions

- `I4re_env_t * I4re_env` (void) `L4_NOTHROW`
Get *L4Re* initial environment.
- `I4_kernel_info_t` const * `I4re_kip` (void) `L4_NOTHROW`
Get Kernel Info Page.
- `I4_cap_idx_t` `I4re_env_get_cap` (char const *name) `L4_NOTHROW`

Get the capability selector for the object named name.

- [l4_cap_idx_t l4re_env_get_cap_e](#) (char const *name, [l4re_env_t](#) const *e) [L4_NOTHROW](#)

Get the capability selector for the object named name.

- [l4re_env_cap_entry_t](#) const * [l4re_env_get_cap_l](#) (char const *name, unsigned l, [l4re_env_t](#) const *e) [L4_NOTHROW](#)

Get the full [l4re_env_cap_entry_t](#) for the object named name.

16.331.1 Detailed Description

Environment interface.

Definition in file [env.h](#).

16.331.2 Typedef Documentation

16.331.2.1 l4re_env_t

```
typedef struct l4re_env_t l4re_env_t
```

Initial environment data structure.

See also

[Initial environment](#)

16.332 env.h

[Go to the documentation of this file.](#)

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #pragma once
00013
00014 #include <l4/sys/consts.h>
00015 #include <l4/sys/types.h>
00016 #include <l4/sys/kip.h>
00017 #include <l4/sys/compiler.h>
00018
00019 #include <l4/re/consts.h>
00020
00021 #ifdef __cplusplus
00022 #include <l4/cxx/string>
00023 #endif
00024
00038
00043 typedef struct l4re_env_cap_entry_t
00044 {
00048     l4_cap_idx_t cap;
00049
00054     l4_umword_t flags;
00055
00059     char name[16];
00060 #ifdef __cplusplus
00061
00065     l4re_env_cap_entry_t() L4\_NOTHROW : cap(L4\_INVALIDID\_CAP), flags(~0UL) {}
00066
00074     l4re_env_cap_entry_t(char const *n, l4_cap_idx_t c, l4_umword_t f = 0) L4\_NOTHROW
```

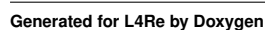
```

00075 : cap(c), flags(f)
00076 {
00077     for (unsigned i = 0; n && i < sizeof(name); ++i, ++n)
00078     {
00079         name[i] = *n;
00080         if (!*n)
00081             break;
00082     }
00083 }
00084
00085 static bool is_valid_name(char const *n) L4_NOTHROW
00086 {
00087     for (unsigned i = 0; *n; ++i, ++n)
00088         if (i > sizeof(name))
00089             return false;
00090
00091     return true;
00092 }
00093
00094 cxx::String get_name() const noexcept
00095 {
00096     unsigned len = 0;
00097     while (len < sizeof(name) && name[len] != '\0')
00098         ++len;
00099
00100     return cxx::String(name, len);
00101 }
00102 #endif
00103 } l4re_env_cap_entry_t;
00104
00105
00111 typedef struct l4re_env_t
00112 {
00113     l4_cap_idx_t parent;
00114     l4_cap_idx_t rm;
00115     l4_cap_idx_t mem_alloc;
00116     l4_cap_idx_t log;
00117     l4_cap_idx_t main_thread;
00118     l4_cap_idx_t factory;
00119     l4_cap_idx_t scheduler;
00120     l4_cap_idx_t itas;
00121     l4_cap_idx_t dbg_events;
00122     l4_cap_idx_t first_free_cap;
00123     l4_umword_t first_free_reply_cap;
00124     l4_fpage_t utcb_area;
00125     l4_addr_t first_free_utcb;
00131     l4re_env_cap_entry_t *caps;
00132 } l4re_env_t;
00133
00139 extern l4re_env_t *l4re_global_env;
00140
00141
00147 L4_INLINE l4re_env_t *l4re_env(void) L4_NOTHROW;
00148
00149 /*
00150  * FIXME: this seems to be at the wrong place here
00151  */
00157 L4_INLINE l4_kernel_info_t const *l4re_kip(void) L4_NOTHROW;
00158
00159
00167 L4_INLINE l4_cap_idx_t
00168 l4re_env_get_cap(char const *name) L4_NOTHROW;
00169
00178 L4_INLINE l4_cap_idx_t
00179 l4re_env_get_cap_e(char const *name, l4re_env_t const *e) L4_NOTHROW;
00180
00191 L4_INLINE l4re_env_cap_entry_t const *
00192 l4re_env_get_cap_l(char const *name, unsigned l, l4re_env_t const *e) L4_NOTHROW;
00193
00194 L4_INLINE
00195 l4re_env_t *l4re_env(void) L4_NOTHROW
00196 { return l4re_global_env; }
00197
00198 L4_INLINE
00199 l4_kernel_info_t const *l4re_kip(void) L4_NOTHROW
00200 { return l4_kip(); }
00201
00202 L4_INLINE l4re_env_cap_entry_t const *
00203 l4re_env_get_cap_l(char const *name, unsigned l, l4re_env_t const *e) L4_NOTHROW
00204 {
00205     l4re_env_cap_entry_t const *c = e->caps;
00206     for (; c && c->flags != ~0UL; ++c)
00207     {
00208         unsigned i;
00209         for (i = 0;
00210              i < sizeof(c->name) && i < l && c->name[i] && name[i] && name[i] == c->name[i];
00211              ++i)

```

16.333 I4/re/error_helper File Reference

```
Include dependency graph for error_helper:
```



16.334 error_helper

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -- Mode: C++ --
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014 #pragma once
00015
00016 #include <l4/sys/types.h>
00017 #include <l4/cxx/exceptions>
00018 #include <l4/cxx/type_traits>
00019 #include <l4/sys/err.h>
00020
00021 #include <stdarg.h>
00022 #include <stdio.h>
00023
00024 namespace L4Re {
00025
00026 #ifdef __EXCEPTIONS
00027
00037 [[noreturn]] inline void throw_error(long err, char const *extra = "")
00038 {
00039     switch (err)
00040     {
00041     case -L4_ENOENT: throw (L4::Element_not_found(extra));
00042     case -L4_ENOMEM: throw (L4::Out_of_memory(extra));
00043     case -L4_EEXIST: throw (L4::Element_already_exists(extra));
00044     case -L4_ERANGE: throw (L4::Bounds_error(extra));
00045     default: throw (L4::Runtime_error(err, extra));
00046     }
00047 }
00048
00049 [[noreturn]] inline void throw_error_fmt(l4_ret_t err, char const *const fmt, ...)
00050     __attribute__((format(printf, 2, 3)));
00051 [[noreturn]] inline void throw_error_fmt(l4_ret_t err, char const *const fmt, ...)
00052 {
00053     char extra[80];
00054     va_list argp;
00055     va_start(argp, fmt);
00056     vsnprintf(extra, sizeof(extra), fmt, argp);
00057     va_end(argp);
00058     throw_error(err, extra);
00059 }
00060
00071 inline
00072 l4_ret_t chksys(l4_ret_t err, char const *extra = "", l4_ret_t ret = 0)
00073 {
00074     if (L4_UNLIKELY(err < 0))
00075         throw_error(ret ? ret : err, extra);
00076
00077     return err;
00078 }
00079
00092 inline
00093 l4_ret_t chksys(l4_msgtag_t const &t, char const *extra = "",
00094                 l4_utcb_t *utcb = l4_utcb(), l4_ret_t ret = 0)
00095 {
00096     if (L4_UNLIKELY(t.has_error()))
00097         throw_error(ret ? ret : l4_error_u(t, utcb), extra);
00098     else if (L4_UNLIKELY(t.label() < 0))
00099         throw_error(ret ? ret : t.label(), extra);
00100
00101     return t.label();
00102 }
00103
00115 inline
00116 l4_ret_t chksys(l4_msgtag_t const &t, l4_utcb_t *utcb, char const *extra = "")
00117 { return chksys(t, extra, utcb); }
00118
00119 #if 0
00120 inline
00121 long chksys(l4_ret_t ret, l4_ret_t err, char const *extra = "")
00122 {
00123     if (L4_UNLIKELY(ret < 0))
00124         throw_error(err, extra);
00125
00126     return ret;
00127 }
00128 #endif

```

```

00129
00146 template<typename T>
00147 inline
00148 #if __cplusplus >= 201103L
00149 T chkcap(T &&cap, char const *extra = "", l4_ret_t err = -L4_ENOMEM)
00150 #else
00151 T chkcap(T cap, char const *extra = "", l4_ret_t err = -L4_ENOMEM)
00152 #endif
00153 {
00154     if (L4_UNLIKELY(!cap.is_valid()))
00155         throw_error(err ? err : cap.invalid_cap_error(), extra);
00156
00157 #if __cplusplus >= 201103L
00158     return cxx::forward<T>(cap);
00159 #else
00160     return cap;
00161 #endif
00162 }
00163
00178 inline
00179 l4_msgtag_t
00180 chkipc(l4_msgtag_t tag, char const *extra = "",
00181        l4_utcb_t *utcb = l4_utcb())
00182 {
00183     if (L4_UNLIKELY(tag.has_error()))
00184         chksys(l4_error_u(tag, utcb), extra);
00185
00186     return tag;
00187 }
00188 #endif
00189
00190 }

```

16.335 event-sys.h

```

00001 /*
00002  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009 namespace L4Re
00010 {
00011     namespace Event_
00012     {
00018         enum Opcodes
00019         {
00020             Get, Get_num_streams, Get_stream_info, Get_stream_info_for_id,
00021             Get_axis_info, Get_stream_state_for_id
00022         };
00023     };
00024 };

```

16.336 event_enums.h

```

00001 #pragma once
00002
00003 /*
00004  *
00005  *
00006  * Constants for L4Re events ...
00007  */
00008
00009
00010 enum L4Re_events_key
00011 {
00012     L4RE_KEY_RESERVED      = 0,
00013     L4RE_KEY_ESC           = 1,
00014     L4RE_KEY_1             = 2,
00015     L4RE_KEY_2             = 3,
00016     L4RE_KEY_3             = 4,
00017     L4RE_KEY_4             = 5,
00018     L4RE_KEY_5             = 6,
00019     L4RE_KEY_6             = 7,
00020     L4RE_KEY_7             = 8,
00021     L4RE_KEY_8             = 9,

```

```
00022 L4RE_KEY_9 = 10,
00023 L4RE_KEY_0 = 11,
00024 L4RE_KEY_MINUS = 12,
00025 L4RE_KEY_EQUAL = 13,
00026 L4RE_KEY_BACKSPACE = 14,
00027 L4RE_KEY_TAB = 15,
00028 L4RE_KEY_Q = 16,
00029 L4RE_KEY_W = 17,
00030 L4RE_KEY_E = 18,
00031 L4RE_KEY_R = 19,
00032 L4RE_KEY_T = 20,
00033 L4RE_KEY_Y = 21,
00034 L4RE_KEY_U = 22,
00035 L4RE_KEY_I = 23,
00036 L4RE_KEY_O = 24,
00037 L4RE_KEY_P = 25,
00038 L4RE_KEY_LEFTBRACE = 26,
00039 L4RE_KEY_RIGHTBRACE = 27,
00040 L4RE_KEY_ENTER = 28,
00041 L4RE_KEY_LEFTCTRL = 29,
00042 L4RE_KEY_A = 30,
00043 L4RE_KEY_S = 31,
00044 L4RE_KEY_D = 32,
00045 L4RE_KEY_F = 33,
00046 L4RE_KEY_G = 34,
00047 L4RE_KEY_H = 35,
00048 L4RE_KEY_J = 36,
00049 L4RE_KEY_K = 37,
00050 L4RE_KEY_L = 38,
00051 L4RE_KEY_SEMICOLON = 39,
00052 L4RE_KEY_APOSTROPHE = 40,
00053 L4RE_KEY_GRAVE = 41,
00054 L4RE_KEY_LEFTSHIFT = 42,
00055 L4RE_KEY_BACKSLASH = 43,
00056 L4RE_KEY_Z = 44,
00057 L4RE_KEY_X = 45,
00058 L4RE_KEY_C = 46,
00059 L4RE_KEY_V = 47,
00060 L4RE_KEY_B = 48,
00061 L4RE_KEY_N = 49,
00062 L4RE_KEY_M = 50,
00063 L4RE_KEY_COMMA = 51,
00064 L4RE_KEY_DOT = 52,
00065 L4RE_KEY_SLASH = 53,
00066 L4RE_KEY_RIGHTSHIFT = 54,
00067 L4RE_KEY_KPASTERISK = 55,
00068 L4RE_KEY_LEFTALT = 56,
00069 L4RE_KEY_SPACE = 57,
00070 L4RE_KEY_CAPSLOCK = 58,
00071 L4RE_KEY_F1 = 59,
00072 L4RE_KEY_F2 = 60,
00073 L4RE_KEY_F3 = 61,
00074 L4RE_KEY_F4 = 62,
00075 L4RE_KEY_F5 = 63,
00076 L4RE_KEY_F6 = 64,
00077 L4RE_KEY_F7 = 65,
00078 L4RE_KEY_F8 = 66,
00079 L4RE_KEY_F9 = 67,
00080 L4RE_KEY_F10 = 68,
00081 L4RE_KEY_NUMLOCK = 69,
00082 L4RE_KEY_SCROLLLOCK = 70,
00083 L4RE_KEY_KP7 = 71,
00084 L4RE_KEY_KP8 = 72,
00085 L4RE_KEY_KP9 = 73,
00086 L4RE_KEY_KPMINUS = 74,
00087 L4RE_KEY_KP4 = 75,
00088 L4RE_KEY_KP5 = 76,
00089 L4RE_KEY_KP6 = 77,
00090 L4RE_KEY_KPPPLUS = 78,
00091 L4RE_KEY_KP1 = 79,
00092 L4RE_KEY_KP2 = 80,
00093 L4RE_KEY_KP3 = 81,
00094 L4RE_KEY_KP0 = 82,
00095 L4RE_KEY_KPDOT = 83,
00096 L4RE_KEY_ZENKAKUHANKAKU = 85,
00097 L4RE_KEY_102ND = 86,
00098 L4RE_KEY_F11 = 87,
00099 L4RE_KEY_F12 = 88,
00100 L4RE_KEY_RO = 89,
00101 L4RE_KEY_KATAKANA = 90,
00102 L4RE_KEY_HIRAGANA = 91,
00103 L4RE_KEY_HENKAN = 92,
00104 L4RE_KEY_KATAKANAHIRAGANA = 93,
00105 L4RE_KEY_MUHENKAN = 94,
00106 L4RE_KEY_KPJPCOMMA = 95,
00107 L4RE_KEY_KPENTER = 96,
00108 L4RE_KEY_RIGHTCTRL = 97,
```

| | | |
|-------|------------------------|---------------------|
| 00109 | L4RE_KEY_KP SLASH | = 98, |
| 00110 | L4RE_KEY_SY SRQ | = 99, |
| 00111 | L4RE_KEY_RIGH TALT | = 100, |
| 00112 | L4RE_KEY_LIN EFEED | = 101, |
| 00113 | L4RE_KEY_HOM E | = 102, |
| 00114 | L4RE_KEY_UP | = 103, |
| 00115 | L4RE_KEY_PAG EUP | = 104, |
| 00116 | L4RE_KEY_LEF T | = 105, |
| 00117 | L4RE_KEY_RIG H | = 106, |
| 00118 | L4RE_KEY_END | = 107, |
| 00119 | L4RE_KEY_DOW N | = 108, |
| 00120 | L4RE_KEY_PAG EDOWN | = 109, |
| 00121 | L4RE_KEY_INS ERT | = 110, |
| 00122 | L4RE_KEY_DELE TE | = 111, |
| 00123 | L4RE_KEY_MAC RO | = 112, |
| 00124 | L4RE_KEY_MUTE | = 113, |
| 00125 | L4RE_KEY_VOLU MEDOWN | = 114, |
| 00126 | L4RE_KEY_VOLU MEUP | = 115, |
| 00127 | L4RE_KEY_POW ER | = 116, |
| 00128 | L4RE_KEY_KPE QUAL | = 117, |
| 00129 | L4RE_KEY_KP PLUSMINUS | = 118, |
| 00130 | L4RE_KEY_PAU SE | = 119, |
| 00131 | L4RE_KEY_KP COMMA | = 121, |
| 00132 | L4RE_KEY_HANG EUL | = 122, |
| 00133 | L4RE_KEY_HANG UEL | = L4RE_KEY_HANGEUL, |
| 00134 | L4RE_KEY_HAN JA | = 123, |
| 00135 | L4RE_KEY_YEN | = 124, |
| 00136 | L4RE_KEY_LEF TMETA | = 125, |
| 00137 | L4RE_KEY_RIGH TMETA | = 126, |
| 00138 | L4RE_KEY_COM POSE | = 127, |
| 00139 | L4RE_KEY_STO P | = 128, |
| 00140 | L4RE_KEY_AGA IN | = 129, |
| 00141 | L4RE_KEY_PRO PS | = 130, |
| 00142 | L4RE_KEY_UNDO | = 131, |
| 00143 | L4RE_KEY_FRO NT | = 132, |
| 00144 | L4RE_KEY_COP Y | = 133, |
| 00145 | L4RE_KEY_OPE N | = 134, |
| 00146 | L4RE_KEY_PAS TE | = 135, |
| 00147 | L4RE_KEY_FIN D | = 136, |
| 00148 | L4RE_KEY_CUT | = 137, |
| 00149 | L4RE_KEY_HELP | = 138, |
| 00150 | L4RE_KEY_MEN U | = 139, |
| 00151 | L4RE_KEY_CALC | = 140, |
| 00152 | L4RE_KEY_SETU P | = 141, |
| 00153 | L4RE_KEY_SLEE P | = 142, |
| 00154 | L4RE_KEY_WAKE UP | = 143, |
| 00155 | L4RE_KEY_FILE | = 144, |
| 00156 | L4RE_KEY_SEN DFILE | = 145, |
| 00157 | L4RE_KEY_DELE TEFILE | = 146, |
| 00158 | L4RE_KEY_XFER | = 147, |
| 00159 | L4RE_KEY_PRO G1 | = 148, |
| 00160 | L4RE_KEY_PRO G2 | = 149, |
| 00161 | L4RE_KEY_WWW | = 150, |
| 00162 | L4RE_KEY_MS DOS | = 151, |
| 00163 | L4RE_KEY_COFF EE | = 152, |
| 00164 | L4RE_KEY_DIRE CTION | = 153, |
| 00165 | L4RE_KEY_CYCLEWINDOWS | = 154, |
| 00166 | L4RE_KEY_MAIL | = 155, |
| 00167 | L4RE_KEY_BOOK MARKS | = 156, |
| 00168 | L4RE_KEY_COMPU TER | = 157, |
| 00169 | L4RE_KEY_BACK | = 158, |
| 00170 | L4RE_KEY_FORW ARD | = 159, |
| 00171 | L4RE_KEY_CLOSE CD | = 160, |
| 00172 | L4RE_KEY_EJEC TCD | = 161, |
| 00173 | L4RE_KEY_EJEC TCLOSECD | = 162, |
| 00174 | L4RE_KEY_NEXT SONG | = 163, |
| 00175 | L4RE_KEY_PLAY PAUSE | = 164, |
| 00176 | L4RE_KEY_PREV IOUSSONG | = 165, |
| 00177 | L4RE_KEY_STOP CD | = 166, |
| 00178 | L4RE_KEY_REC ORD | = 167, |
| 00179 | L4RE_KEY_REWI ND | = 168, |
| 00180 | L4RE_KEY_PHO NE | = 169, |
| 00181 | L4RE_KEY_ISO | = 170, |
| 00182 | L4RE_KEY_CONF IG | = 171, |
| 00183 | L4RE_KEY_HOM EPAGE | = 172, |
| 00184 | L4RE_KEY_REFR ES | = 173, |
| 00185 | L4RE_KEY_EXIT | = 174, |
| 00186 | L4RE_KEY_MOVE | = 175, |
| 00187 | L4RE_KEY_EDIT | = 176, |
| 00188 | L4RE_KEY_SCROLLUP | = 177, |
| 00189 | L4RE_KEY_SCROLLDOWN | = 178, |
| 00190 | L4RE_KEY_KPLEF TPAREN | = 179, |
| 00191 | L4RE_KEY_KPRIGH TPAREN | = 180, |
| 00192 | L4RE_KEY_NEW | = 181, |
| 00193 | L4RE_KEY_REDO | = 182, |
| 00194 | L4RE_KEY_F13 | = 183, |
| 00195 | L4RE_KEY_F14 | = 184, |


```
00196 L4RE_KEY_F15 = 185,
00197 L4RE_KEY_F16 = 186,
00198 L4RE_KEY_F17 = 187,
00199 L4RE_KEY_F18 = 188,
00200 L4RE_KEY_F19 = 189,
00201 L4RE_KEY_F20 = 190,
00202 L4RE_KEY_F21 = 191,
00203 L4RE_KEY_F22 = 192,
00204 L4RE_KEY_F23 = 193,
00205 L4RE_KEY_F24 = 194,
00206 L4RE_KEY_PLAYCD = 200,
00207 L4RE_KEY_PAUSECD = 201,
00208 L4RE_KEY_PROG3 = 202,
00209 L4RE_KEY_PROG4 = 203,
00210 L4RE_KEY_SUSPEND = 205,
00211 L4RE_KEY_CLOSE = 206,
00212 L4RE_KEY_PLAY = 207,
00213 L4RE_KEY_FASTFORWARD = 208,
00214 L4RE_KEY_BASSBOOST = 209,
00215 L4RE_KEY_PRINT = 210,
00216 L4RE_KEY_HP = 211,
00217 L4RE_KEY_CAMERA = 212,
00218 L4RE_KEY_SOUND = 213,
00219 L4RE_KEY_QUESTION = 214,
00220 L4RE_KEY_EMAIL = 215,
00221 L4RE_KEY_CHAT = 216,
00222 L4RE_KEY_SEARCH = 217,
00223 L4RE_KEY_CONNECT = 218,
00224 L4RE_KEY_FINANCE = 219,
00225 L4RE_KEY_SPORT = 220,
00226 L4RE_KEY_SHOP = 221,
00227 L4RE_KEY_ALTERASE = 222,
00228 L4RE_KEY_CANCEL = 223,
00229 L4RE_KEY_BRIGHTNESSDOWN = 224,
00230 L4RE_KEY_BRIGHTNESSUP = 225,
00231 L4RE_KEY_MEDIA = 226,
00232 L4RE_KEY_SWITCHVIDEOMODE = 227,
00233 L4RE_KEY_KBDILLUMTOGGLE = 228,
00234 L4RE_KEY_KBDILLUMDOWN = 229,
00235 L4RE_KEY_KBDILLUMUP = 230,
00236 L4RE_KEY_SEND = 231,
00237 L4RE_KEY_REPLY = 232,
00238 L4RE_KEY_FORWARDMAIL = 233,
00239 L4RE_KEY_SAVE = 234,
00240 L4RE_KEY_DOCUMENTS = 235,
00241 L4RE_KEY_UNKNOWN = 240,
00242 L4RE_KEY_OK = 0x160,
00243 L4RE_KEY_SELECT = 0x161,
00244 L4RE_KEY_GOTO = 0x162,
00245 L4RE_KEY_CLEAR = 0x163,
00246 L4RE_KEY_POWER2 = 0x164,
00247 L4RE_KEY_OPTION = 0x165,
00248 L4RE_KEY_INFO = 0x166,
00249 L4RE_KEY_TIME = 0x167,
00250 L4RE_KEY_VENDOR = 0x168,
00251 L4RE_KEY_ARCHIVE = 0x169,
00252 L4RE_KEY_PROGRAM = 0x16a,
00253 L4RE_KEY_CHANNEL = 0x16b,
00254 L4RE_KEY_FAVORITES = 0x16c,
00255 L4RE_KEY_EPG = 0x16d,
00256 L4RE_KEY_PVR = 0x16e,
00257 L4RE_KEY_MHP = 0x16f,
00258 L4RE_KEY_LANGUAGE = 0x170,
00259 L4RE_KEY_TITLE = 0x171,
00260 L4RE_KEY_SUBTITLE = 0x172,
00261 L4RE_KEY_ANGLE = 0x173,
00262 L4RE_KEY_ZOOM = 0x174,
00263 L4RE_KEY_MODE = 0x175,
00264 L4RE_KEY_KEYBOARD = 0x176,
00265 L4RE_KEY_SCREEN = 0x177,
00266 L4RE_KEY_PC = 0x178,
00267 L4RE_KEY_TV = 0x179,
00268 L4RE_KEY_TV2 = 0x17a,
00269 L4RE_KEY_VCR = 0x17b,
00270 L4RE_KEY_VCR2 = 0x17c,
00271 L4RE_KEY_SAT = 0x17d,
00272 L4RE_KEY_SAT2 = 0x17e,
00273 L4RE_KEY_CD = 0x17f,
00274 L4RE_KEY_TAPE = 0x180,
00275 L4RE_KEY_RADIO = 0x181,
00276 L4RE_KEY_TUNER = 0x182,
00277 L4RE_KEY_PLAYER = 0x183,
00278 L4RE_KEY_TEXT = 0x184,
00279 L4RE_KEY_DVD = 0x185,
00280 L4RE_KEY_AUX = 0x186,
00281 L4RE_KEY_MP3 = 0x187,
00282 L4RE_KEY_AUDIO = 0x188,
```

```
00283 L4RE_KEY_VIDEO = 0x189,
00284 L4RE_KEY_DIRECTORY = 0x18a,
00285 L4RE_KEY_LIST = 0x18b,
00286 L4RE_KEY_MEMO = 0x18c,
00287 L4RE_KEY_CALENDAR = 0x18d,
00288 L4RE_KEY_RED = 0x18e,
00289 L4RE_KEY_GREEN = 0x18f,
00290 L4RE_KEY_YELLOW = 0x190,
00291 L4RE_KEY_BLUE = 0x191,
00292 L4RE_KEY_CHANNELUP = 0x192,
00293 L4RE_KEY_CHANNELDOWN = 0x193,
00294 L4RE_KEY_FIRST = 0x194,
00295 L4RE_KEY_LAST = 0x195,
00296 L4RE_KEY_AB = 0x196,
00297 L4RE_KEY_NEXT = 0x197,
00298 L4RE_KEY_RESTART = 0x198,
00299 L4RE_KEY_SLOW = 0x199,
00300 L4RE_KEY_SHUFFLE = 0x19a,
00301 L4RE_KEY_BREAK = 0x19b,
00302 L4RE_KEY_PREVIOUS = 0x19c,
00303 L4RE_KEY_DIGITS = 0x19d,
00304 L4RE_KEY_TEEN = 0x19e,
00305 L4RE_KEY_TWEN = 0x19f,
00306 L4RE_KEY_DEL_EOL = 0x1c0,
00307 L4RE_KEY_DEL_EOS = 0x1c1,
00308 L4RE_KEY_INS_LINE = 0x1c2,
00309 L4RE_KEY_DEL_LINE = 0x1c3,
00310 L4RE_KEY_FN = 0x1d0,
00311 L4RE_KEY_FN_ESC = 0x1d1,
00312 L4RE_KEY_FN_F1 = 0x1d2,
00313 L4RE_KEY_FN_F2 = 0x1d3,
00314 L4RE_KEY_FN_F3 = 0x1d4,
00315 L4RE_KEY_FN_F4 = 0x1d5,
00316 L4RE_KEY_FN_F5 = 0x1d6,
00317 L4RE_KEY_FN_F6 = 0x1d7,
00318 L4RE_KEY_FN_F7 = 0x1d8,
00319 L4RE_KEY_FN_F8 = 0x1d9,
00320 L4RE_KEY_FN_F9 = 0x1da,
00321 L4RE_KEY_FN_F10 = 0x1db,
00322 L4RE_KEY_FN_F11 = 0x1dc,
00323 L4RE_KEY_FN_F12 = 0x1dd,
00324 L4RE_KEY_FN_1 = 0x1de,
00325 L4RE_KEY_FN_2 = 0x1df,
00326 L4RE_KEY_FN_D = 0x1e0,
00327 L4RE_KEY_FN_E = 0x1e1,
00328 L4RE_KEY_FN_F = 0x1e2,
00329 L4RE_KEY_FN_S = 0x1e3,
00330 L4RE_KEY_FN_B = 0x1e4,
00331 L4RE_KEY_MAX = 0x1ff,
00332 };
00333
00334 enum L4Re_events_rel
00335 {
00336     L4RE_REL_X = 0x00,
00337     L4RE_REL_Y = 0x01,
00338     L4RE_REL_Z = 0x02,
00339     L4RE_REL_RX = 0x03,
00340     L4RE_REL_RY = 0x04,
00341     L4RE_REL_RZ = 0x05,
00342     L4RE_REL_HWHEEL = 0x06,
00343     L4RE_REL_DIAL = 0x07,
00344     L4RE_REL_WHEEL = 0x08,
00345     L4RE_REL_MISC = 0x09,
00346     L4RE_REL_MAX = 0x0f,
00347 };
00348
00349 enum L4Re_events_snd
00350 {
00351     L4RE_SND_CLICK = 0x00,
00352     L4RE_SND_BELL = 0x01,
00353     L4RE_SND_TONE = 0x02,
00354     L4RE_SND_MAX = 0x07,
00355 };
00356
00357 enum L4Re_events_rep
00358 {
00359     L4RE_REP_DELAY = 0x00,
00360     L4RE_REP_PERIOD = 0x01,
00361     L4RE_REP_MAX = 0x01,
00362 };
00363
00364 enum L4Re_events_led
00365 {
00366     L4RE_LED_NUML = 0x00,
00367     L4RE_LED_CAPSL = 0x01,
00368     L4RE_LED_SCROLLL = 0x02,
00369     L4RE_LED_COMPOSE = 0x03,
```

```
00370     L4RE_LED_KANA           = 0x04,
00371     L4RE_LED_SLEEP          = 0x05,
00372     L4RE_LED_SUSPEND        = 0x06,
00373     L4RE_LED_MUTE           = 0x07,
00374     L4RE_LED_MISC           = 0x08,
00375     L4RE_LED_MAIL           = 0x09,
00376     L4RE_LED_CHARGING        = 0x0a,
00377     L4RE_LED_MAX             = 0x0f,
00378 };
00379
00380 enum L4Re_events_btn
00381 {
00382     L4RE_BTN_MISC            = 0x100,
00383     L4RE_BTN_0               = 0x100,
00384     L4RE_BTN_1               = 0x101,
00385     L4RE_BTN_2               = 0x102,
00386     L4RE_BTN_3               = 0x103,
00387     L4RE_BTN_4               = 0x104,
00388     L4RE_BTN_5               = 0x105,
00389     L4RE_BTN_6               = 0x106,
00390     L4RE_BTN_7               = 0x107,
00391     L4RE_BTN_8               = 0x108,
00392     L4RE_BTN_9               = 0x109,
00393     L4RE_BTN_MOUSE           = 0x110,
00394     L4RE_BTN_LEFT            = 0x110,
00395     L4RE_BTN_RIGHT           = 0x111,
00396     L4RE_BTN_MIDDLE           = 0x112,
00397     L4RE_BTN_SIDE            = 0x113,
00398     L4RE_BTN_EXTRA            = 0x114,
00399     L4RE_BTN_FORWARD          = 0x115,
00400     L4RE_BTN_BACK            = 0x116,
00401     L4RE_BTN_TASK            = 0x117,
00402     L4RE_BTN_JOYSTICK         = 0x120,
00403     L4RE_BTN_TRIGGER          = 0x120,
00404     L4RE_BTN_THUMB           = 0x121,
00405     L4RE_BTN_THUMB2          = 0x122,
00406     L4RE_BTN_TOP             = 0x123,
00407     L4RE_BTN_TOP2            = 0x124,
00408     L4RE_BTN_PINKIE           = 0x125,
00409     L4RE_BTN_BASE            = 0x126,
00410     L4RE_BTN_BASE2           = 0x127,
00411     L4RE_BTN_BASE3           = 0x128,
00412     L4RE_BTN_BASE4           = 0x129,
00413     L4RE_BTN_BASE5           = 0x12a,
00414     L4RE_BTN_BASE6           = 0x12b,
00415     L4RE_BTN_DEAD            = 0x12f,
00416     L4RE_BTN_GAMEPAD         = 0x130,
00417     L4RE_BTN_A               = 0x130,
00418     L4RE_BTN_B               = 0x131,
00419     L4RE_BTN_C               = 0x132,
00420     L4RE_BTN_X               = 0x133,
00421     L4RE_BTN_Y               = 0x134,
00422     L4RE_BTN_Z               = 0x135,
00423     L4RE_BTN_TL              = 0x136,
00424     L4RE_BTN_TR              = 0x137,
00425     L4RE_BTN_TL2             = 0x138,
00426     L4RE_BTN_TR2             = 0x139,
00427     L4RE_BTN_SELECT          = 0x13a,
00428     L4RE_BTN_START           = 0x13b,
00429     L4RE_BTN_MODE            = 0x13c,
00430     L4RE_BTN_THUMBL          = 0x13d,
00431     L4RE_BTN_THUMBR          = 0x13e,
00432     L4RE_BTN_DIGI            = 0x140,
00433     L4RE_BTN_TOOL_PEN        = 0x140,
00434     L4RE_BTN_TOOL_RUBBER     = 0x141,
00435     L4RE_BTN_TOOL_BRUSH      = 0x142,
00436     L4RE_BTN_TOOL_PENCIL     = 0x143,
00437     L4RE_BTN_TOOL_AIRBRUSH   = 0x144,
00438     L4RE_BTN_TOOL_FINGER     = 0x145,
00439     L4RE_BTN_TOOL_MOUSE      = 0x146,
00440     L4RE_BTN_TOOL_LENS       = 0x147,
00441     L4RE_BTN_TOUCH           = 0x14a,
00442     L4RE_BTN_STYLUS          = 0x14b,
00443     L4RE_BTN_STYLUS2         = 0x14c,
00444     L4RE_BTN_TOOL_DOUBLETAP  = 0x14d,
00445     L4RE_BTN_TOOL_TRIPLETAP  = 0x14e,
00446     L4RE_BTN_WHEEL           = 0x150,
00447     L4RE_BTN_GEAR_DOWN       = 0x150,
00448     L4RE_BTN_GEAR_UP         = 0x151,
00449 };
00450
00451 enum L4Re_events_sw
00452 {
00453     L4RE_SW_0                = 0x00,
00454     L4RE_SW_1                = 0x01,
00455     L4RE_SW_2                = 0x02,
00456     L4RE_SW_3                = 0x03,
```

```
00457     L4RE_SW_4      = 0x04,
00458     L4RE_SW_5      = 0x05,
00459     L4RE_SW_6      = 0x06,
00460     L4RE_SW_7      = 0x07,
00461     L4RE_SW_MAX     = 0x0f,
00462 };
00463
00464 enum L4Re_events_ev
00465 {
00466     L4RE_EV_SYN      = 0x00,
00467     L4RE_EV_KEY      = 0x01,
00468     L4RE_EV_REL      = 0x02,
00469     L4RE_EV_ABS      = 0x03,
00470     L4RE_EV_MSC      = 0x04,
00471     L4RE_EV_SW       = 0x05,
00472     L4RE_EV_LED      = 0x11,
00473     L4RE_EV_SND      = 0x12,
00474     L4RE_EV_REP      = 0x14,
00475     L4RE_EV_FF       = 0x15,
00476     L4RE_EV_PWR      = 0x16,
00477     L4RE_EV_FF_STATUS = 0x17,
00478     L4RE_EV_WINDOW   = 0x18,
00479     L4RE_EV_PM       = 0x1e, // power management signals
00480     L4RE_EV_MAX      = 0x1f,
00481 };
00482
00483 enum L4Re_events_syn
00484 {
00485     L4RE_SYN_REPORT   = 0,
00486     L4RE_SYN_CONFIG   = 1,
00487     L4RE_SYN_MT_REPORT = 2,
00488
00489     L4RE_SYN_STREAM_CFG = 0x80,
00490 };
00491
00492 enum L4Re_stream_cfg
00493 {
00494     L4RE_SYN_STREAM_NEW = 0,
00495     L4RE_SYN_STREAM_CLOSE = 1,
00496 };
00497
00498 enum L4Re_events_abs
00499 {
00500     L4RE_ABS_X        = 0x00,
00501     L4RE_ABS_Y        = 0x01,
00502     L4RE_ABS_Z        = 0x02,
00503     L4RE_ABS_RX       = 0x03,
00504     L4RE_ABS_RY       = 0x04,
00505     L4RE_ABS_RZ       = 0x05,
00506     L4RE_ABS_THROTTLE = 0x06,
00507     L4RE_ABS_RUDDER   = 0x07,
00508     L4RE_ABS_WHEEL    = 0x08,
00509     L4RE_ABS_GAS      = 0x09,
00510     L4RE_ABS_BRAKE    = 0x0a,
00511     L4RE_ABS_HAT0X    = 0x10,
00512     L4RE_ABS_HAT0Y    = 0x11,
00513     L4RE_ABS_HAT1X    = 0x12,
00514     L4RE_ABS_HAT1Y    = 0x13,
00515     L4RE_ABS_HAT2X    = 0x14,
00516     L4RE_ABS_HAT2Y    = 0x15,
00517     L4RE_ABS_HAT3X    = 0x16,
00518     L4RE_ABS_HAT3Y    = 0x17,
00519     L4RE_ABS_PRESSURE = 0x18,
00520     L4RE_ABS_DISTANCE = 0x19,
00521     L4RE_ABS_TILT_X   = 0x1a,
00522     L4RE_ABS_TILT_Y   = 0x1b,
00523     L4RE_ABS_TOOL_WIDTH = 0x1c,
00524     L4RE_ABS_VOLUME   = 0x20,
00525     L4RE_ABS_MISC     = 0x28,
00526     L4RE_ABS_MT_TOUCH_MAJOR = 0x30,
00527     L4RE_ABS_MT_TOUCH_MINOR = 0x31,
00528     L4RE_ABS_MT_WIDTH_MAJOR = 0x32,
00529     L4RE_ABS_MT_WIDTH_MINOR = 0x33,
00530     L4RE_ABS_MT_ORIENTATION = 0x34,
00531     L4RE_ABS_MT_POSITION_X = 0x35,
00532     L4RE_ABS_MT_POSITION_Y = 0x36,
00533     L4RE_ABS_MT_TOOL_TYPE = 0x37,
00534     L4RE_ABS_MT_BLOB_ID = 0x38,
00535     L4RE_ABS_MT_TRACKING_ID = 0x39,
00536     L4RE_ABS_MT_PRESSURE = 0x3a,
00537     L4RE_ABS_MT_DISTANCE = 0x3b,
00538
00539     L4RE_ABS_MAX      = 0x3f,
00540 };
00541
00542 enum L4Re_events_msc
00543 {
```

16.337 l4/re/impl/dataspace_impl.h File Reference

```
#include <l4/re/dataspace>
#include <l4/sys/cxx/ipc_client>
#include <l4/sys/cxx/consts>
Include dependency graph for dataspace_impl.h:
```



- ### 16.337.1 Detailed Description

Definition in file [dataspace_impl.h](#).

16.338 dataspace_impl.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012
00013 #include <l4/re/dataspace>
00014 #include <l4/sys/cxx/ipc_client>
00015 #include <l4/sys/cxx/consts>
00016
00017 L4_RPC_DEF(L4Re::Dataspace::clear);
00018 L4_RPC_DEF(L4Re::Dataspace::allocate);
00019 L4_RPC_DEF(L4Re::Dataspace::copy_in);
00020 L4_RPC_DEF(L4Re::Dataspace::info);
00021 L4_RPC_DEF(L4Re::Dataspace::map_info);
00022
00023 namespace L4Re {
00024
00025     l4_ret_t
00026     Dataspace::__map(Dataspace::Offset offset, unsigned char *size,
00027                     Dataspace::Flags flags,
00028                     Dataspace::Map_addr local_addr,
00029                     L4::Cap<L4::Task> dst) const noexcept
00030     {
00031         Map_addr spot = local_addr & ~(~0ULL « l4_umword_t(*size));
00032         Map_addr base = local_addr & (~0ULL « l4_umword_t(*size));
00033         L4::Ipc::Rcv_fpage r = L4::Ipc::Rcv_fpage::mem(base, *size, 0, dst);
00034
00035         L4::Ipc::Snd_fpage fp;
00036         l4_ret_t err = map_t::call(c(), offset, spot, flags, r, fp, l4_utcb());
00037         if (L4_UNLIKELY(err < 0))
00038             return err;
00039
00040         *size = fp.rcv_order();
00041         return err;
00042     }
00043
00044     l4_ret_t
00045     Dataspace::map_region(Dataspace::Offset offset, Dataspace::Flags flags,
00046                           Dataspace::Map_addr min_addr,
00047                           Dataspace::Map_addr max_addr,
00048                           L4::Cap<L4::Task> dst) const noexcept
00049     {
00050         min_addr = L4::trunc_page(min_addr);
00051         max_addr = L4::round_page(max_addr);
00052         unsigned char order = L4_LOG2_PAGESIZE;
00053
00054         l4_ret_t err = 0;
00055
00056         while (min_addr < max_addr)
00057         {
00058             unsigned char order_mapped;
00059             order_mapped = order
00060                 = L4::max_order(order, min_addr, min_addr, max_addr, min_addr);
00061
00062             err = __map(offset, &order_mapped, flags, min_addr, dst);
00063             if (L4_UNLIKELY(err < 0))
00064                 return err;
00065
00066             if (order > order_mapped)
00067                 order = order_mapped;
00068
00069             min_addr += Map_addr(1) « order;
00070             offset += Map_addr(1) « order;
00071
00072             if (min_addr >= max_addr)
00073                 return 0;
00074
00075             while (min_addr != L4::trunc_order(min_addr, order)
00076                 || max_addr < L4::round_order(min_addr + 1, order))
00077                 --order;
00078         }
00079
00080         return 0;
00081     }
00082
00083     l4_ret_t
00084     Dataspace::map(Dataspace::Offset offset, Dataspace::Flags flags,
00085                    Dataspace::Map_addr local_addr,

```

```

00086         Dataspace::Map_addr min_addr,
00087         Dataspace::Map_addr max_addr,
00088         L4::Cap<L4::Task> dst) const noexcept
00089 {
00090     min_addr = L4::trunc_page(min_addr);
00091     max_addr = L4::round_page(max_addr);
00092     local_addr = L4::trunc_page(local_addr);
00093     unsigned char order
00094         = L4::max_order(L4_LOG2_PAGE_SIZE, local_addr, min_addr, max_addr, local_addr);
00095
00096     return __map(offset, &order, flags, local_addr, dst);
00097 }
00098
00099 Dataspace::Size
00100 Dataspace::size() const noexcept
00101 {
00102     Stats stats = Stats();
00103
00104     int err = info(&stats);
00105     if (err < 0)
00106         return 0;
00107
00108     return stats.size;
00109 }
00110
00111 Dataspace::Flags
00112 Dataspace::flags() const noexcept
00113 {
00114     Stats stats = Stats();
00115
00116     int err = info(&stats);
00117     if (err < 0)
00118         return Flags(0);
00119
00120     return stats.flags;
00121 }
00122
00123 };

```

16.339 l4/re/impl/mem_alloc_impl.h File Reference

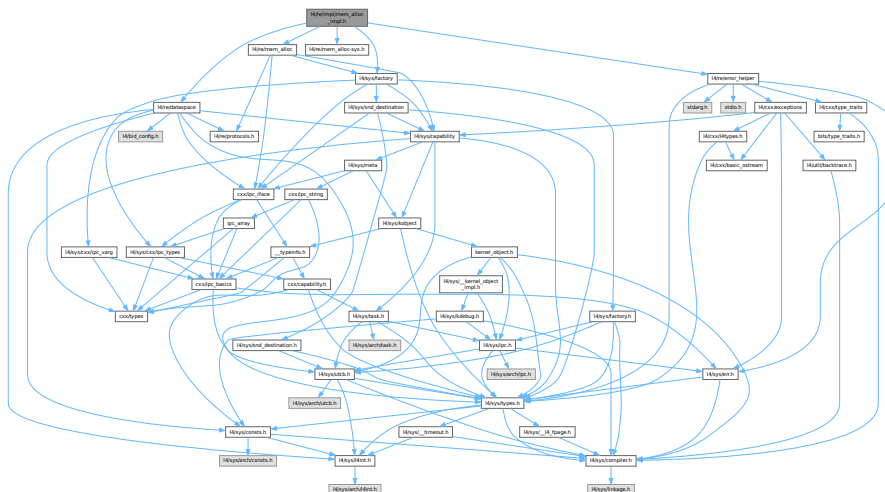
Memory allocator client stub implementation.

```

#include <l4/re/mem_alloc>
#include <l4/re/mem_alloc-sys.h>
#include <l4/re/dataspace>
#include <l4/re/error_helper>
#include <l4/sys/factory>

```

Include dependency graph for mem_alloc_impl.h:



Namespaces

- namespace [L4Re](#)
L4Re C++ Interfaces.

16.339.1 Detailed Description

Memory allocator client stub implementation.

Definition in file [mem_alloc_impl.h](#).

16.340 mem_alloc_impl.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *           Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *           economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #include <l4/re/mem_alloc>
00013 #include <l4/re/mem_alloc-sys.h>
00014 #include <l4/re/dataspace>
00015 #include <l4/re/error_helper>
00016
00017 #include <l4/sys/factory>
00018
00019
00020 namespace L4Re
00021 {
00022
00023     l4_ret_t
00024     Mem_alloc::alloc(long size,
00025                     L4::Cap<Dataspace> mem, unsigned long flags,
00026                     unsigned long align, l4_addr_t paddr) const noexcept
00027     {
00028         L4::Cap<L4::Factory> f(cap());
00029         auto call = f->create(mem, L4Re::Dataspace::Protocol);
00030         call << l4_mword_t(size)
00031              << l4_umword_t(flags)
00032              << l4_umword_t(align);
00033         if (flags & Fixed_paddr)
00034             call << l4_umword_t(paddr);
00035         return l4_error(call);
00036     }
00037
00038 };

```

16.341 l4/re/impl/namespace_impl.h File Reference

Namespace client stub implementation.

```

#include <l4/re/namespace>
#include <l4/util/util.h>
#include <l4/sys/cxx/ipc_client>
#include <l4/sys/assert.h>

```



```

00020 L4_RPC_DEF(L4Re::Namespace::query);
00021 L4_RPC_DEF(L4Re::Namespace::register_obj);
00022 L4_RPC_DEF(L4Re::Namespace::unlink);
00023
00024 namespace L4Re {
00025
00026 l4_ret_t
00027 Namespace::_query(char const *name, unsigned len,
00028                  L4::Cap<void> const &target,
00029                  l4_umword_t *local_id, bool iterate) const noexcept
00030 {
00031     l4_assert(target.is_valid());
00032
00033     L4::Cap<Namespace> ns = c();
00034     L4::Ipc::Array<char const, unsigned long> _name(len, name);
00035
00036     while (_name.length > 0)
00037     {
00038         L4::Ipc::Snd_fpage cap;
00039         L4::Opcode dummy;
00040         int err = query_t::call(ns, _name,
00041                                L4::Ipc::Small_buf(target.cap(),
00042                                                    local_id
00043                                                    ? L4_RCV_ITEM_LOCAL_ID
00044                                                    : 0),
00045                                cap, dummy, _name);
00046         if (err < 0)
00047             return err;
00048
00049         bool const partly = err & Partly_resolved;
00050         if (cap.id_received())
00051         {
00052             *local_id = cap.data();
00053             return _name.length;
00054         }
00055
00056         if (partly && iterate)
00057             ns = L4::cap_cast<Namespace>(target);
00058         else
00059             return err;
00060     }
00061
00062     return _name.length;
00063 }
00064
00065 l4_ret_t
00066 Namespace::query(char const *name, unsigned len, L4::Cap<void> const &target,
00067                  int timeout, l4_umword_t *local_id, bool iterate) const noexcept
00068 {
00069     if (L4_UNLIKELY(len == 0))
00070         return -L4_EINVAL;
00071
00072     if (L4_UNLIKELY(timeout < 0))
00073         return -L4_EINVAL;
00074
00075     l4_ret_t ret;
00076     long rem = timeout;
00077     long to = 0;
00078
00079     if (rem)
00080         to = 10;
00081
00082     do
00083     {
00084         ret = _query(name, len, target, local_id, iterate);
00085
00086         if (ret >= 0)
00087             return ret;
00088
00089         if (L4_UNLIKELY(ret != -L4_EAGAIN))
00090             return ret;
00091
00092         if (rem == to)
00093             return ret;
00094
00095         l4_sleep(to);
00096
00097         if (rem > 0)
00098         {
00099             rem -= to;
00100             if (rem < 0)
00101             {
00102                 to = rem = 0;
00103                 continue; // one final try
00104             }
00105         }
00106     }

```

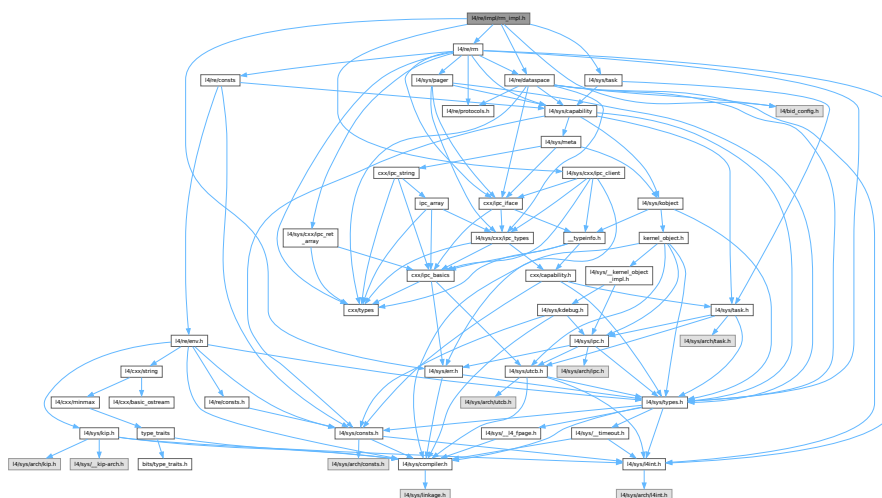
```
00107         if (to < 100)
00108             to += to;
00109     }
00110     while (486);
00111 }
00112
00113 l4_ret_t
00114 Namespace::query(char const *name, L4::Cap<void> const &target,
00115                 int timeout, l4_umword_t *local_id,
00116                 bool iterate) const noexcept
00117 {
00118     return query(name, __builtin_strlen(name), target,
00119                 timeout, local_id, iterate);
00120 }
00121
00122 }
```

16.343 l4/re/impl/rm_impl.h File Reference

Region map client stub implementation.

```
#include <l4/bid_config.h>
#include <l4/re/rm>
#include <l4/re/dataspace>
#include <l4/sys/cxx/ipc_client>
#include <l4/sys/task>
#include <l4/sys/err.h>
```

Include dependency graph for rm_impl.h:



Namespaces

- namespace **L4Re**
L4Re C++ Interfaces.

16.343.1 Detailed Description

Region map client stub implementation.

Definition in file [rm_impl.h](#).

16.344 rm_impl.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012
00013 #include <l4/bid_config.h>
00014 #include <l4/re/rm>
00015 #include <l4/re/dataspace>
00016
00017 #include <l4/sys/cxx/ipc_client>
00018
00019 #include <l4/sys/task>
00020 #include <l4/sys/err.h>
00021
00022 L4_RPC_DEF(L4Re::Rm::reserve_area);
00023 L4_RPC_DEF(L4Re::Rm::free_area);
00024 L4_RPC_DEF(L4Re::Rm::attach);
00025 L4_RPC_DEF(L4Re::Rm::detach);
00026 L4_RPC_DEF(L4Re::Rm::get_regions);
00027 L4_RPC_DEF(L4Re::Rm::get_areas);
00028 L4_RPC_DEF(L4Re::Rm::find);
00029 L4_RPC_DEF(L4Re::Rm::get_info);
00030
00031 namespace L4Re
00032 {
00033
00034     l4_ret_t
00035     Rm::attach(l4_addr_t *start, unsigned long size, Rm::Flags flags,
00036               L4::Ipc::Cap<Dataspace> mem, Rm::Offset offs,
00037               unsigned char align, L4::Cap<L4::Task> const task,
00038               char const *name, Rm::Offset backing_offset) const noexcept
00039     {
00040         if ((flags & F::Rights_mask) == Flags(0)
00041             || (flags & (F::Reserved | F::Kernel)))
00042             mem = L4::Ipc::Cap<L4Re::Dataspace>();
00043
00044         char const n = '\0';
00045         l4_ret_t e = attach_t::call(c(), start, size, flags, mem, offs, align,
00046                                    mem.cap().cap(), name ? name : &n, backing_offset);
00047         if (e < 0)
00048             return e;
00049
00050 #ifdef CONFIG_MMU
00051         if ((flags & (F::Eager_map | F::No_eager_map)) == F::Eager_map)
00052             #else
00053             if (!(flags & F::No_eager_map) && mem.is_valid())
00054             #endif
00055                 e = mem.cap()->map_region(offs, flags.map_flags(), *start, *start + size,
00056                                           task);
00057
00058         return e;
00059     }
00060
00061     l4_ret_t
00062     Rm::detach(l4_addr_t start, unsigned long size, L4::Cap<Dataspace> *mem,
00063               L4::Cap<L4::Task> task, unsigned flags) const noexcept
00064     {
00065         l4_addr_t rstart = 0, rsize = 0;
00066         l4_cap_idx_t mem_cap = L4_INVALID_CAP;
00067         l4_ret_t e = detach_t::call(c(), start, size, flags, rstart, rsize, mem_cap);
00068         if (L4_UNLIKELY(e < 0))
00069             return e;
00070
00071         if (mem)
00072             *mem = L4::Cap<L4Re::Dataspace>(mem_cap);
00073
00074         if (!task.is_valid())
00075             return e;
00076
00077         rsize = l4_round_page(rsize);
00078         unsigned order = L4_LOG2_PAGESIZE;
00079         unsigned long sz = (1UL << order);
00080         for (unsigned long p = rstart; rsize; p += sz, rsize -= sz)
00081         {
00082             while (sz > rsize)
00083             {
00084                 --order;
00085                 sz >>= 1;

```

```

00086     }
00087
00088     for (;;)
00089     {
00090         unsigned long m = sz << 1;
00091         if (m > rsize)
00092             break;
00093
00094         if (p & (m - 1))
00095             break;
00096
00097         ++order;
00098         sz <= 1;
00099     }
00100
00101     task->unmap(l4_fpage(p, order, L4_FPAGE_RWX),
00102                L4_FP_ALL_SPACES);
00103 }
00104
00105 return e;
00106 }
00107
00108 }

```

16.345 inhibitor

```

00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2014 Steffen Liebergeld <steffen.liebergeld@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include <l4/sys/capability>
00010 #include <l4/sys/cxx/ipc_iface>
00011 #include <l4/sys/cxx/ipc_string>
00012 #include <l4/re/protocols.h>
00013
00014 namespace L4Re {
00015
00038 class Inhibitor :
00039     public L4::Kobject<Inhibitor, L4::Kobject, L4RE_PROTO_INHIBITOR>
00040 {
00041 public:
00042     enum
00043     {
00044         Name_max = 20
00045     };
00046
00057 L4_INLINE_RPC(l4_ret_t, acquire, (l4_umword_t id, L4::Ipc::String<> reason));
00058
00067 L4_INLINE_RPC(l4_ret_t, release, (l4_umword_t id));
00068
00084 l4_ret_t next_lock_info(char *name, unsigned len, l4_mword_t current_id = -1,
00085                        l4_utcb_t *utcb = l4_utcb())
00086 {
00087     L4::Ipc::String<char> name_buf(len, name);
00088     l4_ret_t r = next_lock_info_t::call(c(), &current_id, name_buf, utcb);
00089     if (r < 0)
00090         return r;
00091
00092     return current_id;
00093 }
00094
00095 L4_INLINE_RPC_NF(l4_ret_t, next_lock_info, (L4::Ipc::In_out<l4_mword_t *> current_id,
00096                                             L4::Ipc::String<char> &name));
00097
00098 typedef L4::Typeid::Rpcs<acquire_t, release_t, next_lock_info_t> Rpcs;
00099 };
00100
00101 }

```

16.346 inhibitor-sys.h

```

00001 /*
00002  * (c) 2014 Steffen Liebergeld <steffen.liebergeld@kernkonzept.com>
00003  *

```

```

00004  * License: see LICENSE.spdx (in this directory or the directories above)
00005  */
00006  #pragma once
00007
00008  namespace L4Re {
00009      namespace Inhibitor_ {
00015          enum Opcodes { Acquire, Release, Next_lock_info };
00016      }
00017  }

```

16.347 itas

```

00001  // vi:set ft=cpp: -*- Mode: C++ -*-
00002  /*
00003   * Copyright (C) 2025 Kernkonzept GmbH.
00004   * Author(s): Jan Klötzke <jan.kloetzke@kernkonzept.com>
00005   *
00006   * License: see LICENSE.spdx (in this directory or the directories above)
00007   */
00008
00009  #pragma once
00010
00011  #include <l4/sys/cxx/ipc_iface>
00012  #include <l4/sys/cxx/ipc_types>
00013  #include <l4/sys/cxx/types>
00014  #include <l4/sys/l4int.h>
00015  #include <l4/sys/thread>
00016  #include <l4/sys/types.h>
00017
00018  #include <signal.h>
00019  #include <sys/time.h>
00020
00021  namespace L4Re
00022  {
00023
00030  class L4_EXPORT Itas :
00031      public L4::Kobject_t<Itas, L4::Kobject,
00032                          L4RE_PROTO_ITAS,
00033                          L4::Type_info::Demand_t<2> >
00034  {
00035  public:
00048      L4_INLINE_RPC(int, register_thread, (L4::Ipc::Cap<L4::Thread> parent,
00049                                          L4::Ipc::Cap<L4::Thread> thread_cap,
00050                                          l4_addr_t thread_utcb));
00051
00060      L4_INLINE_RPC(int, unregister_thread, (L4::Ipc::Cap<L4::Thread> thread));
00061
00062      // sigaction.sa_flags is usually `unsigned long`, except for MIPS...
00063      enum : unsigned
00064      {
00065          Ignore_sigaction = ~0U
00066      };
00067
00079      L4_INLINE_RPC(int, sigaction, (int signum,
00080                                   const struct sigaction *act,
00081                                   struct sigaction *oldact));
00082
00095      L4_INLINE_RPC(int, sigaltstack, (L4::Ipc::Cap<L4::Thread> thread,
00096                                      const struct sigaltstack *ss,
00097                                      struct sigaltstack *oss));
00098
00113      L4_INLINE_RPC(int, sigprocmask, (L4::Ipc::Cap<L4::Thread> thread,
00114                                      int how, sigset_t const *set,
00115                                      sigset_t *oldset));
00116
00126      L4_INLINE_RPC(int, sigpending, (L4::Ipc::Cap<L4::Thread> thread,
00127                                     sigset_t *set));
00128
00138      L4_INLINE_RPC(int, setitimer, (int which,
00139                                    const struct itimerval *new_value,
00140                                    struct itimerval *old_value));
00141
00150      L4_INLINE_RPC(int, getitimer, (int which, struct itimerval *curr_value));
00151
00158      L4_INLINE_RPC(int, raise, (L4::Ipc::Cap<L4::Thread> thread, int sig));
00159
00160      typedef L4::Typeid::Rpc<
00161          register_thread_t, unregister_thread_t, sigaction_t, sigaltstack_t,
00162          sigprocmask_t, sigpending_t, setitimer_t, getitimer_t, raise_t
00163      > Rpc<
00164      >;
00165
00166  }

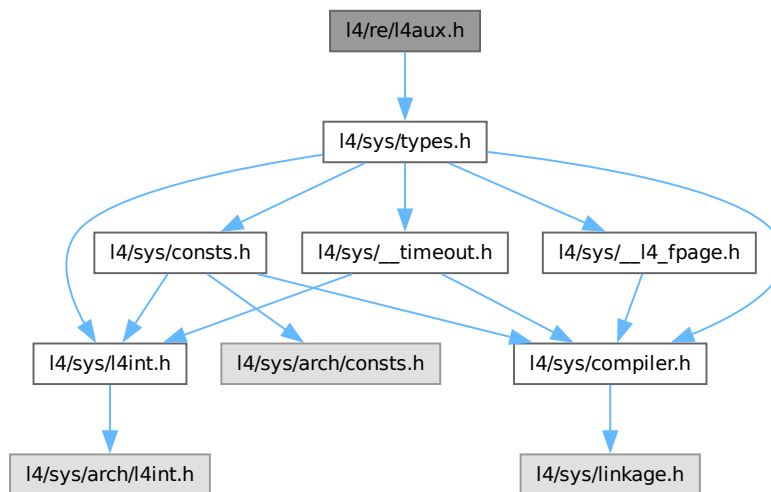
```

16.348 l4/re/l4aux.h File Reference

Auxiliary definitions.

```
#include <l4/sys/types.h>
```

Include dependency graph for l4aux.h:



Data Structures

- struct [l4re_aux_t](#)
Auxiliary descriptor.

Typedefs

- typedef struct l4re_aux_t [l4re_aux_t](#)
Auxiliary descriptor.

Enumerations

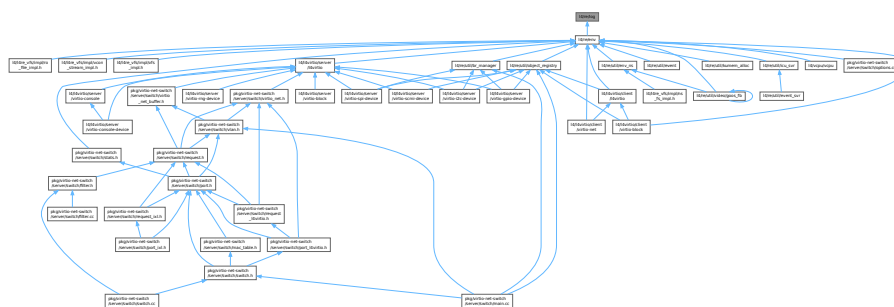
- enum [l4re_aux_ldr_flags_t](#)
Flags for program loading.

16.348.1 Detailed Description

Auxiliary definitions.

Definition in file [l4aux.h](#).

This graph shows which files directly or indirectly include this file:



Data Structures

- class [L4Re::Log](#)
Log interface class.

Namespaces

- namespace [L4Re](#)
L4Re C++ Interfaces.

16.350.1 Detailed Description

Log interface.

Definition in file [log](#).

16.351 log

[Go to the documentation of this file.](#)

```
00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014 #pragma once
00015
00016 #include <l4/sys/vcon>
00017
00018 namespace L4Re {
00019
00033 class L4_EXPORT Log : public L4::Kobject_t<Log, L4::Vcon, L4::PROTO_EMPTY>
00034 {
00035 public:
00036
00043     void println(char const *string, int len) const noexcept;
00044
00050     void print(char const *string) const noexcept;
00051 };
00052 }
```

16.352 l4/re/log-sys.h File Reference

Log protocol definition.

Namespaces

- namespace [L4Re](#)
[L4Re](#) C++ Interfaces.

Enumerations

- enum [L4Re::Log_::Opcodes](#)
Logging-service communication-protocol opcodes.

16.352.1 Detailed Description

Log protocol definition.

Definition in file [log-sys.h](#).

16.353 log-sys.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #pragma once
00013
00014 namespace L4Re
00015 {
00016     namespace Log_
00017     {
00023         enum Opcodes { Print };
00024     };
00025 };

```

16.354 l4/re/mem_alloc File Reference

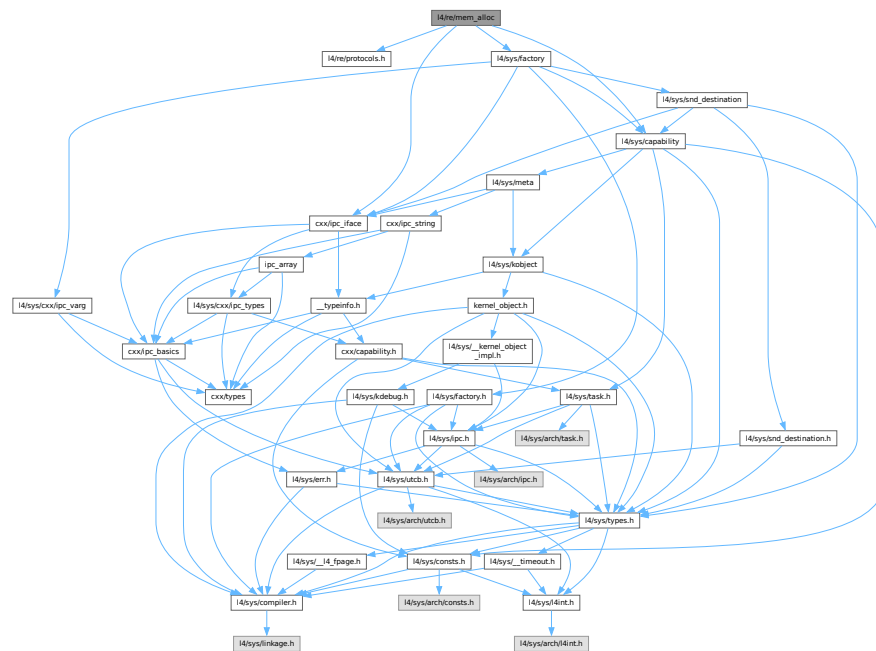
Memory allocator interface.

```

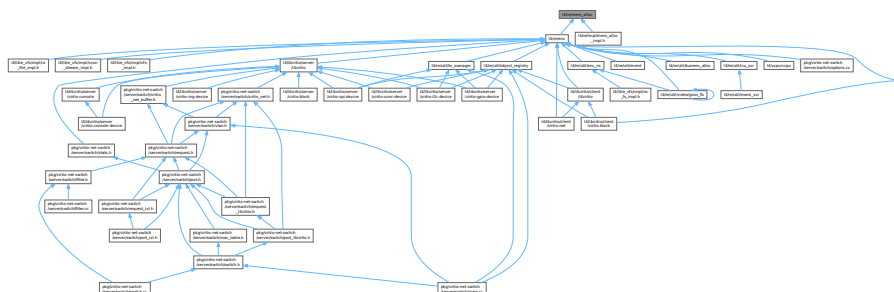
#include <l4/re/protocols.h>
#include <l4/sys/capability>
#include <l4/sys/cxx/ipc_iface>

```

Include dependency graph for mem_alloc:



This graph shows which files directly or indirectly include this file:



Data Structures

- class `L4Re::Mem_alloc`
Memory allocation interface.
- struct `L4Re::Mem_alloc::Stats`
Statistics about memory-allocator.

Namespaces

- namespace **L4Re**
L4Re C++ Interfaces.

16.354.1 Detailed Description

Memory allocator interface.

Definition in file [mem_alloc](#).

16.355 mem_alloc

[Go to the documentation of this file.](#)

```

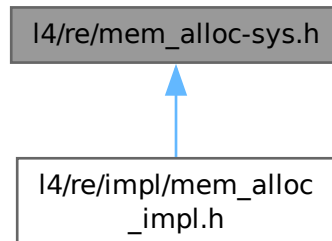
00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008  * Copyright (C) 2014-2016, 2019, 2021, 2024-2025 Kernkonzept GmbH.
00009  * Author(s): Alexander Warg <alexander.warg@kernkonzept.com>
00010  */
00011 /*
00012  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00013  *           Alexander Warg <warg@os.inf.tu-dresden.de>,
00014  *           Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00015  *           economic rights: Technische Universität Dresden (Germany)
00016  *
00017  * License: see LICENSE.spdx (in this directory or the directories above)
00018  */
00019 #pragma once
00020
00021 #include <l4/re/protocols.h>
00022 #include <l4/sys/capability>
00023 #include <l4/sys/cxx/ipc_iface>
00024 #include <l4/sys/factory>
00025
00026 namespace L4Re {
00027 class Dataspace;
00028
00029 // MISSING:
00030 // * alignment constraints
00031 // * shall we support superpages in noncont memory?
00032
00052 class L4_EXPORT Mem_alloc :
00053 public L4::Kobject_t<Mem_alloc, L4::Factory, L4RE_PROTO_MEM_ALLOC>
00054 {
00055 public:
00062 enum Mem_alloc_flags
00063 {
00064     Continuous    = 0x01,
00065     Pinned        = 0x02,
00066     Super_pages   = 0x04,
00067     Fixed_paddr   = 0x08,
00069 };
00070
00074 struct Stats
00075 {
00083     l4_size_t quota;
00084
00094     l4_size_t quota_used;
00095
00102     l4_size_t mem_limit;
00103
00116     l4_size_t mem_used;
00117
00126     l4_size_t mem_free;
00127 };
00128
00157 l4_ret_t alloc(long size, L4::Cap<Dataspace> mem,
00158               unsigned long flags = 0, unsigned long align = 0,
00159               l4_addr_t paddr = 0) const noexcept;
00160
00169 L4_INLINE_RPC(long, info, (Stats &stats));
00170
00171 typedef L4::Typeid::Rpc<info_t> Rpc;
00172 };
00173
00174 };

```

16.356 l4/re/mem_alloc-sys.h File Reference

Memory allocator protocol definitions.

This graph shows which files directly or indirectly include this file:



Namespaces

- namespace [L4Re](#)
L4Re C++ Interfaces.

Enumerations

- enum [L4Re::Mem_alloc_::Opcodes](#)
Memory-allocator communication-protocol opcodes.

16.356.1 Detailed Description

Memory allocator protocol definitions.

Definition in file [mem_alloc-sys.h](#).

16.357 mem_alloc-sys.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #pragma once
00013
00014 namespace L4Re
00015 {
00016     namespace Mem_alloc_
00017     {
00023         enum Opcodes { Alloc, Free };
00024     };
00025 };
  
```


16.359 mmio_space

[Go to the documentation of this file.](#)

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00003 /*
00004  * Copyright (C) 2017-2018, 2022, 2024 Kernkonzept GmbH.
00005  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00013 #pragma once
00014
00015 #include <l4/re/protocols.h>
00016 #include <l4/sys/capability>
00017 #include <l4/sys/cxx/ipc_types>
00018 #include <l4/sys/cxx/ipc_iface>
00019
00020 namespace L4Re
00021 {
00022
00045 struct L4_EXPORT Mmio_space
00046 : public L4::Kobject_t<Mmio_space, L4::Kobject, L4RE_PROTO_MMIO_SPACE>
00047 {
00049     enum Access_width
00050     {
00051         Wd_8bit = 0,
00052         Wd_16bit = 1,
00053         Wd_32bit = 2,
00054         Wd_64bit = 3
00055     };
00056
00058     typedef l4_uint64_t Addr;
00059
00074     L4_INLINE_RPC(l4_ret_t, mmio_read, (Addr addr, char width, l4_uint64_t *value));
00075
00090     L4_INLINE_RPC(l4_ret_t, mmio_write, (Addr addr, char width, l4_uint64_t value));
00091
00092     typedef L4::Typeid::Rpc<mmio_read_t, mmio_write_t> Rpcs;
00093 };
00094
00095 }
```

16.360 l4/re/namespace File Reference

Namespace interface.

```

#include <l4/sys/capability>
#include <l4/re/protocols.h>
#include <l4/sys/cxx/ipc_iface>
#include <l4/sys/cxx/ipc_array>
#include <l4/sys/cxx/ipc_string>
```


16.360.1 Detailed Description

Namespace interface.

Definition in file [namespace](#).

16.361 namespace

[Go to the documentation of this file.](#)

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00003 /*
00008 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009 *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00010 *      Björn Döbel <doebel@os.inf.tu-dresden.de>
00011 *      economic rights: Technische Universität Dresden (Germany)
00012 *
00013 * License: see LICENSE.spdx (in this directory or the directories above)
00014 */
00015 #pragma once
00016
00017 #include <l4/sys/capability>
00018 #include <l4/re/protocols.h>
00019 #include <l4/sys/cxx/ipc_iface>
00020 #include <l4/sys/cxx/ipc_array>
00021 #include <l4/sys/cxx/ipc_string>
00022
00023 namespace L4Re {
00024
00049 class L4_EXPORT Namespace :
00050     public L4::Kobject_t<Namespace, L4::Kobject, L4RE_PROTO_NAMESPACE,
00051         L4::Type_info::Demand_t<1> >
00052 {
00053 public:
00057     enum Register_flags
00058     {
00059         Ro      = L4_CAP_FPAGE_RO,
00060         Rw      = L4_CAP_FPAGE_RW,
00061         Rs      = L4_CAP_FPAGE_RS,
00062         Rws     = L4_CAP_FPAGE_RWS,
00063         Strong  = L4_CAP_FPAGE_S,
00064         Trusted = 0x008,
00065
00066         Cap_flags = Ro | Rw | Strong | Trusted,
00067
00068         Link      = 0x100,
00069         Overwrite = 0x200,
00070     };
00071
00077     enum Query_result_flags
00078     {
00079         Partly_resolved = 0x020,
00080     };
00081
00083     enum Query_timeout
00084     {
00085         To_default      = 3600000,
00086         To_non_blocking = 0,
00087     };
00088
00089     L4_RPC_NF(
00090         l4_ret_t, query, (L4::Ipc::Array_ref<char const, unsigned long> name,
00091             L4::Ipc::Small_buf cap,
00092             L4::Ipc::Snd_fpage &snd_cap,
00093             L4::Ipc::Opt<L4::Opcode &> dummy,
00094             L4::Ipc::Opt<
00095                 L4::Ipc::Array_ref<char const, unsigned long> &> out_name));
00096
00122     l4_ret_t query(char const *name, L4::Cap<void> const &cap,
00123         int timeout = To_default,
00124         l4_umword_t *local_id = 0, bool iterate = true) const noexcept;
00125
00135     l4_ret_t query(char const *name, unsigned len, L4::Cap<void> const &cap,
00136         int timeout = To_default,
00137         l4_umword_t *local_id = 0, bool iterate = true) const noexcept;
00138
00139     L4_RPC_NF(l4_ret_t, register_obj, (unsigned flags,
00140         L4::Ipc::Array<char const, unsigned long> name,
```

```

00141         L4::Ip<::Opt< L4::Ip<::Cap<void> > obj),
00142         L4::Ip<::Call_t<L4_CAP_FPAGE_W>);
00143
00167  l4_ret_t register_obj(char const *name, L4::Ip<::Cap<void> obj,
00168                        unsigned flags = Rw) const noexcept
00169  {
00170      return register_obj_t::call(c(), flags,
00171                                L4::Ip<::Array<char const, unsigned long>(
00172                                    __builtin_strlen(name), name),
00173                                obj);
00174  }
00175
00176  L4_RPC_NF_OP(3, // backward compatibility opcode
00177              l4_ret_t, unlink, (L4::Ip<::Array<char const, unsigned long> name),
00178              L4::Ip<::Call_t<L4_CAP_FPAGE_W>);
00179
00194  l4_ret_t unlink(char const* name)
00195  {
00196      return unlink_t::call(c(), L4::Ip<::Array<char const, unsigned long>(
00197                              __builtin_strlen(name), name));
00198  }
00199
00200  typedef L4::Typeid::Rpc<query_t, register_obj_t, unlink_t> Rpc;
00201
00202 private:
00203  l4_ret_t _query(char const *name, unsigned len,
00204                 L4::Cap<void> const &target, l4_umword_t *local_id,
00205                 bool iterate) const noexcept;
00206
00207 };
00208
00209 };

```

16.362 I4/re/namespace-sys.h File Reference

Namespace protocol definitions.

Namespaces

- namespace [L4Re](#)
L4Re C++ Interfaces.

Enumerations

- enum [L4Re::Namespace_::Opcodes](#)
Name-space communication-protocol opcodes.

16.362.1 Detailed Description

Namespace protocol definitions.

Definition in file [namespace-sys.h](#).

16.363 namespace-sys.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #pragma once
00013
00014 namespace L4Re {
00015     namespace Namespace_
00016     {
00022         enum Opcodes { Query, Register, Link, Unlink };
00023     };
00024 };

```

16.364 I4/re/parent File Reference

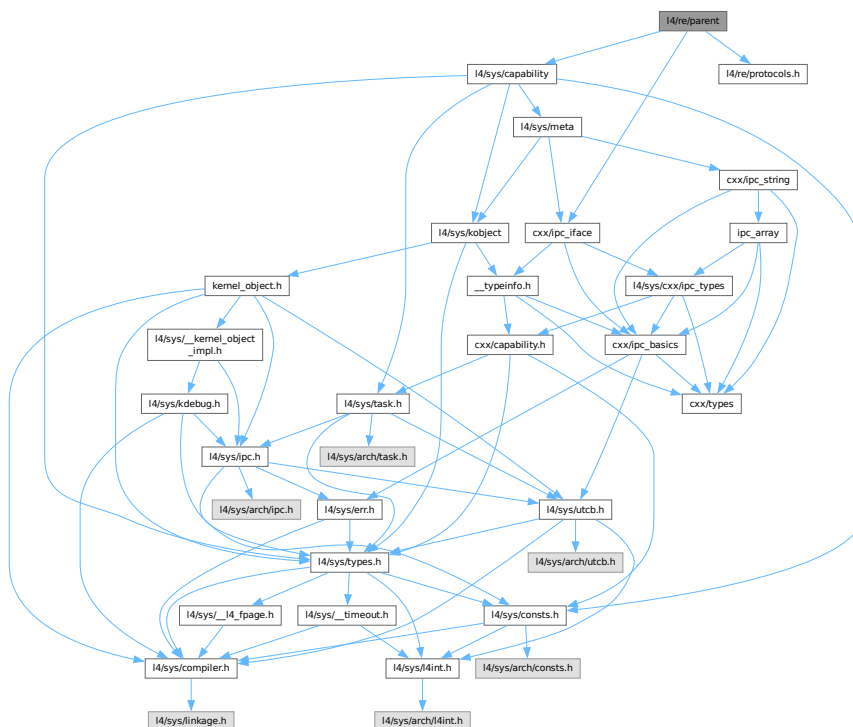
Parent interface.

```

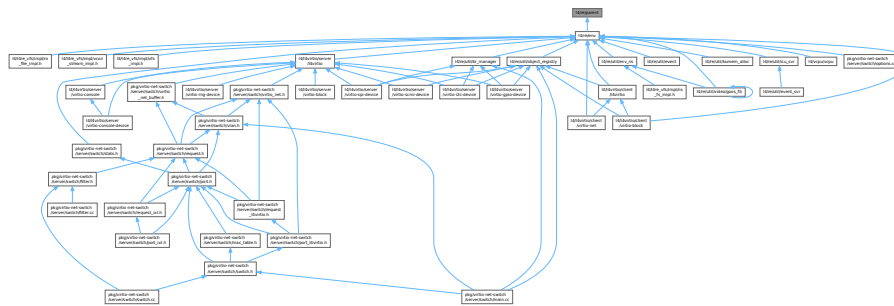
#include <l4/sys/capability>
#include <l4/re/protocols.h>
#include <l4/sys/cxx/ipc_iface>

```

Include dependency graph for parent:



This graph shows which files directly or indirectly include this file:



Data Structures

- class [L4Re::Parent](#)
Parent interface.

Namespaces

- namespace [L4Re](#)
L4Re C++ Interfaces.

16.364.1 Detailed Description

Parent interface.

Definition in file [parent](#).

16.365 parent

[Go to the documentation of this file.](#)

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014 #pragma once
00015
00016 #include <l4/sys/capability>
00017 #include <l4/re/protocols.h>
00018 #include <l4/sys/cxx/ipc_iface>
00019
00020 namespace L4Re {
00021
00034
00042 class L4_EXPORT Parent :
00043     public L4::Kobject_t<Parent, L4::Kobject, L4RE_PROTO_PARENT>
00044 {
00045 public:
00061     L4_INLINE_RPC(l4_ret_t, signal, (unsigned long sig, unsigned long val));
00062     typedef L4::Typeid::Rpc<signal_t> Rpc;
00063 };
00064 };
00065

```

16.366 l4/re/parent-sys.h File Reference

Parent protocol definition.

Namespaces

- namespace [L4Re](#)
L4Re C++ Interfaces.

Enumerations

- enum [L4Re::Parent_::Opcodes](#)
Parent communication-protocol opcodes.

16.366.1 Detailed Description

Parent protocol definition.

Definition in file [parent-sys.h](#).

16.367 parent-sys.h

[Go to the documentation of this file.](#)

```

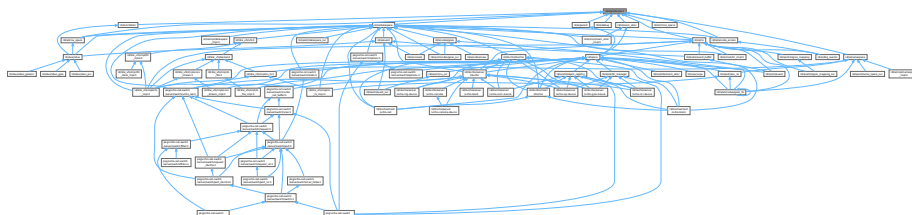
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #pragma once
00013
00014 namespace L4Re
00015 {
00016     namespace Parent_
00017     {
00023         enum Opcodes { Signal };
00024     };
00025 };

```

16.368 l4/re/protocols.h File Reference

[L4Re](#) Protocol Constants (C version).

This graph shows which files directly or indirectly include this file:



Enumerations

```
enum L4re_protocols {
    L4RE_PROTO_DATASPACE = 0x4000 , L4RE_PROTO_NAMESPACE , L4RE_PROTO_PARENT ,
    L4RE_PROTO_GOOS ,
    L4RE_PROTO_RSVD_1 , L4RE_PROTO_RM , L4RE_PROTO_EVENT , L4RE_PROTO_INHIBITOR ,
    L4RE_PROTO_DMA_SPACE , L4RE_PROTO_MMIO_SPACE , L4RE_PROTO_ITAS , L4RE_PROTO_MEM_ALLOC
    ,
    L4RE_PROTO_REMOTE_ACCESS , L4RE_PROTO_DEBUG = ~0x7fffL }

```

Common *L4Re* Protocol Constants.

16.368.1 Detailed Description

L4Re Protocol Constants (C version).

Definition in file [protocols.h](#).

16.369 protocols.h

[Go to the documentation of this file.](#)

```
00001
00005
00006 /*
00007  * (c) 2015 Alexander Warg <alexander.warg@kernkonzept.com>
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011
00012 #pragma once
00013
00019
00024 enum L4re_protocols
00025 {
00026     L4RE_PROTO_DATASPACE = 0x4000,
00027     L4RE_PROTO_NAMESPACE,
00028     L4RE_PROTO_PARENT,
00029     L4RE_PROTO_GOOS,
00030     L4RE_PROTO_RSVD_1,
00031     L4RE_PROTO_RM,
00032     L4RE_PROTO_EVENT,
00033     L4RE_PROTO_INHIBITOR,
00034     L4RE_PROTO_DMA_SPACE,
00035     L4RE_PROTO_MMIO_SPACE,
00036     L4RE_PROTO_ITAS,
00037     L4RE_PROTO_MEM_ALLOC,
00038     L4RE_PROTO_REMOTE_ACCESS,
00039
00040     L4RE_PROTO_DEBUG = ~0x7fffL
00041 };
00042

```

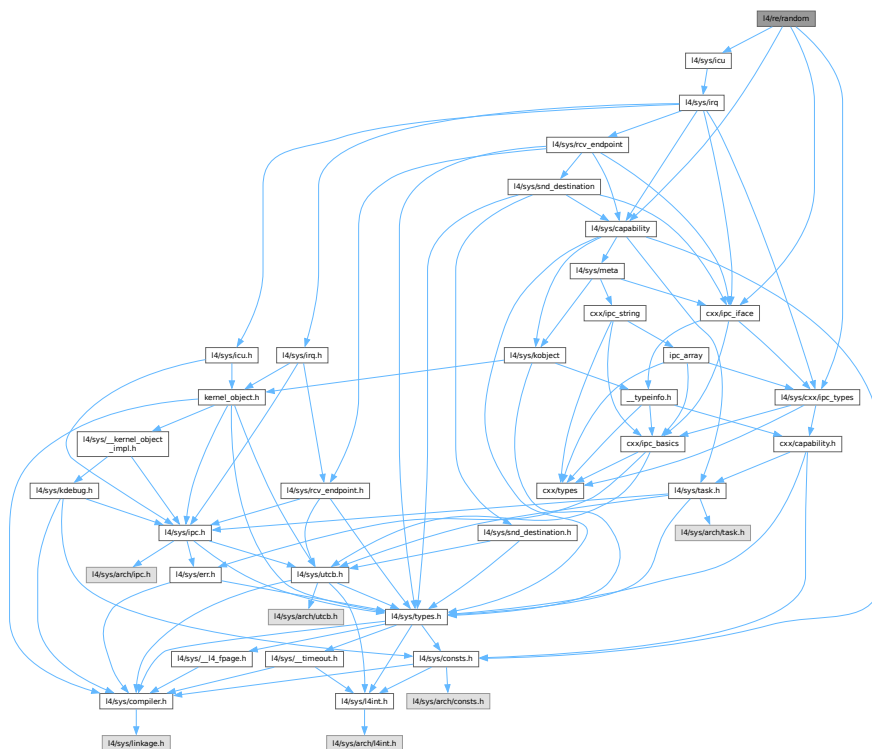
16.370 l4/re/random File Reference

Random number generator interface definition.

```
#include <l4/sys/capability>
#include <l4/sys/cxx/ipc_types>
#include <l4/sys/cxx/ipc_iface>

```

Include dependency graph for random:



- struct L4Re::Random

Low-bandwidth interface for random number generators.

- namespace **L4Re**

L4Re C++ Interfaces.

16.370.1 Detailed Description

Random number generator interface definition.

Definition in file [random](#).

16.371 random

[Go to the documentation of this file.](#)

```

00001 // -*- Mode: C++ -*-
00002 /*
00003  * Copyright (C) 2019-2020, 2022, 2024 Kernkonzept GmbH.
00004  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00012 #pragma once
00013
00014 #include <l4/sys/capability>
00015 #include <l4/sys/cxx/ipc_types>
00016 #include <l4/sys/cxx/ipc_iface>
00017 #include <l4/sys/icu>
00018
00019 namespace L4Re
00020 {
00021
00033 struct L4_EXPORT Random
00034 : public L4::Kobject_t<Random, L4::Icu>
00035 {
00060     L4_INLINE_RPC(long, get_random, (l4_size_t size,
00061                                     L4::Ipc::Array<char, unsigned long> *buffer));
00062
00063     typedef L4::Typeid::Rpcs<get_random_t> Rpcs;
00064 };
00065
00066 } // namespace

```

16.372 remote_access

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * Copyright (C) 2025 Adam Lackorzynski <adam@l4re.org>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00008 #pragma once
00009
00010 #include <l4/re/protocols.h>
00011 #include <l4/re/dataspace>
00012 #include <l4/sys/cxx/ipc_iface>
00013 #include <l4/sys/cxx/ipc_types>
00014 #include <l4/sys/cxx/types>
00015 #include <l4/sys/l4int.h>
00016
00017 namespace L4Re
00018 {
00019
00020 struct L4_EXPORT Remote_access
00021 : public L4::Kobject_t<Remote_access, Dataspace, L4RE_PROTO_REMOTE_ACCESS>
00022 {
00023     enum Access_width
00024     {
00025         Wd_8bit = 0,
00026         Wd_16bit = 1,
00027         Wd_32bit = 2,
00028         Wd_64bit = 3
00029     };
00030
00031     L4_INLINE_RPC(long, read_mem, (l4_addr_t addr, char width, l4_uint64_t *value));
00032     L4_INLINE_RPC(long, write_mem, (l4_addr_t addr, char width, l4_uint64_t value));
00033
00034     //L4_INLINE_RPC(long, get_regs, (unsigned TBD_THREAD_ID, l4_exc_regs_t *regs));
00035     //L4_INLINE_RPC(long, set_regs, (unsigned TBD_THREAD_ID, l4_exc_regs_t regs));
00036
00037     // !!! sizeof(l4_fpu_regs_t) > sizeof(utcb)
00038     //L4_INLINE_RPC(long, get_fpu_regs, (l4_fpu_regs_t *regs));
00039     //L4_INLINE_RPC(long, set_fpu_regs, (l4_fpu_regs_t regs));
00040
00041     L4_INLINE_RPC(long, terminate, (int exit_code), L4::Ipc::Send_only);
00042
00043     typedef L4::Typeid::Rpcs<read_mem_t, write_mem_t, terminate_t> Rpcs;
00044 };
00045
00046 }

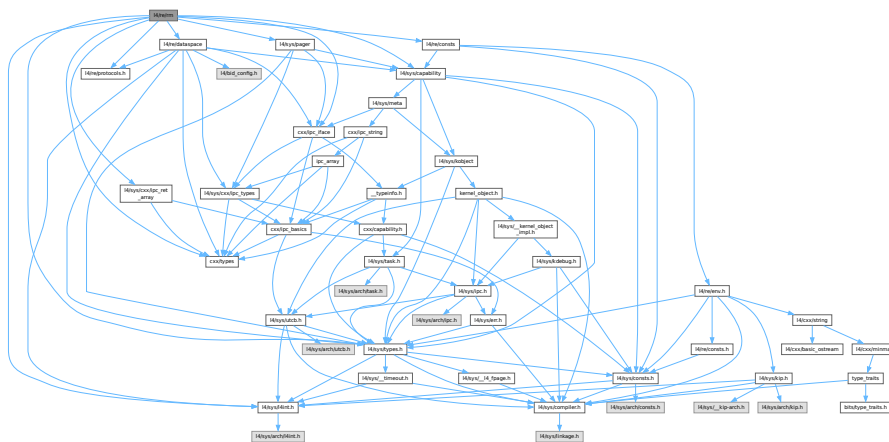
```


16.373 l4/re/rm File Reference

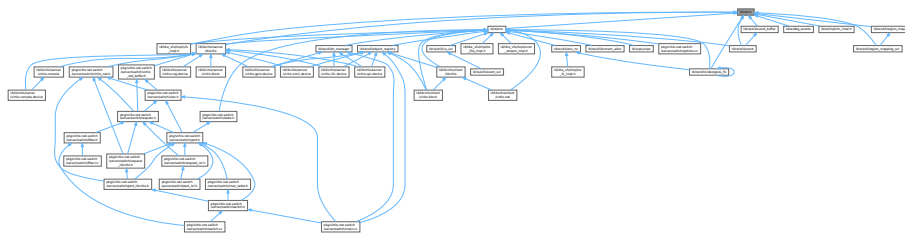
Region mapper interface.

```
#include <l4/sys/types.h>
#include <l4/sys/l4int.h>
#include <l4/sys/capability>
#include <l4/re/protocols.h>
#include <l4/sys/pager>
#include <l4/sys/cxx/ipc_iface>
#include <l4/sys/cxx/ipc_ret_array>
#include <l4/sys/cxx/types>
#include <l4/re/consts>
#include <l4/re/dataspace>
```

Include dependency graph for rm:



This graph shows which files directly or indirectly include this file:



Data Structures

- class [L4Re::Rm](#)
Region map.
- struct [L4Re::Rm::F](#)
Rm flags definitions.
- class [L4Re::Rm::Unique_region< T >](#)
Unique region.
- struct [L4Re::Rm::Region](#)
A region is a range of virtual addresses which is backed by content.
- struct [L4Re::Rm::Area](#)
An area is a range of virtual addresses which is reserved, see [L4Re::Rm::reserve_area\(\)](#).

Namespaces

- namespace [L4Re](#)
[L4Re C++ Interfaces](#).

16.373.1 Detailed Description

Region mapper interface.

Definition in file [rm](#).

16.374 rm

[Go to the documentation of this file.](#)

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *           Alexander Warg <warg@os.inf.tu-dresden.de>,
00010  *           Björn Döbel <doebel@os.inf.tu-dresden.de>,
00011  *           Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00012  *           economic rights: Technische Universität Dresden (Germany)
00013  *
00014  * License: see LICENSE.spdx (in this directory or the directories above)
00015  */
00016 #pragma once
00017
00018 #include <l4/sys/types.h>
00019 #include <l4/sys/l4int.h>
00020 #include <l4/sys/capability>
00021 #include <l4/re/protocols.h>
00022 #include <l4/sys/pager>
00023 #include <l4/sys/cxx/ipc_iface>
00024 #include <l4/sys/cxx/ipc_ret_array>
00025 #include <l4/sys/cxx/types>
00026 #include <l4/re/consts>
00027 #include <l4/re/dataspace>
00028
00029 namespace L4Re {
00030
00073
00081 class L4_EXPORT Rm :
00082     public L4::Kobject_t<Rm, L4::Pager, L4RE_PROTO_RM,
00083         L4::Type_info::Demand_t<1> >
00084 {
00085 public:
00086     typedef L4Re::Dataspace::Offset Offset;
00087
00089     enum Detach_result
00090     {
00091         Detached_ds    = 0,
00092         Kept_ds        = 1,
00093         Split_ds       = 2,
00094         Detach_result_mask = 3,
00095
00096         Detach_again = 4,
00097     };
00098
00100     enum Region_flag_shifts
00101     {
00103         Caching_shift    = Dataspace::F::Caching_shift,
00104     };
00105
00107     struct F
00108     {
00110         enum Attach_flags : l4_uint32_t
00111         {
00113             Search_addr    = 0x20000,
00115             In_area        = 0x40000,
00117             Eager_map      = 0x80000,
00119             No_eager_map   = 0x100000,
00120
00122             Attach_mask    = 0x1f0000,

```

```

00123     };
00124
00125     friend void enum_bitops_enable(Attach_flags);
00126
00127     enum Region_flags : l4_uint16_t
00128     {
00129     {
00131         Rights_mask = 0x0f,
00133         R           = Dataspace::F::R,
00135         W           = Dataspace::F::W,
00137         X           = Dataspace::F::X,
00139         RW          = Dataspace::F::RW,
00141         RX          = Dataspace::F::RX,
00143         RWX         = Dataspace::F::RWX,
00144
00146         Kernel      = 0x100,
00148         Detach_free  = 0x200,
00151         Pager       = 0x400,
00153         Reserved    = 0x800,
00154
00155
00157         Caching_mask = Dataspace::F::Caching_mask,
00160         Cache_normal = Dataspace::F::Normal,
00162         Cache_buffered = Dataspace::F::Bufferable,
00164         Cache_uncached = Dataspace::F::Uncacheable,
00165
00167         Ds_map_mask  = 0xff,
00168
00170         Region_flags_mask = 0xffff,
00171     };
00172
00173     friend void enum_bitops_enable(Region_flags);
00174
00175     friend constexpr Dataspace::Flags map_flags(Region_flags rf)
00176     { return Dataspace::Flags(static_cast<l4_uint16_t>(rf) & Ds_map_mask); }
00177
00178     struct Flags : L4::Types::Flags_ops_t<Flags>
00179     {
00180         l4_uint32_t raw;
00181
00182         Flags() = default;
00183         explicit constexpr Flags(l4_uint32_t f) : raw(f) {}
00184         constexpr Flags(Attach_flags af) : raw(static_cast<l4_uint32_t>(af)) {}
00185         constexpr Flags(Region_flags rf) : raw(static_cast<l4_uint32_t>(rf)) {}
00186
00187         constexpr Dataspace::Flags map_flags() const
00188         { return Dataspace::Flags(raw & Ds_map_mask); }
00189
00190         constexpr Region_flags region_flags() const
00191         { return Region_flags(raw & Region_flags_mask); }
00192
00193         constexpr Attach_flags attach_flags() const
00194         { return Attach_flags(raw & Attach_mask); }
00195
00196         constexpr bool r() const { return raw & L4_FPAGE_RO; }
00197         constexpr bool w() const { return raw & L4_FPAGE_W; }
00198         constexpr bool x() const { return raw & L4_FPAGE_X; }
00199
00200         constexpr unsigned cap_rights() const
00201         { return w() ? L4_CAP_FPAGE_RW : L4_CAP_FPAGE_RO; }
00202     };
00203
00204     friend constexpr Flags operator | (Region_flags l, Attach_flags r)
00205     { return Flags(l) | Flags(r); }
00206
00207     friend constexpr Flags operator | (Attach_flags l, Region_flags r)
00208     { return Flags(l) | Flags(r); }
00209 };
00210
00211 using Attach_flags = F::Attach_flags;
00212 using Region_flags = F::Region_flags;
00213 using Flags = F::Flags;
00214
00216     enum Detach_flags
00217     {
00227         Detach_exact    = 1,
00228
00238         Detach_overlap  = 2,
00239
00247         Detach_keep    = 4,
00248     };
00249
00277     l4_ret_t reserve_area(l4_addr_t *start, unsigned long size,
00278                          Flags flags = Flags(0),
00279                          unsigned char align = L4_PAGESHIFT) const noexcept
00280     { return reserve_area_t::call(c(), start, size, flags, align); }
00281
00282     L4_RPC_NF(l4_ret_t, reserve_area, (L4::Ipc::In_out<l4_addr_t *> start,

```

```

00283             unsigned long size,
00284             Flags flags,
00285             unsigned char align));
00286
00304     template<typename T>
00305     l4_ret_t reserve_area(T **start, unsigned long size,
00306                          Flags flags = Flags(0),
00307                          unsigned char align = L4_PAGESHIFT) const noexcept
00308     {
00309         return reserve_area_t::call(c(), reinterpret_cast<l4_addr_t*>(start), size,
00310                                     flags, align);
00311     }
00312
00326     L4_RPC(l4_ret_t, free_area, (l4_addr_t addr));
00327
00328     L4_RPC_NF(l4_ret_t, attach, (L4::Ipc::In_out<l4_addr_t *> start,
00329                                unsigned long size, Flags flags,
00330                                L4::Ipc::Opt<L4::Ipc::Cap<Dataspace>> mem,
00331                                Offset offs, unsigned char align,
00332                                L4::Ipc::Opt<l4_cap_idx_t> client_cap,
00333                                L4::Ipc::String<> name, Offset backing_offset));
00334
00335     L4_RPC_NF(l4_ret_t, detach, (l4_addr_t addr, unsigned long size, unsigned flags,
00336                                 l4_addr_t &start, l4_addr_t &rsz,
00337                                 l4_cap_idx_t &mem_cap));
00338
00397     l4_ret_t attach(l4_addr_t *start, unsigned long size, Flags flags,
00398                   L4::Ipc::Cap<Dataspace> mem, Offset offs = 0,
00399                   unsigned char align = L4_PAGESHIFT,
00400                   L4::Cap<L4::Task> const task
00401                   = L4::Cap<L4::Task>::Invalid,
00402                   char const *name = nullptr,
00403                   Offset backing_offset = 0) const noexcept;
00404
00408     template<typename T>
00409     l4_ret_t attach(T **start, unsigned long size, Flags flags,
00410                   L4::Ipc::Cap<Dataspace> mem, Offset offs = 0,
00411                   unsigned char align = L4_PAGESHIFT,
00412                   L4::Cap<L4::Task> const task
00413                   = L4::Cap<L4::Task>::Invalid,
00414                   char const *name = nullptr,
00415                   Offset backing_offset = 0) const noexcept
00416     {
00417         union X { l4_addr_t a; T* t; };
00418         X *x = reinterpret_cast<X*>(start);
00419         return attach(&x->a, size, flags, mem, offs, align, task,
00420                     name, backing_offset);
00421     }
00422
00423     #if __cplusplus >= 201103L
00434     template<typename T>
00435     class Unique_region
00436     {
00437     private:
00438         T _addr;
00439         L4::Cap<Rm> _rm;
00440
00441     public:
00442         Unique_region(Unique_region const &) = delete;
00443         Unique_region &operator = (Unique_region const &) = delete;
00444
00448         Unique_region() noexcept
00449             : _addr(0), _rm(L4::Cap<Rm>::Invalid) {}
00450
00456         explicit Unique_region(T addr) noexcept
00457             : _addr(addr), _rm(L4::Cap<Rm>::Invalid) {}
00458
00465         Unique_region(T addr, L4::Cap<Rm> const &rm) noexcept
00466             : _addr(addr), _rm(rm) {}
00467
00473         Unique_region(Unique_region &&o) noexcept : _addr(o.get()), _rm(o._rm)
00474             { o.release(); }
00475
00481         Unique_region &operator = (Unique_region &&o) noexcept
00482         {
00483             if (&o != this)
00484             {
00485                 if (_rm.is_valid())
00486                     _rm->detach(reinterpret_cast<l4_addr_t>(_addr), 0);
00487                 _rm = o._rm;
00488                 _addr = o.release();
00489             }
00490             return *this;
00491         }
00492
00498         ~Unique_region() noexcept
00499         {

```

```

00500         if (_rm.is_valid())
00501             _rm->detach(reinterpret_cast<l4_addr_t>(_addr), 0);
00502     }
00503
00509     T get() const noexcept
00510     { return _addr; }
00511
00517     T release() noexcept
00518     {
00519         _rm = L4::Cap<Rm>::Invalid;
00520         return _addr;
00521     }
00522
00529     void reset(T addr, L4::Cap<Rm> const &rm) noexcept
00530     {
00531         if (_rm.is_valid())
00532             _rm->detach(l4_addr_t(_addr), 0);
00533
00534         _rm = rm;
00535         _addr = addr;
00536     }
00537
00541     void reset() noexcept
00542     { reset(0, L4::Cap<Rm>::Invalid); }
00543
00549     bool is_valid() const noexcept
00550     { return _rm.is_valid(); }
00551
00553     T operator * () const noexcept { return _addr; }
00554
00556     T operator -> () const noexcept { return _addr; }
00557
00558     explicit operator bool() const noexcept
00559     { return is_valid(); }
00560 };
00561
00562 template<typename T>
00563 l4_ret_t attach(Unique_region<T> *start, unsigned long size, Flags flags,
00564                L4::Ipc::Cap<Dataspace> mem, Offset offs = 0,
00565                unsigned char align = L4::PAGESHIFT,
00566                L4::Cap<L4::Task> const task
00567                = L4::Cap<L4::Task>::Invalid,
00568                char const *name = nullptr,
00569                Offset backing_offset = 0) const noexcept
00570 {
00571     l4_addr_t addr = reinterpret_cast<l4_addr_t>(start->get());
00572
00573     long res = attach(&addr, size, flags, mem, offs, align, task,
00574                     name, backing_offset);
00575     if (res < 0)
00576         return res;
00577
00578     start->reset(reinterpret_cast<T>(&addr), L4::Cap<Rm>(cap()));
00579     return res;
00580 }
00581 #endif
00582
00600 l4_ret_t detach(l4_addr_t addr, L4::Cap<Dataspace> *mem,
00601                L4::Cap<L4::Task> const &task = This_task) const noexcept;
00602
00606 l4_ret_t detach(void *addr, L4::Cap<Dataspace> *mem,
00607                L4::Cap<L4::Task> const &task = This_task) const noexcept;
00608
00628 l4_ret_t detach(l4_addr_t start, unsigned long size, L4::Cap<Dataspace> *mem,
00629                L4::Cap<L4::Task> const &task) const noexcept;
00630
00675 l4_ret_t find(l4_addr_t *addr, unsigned long *size, Offset *offset,
00676                L4Re::Rm::Flags *flags, L4::Cap<Dataspace> *m) noexcept
00677 { return find_t::call(c(), addr, size, flags, offset, m); }
00678
00679 L4_RPC_NF(l4_ret_t, find, (L4::Ipc::In_out<l4_addr_t *> addr,
00680                          L4::Ipc::In_out<unsigned long *> size,
00681                          L4Re::Rm::Flags *flags, Offset *offset,
00682                          L4::Ipc::As_value<L4::Cap<Dataspace>> *m));
00683
00689 struct Region
00690 {
00691     l4_addr_t start;
00692     l4_addr_t end;
00693     F::Region_flags flags;
00694 };
00695
00702 struct Area
00703 {
00704     l4_addr_t start;
00705     l4_addr_t end;
00706 };

```

```

00707
00722 L4_RPC(l4_ret_t, get_regions, (l4_addr_t start, L4::Ip::Ret_array<Region> regions));
00723
00738 L4_RPC(long, get_areas, (l4_addr_t start, L4::Ip::Ret_array<Area> areas));
00739
00753 L4_RPC(l4_ret_t, get_info, (l4_addr_t addr, L4::Ip::String<char> &name,
00754                               Offset &backing_offset));
00755
00756 l4_ret_t detach(l4_addr_t start, unsigned long size, L4::Cap<Dataspace> *mem,
00757                 L4::Cap<L4::Task> task, unsigned flags) const noexcept;
00758
00759 typedef L4::Typeid::Rpcs<attach_t, detach_t, find_t,
00760                         reserve_area_t, free_area_t,
00761                         get_regions_t, get_areas_t,
00762                         get_info_t> Rpcs;
00763 };
00764
00765 inline l4_ret_t
00766 Rm::detach(l4_addr_t addr, L4::Cap<Dataspace> *mem,
00767            L4::Cap<L4::Task> const &task) const noexcept
00768 { return detach(addr, 1, mem, task, Detach_overlap); }
00769
00770 inline l4_ret_t
00771 Rm::detach(void *addr, L4::Cap<Dataspace> *mem,
00772            L4::Cap<L4::Task> const &task) const noexcept
00773 {
00774     return detach(reinterpret_cast<l4_addr_t>(addr), 1, mem, task,
00775                  Detach_overlap);
00776 }
00777
00778 inline l4_ret_t
00779 Rm::detach(l4_addr_t addr, unsigned long size, L4::Cap<Dataspace> *mem,
00780            L4::Cap<L4::Task> const &task) const noexcept
00781 { return detach(addr, size, mem, task, Detach_exact); }
00782
00783 };

```

16.375 l4/re/rm-sys.h File Reference

Region mapper protocol definitions.

Namespaces

- namespace [L4Re](#)
L4Re C++ Interfaces.

Enumerations

- enum [L4Re::Rm_::Opcodes](#)
Region-map communication-protocol opcodes.

16.375.1 Detailed Description

Region mapper protocol definitions.

Definition in file [rm-sys.h](#).

Namespaces

- namespace [L4Re](#)
[L4Re](#) C++ Interfaces.
- namespace [L4Re::Util](#)
Documentation of the [L4](#) Runtime Environment utility functionality in C++.

16.377.1 Detailed Description

Bitmap capability allocator.

Definition in file [bitmap_cap_alloc](#).

16.378 bitmap_cap_alloc

[Go to the documentation of this file.](#)

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014
00015 #pragma once
00016
00017 #include <l4/re/util/item_alloc>
00018 #include <l4/sys/assert.h>
00019 #include <l4/sys/capability>
00020 #include <l4/sys/task.h>
00021
00022 namespace L4Re { namespace Util {
00023
00028 class Cap_alloc_base
00029 {
00030 private:
00031     long _bias;
00032     Item_alloc_base _items;
00033
00034 public:
00035     template <unsigned COUNT>
00036     struct Storage
00037     {
00038         typename Bitmap_base::Word<COUNT>::Type _bits[Bitmap_base::Word<COUNT>::Size];
00039     };
00040
00041     enum State { Free = 0, Allocated, Unknown };
00042     Cap_alloc_base(long max, void *mem, long bias = 0, void * = 0)
00043         noexcept : _bias(bias), _items(max, mem) {}
00044
00045     L4::Cap<void> alloc() noexcept
00046     {
00047         long cap = _items.alloc();
00048         if (cap < 0)
00049             return L4::Cap<void>::Invalid;
00050
00051         return L4::Cap<void>((cap + _bias) « L4_CAP_SHIFT);
00052     }
00053
00054     long hint() const { return _items.hint(); }
00055
00059     template< typename T >
00060     L4::Cap<T> alloc() noexcept
00061     { return L4::Cap<T>(alloc().cap()); }
00062
00063     State is_allocated(L4::Cap<void> c) const noexcept
00064     {
00065         long idx = (c.cap() » L4_CAP_SHIFT);
00066
00067         if (idx < _bias)

```



```

00068         return Unknown;
00069
00070         idx -= _bias;
00071         if (idx >= _items.size())
00072             return Unknown;
00073
00074         return _items.is_allocated(idx) ? Allocated : Free;
00075     }
00076
00077     template< typename T>
00081     void free(L4::Cap<T> const &cap, l4_cap_idx_t task = L4_INVALID_CAP,
00082              l4_umword_t unmap_flags = L4_FP_ALL_SPACES) noexcept
00083     {
00084         long idx = (cap.cap() >> L4_CAP_SHIFT);
00085         if (idx < _bias)
00086             return;
00087
00088         idx -= _bias;
00089         if (idx >= _items.size())
00090             return;
00091
00092         l4_assert(_items.is_allocated(idx));
00093
00094         if (l4_is_valid_cap(task))
00095             l4_task_unmap(task, cap.fpage(), unmap_flags | 2);
00096
00097         _items.free(idx);
00098     }
00099
00100     // since we have no counters assume counter always > 0
00101     void take(L4::Cap<void>) noexcept {}
00102     bool release(L4::Cap<void>, l4_cap_idx_t /*task*/ = L4_INVALID_CAP,
00103                 unsigned /*unmap_flags*/ = L4_FP_ALL_SPACES) noexcept
00104     { return false; }
00105
00106     long last() noexcept
00107     {
00108         return _items.size() + _bias - 1;
00109     }
00110 };
00111
00112 template< long Size >
00113 class Cap_alloc : public Cap_alloc_base
00114 {
00115 private:
00116     typename Bitmap_base::Word<Size>::Type _bits[Bitmap_base::Word<Size>::Size];
00117
00118 public:
00119     explicit Cap_alloc(long bias = 0) noexcept
00120         : Cap_alloc_base(Size, _bits, bias) {}
00121
00122 };
00123
00124 }
00125 }

```

16.379 br_manager

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00008 #pragma once
00009
00010 #include <l4/re/env>
00011 #include <l4/re/util/cap_alloc>
00012 #include <l4/sys/cxx/ipc_server_loop>
00013 #include <l4/cxx/ipc_timeout_queue>
00014 #include <l4/cxx/minmax>
00015 #include <l4/sys/assert.h>
00016
00017 namespace L4Re { namespace Util {
00018
00027 class Br_manager : public L4::Ipc_svr::Server_iface,
00028                   private L4::Ipc_svr::Server_iface::Mem_window::Allocator
00029 {
00030 private:
00031     enum { _ports = 0 };
00032     enum { Brs_per_timeout = sizeof(l4_kernel_clock_t) / sizeof(l4_umword_t) };
00033

```

```

00034 void dispose(l4_fpage_t fp) noexcept override
00035 {
00036     auto *env = L4Re::Env::env();
00037
00038     env->task()->unmap(fp, L4_FP_ALL_SPACES);
00039     env->rm()->free_area(l4_fpage_memaddr(fp));
00040 }
00041
00042 public:
00043 Br_manager()
00044 {
00045     _brs[0] = 0; // obj caps terminating entry
00046     _brs[1] = 0; // mem terminating entry
00047 }
00048
00049 Br_manager(Br_manager const &) = delete;
00050 Br_manager &operator = (Br_manager const &) = delete;
00051
00052 Br_manager(Br_manager &&) = delete;
00053 Br_manager &operator = (Br_manager &&) = delete;
00054
00055 ~Br_manager()
00056 {
00057     // Slots for received capabilities are placed at the beginning of the
00058     // (shadowed) buffer registers. Free those.
00059     for (unsigned i = 0; i < _caps; ++i)
00060         cap_alloc.free(L4::Cap<void>(_brs[i] & L4_CAP_MASK));
00061
00062     if (l4_is_fpage_valid(_mem_fp))
00063         dispose(_mem_fp);
00064 }
00065
00066 /*
00067  * This implementation dynamically manages assignment of buffer registers for
00068  * the necessary amount of receive buffers allocated by all calls to this
00069  * function.
00070  */
00071 int alloc_buffer_demand(Demand const &d) override
00072 {
00073     using L4::Ipc::Small_buf;
00074
00075     // IO port receive windows currently not supported
00076     if (d.ports)
00077         return -L4_EINVAL;
00078
00079     // Take extra buffers for a possible timeout and for two zero terminators
00080     // (caps + mem).
00081     if (cxx::max(d.caps, _caps) + 1 // obj BRs + terminator
00082         + (cxx::max(d.mem, _mem_order) ? 2 : 0) + 1 // mem BRs + terminator
00083         + Brs_per_timeout // timeout BR(s)
00084         > L4_UTCB_GENERIC_BUFFERS_SIZE)
00085         return -L4_ERANGE;
00086
00087     if (d.caps > _caps)
00088     {
00089         while (_caps < d.caps)
00090         {
00091             L4::Cap<void> cap = cap_alloc.alloc();
00092             if (!cap)
00093                 return -L4_ENOMEM;
00094
00095             reinterpret_cast<Small_buf*>(_brs[_caps])
00096                 = Small_buf(cap.cap(), _cap_flags);
00097             ++_caps;
00098         }
00099         _brs[_caps] = 0;
00100     }
00101
00102     if (d.mem > _mem_order)
00103     {
00104         l4_addr_t start = 0;
00105         int err = L4Re::Env::env()->rm()
00106             ->reserve_area(&start, 1UL << d.mem,
00107                 L4Re::Rm::F::Search_addr | L4Re::Rm::F::Reserved,
00108                 d.mem);
00109         if (err < 0)
00110             return -L4_ENOMEM;
00111
00112         if (l4_is_fpage_valid(_mem_fp))
00113             dispose(_mem_fp);
00114
00115         _mem_order = d.mem;
00116         _mem_fp = l4_fpage(start, _mem_order, L4_FPAGE_RWX);
00117     }
00118
00119     // Memory receive window is in BRs after object caps...
00120     if (l4_is_fpage_valid(_mem_fp))

```

```

00122     {
00123         L4::Ipc::Rcv_fpage rcv_fpage(_mem_fp);
00124         _brs[_caps + 1] = rcv_fpage.base_x();
00125         _brs[_caps + 2] = rcv_fpage.data();
00126         _brs[_caps + 3] = 0;
00127         _used_brs = _caps + 4;
00128     }
00129     else
00130     {
00131         _brs[_caps + 1] = 0;
00132         _used_brs = _caps + 2;
00133     }
00134
00135     return L4_EOK;
00136 }
00137
00138
00139 L4::Cap<void> get_rcv_cap(int i) const override
00140 {
00141     if (i < 0 || i >= _caps)
00142         return L4::Cap<void>::Invalid;
00143
00144     return L4::Cap<void>(_brs[i] & L4_CAP_MASK);
00145 }
00146
00147 int realloc_rcv_cap(int i) override
00148 {
00149     using L4::Ipc::Small_buf;
00150
00151     if (i < 0 || i >= _caps)
00152         return -L4_EINVAL;
00153
00154     L4::Cap<void> cap = cap_alloc.alloc();
00155     if (!cap)
00156         return -L4_ENOMEM;
00157
00158     reinterpret_cast<Small_buf*>(_brs[i])
00159         = Small_buf(cap.cap(), _cap_flags);
00160
00161     return L4_EOK;
00162 }
00163
00164 cxx::Result<L4::Ipc_svr::Server_iface::Mem_window>
00165 get_rcv_mem() noexcept override
00166 {
00167     if (!_mem_order)
00168         return cxx::Error(-L4_EINVAL);
00169
00170     l4_addr_t start = 0;
00171     int err = L4Re::Env::env()->rm()
00172         ->reserve_area(&start, 1UL << _mem_order,
00173             L4Re::Rm::F::Search_addr | L4Re::Rm::F::Reserved,
00174             _mem_order);
00175     if (err < 0)
00176     {
00177         // There might be pages mapped there already. Unmap them because the
00178         // caller cannot get hold of them any more.
00179         L4Re::Env::env()->task()->unmap(_mem_fp, L4_FP_ALL_SPACES);
00180         return cxx::Error(-L4_ENOMEM);
00181     }
00182
00183     l4_fpage_t ret = _mem_fp;
00184
00185     _mem_fp = l4_fpage(start, _mem_order, L4_FPAGE_RWX);
00186     L4::Ipc::Rcv_fpage rcv_fpage(_mem_fp);
00187     _brs[_caps + 1] = rcv_fpage.base_x();
00188     _brs[_caps + 2] = rcv_fpage.data();
00189
00190     return L4::Ipc_svr::Server_iface::Mem_window(ret, this);
00191 }
00192
00200 void set_rcv_cap_flags(unsigned long flags)
00201 {
00202     l4_assert(_caps == 0);
00203
00204     _cap_flags = flags;
00205 }
00206
00208 int add_timeout(L4::Ipc_svr::Timeout *, l4_kernel_clock_t) override
00209 { return -L4_ENOSYS; }
00210
00212 int remove_timeout(L4::Ipc_svr::Timeout *) override
00213 { return -L4_ENOSYS; }
00214
00216 void setup_wait(l4_utcb_t *utcb, L4::Ipc_svr::Reply_mode)
00217 {
00218     l4_buf_regs_t *br = l4_utcb_br_u(utcb);

```

```

00219     br->bdr = l4_bdr(_caps + 1, 0, 0, 0);
00220     for (unsigned i = 0; i < _used_brs; ++i)
00221         br->br[i] = _brs[i];
00222 }
00223
00224 protected:
00226 unsigned first_free_br() const
00227 {
00228     // The last BR (64-bit) or the last two BRs (32-bit); this is constant.
00229     return L4_UTCB_GENERIC_BUFFERS_SIZE - Brs_per_timeout;
00230     // We could also do the following dynamic approach:
00231     // return _caps + _mem + _ports + 1
00232 }
00233
00234 private:
00235     unsigned char _caps = 0;
00236     unsigned char _mem_order = 0;
00237     unsigned char _used_brs = 2; // always at least two terminating entries
00238     unsigned long _cap_flags = L4_RCV_ITEM_LOCAL_ID;
00239     l4_fpage_t _mem_fp = l4_fpage_invalid();
00240
00241     l4_umword_t _brs[L4_UTCB_GENERIC_BUFFERS_SIZE];
00242 };
00243
00250 struct Br_manager_hooks
00251 : L4::Ipc_svr::Ignore_errors,
00252   L4::Ipc_svr::Default_timeout,
00253   L4::Ipc_svr::Compound_reply,
00254   Br_manager
00255 {};
00256
00264 struct Br_manager_timeout_hooks :
00265     public L4::Ipc_svr::Timeout_queue_hooks<Br_manager_timeout_hooks, Br_manager>,
00266     public L4::Ipc_svr::Ignore_errors
00267 {
00268 public:
00269     static l4_kernel_clock_t now()
00270     { return l4_kip_clock(l4re_kip()); }
00271 };
00272
00273 {}
00274

```

16.380 l4/re/util/cap File Reference

Capability utility functions.

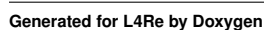

```

00021 namespace L4Re {
00022
00030 class Cap_alloc
00031 {
00032 public:
00033     constexpr Cap_alloc() = default;
00034
00035     // Attention: do *not* define a destructor. We must keep the class trivially
00036     // destructible so that no global destructor is created for
00037     // L4Re::Util::cap_alloc.
00038     // virtual ~Cap_alloc() = default;
00039
00040     Cap_alloc(Cap_alloc const &) = delete;
00041     Cap_alloc &operator = (Cap_alloc const &) = delete;
00042
00047     virtual L4::Cap<void> alloc() noexcept = 0;
00048     virtual void take(L4::Cap<void> cap) noexcept = 0;
00049
00054     template< typename T >
00055     L4::Cap<T> alloc() noexcept
00056     { return L4::cap_cast<T>(alloc()); }
00057
00064     virtual void free(L4::Cap<void> cap, l4_cap_idx_t task = L4_INVALID_CAP,
00065                     unsigned unmap_flags = L4_FP_ALL_SPACES) noexcept = 0;
00066     virtual bool release(L4::Cap<void> cap, l4_cap_idx_t task = L4_INVALID_CAP,
00067                        unsigned unmap_flags = L4_FP_ALL_SPACES) noexcept = 0;
00068 };
00069
00070 template<typename ALLOC>
00071 struct Cap_alloc_t : ALLOC, L4Re::Cap_alloc
00072 {
00073     template<typename ...ARGS>
00074     Cap_alloc_t(ARGS &&...args) : ALLOC(cxx::forward<ARGS>(args)...) {}
00075
00076     L4::Cap<void> alloc() noexcept override { return ALLOC::alloc(); }
00077     void take(L4::Cap<void> cap) noexcept override { ALLOC::take(cap); }
00078
00079     template <typename T>
00080     L4::Cap<T> alloc() noexcept
00081     {
00082         return L4::cap_cast<T>(alloc());
00083     }
00084
00085     void free(L4::Cap<void> cap, l4_cap_idx_t task = L4_INVALID_CAP,
00086             unsigned unmap_flags = L4_FP_ALL_SPACES) noexcept override
00087     { ALLOC::free(cap, task, unmap_flags); }
00088
00089     bool release(L4::Cap<void> cap, l4_cap_idx_t task,
00090                unsigned unmap_flags) noexcept override
00091     { return ALLOC::release(cap, task, unmap_flags); }
00092
00093     void operator delete(void *) {}
00094 };
00095
00096 extern Cap_alloc *virt_cap_alloc;
00097
00102 template< unsigned long Unmap_flags = L4_FP_ALL_SPACES >
00103 class Smart_cap_auto
00104 {
00105 private:
00106     Cap_alloc *_ca;
00107
00108 public:
00109     Smart_cap_auto() : _ca(0) {}
00110     Smart_cap_auto(Cap_alloc *ca) : _ca(ca) {}
00111
00112     void free(L4::Cap_base &c)
00113     {
00114         if (c.is_valid() && _ca)
00115             _ca->free(L4::Cap<void>(c.cap()), This_task, Unmap_flags);
00116
00117         invalidate(c);
00118     }
00119
00120     static void invalidate(L4::Cap_base &c)
00121     {
00122         if (c.is_valid())
00123             c.invalidate();
00124     }
00125 };
00126
00127
00131 template< unsigned long Unmap_flags = L4_FP_ALL_SPACES >
00132 class Smart_count_cap
00133 {
00134 private:
00135     Cap_alloc *_ca;

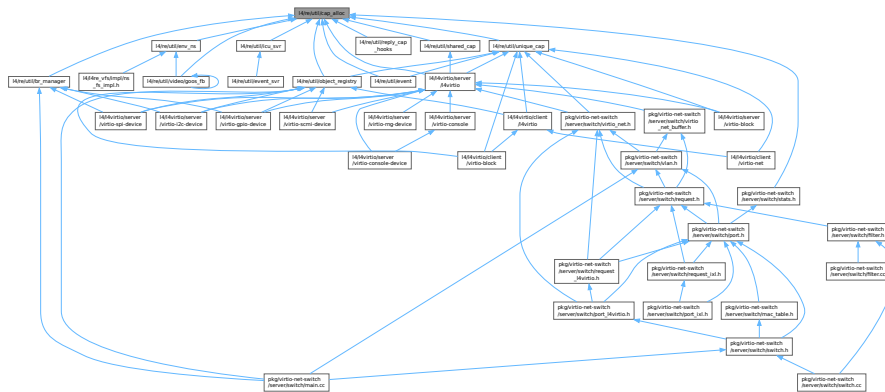
```


16.384 I4/re/util/cap_alloc File Reference

```
#include <l4/re/util/cap_alloc_impl.h>
#include <l4/re/util/item_alloc>
#include <l4/sys/smart_capability>
#include <l4/sys/task>
#include <l4/re/consts>
Include dependency graph for cap_alloc:
```



This graph shows which files directly or indirectly include this file:



Data Structures

- class `L4Re::Util::Smart_cap_auto< Unmap_flags >`
Helper for *Unique_cap* and *Unique_del_cap*.
- class `L4Re::Util::Smart_count_cap< Unmap_flags >`
Helper for *Ref_cap* and *Ref_del_cap*.
- struct `L4Re::Util::Ref_cap< T >`
Automatic capability that implements automatic free and unmap of the capability selector.
- struct `L4Re::Util::Ref_del_cap< T >`
Automatic capability that implements automatic free and unmap+delete of the capability selector.

Namespaces

- namespace `L4Re`
L4Re C++ Interfaces.
- namespace `L4Re::Util`
Documentation of the *L4 Runtime Environment* utility functionality in C++.

Functions

- template<typename T>
`Ref_cap< T >::Cap L4Re::Util::make_ref_cap ()`
Allocate a capability slot and wrap it in a *Ref_cap*.
- template<typename T>
`Ref_del_cap< T >::Cap L4Re::Util::make_ref_del_cap ()`
Allocate a capability slot and wrap it in a *Ref_del_cap*.

Variables

- `_Cap_alloc L4Re::Util::cap_alloc`
Capability allocator.
- `Def_reply_cap_alloc L4Re::Util::reply_cap_alloc`
Reply capability allocator.

16.384.1 Detailed Description

Capability allocator.

Definition in file [cap_alloc](#).

16.385 cap_alloc

[Go to the documentation of this file.](#)

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014
00015 #pragma once
00016
00017 #include <l4/re/util/cap_alloc_impl.h>
00018 #include <l4/re/util/item_alloc>
00019 #include <l4/sys/smart_capability>
00020 #include <l4/sys/task>
00021 #include <l4/re/consts>
00022
00023 namespace L4Re { namespace Util {
00024
00030
00031 template<unsigned Num_caps>
00032 class Reply_cap_alloc final : public L4::Reply_cap_alloc
00033 {
00034 public:
00035     static constexpr unsigned Capacity = Num_caps;
00036
00037     explicit Reply_cap_alloc(unsigned bias) noexcept
00038     : _bias(bias)
00039     {}
00040
00041     L4::Reply_cap_alloc() noexcept override
00042     {
00043         if (long slot = _alloc.alloc(); slot >= 0) [[likely]]
00044         {
00045             auto idx = (static_cast<l4_cap_idx_t>(slot) + _bias) « L4_CAP_SHIFT;
00046             return L4::Reply_cap(L4::Reply_cap_idx(idx), this);
00047         }
00048         else
00049             return L4::Reply_cap();
00050     }
00051
00052 protected:
00053     void free(L4::Reply_cap_idx cap) noexcept override
00054     { _alloc.free((cap.cap() » L4_CAP_SHIFT) - _bias); }
00055
00056 private:
00057     unsigned _bias;
00058     Item_alloc<Capacity> _alloc;
00059 };
00060
00061 using Def_reply_cap_alloc = Reply_cap_alloc<CONFIG_L4RE_CAP_DFL_ALLOCATOR_MAX>;
00062
00073 extern _Cap_alloc cap_alloc;
00074
00081 extern Def_reply_cap_alloc reply_cap_alloc;
00082
00086 template< unsigned long Unmap_flags = L4_FP_ALL_SPACES >
00087 class Smart_cap_auto
00088 {
00089 public:
00093     static void free(L4::Cap_base &c)
00094     {
00095         if (c.is_valid())
00096         {
00097             cap_alloc.free(L4::Cap<void>(c.cap()), This_task, Unmap_flags);
00098             c.invalidate();
00099         }
00100     }

```

```

00101
00105     static void invalidate(L4::Cap_base &c)
00106     {
00107         if (c.is_valid())
00108             c.invalidate();
00109     }
00110
00111 };
00112
00113
00117 template< unsigned long Unmap_flags = L4_FP_ALL_SPACES >
00118 class Smart_count_cap
00119 {
00120 public:
00125     static void free(L4::Cap_base &c) noexcept
00126     {
00127         if (c.is_valid())
00128         {
00129             if (cap_alloc.release(L4::Cap<void>(c.cap()), This_task, Unmap_flags))
00130                 c.invalidate();
00131         }
00132     }
00133
00137     static void invalidate(L4::Cap_base &c) noexcept
00138     {
00139         if (c.is_valid())
00140             c.invalidate();
00141     }
00142
00146     static L4::Cap_base copy(L4::Cap_base const &src)
00147     {
00148         cap_alloc.take(L4::Cap<void>(src.cap()));
00149         return src;
00150     }
00151 };
00152
00153
00183 template< typename T >
00184 struct Ref_cap
00185 {
00186     typedef L4::Smart_cap<T, Smart_count_cap<L4_FP_ALL_SPACES> > Cap;
00187 };
00188
00224 template< typename T >
00225 struct Ref_del_cap
00226 {
00227     typedef L4::Smart_cap<T, Smart_count_cap<L4_FP_DELETE_OBJ> > Cap;
00228 };
00229
00235 template< typename T >
00236 typename Ref_cap<T>::Cap
00237 make_ref_cap() { return typename Ref_cap<T>::Cap(cap_alloc.alloc<T>()); }
00238
00244 template< typename T >
00245 typename Ref_del_cap<T>::Cap
00246 make_ref_del_cap()
00247 { return typename Ref_del_cap<T>::Cap(cap_alloc.alloc<T>()); }
00248
00250
00251 }}
00252

```

16.386 l4/re/util/cap_alloc_impl.h File Reference

Capability allocator implementation.

```

#include <l4/bid_config.h>
#include <l4/re/cap_alloc>
#include <l4/re/util/counting_cap_alloc>
#include <l4/re/util/debug>

```

[illegible][illegible]

- class L4Re::Util::_Cap_alloc

Namespaces

- ## L4Re C++ Interfaces.

- Documentation of the [L4 Runtime Environment](#) utility functionality in C++.

16.386.1 Detailed Description

Capability allocator implementation.

Definition in file [cap_alloc_impl.h](#).

16.387 cap_alloc_impl.h

[Go to the documentation of this file.](#)

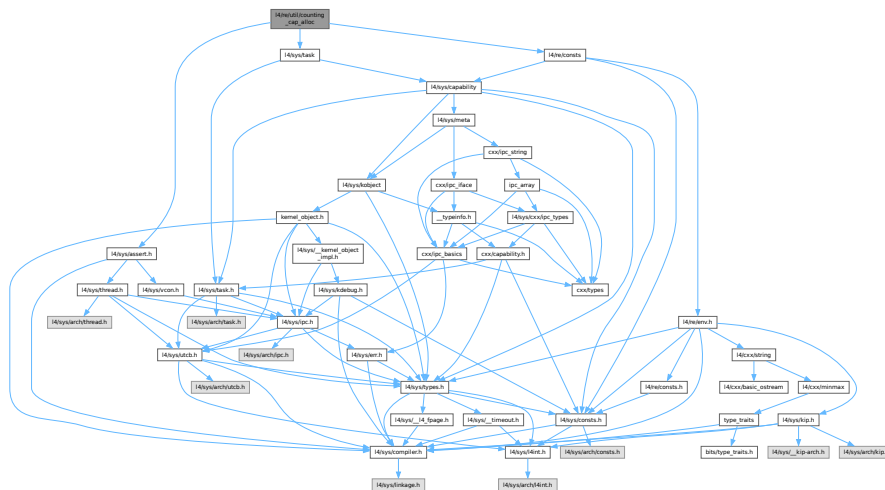
```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00003 /*
00004  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009
00010 #pragma once
00011
00012 #include <l4/bid_config.h>
00013 #include <l4/re/cap_alloc>
00014
00015 #if defined(CONFIG_L4RE_BITMAP_CAP_ALLOC)
00016 #include <l4/re/util/bitmap_cap_alloc>
00017
00018 namespace L4Re { namespace Util {
00019     using _Cap_alloc_impl = Cap_alloc_base;
00020 }}
00021
00022 #elif defined(CONFIG_L4RE_COUNTING_CAP_ALLOC)
00023 #include <l4/re/util/counting_cap_alloc>
00024 #include <l4/re/util/debug>
00025
00026 namespace L4Re { namespace Util {
00027     // RISC-V does not natively support subword atomics, such as __atomic_load_1.
00028     // The RISC-V gcc developers have decided to emulate these via libatomic, which
00029     // is automatically linked against.
00030     #if defined(__GCC_HAVE_SYNC_COMPARE_AND_SWAP_1) || defined(ARCH_arm) || defined(ARCH_riscv)
00031     using _Cap_alloc_impl
00032         = Counting_cap_alloc<L4Re::Util::Counter_atomic<unsigned char>,
00033                             L4Re::Util::Dbg>;
00034     #elif defined(ARCH_sparc)
00035     using _Cap_alloc_impl
00036         = Counting_cap_alloc<L4Re::Util::Counter<unsigned char>,
00037                             L4Re::Util::Dbg>;
00038     #warning "Thread-safe capability allocator not available!"
00039     #else
00040     #error "Unsupported platform"
00041     #endif
00042 }}
00043
00044 #else
00045 #error "No supported capability allocator selected"
00046 #endif
00047
00048 namespace L4Re { namespace Util {
00049     class _Cap_alloc final : public L4Re::Cap_alloc, private _Cap_alloc_impl
00050     {
00051     public:
00052         template <unsigned COUNT>
00053         using Storage = _Cap_alloc_impl::Storage<COUNT>;
00054
00055         using _Cap_alloc_impl::_Cap_alloc_impl; // Expose underlying constructor
00056         void operator delete(void *) {} // Prevent global operator delete reference
00057
00058         L4::Cap<void> alloc() noexcept override
00059         { return _Cap_alloc_impl::alloc(); }
00060
00061         template< typename T >
00062         L4::Cap<T> alloc() noexcept

```

16.388 l4/re/util/counting_cap_alloc File Reference

```
#include <linux/sys/task>
#include <linux/sys/assert.h>
#include <linux/re/consts>
Include dependency graph for counting_cap_alloc:
```




```

00013 #pragma once
00014
00015 #include <l4/sys/task>
00016 #include <l4/sys/assert.h>
00017 #include <l4/re/consts>
00018
00019 namespace L4Re { namespace Util {
00020
00026 template< typename COUNTER = unsigned char >
00027 struct Counter
00028 {
00029     typedef COUNTER Type;
00030     Type _cnt;
00031
00032     static Type nil() { return 0; }
00033     static Type unused() { return 0; }
00034
00035     void free() { _cnt = 0; }
00036     bool is_free() const { return _cnt == 0; }
00037     bool is_saturated() const { return static_cast<Type>(_cnt + 1) == 0; }
00038
00050     bool inc()
00051     {
00052         if (is_saturated())
00053             return true; // no change and no warning
00054         ++_cnt;
00055         if (is_saturated())
00056             return false; // warn caller that counter is now saturated
00057         else
00058             return true; // success
00059     }
00060
00067     Type dec()
00068     {
00069         if (is_saturated())
00070             return _cnt; // no change
00071         else
00072             return --_cnt; // success
00073     }
00074
00075     bool try_alloc()
00076     {
00077         if (_cnt == 0)
00078         {
00079             _cnt = 1;
00080             return true;
00081         }
00082         return false;
00083     }
00084 };
00085
00097 template< typename COUNTER = unsigned char >
00098 struct Counter_atomic
00099 {
00100     typedef COUNTER Type;
00101     Type _cnt;
00102
00103     static Type nil() { return 0; }
00104     static Type unused() { return 1; }
00105
00106     bool is_free() const { return __atomic_load_n(&_cnt, __ATOMIC_RELAXED) == 0; }
00107     static bool is_saturated(Type cnt) { return static_cast<Type>(cnt + 1) == 0; }
00108
00109     bool try_alloc()
00110     {
00111         Type expected = nil();
00112         // Use "acquire" memory ordering. Any operations tied to the capability slot
00113         // must only be observable after the slot has been occupied.
00114         return __atomic_compare_exchange_n(&_cnt, &expected, 2, false,
00115                                           __ATOMIC_ACQUIRE, __ATOMIC_RELAXED);
00116     }
00117
00121     bool inc()
00122     {
00123         Type old_cnt = __atomic_load_n(&_cnt, __ATOMIC_RELAXED);
00124         Type new_cnt;
00125         do
00126         {
00127             if (is_saturated(old_cnt))
00128                 return true; // no change and no warning
00129             new_cnt = old_cnt + 1;
00130         }
00131         while (!__atomic_compare_exchange_n(&_cnt, &old_cnt, new_cnt, false,
00132                                           __ATOMIC_RELAXED, __ATOMIC_RELAXED));
00133         if (is_saturated(new_cnt))
00134             return false; // warn caller that counter is now saturated
00135         else

```

```

00136         return true; // success
00137     }
00138
00142 Type dec()
00143 {
00144     Type old_cnt = __atomic_load_n(&cnt, __ATOMIC_RELAXED);
00145     Type new_cnt;
00146     do
00147     {
00148         if (is_saturated(old_cnt))
00149             return old_cnt; // no change
00150         new_cnt = old_cnt - 1;
00151     }
00152     while (!__atomic_compare_exchange_n(&cnt, &old_cnt, new_cnt, false,
00153                                         __ATOMIC_RELAXED, __ATOMIC_RELAXED));
00154     return new_cnt; // success
00155 }
00156
00157 void free()
00158 {
00159     // Use "release" memory ordering to make sure that any operations tied to
00160     // the capability slot are observable by other threads before the slot can
00161     // be reused.
00162     __atomic_store_n(&cnt, 0, __ATOMIC_RELEASE);
00163 }
00164 };
00165
00190 template <typename COUNTERTYPE, typename Dbg>
00191 class Counting_cap_alloc
00192 {
00193 private:
00194     void operator = (Counting_cap_alloc const &) { }
00195     typedef COUNTERTYPE Counter;
00196
00197     COUNTERTYPE *_items;
00198     long _free_hint;
00199     long _bias;
00200     long _capacity;
00201     Dbg *_dbg;
00202
00203 public:
00204
00205     template <unsigned COUNT>
00206     struct Storage
00207     {
00208         COUNTERTYPE _buf[COUNT];
00209         typedef COUNTERTYPE Buf_type[COUNT];
00210         enum { Size = COUNT };
00211     };
00212
00213     Counting_cap_alloc(long capacity, void *m, long bias, Dbg *dbg) noexcept
00214     : _items((Counter*)m), _free_hint(0), _bias(bias), _capacity(capacity),
00215       _dbg(dbg)
00216     {}
00217
00218 protected:
00224     Counting_cap_alloc() noexcept
00225     : _items(0), _free_hint(0), _bias(0), _capacity(0)
00226     {}
00227
00242     void setup(void *m, long capacity, long bias, Dbg *dbg) noexcept
00243     {
00244         _items = static_cast<Counter*>(m);
00245         _capacity = capacity;
00246         _bias = bias;
00247         _dbg = dbg;
00248     }
00249
00250 public:
00257     L4::Cap<void> alloc() noexcept
00258     {
00259         long free_hint = __atomic_load_n(&_free_hint, __ATOMIC_RELAXED);
00260
00261         for (long i = free_hint; i < _capacity; ++i)
00262             if (_items[i].try_alloc())
00263             {
00264                 _free_hint = i + 1;
00265                 return L4::Cap<void>((i + _bias) << L4_CAP_SHIFT);
00266             }
00267
00268         // _free_hint is not necessarily correct in case of multi-threading! Make
00269         // sure we don't miss any potentially free slots.
00270         for (long i = 0; i < free_hint && i < _capacity; ++i)
00271             if (_items[i].try_alloc())
00272             {
00273                 _free_hint = i + 1;
00274                 return L4::Cap<void>((i + _bias) << L4_CAP_SHIFT);

```

```

00275     }
00276
00277     return L4::Cap<void>::Invalid;
00278 }
00279
00281 template <typename T>
00282 L4::Cap<T> alloc() noexcept
00283 {
00284     return L4::cap_cast<T>(alloc());
00285 }
00286
00287
00296 void take(L4::Cap<void> cap) noexcept
00297 {
00298     long c;
00299     if (!range_check_and_get_idx(cap, &c))
00300         return;
00301
00302     if (!L4_UNLIKELY(_items[c].inc()))
00303         _dbg->printf("Warning: Reference counter of cap 0x%x now saturated!\n",
00304             cap.cap() » L4_CAP_SHIFT);
00305 }
00306
00307
00321 bool free(L4::Cap<void> cap, l4_cap_idx_t task = L4_INVALID_CAP,
00322     unsigned unmap_flags = L4_FP_ALL_SPACES) noexcept
00323 {
00324     long c;
00325     if (!range_check_and_get_idx(cap, &c))
00326         return false;
00327
00328     l4_assert(!_items[c].is_free());
00329
00330     if (l4_is_valid_cap(task))
00331         l4_task_unmap(task, cap.fpage(), unmap_flags);
00332
00333     if (c < _free_hint)
00334         _free_hint = c;
00335
00336     _items[c].free();
00337
00338     return true;
00339 }
00340
00359 bool release(L4::Cap<void> cap, l4_cap_idx_t task = L4_INVALID_CAP,
00360     unsigned unmap_flags = L4_FP_ALL_SPACES) noexcept
00361 {
00362     long c;
00363     if (!range_check_and_get_idx(cap, &c))
00364         return false;
00365
00366     l4_assert(!_items[c].is_free());
00367
00368     if (_items[c].dec() == Counter::unused())
00369     {
00370         if (task != L4_INVALID_CAP)
00371             l4_task_unmap(task, cap.fpage(), unmap_flags);
00372
00373         if (c < _free_hint)
00374             _free_hint = c;
00375
00376         // Let others allocate this slot only after the l4_task_unmap() has
00377         // finished.
00378         _items[c].free();
00379
00380         return true;
00381     }
00382     return false;
00383 }
00384
00388 long last() noexcept
00389 {
00390     return _capacity + _bias - 1;
00391 }
00392
00393 private:
00394 bool range_check_and_get_idx(L4::Cap<void> cap, long *c)
00395 {
00396     *c = cap.cap() » L4_CAP_SHIFT;
00397     if (*c < _bias)
00398         return false;
00399
00400     *c -= _bias;
00401
00402     return *c < _capacity;
00403 }
00404 };

```

```
00405
00406 }}
```

16.390 dataspace_svr

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00005  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00006  *      economic rights: Technische Universität Dresden (Germany)
00007  *
00008  * License: see LICENSE.spdx (in this directory or the directories above)
00009  */
00010 #pragma once
00011
00012 #include <string.h>
00013 #include <stddef.h>
00014 #include <l4/bid_config.h>
00015 #include <l4/sys/types.h>
00016 #include <l4/cxx/minmax>
00017 #include <l4/re/dataspace>
00018 #include <l4/re/dataspace-sys.h>
00019 #include <l4/sys/cxx/ipc_legacy>
00020
00021 namespace L4Re { namespace Util {
00022
00029 class Dataspace_svr
00030 {
00031 public:
00032     L4_RPC_LEGACY_DISPATCH(L4Re::Dataspace);
00033
00034     typedef L4::Ipc::Snd_fpage::Map_type Map_type;
00035     typedef L4::Ipc::Snd_fpage::Cacheopt Cache_type;
00036
00037     Dataspace_svr() noexcept
00038     : _ds_start(0), _ds_size(0), _map_flags(L4::Ipc::Snd_fpage::Map),
00039       _cache_flags(L4::Ipc::Snd_fpage::Cached)
00040     {}
00041
00042     virtual ~Dataspace_svr() noexcept {}
00043
00057     l4_ret_t map(Dataspace::Offset offset,
00058                 Dataspace::Map_addr local_addr,
00059                 Dataspace::Flags flags,
00060                 Dataspace::Map_addr min_addr,
00061                 Dataspace::Map_addr max_addr,
00062                 L4::Ipc::Snd_fpage &memory)
00063     {
00064         memory = L4::Ipc::Snd_fpage();
00065
00066         offset    = l4_trunc_page(offset);
00067         local_addr = l4_trunc_page(local_addr);
00068
00069         if (!check_limit(offset))
00070         {
00071             #if 0
00072                 printf("limit failed: off=%lx sz=%lx\n", offset, size());
00073             #endif
00074             return -L4_ERANGE;
00075         }
00076
00077         min_addr = l4_trunc_page(min_addr);
00078         max_addr = l4_round_page(max_addr);
00079
00080         l4_addr_t addr = _ds_start + offset;
00081         unsigned char order = L4_PAGESHIFT;
00082
00083         while (order < 30 /* limit to 1GB flexpage */)
00084         {
00085             l4_addr_t map_base = l4_trunc_size(addr, order + 1);
00086             if (map_base < _ds_start)
00087                 break;
00088
00089             if (map_base + (1UL << (order + 1)) - 1 > (_ds_start + round_size() - 1))
00090                 break;
00091
00092             map_base = l4_trunc_size(local_addr, order + 1);
00093             if (map_base < min_addr)
00094                 break;
00095
00096             if (map_base + (1UL << (order + 1)) - 1 > max_addr - 1)
00097                 break;
```

```

00098
00099     l4_addr_t mask = ~(~0UL << (order + 1));
00100     if (local_addr == ~0UL || ((addr ^ local_addr) & mask))
00101         break;
00102
00103     ++order;
00104 }
00105
00106 l4_addr_t map_base = l4_trunc_size(addr, order);
00107
00108 Dataspace::Map_addr b = map_base;
00109 unsigned send_order = order;
00110 l4_ret_t err = map_hook(offset /*map_base - _ds_start*/, order, flags,
00111                        &b, &send_order);
00112 if (err < 0)
00113     return err;
00114
00115 l4_fpage_t fpage = l4_fpage(b, send_order, flags.fpage_rights());
00116
00117 memory = L4::Ipc::Snd_fpage(fpage, local_addr, _map_flags, _cache_flags);
00118
00119 return L4_EOK;
00120 }
00121
00136 virtual l4_ret_t map_hook([[maybe_unused]] Dataspace::Offset offs,
00137                        [[maybe_unused]] unsigned order,
00138                        [[maybe_unused]] Dataspace::Flags flags,
00139                        [[maybe_unused]] Dataspace::Map_addr *base,
00140                        [[maybe_unused]] unsigned *send_order)
00141 {
00142     return 0;
00143 }
00144
00150 virtual void take() noexcept
00151 {}
00152
00160 virtual unsigned long release() noexcept
00161 { return 0; }
00162
00175 virtual l4_ret_t copy([[maybe_unused]] l4_addr_t dst_offs,
00176                      [[maybe_unused]] l4_umword_t src_id,
00177                      [[maybe_unused]] l4_addr_t src_offs,
00178                      [[maybe_unused]] unsigned long size) noexcept
00179 {
00180     return -L4_ENODEV;
00181 }
00182
00192 virtual l4_ret_t clear(unsigned long offs, unsigned long size) const noexcept
00193 {
00194     if (!check_limit(offs))
00195         return -L4_ERANGE;
00196
00197     unsigned long sz = size = cxx::min(size, round_size() - offs);
00198
00199     while (sz)
00200     {
00201         unsigned long b_addr = _ds_start + offs;
00202         unsigned long b_sz = cxx::min(size - offs, sz);
00203
00204         memset(reinterpret_cast<void *>(b_addr), 0, b_sz);
00205
00206         offs += b_sz;
00207         sz -= b_sz;
00208     }
00209
00210     return 0;
00211 }
00212
00224 virtual l4_ret_t allocate([[maybe_unused]] l4_addr_t offset,
00225                          [[maybe_unused]] l4_size_t size,
00226                          [[maybe_unused]] unsigned access) noexcept
00227 {
00228     return -L4_ENODEV;
00229 }
00230
00236 virtual unsigned long page_shift() const noexcept
00237 { return L4_LOG2_PAGESIZE; }
00238
00244 virtual bool is_static() const noexcept
00245 { return true; }
00246
00259 virtual l4_ret_t map_info([[maybe_unused]] l4_addr_t &start_addr,
00260                          [[maybe_unused]] l4_addr_t &end_addr) noexcept
00261 { return -L4_EPERM; }
00262
00263
00264 l4_ret_t op_map(L4Re::Dataspace::Rights rights,

```

```

00265         L4Re::Dataspace::Offset offset,
00266         L4Re::Dataspace::Map_addr spot,
00267         L4Re::Dataspace::Flags flags,
00268         L4::Ipc::Snd_fpage &fp)
00269     {
00270         auto rf = map_flags(rights);
00271         if (!rf.w() && flags.w())
00272             return -L4_EPERM;
00273
00274         return map(offset, spot, flags & rf, 0, ~0, fp);
00275     }
00276
00277     l4_ret_t op_allocate(L4Re::Dataspace::Rights rights,
00278                         L4Re::Dataspace::Offset offset,
00279                         L4Re::Dataspace::Size size)
00280     { return allocate(offset, size, rights & 3); }
00281
00282     l4_ret_t op_copy_in(L4Re::Dataspace::Rights rights,
00283                        L4Re::Dataspace::Offset dst_offs,
00284                        L4::Ipc::Snd_fpage const &src_cap,
00285                        L4Re::Dataspace::Offset src_offs,
00286                        L4Re::Dataspace::Size sz)
00287     {
00288         if (!src_cap.id_received())
00289             return -L4_EINVAL;
00290
00291         if (!(rights & L4_CAP_FPAGE_W))
00292             return -L4_EACCESS;
00293
00294         if (sz == 0)
00295             return L4_EOK;
00296
00297         return copy(dst_offs, src_cap.data(), src_offs, sz);
00298     }
00299
00300     l4_ret_t op_info(L4Re::Dataspace::Rights rights, L4Re::Dataspace::Stats &s)
00301     {
00302         s.size = size();
00303         // only return writable if really writable
00304         s.flags = Dataspace::Flags(0);
00305         if (map_flags(rights).w())
00306             s.flags |= Dataspace::F::W;
00307         return L4_EOK;
00308     }
00309
00310     l4_ret_t op_clear(L4Re::Dataspace::Rights rights,
00311                      L4Re::Dataspace::Offset offset,
00312                      L4Re::Dataspace::Size size)
00313     {
00314         if (!map_flags(rights).w())
00315             return -L4_EACCESS;
00316
00317         return clear(offset, size);
00318     }
00319
00320     l4_ret_t op_map_info(L4Re::Dataspace::Rights,
00321                         [[maybe_unused]] l4_addr_t &start_addr,
00322                         [[maybe_unused]] l4_addr_t &end_addr)
00323     {
00324 #ifdef CONFIG_MMU
00325         return 0;
00326 #else
00327         return map_info(start_addr, end_addr);
00328 #endif
00329     }
00330
00331 protected:
00332     unsigned long size() const noexcept
00333     { return _ds_size; }
00334     unsigned long map_flags() const noexcept
00335     { return _map_flags; }
00336     unsigned long page_size() const noexcept
00337     { return 1UL < page_shift(); }
00338     unsigned long round_size() const noexcept
00339     { return l4_round_size(size(), page_shift()); }
00340     bool check_limit(l4_addr_t offset) const noexcept
00341     { return offset < round_size(); }
00342
00343     L4Re::Dataspace::Flags
00344     map_flags(L4Re::Dataspace::Rights rights = L4_CAP_FPAGE_W) const noexcept
00345     {
00346         auto f = (_rw_flags & L4Re::Dataspace::Flags(0x0f)) | L4Re::Dataspace::F::Caching_mask;
00347         if (!(rights & L4_CAP_FPAGE_W))
00348             f -= L4Re::Dataspace::F::W;
00349
00350         return f;
00351     }

```

```

00352
00353 protected:
00354     void size(unsigned long size) noexcept { _ds_size = size; }
00355
00356     l4_addr_t _ds_start;
00357     l4_size_t _ds_size;
00358     Map_type _map_flags;
00359     Cache_type _cache_flags;
00360     L4Re::Dataspace::Flags _rw_flags;
00361 };
00362
00363 }

```

16.391 I4/re/debug File Reference

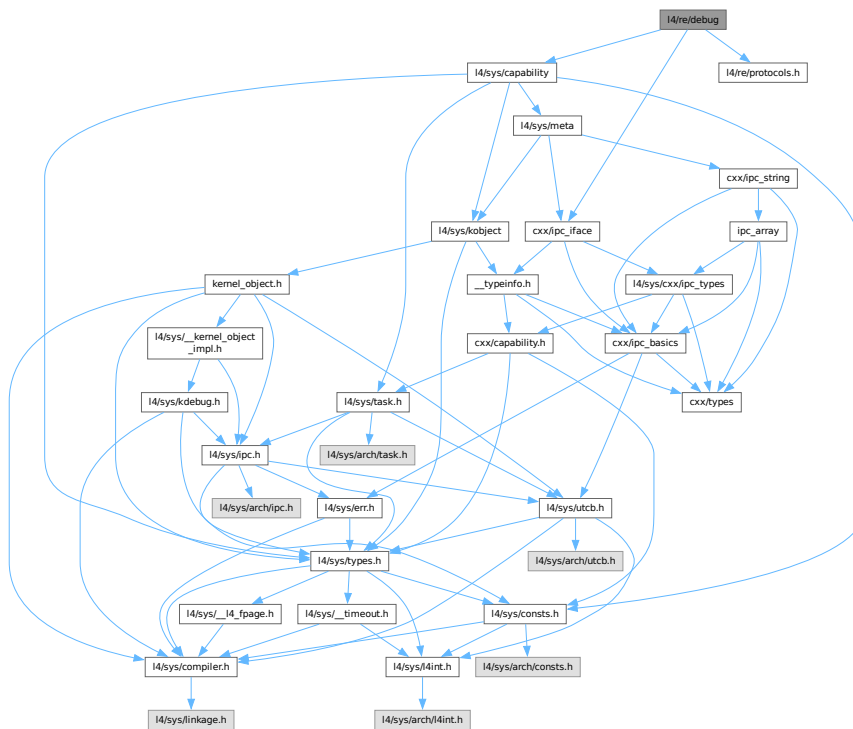
Debug interface.

```

#include <l4/sys/capability>
#include <l4/re/protocols.h>
#include <l4/sys/cxx/ipc_iface>

```

Include dependency graph for debug:



Data Structures

- class [L4Re::Debug_obj](#)
Debug interface.

Namespaces

- namespace [L4Re](#)
L4Re C++ Interfaces.

16.391.1 Detailed Description

Debug interface.

Definition in file [debug](#).

16.392 debug

[Go to the documentation of this file.](#)

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013 #pragma once
00014
00015 #include <l4/sys/capability>
00016 #include <l4/re/protocols.h>
00017 #include <l4/sys/cxx/ipc_iface>
00018
00019 namespace L4Re {
00033
00040 class L4_EXPORT Debug_obj :
00041     public L4::Kobject_t<Debug_obj, L4::Kobject, L4RE_PROTO_DEBUG>
00042 {
00043 public:
00044
00056     L4_INLINE_RPC(l4_ret_t, debug, (unsigned long function));
00057     typedef L4::Typeid::Rpc_nocode<debug_t> Rpc;
00058 };
00059
00060 template<typename BASE>
00061 class Debug_obj_t :
00062     public L4::Kobject_2t<Debug_obj_t<BASE>, BASE, Debug_obj, L4::PROTO_EMPTY>
00063 {
00064     typedef L4::Typeid::Rpc<> Rpc;
00065 };
00066 }
```

16.393 debug

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014 #pragma once
00015
00016 #include <l4/sys/types.h>
00017
00018 namespace L4Re { namespace Util {
00019     class Err
00020     {
00021     public:
00022         enum Level
00023         {
00024             Normal = 0,
00025             Fatal,
00026         };
00027
00028         static char const *const levels[];
00029
00030         void tag() const
00031         { cprintf("%s: %s", _component, levels[_l]); }
00032
00033         int printf(char const *fmt, ...) const
00034             __attribute__((format(printf, 2, 3)));
00035     };
00036 }
```



```

00035
00036 int cprintf(char const *fmt, ...) const
00037     __attribute__((format(printf, 2, 3)));
00038
00039 constexpr Err(Level l, char const *component) : _l(l), _component(component)
00040 {}
00041
00042 private:
00043     Level _l;
00044     char const *_component;
00045 };
00046
00047
00048 class Dbg
00049 {
00050 private:
00051     void tag() const;
00052
00053 #ifndef NDEBUG
00054
00055     unsigned long _m;
00056     char const *const _component;
00057     char const *const _subsys;
00058
00059 # ifdef __clang__
00060
00061     int printf_impl(char const *fmt, ...) const
00062         __attribute__((format(printf, 2, 3)));
00063
00064     int cprintf_impl(char const *fmt, ...) const
00065         __attribute__((format(printf, 2, 3)));
00066
00067 # endif
00068
00069 public:
00070     static unsigned long level;
00071
00072     static void set_level(unsigned long l) { level = l; }
00073
00074     bool is_active() const { return _m & level; }
00075
00076 # ifdef __clang__
00077
00078     int printf(char const *fmt, ...) const
00079         __attribute__((format(printf, 2, 3)));
00080
00081     int cprintf(char const *fmt, ...) const
00082         __attribute__((format(printf, 2, 3)));
00083
00084 # else
00085
00086     int __attribute__((always_inline, format(printf, 2, 3)))
00087     printf(char const *fmt, ...) const
00088     {
00089         if (!(level & _m))
00090             return 0;
00091
00092         return printf_impl(fmt, __builtin_va_arg_pack());
00093     }
00094
00095     int __attribute__((always_inline, format(printf, 2, 3)))
00096     cprintf(char const *fmt, ...) const
00097     {
00098         if (!(level & _m))
00099             return 0;
00100
00101         return cprintf_impl(fmt, __builtin_va_arg_pack());
00102     }
00103
00104 # endif
00105
00106     explicit constexpr
00107     Dbg() : _m(1), _component(0), _subsys(0) { };
00108
00109     explicit constexpr
00110     Dbg(unsigned long mask, char const *comp, char const *subs)
00111     : _m(mask), _component(comp), _subsys(subs)
00112     {}
00113
00114 #else
00115
00116 public:
00117     static void set_level(unsigned long) {}
00118     bool is_active() const { return false; }
00119
00120     int printf(char const * /*fmt*/, ...) const
00121         __attribute__((format(printf, 2, 3)))

```

```

00122     { return 0; }
00123
00124     int cprintf(char const * /*fmt*/, ...) const
00125     __attribute__((format(printf, 2, 3)))
00126     { return 0; }
00127
00128     explicit constexpr
00129     Dbg() {}
00130
00131     explicit constexpr
00132     Dbg(unsigned long, char const *, char const *) {}
00133
00134 #endif
00135
00136 };
00137
00138 }}
00139

```

16.394 env_ns

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002
00003 #pragma once
00004
00005 #include <l4/re/cap_alloc>
00006 #include <l4/re/util/cap_alloc>
00007 #include <l4/re/namespace>
00008 #include <l4/re/env>
00009 #include <string.h>
00010
00011 namespace L4Re { namespace Util {
00012
00013     class Env_ns
00014     {
00015     private:
00016         L4Re::Cap_alloc *_ca;
00017         Env const *_env;
00018
00019     public:
00020         explicit Env_ns(Env const *env = Env::env(),
00021                        L4Re::Cap_alloc *ca = &L4Re::Util::cap_alloc)
00022             : _ca(ca), _env(env) {}
00023
00024         L4::Cap<void>
00025         query(char const *name, unsigned len, int timeout = Namespace::To_default,
00026              l4_umword_t *local_id = 0, bool iterate = true) const noexcept
00027         {
00028             typedef Env::Cap_entry Cap_entry;
00029
00030             // Skip possible first slash
00031             if (len && name[0] == '/')
00032             {
00033                 ++name;
00034                 --len;
00035             }
00036
00037             char const *n = name;
00038             for (; len && *n != '/'; ++n, --len) // Count first path element
00039                 ;
00040
00041             Cap_entry const *e = _env->get(name, n - name);
00042             if (!e)
00043                 return L4::Cap<void>(-L4_ENOENT);
00044
00045             if (len > 0 && *n == '/')
00046             {
00047                 L4::Cap<L4Re::Namespace> ns(e->cap);
00048                 L4::Cap<void> cap = _ca->alloc<void>();
00049
00050                 if (!cap.is_valid())
00051                     return L4::Cap<void>(-L4_ENOMEM);
00052
00053                 long r = ns->query(n + 1, len - 1, cap, timeout, local_id, iterate);
00054                 if (r >= 0)
00055                     return cap;
00056
00057                 _ca->free(cap);
00058
00059                 return L4::Cap<void>(r);
00060             }
00061
00062             return L4::Cap<void>(e->cap);

```

```

00063     }
00064
00065     L4::Cap<void>
00066     query(char const *name, int timeout = Namespace::To_default,
00067           l4_umword_t *local_id = 0, bool iterate = true) const noexcept
00068     { return query(name, __builtin_strlen(name), timeout, local_id, iterate); }
00069
00070     template<typename T >
00071     L4::Cap<T>
00072     query(char const *name, int timeout = Namespace::To_default,
00073           l4_umword_t *local_id = 0, bool iterate = true) const noexcept
00074     {
00075         return L4::cap_cast<T>(query(name, __builtin_strlen(name),
00076                                     timeout, local_id, iterate));
00077     }
00078 };
00079
00080 }}

```

16.395 event

```

00001 // vi:set ft=c++: -- Mode: C++ --
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009
00010 #pragma once
00011
00012 #include <l4/sys/capability>
00013 #include <l4/sys/irq>
00014 #include <l4/sys/cxx/ipc_iface>
00015 #include <l4/sys/cxx/ipc_array>
00016 #include <l4/re/dataspace>
00017 #include <l4/re/event.h>
00018
00019 namespace L4Re {
00020
00021
00022
00023
00024 typedef l4re_event_stream_id_t Event_stream_id;
00025 typedef l4re_event_absinfo_t Event_absinfo;
00026
00027
00028 class L4_EXPORT Event_stream_bitmap_h
00029 {
00030 protected:
00031     static unsigned __get_idx(unsigned idx)
00032     { return idx / (sizeof(unsigned long)*8); }
00033
00034     static unsigned long __get_mask(unsigned idx)
00035     { return 1ul << (idx % (sizeof(unsigned long)*8)); }
00036
00037     static bool __get_bit(unsigned long const *bm, unsigned max, unsigned idx)
00038     {
00039         if (idx <= max)
00040             return bm[__get_idx(idx)] & __get_mask(idx);
00041         return false;
00042     }
00043
00044     static void __set_bit(unsigned long *bm, unsigned max, unsigned idx, bool v)
00045     {
00046         if (idx > max)
00047             return;
00048
00049         if (v)
00050             bm[__get_idx(idx)] |= __get_mask(idx);
00051         else
00052             bm[__get_idx(idx)] &= ~__get_mask(idx);
00053     }
00054 };
00055
00056 class L4_EXPORT Event_stream_info
00057 : public l4re_event_stream_info_t,
00058   private Event_stream_bitmap_h
00059 {
00060 public:
00061     bool get_propbit(unsigned idx) const
00062     { return __get_bit(propbits, L4RE_EVENT_PROP_MAX, idx); }
00063
00064     void set_propbit(unsigned idx, bool v)
00065     { __set_bit(propbits, L4RE_EVENT_PROP_MAX, idx, v); }
00066
00067
00068
00069
00070
00071
00072
00073
00074
00075
00076
00077
00078
00079
00080

```

```

00081
00082 bool get_evbit(unsigned idx) const
00083 { return __get_bit(evbits, L4RE_EVENT_EV_MAX, idx); }
00084
00085 void set_evbit(unsigned idx, bool v)
00086 { __set_bit(evbits, L4RE_EVENT_EV_MAX, idx, v); }
00087
00088 bool get_keybit(unsigned idx) const
00089 { return __get_bit(keybits, L4RE_EVENT_KEY_MAX, idx); }
00090
00091 void set_keybit(unsigned idx, bool v)
00092 { __set_bit(keybits, L4RE_EVENT_KEY_MAX, idx, v); }
00093
00094 bool get_relbit(unsigned idx) const
00095 { return __get_bit(relbits, L4RE_EVENT_REL_MAX, idx); }
00096
00097 void set_relbit(unsigned idx, bool v)
00098 { __set_bit(relbits, L4RE_EVENT_REL_MAX, idx, v); }
00099
00100 bool get_absbit(unsigned idx) const
00101 { return __get_bit(absbits, L4RE_EVENT_ABS_MAX, idx); }
00102
00103 void set_absbit(unsigned idx, bool v)
00104 { __set_bit(absbits, L4RE_EVENT_ABS_MAX, idx, v); }
00105
00106 bool get_swbit(unsigned idx) const
00107 { return __get_bit(swbits, L4RE_EVENT_SW_MAX, idx); }
00108
00109 void set_swbit(unsigned idx, bool v)
00110 { __set_bit(swbits, L4RE_EVENT_SW_MAX, idx, v); }
00111 };
00112
00113 class L4_EXPORT Event_stream_state
00114 : public l4re_event_stream_state_t,
00115   private Event_stream_bitmap_h
00116 {
00117 public:
00118     bool get_keybit(unsigned idx) const
00119     { return __get_bit(keybits, L4RE_EVENT_KEY_MAX, idx); }
00120
00121     void set_keybit(unsigned idx, bool v)
00122     { __set_bit(keybits, L4RE_EVENT_KEY_MAX, idx, v); }
00123
00124     bool get_swbit(unsigned idx) const
00125     { return __get_bit(swbits, L4RE_EVENT_SW_MAX, idx); }
00126
00127     void set_swbit(unsigned idx, bool v)
00128     { __set_bit(swbits, L4RE_EVENT_SW_MAX, idx, v); }
00129 };
00130
00137 class L4_EXPORT Event :
00138     public L4::Kobject_t<Event, L4::Icu, L4RE_PROTO_EVENT>
00139 {
00140 public:
00149     L4_RPC(l4_ret_t, get_buffer, (L4::Ipc::Out<L4::Cap<Dataspace>> ds));
00150
00157     L4_RPC(l4_ret_t, get_num_streams, ());
00158
00170     L4_RPC(l4_ret_t, get_stream_info, (int idx, Event_stream_info *info));
00171
00181     L4_RPC(l4_ret_t, get_stream_info_for_id, (l4_umword_t stream_id, Event_stream_info *info));
00182
00194     L4_RPC_NF(l4_ret_t, get_axis_info, (l4_umword_t stream_id,
00195                                         L4::Ipc::Array<unsigned const,
00196                                         unsigned long> axes,
00197                                         L4::Ipc::Array<Event_absinfo,
00198                                         unsigned long> &info));
00199
00200     l4_ret_t get_axis_info(l4_umword_t stream_id, unsigned naxes,
00201                             unsigned const *axis, Event_absinfo *info) const noexcept
00202     {
00203         L4::Ipc::Array<Event_absinfo, unsigned long> i(naxes, info);
00204         return get_axis_info_t::call(c(), stream_id,
00205                                     L4::Ipc::Array<unsigned const, unsigned long>(naxes, axis), i);
00206     }
00207
00217     L4_RPC(l4_ret_t, get_stream_state_for_id, (l4_umword_t stream_id,
00218                                                Event_stream_state *state));
00219
00220     typedef L4::Typeid::Rpcs<
00221         get_buffer_t,
00222         get_num_streams_t,
00223         get_stream_info_t,
00224         get_stream_info_for_id_t,
00225         get_axis_info_t,
00226         get_stream_state_for_id_t
00227     > Rpcs;

```

```

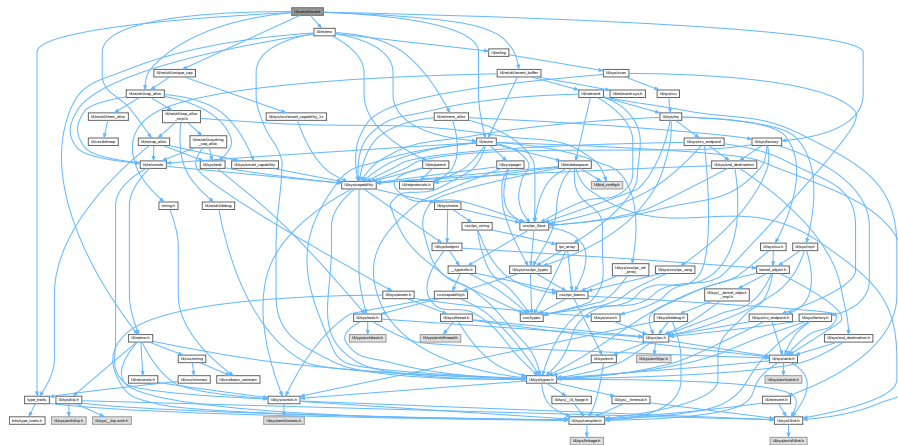
00228 };
00229
00234 struct L4_EXPORT Default_event_payload
00235 {
00236     unsigned short type;
00237     unsigned short code;
00238     int value;
00239     l4_umword_t stream_id;
00240 };
00241
00242
00247 template< typename PAYLOAD = Default_event_payload >
00248 class L4_EXPORT Event_buffer_t
00249 {
00250 public:
00251
00255     struct Event
00256     {
00257         long long time;
00258         PAYLOAD payload;
00259
00263         void free() noexcept { l4_mb(); time = 0; }
00264     };
00265
00266 private:
00267     Event *_current;
00268     Event *_begin;
00269     Event const *_end;
00270
00271     void inc() noexcept
00272     {
00273         ++_current;
00274         if (_current == _end)
00275             _current = _begin;
00276     }
00277
00278 public:
00279
00280     Event_buffer_t() : _current(0), _begin(0), _end(0) {}
00281
00282     void reset()
00283     {
00284         for (Event *i = _begin; i != _end; ++i)
00285             i->time = 0;
00286         _current = _begin;
00287     }
00288
00295     Event_buffer_t(void *buffer, l4_addr_t size)
00296     : _current(static_cast<Event*>(buffer)), _begin(_current),
00297       _end(_begin + size / sizeof(Event))
00298     { reset(); }
00299
00305     Event *next() noexcept
00306     {
00307         Event *c = _current;
00308         if (c->time)
00309         {
00310             inc();
00311             return c;
00312         }
00313         return 0;
00314     }
00315
00322     bool put(Event const &ev) noexcept
00323     {
00324         Event *c = _current;
00325         if (c->time)
00326             return false;
00327
00328         inc();
00329         c->payload = ev.payload;
00330         l4_wmb();
00331         c->time = ev.time;
00332         return true;
00333     }
00334 };
00335
00336 typedef Event_buffer_t<Default_event_payload> Event_buffer;
00337
00338 }

```

16.396 l4/re/util/event File Reference

```
#include <l4/re/cap_alloc>
#include <l4/re/util/cap_alloc>
#include <l4/re/util/unique_cap>
#include <l4/re/env>
#include <l4/re/rm>
#include <l4/re/util/event_buffer>
#include <l4/sys/factory>
#include <l4/cxx/type_traits>
```

Include dependency graph for event:



Data Structures

- class [L4Re::Util::Event_t< PAYLOAD >](#)
Convenience wrapper for getting access to an event object.

Namespaces

- namespace [L4Re](#)
L4Re C++ Interfaces.
- namespace [L4Re::Util](#)
Documentation of the L4 Runtime Environment utility functionality in C++.

16.397 event

[Go to the documentation of this file.](#)

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009 #pragma once
00010
00011 #include <l4/re/cap_alloc>
00012 #include <l4/re/util/cap_alloc>
```

```

00016 #include <l4/re/util/unique_cap>
00017 #include <l4/re/env>
00018 #include <l4/re/rm>
00019 #include <l4/re/util/event_buffer>
00020 #include <l4/sys/factory>
00021 #include <l4/cxx/type_traits>
00022
00023 namespace L4Re { namespace Util {
00024
00031 template< typename PAYLOAD >
00032 class Event_t
00033 {
00034 public:
00038     enum Mode
00039     {
00040         Mode_irq,
00041         Mode_polling,
00042     };
00043
00058     template<typename IRQ_TYPE>
00059     int init(L4::Cap<L4Re::Event> event,
00060             L4Re::Env const *env = L4Re::Env::env(),
00061             L4Re::Cap_alloc *ca = &L4Re::Util::cap_alloc)
00062     {
00063         Unique_cap<L4Re::Dataspace> ev_ds(ca->alloc<L4Re::Dataspace>());
00064         if (!ev_ds.is_valid())
00065             return -L4_ENOMEM;
00066
00067         int r;
00068
00069         Unique_del_cap<IRQ_TYPE> ev_irq(ca->alloc<IRQ_TYPE>());
00070         if (!ev_irq.is_valid())
00071             return -L4_ENOMEM;
00072
00073         if ((r = l4_error(env->factory()->create(ev_irq.get()))))
00074             return r;
00075
00076         if ((r = l4_error(event->bind(0, ev_irq.get()))))
00077             return r;
00078
00079         if ((r = event->get_buffer(ev_ds.get())))
00080             return r;
00081
00082         long sz = ev_ds->size();
00083         if (sz < 0)
00084             return sz;
00085
00086         Rm::Unique_region<void*> buf;
00087
00088         if ((r = env->rm()->attach(&buf, sz,
00089                                   L4Re::Rm::F::Search_addr | L4Re::Rm::F::RW,
00090                                   L4::Ipc::make_cap_rw(ev_ds.get()))))
00091             return r;
00092
00093         _ev_buffer = L4Re::Event_buffer_t<PAYLOAD>(buf.get(), sz);
00094         _ev_ds      = cxx::move(ev_ds);
00095         _ev_irq     = cxx::move(ev_irq);
00096         _buf        = cxx::move(buf);
00097
00098         return 0;
00099     }
00100
00112 int init_poll(L4::Cap<L4Re::Event> event,
00113              L4Re::Env const *env = L4Re::Env::env(),
00114              L4Re::Cap_alloc *ca = &L4Re::Util::cap_alloc)
00115 {
00116     Unique_cap<L4Re::Dataspace> ev_ds(ca->alloc<L4Re::Dataspace>());
00117     if (!ev_ds.is_valid())
00118         return -L4_ENOMEM;
00119
00120     int r;
00121
00122     if ((r = event->get_buffer(ev_ds.get())))
00123         return r;
00124
00125     long sz = ev_ds->size();
00126     if (sz < 0)
00127         return sz;
00128
00129     Rm::Unique_region<void*> buf;
00130
00131     if ((r = env->rm()->attach(&buf, sz,
00132                               L4Re::Rm::F::Search_addr | L4Re::Rm::F::RW,
00133                               L4::Ipc::make_cap_rw(ev_ds.get()))))
00134         return r;
00135
00136     _ev_buffer = L4Re::Event_buffer_t<PAYLOAD>(buf.get(), sz);

```

```

00137     _ev_ds      = cxx::move(ev_ds);
00138     _buf        = cxx::move(buf);
00139
00140     return 0;
00141 }
00142
00148 L4Re::Event_buffer_t<PAYLOAD> &buffer() { return _ev_buffer; }
00149
00155 L4::Cap<L4::Triggerable> irq() const { return _ev_irq.get(); }
00156
00157 private:
00158     Unique_cap<L4Re::Dataspace> _ev_ds;
00159     Unique_del_cap<L4::Triggerable> _ev_irq;
00160     L4Re::Event_buffer_t<PAYLOAD> _ev_buffer;
00161     Rm::Unique_region<void*> _buf;
00162 };
00163
00164 typedef Event_t<Default_event_payload> Event;
00165
00166 }}

```

16.398 event_buffer

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009
00010 #pragma once
00011
00012 #include <l4/re/event>
00013 #include <l4/re/event-sys.h>
00014 #include <l4/re/rm>
00015
00016 #include <string.h>
00017
00018 namespace L4Re { namespace Util {
00019
00024 template< typename PAYLOAD >
00025 class Event_buffer_t : public L4Re::Event_buffer_t<PAYLOAD>
00026 {
00027 private:
00028     void *_buf;
00029 public:
00035     void *buf() const noexcept { return _buf; }
00036
00045     long attach(L4::Cap<L4Re::Dataspace> ds, L4::Cap<L4Re::Rm> rm) noexcept
00046     {
00047         l4_addr_t sz = ds->size();
00048         _buf = 0;
00049
00050         long r = rm->attach(&_buf, sz,
00051                             L4Re::Rm::F::Search_addr | L4Re::Rm::F::RW,
00052                             L4::Ipc::make_cap_rw(ds));
00053         if (r < 0)
00054             return r;
00055
00056         *static_cast<L4Re::Event_buffer_t<PAYLOAD>*>(this)
00057             = L4Re::Event_buffer_t<PAYLOAD>(_buf, sz);
00058         return 0;
00059     }
00060
00068     long detach(L4::Cap<L4Re::Rm> rm) noexcept
00069     {
00070         L4::Cap<L4Re::Dataspace> ds;
00071         if (_buf)
00072             return rm->detach(_buf, &ds);
00073         return 0;
00074     }
00075 };
00076
00077
00082 template< typename PAYLOAD >
00083 class Event_buffer_consumer_t : public Event_buffer_t<PAYLOAD>
00084 {
00085 public:
00086
00093     template< typename CB, typename D >
00094     void foreach_available_event(CB const &cb, D data = D())

```



```

00095 {
00096     typename Event_buffer_t<PAYLOAD>::Event *e;
00097     while ((e = Event_buffer_t<PAYLOAD>::next()))
00098     {
00099         cb(e, data);
00100         e->free();
00101     }
00102 }
00103
00104 template< typename CB, typename D >
00105 void process(L4::Cap<L4::Irq> irq,
00106             L4::Cap<L4::Thread> thread,
00107             CB const &cb, D data = D())
00108 {
00109
00110     if (l4_error(irq->bind_thread(thread, 0)))
00111         return;
00112
00113     while (1)
00114     {
00115         long r;
00116         r = l4_ipc_error(l4_irq_receive(irq.cap(), L4_IPC_NEVER),
00117                         l4_utcb());
00118         if (r)
00119             continue;
00120
00121         foreach_available_event(cb, data);
00122     }
00123 }
00124 };
00125
00126 typedef Event_buffer_t<Default_event_payload> Event_buffer;
00127 typedef Event_buffer_consumer_t<Default_event_payload> Event_buffer_consumer;
00128
00129 }}

```

16.399 event_svr

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009
00010 #pragma once
00011
00012 #include <l4/re/event_enums.h>
00013 #include <l4/re/event>
00014 #include <l4/re/event-sys.h>
00015 #include <l4/re/util/icu_svr>
00016 #include <l4/cxx/minmax>
00017
00018 #include <l4/sys/cxx/ipc_legacy>
00019
00020 namespace L4Re { namespace Util {
00021
00022     template< typename SVR >
00023     class Event_svr : public Icu_cap_array_svr<SVR>
00024     {
00025     private:
00026         typedef Icu_cap_array_svr<SVR> Icu_svr;
00027
00028     protected:
00029         L4::Cap<L4Re::Dataspace> _ds;
00030         typename Icu_svr::Irq _irq;
00031
00032     public:
00033         Event_svr() : Icu_svr(1, &_irq) {}
00034
00035         L4_RPC_LEGACY_DISPATCH(L4Re::Event);
00036         L4_RPC_LEGACY_USING(Icu_svr);
00037
00038         l4_ret_t op_get_buffer(L4Re::Event::Rights, L4::Ipc::Cap<L4Re::Dataspace> &ds)
00039         {
00040             static_cast<SVR*>(this)->reset_event_buffer();
00041             ds = L4::Ipc::Cap<L4Re::Dataspace>(_ds, L4_CAP_FPAGE_RW);
00042             return 0;
00043         }
00044
00045         l4_ret_t op_get_num_streams(L4Re::Event::Rights)

```

```

00052 { return static_cast<SVR*>(this)->get_num_streams(); }
00053
00054 l4_ret_t op_get_stream_info(L4Re::Event::Rights, int idx, Event_stream_info &info)
00055 { return static_cast<SVR*>(this)->get_stream_info(idx, &info); }
00056
00057 l4_ret_t op_get_stream_info_for_id(L4Re::Event::Rights, l4_umword_t id,
00058                                   Event_stream_info &info)
00059 { return static_cast<SVR*>(this)->get_stream_info_for_id(id, &info); }
00060
00061 l4_ret_t op_get_axis_info(L4Re::Event::Rights, l4_umword_t id,
00062                           L4::Ipc::Array_in_buf<unsigned, unsigned long> const &axes,
00063                           L4::Ipc::Array_ref<Event_absinfo, unsigned long> &info)
00064 {
00065     unsigned naxes = cxx::min<unsigned>(L4RE_ABS_MAX, axes.length);
00066
00067     info.length = 0;
00068
00069     Event_absinfo _info[L4RE_ABS_MAX];
00070     int r = static_cast<SVR*>(this)->get_axis_info(id, naxes, axes.data, _info);
00071     if (r < 0)
00072         return r;
00073
00074     for (unsigned i = 0; i < naxes; ++i)
00075         info.data[i] = _info[i];
00076
00077     info.length = naxes;
00078     return r;
00079 }
00080
00081 l4_ret_t op_get_stream_state_for_id(L4Re::Event::Rights, l4_umword_t stream_id,
00082                                     Event_stream_state &state)
00083 { return static_cast<SVR*>(this)->get_stream_state_for_id(stream_id, &state); }
00084
00085 int get_num_streams() const { return 0; }
00086 int get_stream_info(int, L4Re::Event_stream_info *)
00087 { return -L4_EINVAL; }
00088 int get_stream_info_for_id(l4_umword_t, L4Re::Event_stream_info *)
00089 { return -L4_EINVAL; }
00090 int get_axis_info(l4_umword_t, unsigned /*naxes*/, unsigned const * /*axes*/,
00091                  L4Re::Event_absinfo *)
00092 { return -L4_EINVAL; }
00093 int get_stream_state_for_id(l4_umword_t, L4Re::Event_stream_state *)
00094 { return -L4_EINVAL; }
00095 };
00096
00097 }}

```

16.400 icu_svr

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2009-2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00008 #pragma once
00009
00010
00011 #include <l4/sys/types.h>
00012
00013 #include <l4/sys/icu>
00014 #include <l4/sys/task>
00015 #include <l4/re/env>
00016 #include <l4/re/util/cap_alloc>
00017 #include <l4/sys/cxx/ipc_legacy>
00018
00019 namespace L4Re { namespace Util {
00020
00021 template< typename ICU >
00022 class Icu_svr
00023 {
00024 private:
00025     ICU const *this_icu() const { return static_cast<ICU const *>(this); }
00026     ICU *this_icu() { return static_cast<ICU*>(this); }
00027
00028 public:
00029     L4_RPC_LEGACY_DISPATCH(L4::Icu);
00030
00031     l4_ret_t op_bind(L4::Icu::Rights, l4_umword_t irqnum,
00032                     L4::Ipc::Snd_fpage irq_fp);
00033     l4_ret_t op_unbind(L4::Icu::Rights, l4_umword_t irqnum,
00034                       L4::Ipc::Snd_fpage irq_fp);

```

```

00035     l4_ret_t op_info(L4::Icu::Rights, L4::Icu::_Info &info);
00036     l4_ret_t op_msi_info(L4::Icu::Rights, l4_umword_t irqnum,
00037                          l4_uint64_t source, l4_icu_msi_info_t &info);
00038     l4_ret_t op_mask(L4::Icu::Rights, l4_umword_t irqnum);
00039     l4_ret_t op_unmask(L4::Icu::Rights, l4_umword_t irqnum);
00040     l4_ret_t op_set_mode(L4::Icu::Rights, l4_umword_t, l4_umword_t)
00041     { return 0; }
00042 };
00043
00044 template<typename ICU> inline
00045 l4_ret_t
00046 Icu_svr<ICU>::op_bind(L4::Icu::Rights, l4_umword_t irqnum,
00047                      L4::Ipc::Snd_fpage irq_fp)
00048 {
00049     typename ICU::Irq *irq = this_icu()->icu_get_irq(irqnum);
00050     if (!irq)
00051         return -L4_EINVAL;
00052
00053     return irq->bind(this_icu(), irq_fp);
00054 }
00055
00056 template<typename ICU> inline
00057 l4_ret_t
00058 Icu_svr<ICU>::op_unbind(L4::Icu::Rights, l4_umword_t irqnum,
00059                        L4::Ipc::Snd_fpage irq_fp)
00060 {
00061     typename ICU::Irq *irq = this_icu()->icu_get_irq(irqnum);
00062     if (!irq)
00063         return -L4_EINVAL;
00064
00065     return irq->unbind(this_icu(), irq_fp);
00066 }
00067
00068 template<typename ICU> inline
00069 l4_ret_t
00070 Icu_svr<ICU>::op_info(L4::Icu::Rights, L4::Icu::_Info &info)
00071 {
00072     l4_icu_info_t i;
00073     this_icu()->icu_get_info(&i);
00074     info.features = i.features;
00075     info.nr_irqs = i.nr_irqs;
00076     info.nr_msis = i.nr_msis;
00077     return 0;
00078 }
00079
00080 template<typename ICU> inline
00081 l4_ret_t
00082 Icu_svr<ICU>::op_msi_info(L4::Icu::Rights, l4_umword_t irqnum,
00083                           l4_uint64_t source, l4_icu_msi_info_t &info)
00084 {
00085     typename ICU::Irq *irq = this_icu()->icu_get_irq(irqnum);
00086     if (!irq)
00087         return -L4_EINVAL;
00088     return irq->msi_info(source, &info);
00089 }
00090
00091 template<typename ICU> inline
00092 l4_ret_t
00093 Icu_svr<ICU>::op_mask(L4::Icu::Rights, l4_umword_t irqnum)
00094 {
00095     typename ICU::Irq *irq = this_icu()->icu_get_irq(irqnum);
00096     if (irq)
00097         irq->mask(true);
00098     return -L4_ENOREPLY;
00099 }
00100
00101 template<typename ICU> inline
00102 l4_ret_t
00103 Icu_svr<ICU>::op_unmask(L4::Icu::Rights, l4_umword_t irqnum)
00104 {
00105     typename ICU::Irq *irq = this_icu()->icu_get_irq(irqnum);
00106     if (irq)
00107         irq->mask(false);
00108     return -L4_ENOREPLY;
00109 }
00110
00111
00112 template< typename ICU >
00113 class Icu_cap_array_svr : public Icu_svr<ICU>
00114 {
00115 protected:
00116     static void free_irq_cap(L4::Cap<L4::Irq> &cap)
00117     {
00118         if (cap)
00119         {
00120             L4Re::Util::cap_alloc.free(cap);
00121             cap.invalidate();

```

```

00122     }
00123 }
00124
00125 public:
00126     class Irq
00127     {
00128     public:
00129         Irq() {}
00130         ~Irq() { ICU::free_irq_cap(_cap); }
00131
00132         void trigger() const
00133         {
00134             if (_cap)
00135                 _cap->trigger();
00136         }
00137
00138         l4_ret_t bind(ICU *, L4::Ipc::Snd_fpage const &irq_fp);
00139         l4_ret_t unbind(ICU *, L4::Ipc::Snd_fpage const &irq_fp);
00140         void mask(bool /*mask*/) const
00141         { }
00142
00143         l4_ret_t msi_info(l4_uint64_t, l4_icu_msi_info_t *) const
00144         { return -L4_EINVAL; }
00145
00146         L4::Cap<L4::Irq> cap() const { return _cap; }
00147
00148     private:
00149         L4::Cap<L4::Irq> _cap;
00150     };
00151
00152 private:
00153     Irq *_irqs;
00154     unsigned _nr_irqs;
00155
00156 public:
00157
00158     Icu_cap_array_svr(unsigned nr_irqs, Irq *irqs)
00159     : _irqs(irqs), _nr_irqs(nr_irqs)
00160     {}
00161
00162     Irq *icu_get_irq(l4_umword_t irqnum)
00163     {
00164         if (irqnum >= _nr_irqs)
00165             return 0;
00166
00167         return _irqs + irqnum;
00168     }
00169
00170     void icu_get_info(l4_icu_info_t *inf)
00171     {
00172         inf->features = 0;
00173         inf->nr_irqs = _nr_irqs;
00174         inf->nr_msis = 0;
00175     }
00176 };
00177
00178 template< typename ICU >
00179 l4_ret_t
00180 Icu_cap_array_svr<ICU>::Irq::bind(ICU *cfb, L4::Ipc::Snd_fpage const &irq_fp)
00181 {
00182     if (!irq_fp.cap_received())
00183         return -L4_EINVAL;
00184
00185     L4::Cap<L4::Irq> irq = cfb->server_iface()->template_rcv_cap<L4::Irq>(0);
00186     if (!irq)
00187         return -L4_EINVAL;
00188
00189     int r = cfb->server_iface()->realloc_rcv_cap(0);
00190     if (r < 0)
00191         return r;
00192
00193     ICU::free_irq_cap(_cap);
00194     _cap = irq;
00195     return 0;
00196 }
00197
00198 template< typename ICU >
00199 l4_ret_t
00200 Icu_cap_array_svr<ICU>::Irq::unbind(ICU *, L4::Ipc::Snd_fpage const &/*irq_fp*/)
00201 {
00202     ICU::free_irq_cap(_cap);
00203     _cap = L4::Cap<L4::Irq>::Invalid;
00204     return 0;
00205 }
00206
00207
00208 }

```


16.401.1 Detailed Description

Item allocator.

Definition in file [item_alloc](#).

16.402 item_alloc

[Go to the documentation of this file.](#)

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00003 /*
00004  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00005  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00006  *      economic rights: Technische Universität Dresden (Germany)
00007  *
00008  * License: see LICENSE.spdx (in this directory or the directories above)
00009  */
00010
00011 #pragma once
00012
00013 #include <l4/cxx/bitmap>
00014
00015 namespace L4Re { namespace Util {
00016
00017     using cxx::Bitmap_base;
00018     using cxx::Bitmap;
00019
00020     class Item_alloc_base
00021     {
00022     private:
00023         long _capacity;
00024         long _free_hint;
00025         Bitmap_base _bits;
00026
00027         void hint(long hint)
00028         { __atomic_store_n(&_free_hint, hint, __ATOMIC_RELAXED); }
00029
00030     public:
00031         bool is_allocated(long item) const noexcept
00032         { return _bits[item]; }
00033
00034         long hint() const { return __atomic_load_n(&_free_hint, __ATOMIC_RELAXED); }
00035
00036         bool alloc(long item) noexcept
00037         {
00038             return !_bits.atomic_get_and_set(item);
00039         }
00040
00041         void free(long item) noexcept
00042         {
00043             if (item < hint())
00044                 hint(item);
00045
00046             _bits.atomic_clear_bit(item);
00047         }
00048
00049         Item_alloc_base(long size, void *mem) noexcept
00050             : _capacity(size), _free_hint(0), _bits(mem)
00051         {}
00052
00053         long alloc() noexcept
00054         {
00055             long free_hint = hint();
00056
00057             for (long i = free_hint; i < _capacity; ++i)
00058                 if (alloc(i))
00059                     hint(i + 1);
00060             return i;
00061         }
00062
00063         // _free_hint is not necessarily correct in case of multi-threading! Make
00064         // sure we don't miss any potentially free slots.
00065         for (long i = 0; i < free_hint && i < _capacity; ++i)
00066             if (alloc(i))
00067                 hint(i + 1);
00067     }
00068 }

```

```

00077         return i;
00078     }
00079
00080     return -1;
00081 }
00082
00083 long size() const noexcept
00084 {
00085     return _capacity;
00086 }
00087 };
00088
00089 template< long Bits >
00090 class Item_alloc : public Item_alloc_base
00091 {
00092 private:
00093     typename Bitmap_base::Word<Bits>::Type _bits[Bitmap_base::Word<Bits>::Size];
00094
00095 public:
00096     Item_alloc() noexcept : Item_alloc_base(Bits, _bits) {}
00097 };
00098
00099 }

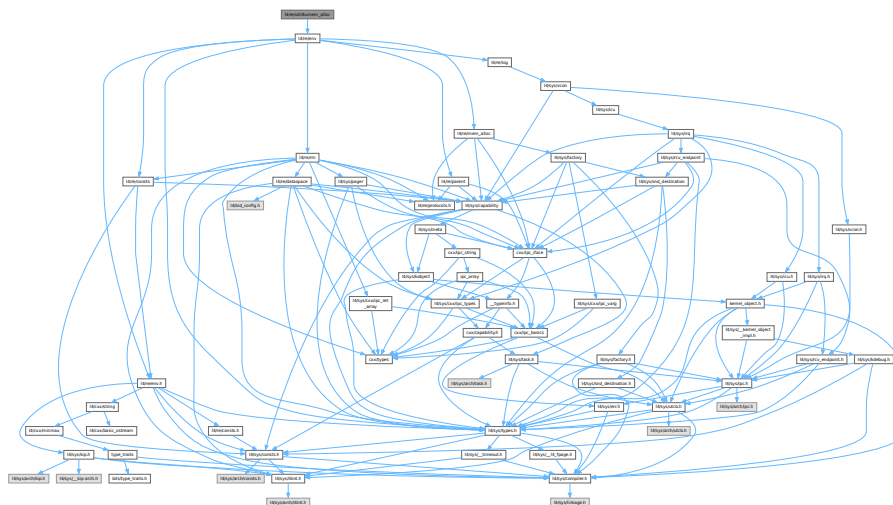
```

16.403 l4/re/util/kumem_alloc File Reference

Kumem allocator helper.

```
#include <l4/re/env>
```

Include dependency graph for kumem_alloc:



Namespaces

- namespace [L4Re](#)
L4Re C++ Interfaces.
- namespace [L4Re::Util](#)
Documentation of the L4 Runtime Environment utility functionality in C++.

Functions

- int [L4Re::Util::kumem_alloc](#) ([l4_addr_t](#) *mem, unsigned pages_order, [L4::Cap](#)< [L4::Task](#) > task=[L4Re::Env::env](#)() ->task(), [L4::Cap](#)< [L4Re::Rm](#) > rm=[L4Re::Env::env](#)() ->rm()) noexcept
Allocate state area.

16.403.1 Detailed Description

Kumem allocator helper.

Definition in file [kumem_alloc](#).

16.404 kumem_alloc

[Go to the documentation of this file.](#)

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008  * (c) 2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014
00015 #pragma once
00016
00017 #include <l4/re/env>
00018
00019 namespace L4Re { namespace Util {
00020
00026
00044 int
00045 kumem_alloc(l4_addr_t *mem, unsigned pages_order,
00046             L4::Cap<L4::Task> task = L4Re::Env::env()->task(),
00047             L4::Cap<L4Re::Rm> rm = L4Re::Env::env()->rm()) noexcept;
00048
00050 }}
```

16.405 name_space_svr

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009 #pragma once
00010
00011 #include <l4/cxx/avl_tree>
00012 #include <l4/cxx/std_ops>
00013 #include <l4/sys/cxx/ipc_epiface>
00014 #include <l4/cxx/string>
00015 #include <l4/re/util/debug>
00016
00017 #include <l4/sys/capability>
00018 #include <l4/re/namespace>
00019
00020 #include <stddef.h>
00021 #include <string.h>
00022
00023 namespace L4Re { namespace Util { namespace Names {
00024
00028 class Name : public cxx::String
00029 {
00030 public:
00031
00032     Name(const char *name = "") : String(name, __builtin_strlen(name)) {}
00033     Name(const char *name, unsigned long len) : String(name, len) {}
00034     Name(cxx::String const &n) : String(n) {}
00035     char const *name() const { return start(); }
00036     bool operator < (Name const &r) const
00037     {
00038         unsigned long l = cxx::min(len(), r.len());
00039         int v = memcmp(start(), r.start(), l);
00040         return v < 0 || (v == 0 && len() < r.len());
00041     }
00042 }
```



```

00042 };
00043
00044
00048 class Obj
00049 {
00050 protected:
00051     unsigned _f;
00052     union
00053     {
00054         l4_cap_idx_t _cap;
00055         L4::Epiface *_obj;
00056     };
00057
00058
00059 public:
00060     enum Flags
00061     {
00062         F_rw          = L4Re::Namespace::Rw,
00063         F_strong       = L4Re::Namespace::Strong,
00064
00065         F_trusted      = L4Re::Namespace::Trusted,
00066
00067         F_rights_mask = F_rw | F_strong | F_trusted,
00068
00069         F_cap          = 0x100,
00070         F_local        = 0x200,
00071         F_replacable   = 0x400,
00072         F_base_mask    = 0xf00,
00073     };
00074
00075
00076     unsigned flags() const { return _f; }
00077     void restrict_flags(unsigned max_rights)
00078     { _f &= (~F_rights_mask | (max_rights & F_rights_mask)); }
00079
00080     bool is_rw() const { return (_f & F_rw) == F_rw; }
00081     bool is_strong() const { return _f & F_strong; }
00082
00083     bool is_valid() const { return _f & F_cap; }
00084     bool is_complete() const { return is_valid(); }
00085     bool is_local() const { return _f & F_local; }
00086     bool is_replacable() const { return _f & F_replacable; }
00087     bool is_trusted() const { return _f & F_trusted; }
00088
00089     L4::Epiface *obj() const { if (is_local()) return _obj; return 0; }
00090     L4::Cap<void> cap() const
00091     {
00092         if (!is_local())
00093             return L4::Cap<void>(_cap);
00094         if (!_obj)
00095             return L4::Cap<void>::Invalid;
00096         return _obj->obj_cap();
00097     }
00098
00099
00100     void set(Obj const &o, unsigned flags)
00101     {
00102         *this = o;
00103         restrict_flags(flags);
00104     }
00105
00106     explicit Obj(unsigned flags = 0)
00107     : _f(flags), _cap(L4_INVALID_CAP)
00108     {}
00109
00110     Obj(unsigned f, L4::Cap<void> const &cap)
00111     : _f((f & ~F_base_mask) | F_cap), _cap(cap.cap())
00112     {}
00113
00114     Obj(unsigned f, L4::Epiface *o)
00115     : _f((f & ~F_base_mask) | F_cap | F_local), _obj(o)
00116     {}
00117
00118     void reset(unsigned flags)
00119     {
00120         _f = (_f & F_replacable) | (flags & ~(F_cap | F_local));
00121         _cap = L4_INVALID_CAP;
00122     }
00123
00124
00125 };
00126
00127
00131 class Entry : public cxx::Avl_tree_node
00132 {
00133 private:
00134     friend class Name_space;

```

```

00135     Name _n;
00136     Obj _o;
00137
00138     bool _dynamic;
00139
00140 public:
00141     Entry(Name const &n, Obj const &o, bool dynamic = false)
00142     : _n(n), _o(o), _dynamic(dynamic) {}
00143
00144     Name const &name() const { return _n; }
00145     Obj const &obj() const { return _o; }
00146     Obj *obj() { return &_o; }
00147     void obj(Obj const &o) { _o = o; }
00148
00149     bool is_placeholder() const
00150     { return !obj()->is_complete(); }
00151
00152     bool is_dynamic() const { return _dynamic; }
00153
00154     void set(Obj const &o)
00155     {
00156         obj()->set(o, obj()->flags());
00157     }
00158
00159 private:
00160     void * operator new (size_t s);
00161     void operator delete(void *b);
00162
00163 };
00164
00165 struct Names_get_key
00166 {
00167     typedef Name Key_type;
00168     static Key_type const &key_of(Entry const *e)
00169     { return e->name(); }
00170 };
00171
00172
00180 class Name_space
00181 {
00182     friend class Entry;
00183
00184 private:
00185     typedef cxx::Avl_tree<Entry, Names_get_key> Tree;
00186     Tree _tree;
00187
00188 protected:
00189     L4Re::Util::Dbg const &_dbg;
00190     L4Re::Util::Err const &_err;
00191
00192 public:
00193
00194     typedef Tree::Const_iterator Const_iterator;
00195
00196     Const_iterator begin() const { return _tree.begin(); }
00197     Const_iterator end() const { return _tree.end(); }
00198
00199     Name_space(L4Re::Util::Dbg const &dbg, L4Re::Util::Err const &err)
00200     : _dbg(dbg), _err(err)
00201     {}
00202
00206     virtual ~Name_space() {}
00207
00208     Entry *find(Name const &name) const { return _tree.find_node(name); }
00209     Entry *remove(Name const &name) { return _tree.remove(name); }
00210     Entry *find_iter(Name const &pname) const
00211     {
00212         Name name = pname;
00213         _dbg.printf("resolve '%.*s': ", name.len(), name.start());
00214         Name_space const *ns = this;
00215         while (ns)
00216         {
00217             cxx::String::Index sep = name.find("/");
00218             cxx::String part;
00219             if (!name.eof(sep))
00220                 part = name.head(sep);
00221             else
00222                 part = name;
00223
00224             _dbg.cprintf(" '%.*s'", part.len(), part.start());
00225             Entry *o = ns->find(Name(part.start(), part.len()));
00226
00227             if (!o)
00228             {
00229                 _dbg.cprintf(": resolution failed: '%.*s' remaining\n",
00230                             name.len(), name.start());
00231                 return 0;

```

```

00232     }
00233
00234     auto const *obj = o->obj()->obj();
00235     ns = dynamic_cast<Name_space const *>(obj);
00236     if (ns)
00237     {
00238         if (!name.eof(sep))
00239         {
00240             name = name.substr(sep + 1);
00241             continue;
00242         }
00243     }
00244
00245     _dbg.cprintf(": found object: %p (%s)\n",
00246                 obj, obj ? typeid(*obj).name() : "");
00247
00248     return o;
00249 }
00250
00251 return 0;
00252 }
00253
00254 bool insert(Entry *e) { return _tree.insert(e).second; }
00255
00256 void dump(bool rec = false, int indent = 0) const;
00257
00258 protected:
00259     // server support -----
00272     virtual Entry *alloc_dynamic_entry(Name const &n, unsigned flags) = 0;
00273
00279     virtual void free_dynamic_entry(Entry *e) = 0;
00280
00303     virtual l4_ret_t get_epiface(l4_umword_t data, bool is_local, L4::Epiface **lo) = 0;
00304
00317     virtual l4_ret_t copy_receive_cap(L4::Cap<void> *cap) = 0;
00318
00327     virtual void free_capability(L4::Cap<void> cap) = 0;
00328
00337     virtual void free_epiface(L4::Epiface *epiface) = 0;
00338
00339     l4_ret_t insert_entry(Name const &name, unsigned flags, Entry **e)
00340     {
00341         Entry *n = find(name);
00342         if (n && n->obj()->is_valid())
00343         {
00344             if (!(flags & L4Re::Namespace::Overwrite)
00345                 && n->obj()->cap().validate(L4_BASE_TASK_CAP).label() > 0)
00346                 return -L4_EEXIST;
00347
00348             if (n->obj()->is_local())
00349                 free_epiface(n->obj()->obj());
00350             else
00351                 free_capability(n->obj()->cap());
00352
00353             if (n->is_dynamic())
00354             {
00355                 remove(n->name());
00356                 free_dynamic_entry(n);
00357                 n = 0;
00358             }
00359             else
00360             {
00361                 if (!n->obj()->is_replacable())
00362                     return -L4_EEXIST;
00363                 n->obj()->reset(Obj::F_rw);
00364             }
00365         }
00366
00367         flags &= L4Re::Namespace::Cap_flags;
00368         if (!n)
00369         {
00370             if (!(n = alloc_dynamic_entry(name, flags)))
00371                 return -L4_ENOMEM;
00372             else
00373             {
00374                 if (!insert(n))
00375                 {
00376                     free_dynamic_entry(n);
00377                     return -L4_EEXIST;
00378                 }
00379             }
00380         }
00381
00382         *e = n;
00383         return 0;
00384     }
00385

```

```

00386 public:
00387     // server interface -----
00388     L4_ret_t op_query(L4Re::Namespace::Rights,
00389                     L4::Ipc::Array_in_buf<char, unsigned long> const &name,
00390                     L4::Ipc::Snd_fpage &snd_cap, L4::Ipc::Opt<L4::Opcode> &dummy,
00391                     L4::Ipc::Opt<L4::Ipc::Array_ref<char, unsigned long> > &out_name)
00392     {
00393     #if 1
00394         _dbg.printf("query: [%ld] '%.*s'\n", name.length,
00395                     static_cast<int>(name.length), name.data);
00396     #endif
00397
00398         char const *sep
00399             = static_cast<char const*>(memchr(name.data, '/', name.length));
00400         unsigned long part;
00401         if (sep)
00402             part = sep - name.data;
00403         else
00404             part = name.length;
00405
00406         Entry *n = find(Name(name.data, part));
00407         if (!n)
00408             return -L4_ENOENT;
00409         else if (!n->obj()->is_valid())
00410             return -L4_EAGAIN;
00411         else
00412         {
00413             if (n->obj()->cap().validate(L4_BASE_TASK_CAP).label() <= 0)
00414             {
00415                 if (n->obj()->is_local())
00416                     free_epiface(n->obj()->obj());
00417                 else
00418                     free_capability(n->obj()->cap());
00419
00420                 if (n->is_dynamic())
00421                 {
00422                     remove(n->name());
00423                     free_dynamic_entry(n);
00424                 }
00425                 return -L4_ENOENT;
00426             }
00427
00428             // make picky clients happy
00429             dummy.set_valid();
00430             // prevent warning about writing uninitialized data in IPC framework
00431             dummy = 0;
00432
00433             L4_ret_t result = 0;
00434
00435             out_name.set_valid();
00436             if (part < name.length)
00437             {
00438                 result |= L4Re::Namespace::Partly_resolved;
00439                 memcpy(out_name->data, name.data + part + 1, name.length - part - 1);
00440                 out_name->length = name.length - part - 1;
00441             }
00442             else
00443                 out_name->length = 0;
00444
00445             unsigned flags = L4_CAP_FPAGE_R;
00446             if (n->obj()->is_rw()) flags |= L4_CAP_FPAGE_W;
00447             if (n->obj()->is_strong()) flags |= L4_CAP_FPAGE_S;
00448
00449             snd_cap = L4::Ipc::Snd_fpage(n->obj()->cap(), flags);
00450             _dbg.printf(" result = %x flgs=%x strg=%d\n",
00451                         result, flags, static_cast<int>(n->obj()->is_strong()));
00452             return result;
00453         }
00454     }
00455
00456     L4_ret_t op_register_obj(L4Re::Namespace::Rights, unsigned flags,
00457                             L4::Ipc::Array_in_buf<char, unsigned long> const &name,
00458                             L4::Ipc::Snd_fpage &cap)
00459     {
00460         if (name.length == 0 || memchr(name.data, '/', name.length))
00461             return -L4_EINVAL;
00462
00463         L4::Cap<void> reg_cap(L4_INVALID_CAP);
00464         L4::Epiface *src_o = 0;
00465
00466         // Did we receive something we have handed out ourselves? If yes,
00467         // register the object under the given name but do not allocate
00468         // anything more.
00469         if (cap.id_received() || cap.local_id_received())
00470         {
00471             if (L4_ret_t ret = get_epiface(cap.data(), cap.local_id_received(), &src_o))
00472                 return ret;

```

```

00473
00474     // Make sure rights are restricted to the mapped rights.
00475     flags &= (cap.data() & 0x3UL) | ~0x3UL;
00476 }
00477 else if (cap.cap_received())
00478 {
00479     if (l4_ret_t ret = copy_receive_cap(&reg_cap))
00480         return ret;
00481 }
00482 else if (!cap.is_valid())
00483 {
00484     reg_cap = L4::Cap<void>::Invalid;
00485 }
00486 else
00487     return -L4_EINVAL;
00488
00489 // got a valid entry to register
00490 _dbg.printf("register: '%.*s' flags=%x\n", static_cast<int>(name.length),
00491             name.data, flags);
00492
00493 Name _name(name.data, name.length);
00494
00495 Entry *n;
00496 if (l4_ret_t r = insert_entry(_name, flags, &n))
00497 {
00498     if (cap.cap_received())
00499         free_capability(reg_cap);
00500     if (src_o)
00501         free_epiface(src_o);
00502
00503     return r;
00504 }
00505
00506 if (src_o)
00507     n->set(Names::Obj(flags & L4Re::Namespace::Cap_flags, src_o));
00508 else if (reg_cap.is_valid())
00509     n->set(Names::Obj(flags & L4Re::Namespace::Cap_flags, reg_cap));
00510
00511 return 0;
00512 }
00513
00514 l4_ret_t op_unlink(L4Re::Namespace::Rights,
00515                  L4::Ipcc::Array_in_buf<char, unsigned long> const &name)
00516 {
00517     {
00518 #if 1
00519         _dbg.printf("unlink: [%ld] '%.*s'\n", name.length,
00520                     static_cast<int>(name.length), name.data);
00521 #endif
00522
00523         char const *sep
00524             = static_cast<char const*>(memchr(name.data, '/', name.length));
00525         unsigned long part;
00526         if (sep)
00527             part = sep - name.data;
00528         else
00529             part = name.length;
00530
00531         Entry *n = find(Name(name.data, part));
00532         if (!n || !n->obj()->is_valid())
00533             return -L4_ENOENT;
00534
00535         if (n->obj()->is_local())
00536             free_epiface(n->obj()->obj());
00537         else
00538             free_capability(n->obj()->cap());
00539
00540         if (n->is_dynamic())
00541         {
00542             remove(n->name());
00543             free_dynamic_entry(n);
00544         }
00545         else
00546             return -L4_EACCESS;
00547
00548         return 0;
00549     }
00550 };
00551
00552 }}}
00553

```

16.406 object_registry

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009
00010 #pragma once
00011
00012 #include <l4/re/util/cap_alloc>
00013 #include <l4/re/util/unique_cap>
00014 #include <l4/re/consts>
00015 #include <l4/re/env>
00016
00017 #include <l4/sys/cxx/ipc_server_loop>
00018 #include <l4/sys/factory>
00019 #include <l4/sys/task>
00020 #include <l4/sys/thread>
00021 #include <l4/sys/ipc_gate>
00022
00023 #include <l4/cxx/exceptions>
00024
00025 namespace L4Re { namespace Util {
00026
00041 class Object_registry :
00042     public L4::Basic_registry,
00043     public L4::Registry_iface
00044 {
00049     struct Null_handler : L4::Epiface_t<Null_handler, L4::Kobject>
00050     {};
00051
00052 protected:
00053     L4::Cap<L4::Thread> _server;
00054     L4::Cap<L4::Factory> _factory;
00055     L4::Ipc_svr::Server_iface *_sif;
00056
00057 private:
00058     Null_handler _null_handler;
00059
00060 public:
00066     explicit
00067     Object_registry(L4::Ipc_svr::Server_iface *sif)
00068     : _server(L4Re::Env::env()->main_thread()),
00069       _factory(L4Re::Env::env()->factory()),
00070       _sif(sif)
00071     {}
00072
00081     Object_registry(L4::Ipc_svr::Server_iface *sif,
00082                     L4::Cap<L4::Thread> server,
00083                     L4::Cap<L4::Factory> factory)
00084     : _server(server), _factory(factory), _sif(sif)
00085     {}
00086
00087 private:
00088     typedef L4::Ipc_svr::Server_iface Server_iface;
00089     typedef Server_iface::Demand Demand;
00090
00091     L4::Cap<L4::Rcv_endpoint>
00092     _register_ep(L4::Epiface *o, L4::Cap<L4::Rcv_endpoint> ep,
00093                 Demand const &demand)
00094     {
00095         int err = _sif->alloc_buffer_demand(demand);
00096         if (err < 0)
00097             return L4::Cap<L4::Rcv_endpoint>(err | L4_INVALID_CAP_BIT);
00098
00099         err = o->set_server(_sif, ep);
00100         if (err < 0)
00101             return L4::Cap<L4::Rcv_endpoint>(err | L4_INVALID_CAP_BIT);
00102
00103         l4_umword_t id = l4_umword_t(o);
00104         err = l4_error(ep->bind_thread(_server, id));
00105         if (err < 0)
00106             return L4::Cap<L4::Rcv_endpoint>(err | L4_INVALID_CAP_BIT);
00107
00108         return ep;
00109     }
00110
00111     L4::Cap<void> _register_ep(L4::Epiface *o, char const *service,
00112                               Demand const &demand)
00113     {
00114         L4::Cap<L4::Rcv_endpoint> cap = L4Re::Env::env()->get_cap<L4::Rcv_endpoint>(service);
00115         if (!cap.is_valid())

```

```

00116         return cap;
00117
00118     return _register_ep(o, cap, demand);
00119 }
00120
00121 L4::Cap<void> _register_gate(L4::Epiface *o, Demand const &demand)
00122 {
00123     int err = _sif->alloc_buffer_demand(demand);
00124     if (err < 0)
00125         return L4::Cap<void>(err | L4_INVALID_CAP_BIT);
00126
00127     auto cap = L4Re::Util::make_unique_cap<L4::Kobject>();
00128
00129     if (!cap.is_valid())
00130         return cap.get();
00131
00132     l4_umword_t id = l4_umword_t(o);
00133     err = l4_error(_factory->create_gate(cap.get(), _server, id));
00134     if (err < 0)
00135         return L4::Cap<void>(err | L4_INVALID_CAP_BIT);
00136
00137     err = o->set_server(_sif, cap.get(), true);
00138     if (err < 0)
00139         return L4::Cap<void>(err | L4_INVALID_CAP_BIT);
00140
00141     return cap.release();
00142 }
00143
00144 L4::Cap<L4::Irq> _register_irq(L4::Epiface *o,
00145                               Demand const &demand)
00146 {
00147     int err = _sif->alloc_buffer_demand(demand);
00148     if (err < 0)
00149         return L4::Cap<L4::Irq>(err | L4_INVALID_CAP_BIT);
00150
00151     auto cap = L4Re::Util::make_unique_cap<L4::Irq>();
00152
00153     if (!cap.is_valid())
00154         return cap.get();
00155
00156     l4_umword_t id = l4_umword_t(o);
00157     err = l4_error(_factory->create(cap.get()));
00158     if (err < 0)
00159         return L4::Cap<L4::Irq>(err | L4_INVALID_CAP_BIT);
00160
00161     err = o->set_server(_sif, cap.get(), true);
00162     if (err < 0)
00163         return L4::Cap<L4::Irq>(err | L4_INVALID_CAP_BIT);
00164
00165     err = l4_error(cap->bind_thread(_server, id));
00166     if (err < 0)
00167         return L4::Cap<L4::Irq>(err | L4_INVALID_CAP_BIT);
00168
00169     return cap.release();
00170 }
00171
00172 static Demand _get_buffer_demand(L4::Epiface *o)
00173 { return o->get_buffer_demand(); }
00174
00175 template<typename T>
00176 static Demand _get_buffer_demand(T *,
00177     typename L4::Kobject_typeid<typename T::Interface>::Demand
00178     d = typename L4::Kobject_typeid<typename T::Interface>::Demand())
00179 { return d; }
00180
00181 public:
00194 L4::Cap<void> register_obj(L4::Epiface *o, char const *service) override
00195 {
00196     return _register_ep(o, service, _get_buffer_demand(o));
00197 }
00198
00211 L4::Cap<void> register_obj(L4::Epiface *o) override
00212 {
00213     return _register_gate(o, _get_buffer_demand(o));
00214 }
00215
00227 L4::Cap<L4::Irq> register_irq_obj(L4::Epiface *o) override
00228 {
00229     return _register_irq(o, _get_buffer_demand(o));
00230 }
00231
00245 L4::Cap<L4::Rcv_endpoint>
00246 register_obj(L4::Epiface *o, L4::Cap<L4::Rcv_endpoint> ep) override
00247 {
00248     return _register_ep(o, ep, _get_buffer_demand(o));
00249 }
00250

```

```

00251
00262 void unregister_obj(L4::Epiface *o, bool unmap = true) override
00263 {
00264     L4::Epiface::Stored_cap c;
00265
00266     if (!o || !o->obj_cap().is_valid())
00267         return;
00268
00269     c = o->obj_cap();
00270
00271     if (unmap)
00272         L4::Cap<L4::Task>(L4Re::This_task)->unmap(c.fpage(), L4_FP_ALL_SPACES);
00273
00274     // make sure unhandled ipc ends up with the null handler
00275     L4::Thread::Modify_senders todo;
00276     todo.add(~3UL, reinterpret_cast<l4_umword_t>(o),
00277             ~0UL, reinterpret_cast<l4_umword_t>
00278                 (static_cast<L4::Epiface *>(&_null_handler)));
00279     _server->modify_senders(todo);
00280
00281     // we use bit 4 to indicate an internally allocated cap
00282     if (c.managed())
00283         cap_alloc.free(c);
00284
00285     o->set_server(0, L4::Cap<void>::Invalid);
00286 }
00287 };
00288
00292 template< typename LOOP_HOOKS = L4::Ipc_svr::Default_loop_hooks >
00293 class Registry_server : public L4::Server<LOOP_HOOKS>
00294 {
00295 private:
00296     typedef L4::Server<LOOP_HOOKS> Base;
00297     Object_registry _registry;
00298
00299 public:
00305     Registry_server() : _registry(this)
00306     {}
00307
00317     Registry_server(l4_utcb_t *, L4::Cap<L4::Thread> server,
00318                   L4::Cap<L4::Factory> factory) L4_DEPRECATED("Omit UTCB pointer argument")
00319     : _registry(this, server, factory)
00320     {}
00321
00328     Registry_server(L4::Cap<L4::Thread> server,
00329                   L4::Cap<L4::Factory> factory)
00330     : _registry(this, server, factory)
00331     {}
00332
00334     Object_registry const *registry() const { return &_registry; }
00336     Object_registry *registry() { return &_registry; }
00337
00344     void L4_NORETURN loop(l4_utcb_t *utcb = l4_utcb())
00345     { Base::template loop<L4::Runtime_error, Object_registry &>(_registry, utcb); }
00346
00355     template <typename Printer>
00356     void L4_NORETURN loop_dbg(Printer printer, l4_utcb_t *utcb = l4_utcb())
00357     {
00358         Base::template loop_dbg<L4::Runtime_error, Object_registry &, Printer>
00359             (_registry, printer, utcb);
00360     }
00361 };
00362 }

```

16.407 poll_timeout_kipclock

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2012 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *     economic rights: Technische Universität Dresden (Germany)
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include <l4/sys/kip.h>
00010 #include <l4/re/env.h>
00011
00012 namespace L4 {
00013
00035 class Poll_timeout_kipclock
00036 {
00037 public:
00042     Poll_timeout_kipclock(unsigned poll_time_us)

```



```

00043 {
00044     set(poll_time_us);
00045 }
00046
00051 void set(unsigned poll_time_us)
00052 {
00053     _timeout = l4_kip_clock(l4re_kip()) + poll_time_us;
00054     _last_check = true;
00055 }
00056
00065 bool test(bool expression = true)
00066 {
00067     if (!expression)
00068         return false;
00069
00070     return _last_check = l4_kip_clock(l4re_kip()) < _timeout;
00071 }
00072
00077 bool timed_out() const { return !_last_check; }
00078
00079 private:
00080     l4_cpu_time_t _timeout;
00081     bool _last_check;
00082 };
00083 }

```

16.408 l4/re/util/region_mapping File Reference

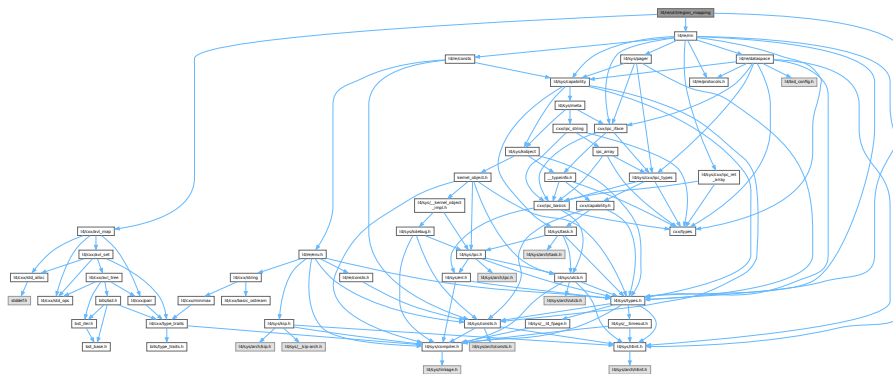
Region handling.

```
#include <l4/cxx/avl_map>
```

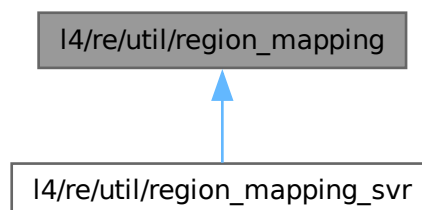
```
#include <l4/sys/types.h>
```

```
#include <l4/re/rm>
```

Include dependency graph for region_mapping:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace [L4Re](#)
[L4Re](#) C++ Interfaces.
- namespace [L4Re::Util](#)
Documentation of the [L4](#) Runtime Environment utility functionality in C++.

16.408.1 Detailed Description

Region handling.

Definition in file [region_mapping](#).

16.409 [region_mapping](#)

[Go to the documentation of this file.](#)

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00010  *      Björn Döbel <doebel@os.inf.tu-dresden.de>
00011  *      economic rights: Technische Universität Dresden (Germany)
00012  *
00013  * License: see LICENSE.spdx (in this directory or the directories above)
00014  */
00015
00016 #pragma once
00017
00018 #include <l4/cxx/avl_map>
00019 #include <l4/sys/types.h>
00020 #include <l4/re/rm>
00021
00022 namespace L4Re { namespace Util {
00023
00024     class Region
00025     {
00026     private:
00027         l4_addr_t _start, _end;
00028         #ifdef CONFIG_L4RE_REGION_INFO
00029         char _dbg_name[40]; // Not a 0-terminating string
00030         unsigned char _dbg_name_len = 0;
00031         static_assert(sizeof(_dbg_name) < 256);
00032         Rm::Offset _dbg_backing_offset = 0;
00033         #endif
00034
00035     public:
00036         Region() noexcept : _start(~0UL), _end(~0UL) {}
00037         Region(l4_addr_t addr) noexcept : _start(addr), _end(addr) {}
00038         Region(l4_addr_t start, l4_addr_t end) noexcept
00039             : _start(start), _end(end) {}
00040         Region(l4_addr_t start, l4_addr_t end,
00041             char const *name, unsigned name_len,
00042             Rm::Offset backing_offset) noexcept
00043             : _start(start), _end(end)
00044             {
00045         #ifdef CONFIG_L4RE_REGION_INFO
00046             _dbg_name_len = name_len > sizeof(_dbg_name)
00047                 ? sizeof(_dbg_name) : name_len;
00048             for (unsigned i = 0; i < _dbg_name_len; ++i)
00049                 _dbg_name[i] = name[i];
00050
00051             _dbg_backing_offset = backing_offset;
00052         #else
00053             (void)name;
00054             (void)name_len;
00055             (void)backing_offset;
00056         #endif
00057         }
00058         l4_addr_t start() const noexcept { return _start; }
00059         l4_addr_t end() const noexcept { return _end; }
00060         unsigned long size() const noexcept { return end() - start() + 1; }

```

```

00061 bool invalid() const noexcept { return _start == ~0UL && _end == ~0UL; }
00062 bool operator < (Region const &o) const noexcept
00063 { return end() < o.start(); }
00064 bool contains(Region const &o) const noexcept
00065 { return o.start() >= start() && o.end() <= end(); }
00066 bool operator == (Region const &o) const noexcept
00067 { return o.start() == start() && o.end() == end(); }
00068 ~Region() noexcept {}
00069
00070 #ifndef CONFIG_L4RE_REGION_INFO
00071 char const *name() const { return _dbg_name; }
00072 unsigned char name_len() const { return _dbg_name_len; }
00073 Rm::Offset backing_offset() const { return _dbg_backing_offset; }
00074 #else
00075 char const *name() const { return "N/A"; }
00076 unsigned char name_len() const { return 3; }
00077 Rm::Offset backing_offset() const { return 0; }
00078 #endif
00079 };
00080
00081
00082 template< typename Hdlr, template<typename T> class Alloc >
00083 class Region_map
00084 {
00085 protected:
00086     typedef cxx::Avl_map<Region, Hdlr, cxx::Lt_functor, Alloc> Tree;
00087
00088     Tree _rm;
00089     Tree _am;
00090
00091 private:
00092     l4_addr_t _start;
00093     l4_addr_t _end;
00094
00095 protected:
00096     void set_limits(l4_addr_t start, l4_addr_t end) noexcept
00097     {
00098         _start = start;
00099         _end = end;
00100     }
00101
00102 public:
00103     typedef typename Tree::Item_type Item;
00104     typedef typename Tree::Node Node;
00105     typedef typename Tree::Key_type Key_type;
00106     typedef Hdlr Region_handler;
00107
00108     typedef typename Tree::Iterator Iterator;
00109     typedef typename Tree::Const_iterator Const_iterator;
00110     typedef typename Tree::Rev_iterator Rev_iterator;
00111     typedef typename Tree::Const_rev_iterator Const_rev_iterator;
00112
00113     Iterator begin() noexcept { return _rm.begin(); }
00114     Const_iterator begin() const noexcept { return _rm.begin(); }
00115     Iterator end() noexcept { return _rm.end(); }
00116     Const_iterator end() const noexcept { return _rm.end(); }
00117
00118     Iterator area_begin() noexcept { return _am.begin(); }
00119     Const_iterator area_begin() const noexcept { return _am.begin(); }
00120     Iterator area_end() noexcept { return _am.end(); }
00121     Const_iterator area_end() const noexcept { return _am.end(); }
00122     Node area_find(Key_type const &c) const noexcept { return _am.find_node(c); }
00123
00124     l4_addr_t min_addr() const noexcept { return _start; }
00125     l4_addr_t max_addr() const noexcept { return _end; }
00126
00127     Region_map(l4_addr_t start, l4_addr_t end) noexcept : _start(start), _end(end) {}
00128
00129     Node find(Key_type const &key) const noexcept
00130     {
00131         Node n = _rm.find_node(key);
00132         if (!n)
00133             return Node();
00134
00135         // 'find' should find any region overlapping with the searched one, the
00136         // caller should check for further requirements
00137         if (0)
00138             if (!n->first.contains(key))
00139                 return Node();
00140
00141         return n;
00142     }
00143
00144     Node lower_bound(Key_type const &key) const noexcept
00145     {
00146         Node n = _rm.lower_bound_node(key);
00147         return n;

```

```

00148     }
00149
00150     Node lower_bound_area(Key_type const &key) const noexcept
00151     {
00152         Node n = _am.lower_bound_node(key);
00153         return n;
00154     }
00155
00156     l4_addr_t attach_area(l4_addr_t addr, unsigned long size,
00157                          L4Re::Rm::Flags flags = L4Re::Rm::Flags(0),
00158                          unsigned char align = L4_PAGESHIFT) noexcept
00159     {
00160         if (size < 2)
00161             return L4_INVALID_ADDR;
00162
00163         Region c;
00164
00165         if (!(flags & L4Re::Rm::F::Search_addr))
00166         {
00167             c = Region(addr, addr + size - 1);
00168             Node r = _am.find_node(c);
00169             if (r)
00170                 return L4_INVALID_ADDR;
00171         }
00172
00173         while (flags & L4Re::Rm::F::Search_addr)
00174         {
00175             if (addr < min_addr() || (addr + size - 1) > max_addr())
00176                 addr = min_addr();
00177
00178             addr = find_free(addr, max_addr(), size, align, flags);
00179             if (addr == L4_INVALID_ADDR)
00180                 return L4_INVALID_ADDR;
00181
00182             c = Region(addr, addr + size - 1);
00183             Node r = _am.find_node(c);
00184             if (!r)
00185                 break;
00186
00187             if (r->first.end() >= max_addr())
00188                 return L4_INVALID_ADDR;
00189
00190             addr = r->first.end() + 1;
00191         }
00192
00193         if (_am.insert(c, Hdlr(typename Hdlr::Dataspace(), 0, 0, flags.region_flags())).second == 0)
00194             return addr;
00195
00196         return L4_INVALID_ADDR;
00197     }
00198
00199     bool detach_area(l4_addr_t addr) noexcept
00200     {
00201         if (_am.remove(addr))
00202             return false;
00203
00204         return true;
00205     }
00206
00207     void *attach(void *addr, unsigned long size, Hdlr const &hdlr,
00208                 L4Re::Rm::Flags attach_flags = L4Re::Rm::Flags(0),
00209                 unsigned char align = L4_PAGESHIFT,
00210                 char const *name = nullptr, unsigned name_len = 0,
00211                 L4Re::Rm::Offset backing_offset = 0) noexcept
00212     {
00213         if (size < 2)
00214             return L4_INVALID_PTR;
00215
00216         l4_addr_t beg, end;
00217         int err = hdlr.map_info(&beg, &end);
00218         if (err > 0)
00219         {
00220             // Mapping address determined by underlying dataspace. Make sure we
00221             // prevent any additional alignment. We already know the place!
00222             beg += hdlr.offset();
00223             end = beg + size - 1U;
00224             align = L4_PAGESHIFT;
00225
00226             // In case of exact mappings, the supplied address must match because
00227             // we cannot remap.
00228             if (!(attach_flags & L4Re::Rm::F::Search_addr)
00229                 && reinterpret_cast<l4_addr_t>(addr) != beg)
00230                 return L4_INVALID_PTR;
00231
00232             // When searching for a suitable address, the start must cover the
00233             // dataspace beginning to "find" the right spot.
00234             if ((attach_flags & L4Re::Rm::F::Search_addr)

```

```

00235         && reinterpret_cast<l4_addr_t>(addr) > beg)
00236         return L4_INVALID_PTR;
00237     }
00238     else if (err == 0)
00239     {
00240         beg = reinterpret_cast<l4_addr_t>(addr);
00241         end = max_addr();
00242     }
00243     else if (err < 0)
00244         return L4_INVALID_PTR;
00245
00246     if (attach_flags & L4Re::Rm::F::In_area)
00247     {
00248         Node r = _am.find_node(Region(beg, beg + size - 1));
00249         if (!r || (r->second.flags() & L4Re::Rm::F::Reserved))
00250             return L4_INVALID_PTR;
00251
00252         end = r->first.end();
00253     }
00254
00255     if (attach_flags & L4Re::Rm::F::Search_addr)
00256     {
00257         beg = find_free(beg, end, size, align, attach_flags);
00258         if (beg == L4_INVALID_ADDR)
00259             return L4_INVALID_PTR;
00260     }
00261
00262     if (!(attach_flags & (L4Re::Rm::F::Search_addr | L4Re::Rm::F::In_area))
        && _am.find_node(Region(beg, beg + size - 1)))
00263         return L4_INVALID_PTR;
00264
00265     if (beg < min_addr() || beg + size - 1 > end)
00266         return L4_INVALID_PTR;
00267
00268     if (_rm.insert(Region(beg, beg + size - 1,
00269                          name, name_len, backing_offset), hdlr).second
        == 0)
00270     {
00271         hdlr.attached(beg, beg + size - 1);
00272         return reinterpret_cast<void*>(beg);
00273     }
00274
00275     return L4_INVALID_PTR;
00276 }
00277
00278 int detach(void *addr, unsigned long sz, unsigned flags,
00279            Region *reg, Hdlr *hdlr) noexcept
00280 {
00281     l4_addr_t a = reinterpret_cast<l4_addr_t>(addr);
00282     Region dr(a, a + sz - 1);
00283     Region res(~0UL, 0);
00284
00285     Node r = find(dr);
00286     if (!r)
00287         return -L4_ENOENT;
00288
00289     Region g = r->first;
00290     Hdlr const &h = r->second;
00291
00292     if (flags & L4Re::Rm::Detach_overlap || dr.contains(g))
00293     {
00294         // Successful removal of the AVL tree item also frees the node.
00295         Hdlr h_copy = h;
00296
00297         if (_rm.remove(g))
00298             return -L4_ENOENT;
00299
00300         if (!(flags & L4Re::Rm::Detach_keep) && (h_copy.flags() & L4Re::Rm::F::Detach_free))
00301             h_copy.free(0, g.size());
00302
00303         h_copy.detached(g.start(), g.end());
00304
00305         if (hdlr)
00306             *hdlr = h_copy;
00307         if (reg)
00308             *reg = g;
00309
00310         if (find(dr))
00311             return Rm::Detached_ds | Rm::Detach_again;
00312         else
00313             return Rm::Detached_ds;
00314     }
00315     else if (dr.start() <= g.start())
00316     {
00317         // Move the start of a region.
00318
00319         if (!(flags & L4Re::Rm::Detach_keep) && (h.flags() & L4Re::Rm::F::Detach_free))

```

```

00322         h.free(0, dr.end() + 1 - g.start());
00323
00324     h.detached(g.start(), dr.end());
00325
00326     unsigned long sz = dr.end() + 1 - g.start();
00327     Item &cn = const_cast<Item &>(*r);
00328     cn.first = Region(dr.end() + 1, g.end(), g.name(), g.name_len(),
00329                     g.backing_offset() + sz);
00330     cn.second = cn.second + sz;
00331     if (hdlr)
00332         *hdlr = Hdlr();
00333     if (reg)
00334         *reg = Region(g.start(), dr.end());
00335     if (find(dr))
00336         return Rm::Kept_ds | Rm::Detach_again;
00337     else
00338         return Rm::Kept_ds;
00339 }
00340 else if (dr.end() >= g.end())
00341 {
00342     // Move the end of a region.
00343
00344     if (!(flags & L4Re::Rm::Detach_keep)
00345         && (h.flags() & L4Re::Rm::F::Detach_free))
00346         h.free(dr.start() - g.start(), g.end() + 1 - dr.start());
00347
00348     h.detached(dr.start(), g.end());
00349
00350     Item &cn = const_cast<Item &>(*r);
00351     cn.first = Region(g.start(), dr.start() - 1, g.name(), g.name_len(),
00352                     g.backing_offset());
00353     if (hdlr)
00354         *hdlr = Hdlr();
00355     if (reg)
00356         *reg = Region(dr.start(), g.end());
00357
00358     if (find(dr))
00359         return Rm::Kept_ds | Rm::Detach_again;
00360     else
00361         return Rm::Kept_ds;
00362 }
00363 else if (g.contains(dr))
00364 {
00365     // Split a single region that contains the new region.
00366
00367     if (!(flags & L4Re::Rm::Detach_keep) && (h.flags() & L4Re::Rm::F::Detach_free))
00368         h.free(dr.start() - g.start(), dr.size());
00369
00370     h.detached(dr.start(), dr.end());
00371
00372     // First move the end off the existing region before the new one.
00373     Item &cn = const_cast<Item &>(*r);
00374     cn.first = Region(g.start(), dr.start() - 1, g.name(), g.name_len(),
00375                     g.backing_offset());
00376
00377     int err;
00378
00379     // Insert a second region for the remaining tail of
00380     // the old existing region.
00381     auto tail_sz = dr.end() + 1 - g.start();
00382     err = _rm.insert(Region(dr.end() + 1, g.end(), g.name(), g.name_len(),
00383                         g.backing_offset() + tail_sz),
00384                     h + tail_sz).second;
00385
00386     if (err)
00387         return err;
00388
00389     if (hdlr)
00390         *hdlr = h;
00391     if (reg)
00392         *reg = dr;
00393     return Rm::Split_ds;
00394 }
00395
00396 return -L4_ENOENT;
00397 }
00398
00399 l4_addr_t find_free(l4_addr_t start, l4_addr_t end, l4_addr_t size,
00400                     unsigned char align, L4Re::Rm::Flags attach_flags) const noexcept;
00401 };
00402
00403 template<typename Hdlr, template<typename T> class Alloc>
00404 l4_addr_t
00405 Region_map<Hdlr, Alloc>::find_free(l4_addr_t start, l4_addr_t end,
00406                                     unsigned long size, unsigned char align, L4Re::Rm::Flags attach_flags) const noexcept
00407 {
00408     l4_addr_t addr = start;

```

```

00409
00410     if (addr == ~0UL || addr < min_addr() || addr >= end)
00411         addr = min_addr();
00412
00413     addr = l4_round_size(addr, align);
00414     Node r;
00415
00416     for (;;)
00417     {
00418         if (addr > 0 && addr - 1 > end - size)
00419             return L4_INVALID_ADDR;
00420
00421         Region c(addr, addr + size - 1);
00422         r = _rm.find_node(c);
00423
00424         if (!r)
00425         {
00426             if (!(attach_flags & L4Re::Rm::F::In_area) && (r = _am.find_node(c)))
00427             {
00428                 if (r->first.end() > end - size)
00429                     return L4_INVALID_ADDR;
00430
00431                 addr = l4_round_size(r->first.end() + 1, align);
00432                 continue;
00433             }
00434
00435             break;
00436         }
00437         else if (r->first.end() > end - size)
00438             return L4_INVALID_ADDR;
00439
00440         addr = l4_round_size(r->first.end() + 1, align);
00441     }
00442
00443     if (!r)
00444         return addr;
00445
00446     return L4_INVALID_ADDR;
00447 }
00448
00449 }}

```

16.410 region_mapping_svr

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include <l4/sys/types.h>
00010 #include <l4/re/rm>
00011 #include <l4/re/util/region_mapping>
00012
00013 namespace L4Re { namespace Util {
00014
00015     template<typename DERIVED, typename Dbg>
00016     struct Rm_server
00017     {
00018     private:
00019         DERIVED *rm() { return static_cast<DERIVED*>(this); }
00020         DERIVED const *rm() const { return static_cast<DERIVED const*>(this); }
00021
00022     public:
00023
00024         l4_ret_t op_attach(L4Re::Rm::Rights, l4_addr_t &start,
00025                             unsigned long size, Rm::Flags flags,
00026                             L4::Ipc::Snd_fpage ds_cap, L4Re::Rm::Offset offs,
00027                             unsigned char align, l4_cap_idx_t client_cap_idx,
00028                             L4::Ipc::String<> name, L4Re::Rm::Offset backing_offset)
00029         {
00030             typename DERIVED::Dataspace ds;
00031
00032             if (!(flags & (Rm::F::Reserved | Rm::F::Kernel)))
00033             {
00034                 if (l4_ret_t r = rm()->validate_ds(ds_cap, flags.region_flags(), &ds))
00035                     return r;
00036             }
00037
00038             size = l4_round_page(size);
00039             l4_addr_t start = l4_trunc_page(_start);

```

```

00048
00049     if (size < L4_PAGESIZE)
00050         return -L4_EINVAL;
00051
00052     Rm::Region_flags r_flags = flags.region_flags();
00053     Rm::Attach_flags a_flags = flags.attach_flags();
00054
00055     typename DERIVED::Region_handler handler(ds, client_cap_idx, offs, r_flags);
00056     start = l4_addr_t(rm()->attach(reinterpret_cast<void*>(start), size,
00057                                     handler, a_flags, align,
00058                                     name.data,
00059                                     // L4::Ipc::String includes terminating '\0'
00060                                     name.length ? name.length - 1 : 0,
00061                                     backing_offset));
00062
00063     if (start == L4_INVALID_ADDR)
00064         return -L4_EADDRNOTAVAIL;
00065
00066     _start = start;
00067     return L4_EOK;
00068 }
00069
00073 l4_ret_t op_free_area(L4Re::Rm::Rights, l4_addr_t start)
00074 {
00075     if (!rm()->detach_area(start))
00076         return -L4_ENOENT;
00077
00078     return L4_EOK;
00079 }
00080
00084 l4_ret_t op_find(L4Re::Rm::Rights, l4_addr_t &addr, unsigned long &size,
00085                 L4Re::Rm::Flags &flags, L4Re::Rm::Offset &offset,
00086                 L4::Cap<L4Re::Dataspace> &m)
00087 {
00088     if (!DERIVED::Have_find)
00089         return -L4_EPERM;
00090
00091     Rm::Flags flag_area { 0 };
00092
00093     typename DERIVED::Node r = rm()->find(Region(addr, addr + size - 1));
00094     if (!r)
00095     {
00096         r = rm()->area_find(Region(addr, addr + size - 1));
00097         if (!r)
00098             return -L4_ENOENT;
00099         flag_area = Rm::F::In_area;
00100     }
00101
00102     addr = r->first.start();
00103     size = r->first.end() + 1 - addr;
00104
00105     flags = r->second.flags() | flag_area;
00106     offset = r->second.offset();
00107     m = L4::Cap<L4Re::Dataspace>(DERIVED::find_res(r->second.memory()));
00108     return L4_EOK;
00109 }
00110
00114 l4_ret_t op_detach(L4Re::Rm::Rights, l4_addr_t addr,
00115                  unsigned long size, unsigned flags,
00116                  l4_addr_t &start, l4_addr_t &rsz,
00117                  l4_cap_idx_t &mem_cap)
00118 {
00119     Region r;
00120     typename DERIVED::Region_handler h;
00121     int err = rm()->detach(reinterpret_cast<void*>(addr), size, flags, &r, &h);
00122     if (err < 0)
00123     {
00124         start = rsz = 0;
00125         mem_cap = L4_INVALID_CAP;
00126         return err;
00127     }
00128
00129     if (r.invalid())
00130     {
00131         start = rsz = 0;
00132         mem_cap = L4_INVALID_CAP;
00133         return -L4_ENOENT;
00134     }
00135
00136     start = r.start();
00137     rsz = r.size();
00138     mem_cap = h.client_cap_idx();
00139     return err;
00140 }
00141
00145 l4_ret_t op_reserve_area(L4Re::Rm::Rights, l4_addr_t &start, unsigned long size,
00146                        L4Re::Rm::Flags flags, unsigned char align)

```



```

00147 {
00148     start = rm()->attach_area(start, size, flags, align);
00149     if (start == L4_INVALID_ADDR)
00150         return -L4_EADDRNOTAVAIL;
00151     return L4_EOK;
00152 }
00153
00157 l4_ret_t op_get_regions(L4Re::Rm::Rights, l4_addr_t addr,
00158                        L4::Ipc::Ret_array<L4Re::Rm::Region> regions)
00159 {
00160     typename DERIVED::Node r;
00161     unsigned num = 0;
00162     while ((r = rm()->lower_bound(Region(addr))))
00163     {
00164         Rm::Region &x = regions.value[num];
00165         x.start = r->first.start();
00166         x.end = r->first.end();
00167         x.flags = r->second.flags();
00168
00169         if (++num >= regions.max)
00170             break;
00171
00172         if (x.end >= rm()->max_addr())
00173             break;
00174         addr = x.end + 1;
00175     }
00176     return num;
00177 }
00178
00182 l4_ret_t op_get_areas(L4Re::Rm::Rights, l4_addr_t addr,
00183                      L4::Ipc::Ret_array<L4Re::Rm::Area> areas)
00184 {
00185     typename DERIVED::Node r;
00186     unsigned num = 0;
00187     while ((r = rm()->lower_bound_area(Region(addr))))
00188     {
00189         Rm::Area &x = areas.value[num];
00190         x.start = r->first.start();
00191         x.end = r->first.end();
00192
00193         if (++num >= areas.max)
00194             break;
00195
00196         if (x.end >= rm()->max_addr())
00197             break;
00198         addr = x.end + 1;
00199     }
00200
00201     return num;
00202 }
00203
00204 private:
00206 static void pager_set_result(L4::Ipc::Opt<L4::Ipc::Snd_fpage> *fp,
00207                             L4::Ipc::Snd_fpage const &f)
00208 { *fp = f; }
00209
00210 static void pager_set_result(L4::Ipc::Opt<L4::Ipc::Snd_fpage> *, ...)
00211 {}
00212
00213 public:
00214
00218 l4_ret_t op_io_page_fault(L4::Io_pager::Rights, l4_fpage_t, l4_umword_t,
00219                           L4::Ipc::Opt<L4::Ipc::Snd_fpage> &)
00220 {
00221     // generate exception
00222     return -L4_ENOMEM;
00223 }
00224
00225 l4_ret_t op_page_fault(L4::Pager::Rights, l4_umword_t addr, l4_umword_t pc,
00226                       L4::Ipc::Opt<L4::Ipc::Snd_fpage> &fp)
00227 {
00228     Dbg(Dbg::Server).printf("page fault: %lx pc=%lx\n", addr, pc);
00229
00230     bool need_w = addr & 2;
00231     bool need_x = addr & 4;
00232
00233     typename DERIVED::Node n = rm()->find(addr);
00234
00235     if (!n || !n->second.memory())
00236     {
00237         Dbg(Dbg::Warn, "rm").printf("unhandled %s page fault at 0x%lx pc=0x%lx\n",
00238                                     need_w ? "write" :
00239                                     need_x ? "instruction" : "read", addr, pc);
00240         // generate exception
00241         return -L4_ENOMEM;
00242     }

```

```

00243
00244     if (!(n->second.flags() & L4Re::Rm::F::W) && need_w)
00245     {
00246         Dbg(Dbg::Warn, "rm").printf("write page fault in readonly region at 0x%lx pc=0x%lx\n",
00247                                     addr, pc);
00248         // generate exception
00249         return -L4_EACCESS;
00250     }
00251
00252     if (!(n->second.flags() & L4Re::Rm::F::X) && need_x)
00253     {
00254         Dbg(Dbg::Warn, "rm").printf("instruction page fault in non-exec region at 0x%lx pc=0x%lx\n",
00255                                     addr, pc);
00256         // generate exception
00257         return -L4_EACCESS;
00258     }
00259
00260     // This check is optional but avoids doing a map operation that will not
00261     // work. We shall never get here from a page-fault but only from
00262     // artificial page handling.
00263     if (n->second.flags() & (L4Re::Rm::F::Kernel | L4Re::Rm::F::Reserved))
00264     {
00265         Dbg(Dbg::Warn, "rm").printf("page fault handling in kernel-memory provided region or reserved
00266                                     region at 0x%lx pc=0x%lx\n",
00267                                     addr, pc);
00268         // generate exception
00269         return -L4_ENODEV;
00270     }
00271
00272     typename DERIVED::Region_handler::Map_result map_res;
00273     if (int err = n->second.map(addr, n->first, need_w, &map_res))
00274     {
00275         Dbg(Dbg::Warn, "rm").printf("mapping for page fault failed with error %d at 0x%lx pc=0x%lx\n",
00276                                     err, addr, pc);
00277         // generate exception
00278         return -L4_ENOMEM;
00279     }
00280
00281     pager_set_result(&fp, map_res);
00282     return L4_EOK;
00283 }
00284
00285 long op_get_info(L4Re::Rm::Rights, l4_addr_t addr,
00286                 L4::Ipc::String<char> &name, L4Re::Rm::Offset &backing_offset)
00287 {
00288     #ifdef CONFIG_L4RE_REGION_INFO
00289         typename DERIVED::Node r = rm()->find(Region(addr));
00290         if (!r)
00291             return -L4_ENOENT;
00292         backing_offset = r->first.backing_offset();
00293         unsigned long i;
00294         char const *src = r->first.name();
00295         unsigned src_len = r->first.name_len();
00296         for (i = 0; i < src_len && i < name.length - 1; ++i)
00297             name.data[i] = src[i];
00298         name.length = i + 1;
00299         name.data[i] = '\0';
00300         return L4_EOK;
00301     #else
00302         (void)addr;
00303         (void)name;
00304         (void)backing_offset;
00305         return -L4_ENOSYS;
00306     #endif
00307 }
00308
00309 }

```

16.411 reply_cap_hooks

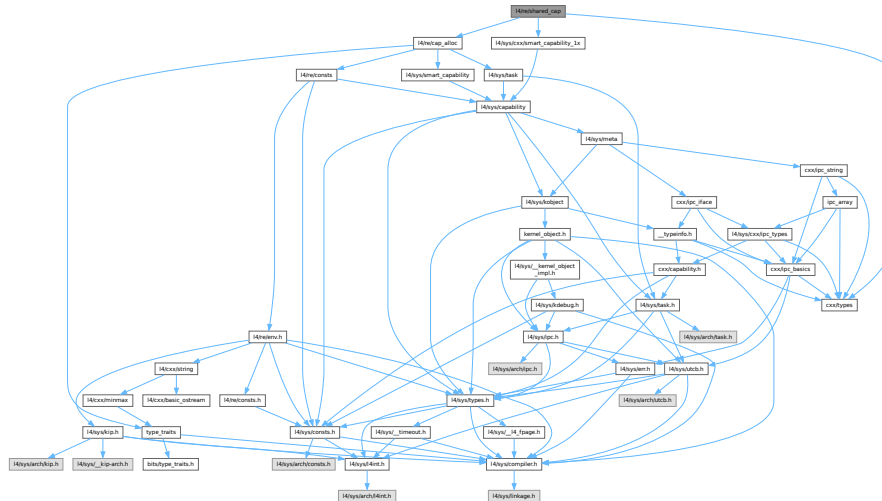
```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002
00003 #pragma once
00004
00005 #include <l4/cxx/result>
00006 #include <l4/re/util/cap_alloc>
00007
00008 namespace L4Re { namespace Util {
00009
00010     template<typename HOOKS>
00011     class Reply_cap_hooks : public HOOKS
00012     {

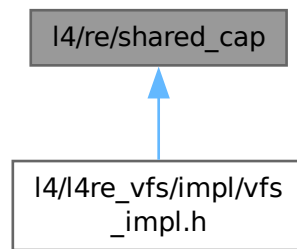
```

16.412 I4/re/shared_cap File Reference

```
#include <l4/re/cap_alloc>
#include <l4/sys/cxx/smart_capability_lx>
#include <l4/sys/cxx/types>
Include dependency graph for shared_cap:
```



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace [L4Re](#)
[L4Re](#) C++ Interfaces.

Typedefs

- `template<typename T>`
`using L4Re::Shared_cap = L4::Detail::Shared_cap_impl<T, L4Re::Smart_count_cap<L4_FP_ALL_SPACES>>`
Shared capability that implements automatic free and unmap of the capability selector.
- `template<typename T>`
`using L4Re::Shared_del_cap = L4::Detail::Shared_cap_impl<T, L4Re::Smart_count_cap<L4_FP_DELETE_OBJ>>`
Shared capability that implements automatic free and unmap+delete of the capability selector.

Functions

- `template<typename T>`
`Shared_cap< T > L4Re::make_shared_cap (L4Re::Cap_alloc *ca)`
Allocate a capability slot and wrap it in a [Shared_cap](#).
- `template<typename T, typename U>`
`Shared_cap< T > L4Re::shared_cap_cast (Shared_cap< U > const &from) noexcept`
Create a new shared capability by an explicit cast from another shared capability.
- `template<typename T, typename U>`
`Shared_cap< T > L4Re::shared_cap_cast (Shared_cap< U > &&from) noexcept`
Create a new shared capability by an explicit cast from another shared capability with move semantics.
- `template<typename T, typename U>`
`Shared_cap< T > L4Re::shared_cap_reinterpret_cast (Shared_cap< U > const &from) noexcept`
Create a new shared capability by a reinterpret cast from another shared capability.
- `template<typename T, typename U>`
`Shared_cap< T > L4Re::shared_cap_reinterpret_cast (Shared_cap< U > &&from) noexcept`
Create a new shared capability by a reinterpret cast from another shared capability with move semantics.
- `template<typename T, typename U>`
`Shared_cap< T > L4Re::shared_cap_dynamic_cast (Shared_cap< U > const &from) noexcept`
Create a new shared capability by a dynamic cast from another shared capability.

- `template<typename T, typename U>`
`Shared_cap< T > L4Re::shared_cap_dynamic_cast (Shared_cap< U > &&from) noexcept`
Create a new shared capability by an dynamic cast from another shared capability with move semantics.
- `template<typename T>`
`Shared_del_cap< T > L4Re::make_shared_del_cap (L4Re::Cap_alloc *ca)`
Allocate a capability slot and wrap it in a `Shared_del_cap`.
- `template<typename T, typename U>`
`Shared_del_cap< T > L4Re::shared_del_cap_cast (Shared_del_cap< U > const &from) noexcept`
Create a new shared capability by an explicit cast from another shared capability.
- `template<typename T, typename U>`
`Shared_del_cap< T > L4Re::shared_del_cap_cast (Shared_del_cap< U > &&from) noexcept`
Create a new shared capability by an explicit cast from another shared capability with move semantics.
- `template<typename T, typename U>`
`Shared_del_cap< T > L4Re::shared_del_cap_reinterpret_cast (Shared_del_cap< U > const &from) noexcept`
Create a new shared capability by a reinterpret cast from another shared capability.
- `template<typename T, typename U>`
`Shared_del_cap< T > L4Re::shared_del_cap_reinterpret_cast (Shared_del_cap< U > &&from) noexcept`
Create a new shared capability by a reinterpret cast from another shared capability with move semantics.
- `template<typename T, typename U>`
`Shared_del_cap< T > L4Re::shared_del_cap_dynamic_cast (Shared_del_cap< U > const &from) noexcept`
Create a new shared capability by a dynamic cast from another shared capability.
- `template<typename T, typename U>`
`Shared_del_cap< T > L4Re::shared_del_cap_dynamic_cast (Shared_del_cap< U > &&from) noexcept`
Create a new shared capability by an dynamic cast from another shared capability with move semantics.

16.412.1 Detailed Description

Shared_cap / Shared_del_cap.

Definition in file [shared_cap](#).

16.413 shared_cap

[Go to the documentation of this file.](#)

```
00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2018 Alexander Warg <alexander.warg@kernkonzept.com>
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011
00012 #pragma once
00013
00014 #include <l4/re/cap_alloc>
00015 #include <l4/sys/cxx/smart_capability_1x>
00016 #include <l4/sys/cxx/types>
00017
00018 namespace L4Re {
00019
00033 template< typename T >
00034 using Shared_cap
00035     = L4::Detail::Shared_cap_impl<T, L4Re::Smart_count_cap<L4_FP_ALL_SPACES>;
00037 template< typename T >
00038 using shared_cap [[deprecated("Use L4Re::Shared_cap.")]]
00039     = L4::Detail::Shared_cap_impl<T, L4Re::Smart_count_cap<L4_FP_ALL_SPACES>;
00040
00050 template< typename T >
00051 Shared_cap<T>
```

```

00052 make_shared_cap(L4Re::Cap_alloc *ca)
00053 { return Shared_cap<T>(ca->alloc<T>(), ca); }
00054
00067 template<typename T, typename U>
00068 Shared_cap<T> shared_cap_cast(Shared_cap<U> const &from) noexcept
00069 {
00070     auto to = L4::cap_cast<T>(from.get());
00071     return Shared_cap<T>(from, to);
00072 }
00073
00086 template<typename T, typename U>
00087 Shared_cap<T> shared_cap_cast(Shared_cap<U> &&from) noexcept
00088 {
00089     auto to = L4::cap_cast<T>(from.get());
00090     return Shared_cap<T>(L4::Types::move(from), to);
00091 }
00092
00103 template<typename T, typename U>
00104 Shared_cap<T> shared_cap_reinterpret_cast(Shared_cap<U> const &from) noexcept
00105 {
00106     auto to = L4::cap_reinterpret_cast<T>(from.get());
00107     return Shared_cap<T>(from, to);
00108 }
00109
00122 template<typename T, typename U>
00123 Shared_cap<T> shared_cap_reinterpret_cast(Shared_cap<U> &&from) noexcept
00124 {
00125     auto to = L4::cap_reinterpret_cast<T>(from.get());
00126     return Shared_cap<T>(L4::Types::move(from), to);
00127 }
00128
00139 template<typename T, typename U>
00140 Shared_cap<T> shared_cap_dynamic_cast(Shared_cap<U> const &from) noexcept
00141 {
00142     if (auto to = L4::cap_dynamic_cast<T>(from.get()))
00143         return Shared_cap<T>(from, to);
00144     else
00145         return Shared_cap<T>();
00146 }
00147
00160 template<typename T, typename U>
00161 Shared_cap<T> shared_cap_dynamic_cast(Shared_cap<U> &&from) noexcept
00162 {
00163     if (auto to = L4::cap_dynamic_cast<T>(from.get()))
00164         return Shared_cap<T>(L4::Types::move(from), to);
00165     else
00166         return Shared_cap<T>();
00167 }
00168
00185 template< typename T >
00186 using Shared_del_cap
00187     = L4::Detail::Shared_cap_impl<T, L4Re::Smart_count_cap<L4_FP_DELETE_OBJ>;
00189 template<typename T>
00190 using shared_del_cap [[deprecated("Use L4Re::Shared_del_cap.")]]
00191     = L4::Detail::Shared_cap_impl<T, L4Re::Smart_count_cap<L4_FP_DELETE_OBJ>;
00192
00202 template< typename T >
00203 Shared_del_cap<T>
00204 make_shared_del_cap(L4Re::Cap_alloc *ca)
00205 { return Shared_del_cap<T>(ca->alloc<T>(), ca); }
00206
00219 template<typename T, typename U>
00220 Shared_del_cap<T> shared_del_cap_cast(Shared_del_cap<U> const &from) noexcept
00221 {
00222     auto to = L4::cap_cast<T>(from.get());
00223     return Shared_del_cap<T>(from, to);
00224 }
00225
00238 template<typename T, typename U>
00239 Shared_del_cap<T> shared_del_cap_cast(Shared_del_cap<U> &&from) noexcept
00240 {
00241     auto to = L4::cap_cast<T>(from.get());
00242     return Shared_del_cap<T>(L4::Types::move(from), to);
00243 }
00244
00255 template<typename T, typename U>
00256 Shared_del_cap<T>
00257 shared_del_cap_reinterpret_cast(Shared_del_cap<U> const &from) noexcept
00258 {
00259     auto to = L4::cap_reinterpret_cast<T>(from.get());
00260     return Shared_del_cap<T>(from, to);
00261 }
00262
00275 template<typename T, typename U>
00276 Shared_del_cap<T>
00277 shared_del_cap_reinterpret_cast(Shared_del_cap<U> &&from) noexcept
00278 {

```

```

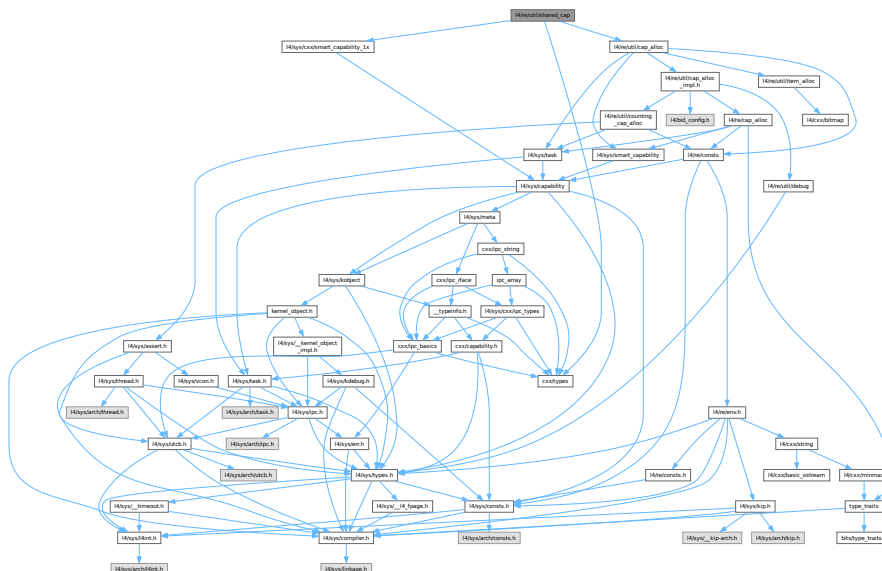
00279     auto to = L4::cap_reinterpret_cast<T>(from.get());
00280     return Shared_del_cap<T>(L4::Types::move(from), to);
00281 }
00282
00293 template<typename T, typename U>
00294 Shared_del_cap<T>
00295 shared_del_cap_dynamic_cast(Shared_del_cap<U> const &from) noexcept
00296 {
00297     if (auto to = L4::cap_dynamic_cast<T>(from.get()))
00298         return Shared_del_cap<T>(from, to);
00299     else
00300         return Shared_del_cap<T>();
00301 }
00302
00315 template<typename T, typename U>
00316 Shared_del_cap<T>
00317 shared_del_cap_dynamic_cast(Shared_del_cap<U> &&from) noexcept
00318 {
00319     if (auto to = L4::cap_dynamic_cast<T>(from.get()))
00320         return Shared_del_cap<T>(L4::Types::move(from), to);
00321     else
00322         return Shared_del_cap<T>();
00323 }
00324
00325 } // namespace L4Re

```

16.414 l4/re/util/shared_cap File Reference

Shared cap / Shared del cap.

```
#include <l4/re/util/cap_alloc>
#include <l4/sys/cxx/smart_capability_lx>
#include <l4/sys/cxx/types>
Include dependency graph for shared cap:
```



Create a new shared capability by a dynamic cast from another shared capability.

- `template<typename T, typename U>`
`Shared_cap< T > L4Re::Util::shared_cap_dynamic_cast (Shared_cap< U > &&from) noexcept`

Create a new shared capability by an dynamic cast from another shared capability with move semantics.

- `template<typename T>`
`Shared_del_cap< T > L4Re::Util::make_shared_del_cap ()`

Allocate a capability slot and wrap it in a `Shared_del_cap`.

- `template<typename T, typename U>`
`Shared_del_cap< T > L4Re::Util::shared_del_cap_cast (Shared_del_cap< U > const &from) noexcept`

Create a new shared capability by an explicit cast from another shared capability.

- `template<typename T, typename U>`
`Shared_del_cap< T > L4Re::Util::shared_del_cap_cast (Shared_del_cap< U > &&from) noexcept`

Create a new shared capability by an explicit cast from another shared capability with move semantics.

- `template<typename T, typename U>`
`Shared_del_cap< T > L4Re::Util::shared_del_cap_reinterpret_cast (Shared_del_cap< U > const &from)`
`noexcept`

Create a new shared capability by a reinterpret cast from another shared capability.

- `template<typename T, typename U>`
`Shared_del_cap< T > L4Re::Util::shared_del_cap_reinterpret_cast (Shared_del_cap< U > &&from) noex-`
`cept`

Create a new shared capability by a reinterpret cast from another shared capability with move semantics.

- `template<typename T, typename U>`
`Shared_del_cap< T > L4Re::Util::shared_del_cap_dynamic_cast (Shared_del_cap< U > const &from)`
`noexcept`

Create a new shared capability by a dynamic cast from another shared capability.

- `template<typename T, typename U>`
`Shared_del_cap< T > L4Re::Util::shared_del_cap_dynamic_cast (Shared_del_cap< U > &&from) noex-`
`cept`

Create a new shared capability by an dynamic cast from another shared capability with move semantics.

16.414.1 Detailed Description

Shared_cap / Shared_del_cap.

Definition in file [shared_cap](#).

16.415 shared_cap

[Go to the documentation of this file.](#)

```
00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2017 Alexander Warg <alexander.warg@kernkonzept.com>
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011
00012 #pragma once
00013
00014 #include <l4/re/util/cap_alloc>
00015 #include <l4/sys/cxx/smart_capability_lx>
00016 #include <l4/sys/cxx/types>
00017
00018 namespace L4Re { namespace Util {
00019
00048 template< typename T >
00049 using Shared_cap
00050     = L4::Detail::Shared_cap_impl<T, L4Re::Util::Smart_count_cap<L4_FP_ALL_SPACES>;
```

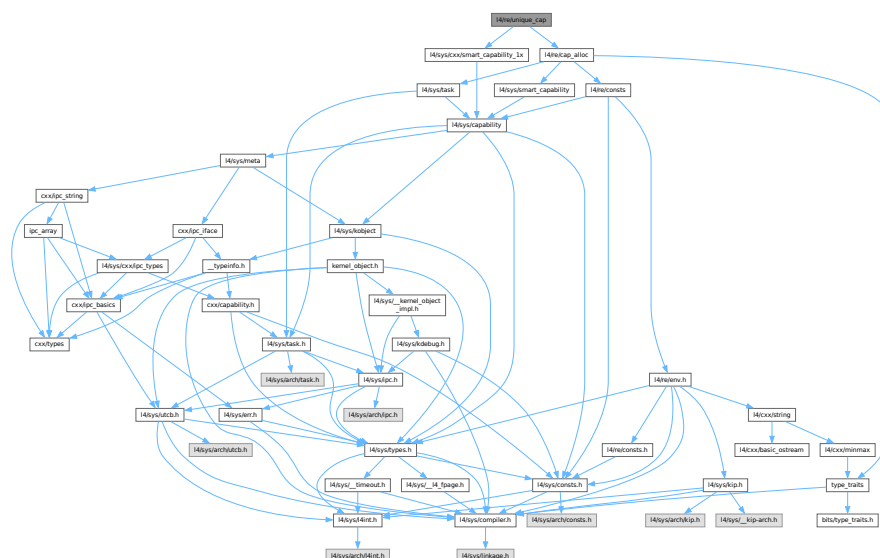
```

00052 template< typename T >
00053 using shared_cap [[deprecated("Use L4Re::Util::Shared_cap.")]]
00054 = L4::Detail::Shared_cap_impl<T, L4Re::Util::Smart_count_cap<L4_FP_ALL_SPACES>;
00055
00061 template< typename T >
00062 Shared_cap<T>
00063 make_shared_cap()
00064 { return Shared_cap<T>(cap_alloc.alloc<T>()); }
00065
00078 template<typename T, typename U>
00079 Shared_cap<T> shared_cap_cast(Shared_cap<U> const &from) noexcept
00080 {
00081     auto to = L4::cap_cast<T>(from.get());
00082     return Shared_cap<T>(from, to);
00083 }
00084
00097 template<typename T, typename U>
00098 Shared_cap<T> shared_cap_cast(Shared_cap<U> &&from) noexcept
00099 {
00100     auto to = L4::cap_cast<T>(from.get());
00101     return Shared_cap<T>(L4::Types::move(from), to);
00102 }
00103
00114 template<typename T, typename U>
00115 Shared_cap<T> shared_cap_reinterpret_cast(Shared_cap<U> const &from) noexcept
00116 {
00117     auto to = L4::cap_reinterpret_cast<T>(from.get());
00118     return Shared_cap<T>(from, to);
00119 }
00120
00133 template<typename T, typename U>
00134 Shared_cap<T> shared_cap_reinterpret_cast(Shared_cap<U> &&from) noexcept
00135 {
00136     auto to = L4::cap_reinterpret_cast<T>(from.get());
00137     return Shared_cap<T>(L4::Types::move(from), to);
00138 }
00139
00150 template<typename T, typename U>
00151 Shared_cap<T> shared_cap_dynamic_cast(Shared_cap<U> const &from) noexcept
00152 {
00153     if (auto to = L4::cap_dynamic_cast<T>(from.get()))
00154         return Shared_cap<T>(from, to);
00155     else
00156         return Shared_cap<T>();
00157 }
00158
00171 template<typename T, typename U>
00172 Shared_cap<T> shared_cap_dynamic_cast(Shared_cap<U> &&from) noexcept
00173 {
00174     if (auto to = L4::cap_dynamic_cast<T>(from.get()))
00175         return Shared_cap<T>(L4::Types::move(from), to);
00176     else
00177         return Shared_cap<T>();
00178 }
00179
00214 template< typename T >
00215 using Shared_del_cap
00216 = L4::Detail::Shared_cap_impl<T, L4Re::Util::Smart_count_cap<L4_FP_DELETE_OBJ>;
00218 template< typename T >
00219 using Shared_del_cap [[deprecated("Use L4Re::Util::Shared_del_cap.")]]
00220 = L4::Detail::Shared_cap_impl<T, L4Re::Util::Smart_count_cap<L4_FP_DELETE_OBJ>;
00221
00227 template< typename T >
00228 Shared_del_cap<T>
00229 make_shared_del_cap()
00230 { return Shared_del_cap<T>(cap_alloc.alloc<T>()); }
00231
00244 template<typename T, typename U>
00245 Shared_del_cap<T> shared_del_cap_cast(Shared_del_cap<U> const &from) noexcept
00246 {
00247     auto to = L4::cap_cast<T>(from.get());
00248     return Shared_del_cap<T>(from, to);
00249 }
00250
00263 template<typename T, typename U>
00264 Shared_del_cap<T> shared_del_cap_cast(Shared_del_cap<U> &&from) noexcept
00265 {
00266     auto to = L4::cap_cast<T>(from.get());
00267     return Shared_del_cap<T>(L4::Types::move(from), to);
00268 }
00269
00280 template<typename T, typename U>
00281 Shared_del_cap<T>
00282 shared_del_cap_reinterpret_cast(Shared_del_cap<U> const &from) noexcept
00283 {
00284     auto to = L4::cap_reinterpret_cast<T>(from.get());
00285     return Shared_del_cap<T>(from, to);

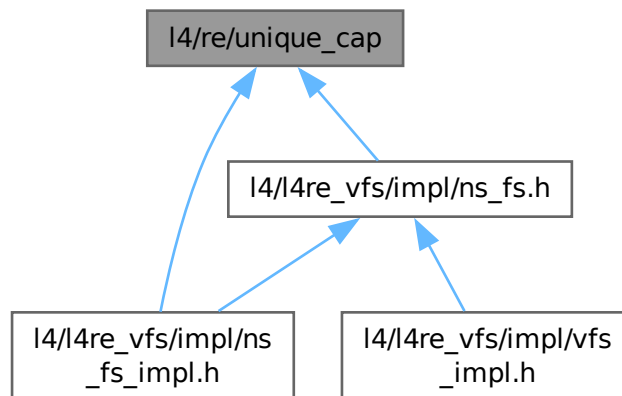
```

16.416 I4/re/unique_cap File Reference

```
#include <l4/re/cap_alloc>
#include <l4/sys/cxx/smart_capability_1x>
Include dependency graph for unique_cap:
```



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace [L4Re](#)
[L4Re](#) C++ Interfaces.

Typedefs

- `template<typename T>`
`using L4Re::Unique_cap = L4::Detail::Unique_cap_impl<T, L4Re::Smart_cap_auto<L4_FP_ALL_SPACES>>`
Unique capability that implements automatic free and unmap of the capability selector.
- `template<typename T>`
`using L4Re::Unique_del_cap = L4::Detail::Unique_cap_impl<T, L4Re::Smart_cap_auto<L4_FP_DELETE_OBJ>>`
Unique capability that implements automatic free and unmap+delete of the capability selector.

Functions

- `template<typename T>`
`Unique_cap< T > L4Re::make_unique_cap (L4Re::Cap_alloc *ca)`
Allocate a capability slot and wrap it in an [Unique_cap](#).
- `template<typename T>`
`Unique_del_cap< T > L4Re::make_unique_del_cap (L4Re::Cap_alloc *ca)`
Allocate a capability slot and wrap it in an [Unique_del_cap](#).

16.416.1 Detailed Description

`Unique_cap` / `Unique_del_cap`.

Definition in file [unique_cap](#).

16.419 unique_cap

[Go to the documentation of this file.](#)

```

00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2017 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00008 #pragma once
00009
00010 #include <l4/re/util/cap_alloc>
00011 #include <l4/sys/cxx/smart_capability_lx>
00012
00013 namespace L4Re { namespace Util {
00014
00015     template< typename T >
00016     using Unique_cap
00017     = L4::Detail::Unique_cap_impl<T, L4Re::Util::Smart_cap_auto<L4_FP_ALL_SPACES>;
00018     template< typename T >
00019     using unique_cap [[deprecated("Use L4Re::Util::Unique_cap.")]]
00020     = L4::Detail::Unique_cap_impl<T, L4Re::Util::Smart_cap_auto<L4_FP_ALL_SPACES>;
00021
00022     template< typename T >
00023     Unique_cap<T>
00024     make_unique_cap()
00025     { return Unique_cap<T>(cap_alloc.alloc<T>()); }
00026
00027     template< typename T >
00028     using Unique_del_cap
00029     = L4::Detail::Unique_cap_impl<T, L4Re::Util::Smart_cap_auto<L4_FP_DELETE_OBJ>;
00030     template< typename T >
00031     using unique_del_cap [[deprecated("Use L4Re::Util::Unique_del_cap.")]]
00032     = L4::Detail::Unique_cap_impl<T, L4Re::Util::Smart_cap_auto<L4_FP_DELETE_OBJ>;
00033
00034     template< typename T >
00035     Unique_del_cap<T>
00036     make_unique_del_cap()
00037     { return Unique_del_cap<T>(cap_alloc.alloc<T>()); }
00038
00039 }}
00040
00041

```

16.420 vcon_svr

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2011 Alexander Warg <warg@os.inf.tu-dresden.de>,
00004  *
00005  * Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00006  * Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *
00008  * economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012
00013 #pragma once
00014
00015 #include <l4/sys/types.h>
00016 #include <l4/sys/vcon>
00017 #include <l4/sys/cxx/ipc_legacy>
00018 #include <l4/cxx/minmax>
00019
00020 namespace L4Re { namespace Util {
00021
00022     template< typename SVR >
00023     class Vcon_svr
00024     {
00025     public:
00026         L4_RPC_LEGACY_DISPATCH(L4::Vcon);
00027
00028         l4_msgtag_t op_dispatch(l4_utcb_t *utcb, l4_msgtag_t tag, L4::Vcon::Rights)
00029         {
00030             l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00031             L4::Opcode op = m->mr[0] & 0xFFFF;
00032
00033             switch (op)
00034             {
00035             case L4_VCON_WRITE_OP:
00036                 if (tag.words() < 3)
00037                     return l4_msgtag(-L4_ENOREPLY, 0, 0, 0);
00038             }
00039         }
00040     };
00041

```

```

00051
00052     this_vcon()->vcon_write(reinterpret_cast<char const *>(&m->mr[2]),
00053                             m->mr[1]);
00054     return l4_msgtag(-L4_ENOREPLY, 0, 0, 0);
00055
00056 case L4_VCON_SET_ATTR_OP:
00057 {
00058     if (tag.words() < 4)
00059         return l4_msgtag(-L4_EINVAL, 0, 0, 0);
00060
00061     auto attr = reinterpret_cast<l4_vcon_attr_t const *>(&m->mr[1]);
00062     return l4_msgtag(this_vcon()->vcon_set_attr(attr), 0, 0, 0);
00063 }
00064 case L4_VCON_GET_ATTR_OP:
00065 {
00066     auto attr = reinterpret_cast<l4_vcon_attr_t *>(&m->mr[1]);
00067     return l4_msgtag(this_vcon()->vcon_get_attr(attr), 4, 0, 0);
00068 }
00069 case L4_VCON_READ_OP:
00070     break;
00071
00072 default:
00073     return l4_msgtag(-L4_ENOSYS, 0, 0, 0);
00074 }
00075
00076 unsigned const max_size = L4_VCON_READ_SIZE;
00077 char buf[max_size];
00078
00079 unsigned size = cxx::min<unsigned>(m->mr[0] » 16, max_size);
00080
00081 // Hmm, could we avoid the double copy here?
00082 l4_umword_t v = this_vcon()->vcon_read(buf, size);
00083 unsigned bytes = v & L4_VCON_READ_SIZE_MASK;
00084
00085 if (bytes <= size)
00086     v |= L4_VCON_READ_STAT_DONE;
00087 else
00088     // In case of size == max_size, the returned bytes value is greater than
00089     // the buffer.
00090     bytes = size;
00091
00092 m->mr[0] = v;
00093 __builtin_memcpy(&m->mr[1], buf, bytes);
00094
00095 return l4_msgtag(0, l4_bytes_to_mwords(bytes) + 1, 0, 0);
00096 }
00097
00098 unsigned vcon_read(char *buf, unsigned size) noexcept;
00099 void vcon_write(const char *buf, unsigned size) noexcept;
00100 int vcon_set_attr(l4_vcon_attr_t const *) noexcept
00101 { return -L4_EOK; }
00102 int vcon_get_attr(l4_vcon_attr_t *attr) noexcept
00103 {
00104     attr->l_flags = attr->o_flags = attr->i_flags = 0;
00105     return -L4_EOK;
00106 }
00107
00108 private:
00109     SVR const *this_vcon() const { return static_cast<SVR const *>(this); }
00110     SVR *this_vcon() { return static_cast<SVR *>(this); }
00111 };
00112
00113 }

```

16.421 goos_fb

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009 #pragma once
00010
00011 #include <l4/re/env>
00012 #include <l4/re/namespace>
00013 #include <l4/re/rm>
00014 #include <l4/re/util/cap_alloc>
00015 #include <l4/re/util/env_ns>
00016 #include <l4/re/util/video/goos_fb>
00017 #include <l4/re/video/goos>

```



```

00018
00019 namespace L4Re { namespace Util { namespace Video {
00020
00021 class Goos_fb
00022 {
00023 private:
00024     L4::Cap<L4Re::Video::Goos> _goos;
00025     L4Re::Video::View _view;
00026     L4::Cap<L4Re::Dataspace> _buffer;
00027
00028     enum Flags
00029     {
00030         F_dyn_buffer = 0x01,
00031         F_dyn_view   = 0x02,
00032         F_dyn_goos   = 0x04,
00033     };
00034     unsigned _flags;
00035
00036     unsigned _buffer_index;
00037
00038 private:
00039     long init()
00040     {
00041         using namespace L4Re::Video;
00042
00043         Goos::Info gi;
00044         long ret = _goos->info(&gi);
00045         if (ret < 0)
00046             return ret;
00047
00048         if (gi.has_dynamic_views())
00049         {
00050             ret = _goos->create_view(&_view);
00051             if (ret < 0)
00052                 return ret;
00053
00054             _flags |= F_dyn_view;
00055         }
00056         else // we just assume view 0 to be our's and ignore other possible views
00057             _view = _goos->view(0);
00058
00059         View::Info vi;
00060         ret = _view.info(&vi);
00061         if (ret < 0)
00062             return ret;
00063
00064         _buffer = cap_alloc.alloc<L4Re::Dataspace>();
00065         if (!_buffer)
00066             return -L4_ENOMEM;
00067
00068         if (vi.has_static_buffer())
00069         {
00070             ret = _goos->get_static_buffer(vi.buffer_index, _buffer);
00071             if (ret < 0)
00072                 return ret;
00073         }
00074         else
00075         {
00076             unsigned long buffer_sz = gi.pixel_info.bytes_per_pixel() * gi.width
00077                                     * gi.height;
00078             ret = _goos->create_buffer(buffer_sz, _buffer);
00079             if (ret < 0)
00080                 return ret;
00081
00082             _buffer_index = static_cast<unsigned>(ret);
00083             _flags |= F_dyn_buffer;
00084
00085             // use the allocated buffer, at offset 0
00086             vi.buffer_index = _buffer_index;
00087             vi.buffer_offset = 0;
00088             vi.pixel_info = gi.pixel_info;
00089             vi.bytes_per_line = gi.width * gi.pixel_info.bytes_per_pixel();
00090
00091             // we want a fullscreen view
00092             vi.xpos = 0;
00093             vi.ypos = 0;
00094             vi.width = gi.width;
00095             vi.height = gi.height;
00096
00097             ret = _view.set_info(vi);
00098             if (ret < 0)
00099                 return ret;
00100
00101             ret = _view.push_top();
00102             if (ret < 0)
00103                 return ret;
00104         }

```

```

00105
00106     return 0;
00107 }
00108
00109 Goos_fb(Goos_fb const &);
00110 void operator = (Goos_fb const &);
00111
00112 public:
00113     Goos_fb()
00114     : _goos(L4_INVALID_CAP), _buffer(L4_INVALID_CAP), _flags(0), _buffer_index(0)
00115     {}
00116
00117     long init(L4::Cap<L4Re::Video::Goos> goos)
00118     {
00119         _goos = goos;
00120         return init();
00121     }
00122
00123     long init(char const *name)
00124     {
00125         Env_ns ns;
00126         _goos = ns.query<L4Re::Video::Goos>(name);
00127         if (!_goos)
00128             return _goos.cap();
00129
00130         _flags |= F_dyn_goos;
00131
00132         return init();
00133     }
00134
00135     ~Goos_fb()
00136     {
00137         if (!_goos.is_valid())
00138             return;
00139
00140         if (_flags & F_dyn_view)
00141             _goos->delete_view(_view);
00142
00143         if (_flags & F_dyn_buffer)
00144             _goos->delete_buffer(_buffer_index);
00145
00146         if (_buffer.is_valid())
00147             cap_alloc.free(_buffer);
00148
00149         if (_flags & F_dyn_goos)
00150             cap_alloc.free(_goos);
00151     }
00152
00153     int view_info(L4Re::Video::View::Info *info)
00154     { return _view.info(info); }
00155
00156     L4Re::Video::View const *view() const { return &_view; }
00157     L4Re::Video::View *view() { return &_view; }
00158
00159     L4::Cap<L4Re::Dataspace> buffer() const { return _buffer; }
00160     void *attach_buffer()
00161     {
00162         void *fb_addr = 0;
00163         if (!_goos)
00164             return nullptr;
00165
00166         long ret = L4Re::Env::env()->rm()
00167             ->attach(&fb_addr, _buffer->size(),
00168                 L4Re::Rm::F::Search_addr | L4Re::Rm::F::RW, _buffer,
00169                 0, L4_SUPERPAGESHIFT);
00170         if (ret < 0)
00171             return nullptr;
00172
00173         return fb_addr;
00174     }
00175
00176     int refresh(int x, int y, int w, int h)
00177     { return _view.refresh(x, y, w, h); }
00178
00179     L4::Cap<L4Re::Video::Goos> goos() const { return _goos; }
00180 };
00181 }}}

```

16.422 goos_svr

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,

```

```

00004  *           Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *           economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009
00010 #pragma once
00011
00012 #include <l4/re/dataspace>
00013 #include <l4/re/video/goos>
00014 #include <l4/re/video/goos-sys.h>
00015
00016 #include <l4/sys/capability>
00017 #include <l4/sys/cxx/ipc_legacy>
00018
00019 namespace L4Re { namespace Util { namespace Video {
00020
00025 class Goos_svr
00026 {
00027     typedef L4Re::Video::Goos::Rights Rights;
00028 protected:
00030     L4::Cap<L4Re::Dataspace> _fb_ds;
00032     L4Re::Video::Goos::Info _screen_info;
00034     L4Re::Video::View::Info _view_info;
00035
00036 public:
00037     L4_RPC_LEGACY_DISPATCH(L4Re::Video::Goos);
00042     L4::Cap<L4Re::Dataspace> get_fb() const { return _fb_ds; }
00043
00048     L4Re::Video::Goos::Info const *screen_info() const { return &_screen_info; }
00049
00054     L4Re::Video::View::Info const *view_info() const { return &_view_info; }
00055
00066     virtual int refresh(int x, int y, int w, int h)
00067     { (void)x; (void)y; (void)w; (void)h; return -L4_ENOSYS; }
00068
00069
00078     void init_infos()
00079     {
00080         using L4Re::Video::View;
00082         _view_info.flags = View::F_none;
00083
00084         _view_info.view_index = 0;
00085         _view_info.xpos = 0;
00086         _view_info.ypos = 0;
00087         _view_info.width = _screen_info.width;
00088         _view_info.height = _screen_info.height;
00089         _view_info.pixel_info = _screen_info.pixel_info;
00090         _view_info.buffer_index = 0;
00091     }
00092
00096     virtual ~Goos_svr() {}
00097
00098     l4_ret_t op_view_info(Rights, unsigned idx, L4Re::Video::View::Info &info)
00099     {
00100         if (idx != 0)
00101             return -L4_ERANGE;
00102
00103         info = _view_info;
00104         return L4_EOK;
00105     }
00106
00107     l4_ret_t op_info(Rights, L4Re::Video::Goos::Info &info)
00108     {
00109         info = _screen_info;
00110         return L4_EOK;
00111     }
00112
00113     l4_ret_t op_get_static_buffer(Rights, unsigned idx,
00114                                   L4::Ipc::Cap<L4Re::Dataspace> &ds)
00115     {
00116         if (idx != 0)
00117             return -L4_ERANGE;
00118
00119         ds = L4::Ipc::Cap<L4Re::Dataspace>(_fb_ds, L4_CAP_FPAGE_RW);
00120         return L4_EOK;
00121     }
00122
00123     l4_ret_t op_refresh(Rights, int x, int y, int w, int h)
00124     { return refresh(x, y, w, h); }
00125
00126     l4_ret_t op_view_refresh(Rights, unsigned idx, int x, int y, int w, int h)
00127     {
00128         if (idx != 0)
00129             return -L4_ERANGE;
00130

```

```

00131     return refresh(x, y, w, h);
00132 }
00133
00134 l4_ret_t op_set_view_info(Rights, unsigned, L4Re::Video::View::Info)
00135 { return -L4_ENOSYS; }
00136
00137 l4_ret_t op_view_stack(Rights, unsigned, unsigned, bool)
00138 { return -L4_ENOSYS; }
00139
00140 l4_ret_t op_delete_view(Rights, unsigned)
00141 { return -L4_ENOSYS; }
00142
00143 l4_ret_t op_create_view(Rights)
00144 { return -L4_ENOSYS; }
00145
00146 l4_ret_t op_create_buffer(Rights, unsigned long,
00147                           L4::Ipc::Cap<L4Re::Dataspace> &)
00148 { return -L4_ENOSYS; }
00149
00150 l4_ret_t op_delete_buffer(Rights, unsigned)
00151 { return -L4_ENOSYS; }
00152 };
00153
00154
00155 }}}

```

16.423 colors

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009
00010 #pragma once
00011
00012 #include <l4/sys/compiler.h>
00013 #include <l4/cxx/minmax>
00014
00015 namespace L4Re { namespace Video {
00016
00021 class L4_EXPORT Color_component
00022 {
00023 private:
00024     unsigned char _bits;
00025     unsigned char _shift;
00026
00027 public:
00029     Color_component() : _bits(0), _shift(0) {}
00030
00036     Color_component(unsigned char bits, unsigned char shift)
00037     : _bits(bits), _shift(shift) {}
00038
00043     unsigned char size() const { return _bits; }
00044
00049     unsigned char shift() const { return _shift; }
00050
00055     bool operator == (Color_component const &o) const
00056     { return _shift == o._shift && _bits == o._bits; }
00057
00063     int get(unsigned long v) const
00064     {
00065         return ((v > _shift) & ~(~0UL < _bits)) < (16UL - _bits);
00066     }
00067
00073     long unsigned set(int v) const
00074     { return (static_cast<unsigned long>(v) > (16UL - _bits)) < _shift; }
00075
00080     template< typename OUT >
00081     void dump(OUT &s) const
00082     {
00083         s.printf("%d(%d)", static_cast<int>(size()), static_cast<int>(shift()));
00084     }
00085 } __attribute__((packed));
00086
00094 class L4_EXPORT Pixel_info
00095 {
00096 private:
00097     Color_component _r, _g, _b, _a;
00098     unsigned char _bpp;

```

```

00099
00100 public:
00105   Color_component const &r() const { return _r; }
00106
00111   Color_component const &g() const { return _g; }
00112
00117   Color_component const &b() const { return _b; }
00118
00123   Color_component const &a() const { return _a; }
00124
00131   Color_component const padding() const
00132   {
00133       unsigned char top_bit = cxx::max<unsigned char>(_r.size() + _r.shift(),
00134                                                       _g.size() + _g.shift());
00135       top_bit = cxx::max<unsigned char>(top_bit, _b.size() + _b.shift());
00136       top_bit = cxx::max<unsigned char>(top_bit, _a.size() + _a.shift());
00137
00138       unsigned char bits = _bpp * 8;
00139
00140       if (top_bit < bits)
00141           return Color_component(bits - top_bit, top_bit);
00142
00143       return Color_component(0, 0);
00144   }
00145
00150   unsigned char bytes_per_pixel() const { return _bpp; }
00151
00156   unsigned char bits_per_pixel() const
00157   { return _r.size() + _g.size() + _b.size() + _a.size(); }
00158
00163   bool has_alpha() const { return _a.size() > 0; }
00164
00169   void r(Color_component const &c) { _r = c; }
00170
00175   void g(Color_component const &c) { _g = c; }
00176
00181   void b(Color_component const &c) { _b = c; }
00182
00187   void a(Color_component const &c) { _a = c; }
00188
00193   void bytes_per_pixel(unsigned char bpp) { _bpp = bpp; }
00194
00198   Pixel_info() : _bpp(0) {};
00199
00212   Pixel_info(unsigned char bpp, char r, char rs, char g, char gs,
00213              char b, char bs, char a = 0, char as = 0)
00214   : _r(r, rs), _g(g, gs), _b(b, bs), _a(a, as), _bpp(bpp)
00215   {}
00216
00223   template<typename VBI>
00224   explicit Pixel_info(VBI const *vbi)
00225   : _r(vbi->red_mask_size, vbi->red_field_position),
00226     _g(vbi->green_mask_size, vbi->green_field_position),
00227     _b(vbi->blue_mask_size, vbi->blue_field_position),
00228     _bpp((vbi->bits_per_pixel + 7) / 8)
00229   {}
00230
00236   bool operator == (Pixel_info const &o) const
00237   {
00238       return _r == o._r && _g == o._g && _b == o._b && _a == o._a && _bpp == o._bpp;
00239   }
00240
00245   template< typename OUT >
00246   void dump(OUT &s) const
00247   {
00248       s.printf("RGBA(%d):%d(%d):%d(%d):%d(%d)",
00249               static_cast<int>(bytes_per_pixel()),
00250               static_cast<int>(r().size()), static_cast<int>(r().shift()),
00251               static_cast<int>(g().size()), static_cast<int>(g().shift()),
00252               static_cast<int>(b().size()), static_cast<int>(b().shift()),
00253               static_cast<int>(a().size()), static_cast<int>(a().shift()));
00254   }
00255 };
00256
00257
00258 }}
00259
00260

```

16.424 goos

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*

```

```

00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009 #pragma once
00010
00011 #include <l4/sys/capability>
00012 #include <l4/re/dataspace>
00013 #include <l4/re/video/colors>
00014 #include <l4/sys/cxx/ipc_iface>
00015
00016 namespace L4Re { namespace Video {
00017
00026 class L4_EXPORT Goos;
00027
00039 class L4_EXPORT View
00040 {
00041 private:
00042     friend class Goos;
00043
00044     L4::Cap<Goos> _goos;
00045     unsigned _view_idx;
00046
00047     View(l4_cap_idx_t goos, unsigned idx)
00048     : _goos(goos), _view_idx(_goos.is_valid() ? idx : ~0U) {}
00049
00050     unsigned view_index() const noexcept
00051     { return _goos.is_valid() ? _view_idx : ~0U; }
00052
00053 public:
00054     View() : _goos(L4::Cap<Goos>::Invalid), _view_idx(~0U) {}
00055
00059     enum Flags
00060     {
00061         F_none                = 0x00,
00062         F_set_buffer           = 0x01,
00063         F_set_buffer_offset    = 0x02,
00064         F_set_bytes_per_line   = 0x04,
00065         F_set_pixel            = 0x08,
00066         F_set_position         = 0x10,
00067         F_dyn_allocated        = 0x20,
00068         F_set_background       = 0x40,
00069         F_set_flags            = 0x80,
00070
00072         F_fully_dynamic        = F_set_buffer | F_set_buffer_offset | F_set_bytes_per_line
00073                                | F_set_pixel | F_set_position | F_dyn_allocated,
00074     };
00075
00082     enum V_flags
00083     {
00084         F_above                = 0x1000,
00085         F_flags_mask           = 0xff000,
00086     };
00087
00091     struct Info
00092     {
00093         unsigned flags          = 0;
00094         unsigned view_index     = 0;
00095
00096         unsigned long xpos      = 0;
00097         unsigned long ypos      = 0;
00098         unsigned long width     = 0;
00099         unsigned long height    = 0;
00100         unsigned long buffer_offset = 0;
00101         unsigned long bytes_per_line = 0;
00102         Pixel_info pixel_info;
00103         unsigned buffer_index   = 0;
00104
00106         bool has_static_buffer() const { return !(flags & F_set_buffer); }
00108         bool has_static_buffer_offset() const { return !(flags & F_set_buffer_offset); }
00109
00111         bool has_set_buffer() const { return flags & F_set_buffer; }
00113         bool has_set_buffer_offset() const { return flags & F_set_buffer_offset; }
00115         bool has_set_bytes_per_line() const { return flags & F_set_bytes_per_line; }
00117         bool has_set_pixel() const { return flags & F_set_pixel; }
00119         bool has_set_position() const { return flags & F_set_position; }
00120
00122     template< typename OUT >
00123     void dump(OUT &s) const
00124     {
00125         s.printf("View::Info:\n"
00126                 "  flags: %x\n"
00127                 "  size: %ldx%ld\n"
00128                 "  pos: %ldx%ld\n"
00129                 "  bytes_per_line: %ld\n"

```

```

00130         "    buffer_offset:  %lx\n"
00131         "    ",
00132         flags, width, height, xpos, ypos,
00133         bytes_per_line, buffer_offset);
00134     pixel_info.dump(s);
00135     s.printf("\n");
00136 }
00137 };
00138
00139 int info(Info *info) const noexcept;
00140
00141 int set_info(Info const &info) const noexcept;
00142
00143 int set_viewport(int scr_x, int scr_y, int w, int h, unsigned long buf_offset) const noexcept;
00144
00145 int stack(View const &pivot, bool behind = true) const noexcept;
00146
00147 int push_top() const noexcept
00148 { return stack(View(), true); }
00149
00150 int push_bottom() const noexcept
00151 { return stack(View(), false); }
00152
00153 int refresh(int x, int y, int w, int h) const noexcept;
00154
00155 bool valid() const { return _goos.is_valid(); }
00156 };
00157
00158 class L4_EXPORT Goos :
00159 public L4::Kobject_t<Goos, L4::Kobject, L4RE_PROTO_GOOS>
00160 {
00161 public:
00162     enum Flags
00163     {
00164         F_auto_refresh      = 0x01,
00165         F_pointer           = 0x02,
00166         F_dynamic_views     = 0x04,
00167         F_dynamic_buffers   = 0x08,
00168     };
00169
00170     struct Info
00171     {
00172         unsigned long width;
00173         unsigned long height;
00174         unsigned flags;
00175         unsigned num_static_views;
00176         unsigned num_static_buffers;
00177         Pixel_info pixel_info;
00178
00179         bool auto_refresh() const { return flags & F_auto_refresh; }
00180         bool has_pointer() const { return flags & F_pointer; }
00181         bool has_dynamic_views() const { return flags & F_dynamic_views; }
00182         bool has_dynamic_buffers() const { return flags & F_dynamic_buffers; }
00183
00184         Info()
00185         : width(0), height(0), flags(0), num_static_views(0),
00186           num_static_buffers(0) {}
00187     };
00188
00189     L4_INLINE_RPC(l4_ret_t, info, (Info *info));
00190
00191     L4_RPC(l4_ret_t, get_static_buffer, (unsigned idx,
00192                                         L4::Ipc::Out<L4::Cap<L4Re::Dataspace> > rbuf));
00193
00194     L4_RPC(l4_ret_t, create_buffer, (unsigned long size,
00195                                     L4::Ipc::Out<L4::Cap<L4Re::Dataspace> > rbuf));
00196
00197     L4_INLINE_RPC(l4_ret_t, delete_buffer, (unsigned idx));
00198
00199     // Use a wrapper for this RPC as we encapsulate the View
00200     L4_INLINE_RPC_NF(l4_ret_t, create_view, ());
00201
00202     l4_ret_t create_view(View *view, l4_utcb_t *utcb = l4_utcb()) const noexcept
00203     {
00204         l4_ret_t r = create_view_t::call(c(), utcb);
00205         if (r < 0)
00206             return r;
00207         *view = View(cap(), r);
00208         return r;
00209     }
00210
00211     // Use a wrapper as Views are encapsulated
00212     L4_INLINE_RPC_NF(l4_ret_t, delete_view, (unsigned index));
00213
00214     l4_ret_t delete_view(View const &v, l4_utcb_t *utcb = l4_utcb()) const noexcept
00215     {

```

```

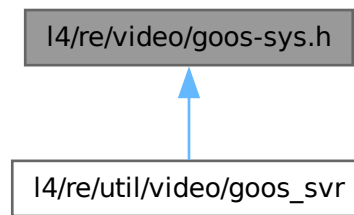
00334     return delete_view_t::call(c(), v._view_idx, utcb);
00335 }
00336
00342 View view(unsigned index) const noexcept;
00343
00347 L4_INLINE_RPC(l4_ret_t, refresh, (int x, int y, int w, int h));
00348
00349 // those are used by the View
00350 L4_INLINE_RPC(l4_ret_t, view_info, (unsigned index, View::Info *info));
00351 L4_INLINE_RPC(l4_ret_t, set_view_info, (unsigned index, View::Info const &info));
00352 L4_INLINE_RPC(l4_ret_t, view_stack, (unsigned index, unsigned pivot, bool behind));
00353 L4_INLINE_RPC(l4_ret_t, view_refresh, (unsigned index, int x, int y, int w, int h));
00354
00355 typedef L4::Typeid::Rpc<
00356     info_t, get_static_buffer_t, create_buffer_t, create_view_t, delete_buffer_t,
00357     delete_view_t, view_info_t, set_view_info_t, view_stack_t, view_refresh_t,
00358     refresh_t
00359 > Rpc;
00360 };
00361
00362 inline View
00363 Goos::view(unsigned index) const noexcept
00364 { return View(cap(), index); }
00365
00366 inline int
00367 View::info(Info *info) const noexcept
00368 { return _goos->view_info(_view_idx, info); }
00369
00370 inline int
00371 View::set_info(Info const &info) const noexcept
00372 { return _goos->set_view_info(_view_idx, info); }
00373
00374 inline int
00375 View::stack(View const &pivot, bool behind) const noexcept
00376 { return _goos->view_stack(_view_idx, pivot._view_idx, behind); }
00377
00378 inline int
00379 View::refresh(int x, int y, int w, int h) const noexcept
00380 { return _goos->view_refresh(_view_idx, x, y, w, h); }
00381
00382 inline int
00383 View::set_viewport(int scr_x, int scr_y, int w, int h,
00384                   unsigned long buf_offset) const noexcept
00385 {
00386     Info i;
00387     i.flags = F_set_buffer_offset | F_set_position;
00388     i.buffer_offset = buf_offset;
00389     i.buffer_index = 0;
00390     i.view_index = 0;
00391     i.bytes_per_line = 0;
00392     i.pixel_info = Pixel_info();
00393     i.xpos = scr_x;
00394     i.ypos = scr_y;
00395     i.width = w;
00396     i.height = h;
00397     return set_info(i);
00398 }
00399
00400 }

```

16.425 l4/re/video/goos-sys.h File Reference

Goos protocol definition.

This graph shows which files directly or indirectly include this file:



Namespaces

- namespace [L4Re](#)
[L4Re](#) C++ Interfaces.

Enumerations

- enum [L4Re::Video::Goos_::Opcodes](#)
 Frame buffer communication-protocol opcodes.

16.425.1 Detailed Description

Goos protocol definition.

Definition in file [goos-sys.h](#).

16.426 goos-sys.h

[Go to the documentation of this file.](#)

```

00001 #pragma once
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *               Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *               Björn Döbel <doebel@os.inf.tu-dresden.de>
00010  *               economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014
00015 namespace L4Re { namespace Video {
00016     namespace Goos_
00017     {
00023         enum Opcodes
00024         {
00025             Info, Get_buffer, Create_buffer, Create_view,
00026             Delete_buffer, Delete_view,
00027             View_info, View_set_info, View_stack, View_refresh,
00028             Screen_refresh
00029         };
00030     };
00031 }}
  
```

16.427 view

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009 #pragma once
00010
00011 #include <l4/re/video/goos>
00012

```

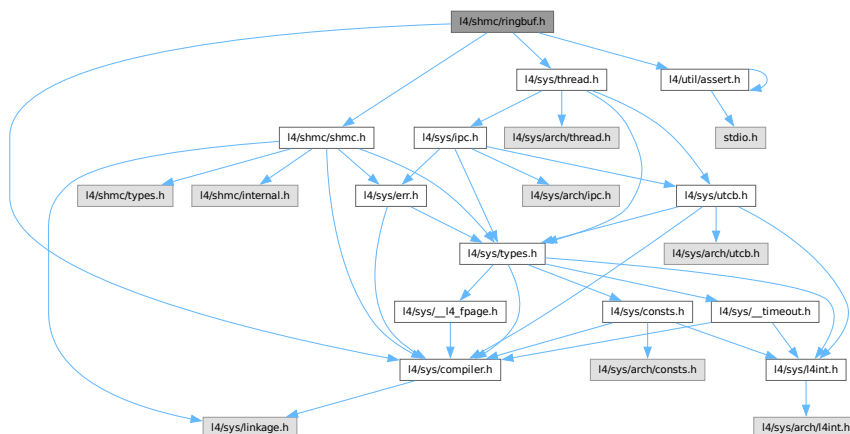
16.428 I4/shmc/ringbuf.h File Reference

```

#include <l4/shmc/shmc.h>
#include <l4/util/assert.h>
#include <l4/sys/compiler.h>
#include <l4/sys/thread.h>

```

Include dependency graph for ringbuf.h:



Data Structures

- struct [I4shmc_ringbuf_head_t](#)
Head field of a ring buffer.
- struct [I4shmc_ringbuf_t](#)
Ring buffer.

Macros

- #define [L4SHMC_RINGBUF_HEAD](#)(ringbuf)
Get ring buffer head pointer.
- #define [L4SHMC_RINGBUF_DATA](#)(ringbuf)
Get ring buffer data pointer.
- #define [L4SHMC_RINGBUF_DATA_SIZE](#)(ringbuf)
Get size of data area.

Functions

- `int l4shmc_rb_init_buffer (l4shmc_ringbuf_t *buf, l4shmc_area_t *area, char const *chunk_name, char const *signal_name, unsigned size)`
Initialize a ring buffer by creating an SHMC chunk and the corresponding signals.
- `void l4shmc_rb_deinit_buffer (l4shmc_ringbuf_t *buf)`
De-init a ring buffer.
- `int l4shmc_rb_attach_sender (l4shmc_ringbuf_t *buf, char const *signal_name, l4_cap_idx_t owner)`
Attach to sender signal of a ring buffer.
- `char * l4shmc_rb_sender_alloc_packet (l4shmc_ringbuf_head_t *head, unsigned psize)`
Allocate a packet of a given size within the ring buffer.
- `void l4shmc_rb_sender_put_data (l4shmc_ringbuf_t *buf, char *addr, char *data, unsigned dsize)`
Copy data into a previously allocated packet.
- `int l4shmc_rb_sender_next_copy_in (l4shmc_ringbuf_t *buf, char *data, unsigned size, int block_if_necessary)`
Copy in packet from an external data source.
- `void l4shmc_rb_sender_commit_packet (l4shmc_ringbuf_t *buf)`
Tell the consumer that new data is available.
- `int l4shmc_rb_init_receiver (l4shmc_ringbuf_t *buf, l4shmc_area_t *area, char const *chunk_name, char const *signal_name)`
Initialize receive buffer.
- `void l4shmc_rb_attach_receiver (l4shmc_ringbuf_t *buf, l4_cap_idx_t owner)`
Attach to receiver signal of a ring buffer.
- `int l4shmc_rb_receiver_wait_for_data (l4shmc_ringbuf_t *buf, int blocking)`
Check if (and optionally block until) new data is ready.
- `int l4shmc_rb_receiver_copy_out (l4shmc_ringbuf_head_t *head, char *target, unsigned *tsize)`
Copy data out of the buffer.
- `void l4shmc_rb_receiver_notify_done (l4shmc_ringbuf_t *buf)`
Notify producer that space is available.
- `int l4shmc_rb_receiver_read_next_size (l4shmc_ringbuf_head_t *head)`
Have a look at the ring buffer and see which size the next packet to be read has.

16.428.1 Function Documentation

16.428.1.1 l4shmc_rb_attach_receiver()

```
void l4shmc_rb_attach_receiver (
    l4shmc_ringbuf_t * buf,
    l4_cap_idx_t owner)
```

Attach to receiver signal of a ring buffer.

Attach owner to the receiver-side signal of a ring buffer, which is triggered whenever new data has been produced.

This is split from initialization, because you may not know the owner cap when initializing the buffer.

Parameters

| | |
|------------|-------------------------------|
| <i>buf</i> | pointer to ring buffer struct |
|------------|-------------------------------|

| | |
|--------------|--------------|
| <i>owner</i> | owner thread |
|--------------|--------------|

References [L4_CV](#).

16.428.1.2 l4shmc_rb_attach_sender()

```
int l4shmc_rb_attach_sender (  
    l4shmc_ringbuf_t * buf,  
    char const * signal_name,  
    l4_cap_idx_t owner)
```

Attach to sender signal of a ring buffer.

Attach owner to the sender-side signal of a ring buffer, which is triggered whenever new space has been freed in the buffer for the sender to write to.

This is split from initialization, because you may not know the owner cap when initializing the buffer.

Parameters

| | |
|--------------------|-------------------------------|
| <i>buf</i> | pointer to ring buffer struct |
| <i>signal_name</i> | signal base name |
| <i>owner</i> | owner thread |

Returns

0 on success, error otherwise

References [L4_CV](#).

16.428.1.3 l4shmc_rb_deinit_buffer()

```
void l4shmc_rb_deinit_buffer (  
    l4shmc_ringbuf_t * buf)
```

De-init a ring buffer.

Parameters

| | |
|------------|-------------------------------|
| <i>buf</i> | pointer to ring buffer struct |
|------------|-------------------------------|

References [L4_CV](#).

16.428.1.4 l4shmc_rb_init_buffer()

```
int l4shmc_rb_init_buffer (
    l4shmc_ringbuf_t * buf,
    l4shmc_area_t * area,
    char const * chunk_name,
    char const * signal_name,
    unsigned size)
```

Initialize a ring buffer by creating an SHMC chunk and the corresponding signals.

This needs to be done by one of the participating parties when setting up communication channel.

Precondition

area has been attached using [l4shmc_attach\(\)](#).

Parameters

| | |
|--------------------|--------------------------------------|
| <i>buf</i> | pointer to ring buffer struct |
| <i>area</i> | pointer to SHMC area |
| <i>chunk_name</i> | name of SHMC chunk to create in area |
| <i>signal_name</i> | base name for SHMC signals to create |
| <i>size</i> | chunk size |

Returns

0 on success, error otherwise

References [L4_CV](#).

16.428.1.5 l4shmc_rb_init_receiver()

```
int l4shmc_rb_init_receiver (
    l4shmc_ringbuf_t * buf,
    l4shmc_area_t * area,
    char const * chunk_name,
    char const * signal_name)
```

Initialize receive buffer.

Initialize the receiver-side of a ring buffer. This requires the underlying SHMC chunk and the corresponding signals to be valid already (read: to be initialized by the sender).

Precondition

chunk & signals have been created and initialized by the sender side

Parameters

| | |
|--------------------|--------------------------------------|
| <i>buf</i> | pointer to ring buffer struct |
| <i>area</i> | pointer to SHMC area |
| <i>chunk_name</i> | name of SHMC chunk to create in area |
| <i>signal_name</i> | base name for SHMC signals to create |

Returns

0 on success, error otherwise

References [L4_CV](#).

16.428.1.6 l4shmc_rb_receiver_copy_out()

```
int l4shmc_rb_receiver_copy_out (
    l4shmc_ringbuf_head_t * head,
    char * target,
    unsigned * tsize)
```

Copy data out of the buffer.

Parameters

| | | |
|----------------|---------------|--|
| | <i>head</i> | ring buffer head pointer |
| | <i>target</i> | valid target buffer |
| <i>in, out</i> | <i>tsize</i> | size of target buffer (must be \geq packet size!); contains the real data size |

Returns

0 on success, negative error otherwise

References [L4_CV](#).

16.428.1.7 l4shmc_rb_receiver_notify_done()

```
void l4shmc_rb_receiver_notify_done (
    l4shmc_ringbuf_t * buf)
```

Notify producer that space is available.

Parameters

| | |
|------------|-------------------------------|
| <i>buf</i> | pointer to ring buffer struct |
|------------|-------------------------------|

References [L4_CV](#).

16.428.1.8 l4shmc_rb_receiver_read_next_size()

```
int l4shmc_rb_receiver_read_next_size (  
    l4shmc_ringbuf_head_t * head)
```

Have a look at the ring buffer and see which size the next packet to be read has.

Does not modify anything.

Returns

size of next buffer or -1 if no data available

References [L4_CV](#), and [L4_END_DECLS](#).

16.428.1.9 l4shmc_rb_receiver_wait_for_data()

```
int l4shmc_rb_receiver_wait_for_data (  
    l4shmc_ringbuf_t * buf,  
    int blocking)
```

Check if (and optionally block until) new data is ready.

Parameters

| | |
|-----------------|--|
| <i>buf</i> | pointer to ring buffer struct |
| <i>blocking</i> | block if data is not available immediately |

Returns immediately, if data is available.

Returns

0 success, data available, != 0 otherwise

References [L4_CV](#).

16.428.1.10 l4shmc_rb_sender_alloc_packet()

```
char * l4shmc_rb_sender_alloc_packet (
    l4shmc_ringbuf_head_t * head,
    unsigned psize)
```

Allocate a packet of a given size within the ring buffer.

This packet may wrap around at the end of the buffer. Users need to be aware of that.

Parameters

| | |
|--------------|--------------------------|
| <i>head</i> | ring buffer head pointer |
| <i>psize</i> | packet size |

Returns

valid address on success

Return values

| | |
|-------------|-------------------------------|
| <i>NULL</i> | if not enough space available |
|-------------|-------------------------------|

References [L4_CV](#).

16.428.1.11 l4shmc_rb_sender_commit_packet()

```
void l4shmc_rb_sender_commit_packet (
    l4shmc_ringbuf_t * buf)
```

Tell the consumer that new data is available.

Parameters

| | |
|------------|-------------------------------|
| <i>buf</i> | pointer to ring buffer struct |
|------------|-------------------------------|

References [L4_CV](#).

16.428.1.12 l4shmc_rb_sender_next_copy_in()

```
int l4shmc_rb_sender_next_copy_in (
    l4shmc_ringbuf_t * buf,
    char * data,
    unsigned size,
    int block_if_necessary)
```

Copy in packet from an external data source.

This is the function you'll want to use. Just pass it a buffer pointer and let the lib do the work.

Parameters

| | |
|---------------------------|--------------------------------------|
| <i>buf</i> | pointer to ring buffer struct |
| <i>data</i> | valid buffer |
| <i>size</i> | data size |
| <i>block_if_necessary</i> | bool: block if buffer currently full |

Return values

| | |
|------------|--|
| 0 | on success |
| -L4_ENOMEM | if block == false and no space available |

References [L4_CV](#).

16.428.1.13 l4shmc_rb_sender_put_data()

```
void l4shmc_rb_sender_put_data (
    l4shmc_ringbuf_t * buf,
    char * addr,
    char * data,
    unsigned dsize)
```

Copy data into a previously allocated packet.

This function is wrap-around aware.

Parameters

| | |
|--------------|--|
| <i>buf</i> | pointer to ring buffer struct |
| <i>addr</i> | valid destination (allocate with alloc_packet()) |
| <i>data</i> | data source |
| <i>dsize</i> | data size |

References [L4_CV](#).

16.429 ringbuf.h

[Go to the documentation of this file.](#)

```
00001 /*
00002  * (c) 2010 Björn Döbel <doebel@os.inf.tu-dresden.de>
00003  *     economic rights: Technische Universität Dresden (Germany)
00004  * This file is part of TUD:OS and distributed under the terms of the
00005  * GNU Lesser General Public License 2.1.
00006  * Please see the COPYING-LGPL-2.1 file for details.
00007 */
00008
00012 #pragma once
00013
00014 #include <l4/shmc/shmc.h>
00015 #include <l4/util/assert.h>
```

```

00016 #include <l4/sys/compiler.h>
00017 #include <l4/sys/thread.h>
00018
00019 L4_BEGIN_DECLS
00020
00035
00046
00047 /*
00048  * Turn on ringbuf poisoning. This will add magic values to the ringbuf
00049  * header as well as each packet header and check that these values are
00050  * valid all the time.
00051  */
00052 #define L4SHMC_RINGBUF_POISONING 1
00053
00059 typedef struct
00060 {
00061     volatile l4_uint32_t lock;
00062     unsigned data_size;
00063 #if L4SHMC_RINGBUF_POISONING
00064     char magic1;
00065 #endif
00066     unsigned next_read;
00067     unsigned next_write;
00068 #if L4SHMC_RINGBUF_POISONING
00069     char magic2;
00070 #endif
00071     unsigned bytes_filled;
00072     unsigned sender_waits;
00073 #if L4SHMC_RINGBUF_POISONING
00074     char magic3;
00075 #endif
00076     char data[];
00077 } l4shmc_ringbuf_head_t;
00078
00079
00085 typedef struct
00086 {
00087     l4shmc_area_t *_area;
00088     l4_cap_idx_t _owner;
00089     l4shmc_chunk_t _chunk;
00090     unsigned _size;
00091     char *_chunkname;
00092     char *_signame;
00093     l4shmc_ringbuf_head_t *_addr;
00094     l4shmc_signal_t _signal_full;
00095     l4shmc_signal_t _signal_empty;
00096 } l4shmc_ringbuf_t;
00097
00098
00105 #define L4SHMC_RINGBUF_HEAD(ringbuf) ((l4shmc_ringbuf_head_t*) ((ringbuf)->_addr))
00106
00107
00114 #define L4SHMC_RINGBUF_DATA(ringbuf) (L4SHMC_RINGBUF_HEAD(ringbuf)->data)
00115
00116
00123 #define L4SHMC_RINGBUF_DATA_SIZE(ringbuf) ((ringbuf)->_size - sizeof(l4shmc_ringbuf_head_t))
00124
00125 enum lock_content
00126 {
00127     lock_cont_min = 4,
00128     locked = 5,
00129     unlocked = 6,
00130     lock_cont_max = 7,
00131 };
00132
00133 static L4_CV inline void l4shmc_rb_lock(l4shmc_ringbuf_head_t *head)
00134 {
00135     ASSERT_NOT_NULL(head);
00136     ASSERT_ASSERT(head->lock > lock_cont_min);
00137     ASSERT_ASSERT(head->lock < lock_cont_max);
00138
00139     while (!l4util_cmpxchg32(&head->lock, unlocked, locked))
00140         l4_thread_yield();
00141 }
00142
00143
00144 static L4_CV inline void l4shmc_rb_unlock(l4shmc_ringbuf_head_t *head)
00145 {
00146     ASSERT_NOT_NULL(head);
00147     ASSERT_ASSERT(head->lock > lock_cont_min);
00148     ASSERT_ASSERT(head->lock < lock_cont_max);
00149
00150     head->lock = unlocked;
00151 }
00152
00153 /*****
00154  * Initialization *

```

```

00155  *****/
00156
00173 L4_CV int  l4shmc_rb_init_buffer(l4shmc_ringbuf_t *buf, l4shmc_area_t *area,
00174                                     char const *chunk_name,
00175                                     char const *signal_name, unsigned size);
00176
00182 L4_CV void l4shmc_rb_deinit_buffer(l4shmc_ringbuf_t *buf);
00183
00184
00185
00186 /*****
00187  *      RINGBUF SENDER      *
00188  *****/
00189
00206 L4_CV int l4shmc_rb_attach_sender(l4shmc_ringbuf_t *buf, char const *signal_name,
00207                                     l4_cap_idx_t owner);
00208
00209
00222 L4_CV char *l4shmc_rb_sender_alloc_packet(l4shmc_ringbuf_head_t *head,
00223                                             unsigned psize);
00224
00225
00236 L4_CV void l4shmc_rb_sender_put_data(l4shmc_ringbuf_t *buf, char *addr,
00237                                       char *data, unsigned dsize);
00238
00239
00254 L4_CV int  l4shmc_rb_sender_next_copy_in(l4shmc_ringbuf_t *buf, char *data,
00255                                             unsigned size, int block_if_necessary);
00256
00257
00263 L4_CV void l4shmc_rb_sender_commit_packet(l4shmc_ringbuf_t *buf);
00264
00265
00266 /*****
00267  *      RINGBUF RECEIVER   *
00268  *****/
00269
00286 L4_CV int  l4shmc_rb_init_receiver(l4shmc_ringbuf_t *buf, l4shmc_area_t *area,
00287                                       char const *chunk_name,
00288                                       char const *signal_name);
00289
00290
00303 L4_CV void l4shmc_rb_attach_receiver(l4shmc_ringbuf_t *buf, l4_cap_idx_t owner);
00304
00305
00316 L4_CV int l4shmc_rb_receiver_wait_for_data(l4shmc_ringbuf_t *buf, int blocking);
00317
00318
00328 L4_CV int  l4shmc_rb_receiver_copy_out(l4shmc_ringbuf_head_t *head, char *target,
00329                                          unsigned *tsize);
00330
00331
00337 L4_CV void l4shmc_rb_receiver_notify_done(l4shmc_ringbuf_t *buf);
00338
00339
00346 L4_CV int l4shmc_rb_receiver_read_next_size(l4shmc_ringbuf_head_t *head);
00347
00348 L4_END_DECLS

```

16.430 l4/shmc/shmc.h File Reference

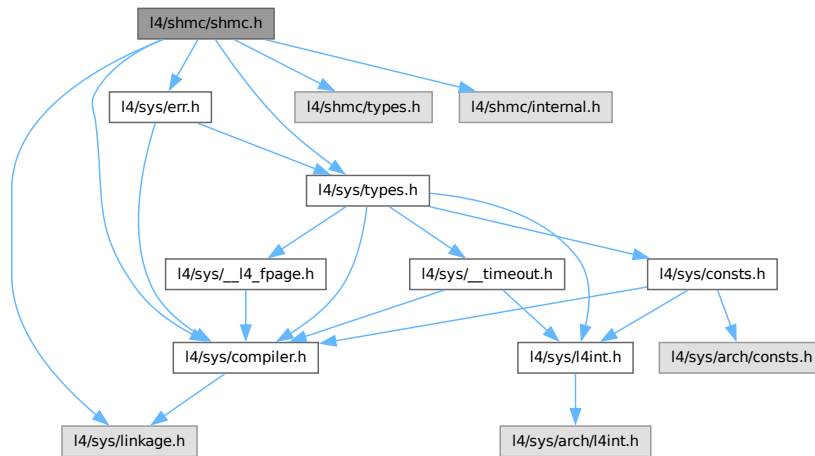
Shared memory library header file.

```

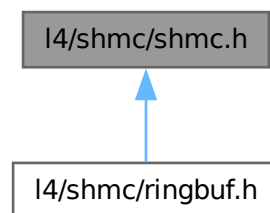
#include <l4/sys/compiler.h>
#include <l4/sys/linkage.h>
#include <l4/sys/types.h>
#include <l4/sys/err.h>
#include <l4/shmc/types.h>
#include <l4/shmc/internal.h>

```

Include dependency graph for shmc.h:



This graph shows which files directly or indirectly include this file:



Functions

- [L4_BEGIN_DECLS](#) long [l4shmc_create](#) (char const *shmc_name)
Create a shared memory area.
- long [l4shmc_attach](#) (char const *shmc_name, l4shmc_area_t *shmarea)
Attach to a shared memory area.
- long [l4shmc_get_client_nr](#) (l4shmc_area_t const *shmarea)
Determine the client number of the shared memory region.
- long [l4shmc_mark_client_initialized](#) (l4shmc_area_t *shmarea)
Mark this shared memory client as 'initialized'.
- long [l4shmc_get_initialized_clients](#) (l4shmc_area_t *shmarea, [l4_umword_t](#) *bitmask)
Fetch the `_clients_init_done` bitmask of the shared memory area.
- long [l4shmc_add_chunk](#) (l4shmc_area_t *shmarea, char const *chunk_name, [l4_umword_t](#) chunk_capacity, l4shmc_chunk_t *chunk)
Add a chunk in the shared memory area.

- long [l4shmc_add_signal](#) (l4shmc_area_t *shmarea, char const *signal_name, l4shmc_signal_t *signal)
Add a signal for the shared memory area.
- long [l4shmc_trigger](#) (l4shmc_signal_t *signal)
Trigger a signal.
- long [l4shmc_chunk_try_to_take](#) (l4shmc_chunk_t *chunk)
Try to mark chunk busy.
- long [l4shmc_chunk_try_to_take_for_writing](#) (l4shmc_chunk_t *chunk)
Try to mark chunk busy writing.
- long [l4shmc_chunk_try_to_take_for_overwriting](#) (l4shmc_chunk_t *chunk)
Try to mark the chunk busy writing after it was ready for reading.
- long [l4shmc_chunk_try_to_take_for_reading](#) (l4shmc_chunk_t *chunk)
Try to mark chunk busy reading.
- long [l4shmc_chunk_ready](#) (l4shmc_chunk_t *chunk, l4_umword_t size)
Mark chunk as filled (ready).
- long [l4shmc_chunk_ready_sig](#) (l4shmc_chunk_t *chunk, l4_umword_t size)
Mark chunk as filled (ready) and signal consumer.
- long [l4shmc_get_chunk](#) (l4shmc_area_t *shmarea, char const *chunk_name, l4shmc_chunk_t *chunk)
Get chunk out of shared memory area.
- long [l4shmc_get_chunk_to](#) (l4shmc_area_t *shmarea, char const *chunk_name, l4_umword_t timeout_ms, l4shmc_chunk_t *chunk)
Get chunk out of shared memory area, with timeout.
- long [l4shmc_iterate_chunk](#) (l4shmc_area_t const *shmarea, char const **chunk_name, long offs)
Iterate over names of all existing chunks.
- long [l4shmc_attach_signal](#) (l4shmc_area_t *shmarea, char const *signal_name, l4_cap_idx_t thread, l4shmc_signal_t *signal)
Attach to signal.
- long [l4shmc_get_signal](#) (l4shmc_area_t *shmarea, char const *signal_name, l4shmc_signal_t *signal)
Get signal object from the shared memory area.
- long [l4shmc_enable_signal](#) (l4shmc_signal_t *signal)
Enable a signal.
- long [l4shmc_enable_chunk](#) (l4shmc_chunk_t *chunk)
Enable a signal connected with a chunk.
- long [l4shmc_wait_any](#) (l4shmc_signal_t **retsignal)
Wait on any signal.
- long [l4shmc_wait_any_try](#) (l4shmc_signal_t **retsignal)
Check whether any waited signal has an event pending.
- long [l4shmc_wait_any_to](#) (l4_timeout_t timeout, l4shmc_signal_t **retsignal)
Wait for any signal with timeout.
- long [l4shmc_wait_signal](#) (l4shmc_signal_t *signal)
Wait on a specific signal.
- long [l4shmc_wait_signal_to](#) (l4shmc_signal_t *signal, l4_timeout_t timeout)
Wait on a specific signal, with timeout.
- long [l4shmc_wait_signal_try](#) (l4shmc_signal_t *signal)
Check whether a specific signal has an event pending.
- long [l4shmc_wait_chunk](#) (l4shmc_chunk_t *chunk)
Wait on a specific chunk.
- long [l4shmc_wait_chunk_to](#) (l4shmc_chunk_t *chunk, l4_timeout_t timeout)
Check whether a specific chunk has an event pending, with timeout.
- long [l4shmc_wait_chunk_try](#) (l4shmc_chunk_t *chunk)
Check whether a specific chunk has an event pending.
- long [l4shmc_chunk_consumed](#) (l4shmc_chunk_t *chunk)

- Mark a chunk as free.*

 - long [l4shmc_connect_chunk_signal](#) (l4shmc_chunk_t *chunk, l4shmc_signal_t *signal)
- Connect a signal with a chunk.*

 - long [l4shmc_is_chunk_ready](#) (l4shmc_chunk_t const *chunk)
- Check whether data is available.*

 - long [l4shmc_is_chunk_clear](#) (l4shmc_chunk_t const *chunk)
- Check whether chunk is free.*

 - void * [l4shmc_chunk_ptr](#) (l4shmc_chunk_t const *chunk)
- Get data pointer to chunk.*

 - long [l4shmc_chunk_size](#) (l4shmc_chunk_t const *chunk)
- Get current size of a chunk.*

 - long [l4shmc_chunk_capacity](#) (l4shmc_chunk_t const *chunk)
- Get capacity of a chunk.*

 - l4shmc_signal_t * [l4shmc_chunk_signal](#) (l4shmc_chunk_t const *chunk)
- Get the registered signal of a chunk.*

 - [l4_cap_idx_t](#) [l4shmc_signal_cap](#) (l4shmc_signal_t const *signal)
- Get the signal capability of a signal.*

 - long [l4shmc_check_magic](#) (l4shmc_chunk_t const *chunk)
- Check magic value of a chunk.*

 - long [l4shmc_area_size](#) (l4shmc_area_t const *shmarea)
- Get size of shared memory area.*

 - long [l4shmc_area_size_free](#) (l4shmc_area_t const *shmarea)
- Get free size of shared memory area.*

 - long [l4shmc_area_overhead](#) (void)
- Get memory overhead per area that is not available for chunks.*

 - long [l4shmc_chunk_overhead](#) (void)
- Get memory overhead required in addition to the chunk capacity for adding one chunk.*

16.430.1 Detailed Description

Shared memory library header file.

Definition in file [shmc.h](#).

16.431 shmc.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Björn Döbel <doebel@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  * This file is part of TUD:OS and distributed under the terms of the
00010  * GNU Lesser General Public License 2.1.
00011  * Please see the COPYING-LGPL-2.1 file for details.
00012  */
00013 #pragma once
00014
00015 #include <l4/sys/compiler.h>
00016 #include <l4/sys/linkage.h>
00017 #include <l4/sys/types.h>
00018 #include <l4/sys/err.h>
00019
00069
00070 #define __INCLUDED_FROM_L4SHMC_H__
00071 #include <l4/shmc/types.h>

```

```
00072
00073 L4_BEGIN_DECLS
00074
00087 L4_CV long
00088 l4shmc_create(char const *shmc_name);
00089
00111 L4_CV long
00112 l4shmc_attach(char const *shmc_name, l4shmc_area_t *shmarea);
00113
00122 L4_CV long
00123 l4shmc_get_client_nr(l4shmc_area_t const *shmarea);
00124
00137 L4_CV long
00138 l4shmc_mark_client_initialized(l4shmc_area_t *shmarea);
00139
00152 L4_CV long
00153 l4shmc_get_initialized_clients(l4shmc_area_t *shmarea, l4_umword_t *bitmask);
00154
00167 L4_CV long
00168 l4shmc_add_chunk(l4shmc_area_t *shmarea, char const *chunk_name,
00169                  l4_umword_t chunk_capacity, l4shmc_chunk_t *chunk);
00170
00182 L4_CV long
00183 l4shmc_add_signal(l4shmc_area_t *shmarea, char const *signal_name,
00184                  l4shmc_signal_t *signal);
00185
00195 L4_CV L4_INLINE long
00196 l4shmc_trigger(l4shmc_signal_t *signal);
00197
00207 L4_CV L4_INLINE long
00208 l4shmc_chunk_try_to_take(l4shmc_chunk_t *chunk);
00209
00221 L4_CV L4_INLINE long
00222 l4shmc_chunk_try_to_take_for_writing(l4shmc_chunk_t *chunk);
00223
00238 L4_CV L4_INLINE long
00239 l4shmc_chunk_try_to_take_for_overwriting(l4shmc_chunk_t *chunk);
00240
00250 L4_CV L4_INLINE long
00251 l4shmc_chunk_try_to_take_for_reading(l4shmc_chunk_t *chunk);
00252
00263 L4_CV L4_INLINE long
00264 l4shmc_chunk_ready(l4shmc_chunk_t *chunk, l4_umword_t size);
00265
00276 L4_CV L4_INLINE long
00277 l4shmc_chunk_ready_sig(l4shmc_chunk_t *chunk, l4_umword_t size);
00278
00290 L4_CV L4_INLINE long
00291 l4shmc_get_chunk(l4shmc_area_t *shmarea, char const *chunk_name,
00292                  l4shmc_chunk_t *chunk);
00293
00307 L4_CV long
00308 l4shmc_get_chunk_to(l4shmc_area_t *shmarea, char const *chunk_name,
00309                     l4_umword_t timeout_ms, l4shmc_chunk_t *chunk);
00310
00324 L4_CV long
00325 l4shmc_iterate_chunk(l4shmc_area_t const *shmarea, char const **chunk_name,
00326                      long offs);
00327
00340 L4_CV long
00341 l4shmc_attach_signal(l4shmc_area_t *shmarea, char const *signal_name,
00342                      l4_cap_idx_t thread, l4shmc_signal_t *signal);
00343
00344
00356 L4_CV long
00357 l4shmc_get_signal(l4shmc_area_t *shmarea, char const *signal_name,
00358                  l4shmc_signal_t *signal);
00359
00373 L4_CV long
00374 l4shmc_enable_signal(l4shmc_signal_t *signal);
00375
00389 L4_CV long
00390 l4shmc_enable_chunk(l4shmc_chunk_t *chunk);
00391
00401 L4_CV L4_INLINE long
00402 l4shmc_wait_any(l4shmc_signal_t **retsignal);
00403
00417 L4_CV L4_INLINE long
00418 l4shmc_wait_any_try(l4shmc_signal_t **retsignal);
00419
00434 L4_CV long
00435 l4shmc_wait_any_to(l4_timeout_t timeout, l4shmc_signal_t **retsignal);
00436
00446 L4_CV L4_INLINE long
00447 l4shmc_wait_signal(l4shmc_signal_t *signal);
00448
00459 L4_CV long
```

```

00460 l4shmc_wait_signal_to(l4shmc_signal_t *signal, l4_timeout_t timeout);
00461
00475 L4_CV L4_INLINE long
00476 l4shmc_wait_signal_try(l4shmc_signal_t *signal);
00477
00487 L4_CV L4_INLINE long
00488 l4shmc_wait_chunk(l4shmc_chunk_t *chunk);
00489
00504 L4_CV long
00505 l4shmc_wait_chunk_to(l4shmc_chunk_t *chunk, l4_timeout_t timeout);
00506
00520 L4_CV L4_INLINE long
00521 l4shmc_wait_chunk_try(l4shmc_chunk_t *chunk);
00522
00532 L4_CV L4_INLINE long
00533 l4shmc_chunk_consumed(l4shmc_chunk_t *chunk);
00534
00545 L4_CV long
00546 l4shmc_connect_chunk_signal(l4shmc_chunk_t *chunk, l4shmc_signal_t *signal);
00547
00557 L4_CV L4_INLINE long
00558 l4shmc_is_chunk_ready(l4shmc_chunk_t const *chunk);
00559
00569 L4_CV L4_INLINE long
00570 l4shmc_is_chunk_clear(l4shmc_chunk_t const *chunk);
00571
00580 L4_CV L4_INLINE void *
00581 l4shmc_chunk_ptr(l4shmc_chunk_t const *chunk);
00582
00591 L4_CV L4_INLINE long
00592 l4shmc_chunk_size(l4shmc_chunk_t const *chunk);
00593
00602 L4_CV L4_INLINE long
00603 l4shmc_chunk_capacity(l4shmc_chunk_t const *chunk);
00604
00614 L4_CV L4_INLINE l4shmc_signal_t *
00615 l4shmc_chunk_signal(l4shmc_chunk_t const *chunk);
00616
00625 L4_CV L4_INLINE l4_cap_idx_t
00626 l4shmc_signal_cap(l4shmc_signal_t const *signal);
00627
00637 L4_CV L4_INLINE long
00638 l4shmc_check_magic(l4shmc_chunk_t const *chunk);
00639
00649 L4_CV long
00650 l4shmc_area_size(l4shmc_area_t const *shmarea);
00651
00661 L4_CV long
00662 l4shmc_area_size_free(l4shmc_area_t const *shmarea);
00663
00670 L4_CV long
00671 l4shmc_area_overhead(void);
00672
00680 L4_CV long
00681 l4shmc_chunk_overhead(void);
00682
00683 #include <l4/shmc/internal.h>
00684
00685 L4_END_DECLS

```

16.432 l4/sigma0/sigma0.h File Reference

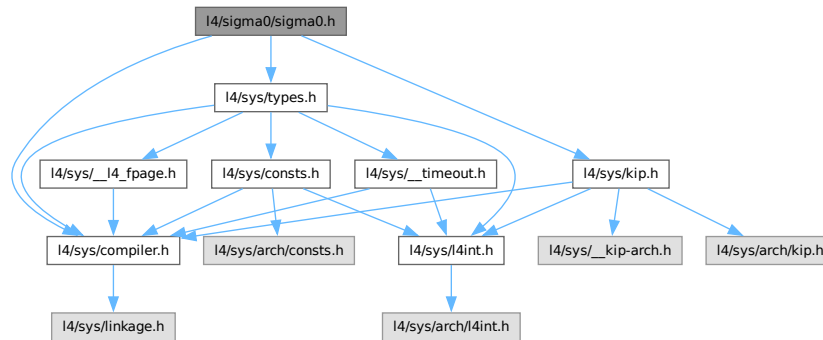
Sigma0 interface.

```

#include <l4/sys/compiler.h>
#include <l4/sys/types.h>
#include <l4/sys/kip.h>

```


Include dependency graph for sigma0.h:



Macros

- **#define SIGMA0_REQ_MAGIC** ~0xFFUL
Request magic.
- **#define SIGMA0_REQ_MASK** ~0xFFUL
Request mask.
- **#define SIGMA0_REQ_ID_MASK** 0xF0
ID mask.
- **#define SIGMA0_REQ_ID_FPAGE_RAM** 0x60
RAM.
- **#define SIGMA0_REQ_ID_FPAGE_IOMEM** 0x70
I/O memory.
- **#define SIGMA0_REQ_ID_FPAGE_IOMEM_CACHED** 0x80
Cached I/O memory.
- **#define SIGMA0_REQ_ID_FPAGE_ANY** 0x90
Any.
- **#define SIGMA0_REQ_ID_KIP** 0xA0
KIP.
- **#define SIGMA0_REQ_ID_DEBUG_DUMP** 0xC0
Debug dump.
- **#define SIGMA0_REQ_ID_COV** 0xE0
Trigger cov dump.
- **#define SIGMA0_IS_MAGIC_REQ**(d1)
Check if magic.
- **#define SIGMA0_REQ**(x)
Construct.
- **#define SIGMA0_REQ_FPAGE_RAM** ([SIGMA0_REQ](#)(FPAGE_RAM))
RAM.
- **#define SIGMA0_REQ_FPAGE_IOMEM** ([SIGMA0_REQ](#)(FPAGE_IOMEM))
I/O memory.
- **#define SIGMA0_REQ_FPAGE_IOMEM_CACHED** ([SIGMA0_REQ](#)(FPAGE_IOMEM_CACHED))
Cache I/O memory.
- **#define SIGMA0_REQ_FPAGE_ANY** ([SIGMA0_REQ](#)(FPAGE_ANY))
Any.

- #define **SIGMA0_REQ_KIP** ([SIGMA0_REQ](#)(KIP))
KIP.
- #define **SIGMA0_REQ_DEBUG_DUMP** ([SIGMA0_REQ](#)(DEBUG_DUMP))
Debug dump.
- #define **SIGMA0_REQ_COV** ([SIGMA0_REQ](#)(COV))
Cov.

Enumerations

- enum [l4sigma0_return_flags_t](#) {
 [L4SIGMA0_OK](#) , [L4SIGMA0_NOTALIGNED](#) , [L4SIGMA0_IPCERROR](#) , [L4SIGMA0_NOFPAGE](#) ,
 [L4SIGMA0_4](#) , [L4SIGMA0_5](#) , [L4SIGMA0_SMALLERFPAGE](#) }
Return flags of libsigma0 functions.

Functions

- [L4_BEGIN_DECLS](#) [l4_kernel_info_t](#) * [l4sigma0_map_kip](#) ([l4_cap_idx_t](#) sigma0, void *addr, unsigned log2↵_size)
Map the kernel info page from sigma0 to addr.
- int [l4sigma0_map_mem](#) ([l4_cap_idx_t](#) sigma0, [l4_addr_t](#) phys, [l4_addr_t](#) virt, [l4_addr_t](#) size)
Request a memory mapping from sigma0.
- int [l4sigma0_map_iomem](#) ([l4_cap_idx_t](#) sigma0, [l4_addr_t](#) phys, [l4_addr_t](#) virt, [l4_addr_t](#) size, int cached)
Request IO memory from sigma0.
- int [l4sigma0_map_anypage](#) ([l4_cap_idx_t](#) sigma0, [l4_addr_t](#) map_area, unsigned log2_map_size, [l4_addr_t](#) *base, unsigned sz)
Request an arbitrary free page of RAM.
- void [l4sigma0_debug_dump](#) ([l4_cap_idx_t](#) sigma0)
Request sigma0 to dump internal debug information.
- char const * [l4sigma0_map_errstr](#) (int err)
Get user readable error messages for the return codes.

16.432.1 Detailed Description

Sigma0 interface.

Definition in file [sigma0.h](#).

16.433 sigma0.h

[Go to the documentation of this file.](#)

```

00001
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *           Alexander Warg <warg@os.inf.tu-dresden.de>,
00010  *           Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00011  *           economic rights: Technische Universität Dresden (Germany)
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014 #ifndef __L4_SIGMA0_SIGMA0_H
00015 #define __L4_SIGMA0_SIGMA0_H
00016
00024
00025 #include <l4/sys/compiler.h>

```

```

00026 #include <l4/sys/types.h>
00027 #include <l4/sys/kip.h>
00028
00035 #undef SIGMA0_REQ_MAGIC
00036 #undef SIGMA0_REQ_MASK
00037
00038 # define SIGMA0_REQ_MAGIC    ~0xFFUL
00039 # define SIGMA0_REQ_MASK    ~0xFFUL
00040
00041 /* Starting with 0x60 allows to detect components which still use the old
00042  * constants (0x00 ... 0x50) */
00043 #define SIGMA0_REQ_ID_MASK    0xF0
00044 #define SIGMA0_REQ_ID_FPAGE_RAM    0x60
00045 #define SIGMA0_REQ_ID_FPAGE_IOMEM    0x70
00046 #define SIGMA0_REQ_ID_FPAGE_IOMEM_CACHED    0x80
00047 #define SIGMA0_REQ_ID_FPAGE_ANY    0x90
00048 #define SIGMA0_REQ_ID_KIP    0xA0
00049 #define SIGMA0_REQ_ID_DEBUG_DUMP    0xC0
00050 #define SIGMA0_REQ_ID_COV    0xE0
00051
00052 #define SIGMA0_IS_MAGIC_REQ(d1) \
00053     ((d1 & SIGMA0_REQ_MASK) == SIGMA0_REQ_MAGIC)
00054
00055 #define SIGMA0_REQ(x) \
00056     (SIGMA0_REQ_MAGIC + SIGMA0_REQ_ID_ ## x)
00057
00058 /* Use these constants in your code! */
00059 #define SIGMA0_REQ_FPAGE_RAM    (SIGMA0_REQ(FPAGE_RAM))
00060 #define SIGMA0_REQ_FPAGE_IOMEM    (SIGMA0_REQ(FPAGE_IOMEM))
00061 #define SIGMA0_REQ_FPAGE_IOMEM_CACHED    (SIGMA0_REQ(FPAGE_IOMEM_CACHED))
00062 #define SIGMA0_REQ_FPAGE_ANY    (SIGMA0_REQ(FPAGE_ANY))
00063 #define SIGMA0_REQ_KIP    (SIGMA0_REQ(KIP))
00064 #define SIGMA0_REQ_DEBUG_DUMP    (SIGMA0_REQ(DEBUG_DUMP))
00065 #define SIGMA0_REQ_COV    (SIGMA0_REQ(COV))
00066
00072
00076 enum l4sigma0_return_flags_t {
00077     L4SIGMA0_OK,
00078     L4SIGMA0_NOTALIGNED,
00079     L4SIGMA0_IPCERROR,
00080     L4SIGMA0_NOFPAGE,
00081     L4SIGMA0_4,
00082     L4SIGMA0_5,
00083     L4SIGMA0_SMALLERFPAGE,
00084 };
00085
00086 L4_BEGIN_DECLS
00087
00097 L4_CV l4_kernel_info_t *
00098 l4sigma0_map_kip(l4_cap_idx_t sigma0, void *addr, unsigned log2_size);
00099
00129 L4_CV int l4sigma0_map_mem(l4_cap_idx_t sigma0,
00130                             l4_addr_t phys, l4_addr_t virt, l4_addr_t size);
00131
00154 L4_CV int l4sigma0_map_iomem(l4_cap_idx_t sigma0, l4_addr_t phys,
00155                               l4_addr_t virt, l4_addr_t size, int cached);
00180 L4_CV int l4sigma0_map_anypage(l4_cap_idx_t sigma0, l4_addr_t map_area,
00181                                unsigned log2_map_size, l4_addr_t *base,
00182                                unsigned sz);
00183
00192 L4_CV void l4sigma0_debug_dump(l4_cap_idx_t sigma0);
00193
00201 L4_INLINE char const *l4sigma0_map_errstr(int err);
00202
00203 L4_CV void l4sigma0_print_cov_data(l4_cap_idx_t pager);
00204
00206
00207
00208 /* Implementations */
00209
00210 L4_INLINE char const *l4sigma0_map_errstr(int err)
00211 {
00212     switch (err)
00213     {
00214         case 0: return "No error";
00215         case -1: return "Phys, virt or size not aligned";
00216         case -2: return "IPC error";
00217         case -3: return "No fpage received";
00218         #ifndef SIGMA0_REQ_MAGIC
00219             case -4: return "Bad physical address (old protocol only)";
00220         #endif
00221         case -6: return "Superpage requested but smaller flexpage received";
00222         case -7: return "Cannot map I/O memory cacheable (old protocol only)";
00223         default: return "Unknown error";
00224     }
00225 }
00226

```

```

00227
00228 L4\_END\_DECLS
00229
00230 #endif /* ! __L4_SIGMA0_SIGMA0_H */

```

16.434 [__kernel_object_impl.h](#)

```

00001
00006 #pragma once
00007
00008 #include <l4/sys/ipc.h>
00009 #include <l4/sys/kdebug.h>
00010
00011 L4\_INLINE l4\_msgtag\_t
00012 l4\_invoke\_debugger(l4\_cap\_idx\_t obj, l4\_msgtag\_t tag, l4\_utcb\_t *utcb) L4\_NOTHROW
00013 {
00014     l4\_msgtag\_t t2;
00015     unsigned const words = l4\_msgtag\_words(tag);
00016     l4\_msg\_regs\_t *mr = l4\_utcb\_mr\_u(utcb);
00017
00018     if (l4\_is\_invalid\_cap(obj))
00019         return l4\_msgtag(-L4\_EINVAL, 0, 0, 0);
00020
00021     if (words + 2 > L4\_UTCB\_GENERIC\_DATA\_SIZE)
00022         return l4\_msgtag(-L4\_MSGTOOLONG, 0, 0, 0);
00023
00024     mr->mr[0] += L4\_KDEBUG\_GROUP\_KOBJ;
00025     mr->mr[words] = L4\_ITEM\_MAP;
00026     mr->mr[words + 1] = l4\_obj\_fpage(obj, 0, L4\_CAP\_FPAGE\_RWS).raw;
00027     t2 = l4\_msgtag(L4\_PROTO\_DEBUGGER, words, 1, l4\_msgtag\_flags(tag));
00028
00029     return l4\_ipc\_call(L4\_BASE\_DEBUGGER\_CAP, utcb, t2, L4\_IPC\_NEVER);
00030 }
00031

```

16.435 [l4/sys/__ktrace-impl.h](#) File Reference

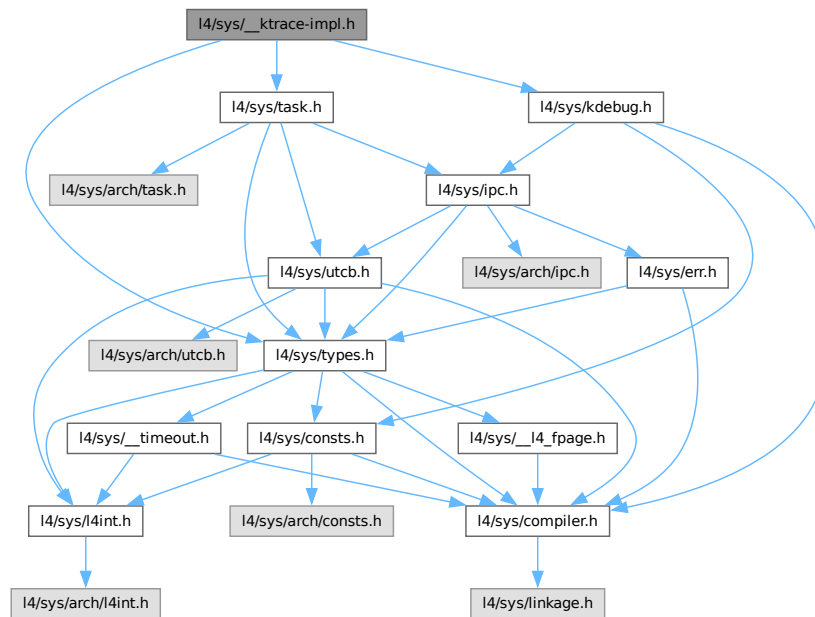
[L4](#) kernel event tracing.

```

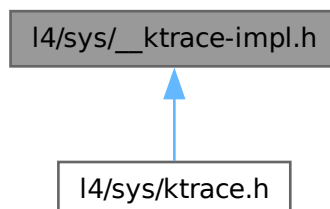
#include <l4/sys/types.h>
#include <l4/sys/kdebug.h>
#include <l4/sys/task.h>

```

Include dependency graph for ___ktrace-impl.h:



This graph shows which files directly or indirectly include this file:



Functions

- [l4_msgtag_t fiasco_tbuf_validate](#) (void)
Validate the kernel base debugger capability.
- [l4_umword_t fiasco_tbuf_log](#) (const char *text)
Create new trace-buffer entry with describing <text>.
- [l4_umword_t fiasco_tbuf_log_3val](#) (const char *text, [l4_umword_t](#) v1, [l4_umword_t](#) v2, [l4_umword_t](#) v3)
Create new trace-buffer entry with describing <text> and three additional values.
- void [fiasco_tbuf_clear](#) (void)
Clear trace-buffer.
- void [fiasco_tbuf_dump](#) (void)

Dump trace-buffer to kernel console.

- [l4_umword_t fiasco_tbuf_log_binary](#) (const unsigned char *data)

Create new trace-buffer entry with binary data.

- [l4_msgtag_t fiasco_tbuf_map_status](#) (l4_fpage_t *ku_mem)

Map kernel trace-buffer status page.

- [l4_msgtag_t fiasco_tbuf_map_slots](#) (l4_fpage_t *ku_mem)

Map kernel trace-buffer slots.

16.435.1 Detailed Description

[L4](#) kernel event tracing.

Definition in file [__ktrace-impl.h](#).

16.436 [__ktrace-impl.h](#)

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *          Björn Döbel <doebel@os.inf.tu-dresden.de>,
00009  *          Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010  *          economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014 #pragma once
00015
00016 #include <l4/sys/types.h>
00017 #include <l4/sys/kdebug.h>
00018 #include <l4/sys/task.h>
00019
00020 /*****
00021  *** Implementation
00022  *****/
00023
00024 L4_INLINE l4_msgtag_t
00025 fiasco_tbuf_validate(void)
00026 { return l4_task_cap_valid(L4_BASE_TASK_CAP, L4_BASE_DEBUGGER_CAP); }
00027
00028 L4_INLINE l4_umword_t
00029 fiasco_tbuf_log(const char *text)
00030 {
00031     enum { TBUF_LOG = L4_KDEBUG_GROUP_TRACE + 0x01 };
00032     return l4_error(__l4_kdebug_text(TBUF_LOG, text, __builtin_strlen(text)));
00033 }
00034
00035 L4_INLINE l4_umword_t
00036 fiasco_tbuf_log_3val(const char *text, l4_umword_t v1, l4_umword_t v2,
00037                     l4_umword_t v3)
00038 {
00039     enum { TBUF_LOG_3VAL = L4_KDEBUG_GROUP_TRACE + 0x04 };
00040     return l4_error(__l4_kdebug_3_text(TBUF_LOG_3VAL, text,
00041                                       __builtin_strlen(text), v1, v2, v3));
00042 }
00043
00044 L4_INLINE void
00045 fiasco_tbuf_clear(void)
00046 {
00047     enum { TBUF_CLEAR = L4_KDEBUG_GROUP_TRACE + 0x02 };
00048     __l4_kdebug_op(TBUF_CLEAR);
00049 }
00050
00051 L4_INLINE void
00052 fiasco_tbuf_dump(void)
00053 {
00054     enum { TBUF_DUMP = L4_KDEBUG_GROUP_TRACE + 0x03 };
00055     __l4_kdebug_op(TBUF_DUMP);
00056 }
00057

```

```

00058 L4_INLINE l4_umword_t
00059 fiasco_tbuf_log_binary(const unsigned char *data)
00060 {
00061     enum { TBUF_LOG_BIN = L4_KDEBUG_GROUP_TRACE + 0x08 };
00062     return l4_error(__l4_kdebug_text(TBUF_LOG_BIN, (const char *)data, 24));
00063 }
00064
00065 L4_INLINE l4_msgtag_t
00066 fiasco_tbuf_map_status(l4_fpage_t *ku_mem)
00067 {
00068     enum { TBUF_MAP_STATUS = L4_KDEBUG_GROUP_TRACE + 0x10 };
00069
00070     l4_utcb_t *utcb = l4_utcb();
00071     l4_msg_regs_t *mr = l4_utcb_mr_u(utcb);
00072     l4_msgtag_t ret;
00073
00074     mr->mr[0] = TBUF_MAP_STATUS;
00075     mr->mr[1] = ku_mem->raw;
00076     ret = l4_ipc_call(L4_BASE_DEBUGGER_CAP, utcb,
00077                     l4_msgtag(L4_PROTO_DEBUGGER, 2, 0, 0), L4_IPC_NEVER);
00078     if (!l4_msgtag_has_error(ret))
00079         ku_mem->raw = mr->mr[0];
00080
00081     return ret;
00082 }
00083
00084 L4_INLINE l4_msgtag_t
00085 fiasco_tbuf_map_slots(l4_fpage_t *ku_mem)
00086 {
00087     enum { TBUF_MAP_SLOTS = L4_KDEBUG_GROUP_TRACE + 0x11 };
00088
00089     l4_utcb_t *utcb = l4_utcb();
00090     l4_msg_regs_t *mr = l4_utcb_mr_u(utcb);
00091     l4_msgtag_t ret;
00092
00093     mr->mr[0] = TBUF_MAP_SLOTS;
00094     mr->mr[1] = ku_mem->raw;
00095     ret = l4_ipc_call(L4_BASE_DEBUGGER_CAP, utcb,
00096                     l4_msgtag(L4_PROTO_DEBUGGER, 2, 0, 0), L4_IPC_NEVER);
00097     if (!l4_msgtag_has_error(ret))
00098         ku_mem->raw = mr->mr[0];
00099
00100     return ret;
00101 }

```

16.437 __l4_fpage.h

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *      Björn Döbel <doebel@os.inf.tu-dresden.de>,
00010  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00011  *      economic rights: Technische Universität Dresden (Germany)
00012  *
00013  * License: see LICENSE.spdx (in this directory or the directories above)
00014  */
00015 #pragma once
00016
00017 #include <l4/sys/compiler.h>
00018
00043
00048 enum l4_fpage_consts
00049 {
00050     L4_FPAGE_RIGHTS_SHIFT = 0,
00051     L4_FPAGE_TYPE_SHIFT   = 4,
00052     L4_FPAGE_SIZE_SHIFT   = 6,
00053     L4_FPAGE_ADDR_SHIFT   = 12,
00054
00055     L4_FPAGE_RIGHTS_BITS = 4,
00056     L4_FPAGE_TYPE_BITS  = 2,
00057     L4_FPAGE_SIZE_BITS  = 6,
00058     L4_FPAGE_ADDR_BITS  = L4_MWORD_BITS - L4_FPAGE_ADDR_SHIFT,
00059
00061     L4_FPAGE_RIGHTS_MASK = ((1UL < L4_FPAGE_RIGHTS_BITS) - 1)
00062                          < L4_FPAGE_RIGHTS_SHIFT,
00063     L4_FPAGE_TYPE_MASK  = ((1UL < L4_FPAGE_TYPE_BITS) - 1)
00064                          < L4_FPAGE_TYPE_SHIFT,
00065     L4_FPAGE_SIZE_MASK  = ((1UL < L4_FPAGE_SIZE_BITS) - 1)
00066                          < L4_FPAGE_SIZE_SHIFT,
00067     L4_FPAGE_ADDR_MASK  = ~0UL < L4_FPAGE_ADDR_SHIFT,
00069     L4_FPAGE_RIGHTS_ALL = L4_FPAGE_RIGHTS_MASK,
00070 };

```

```

00071
00076 typedef union {
00077     l4_umword_t fpage;
00078     l4_umword_t raw;
00079 } l4_fpage_t;
00080
00084 enum
00085 {
00092     L4_WHOLE_ADDRESS_SPACE = 63
00093 };
00094
00099 typedef struct {
00100     l4_umword_t snd_base;
00101     l4_fpage_t fpage;
00102 } l4_snd_fpage_t;
00103
00104
00118 enum L4_fpage_rights
00119 {
00120     L4_FPAGE_X      = 1,
00121     L4_FPAGE_W      = 2,
00122     L4_FPAGE_RO     = 4,
00123     L4_FPAGE_RW     = L4_FPAGE_RO | L4_FPAGE_W,
00124     L4_FPAGE_RX     = L4_FPAGE_RO | L4_FPAGE_X,
00125     L4_FPAGE_RWX    = L4_FPAGE_RW | L4_FPAGE_X,
00126 };
00127
00148 enum L4_cap_fpage_rights
00149 {
00157     L4_CAP_FPAGE_W    = 0x1,
00169     L4_CAP_FPAGE_S    = 0x2,
00175     L4_CAP_FPAGE_R    = 0x4,
00176     L4_CAP_FPAGE_RO   = 0x4,
00185     L4_CAP_FPAGE_D    = 0x8,
00192     L4_CAP_FPAGE_RW   = L4_CAP_FPAGE_R | L4_CAP_FPAGE_W,
00199     L4_CAP_FPAGE_RS   = L4_CAP_FPAGE_R | L4_CAP_FPAGE_S,
00206     L4_CAP_FPAGE_RWS  = L4_CAP_FPAGE_RW | L4_CAP_FPAGE_S,
00212     L4_CAP_FPAGE_RWSD = L4_CAP_FPAGE_RWS | L4_CAP_FPAGE_D,
00218     L4_CAP_FPAGE_RWD  = L4_CAP_FPAGE_RW | L4_CAP_FPAGE_D,
00224     L4_CAP_FPAGE_RSD  = L4_CAP_FPAGE_RS | L4_CAP_FPAGE_D,
00225 };
00226
00230 enum L4_fpage_type
00231 {
00232     L4_FPAGE_SPECIAL = 0,
00235     L4_FPAGE_MEMORY  = 1,
00236     L4_FPAGE_IO      = 2,
00237     L4_FPAGE_OBJ     = 3,
00238 };
00239
00243 enum L4_fpage_control
00244 {
00247     L4_FPAGE_CONTROL_OFFSET_SHIFT = 12,
00250     L4_FPAGE_CONTROL_MASK = ~0UL << L4_FPAGE_CONTROL_OFFSET_SHIFT,
00251 };
00252
00264
00266 #define L4_FPAGE_C_REF_CNT      0x00
00267
00269 #define L4_FPAGE_C_NO_REF_CNT  0x10
00270
00272 #define L4_FPAGE_C_OBJ_RIGHT1  0x20
00273
00275 #define L4_FPAGE_C_OBJ_RIGHT2  0x40
00276
00278 #define L4_FPAGE_C_OBJ_RIGHT3  0x80
00279
00281 #define L4_FPAGE_C_OBJ_RIGHTS  0xe0
00282
00288 #define L4_FPAGE_C_IPCGATE_SVR L4_FPAGE_C_OBJ_RIGHT1
00289
00291
00292
00303 enum l4_fpage_cacheability_opt_t
00304 {
00307     L4_FPAGE_CACHE_OPT    = 0x1,
00308
00311     L4_FPAGE_CACHEABLE    = 0x3,
00312
00315     L4_FPAGE_BUFFERABLE   = 0x5,
00316
00319     L4_FPAGE_UNCACHEABLE  = 0x1
00320 };
00321
00322
00326 enum
00327 {

```



```

00331     L4_WHOLE_IOADDRESS_SPACE = 16,
00332
00334     L4_IOPORT_MAX             = (1L << L4_WHOLE_IOADDRESS_SPACE)
00335 };
00336
00337
00338
00353 L4_INLINE L4_CONSTEXPR l4_fpage_t
00354 l4_fpage(l4_addr_t address, unsigned int order, unsigned char rights) L4_NOTHROW;
00355
00367 L4_INLINE L4_CONSTEXPR l4_fpage_t
00368 l4_fpage_all(void) L4_NOTHROW;
00369
00376 L4_INLINE L4_CONSTEXPR l4_fpage_t
00377 l4_fpage_invalid(void) L4_NOTHROW;
00378
00379
00390 L4_INLINE L4_CONSTEXPR l4_fpage_t
00391 l4_iofpage(unsigned long port, unsigned int order) L4_NOTHROW;
00392
00393
00406 L4_INLINE L4_CONSTEXPR l4_fpage_t
00407 l4_obj_fpage(l4_cap_idx_t obj, unsigned int order, unsigned char rights) L4_NOTHROW;
00408
00418 L4_INLINE L4_CONSTEXPR int
00419 l4_is_fpage_writable(l4_fpage_t fp) L4_NOTHROW;
00420
00421
00452
00468 L4_INLINE L4_CONSTEXPR l4_umword_t
00469 l4_map_control(l4_umword_t spot, unsigned char cache, unsigned grant) L4_NOTHROW;
00470
00484 L4_INLINE L4_CONSTEXPR l4_umword_t
00485 l4_map_obj_control(l4_umword_t spot, unsigned grant) L4_NOTHROW;
00486
00495 L4_INLINE L4_CONSTEXPR unsigned
00496 l4_fpage_rights(l4_fpage_t f) L4_NOTHROW;
00497
00506 L4_INLINE L4_CONSTEXPR unsigned
00507 l4_fpage_type(l4_fpage_t f) L4_NOTHROW;
00508
00519 L4_INLINE L4_CONSTEXPR unsigned
00520 l4_fpage_size(l4_fpage_t f) L4_NOTHROW;
00521
00532 L4_INLINE L4_CONSTEXPR unsigned long
00533 l4_fpage_page(l4_fpage_t f) L4_NOTHROW;
00534
00548 L4_INLINE L4_CONSTEXPR l4_addr_t
00549 l4_fpage_memaddr(l4_fpage_t f) L4_NOTHROW;
00550
00564 L4_INLINE L4_CONSTEXPR l4_cap_idx_t
00565 l4_fpage_obj(l4_fpage_t f) L4_NOTHROW;
00566
00580 L4_INLINE L4_CONSTEXPR unsigned long
00581 l4_fpage_ioport(l4_fpage_t f) L4_NOTHROW;
00582
00592 L4_INLINE L4_CONSTEXPR l4_fpage_t
00593 l4_fpage_set_rights(l4_fpage_t src, unsigned char new_rights) L4_NOTHROW;
00594
00606 L4_INLINE L4_CONSTEXPR int
00607 l4_fpage_contains(l4_fpage_t fpage, l4_addr_t addr, unsigned order) L4_NOTHROW;
00608
00625 L4_INLINE L4_CONSTEXPR unsigned char
00626 l4_fpage_max_order(unsigned char order, l4_addr_t addr,
00627                    l4_addr_t min_addr, l4_addr_t max_addr,
00628                    l4_addr_t hotspot L4_DEFAULT_PARAM(0));
00629
00639 L4_INLINE L4_CONSTEXPR int
00640 l4_is_fpage_valid(l4_fpage_t fp) L4_NOTHROW;
00641
00642 /*****
00643  * Implementations
00644  *****/
00645
00646 L4_INLINE L4_CONSTEXPR unsigned
00647 l4_fpage_rights(l4_fpage_t f) L4_NOTHROW
00648 {
00649     return (f.raw & L4_FPAGE_RIGHTS_MASK) >> L4_FPAGE_RIGHTS_SHIFT;
00650 }
00651
00652 L4_INLINE L4_CONSTEXPR unsigned
00653 l4_fpage_type(l4_fpage_t f) L4_NOTHROW
00654 {
00655     return (f.raw & L4_FPAGE_TYPE_MASK) >> L4_FPAGE_TYPE_SHIFT;
00656 }
00657
00658 L4_INLINE L4_CONSTEXPR unsigned

```

```

00659 l4_fpage_size(l4_fpage_t f) L4_NOTHROW
00660 {
00661     return (f.raw & L4_FPAGE_SIZE_MASK) » L4_FPAGE_SIZE_SHIFT;
00662 }
00663
00664 L4_INLINE L4_CONSTEXPR unsigned long
00665 l4_fpage_page(l4_fpage_t f) L4_NOTHROW
00666 {
00667     return (f.raw & L4_FPAGE_ADDR_MASK) » L4_FPAGE_ADDR_SHIFT;
00668 }
00669
00670 L4_INLINE L4_CONSTEXPR unsigned long
00671 l4_fpage_ioport(l4_fpage_t f) L4_NOTHROW
00672 {
00673     return (f.raw & L4_FPAGE_ADDR_MASK) » L4_FPAGE_ADDR_SHIFT;
00674 }
00675
00676 L4_INLINE L4_CONSTEXPR l4_addr_t
00677 l4_fpage_memaddr(l4_fpage_t f) L4_NOTHROW
00678 {
00679     return f.raw & L4_FPAGE_ADDR_MASK;
00680 }
00681
00682 L4_INLINE L4_CONSTEXPR l4_cap_idx_t
00683 l4_fpage_obj(l4_fpage_t f) L4_NOTHROW
00684 {
00685     return f.raw & L4_FPAGE_ADDR_MASK;
00686 }
00687
00688 L4_INLINE L4_CONSTEXPR l4_fpage_t
00689 __l4_fpage_generic(unsigned long address, unsigned int type,
00690                    unsigned int order, unsigned char rights) L4_NOTHROW;
00691
00692 L4_INLINE L4_CONSTEXPR l4_fpage_t
00693 __l4_fpage_generic(unsigned long address, unsigned int type,
00694                    unsigned int order, unsigned char rights) L4_NOTHROW
00695 {
00696     l4_fpage_t t =
00697     {
00698         ((rights « L4_FPAGE_RIGHTS_SHIFT) & L4_FPAGE_RIGHTS_MASK)
00699         | ((type « L4_FPAGE_TYPE_SHIFT) & L4_FPAGE_TYPE_MASK)
00700         | ((order « L4_FPAGE_SIZE_SHIFT) & L4_FPAGE_SIZE_MASK)
00701         | ((address & L4_FPAGE_ADDR_MASK) & L4_FPAGE_ADDR_MASK)
00702     };
00703     return t;
00704 }
00705
00706 L4_INLINE L4_CONSTEXPR l4_fpage_t
00707 l4_fpage_set_rights(l4_fpage_t src, unsigned char new_rights) L4_NOTHROW
00708 {
00709     l4_fpage_t f =
00710     {
00711         ((L4_FPAGE_TYPE_MASK | L4_FPAGE_SIZE_MASK | L4_FPAGE_ADDR_MASK) & src.raw)
00712         | ((new_rights « L4_FPAGE_RIGHTS_SHIFT) & L4_FPAGE_RIGHTS_MASK)
00713     };
00714     return f;
00715 }
00716
00717 L4_INLINE L4_CONSTEXPR l4_fpage_t
00718 l4_fpage(l4_addr_t address, unsigned int order, unsigned char rights) L4_NOTHROW
00719 {
00720     return __l4_fpage_generic(address, L4_FPAGE_MEMORY, order, rights);
00721 }
00722
00723 L4_INLINE L4_CONSTEXPR l4_fpage_t
00724 l4_iofpage(unsigned long port, unsigned int order) L4_NOTHROW
00725 {
00726     return __l4_fpage_generic(port « L4_FPAGE_ADDR_SHIFT, L4_FPAGE_IO, order, L4_FPAGE_RW);
00727 }
00728
00729 L4_INLINE L4_CONSTEXPR l4_fpage_t
00730 l4_obj_fpage(l4_cap_idx_t obj, unsigned int order, unsigned char rights) L4_NOTHROW
00731 {
00732     static_assert((unsigned long)L4_CAP_SHIFT >= L4_FPAGE_ADDR_SHIFT,
00733                  "Capability index does not fit into fpage.");
00734     return __l4_fpage_generic(obj, L4_FPAGE_OBJ, order, rights);
00735 }
00736
00737 L4_INLINE L4_CONSTEXPR l4_fpage_t
00738 l4_fpage_all(void) L4_NOTHROW
00739 {
00740     return __l4_fpage_generic(0, L4_FPAGE_SPECIAL, L4_WHOLE_ADDRESS_SPACE, 0);
00741 }
00742
00743 L4_INLINE L4_CONSTEXPR l4_fpage_t
00744 l4_fpage_invalid(void) L4_NOTHROW
00745 {
00746 }

```

```

00747     return __l4_fpage_generic(0, L4_FPAGE_SPECIAL, 0, 0);
00748 }
00749
00750
00751 L4_INLINE L4_CONSTEXPR int
00752 l4_is_fpage_writable(l4_fpage_t fp) L4_NOTHROW
00753 {
00754     return l4_fpage_rights(fp) & L4_FPAGE_W;
00755 }
00756
00757 L4_INLINE L4_CONSTEXPR l4_umword_t
00758 l4_map_control(l4_umword_t snd_base, unsigned char cache, unsigned grant) L4_NOTHROW
00759 {
00760     return (snd_base & L4_FPAGE_CONTROL_MASK)
00761         | ((l4_umword_t)cache << 4) | L4_ITEM_MAP | grant;
00762 }
00763
00764 L4_INLINE L4_CONSTEXPR l4_umword_t
00765 l4_map_obj_control(l4_umword_t snd_base, unsigned grant) L4_NOTHROW
00766 {
00767     return l4_map_control(snd_base, 0, grant);
00768 }
00769
00770 L4_INLINE L4_CONSTEXPR int
00771 l4_fpage_contains(l4_fpage_t fpage, l4_addr_t addr, unsigned log2size) L4_NOTHROW
00772 {
00773     l4_addr_t fa = l4_fpage_memaddr(fpage);
00774     return (fa <= addr)
00775         && (fa + (1UL << l4_fpage_size(fpage)) >= addr + (1UL << log2size));
00776 }
00777
00778 L4_INLINE L4_CONSTEXPR unsigned char
00779 l4_fpage_max_order(unsigned char order, l4_addr_t addr,
00780                   l4_addr_t min_addr, l4_addr_t max_addr,
00781                   l4_addr_t hotspot)
00782 {
00783     while (order < 30 /* limit to 1GB flexpages */)
00784     {
00785         l4_addr_t mask = ~(~0UL << (order + 1));
00786         l4_addr_t base = l4_trunc_size(addr, order + 1);
00787         if (base < min_addr)
00788             return order;
00789
00790         if (base + (1UL << (order + 1)) - 1 > max_addr - 1)
00791             return order;
00792
00793         if (hotspot == ~0UL || ((addr ^ hotspot) & mask))
00794             break;
00795
00796         ++order;
00797     }
00798
00799     return order;
00800 }
00801
00802 L4_INLINE L4_CONSTEXPR int
00803 l4_is_fpage_valid(l4_fpage_t fp) L4_NOTHROW
00804 {
00805     return l4_fpage_type(fp) != L4_FPAGE_SPECIAL || l4_fpage_size(fp) != 0;
00806 }

```

16.438 __platform_control-arm.h

```

00001 /*
00002  * Copyright (C) 2024 Kernkonzept GmbH.
00003  * Author(s): Jan Klötzke <jan.kloetzke@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include <l4/sys/types.h>
00010
00011
00029 L4_INLINE l4_msgtag_t
00030 l4_platform_ctl_set_task_asid(l4_cap_idx_t pfc,
00031                               l4_cap_idx_t task,
00032                               l4_umword_t asid) L4_NOTHROW;
00033
00034
00038 L4_INLINE l4_msgtag_t
00039 l4_platform_ctl_set_task_asid_u(l4_cap_idx_t pfc,
00040                                 l4_cap_idx_t task,

```

```

00041                                     l4_umword_t asid,
00042                                     l4_utcb_t *utcb) L4_NOTHROW;
00043
00044 /* IMPLEMENTATION ----- */
00045
00046 L4_INLINE l4_msgtag_t
00047 l4_platform_ctl_set_task_asid_u(l4_cap_idx_t pfc,
00048                                 l4_cap_idx_t task,
00049                                 l4_umword_t asid,
00050                                 l4_utcb_t *utcb) L4_NOTHROW
00051 {
00052     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00053     v->mr[0] = L4_PLATFORM_CTL_SET_TASK_ASID_OP;
00054     v->mr[1] = asid;
00055     v->mr[2] = l4_map_obj_control(0,0);
00056     v->mr[3] = l4_obj_fpage(task, 0, L4_CAP_FPAGE_RWS).raw;
00057     return l4_ipc_call(pfc, utcb, l4_msgtag(L4_PROTO_PLATFORM_CTL, 2, 1, 0),
00058                       L4_IPC_NEVER);
00059 }
00060
00061 L4_INLINE l4_msgtag_t
00062 l4_platform_ctl_set_task_asid(l4_cap_idx_t pfc,
00063                               l4_cap_idx_t task,
00064                               l4_umword_t asid) L4_NOTHROW
00065 {
00066     return l4_platform_ctl_set_task_asid_u(pfc, task, asid, l4_utcb());
00067 }

```

16.439 __task-arm.h

```

00001 /*
00002  * (c) 2018 Adam Lackorzynski <adam@l4re.org>
00003  *
00004  * License: see LICENSE.spdx (in this directory or the directories above)
00005  */
00006 #pragma once
00007
00008 #include <l4/sys/types.h>
00009
00013 L4_INLINE l4_msgtag_t
00014 l4_task_vgicc_map_u(l4_cap_idx_t task, l4_fpage_t vgicc_fpage,
00015                    l4_utcb_t *u) L4_NOTHROW;
00016
00028 L4_INLINE l4_msgtag_t
00029 l4_task_vgicc_map(l4_cap_idx_t task, l4_fpage_t vgicc_fpage) L4_NOTHROW;
00030
00031 /* IMPLEMENTATION ----- */
00032
00033 #include <l4/sys/ipc.h>
00034
00035 L4_INLINE l4_msgtag_t
00036 l4_task_vgicc_map_u(l4_cap_idx_t task, l4_fpage_t vgicc_fpage,
00037                    l4_utcb_t *u) L4_NOTHROW
00038 {
00039     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00040     v->mr[0] = L4_TASK_MAP_VGICC_ARM_OP;
00041     v->mr[1] = vgicc_fpage.raw;
00042     return l4_ipc_call(task, u, l4_msgtag(L4_PROTO_TASK, 2, 0, 0), L4_IPC_NEVER);
00043 }
00044
00045 L4_INLINE l4_msgtag_t
00046 l4_task_vgicc_map(l4_cap_idx_t task, l4_fpage_t vgicc_fpage) L4_NOTHROW
00047 {
00048     return l4_task_vgicc_map_u(task, vgicc_fpage, l4_utcb());
00049 }

```

16.440 __timeout.h

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014 #ifndef L4_SYS_TIMEOUT_H__

```

```

00015 #define L4_SYS_TIMEOUT_H__
00016
00017 #include <l4/sys/l4int.h>
00018 #include <l4/sys/compiler.h>
00019
00025
00040 typedef struct l4_timeout_s {
00041     l4_uint16_t t;
00042 } __attribute__((packed)) l4_timeout_s;
00043
00044
00052 typedef union l4_timeout_t
00053 {
00054     l4_uint32_t raw;
00055     struct
00056     {
00057         #ifdef __BIG_ENDIAN__
00058             l4_timeout_s snd;
00059             l4_timeout_s rcv;
00060         #else
00061             l4_timeout_s rcv;
00062             l4_timeout_s snd;
00063         #endif
00064     } p;
00065 } l4_timeout_t;
00066
00067
00073 #define L4_IPC_TIMEOUT_0 ((l4_timeout_s){0x0400})
00074 #define L4_IPC_TIMEOUT_NEVER ((l4_timeout_s){0})
00075 #define L4_IPC_NEVER_INITIALIZER {0}
00076 #define L4_IPC_NEVER ((l4_timeout_t){0})
00077 #define L4_IPC_RECV_TIMEOUT_0 ((l4_timeout_t){0x00000400})
00078 #define L4_IPC_SEND_TIMEOUT_0 ((l4_timeout_t){0x04000000})
00079 #define L4_IPC_BOTH_TIMEOUT_0 ((l4_timeout_t){0x04000400})
00080
00085 #define L4_TIMEOUT_US_NEVER (~0ULL)
00086
00091 #define L4_TIMEOUT_US_MAX ((1ULL << 41) - 1)
00092
00094
00104 L4_CONSTEXPR L4_INLINE
00105 l4_timeout_s l4_timeout_rel(unsigned man, unsigned exp) L4_NOTHROW;
00106
00107
00117 L4_CONSTEXPR L4_INLINE
00118 l4_timeout_t l4_ipc_timeout(unsigned snd_man, unsigned snd_exp,
00119                             unsigned rcv_man, unsigned rcv_exp) L4_NOTHROW;
00120
00130 L4_CONSTEXPR L4_INLINE
00131 l4_timeout_t l4_timeout(l4_timeout_s snd, l4_timeout_s rcv) L4_NOTHROW;
00132
00140 L4_CONSTEXPR L4_INLINE
00141 void l4_snd_timeout(l4_timeout_s snd, l4_timeout_t *to) L4_NOTHROW;
00142
00150 L4_CONSTEXPR L4_INLINE
00151 void l4_rcv_timeout(l4_timeout_s rcv, l4_timeout_t *to) L4_NOTHROW;
00152
00161 L4_CONSTEXPR L4_INLINE
00162 l4_kernel_clock_t l4_timeout_rel_get(l4_timeout_s to) L4_NOTHROW;
00163
00164
00173 L4_CONSTEXPR L4_INLINE
00174 unsigned l4_timeout_is_absolute(l4_timeout_s to) L4_NOTHROW;
00175
00185 L4_CONSTEXPR L4_INLINE
00186 l4_kernel_clock_t l4_timeout_get(l4_kernel_clock_t cur, l4_timeout_s to) L4_NOTHROW;
00187
00199 L4_CONSTEXPR L4_INLINE
00200 l4_timeout_s l4_timeout_from_us(l4_uint64_t us) L4_NOTHROW;
00201
00202 /*
00203  * Implementation
00204  */
00205
00206 L4_CONSTEXPR L4_INLINE
00207 l4_timeout_t l4_ipc_timeout(unsigned snd_man, unsigned snd_exp,
00208                             unsigned rcv_man, unsigned rcv_exp) L4_NOTHROW
00209 {
00210     l4_uint16_t snd = (snd_man & 0x3ff) | ((snd_exp << 10) & 0x7c00);
00211     l4_uint16_t rcv = (rcv_man & 0x3ff) | ((rcv_exp << 10) & 0x7c00);
00212     return l4_timeout((l4_timeout_s){snd}, (l4_timeout_s){rcv});
00213 }
00214
00215
00216 L4_CONSTEXPR L4_INLINE
00217 l4_timeout_t l4_timeout(l4_timeout_s snd, l4_timeout_s rcv) L4_NOTHROW
00218 {

```

```

00219     return (l4_timeout_t){ ((l4_uint32_t){snd.t} << 16) | rcv.t };
00220 }
00221
00222
00223 L4_CONSTEXPR L4_INLINE
00224 void l4_snd_timeout(l4_timeout_s snd, l4_timeout_t *to) L4_NOTHROW
00225 {
00226     to->p.snd = snd;
00227 }
00228
00229
00230 L4_CONSTEXPR L4_INLINE
00231 void l4_rcv_timeout(l4_timeout_s rcv, l4_timeout_t *to) L4_NOTHROW
00232 {
00233     to->p.rcv = rcv;
00234 }
00235
00236
00237 L4_CONSTEXPR L4_INLINE
00238 l4_timeout_s l4_timeout_rel(unsigned man, unsigned exp) L4_NOTHROW
00239 {
00240     return (l4_timeout_s){ (l4_uint16_t){ (man & 0x3ff) | ((exp << 10) & 0x7c00) } };
00241 }
00242
00243
00244 L4_CONSTEXPR L4_INLINE
00245 l4_kernel_clock_t l4_timeout_rel_get(l4_timeout_s to) L4_NOTHROW
00246 {
00247     if (to.t == 0)
00248         return ~0ULL;
00249     return (l4_kernel_clock_t)(to.t & 0x3ff) << ((to.t > 10) & 0x1f);
00250 }
00251
00252
00253 L4_CONSTEXPR L4_INLINE
00254 unsigned l4_timeout_is_absolute(l4_timeout_s to) L4_NOTHROW
00255 {
00256     return to.t & 0x8000;
00257 }
00258
00259
00260 L4_CONSTEXPR L4_INLINE
00261 l4_kernel_clock_t l4_timeout_get(l4_kernel_clock_t cur, l4_timeout_s to) L4_NOTHROW
00262 {
00263     if (l4_timeout_is_absolute(to))
00264         return 0; /* We cannot retrieve the value ... */
00265     else
00266         return cur + l4_timeout_rel_get(to);
00267 }
00268
00269 L4_CONSTEXPR L4_INLINE
00270 l4_timeout_s l4_timeout_from_us(l4_uint64_t us) L4_NOTHROW
00271 {
00272     if (us == 0)
00273         return L4_IPC_TIMEOUT_0;
00274     else if (us == L4_TIMEOUT_US_NEVER || us > L4_TIMEOUT_US_MAX)
00275         return L4_IPC_TIMEOUT_NEVER;
00276     else
00277     {
00278         /* Here it is certain that at least one bit in 'us' is set. */
00279
00280         enum { m_max = 0x3ff, e_max = 0x1f, }; // max values also serve as mask
00281
00282         l4_uint16_t m = 0; // initialization required by constexpr, optimized away
00283         l4_uint16_t v = 0; // initialization required by constexpr, optimized away
00284         int e = (63 - __builtin_clzll(us)) - 9;
00285         if (e < 0)
00286             e = 0;
00287
00288         /* Here it is certain that '0 <= e <= 31' and '1 <= 2^e <= 2^31':
00289          * L4_TIMEOUT_US_MAX = 2^41-1 = 0x000001fffffffffff => e = 31.
00290          * Note: 2^41-1 (0x000001fffffffffff) > 1023*2^31 (0x00001ff800000000). */
00291
00292         /* Round up to next aligned timeout value to not return too early. */
00293         m = (us + ((1ULL << e) - 1)) >> e;
00294         /* Bounds check */
00295         if (m > m_max)
00296         {
00297             if (e < e_max)
00298             {
00299                 ++e;
00300                 m >>= 1;
00301             }
00302             else
00303                 m = m_max;
00304         }
00305     }

```

```

00306      /* Here it is certain that '1 <= m <= 1023. Consider the following cases:
00307      * o 1 <= us <= 1023: e = 0; 2^e = 1; 1 <= us/1 <= 1023
00308      * o 1024 <= us <= 2047: e = 1; 2^e = 2; 512 <= us/2 <= 1023
00309      * o 2048 <= us <= 4095: e = 2; 2^e = 4; 512 <= us/4 <= 1023
00310      * ...
00311      * o 2^31 <= us <= 2^32-1: e = 22; 512 <= us/2^22 <= 1023
00312      * o 2^40 <= us <= 2^41-1: e = 31; 512 <= us/2^31 <= 1023
00313      *
00314      * Dividing by (1<e) ensures that for all us < 2^41: m < 2^10.
00315      *
00316      * Maximum possible timeout using this format: L4_TIMEOUT_US_MAX = 2^41-1:
00317      * e = 31, m = 1023 => 2'196'875'771'904 us = 610h 14m 35s.
00318      */
00319
00320      /* Without introducing 'v' we had to type-cast the expression to
00321      * l4_uint16_t. This cannot be avoided by declaring m and e_pow_10 as
00322      * l4_uint16_t due to C++ integer promotion. */
00323      v = (e << 10) | m;
00324      return (l4_timeout_s){v};
00325    }
00326  }
00327
00328 #endif

```

16.441 l4/sys/__typeinfo.h File Reference

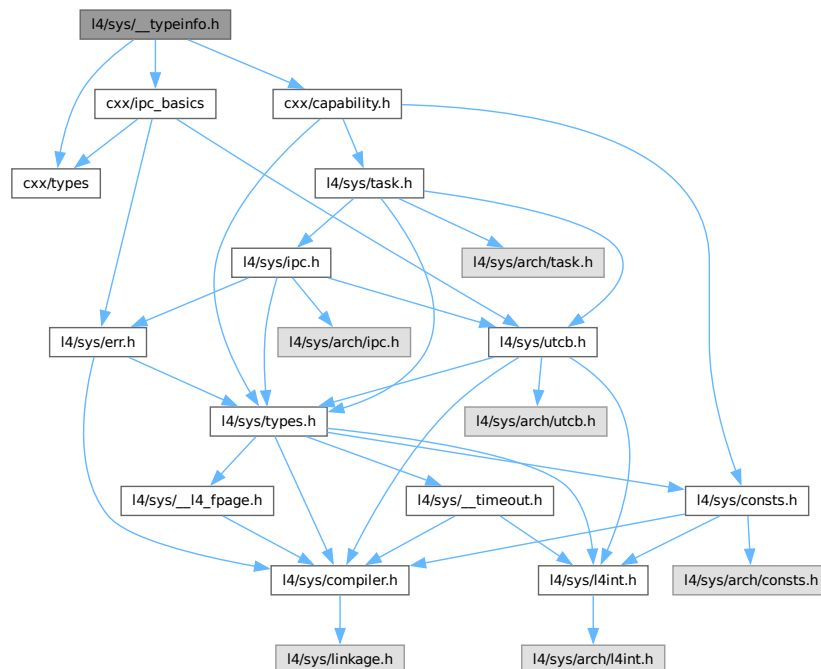
Type information handling.

```

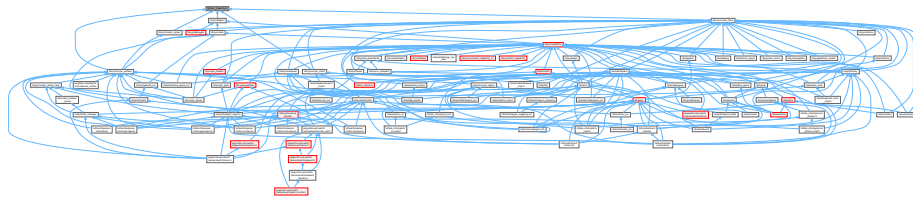
#include "cxx/types"
#include "cxx/ipc_basics"
#include "cxx/capability.h"

```

Include dependency graph for __typeinfo.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [L4::Typeid::P_dispatch< LIST >](#)
Use for protocol based dispatch stage.
- struct [L4::Typeid::Detail::Rpc_end](#)
Internal end-of-list marker.
- struct [L4::Typeid::Detail::_Rpc< OPCODE, O, X >](#)
Empty list of RPCs.
- struct [L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... >](#)
Non-empty list of RPCs.
- struct [L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... >::Rpc< Y >](#)
Find the given RPC in the list.
- struct [L4::Typeid::Detail::_Rpc< OPCODE, O, Default_op< R > >::Rpc< Y >](#)
Find the given RPC in the list.
- struct [L4::Typeid::Raw_ipc< CLASS >](#)
RPCs list for passing raw incoming IPC to the server object.
- struct [L4::Typeid::Rpc< RPCS >](#)
Standard list of RPCs of an interface.
- struct [L4::Typeid::Rpc_code< OPCODE_TYPE >](#)
List of RPCs of an interface using a special opcode type.
- struct [L4::Typeid::Rpc_code< OPCODE_TYPE >::F< RPCS >](#)
- struct [L4::Typeid::Rpc_nocode< OPERATION >](#)
List of RPCs of an interface using a single operation without an opcode.
- struct [L4::Typeid::Rpc_sys< ARG >](#)
List of RPCs typically used for kernel interfaces.
- struct [L4::Type_info](#)
Dynamic Type Information for [L4Re](#) Interfaces.
- class [L4::Type_info::Demand](#)
Data type for expressing the needed receive buffers at the server-side of an interface.
- struct [L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS >](#)
Template type statically describing demand of receive buffers.
- struct [L4::Type_info::Demand_union_t< D1, D2 >](#)
Template type statically describing the combination of two [Demand](#) object.
- struct [L4::Kobject_typeid< T >](#)
Meta object for handling access to type information of Kobjects.
- struct [L4::Kobject_typeid< void >](#)
Minimalistic ID for `void` interface.
- class [L4::Kobject_t< Derived, Base, PROTO, S_DEMAND >](#)
Helper class to create an [L4Re](#) interface class that is derived from a single base class.
- class [L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >](#)
Helper class to create an [L4Re](#) interface class that is derived from two base classes (see [L4::Kobject_t](#)).

- struct [L4::Kobject_3t](#)< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >
Helper class to create an [L4Re](#) interface class that is derived from three base classes (see [L4::Kobject_t](#)).
- struct [L4::Proto_t](#)< P >
Data type for defining protocol numbers.

Namespaces

- namespace [L4](#)
[L4](#) low-level kernel interface.
- namespace [L4::Typeid](#)
Definition of interface data-type helpers.

Typedefs

- typedef int [L4::Opcode](#)
Data type for RPC opcodes.

Enumerations

- enum [l4_proto_t](#) { [L4::PROTO_ANY](#) = 0 , [L4::PROTO_EMPTY](#) = -19 }

Functions

- template<typename T>
[Type_info](#) const * [L4::kobject_typeid](#) () noexcept
*Get the [L4::Type_info](#) for the [L4Re](#) interface given in *T*.*

16.441.1 Detailed Description

Type information handling.

Definition in file [__typeinfo.h](#).

16.442 __typeinfo.h

[Go to the documentation of this file.](#)

```
00001
00005 /*
00006  * Copyright (C) 2014-2017, 2019, 2022-2025 Kernkonzept GmbH.
00007  * Author(s): Alexander Warg <alexander.warg@kernkonzept.com>
00008  */
00009 /*
00010  * (c) 2010 Alexander Warg <warg@os.inf.tu-dresden.de>
00011  *      economic rights: Technische Universität Dresden (Germany)
00012  *
00013  * License: see LICENSE.spdx (in this directory or the directories above)
00014  */
00015 #pragma once
00016 #pragma GCC system_header
00017
00018 #include "cxx/types"
00019 #include "cxx/ipc_basics"
00020 #include "cxx/capability.h"
```

```

00021
00022 #if defined(__GXX_RTTI) && !defined(L4_NO_RTTI)
00023 # include <typeinfo>
00024 typedef std::type_info const *L4_std_type_info_ptr;
00025 # define L4_KOBJECT_META_RTTI(type) (&typeid(type))
00026 inline char const *L4_kobject_type_name(L4_std_type_info_ptr n) noexcept
00027 { return n ? n->name() : 0; }
00028 #else
00029 typedef void const *L4_std_type_info_ptr;
00030 # define L4_KOBJECT_META_RTTI(type) (0)
00031 inline char const *L4_kobject_type_name(L4_std_type_info_ptr) noexcept
00032 { return 0; }
00033 #endif
00034
00035 namespace L4 {
00036 typedef int Opcode;
00037 // internal max helpers
00038 namespace __I {
00039 // internal max of A and B helper
00040 template< unsigned char A, unsigned char B>
00041 struct Max { enum { Res = A > B ? A : B }; };
00042 } // namespace __I
00043
00044 enum : l4_proto_t
00045 {
00047     PROTO_ANY = 0,
00049     PROTO_EMPTY = -19,
00050 };
00051
00052 namespace Typeid {
00053     using namespace L4::Types;
00054
00055     /*****/
00056     template<l4_proto_t P, typename T>
00057     struct Iface
00058     {
00059         typedef Iface type;
00060         typedef T iface_type;
00061         enum { Proto = P };
00062     };
00063
00064     /*****/
00065     struct Iface_list_end
00066     {
00067         typedef Iface_list_end type;
00068         static bool contains(l4_proto_t) noexcept { return false; }
00069     };
00070
00071     template<typename I, typename N = Iface_list_end>
00072     struct Iface_list
00073     {
00074         typedef Iface_list<I, N> type;
00075
00076         typedef typename I::iface_type iface_type;
00077         typedef N Next;
00078
00079         enum { Proto = I::Proto };
00080
00081         static bool contains(l4_proto_t proto) noexcept
00082         { return (proto == Proto) || Next::contains(proto); }
00083     };
00084
00085     // do not insert PROTO_EMPTY interfaces
00086     template<typename I, typename N>
00087     struct Iface_list<Iface<PROTO_EMPTY, I>, N> : N {};
00088
00089     // do not insert 'void' type interfaces
00090     template<l4_proto_t P, typename N>
00091     struct Iface_list<Iface<P, void>, N> : N {};
00092
00093     /*****/
00094     /*
00095     * \internal
00096     * Test if an interface I is in list L
00097     * \tparam I Interface for lookup
00098     * \tparam L Iface_list for search
00099     */
00100     template< typename I, typename L >
00101     struct _In_list;
00102
00103     template< typename I >
00104     struct _In_list<I, Iface_list_end> : False {};

```

```

00146
00147 template< typename I, typename N >
00148 struct _In_list<I, Iface_list<I, N> > : True {};
00149
00150 template< typename I, typename I2, typename N >
00151 struct _In_list<I, Iface_list<I2, N> > : _In_list<I, typename N::type> {};
00152
00153 template<typename I, typename L>
00154 struct In_list : _In_list<typename I::type, typename L::type> {};
00155
00156
00157 /*****/
00158 /*
00159  * \internal
00160  * Add Helper: add I to interface list L if ADD is true
00161  * \ingroup l4_cxx_ipc_internal
00162  */
00163 template< bool ADD, typename I, typename L>
00164 struct _Iface_list_add;
00165
00166 template< typename I, typename L>
00167 struct _Iface_list_add<false, I, L> : L {};
00168
00169 template< typename I, typename L>
00170 struct _Iface_list_add<true, I, L> : Iface_list<I, L> {};
00171
00172 /*
00173  * \internal
00174  * Add Helper: add I to interface list L if not already in L.
00175  * \ingroup l4_cxx_ipc_internal
00176  */
00177 template< typename I, typename L >
00178 struct Iface_list_add :
00179     _Iface_list_add<
00180         !In_list<I, typename L::type>::value, I, typename L::type>
00181     > {};
00182
00183 /*****/
00184 /*
00185  * \internal
00186  * Helper: checking for a conflict between I2 and I2.
00187  * A conflict means I1 and I2 have the same protocol ID but a different
00188  * iface_type.
00189  */
00190 template< typename I1, typename I2 >
00191 struct __Iface_conflict : Bool<I1::Proto != PROTO_EMPTY && I1::Proto == I2::Proto> {};
00192
00193 template< typename I >
00194 struct __Iface_conflict<I, I> : False {};
00195
00196 /*
00197  * \internal
00198  * Helper: checking for a conflict between I and any interface in LIST.
00199  */
00200 template< typename I, typename LIST >
00201 struct _Iface_conflict;
00202
00203 template< typename I >
00204 struct _Iface_conflict<I, Iface_list_end> : False {};
00205
00206 template< typename I, typename I2, typename LIST >
00207 struct _Iface_conflict<I, Iface_list<I2, LIST> > :
00208     Bool<__Iface_conflict<I, I2>::value || _Iface_conflict<I, typename LIST::type>::value>
00209     > {};
00210
00211 template< typename I, typename LIST >
00212 struct Iface_conflict : _Iface_conflict<typename I::type, typename LIST::type> {};
00213
00214 /*****/
00215 /*
00216  * \internal
00217  * Helper: merge two interface lists
00218  */
00219 template< typename L1, typename L2 >
00220 struct _Merge_list;
00221
00222 template< typename L >
00223 struct _Merge_list<Iface_list_end, L> : L {};
00224
00225 template< typename I, typename L1, typename L2 >
00226 struct _Merge_list<Iface_list<I, L1>, L2> :
00227     _Merge_list<typename L1::type, typename Iface_list_add<I, L2>::type> {};
00228
00229 template<typename L1, typename L2>
00230 struct Merge_list : _Merge_list<typename L1::type, typename L2::type> {};
00231
00232 /*****/

```

```

00237  /*
00238  * \internal
00239  * check for conflicts among all interfaces in L1 with any interfaces in L2.
00240  */
00241  template< typename L1, typename L2 >
00242  struct _Conflict;
00243
00244  template< typename L >
00245  struct _Conflict<Iface_list_end, L> : False {};
00246
00247  template< typename I, typename L1, typename L2 >
00248  struct _Conflict<Iface_list<I, L1>, L2> :
00249      Bool<Iface_conflict<I, typename L2::type>::value
00250          || _Conflict<typename L1::type, typename L2::type>::value> {};
00251
00252  template< typename L1, typename L2 >
00253  struct Conflict : _Conflict<typename L1::type, typename L2::type> {};
00254
00255  // to be removed -----
00256  // p_dispatch code -- for legacy dispatch -----
00257  /***** */
00258  /*
00259  * \internal
00260  * helper: Dispatch helper for calling server-side p_dispatch() functions.
00261  */
00262  template<typename LIST>
00263  struct _P_dispatch;
00264
00265  // No matching dispatcher found
00266  template<>
00267  struct _P_dispatch<Iface_list_end>
00268  {
00269      template< typename THIS, typename A1, typename A2 >
00270      static int f(THIS *, l4_proto_t, A1, A2 &) noexcept
00271      { return -L4_EBADPROTO; }
00272  };
00273
00274
00275  // call matching p_dispatch() function
00276  template< typename I, typename LIST >
00277  struct _P_dispatch<Iface_list<I, LIST> >
00278  {
00279      // special handling for the meta protocol, to avoid 'using' murx
00280      template< typename THIS, typename A1, typename A2 >
00281      static int _f(THIS self, A1, A2 &a2, True::type)
00282      {
00283          return self->dispatch_meta_request(a2);
00284      }
00285
00286      // normal p_dispatch() dispatching
00287      template< typename THIS, typename A1, typename A2 >
00288      static int _f(THIS self, A1 a1, A2 &a2, False::type)
00289      {
00290          return self->p_dispatch(reinterpret_cast<typename I::iface_type *>(0),
00291                                  a1, a2);
00292      }
00293
00294      // dispatch function with switch for meta protocol
00295      template< typename THIS, typename A1, typename A2 >
00296      static int f(THIS *self, l4_proto_t proto, A1 a1, A2 &a2)
00297      {
00298          if (I::Proto == proto)
00299              return _f(self, a1, a2,
00300                      Bool<I::Proto == static_cast<l4_proto_t>(L4_PROTO_META)>());
00301
00302          return _P_dispatch<typename LIST::type>::f(self, proto, a1, a2);
00303      }
00304  };
00305
00306  template<typename LIST>
00307  struct P_dispatch : _P_dispatch<typename LIST::type> {};
00308  // end: p_dispatch -----
00309  // end: to be removed -----
00310
00311  template<typename RPC> struct Default_op;
00312
00313  namespace Detail {
00314
00315  00317  struct Rpcs_end
00316  {
00317      typedef void opcode_type;
00318      typedef Rpcs_end rpc;
00319      typedef Rpcs_end type;
00320  };
00321
00322  template<typename O1, typename O2, typename RPCS>
00323  struct _Rpc : _Rpc<typename RPCS::next::rpc, O2, typename RPCS::next>::type {};

```

```

00328
00329 template<typename O1, typename O2>
00330 struct _Rpc<O1, O2, Rpc_end> {};
00331
00332 template<typename OP, typename RPCS>
00333 struct _Rpc<OP, OP, RPCS> : RPCS
00334 {
00335     typedef _Rpc type;
00336 };
00337
00338 template<typename OP, typename RPCS>
00339 struct Rpc : _Rpc<typename RPCS::rpc, OP, RPCS> {};
00340
00341 template<typename T, unsigned CODE>
00342 struct _Get_opcode
00343 {
00344     template<bool, typename> struct Invalid_opcode {};
00345     template<typename X> struct Invalid_opcode<true, X>;
00346
00347 private:
00348     template<typename U, U> struct _chk;
00349     template<typename U> static l4_proto_t _opc(_chk<int, U::Opcode> *);
00350     template<typename U> static char _opc(...);
00351
00352     template<unsigned SZ, typename U>
00353     struct _Opc { enum { value = CODE }; };
00354
00355     template<typename U>
00356     struct _Opc<sizeof(l4_proto_t), U> { enum { value = U::Opcode }; };
00357
00358 public:
00359     enum { value = _Opc<sizeof(_opc<T>())>, T>::value };
00360     Invalid_opcode<(value < CODE), T> invalid_opcode;
00361 };
00362
00363 template<typename OPCODE, unsigned O, typename ...X>
00364 struct _Rpc : Rpc_end {};
00365
00366 template<typename OPCODE, unsigned O, typename R, typename ...X>
00367 struct _Rpc<OPCODE, O, R, X...>
00368 {
00369     typedef _Rpc type;
00370     typedef OPCODE opcode_type;
00371     typedef R rpc;
00372     typedef typename _Rpc<OPCODE, _Get_opcode<R, O>::value + 1, X...>::type next;
00373     enum { Opcode = _Get_opcode<R, O>::value };
00374     template<typename Y> struct Rpc : Typeid::Detail::Rpc<Y, _Rpc> {};
00375 };
00376
00377 template<typename OPCODE, unsigned O, typename R>
00378 struct _Rpc<OPCODE, O, Default_op<R> >
00379 {
00380     typedef _Rpc type;
00381     typedef void opcode_type;
00382     typedef R rpc;
00383     typedef Rpc_end next;
00384     enum { Opcode = -99 };
00385     template<typename Y> struct Rpc : Typeid::Detail::Rpc<Y, _Rpc> {};
00386 };
00387
00388 } // namespace Detail
00389
00390 template<typename CLASS>
00391 struct Raw_ipc
00392 {
00393     typedef Raw_ipc type;
00394     typedef Detail::Rpc_end next;
00395     typedef void opcode_type;
00396 };
00397
00398 template<typename ...RPCS>
00399 struct Rpc : Detail::_Rpc<L4::Opcode, 0, RPCS...> {};
00400
00401 template<typename OPCODE_TYPE>
00402 struct Rpc_code
00403 {
00404     template<typename ...RPCS>
00405     struct F : Detail::_Rpc<OPCODE_TYPE, 0, RPCS...> {};
00406 };
00407
00408 template<typename OPERATION>
00409 struct Rpc_nocode : Detail::_Rpc<void, 0, OPERATION> {};
00410
00411 template<typename ...ARG>
00412 struct Rpc_sys : Detail::_Rpc<l4_umword_t, 0, ARG...> {};
00413
00414 template<typename CLASS>

```

```

00468 struct Rights
00469 {
00470     unsigned rights;
00471     Rights(unsigned rights) noexcept : rights(rights) {}
00472     unsigned operator & (unsigned rhs) const noexcept { return rights & rhs; }
00473 };
00474
00475 } // namespace Typeid
00476
00499 struct L4_EXPORT Type_info
00500 {
00506     class L4_EXPORT Demand
00507     {
00508     private:
00510         static unsigned char max(unsigned char a, unsigned char b) noexcept
00511         { return a > b ? a : b; }
00512     public:
00514         unsigned char caps;
00515         unsigned char flags;
00516         unsigned char mem;
00517         unsigned char ports;
00518
00526         explicit
00527         Demand(unsigned char caps = 0, unsigned char flags = 0,
00528                unsigned char mem = 0, unsigned char ports = 0) noexcept
00529         : caps(caps), flags(flags), mem(mem), ports(ports) {}
00530
00532         bool no_demand() const noexcept
00533         { return caps == 0 && mem == 0 && ports == 0 && flags == 0; }
00534
00536         Demand operator | (Demand const &rhs) const noexcept
00537         {
00538             return Demand(max(caps, rhs.caps), flags | rhs.flags,
00539                            max(mem, rhs.mem), max(ports, rhs.ports));
00540         }
00541     };
00542
00551     template<unsigned char CAPS = 0, unsigned char FLAGS = 0,
00552              unsigned char MEM = 0, unsigned char PORTS = 0>
00553     struct Demand_t : Demand
00554     {
00555         enum
00556         {
00557             Caps = CAPS,
00558             Flags = FLAGS,
00559             Mem = MEM,
00560             Ports = PORTS
00561         };
00562         Demand_t() noexcept : Demand(CAPS, FLAGS, MEM, PORTS) {}
00563     };
00564
00572     template<typename D1, typename D2>
00573     struct Demand_union_t : Demand_t<__I::Max<D1::Caps, D2::Caps>::Res,
00574                                       D1::Flags | D2::Flags,
00575                                       __I::Max<D1::Mem, D2::Mem>::Res,
00576                                       __I::Max<D1::Ports, D2::Ports>::Res>
00577     {};
00578
00579     L4_std_type_info_ptr _type;
00580     Type_info const *const *_bases;
00581     unsigned _num_bases;
00582     l4_proto_t _proto;
00583
00584     L4_std_type_info_ptr type() const noexcept { return _type; }
00585     Type_info const *base(unsigned idx) const noexcept { return _bases[idx]; }
00586     unsigned num_bases() const noexcept { return _num_bases; }
00587     l4_proto_t proto() const noexcept { return _proto; }
00588     char const *name() const noexcept { return L4_kobject_type_name(type()); }
00589     bool has_proto(l4_proto_t proto) const noexcept
00590     {
00591         if (_proto && _proto == proto)
00592             return true;
00593
00594         if (!proto)
00595             return false;
00596
00597         for (unsigned i = 0; i < _num_bases; ++i)
00598             if (base(i)->has_proto(proto))
00599                 return true;
00600
00601         return false;
00602     }
00603 };
00604
00610 template<typename T> struct Kobject_typeid
00611 {

```

```

00622 typedef typename T::__Kobject_typeid::Demand Demand;
00623 typedef typename T::__Iface::iface_type Iface;
00624 typedef typename T::__Iface_list Iface_list;
00625
00630 static Type_info const *id() noexcept { return &T::__Kobject_typeid::_m; }
00631
00639 static Type_info::Demand demand() noexcept
00640 { return T::__Kobject_typeid::Demand(); }
00641
00642 // to be removed -----
00643 // p_dispatch -----
00659 template<typename THIS, typename A1, typename A2>
00660 static int proto_dispatch(THIS *self, l4_proto_t proto, A1 a1, A2 &a2)
00661 { return Typeid::P_dispatch<typename T::__Iface_list>::f(self, proto, a1, a2); }
00662 // p_dispatch -----
00663 // end: to be removed -----
00664 };
00665
00667 template<> struct Kobject_typeid<void>
00668 {
00669     typedef Type_info::Demand_t<> Demand;
00670 };
00671
00680 template<typename T>
00681 inline
00682 Type_info const *kobject_typeid() noexcept
00683 { return Kobject_typeid<T>::id(); }
00684
00689 #define L4___GEN_TI(t...)
00690 Type_info const t::__Kobject_typeid::_m =
00691 {
00692     L4_KOBJECT_META_RTTI(Derived),
00693     &t::__Kobject_typeid::_b[0],
00694     sizeof(t::__Kobject_typeid::_b) / sizeof(t::__Kobject_typeid::_b[0]),
00695     PROTO
00696 }
00697
00702 #define L4___GEN_TI_MEMBERS(BASE_DEMAND...)
00703 private:
00704     template< typename T > friend struct Kobject_typeid;
00705 protected:
00706     struct __Kobject_typeid {
00707         typedef Type_info::Demand_union_t<S_DEMAND, BASE_DEMAND> Demand;
00708         static Type_info const *const _b[];
00709         static Type_info const _m;
00710     };
00711 public:
00712     static l4_proto_t const Protocol = PROTO;
00713     typedef L4::Typeid::Rights<Class> Rights;
00714
00743 template<
00744     typename Derived,
00745     typename Base,
00746     l4_proto_t PROTO = PROTO_ANY,
00747     typename S_DEMAND = Type_info::Demand_t<>
00748 >
00749 class Kobject_t : public Base
00750 {
00751 protected:
00753     typedef Derived Class;
00755     typedef Typeid::Iface<PROTO, Derived> __Iface;
00757     typedef Typeid::Merge_list<
00758         Typeid::Iface_list<__Iface>, typename Base::__Iface_list
00759     > __Iface_list;
00760
00762     static void __check_protocols__() noexcept
00763     {
00764         typedef Typeid::Iface_conflict<__Iface, typename Base::__Iface_list> Base_conflict;
00765         static_assert(!Base_conflict::value, "ambiguous protocol ID: protocol also used by Base");
00766     }
00767
00769     L4::Cap<Class> c() const noexcept { return L4::Cap<Class>(this->cap()); }
00770
00771     // Generate the remaining type information
00772     L4___GEN_TI_MEMBERS(typename Base::__Kobject_typeid::Demand)
00773 };
00774
00775
00777 template< typename Derived, typename Base, l4_proto_t PROTO, typename S_DEMAND>
00778 Type_info const *const
00779 Kobject_t<Derived, Base, PROTO, S_DEMAND>::
00780     __Kobject_typeid::_b[] = { &Base::__Kobject_typeid::_m };
00781
00782
00787 template< typename Derived, typename Base, l4_proto_t PROTO, typename S_DEMAND>
00788 L4___GEN_TI(Kobject_t<Derived, Base, PROTO, S_DEMAND>);
00789
00790

```

```

00820 template<
00821     typename Derived,
00822     typename Base1,
00823     typename Base2,
00824     l4_proto_t PROTO = PROTO_ANY,
00825     typename S_DEMAND = Type_info::Demand_t<>
00826 >
00827 class Kobject_2t : public Base1, public Base2
00828 {
00829 protected:
00831     typedef Derived Class;
00833     typedef Typeid::Iface<PROTO, Derived> __Iface;
00835     typedef Typeid::Merge_list<
00836         Typeid::Iface_list<__Iface>,
00837         Typeid::Merge_list<
00838             typename Base1::__Iface_list,
00839             typename Base2::__Iface_list
00840         >
00841     > __Iface_list;
00842
00844     static void __check_protocols__() noexcept
00845     {
00846         typedef typename Base1::__Iface_list Base1_proto_list;
00847         typedef typename Base2::__Iface_list Base2_proto_list;
00848
00849         typedef Typeid::Iface_conflict<__Iface, Base1_proto_list> Base1_conflict;
00850         typedef Typeid::Iface_conflict<__Iface, Base2_proto_list> Base2_conflict;
00851         static_assert(!Base1_conflict::value, "ambiguous protocol ID, also in Base1");
00852         static_assert(!Base2_conflict::value, "ambiguous protocol ID, also in Base2");
00853
00854         typedef Typeid::Conflict<Base1_proto_list, Base2_proto_list> Bases_conflict;
00855         static_assert(!Bases_conflict::value, "ambiguous protocol IDs in base classes");
00856     }
00857
00858     // disambiguate cap()
00859     l4_cap_idx_t cap() const noexcept
00860     { return Base1::cap(); }
00861
00863     L4::Cap<Class> c() const noexcept { return L4::Cap<Class>(this->cap()); }
00864
00865     L4__GEN_TI_MEMBERS(Type_info::Demand_union_t<
00866         typename Base1::__Kobject_typeid::Demand,
00867         typename Base2::__Kobject_typeid::Demand>
00868     )
00869
00870 public:
00871     // Provide non-ambiguous conversion to Kobject
00872     operator Kobject const & () const noexcept
00873     { return *static_cast<Base1 const *>(this); }
00874
00875     // Provide non-ambiguous access of dec_refcnt()
00876     l4_msgtag_t dec_refcnt(l4_mword_t diff, l4_utcb_t *utcb = l4_utcb())
00877     noexcept(noexcept(static_cast<Base1*>(nullptr)->dec_refcnt(diff, utcb)))
00878     { return Base1::dec_refcnt(diff, utcb); }
00879 };
00880
00881
00883 template< typename Derived, typename Base1, typename Base2,
00884     l4_proto_t PROTO, typename S_DEMAND >
00885 Type_info const *const
00886 Kobject_2t<Derived, Base1, Base2, PROTO, S_DEMAND>::__Kobject_typeid::b[] =
00887 {
00888     &Base1::__Kobject_typeid::_m,
00889     &Base2::__Kobject_typeid::_m
00890 };
00892
00897 template< typename Derived, typename Base1, typename Base2,
00898     l4_proto_t PROTO, typename S_DEMAND >
00899 L4__GEN_TI(Kobject_2t<Derived, Base1, Base2, PROTO, S_DEMAND>);
00900
00901
00902
00922 template<
00923     typename Derived,
00924     typename Base1,
00925     typename Base2,
00926     typename Base3,
00927     l4_proto_t PROTO = PROTO_ANY,
00928     typename S_DEMAND = Type_info::Demand_t<>
00929 >
00930 struct Kobject_3t : Base1, Base2, Base3
00931 {
00932 protected:
00934     typedef Derived Class;
00936     typedef Typeid::Iface<PROTO, Derived> __Iface;
00938     typedef Typeid::Merge_list<
00939         Typeid::Iface_list<__Iface>,

```



```

00940     typeid::Merge_list<
00941         typename Base1::__Iface_list,
00942         typeid::Merge_list<
00943             typename Base2::__Iface_list,
00944             typename Base3::__Iface_list
00945         >
00946     >
00947 > __Iface_list;
00948
00950 static void __check_protocols__() noexcept
00951 {
00952     typedef typename Base1::__Iface_list Base1_proto_list;
00953     typedef typename Base2::__Iface_list Base2_proto_list;
00954     typedef typename Base3::__Iface_list Base3_proto_list;
00955
00956     typedef typeid::Iface_conflict<__Iface, Base1_proto_list> Base1_conflict;
00957     typedef typeid::Iface_conflict<__Iface, Base2_proto_list> Base2_conflict;
00958     typedef typeid::Iface_conflict<__Iface, Base3_proto_list> Base3_conflict;
00959
00960     static_assert(!Base1_conflict::value, "ambiguous protocol ID, also in Base1");
00961     static_assert(!Base2_conflict::value, "ambiguous protocol ID, also in Base2");
00962     static_assert(!Base3_conflict::value, "ambiguous protocol ID, also in Base3");
00963
00964     typedef typeid::Conflict<Base1_proto_list, Base2_proto_list> Conflict_bases12;
00965     typedef typeid::Conflict<Base1_proto_list, Base3_proto_list> Conflict_bases13;
00966     typedef typeid::Conflict<Base2_proto_list, Base3_proto_list> Conflict_bases23;
00967
00968     static_assert(!Conflict_bases12::value, "ambiguous protocol IDs in base classes: Base1 and
Base2");
00969     static_assert(!Conflict_bases13::value, "ambiguous protocol IDs in base classes: Base1 and
Base3");
00970     static_assert(!Conflict_bases23::value, "ambiguous protocol IDs in base classes: Base2 and
Base3");
00971 }
00972
00973 // disambiguate cap()
00974 l4_cap_idx_t cap() const noexcept
00975 { return Base1::cap(); }
00976
00978 L4::Cap<Class> c() const noexcept { return L4::Cap<Class>(this->cap()); }
00979
00980 L4__GEN_TI_MEMBERS(Type_info::Demand_union_t<Type_info::Demand_union_t<
00981     typename Base1::__Kobject_typeid::Demand,
00982     typename Base2::__Kobject_typeid::Demand>,
00983     typename Base3::__Kobject_typeid::Demand>
00984 )
00985
00986 public:
00987     // Provide non-ambiguous conversion to Kobject
00988     operator Kobject const & () const noexcept
00989     { return *static_cast<Base1 const *>(this); }
00990
00991     // Provide non-ambiguous access of dec_refcnt()
00992     l4_msgtag_t dec_refcnt(l4_mword_t diff, l4_utcb_t *utcb = l4_utcb())
00993     noexcept(noexcept(static_cast<Base1*>(nullptr)->dec_refcnt(diff, utcb)))
00994     { return Base1::dec_refcnt(diff, utcb); }
00995 };
00996
00997
00999 template< typename Derived, typename Base1, typename Base2, typename Base3,
10000     l4_proto_t PROTO, typename S_DEMAND >
10001 Type_info const *const
10002 Kobject_3t<Derived, Base1, Base2, Base3, PROTO, S_DEMAND>::__Kobject_typeid::_b[] =
10003 {
10004     &Base1::__Kobject_typeid::_m,
10005     &Base2::__Kobject_typeid::_m,
10006     &Base3::__Kobject_typeid::_m
10007 };
10008
10009
10014 template< typename Derived, typename Base1, typename Base2, typename Base3,
10015     l4_proto_t PROTO, typename S_DEMAND >
10016 L4__GEN_TI(Kobject_3t<Derived, Base1, Base2, Base3, PROTO, S_DEMAND>);
10017
10018 }
10019
10020 #if __cplusplus >= 201103L
10021
10022 namespace L4 {
10023
10030 template< typename ...T >
10031 struct Kobject_demand;
10032
10033 template<>
10034 struct Kobject_demand<> : Type_info::Demand_t<> {};
10035
10036 template<typename T>
10037 struct Kobject_demand<T> : Kobject_typeid<T>::Demand {};

```

```

01038
01039 template<typename T1, typename ...T2>
01040 struct Kobject_demand<T1, T2...> :
01041     Type_info::Demand_union_t<typename Kobject_typeid<T1>::Demand,
01042         Kobject_demand<T2...> >
01043 {};
01044
01045 namespace Typeid_xx {
01046
01047     template<typename ...LISTS>
01048     struct Merge_list;
01049
01050     template<typename L>
01051     struct Merge_list<L> : L {};
01052
01053     template<typename L1, typename L2>
01054     struct Merge_list<L1, L2> : Typeid::Merge_list<L1, L2> {};
01055
01056     template<typename L1, typename L2, typename ...LISTS>
01057     struct Merge_list<L1, L2, LISTS...> :
01058         Merge_list<typename Typeid::Merge_list<L1, L2>::type, LISTS...> {};
01059
01060     template< typename I, typename ...LIST >
01061     struct Iface_conflict;
01062
01063     template< typename I >
01064     struct Iface_conflict<I> : Typeid::False {};
01065
01066     template< typename I, typename L, typename ...LIST >
01067     struct Iface_conflict<I, L, LIST...> :
01068         Typeid::Bool<Typeid::Iface_conflict<typename I::type, typename L::type>::value
01069             || Iface_conflict<I, LIST...>::value>
01070     {};
01071
01072     template< typename ...LIST >
01073     struct Conflict;
01074
01075     template< typename L >
01076     struct Conflict<L> : Typeid::False {};
01077
01078     template< typename L1, typename L2, typename ...LIST >
01079     struct Conflict<L1, L2, LIST...> :
01080         Typeid::Bool<Typeid::Conflict<typename L1::type, typename L2::type>::value
01081             || Conflict<L1, LIST...>::value
01082             || Conflict<L2, LIST...>::value>
01083     {};
01084
01085     template< typename T >
01086     struct Is_demand
01087     {
01088         static l4_proto_t test(Type_info::Demand const *);
01089         static char test(...);
01090         enum { value = sizeof(test(static_cast<T*>(nullptr))) == sizeof(l4_proto_t) };
01091     };
01092
01093     template< typename T, typename ... >
01094     struct First : T { typedef T type; };
01095 } // Typeid
01096
01102 template< typename Derived, l4_proto_t PROTO, typename S_DEMAND, typename ...BASES>
01103 struct __Kobject_base : BASES...
01104 {
01105     protected:
01106         typedef Derived Class;
01107         typedef Typeid::Iface<PROTO, Derived> __Iface;
01108         typedef Typeid_xx::Merge_list<
01109             Typeid::Iface_list<__Iface>,
01110             typename BASES::__Iface_list...
01111         > __Iface_list;
01112
01113         static void __check_protocols__() noexcept
01114         {
01115             typedef Typeid_xx::Iface_conflict<__Iface, typename BASES::__Iface_list...> Conflict;
01116             static_assert(!Conflict::value, "ambiguous protocol ID, protocol also used in base class");
01117
01118             typedef Typeid_xx::Conflict<typename BASES::__Iface_list...> Base_conflict;
01119             static_assert(!Base_conflict::value, "ambiguous protocol IDs in base classes");
01120         }
01121
01122         // disambiguate cap()
01123         l4_cap_idx_t cap() const noexcept
01124         { return Typeid_xx::First<BASES...>::type::cap(); }
01125
01126         L4::Cap<Class> c() const noexcept { return L4::Cap<Class>(this->cap()); }
01127
01128         L4____GEN_TI_MEMBERS(Kobject_demand<BASES...>)
01129

```

```

01130 private:
01131     // This function returns the first base class (used below)
01132     template<typename B1, typename ...> struct Basel { typedef B1 type; };
01133
01134 public:
01135     // Provide non-ambiguous conversion to Kobject
01136     operator Kobject const & () const noexcept
01137     { return *static_cast<typename Basel<BASES...>::type const *>(this); }
01138
01139     // Provide non-ambiguous access of dec_refcnt()
01140     l4_msgtag_t dec_refcnt(l4_mword_t diff, l4_utcb_t *utcb = l4_utcb())
01141     noexcept(noexcept(static_cast<typename Basel<BASES...>::type *>(nullptr)
01142         ->dec_refcnt(diff, utcb)))
01143     { return Basel<BASES...>::type::dec_refcnt(diff, utcb); }
01144 };
01145
01146 template< typename Derived, l4_proto_t PROTO, typename S_DEMAND, typename ...BASES>
01147 Type_info const *const
01148 __Kobject_base<Derived, PROTO, S_DEMAND, BASES...>::__Kobject_typeid::_b[] =
01149 {
01150     (&BASES::__Kobject_typeid::_m)...
01151 };
01152
01153 template< typename Derived, l4_proto_t PROTO, typename S_DEMAND, typename ...BASES>
01154 L4___GEN_TI(__Kobject_base<Derived, PROTO, S_DEMAND, BASES...>);
01155
01156 // Test if the there is a Demand argument to Kobject_x
01157 template< typename Derived, l4_proto_t PROTO, bool HAS_DEMAND, typename DEMAND, typename ...ARGS >
01158 struct __Kobject_x_proto;
01159
01160 // YES: pass it to __Kobject_base
01161 template< typename Derived, l4_proto_t PROTO, typename DEMAND, typename ...BASES>
01162 struct __Kobject_x_proto<Derived, PROTO, true, DEMAND, BASES...> :
01163     __Kobject_base<Derived, PROTO, DEMAND, BASES...> {};
01164
01165 // NO: pass it empty Type_info::Demand_t
01166 template< typename Derived, l4_proto_t PROTO, typename B1, typename ...BASES>
01167 struct __Kobject_x_proto<Derived, PROTO, false, B1, BASES...> :
01168     __Kobject_base<Derived, PROTO, Type_info::Demand_t<>, B1, BASES...> {};
01169
01170 template< l4_proto_t P = PROTO_EMPTY >
01171 struct Proto_t {};
01172
01173 template< typename Derived, typename ...ARGS >
01174 struct Kobject_x;
01175
01176 template< typename Derived, typename A, typename ...ARGS >
01177 struct Kobject_x<Derived, A, ARGS...> :
01178     __Kobject_x_proto<Derived, PROTO_ANY, Typeid_xx::Is_demand<A>::value, A, ARGS...>
01179 {};
01180
01181 template< typename Derived, l4_proto_t PROTO, typename A, typename ...ARGS >
01182 struct Kobject_x<Derived, Proto_t<PROTO>, A, ARGS...> :
01183     __Kobject_x_proto<Derived, PROTO, Typeid_xx::Is_demand<A>::value, A, ARGS...>
01184 {};
01185
01186 #endif
01187
01188 #undef L4___GEN_TI
01189 #undef L4___GEN_TI_MEMBERS

```

16.443 __vcpu-arm.h

```

00001 /*
00002  * (c) 2017 Alexander Warg <alexander.warg@kernkonzept.com>
00003  *
00004  * License: see LICENSE.spdx (in this directory or the directories above)
00005  */
00006 #pragma once
00007
00008 typedef struct l4_arm_vcpu_e_info_t
00009 {
00010     l4_uint8_t version; // must be 0
00011     l4_uint8_t gic_version;
00012     l4_uint8_t _rsvd0[2];
00013     l4_uint32_t features;
00014     l4_uint32_t _rsvd1[14];
00015     l4_umword_t user[8];
00016 } l4_arm_vcpu_e_info_t;
00017
00018 L4_INLINE void *l4_vcpu_e_ptr(void const *vcpu, unsigned id) L4_NOTHROW;

```

```

00019
00020 enum L4_vcpu_e_consts
00021 {
00022     L4_VCPU_E_NUM_LR = 4,
00023 };
00024
00025 L4_INLINE l4_arm_vcpu_e_info_t const *
00026 l4_vcpu_e_info(void const *vcpu) L4_NOTHROW;
00027
00028 L4_INLINE l4_umword_t *
00029 l4_vcpu_e_info_user(void *vcpu) L4_NOTHROW;
00030
00031 L4_INLINE l4_umword_t *
00032 l4_vcpu_e_info_user(void *vcpu) L4_NOTHROW
00033 {
00034     return ((l4_arm_vcpu_e_info_t *)l4_vcpu_e_info(vcpu))->user;
00035 }
00036
00037
00045 L4_INLINE l4_uint32_t
00046 l4_vcpu_e_read_32(void const *vcpu, unsigned id) L4_NOTHROW;
00047
00048 L4_INLINE l4_uint32_t
00049 l4_vcpu_e_read_32(void const *vcpu, unsigned id) L4_NOTHROW
00050 { return *(l4_uint32_t const *)l4_vcpu_e_ptr(vcpu, id); }
00051
00059 L4_INLINE void
00060 l4_vcpu_e_write_32(void *vcpu, unsigned id, l4_uint32_t val) L4_NOTHROW;
00061
00062 L4_INLINE void
00063 l4_vcpu_e_write_32(void *vcpu, unsigned id, l4_uint32_t val) L4_NOTHROW
00064 { *((l4_uint32_t *)l4_vcpu_e_ptr(vcpu, + id)) = val; }
00065
00073 L4_INLINE l4_uint64_t
00074 l4_vcpu_e_read_64(void const *vcpu, unsigned id) L4_NOTHROW;
00075
00076 L4_INLINE l4_uint64_t
00077 l4_vcpu_e_read_64(void const *vcpu, unsigned id) L4_NOTHROW
00078 { return *(l4_uint64_t const *)l4_vcpu_e_ptr(vcpu, id); }
00079
00087 L4_INLINE void
00088 l4_vcpu_e_write_64(void *vcpu, unsigned id, l4_uint64_t val) L4_NOTHROW;
00089
00090 L4_INLINE void
00091 l4_vcpu_e_write_64(void *vcpu, unsigned id, l4_uint64_t val) L4_NOTHROW
00092 { *((l4_uint64_t *)l4_vcpu_e_ptr(vcpu, id)) = val; }
00093
00101 L4_INLINE l4_umword_t
00102 l4_vcpu_e_read(void const *vcpu, unsigned id) L4_NOTHROW;
00103
00104 L4_INLINE l4_umword_t
00105 l4_vcpu_e_read(void const *vcpu, unsigned id) L4_NOTHROW
00106 { return *(l4_umword_t const *)l4_vcpu_e_ptr(vcpu, id); }
00107
00115 L4_INLINE void
00116 l4_vcpu_e_write(void *vcpu, unsigned id, l4_umword_t val) L4_NOTHROW;
00117
00118 L4_INLINE void
00119 l4_vcpu_e_write(void *vcpu, unsigned id, l4_umword_t val) L4_NOTHROW
00120 { *((l4_umword_t *)l4_vcpu_e_ptr(vcpu, id)) = val; }

```

16.444 __vm-svm.h

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013 #pragma once
00014
00015 #include <l4/sys/types.h>
00016
00022
00023
00028 typedef struct l4_vm_svm_vmcb_control_area
00029 {
00030     l4_uint16_t intercept_rd_crX;
00031     l4_uint16_t intercept_wr_crX;
00032
00033     l4_uint16_t intercept_rd_drX;

```

```

00034 14_uint16_t intercept_wr_drX;
00035
00036 14_uint32_t intercept_exceptions;
00037
00038 14_uint32_t intercept_instruction0;
00039 14_uint32_t intercept_instruction1;
00040
00041 14_uint8_t _reserved0[40];
00042
00043 14_uint16_t pause_filter_threshold;
00044 14_uint16_t pause_filter_count;
00045
00046 14_uint64_t iopm_base_pa;
00047 14_uint64_t msrpm_base_pa;
00048 14_uint64_t tsc_offset;
00049 14_uint64_t guest_asid_tlb_ctl;
00050 14_uint64_t interrupt_ctl;
00051 14_uint64_t interrupt_shadow;
00052 14_uint64_t exitcode;
00053 14_uint64_t exitinfo1;
00054 14_uint64_t exitinfo2;
00055 14_uint64_t exitintinfo;
00056 14_uint64_t np_enable;
00057
00058 14_uint8_t _reserved1[16];
00059
00060 14_uint64_t eventinj;
00061 14_uint64_t n_cr3;
00062 14_uint64_t lbr_virtualization_enable;
00063 14_uint64_t clean_bits;
00064 14_uint64_t n_rip;
00065
00066 14_uint8_t _reserved2[816];
00067 } __attribute__((packed)) 14_vm_svm_vmcb_control_area_t;
00068
00073 typedef struct 14_vm_svm_vmcb_state_save_area_seg
00074 {
00075     14_uint16_t selector;
00076     14_uint16_t attrib;
00077     14_uint32_t limit;
00078     14_uint64_t base;
00079 } __attribute__((packed)) 14_vm_svm_vmcb_state_save_area_seg_t;
00080
00085 typedef struct 14_vm_svm_vmcb_state_save_area
00086 {
00087     struct 14_vm_svm_vmcb_state_save_area_seg es;
00088     struct 14_vm_svm_vmcb_state_save_area_seg cs;
00089     struct 14_vm_svm_vmcb_state_save_area_seg ss;
00090     struct 14_vm_svm_vmcb_state_save_area_seg ds;
00091     struct 14_vm_svm_vmcb_state_save_area_seg fs;
00092     struct 14_vm_svm_vmcb_state_save_area_seg gs;
00093     struct 14_vm_svm_vmcb_state_save_area_seg gdtr;
00094     struct 14_vm_svm_vmcb_state_save_area_seg ldtr;
00095     struct 14_vm_svm_vmcb_state_save_area_seg idtr;
00096     struct 14_vm_svm_vmcb_state_save_area_seg tr;
00097
00098     14_uint8_t _reserved0[43];
00099
00100     14_uint8_t cpl;
00101
00102     14_uint32_t _reserved1;
00103
00104     14_uint64_t efer;
00105
00106     14_uint8_t _reserved2[112];
00107
00108     14_uint64_t cr4;
00109     14_uint64_t cr3;
00110     14_uint64_t cr0;
00111     14_uint64_t dr7;
00112     14_uint64_t dr6;
00113     14_uint64_t rflags;
00114     14_uint64_t rip;
00115
00116     14_uint8_t _reserved3[88];
00117
00118     14_uint64_t rsp;
00119
00120     14_uint8_t _reserved4[24];
00121
00122     14_uint64_t rax;
00123     14_uint64_t star;
00124     14_uint64_t lstar;
00125     14_uint64_t cstar;
00126     14_uint64_t sfmask;
00127     14_uint64_t kernelgsbase;
00128     14_uint64_t sysenter_cs;

```

```

00129  l4_uint64_t sysenter_esp;
00130  l4_uint64_t sysenter_eip;
00131  l4_uint64_t cr2;
00132
00133  l4_uint8_t _reserved5[32];
00134
00135  l4_uint64_t g_pat;
00136  l4_uint64_t dbgctl;
00137  l4_uint64_t br_from;
00138  l4_uint64_t br_to;
00139  l4_uint64_t lastexcpfrom;
00140  l4_uint64_t last_excpto;
00141
00142  // this field is _NOT_ part of the official VMCB specification
00143  // a (userlevel) VMM needs this for proper FPU state virtualization
00144  l4_uint64_t xcr0;
00145
00146  l4_uint8_t _reserved6[2400];
00147 } __attribute__((packed)) l4_vm_svm_vmc_b_state_save_area_t;
00148
00149
00154 typedef struct l4_vm_svm_vmc_b_t
00155 {
00156     l4_vm_svm_vmc_b_control_area_t control_area;
00157     l4_vm_svm_vmc_b_state_save_area_t state_save_area;
00158 } l4_vm_svm_vmc_b_t;

```

16.445 __vm-vmx.h

```

00001
00006 /*
00007  * (c) 2010-2013 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013
00014 #pragma once
00015
00016 #include <l4/sys/vcpu.h>
00017
00023
00028 enum l4_vm_vmx_caps_regs
00029 {
00030     L4_VM_VMX_BASIC_REG = 0,
00031     L4_VM_VMX_TRUE_PINBASED_CTLs_REG = 1,
00032     L4_VM_VMX_TRUE_PROCBASED_CTLs_REG = 2,
00033     L4_VM_VMX_TRUE_EXIT_CTLs_REG = 3,
00034     L4_VM_VMX_TRUE_ENTRY_CTLs_REG = 4,
00035     L4_VM_VMX_MISC_REG = 5,
00036     L4_VM_VMX_CR0_FIXED0_REG = 6,
00037     L4_VM_VMX_CR0_FIXED1_REG = 7,
00038     L4_VM_VMX_CR4_FIXED0_REG = 8,
00039     L4_VM_VMX_CR4_FIXED1_REG = 9,
00040     L4_VM_VMX_VMCS_ENUM_REG = 10,
00041     L4_VM_VMX_PROCBASED_CTLs2_REG = 11,
00042     L4_VM_VMX_EPT_VPID_CAP_REG = 12,
00043     L4_VM_VMX_NESTED_REVISION = 13,
00044     L4_VM_VMX_NUM_CAPS_REGS
00045 };
00046
00051 enum l4_vm_vmx_dfl1_regs
00052 {
00053     L4_VM_VMX_PINBASED_CTLs_DFL1_REG = 0,
00054     L4_VM_VMX_PROCBASED_CTLs_DFL1_REG = 1,
00055     L4_VM_VMX_EXIT_CTLs_DFL1_REG = 2,
00056     L4_VM_VMX_ENTRY_CTLs_DFL1_REG = 3,
00057     L4_VM_VMX_NUM_DFL1_REGS
00058 };
00059
00069 enum l4_vm_vmx_sw_fields
00070 {
00078     L4_VM_VMX_VMCS_CR2 = 0x6880,
00080     L4_VM_VMX_VMCS_NAT_ARG0 = 0x6882,
00082     L4_VM_VMX_VMCS_NAT_ARG1 = 0x6884,
00084     L4_VM_VMX_VMCS_NAT_ARG2 = 0x6886,
00086     L4_VM_VMX_VMCS_NAT_ARG3 = 0x6888,
00088     L4_VM_VMX_VMCS_XCR0 = 0x2880,
00090     L4_VM_VMX_VMCS_MSR_SYSCALL_MASK = 0x2882,
00092     L4_VM_VMX_VMCS_MSR_LSTAR = 0x2884,
00094     L4_VM_VMX_VMCS_MSR_CSTAR = 0x2886,
00096     L4_VM_VMX_VMCS_MSR_TSC_AUX = 0x2888,

```

```

00098  L4_VM_VMX_VMCS_MSR_STAR          = 0x288a,
00100  L4_VM_VMX_VMCS_MSR_KERNEL_GS_BASE = 0x288c,
00101  };
00102
00155  typedef struct l4_vmx_offset_table_t
00156  {
00157      l4_uint8_t  offsets[4][4];
00158      l4_uint8_t  limits[4][4];
00159      l4_uint8_t  index_shifts[4];
00160      l4_uint8_t  base_offset;
00161      l4_uint8_t  size;
00162
00163      l4_uint8_t  reserved[2];
00164  } l4_vmx_offset_table_t;
00165
00170  enum L4_vm_vmx_vmcs_sizes
00171  {
00173      L4_VM_VMX_VMCS_SIZE_VALUES = 2560,
00175      L4_VM_VMX_VMCS_SIZE_DIRTY_BITMAP = 320,
00176  };
00177
00205  typedef struct l4_vm_vmx_vcpu_vmcs_t
00206  {
00207      l4_uint64_t reserved0;
00208
00209      l4_uint64_t user_data;
00210      l4_uint32_t cr2_index;
00211      l4_uint8_t reserved1[4];
00212
00213      l4_cap_idx_t vmcs;
00214
00215      /*
00216       * Since the capability type size depends on the platform, we add a 32-bit
00217       * padding if necessary.
00218       */
00219
00220      #if L4_MWORD_BITS == 32
00221          l4_uint32_t padding0;
00222      #elif L4_MWORD_BITS == 64
00223          /* No padding needed. */
00224      #else
00225          #error Unsupported machine word size.
00226      #endif
00227
00228      l4_vmx_offset_table_t offset_table;
00229      l4_uint8_t reserved2[120];
00230
00231      l4_uint8_t values[L4_VM_VMX_VMCS_SIZE_VALUES];
00232      l4_uint8_t dirty_bitmap[L4_VM_VMX_VMCS_SIZE_DIRTY_BITMAP];
00233  } l4_vm_vmx_vcpu_vmcs_t;
00234
00239  typedef struct l4_vm_vmx_vcpu_infos_t
00240  {
00242      l4_uint64_t caps[L4_VM_VMX_NUM_CAPS_REGS];
00243
00246      l4_uint32_t dfl1[L4_VM_VMX_NUM_DFL1_REGS];
00247  } l4_vm_vmx_vcpu_infos_t;
00248
00267  typedef struct l4_vm_vmx_vcpu_state_t
00268  {
00269      l4_vcpu_state_t vcpu_state;
00270      l4_uint8_t padding0[L4_VCPU_OFFSET_EXT_INFOS - sizeof(l4_vcpu_state_t)];
00271
00272      l4_vm_vmx_vcpu_infos_t infos;
00273      l4_uint8_t padding1[L4_VCPU_OFFSET_EXT_STATE - L4_VCPU_OFFSET_EXT_INFOS
00274                          - sizeof(l4_vm_vmx_vcpu_infos_t)];
00275
00276      l4_vm_vmx_vcpu_vmcs_t vmcs;
00277  } l4_vm_vmx_vcpu_state_t;
00278
00288  L4_INLINE
00289  l4_uint64_t
00290  l4_vm_vmx_get_caps(l4_vm_vmx_vcpu_state_t const *vcpu_state,
00291                    enum L4_vm_vmx_caps_regs caps_reg) L4_NOTHROW;
00292
00303  L4_INLINE
00304  l4_uint32_t
00305  l4_vm_vmx_get_caps_default1(l4_vm_vmx_vcpu_state_t const *vcpu_state,
00306                              enum L4_vm_vmx_dfl1_regs dfl1_reg) L4_NOTHROW;
00307
00315  L4_INLINE
00316  unsigned
00317  l4_vm_vmx_field_len(unsigned field) L4_NOTHROW;
00318
00326  L4_INLINE
00327  unsigned
00328  l4_vm_vmx_field_order(unsigned field) L4_NOTHROW;

```

```

00329
00344 L4_INLINE
00345 void *
00346 l4_vm_vmx_field_ptr(l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field) L4_NOTHROW;
00347
00357 L4_INLINE
00358 void
00359 l4_vm_vmx_clear(l4_vm_vmx_vcpu_vmcs_t *vmcs,
00360                l4_vm_vmx_vcpu_vmcs_t *dest_vmcs) L4_NOTHROW;
00361
00371 L4_INLINE
00372 void
00373 l4_vm_vmx_ptr_load(l4_vm_vmx_vcpu_vmcs_t *vmcs,
00374                   l4_vm_vmx_vcpu_vmcs_t *src_vmcs) L4_NOTHROW;
00375
00391 L4_INLINE
00392 l4_uint32_t
00393 l4_vm_vmx_get_cr2_index(l4_vm_vmx_vcpu_vmcs_t const *vmcs) L4_NOTHROW;
00394
00404 L4_INLINE
00405 l4_umword_t
00406 l4_vm_vmx_read_nat(l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field) L4_NOTHROW;
00407
00417 L4_INLINE
00418 l4_uint16_t
00419 l4_vm_vmx_read_16(l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field) L4_NOTHROW;
00420
00430 L4_INLINE
00431 l4_uint32_t
00432 l4_vm_vmx_read_32(l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field) L4_NOTHROW;
00433
00443 L4_INLINE
00444 l4_uint64_t
00445 l4_vm_vmx_read_64(l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field) L4_NOTHROW;
00446
00456 L4_INLINE
00457 l4_uint64_t
00458 l4_vm_vmx_read(l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field) L4_NOTHROW;
00459
00468 L4_INLINE
00469 void
00470 l4_vm_vmx_write_nat(l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field,
00471                    l4_umword_t val) L4_NOTHROW;
00472
00481 L4_INLINE
00482 void
00483 l4_vm_vmx_write_16(l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field,
00484                   l4_uint16_t val) L4_NOTHROW;
00485
00494 L4_INLINE
00495 void
00496 l4_vm_vmx_write_32(l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field,
00497                   l4_uint32_t val) L4_NOTHROW;
00498
00507 L4_INLINE
00508 void
00509 l4_vm_vmx_write_64(l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field,
00510                   l4_uint64_t val) L4_NOTHROW;
00511
00520 L4_INLINE
00521 void
00522 l4_vm_vmx_write(l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field,
00523                l4_uint64_t val) L4_NOTHROW;
00524
00571 L4_INLINE
00572 void
00573 l4_vm_vmx_set_hw_vmcs(l4_vm_vmx_vcpu_vmcs_t *vmcs,
00574                      l4_cap_idx_t vmcs_cap) L4_NOTHROW;
00575
00585 L4_INLINE
00586 l4_cap_idx_t
00587 l4_vm_vmx_get_hw_vmcs(l4_vm_vmx_vcpu_vmcs_t *vmcs) L4_NOTHROW;
00588
00589 /* Implementations */
00590
00591 L4_INLINE
00592 unsigned
00593 l4_vm_vmx_field_len(unsigned field) L4_NOTHROW
00594 {
00595     return 1 « l4_vm_vmx_field_order(field);
00596 }
00597
00598 L4_INLINE
00599 unsigned
00600 l4_vm_vmx_field_order(unsigned field) L4_NOTHROW
00601 {
00602     unsigned size = (field » 13) & 0x03U;

```



```

00603
00604     switch (size)
00605     {
00606         case 0: return 1; /* 16 bits */
00607         case 1: return 3; /* 64 bits */
00608         case 2: return 2; /* 32 bits */
00609         case 3: return (sizeof(l4_umword_t) == 8) ? 3 : 2; /* Natural width */
00610     }
00611
00612     __builtin_trap();
00613 }
00614
00620 L4_INLINE
00621 unsigned
00622 l4_vm_vmx_field_offset(l4_vm_vmx_vcpu_vmcs_t const *vmcs,
00623                       unsigned field) L4_NOTHROW
00624 {
00625     unsigned index = field & 0x3feU;
00626     unsigned size = (field >> 13) & 0x03U;
00627     unsigned group = (field >> 10) & 0x03U;
00628
00629     unsigned shifted_index = index << vmcs->offset_table.index_shifts[size];
00630
00631     if (shifted_index >= (unsigned)vmcs->offset_table.limits[size][group] * 64)
00632         return ~0U;
00633
00634     return (unsigned)vmcs->offset_table.offsets[size][group] * 64
00635         + shifted_index;
00636 }
00637
00638 L4_INLINE
00639 void *
00640 l4_vm_vmx_field_ptr(l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field) L4_NOTHROW
00641 {
00642     unsigned offset = l4_vm_vmx_field_offset(vmcs, field);
00643     if (offset == ~0U)
00644         return 0;
00645
00646     return (void *) (vmcs->values + offset);
00647 }
00648
00654 L4_INLINE
00655 void *
00656 l4_vm_vmx_field_ptr_offset(l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field,
00657                           unsigned *offset) L4_NOTHROW
00658 {
00659     *offset = l4_vm_vmx_field_offset(vmcs, field);
00660     if (*offset == ~0U)
00661         return 0;
00662
00663     return (void *) (vmcs->values + *offset);
00664 }
00665
00671 L4_INLINE
00672 void
00673 l4_vm_vmx_offset_dirty(l4_vm_vmx_vcpu_vmcs_t *vmcs,
00674                      unsigned offset) L4_NOTHROW
00675 {
00676     vmcs->dirty_bitmap[offset / 8] |= 1U << (offset % 8);
00677 }
00678
00683 L4_INLINE
00684 void
00685 l4_vm_vmx_copy_values(l4_vm_vmx_vcpu_vmcs_t const *vmcs, l4_uint8_t *_dst,
00686                     l4_uint8_t const *_src) L4_NOTHROW
00687 {
00688     unsigned base_offset = vmcs->offset_table.base_offset * 64;
00689     unsigned size = vmcs->offset_table.size * 64;
00690
00691     void *dst = _dst + base_offset;
00692     void const *src = _src + base_offset;
00693     __builtin_memcpy(dst, src, size);
00694 }
00695
00696 L4_INLINE
00697 void
00698 l4_vm_vmx_clear(l4_vm_vmx_vcpu_vmcs_t *vmcs,
00699               l4_vm_vmx_vcpu_vmcs_t *dest_vmcs) L4_NOTHROW
00700 {
00701     l4_vm_vmx_vcpu_vmcs_t **current_vmcs_ptr
00702         = (l4_vm_vmx_vcpu_vmcs_t **)&vmcs->user_data;
00703
00704     if (*current_vmcs_ptr != dest_vmcs)
00705         return;
00706
00707     l4_vm_vmx_set_hw_vmcs(dest_vmcs, l4_vm_vmx_get_hw_vmcs(vmcs));
00708     l4_vm_vmx_copy_values(vmcs, dest_vmcs->values, vmcs->values);

```

```

00709
00710 /* Due to its size, the dirty bitmap is always copied in its entirety. */
00711 __builtin_memcpy(dest_vmcs->dirty_bitmap, vmcs->dirty_bitmap,
00712                 L4_VM_VMX_VMCS_SIZE_DIRTY_BITMAP);
00713
00714 *current_vmcs_ptr = 0;
00715 }
00716
00717 L4_INLINE
00718 void
00719 l4_vm_vmx_ptr_load(l4_vm_vmx_vcpu_vmcs_t *vmcs,
00720                  l4_vm_vmx_vcpu_vmcs_t *src_vmcs) L4_NOTHROW
00721 {
00722     l4_vm_vmx_vcpu_vmcs_t **current_vmcs_ptr
00723     = (l4_vm_vmx_vcpu_vmcs_t **)&vmcs->user_data;
00724
00725     if (*current_vmcs_ptr == src_vmcs)
00726         return;
00727
00728     if (*current_vmcs_ptr && *current_vmcs_ptr != src_vmcs)
00729         l4_vm_vmx_clear(vmcs, *current_vmcs_ptr);
00730
00731     *current_vmcs_ptr = src_vmcs;
00732
00733     l4_vm_vmx_set_hw_vmcs(vmcs, l4_vm_vmx_get_hw_vmcs(src_vmcs));
00734     l4_vm_vmx_copy_values(vmcs, vmcs->values, src_vmcs->values);
00735
00736 /* Due to its size, the dirty bitmap is always copied in its entirety. */
00737 __builtin_memcpy(vmcs->dirty_bitmap, src_vmcs->dirty_bitmap,
00738                 L4_VM_VMX_VMCS_SIZE_DIRTY_BITMAP);
00739 }
00740
00741 L4_INLINE
00742 l4_umword_t
00743 l4_vm_vmx_read_nat(l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field) L4_NOTHROW
00744 {
00745     l4_umword_t *ptr = (l4_umword_t *)l4_vm_vmx_field_ptr(vmcs, field);
00746     if (!ptr)
00747         return 0;
00748
00749     return *ptr;
00750 }
00751
00752 L4_INLINE
00753 l4_uint16_t
00754 l4_vm_vmx_read_16(l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field) L4_NOTHROW
00755 {
00756     l4_uint16_t *ptr = (l4_uint16_t *)l4_vm_vmx_field_ptr(vmcs, field);
00757     if (!ptr)
00758         return 0;
00759
00760     return *ptr;
00761 }
00762
00763 L4_INLINE
00764 l4_uint32_t
00765 l4_vm_vmx_read_32(l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field) L4_NOTHROW
00766 {
00767     l4_uint32_t *ptr = (l4_uint32_t *)l4_vm_vmx_field_ptr(vmcs, field);
00768     if (!ptr)
00769         return 0;
00770
00771     return *ptr;
00772 }
00773
00774 L4_INLINE
00775 l4_uint64_t
00776 l4_vm_vmx_read_64(l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field) L4_NOTHROW
00777 {
00778     l4_uint64_t *ptr = (l4_uint64_t *)l4_vm_vmx_field_ptr(vmcs, field);
00779     if (!ptr)
00780         return 0;
00781
00782     return *ptr;
00783 }
00784
00785 L4_INLINE
00786 l4_uint64_t
00787 l4_vm_vmx_read(l4_vm_vmx_vcpu_vmcs_t *vmcs, unsigned field) L4_NOTHROW
00788 {
00789     unsigned size = (field >> 13) & 0x03U;
00790
00791     switch (size)
00792     {
00793     case 0: return l4_vm_vmx_read_16(vmcs, field);
00794     case 1: return l4_vm_vmx_read_64(vmcs, field);
00795     case 2: return l4_vm_vmx_read_32(vmcs, field);

```

```

00796     case 3: return l4_vm_vmx_read_nat(vmc, field);
00797     }
00798
00799     __builtin_trap();
00800 }
00801
00802 L4_INLINE
00803 void
00804 l4_vm_vmx_write_nat(l4_vm_vmx_vcpu_vmc_t *vmc, unsigned field,
00805                    l4_umword_t val) L4_NOTHROW
00806 {
00807     unsigned offset;
00808     l4_umword_t *ptr
00809         = (l4_umword_t *)l4_vm_vmx_field_ptr_offset(vmc, field, &offset);
00810
00811     if ((ptr) && (*ptr != val))
00812     {
00813         *ptr = val;
00814         l4_vm_vmx_offset_dirty(vmc, offset);
00815     }
00816 }
00817
00818 L4_INLINE
00819 void
00820 l4_vm_vmx_write_16(l4_vm_vmx_vcpu_vmc_t *vmc, unsigned field,
00821                   l4_uint16_t val) L4_NOTHROW
00822 {
00823     unsigned offset;
00824     l4_uint16_t *ptr
00825         = (l4_uint16_t *)l4_vm_vmx_field_ptr_offset(vmc, field, &offset);
00826
00827     if ((ptr) && (*ptr != val))
00828     {
00829         *ptr = val;
00830         l4_vm_vmx_offset_dirty(vmc, offset);
00831     }
00832 }
00833
00834 L4_INLINE
00835 void
00836 l4_vm_vmx_write_32(l4_vm_vmx_vcpu_vmc_t *vmc, unsigned field,
00837                   l4_uint32_t val) L4_NOTHROW
00838 {
00839     unsigned offset;
00840     l4_uint32_t *ptr
00841         = (l4_uint32_t *)l4_vm_vmx_field_ptr_offset(vmc, field, &offset);
00842
00843     if ((ptr) && (*ptr != val))
00844     {
00845         *ptr = val;
00846         l4_vm_vmx_offset_dirty(vmc, offset);
00847     }
00848 }
00849
00850 L4_INLINE
00851 void
00852 l4_vm_vmx_write_64(l4_vm_vmx_vcpu_vmc_t *vmc, unsigned field,
00853                   l4_uint64_t val) L4_NOTHROW
00854 {
00855     unsigned offset;
00856     l4_uint64_t *ptr
00857         = (l4_uint64_t *)l4_vm_vmx_field_ptr_offset(vmc, field, &offset);
00858
00859     if ((ptr) && (*ptr != val))
00860     {
00861         *ptr = val;
00862         l4_vm_vmx_offset_dirty(vmc, offset);
00863     }
00864 }
00865
00866 L4_INLINE
00867 void
00868 l4_vm_vmx_write(l4_vm_vmx_vcpu_vmc_t *vmc, unsigned field,
00869                 l4_uint64_t val) L4_NOTHROW
00870 {
00871     unsigned size = (field >> 13) & 0x03U;
00872
00873     switch (size)
00874     {
00875     case 0: l4_vm_vmx_write_16(vmc, field, val); break;
00876     case 1: l4_vm_vmx_write_64(vmc, field, val); break;
00877     case 2: l4_vm_vmx_write_32(vmc, field, val); break;
00878     case 3: l4_vm_vmx_write_nat(vmc, field, val); break;
00879     }
00880 }
00881
00882 L4_INLINE

```

```

00883 l4_uint64_t
00884 l4_vm_vmx_get_caps(l4_vm_vmx_vcpu_state_t const *vcpu_state,
00885                     enum L4_vm_vmx_caps_regs caps_reg) L4_NOTHROW
00886 {
00887     return vcpu_state->infos.caps[caps_reg];
00888 }
00889
00890 L4_INLINE
00891 l4_uint32_t
00892 l4_vm_vmx_get_caps_default1(l4_vm_vmx_vcpu_state_t const *vcpu_state,
00893                             enum L4_vm_vmx_dfll_regs dfll_reg) L4_NOTHROW
00894 {
00895     return vcpu_state->infos.dfll[dfll_reg];
00896 }
00897
00898 L4_INLINE
00899 l4_uint32_t
00900 l4_vm_vmx_get_cr2_index(l4_vm_vmx_vcpu_vmcs_t const *vmcs) L4_NOTHROW
00901 {
00902     return vmcs->cr2_index;
00903 }
00904
00905 L4_INLINE
00906 void
00907 l4_vm_vmx_set_hw_vmcs(l4_vm_vmx_vcpu_vmcs_t *vmcs,
00908                      l4_cap_idx_t vmcs_cap) L4_NOTHROW
00909 {
00910     vmcs->vmcs = vmcs_cap;
00911 }
00912
00913 L4_INLINE
00914 l4_cap_idx_t
00915 l4_vm_vmx_get_hw_vmcs(l4_vm_vmx_vcpu_vmcs_t *vmcs) L4_NOTHROW
00916 {
00917     return vmcs->vmcs & L4_CAP_MASK;
00918 }

```

16.446 l4/sys/arm_smccc File Reference

ARM secure monitor call functions.

```

#include <l4/sys/capability>
#include <l4/sys/cxx/ipc_iface>

```

[illegible]

- class `L4::Arm_smccc`
Wrapper for function calls that follow the ARM SMC/HVC calling convention.

- namespace **L4**
L4 low-level kernel interface.

Definition in file [arm_smccc](#).

16.447 arm_smccc

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * Copyright (C) 2018, 2022, 2024 Kernkonzept GmbH.
00004  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00012 #pragma once
00013
00014 #include <l4/sys/capability>
00015 #include <l4/sys/cxx/ipc_iface>
00016
00017 namespace L4 {
00018
00023 class L4_EXPORT Arm_smccc : public Kobject_0t<Arm_smccc, L4_PROTO_SMCCC>
00024 {
00025 public:
00064     L4_INLINE_RPC(l4_msgtag_t, call,
00065                   (l4_umword_t func, l4_umword_t in0, l4_umword_t in1,
00066                    l4_umword_t in2, l4_umword_t in3, l4_umword_t in4,
00067                    l4_umword_t in5, l4_umword_t *out0, l4_umword_t *out1,
00068                    l4_umword_t *out2, l4_umword_t *out3,
00069                    l4_umword_t client_id));
00070
00071     typedef L4::Typeid::Rpc_nocode<call_t> Rpcs;
00072 };
00073
00074 }

```

16.448 l4/sys/arm_smccc.h File Reference

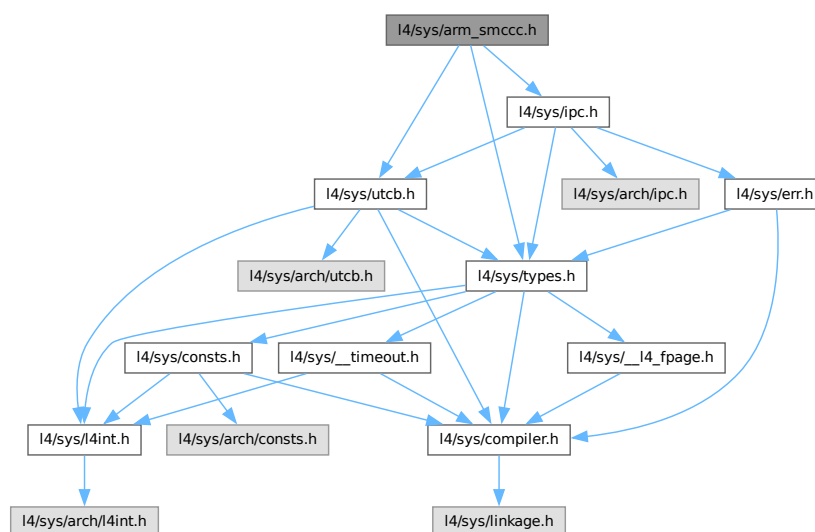
ARM secure monitor call functions.

```

#include <l4/sys/types.h>
#include <l4/sys/utcb.h>
#include <l4/sys/ipc.h>

```

Include dependency graph for arm_smccc.h:



Functions

- [l4_msgtag_t l4_arm_smccc_call](#) ([l4_cap_idx_t pfc](#), [l4_umword_t func](#), [l4_umword_t in0](#), [l4_umword_t in1](#), [l4_umword_t in2](#), [l4_umword_t in3](#), [l4_umword_t in4](#), [l4_umword_t in5](#), [l4_umword_t *out0](#), [l4_umword_t *out1](#), [l4_umword_t *out2](#), [l4_umword_t *out3](#), [l4_umword_t client_id](#)) [L4_NOTHROW](#)

C interface for calling the ARM secure monitor, see [L4::Arm_smccc::call\(\)](#) for the C++ interface.

16.448.1 Detailed Description

ARM secure monitor call functions.

Definition in file [arm_smccc.h](#).

16.448.2 Function Documentation

16.448.2.1 l4_arm_smccc_call()

```
l4_msgtag_t l4_arm_smccc_call (
    l4_cap_idx_t pfc,
    l4_umword_t func,
    l4_umword_t in0,
    l4_umword_t in1,
    l4_umword_t in2,
    l4_umword_t in3,
    l4_umword_t in4,
    l4_umword_t in5,
    l4_umword_t * out0,
    l4_umword_t * out1,
    l4_umword_t * out2,
    l4_umword_t * out3,
    l4_umword_t client_id) [inline]
```

C interface for calling the ARM secure monitor, see [L4::Arm_smccc::call\(\)](#) for the C++ interface.

Parameters

| | |
|-------------|--|
| <i>pfc</i> | Capability of the SMC kernel object. The input parameters consist of a function identifier, 6 arguments and a client id. Results are returned in 4 output parameters. |
| <i>func</i> | Function identifier. <ul style="list-style-type: none"> • Bit 31 has to be set: This marks the call as <i>Fast Call</i>. <i>Yielding Calls</i> (bit 31 unset) are rejected by the kernel. • Bit 30 defines the calling convention: • Bit 30 == 1: 64-bit calling convention. • Bit 30 == 0: 32-bit calling convention. • Bits 24..29 determine the service call ID. The permitted IDs are set in the kernel configuration. By default only service IDs >= 0x30000000 (<i>Trusted Application Calls</i> and <i>Trusted OS Calls</i>) are allowed. |

| | | |
|-----|------------------|--|
| in | <i>in0</i> | First input parameter. |
| in | <i>in1</i> | Second input parameter. |
| in | <i>in2</i> | Third input parameter. |
| in | <i>in3</i> | Fourth input parameter. |
| in | <i>in4</i> | Fifth input parameter. |
| in | <i>in5</i> | Sixth input parameter. |
| out | <i>out0</i> | First output parameter. |
| out | <i>out1</i> | Second output parameter. |
| out | <i>out2</i> | Third output parameter. |
| out | <i>out3</i> | Fourth output parameter. |
| in | <i>client_id</i> | Client ID. According to the specification, this value might be ignored by certain functions. |

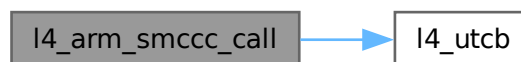
Return values

| | |
|-------------------------|---|
| <code>-L4_ENOSYS</code> | Either bit 31 of the function call not set or service ID outside the range permitted by kernel configuration. |
| <code>-L4_EINVAL</code> | Invalid number of parameters. |
| <code><0</code> | Other L4 error. |
| <code>0</code> | Success. |

Definition at line 42 of file [arm_smccc.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



16.449 arm_smccc.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (C) 2018, 2022, 2024 Kernkonzept GmbH.
00003  * Author(s): Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00011 #pragma once
00012
00013 #include <l4/sys/types.h>
00014 #include <l4/sys/utcb.h>
  
```



```

00015
00016 L4_INLINE l4_msgtag_t
00017 l4_arm_smccc_call(l4_cap_idx_t pfc, l4_umword_t func, l4_umword_t in0,
00018                  l4_umword_t in1, l4_umword_t in2, l4_umword_t in3,
00019                  l4_umword_t in4, l4_umword_t in5, l4_umword_t *out0,
00020                  l4_umword_t *out1, l4_umword_t *out2, l4_umword_t *out3,
00021                  l4_umword_t client_id) L4_NOTHROW;
00022
00023 L4_INLINE l4_msgtag_t
00024 l4_arm_smccc_call_u(l4_cap_idx_t pfc, l4_umword_t func, l4_umword_t in0,
00025                    l4_umword_t in1, l4_umword_t in2, l4_umword_t in3,
00026                    l4_umword_t in4, l4_umword_t in5, l4_umword_t *out0,
00027                    l4_umword_t *out1, l4_umword_t *out2, l4_umword_t *out3,
00028                    l4_umword_t client_id, l4_utcb_t *utcb) L4_NOTHROW;
00029
00030 /* IMPLEMENTATION ----- */
00031
00032 #include <l4/sys/ipc.h>
00033
00041 L4_INLINE l4_msgtag_t
00042 l4_arm_smccc_call(l4_cap_idx_t pfc, l4_umword_t func,
00043                  l4_umword_t in0, l4_umword_t in1,
00044                  l4_umword_t in2, l4_umword_t in3,
00045                  l4_umword_t in4, l4_umword_t in5,
00046                  l4_umword_t *out0, l4_umword_t *out1,
00047                  l4_umword_t *out2, l4_umword_t *out3,
00048                  l4_umword_t client_id) L4_NOTHROW
00049 {
00050     return l4_arm_smccc_call_u(pfc, func, in0, in1, in2, in3, in4, in5,
00051                                out0, out1, out2, out3, client_id, l4_utcb());
00052 }
00053
00054
00055 L4_INLINE l4_msgtag_t
00056 l4_arm_smccc_call_u(l4_cap_idx_t pfc, l4_umword_t func, l4_umword_t in0,
00057                    l4_umword_t in1, l4_umword_t in2, l4_umword_t in3,
00058                    l4_umword_t in4, l4_umword_t in5, l4_umword_t *out0,
00059                    l4_umword_t *out1, l4_umword_t *out2, l4_umword_t *out3,
00060                    l4_umword_t client_id, l4_utcb_t *utcb) L4_NOTHROW
00061 {
00062     l4_msgtag_t ret;
00063     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00064     v->mr[0] = func;
00065     v->mr[1] = in0;
00066     v->mr[2] = in1;
00067     v->mr[3] = in2;
00068     v->mr[4] = in3;
00069     v->mr[5] = in4;
00070     v->mr[6] = in5;
00071     v->mr[7] = client_id;
00072
00073     ret = l4_ipc_call(pfc, utcb, l4_msgtag(L4_PROTO_SMCCC, 8, 0, 0),
00074                     L4_IPC_NEVER);
00075
00076     if (l4_error(ret) >= 0)
00077     {
00078         *out0 = v->mr[0];
00079         *out1 = v->mr[1];
00080         *out2 = v->mr[2];
00081         *out3 = v->mr[3];
00082     }
00083
00084     return ret;
00085 }

```

16.450 amd64/l4/sys/arch/cache.h File Reference

Cache functions.

Functions

- int [l4_cache_clean_data](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)
Cache clean a range in D-cache; writes back to PoC.
- int [l4_cache_flush_data](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)
Cache flush a range; writes back to PoC.

- int [l4_cache_inv_data](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)
Cache invalidate a range; might write back to PoC.
- int [l4_cache_coherent](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)
Make memory coherent between I-cache and D-cache; writes back to PoU.
- int [l4_cache_dma_coherent](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)
Make memory coherent for use with external memory; writes back to PoC.
- int [l4_cache_dma_coherent_full](#) (void) [L4_NOTHROW](#)
Make memory coherent for use with external memory; writes back to PoC.

16.450.1 Detailed Description

Cache functions.

Definition in file [cache.h](#).

16.451 `cache.h`

[Go to the documentation of this file.](#)

```

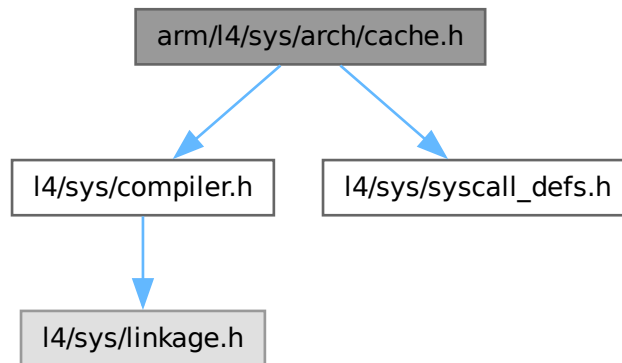
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *      economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011 #ifndef __L4SYS__INCLUDE__ARCH_AMD64__CACHE_H__
00012 #define __L4SYS__INCLUDE__ARCH_AMD64__CACHE_H__
00013
00014 L4_INLINE int
00015 l4_cache_clean_data(unsigned long start,
00016                    unsigned long end) L4_NOTHROW
00017 {
00018     (void)start; (void)end;
00019     return 0;
00020 }
00021
00022 L4_INLINE int
00023 l4_cache_flush_data(unsigned long start,
00024                    unsigned long end) L4_NOTHROW
00025 {
00026     (void)start; (void)end;
00027     return 0;
00028 }
00029
00030 L4_INLINE int
00031 l4_cache_inv_data(unsigned long start,
00032                  unsigned long end) L4_NOTHROW
00033 {
00034     (void)start; (void)end;
00035     return 0;
00036 }
00037
00038 L4_INLINE int
00039 l4_cache_coherent(unsigned long start,
00040                  unsigned long end) L4_NOTHROW
00041 {
00042     (void)start; (void)end;
00043     return 0;
00044 }
00045
00046 L4_INLINE int
00047 l4_cache_dma_coherent(unsigned long start,
00048                      unsigned long end) L4_NOTHROW
00049 {
00050     (void)start; (void)end;
00051     return 0;
00052 }
00053
00054 L4_INLINE int
00055 l4_cache_dma_coherent_full(void) L4_NOTHROW
00056 {
00057     return 0;
00058 }
00059
00060 #endif /* ! __L4SYS__INCLUDE__ARCH_AMD64__CACHE_H__ */

```

16.452 arm/l4/sys/arch/cache.h File Reference

Cache functions.

```
#include <l4/sys/compiler.h>
#include <l4/sys/syscall_defs.h>
Include dependency graph for cache.h:
```



Functions

- `int l4_cache_clean_data` (unsigned long start, unsigned long end) `L4_NOTHROW`
Cache clean a range in D-cache; writes back to PoC.
- `int l4_cache_flush_data` (unsigned long start, unsigned long end) `L4_NOTHROW`
Cache flush a range; writes back to PoC.
- `int l4_cache_inv_data` (unsigned long start, unsigned long end) `L4_NOTHROW`
Cache invalidate a range; might write back to PoC.
- `int l4_cache_coherent` (unsigned long start, unsigned long end) `L4_NOTHROW`
Make memory coherent between I-cache and D-cache; writes back to PoU.
- `int l4_cache_dma_coherent` (unsigned long start, unsigned long end) `L4_NOTHROW`
Make memory coherent for use with external memory; writes back to PoC.
- `int l4_cache_dma_coherent_full` (void) `L4_NOTHROW`
Make memory coherent for use with external memory; writes back to PoC.

16.452.1 Detailed Description

Cache functions.

Date

2007-11

Author

Adam Lackorzynski adam@os.inf.tu-dresden.de

Definition in file [cache.h](#).

16.453 cache.h

[Go to the documentation of this file.](#)

```

00001
00009 /*
00010  * (c) 2007-2009 Author(s)
00011  *     economic rights: Technische Universität Dresden (Germany)
00012  *
00013  * License: see LICENSE.spdx (in this directory or the directories above)
00014  */
00015 #ifndef __L4SYS__INCLUDE__ARCH_ARM__CACHE_H__
00016 #define __L4SYS__INCLUDE__ARCH_ARM__CACHE_H__
00017
00018 #include <l4/sys/compiler.h>
00019 #include <l4/sys/syscall_defs.h>
00020
00024 L4_INLINE void
00025 l4_cache_op_arm_call(unsigned long op,
00026                     unsigned long start,
00027                     unsigned long end);
00028
00029 L4_INLINE void
00030 l4_cache_op_arm_call(unsigned long op,
00031                     unsigned long start,
00032                     unsigned long end)
00033 {
00034     register unsigned long _op    __asm__ ("r0") = op;
00035     register unsigned long _start __asm__ ("r1") = start;
00036     register unsigned long _end   __asm__ ("r2") = end;
00037
00038     __asm__ __volatile__
00039     ("@ l4_cache_op_arm_call(start) \n\t"
00040      "mov     r5, %[sc]          \n\t"
00041      "blx     __l4_sys_syscall   \n\t"
00042      "@ l4_cache_op_arm_call(end) \n\t"
00043      :
00044      "=r" (_op),
00045      "=r" (_start),
00046      "=r" (_end)
00047      :
00048      [sc] "i" (L4_SYSCALL_MEM_OP),
00049      "0" (_op),
00050      "1" (_start),
00051      "2" (_end)
00052      :
00053      "cc", "memory", "r5", "ip", "lr"
00054      );
00055 }
00056
00057 enum L4_mem_cache_ops
00058 {
00059     L4_MEM_CACHE_OP_CLEAN_DATA      = 0,
00060     L4_MEM_CACHE_OP_FLUSH_DATA      = 1,
00061     L4_MEM_CACHE_OP_INV_DATA        = 2,
00062     L4_MEM_CACHE_OP_COHERENT        = 3,
00063     L4_MEM_CACHE_OP_DMA_COHERENT    = 4,
00064     L4_MEM_CACHE_OP_DMA_COHERENT_FULL = 5,
00065 };
00066
00067 L4_INLINE int
00068 l4_cache_clean_data(unsigned long start,
00069                    unsigned long end) L4_NOTHROW
00070 {
00071     l4_cache_op_arm_call(L4_MEM_CACHE_OP_CLEAN_DATA, start, end);
00072     return 0;
00073 }
00074
00075 L4_INLINE int
00076 l4_cache_flush_data(unsigned long start,
00077                    unsigned long end) L4_NOTHROW
00078 {
00079     l4_cache_op_arm_call(L4_MEM_CACHE_OP_FLUSH_DATA, start, end);
00080     return 0;
00081 }
00082
00083 L4_INLINE int
00084 l4_cache_inv_data(unsigned long start,
00085                  unsigned long end) L4_NOTHROW
00086 {
00087     l4_cache_op_arm_call(L4_MEM_CACHE_OP_INV_DATA, start, end);
00088     return 0;
00089 }
00090
00091 L4_INLINE int
00092 l4_cache_coherent(unsigned long start,
```

```

00093             unsigned long end) L4_NOTHROW
00094 {
00095     l4_cache_op_arm_call(L4_MEM_CACHE_OP_COHERENT, start, end);
00096     return 0;
00097 }
00098
00099 L4_INLINE int
00100 l4_cache_dma_coherent(unsigned long start,
00101                      unsigned long end) L4_NOTHROW
00102 {
00103     l4_cache_op_arm_call(L4_MEM_CACHE_OP_DMA_COHERENT, start, end);
00104     return 0;
00105 }
00106
00107 L4_INLINE int
00108 l4_cache_dma_coherent_full(void) L4_NOTHROW
00109 {
00110     l4_cache_op_arm_call(L4_MEM_CACHE_OP_DMA_COHERENT_FULL, 0, 0);
00111     return 0;
00112 }
00113
00114 #endif /* ! __L4SYS__INCLUDE__ARCH_ARM__CACHE_H__ */

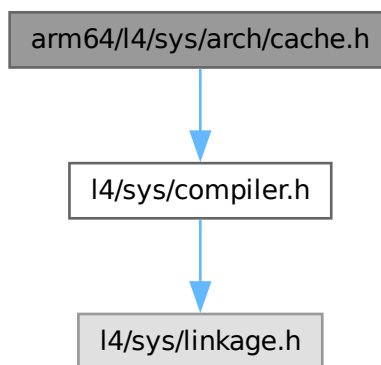
```

16.454 arm64/l4/sys/arch/cache.h File Reference

Cache functions.

```
#include <l4/sys/compiler.h>
```

Include dependency graph for cache.h:



Functions

- int [l4_cache_clean_data](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)
Cache clean a range in D-cache; writes back to PoC.
- int [l4_cache_flush_data](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)
Cache flush a range; writes back to PoC.
- int [l4_cache_inv_data](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)
Cache invalidate a range; might write back to PoC.
- int [l4_cache_coherent](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)
Make memory coherent between I-cache and D-cache; writes back to PoU.
- int [l4_cache_dma_coherent](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)
Make memory coherent for use with external memory; writes back to PoC.

16.454.1 Detailed Description

Cache functions.

Date

2007-11

Author

Adam Lackorzynski adam@os.inf.tu-dresden.de

Definition in file [cache.h](#).

16.455 cache.h

[Go to the documentation of this file.](#)

```

00001
00009 /*
00010  * (c) 2007-2009 Author(s)
00011  *      economic rights: Technische Universität Dresden (Germany)
00012  *
00013  * License: see LICENSE.spdx (in this directory or the directories above)
00014  */
00015 #ifndef __L4SYS__INCLUDE__ARCH_ARM__CACHE_H__
00016 #define __L4SYS__INCLUDE__ARCH_ARM__CACHE_H__
00017
00018 #include <l4/sys/compiler.h>
00019
00020 L4_INLINE unsigned long __attribute__((pure, always_inline))
00021 l4_cache_arm_ctr(void);
00022
00023 L4_INLINE unsigned long __attribute__((pure, always_inline))
00024 l4_cache_arm_ctr(void)
00025 {
00026     unsigned long v;
00027     asm ("mrs %0, CTR_EL0" : "=r"(v));
00028     return v;
00029 }
00030
00031 L4_INLINE unsigned __attribute__((pure, always_inline))
00032 l4_cache_dmin_line(void);
00033
00034 L4_INLINE unsigned __attribute__((pure, always_inline))
00035 l4_cache_dmin_line(void)
00036 {
00037     return 4U < ((l4_cache_arm_ctr() > 16) & 0xf);
00038 }
00039
00040 #define L4_ARM_CACHE_LOOP(op)
00041     unsigned long step;
00042
00043     if (start > end)
00044         __builtin_unreachable();
00045
00046     step = l4_cache_dmin_line();
00047     start &= ~(step - 1);
00048     end = (end + step - 1) & ~(step - 1);
00049     for (; start != end; start += step)
00050         asm volatile (op " ", %0" : : "r"(start) : "memory");
00051     asm volatile ("dsb ish");
00052
00053
00054 L4_INLINE int
00055 l4_cache_clean_data(unsigned long start,
00056                     unsigned long end) L4_NOTHROW
00057 {
00058     L4_ARM_CACHE_LOOP("dc cvac");
00059     return 0;
00060 }
00061
00062 L4_INLINE int
```

```

00063 l4_cache_flush_data(unsigned long start,
00064                     unsigned long end) L4_NOTHROW
00065 {
00066     L4_ARM_CACHE_LOOP("dc civac");
00067     return 0;
00068 }
00069
00070 L4_INLINE int
00071 l4_cache_inv_data(unsigned long start,
00072                 unsigned long end) L4_NOTHROW
00073 {
00074     // DC IVAC is always privileged, use DC CIVAC instead
00075     L4_ARM_CACHE_LOOP("dc civac");
00076     return 0;
00077 }
00078
00079 L4_INLINE int
00080 l4_cache_coherent(unsigned long start,
00081                 unsigned long end) L4_NOTHROW
00082 {
00083     L4_ARM_CACHE_LOOP("dc cvau, %0; ic ivau");
00084     asm volatile ("isb");
00085     return 0;
00086 }
00087
00088 L4_INLINE int
00089 l4_cache_dma_coherent(unsigned long start,
00090                     unsigned long end) L4_NOTHROW
00091 {
00092     L4_ARM_CACHE_LOOP("dc civac");
00093     return 0;
00094 }
00095
00096 #undef L4_ARM_CACHE_LOOP
00097
00098 #endif /* ! __L4SYS__INCLUDE__ARCH_ARM__CACHE_H__ */

```

16.456 l4/sys/cache.h File Reference

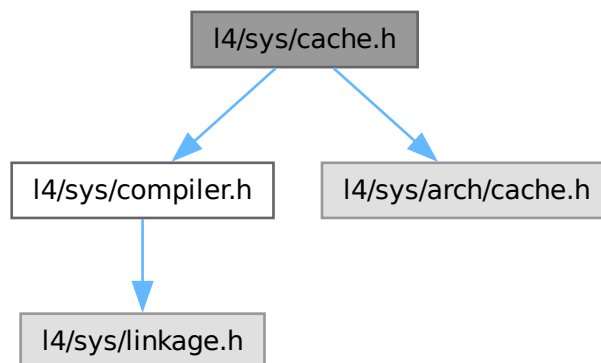
Cache-consistency functions.

```

#include <l4/sys/compiler.h>
#include <l4/sys/arch/cache.h>

```

Include dependency graph for cache.h:



Functions

- [L4_BEGIN_DECLS](#) int [l4_cache_clean_data](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)
Cache clean a range in D-cache; writes back to PoC.
- int [l4_cache_flush_data](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)
Cache flush a range; writes back to PoC.
- int [l4_cache_inv_data](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)
Cache invalidate a range; might write back to PoC.
- int [l4_cache_coherent](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)
Make memory coherent between I-cache and D-cache; writes back to PoU.
- int [l4_cache_dma_coherent](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)
Make memory coherent for use with external memory; writes back to PoC.
- int [l4_cache_dma_coherent_full](#) (void) [L4_NOTHROW](#)
Make memory coherent for use with external memory; writes back to PoC.

16.456.1 Detailed Description

Cache-consistency functions.

Date

2007-11

Author

Adam Lackorzynski adam@os.inf.tu-dresden.de

Definition in file [cache.h](#).

16.457 cache.h

[Go to the documentation of this file.](#)

```

00001
00010 /*
00011  * (c) 2007-2009 Author(s)
00012  *      economic rights: Technische Universität Dresden (Germany)
00013  *
00014  * License: see LICENSE.spdx (in this directory or the directories above)
00015  */
00016
00017 #ifndef __L4SYS__INCLUDE__CACHE_H__
00018 #define __L4SYS__INCLUDE__CACHE_H__
00019
00020 #include <l4/sys/compiler.h>
00021
00036
00037 L4_BEGIN_DECLS
00038
00053 L4_INLINE int
00054 l4_cache_clean_data(unsigned long start,
00055                    unsigned long end) L4_NOTHROW;
00056
00071 L4_INLINE int
00072 l4_cache_flush_data(unsigned long start,
00073                    unsigned long end) L4_NOTHROW;
00074
00093 L4_INLINE int
00094 l4_cache_inv_data(unsigned long start,
00095                  unsigned long end) L4_NOTHROW;
00096

```



```

00108 L4_INLINE int
00109 l4_cache_coherent(unsigned long start,
00110                  unsigned long end) L4_NOTHROW;
00111
00123 L4_INLINE int
00124 l4_cache_dma_coherent(unsigned long start,
00125                      unsigned long end) L4_NOTHROW;
00126
00127 #if !defined(ARCH_arm64)
00132 L4_INLINE int
00133 l4_cache_dma_coherent_full(void) L4_NOTHROW;
00134 #endif
00135
00136 L4_END_DECLS
00137
00138 #include <l4/sys/arch/cache.h>
00139
00140 #endif /* ! __L4SYS__INCLUDE__CACHE_H__ */

```

16.458 riscv/l4/sys/arch/cache.h File Reference

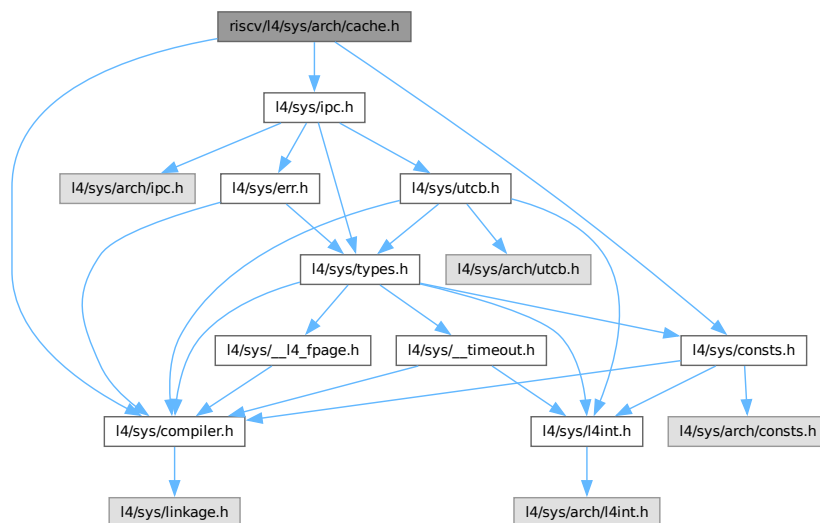
Cache functions.

```

#include <l4/sys/ipc.h>
#include <l4/sys/consts.h>
#include <l4/sys/compiler.h>

```

Include dependency graph for cache.h:



Functions

- int [l4_cache_clean_data](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)
Cache clean a range in D-cache; writes back to PoC.
- int [l4_cache_flush_data](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)
Cache flush a range; writes back to PoC.
- int [l4_cache_inv_data](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)
Cache invalidate a range; might write back to PoC.
- int [l4_cache_coherent](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)

Make memory coherent between I-cache and D-cache; writes back to PoU.

- int `l4_cache_dma_coherent` (unsigned long start, unsigned long end) `L4_NOTHROW`

Make memory coherent for use with external memory; writes back to PoC.

- int `l4_cache_dma_coherent_full` (void) `L4_NOTHROW`

Make memory coherent for use with external memory; writes back to PoC.

16.458.1 Detailed Description

Cache functions.

Definition in file `cache.h`.

16.459 `cache.h`

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * Copyright (C) 2021, 2024 Kernkonzept GmbH.
00007  * Author(s): Georg Kotheimer <georg.kotheimer@kernkonzept.com>
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011 #pragma once
00012
00013 #include <l4/sys/ipc.h>
00014 #include <l4/sys/consts.h>
00015 #include <l4/sys/compiler.h>
00016
00017 L4_INLINE int
00018 l4_cache_clean_data(unsigned long start,
00019                    unsigned long end) L4_NOTHROW
00020 {
00021     (void)start; (void)end;
00022     return 0;
00023 }
00024
00025 L4_INLINE int
00026 l4_cache_flush_data(unsigned long start,
00027                    unsigned long end) L4_NOTHROW
00028 {
00029     (void)start; (void)end;
00030     return 0;
00031 }
00032
00033 L4_INLINE int
00034 l4_cache_inv_data(unsigned long start,
00035                  unsigned long end) L4_NOTHROW
00036 {
00037     (void)start; (void)end;
00038     return 0;
00039 }
00040
00041 L4_INLINE int
00042 l4_cache_coherent(unsigned long start,
00043                  unsigned long end) L4_NOTHROW
00044 {
00045     (void)start; (void)end;
00046     asm volatile ("fence.i");
00047     return 0;
00048 }
00049
00050 L4_INLINE int
00051 l4_cache_dma_coherent(unsigned long start,
00052                      unsigned long end) L4_NOTHROW
00053 {
00054     (void)start; (void)end;
00055     return 0;
00056 }
00057
00058 L4_INLINE int
00059 l4_cache_dma_coherent_full(void) L4_NOTHROW
00060 {
00061     return 0;
00062 }

```

16.460 x86/I4/sys/arch/cache.h File Reference

Cache functions.

Functions

- int [l4_cache_clean_data](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)
Cache clean a range in D-cache; writes back to PoC.
- int [l4_cache_flush_data](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)
Cache flush a range; writes back to PoC.
- int [l4_cache_inv_data](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)
Cache invalidate a range; might write back to PoC.
- int [l4_cache_coherent](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)
Make memory coherent between I-cache and D-cache; writes back to PoU.
- int [l4_cache_dma_coherent](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)
Make memory coherent for use with external memory; writes back to PoC.
- int [l4_cache_dma_coherent_full](#) (void) [L4_NOTHROW](#)
Make memory coherent for use with external memory; writes back to PoC.

16.460.1 Detailed Description

Cache functions.

Definition in file [cache.h](#).

16.461 cache.h

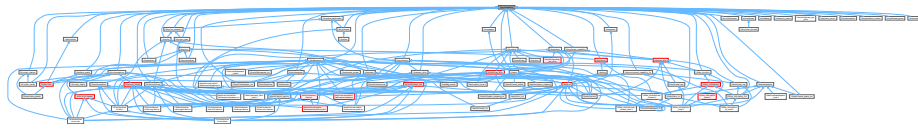
[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *      economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011 #ifndef __L4SYS__INCLUDE__ARCH_X86__CACHE_H__
00012 #define __L4SYS__INCLUDE__ARCH_X86__CACHE_H__
00013
00014 L4_INLINE int
00015 l4_cache_clean_data(unsigned long start,
00016                    unsigned long end) L4_NOTHROW
00017 {
00018     (void)start; (void)end;
00019     return 0;
00020 }
00021
00022 L4_INLINE int
00023 l4_cache_flush_data(unsigned long start,
00024                    unsigned long end) L4_NOTHROW
00025 {
00026     (void)start; (void)end;
00027     return 0;
00028 }
00029
00030 L4_INLINE int
00031 l4_cache_inv_data(unsigned long start,
00032                  unsigned long end) L4_NOTHROW
00033 {
00034     (void)start; (void)end;
00035     return 0;
00036 }

```


This graph shows which files directly or indirectly include this file:



Namespaces

- namespace [L4](#)
[L4](#) low-level kernel interface.

Macros

- `#define L4_DISABLE_COPY(_class)`
Disable copy of a class.

Functions

- `template<typename T, typename F>`
`Cap< T > L4::cap_dynamic_cast (Cap< F > const &c) noexcept`
dynamic_cast for capabilities.

16.462.1 Detailed Description

[L4::Cap](#) related definitions.

Author

Alexander Warg alexander.warg@os.inf.tu-dresden.de

Definition in file [capability](#).

16.462.2 Macro Definition Documentation

16.462.2.1 [L4_DISABLE_COPY](#)

```
#define L4_DISABLE_COPY(  
    _class)
```

Value:

```
public:
    _class(_class const &) = delete; \
    _class operator = (_class const &) = delete; \
private:
```

Disable copy of a class.

Parameters

| | |
|---------------------|---|
| <code>_class</code> | Name of the class that shall not have value copy semantics. |
|---------------------|---|

The typical use of this is:

```
class Non_value
{
    L4_DISABLE_COPY(Non_value)

    ...
}
```

Definition at line 52 of file [capability](#).

16.463 capability

[Go to the documentation of this file.](#)

```
00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00009 /*
00010  * (c) 2008-2009,2015 Author(s)
00011  *     economic rights: Technische Universität Dresden (Germany)
00012  *
00013  * License: see LICENSE.spdx (in this directory or the directories above)
00014  */
00015 #pragma once
00016
00017 #include <l4/sys/consts.h>
00018 #include <l4/sys/types.h>
00019 #include <l4/sys/kobject>
00020 #include <l4/sys/task.h>
00021
00022 namespace L4
00023 {
00024
00025     /* Forward declarations for our kernel object classes. */
00026     class Task;
00027     class Thread;
00028     class Thread_group;
00029     class Factory;
00030     class Irq;
00031     class Log;
00032     class Vm;
00033     class Vcpu_context;
00034     class Kobject;
00035
00051     #if __cplusplus >= 201103L
00052     #   define L4_DISABLE_COPY(_class) \
00053         public: \
00054             _class(_class const &) = delete; \
00055             _class operator = (_class const &) = delete; \
00056         private:
00057     #else
00058     #   define L4_DISABLE_COPY(_class) \
00059         private: \
00060             _class(_class const &); \
00061             _class operator = (_class const &);
00062     #endif
00063
00064
00065     #define L4_KOBJECT_DISABLE_COPY(_class) \
00066         protected: \
00067             _class(); \
00068             L4_DISABLE_COPY(_class)
00069
00070
00071     #define L4_KOBJECT(_class) L4_KOBJECT_DISABLE_COPY(_class)
00072
00073     inline l4_msgtag_t
00074     Cap_base::validate(Cap<Task> task, l4_utcb_t *u) const noexcept
00075     {
00076         return is_valid() ? l4_task_cap_valid_u(task.cap(), _c, u)
00077             : l4_msgtag(0, 0, 0, 0);
00078     }
00079
00080     inline l4_msgtag_t
```

```

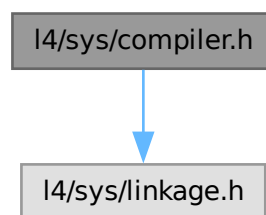
00081 Cap_base::validate(l4_utcb_t *u) const noexcept
00082 {
00083     return is_valid() ? l4_task_cap_valid_u(L4_BASE_TASK_CAP, _c, u)
00084         : l4_msgtag(0, 0, 0, 0);
00085 }
00086
00087 }; // namespace L4
00088
00089 #include <l4/sys/meta>
00090
00091 namespace L4 {
00092
00114 template< typename T, typename F >
00115 inline
00116 Cap<T> cap_dynamic_cast(Cap<F> const &c) noexcept
00117 {
00118     if (!c.is_valid())
00119         return Cap<T>::Invalid;
00120
00121     Cap<Meta> mc = cap_reinterpret_cast<Meta>(c);
00122     Type_info const *m = kobject_typeid<T>();
00123     if (m->proto() && l4_error(mc->supports(m->proto())) > 0)
00124         return Cap<T>(c.cap());
00125
00126     // FIXME: use generic checker
00127     #if 0
00128     if (l4_error(mc->supports(T::kobject_proto())) > 0)
00129         return Cap<T>(c.cap());
00130     #endif
00131
00132     return Cap<T>::Invalid;
00133 }
00134
00135 }

```

16.464 l4/sys/compiler.h File Reference

L4 compiler related defines.

#include <l4/sys/linkage.h>
Include dependency graph for compiler.h:



This graph shows which files directly or indirectly include this file:



Macros

- **#define L4_INLINE**
L4 Inline function attribute.
- **#define L4_ALWAYS_INLINE**
Always inline a function.
- **#define L4_NOTHROW**
Mark a function declaration and definition as never throwing an exception.
- **#define L4_BEGIN_DECLS**
Start section with C types and functions.
- **#define L4_END_DECLS**
End section with C types and functions.
- **#define L4_CONSTEXPR**
Constexpr function attribute.
- **#define L4_NORETURN**
Noreturn function attribute.
- **#define L4_NOINSTRUMENT**
No instrumentation function attribute.
- **#define L4_HIDDEN**
Attribute to mark functions, variables, and data types as being explicitly hidden from users of a library.
- **#define L4_EXPORT**
Attribute to mark functions, variables, and data types as being exported from a library.
- **#define L4_LIKELY(x)**
Expression is likely to execute.
- **#define L4_UNLIKELY(x)**
Expression is unlikely to execute.
- **#define L4_STICKY(x)**
Mark symbol sticky (even not there).
- **#define L4_DEPRECATED(s)**
Mark symbol deprecated.
- **#define L4_stringify_helper(x)**
stringify helper.
- **#define L4_stringify(x)**
stringify.

Functions

- unsigned long **l4_align_stack_for_direct_fncall** (unsigned long stack)
Specify the desired alignment of the stack pointer.
- void **l4_barrier** (void)
Memory barrier.
- void **l4_mb** (void)
Memory barrier.
- void **l4_wmb** (void)
Write memory barrier.
- **L4_NORETURN** void **l4_infinite_loop** (void)
Infinite loop.

16.464.1 Detailed Description

[L4](#) compiler related defines.

Definition in file [compiler.h](#).

16.465 compiler.h

[Go to the documentation of this file.](#)

```

00001 /*****
00007 */
00008 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009 *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00010 *      Frank Mehnert <fm3@os.inf.tu-dresden.de>,
00011 *      Jork Löser <jork@os.inf.tu-dresden.de>,
00012 *      Ronald Aigner <ra3@os.inf.tu-dresden.de>
00013 *      economic rights: Technische Universität Dresden (Germany)
00014 *
00015 * License: see LICENSE.spdx (in this directory or the directories above)
00016 */
00017 /*****
00018 #ifndef __L4_COMPILER_H__
00019 #define __L4_COMPILER_H__
00020
00021 #if !defined(__ASSEMBLY__) && !defined(__ASSEMBLER__)
00022
00029
00034 #ifndef L4_INLINE
00035 #ifndef __cplusplus
00036 #   ifdef __OPTIMIZE__
00037 #       define L4_INLINE_STATIC static __inline__
00038 #       define L4_INLINE_EXTERN extern __inline__
00039 #       ifdef __GNUC_STDC_INLINE__
00040 #           define L4_INLINE L4_INLINE_STATIC
00041 #       else
00042 #           define L4_INLINE L4_INLINE_EXTERN
00043 #       endif
00044 #   else /* !__OPTIMIZE__ */
00045 #       define L4_INLINE static
00046 #   endif /* !__OPTIMIZE__ */
00047 #else /* __cplusplus */
00048 #   define L4_INLINE inline
00049 #endif /* __cplusplus */
00050 #elif defined DOXYGEN
00051 #   define L4_INLINE inline
00052 #endif /* L4_INLINE */
00053
00058 #ifdef __OPTIMIZE__
00059 #define L4_ALWAYS_INLINE L4_INLINE __attribute__((__always_inline__))
00060 #elif defined(__cplusplus)
00061 #define L4_ALWAYS_INLINE inline __attribute__((__always_inline__))
00062 #else
00063 #define L4_ALWAYS_INLINE static inline __attribute__((__always_inline__))
00064 #endif
00065
00066
00067 #define L4_DECLARE_CONSTRUCTOR(func, prio) \
00068     static inline __attribute__((constructor(prio))) void func ## _ctor_func(void) { func(); }
00069
00070
00081
00104
00130
00153 #ifndef __cplusplus
00154 #   define L4_NOTHROW_A      __attribute__((nothrow))
00155 #   define L4_NOTHROW
00156 #   ifndef __BEGIN_DECLS
00157 #       define __BEGIN_DECLS
00158 #   endif
00159 #   ifndef __END_DECLS
00160 #       define __END_DECLS
00161 #   endif
00162 #   define L4_BEGIN_DECLS
00163 #   define L4_END_DECLS
00164 #   define L4_DEFAULT_PARAM(x)
00165 #else /* __cplusplus */
00166 #   if __cplusplus >= 201103L
00167 #       define L4_NOTHROW noexcept

```

```

00168 # else /* C++ < 11 */
00169 #     define L4_NOTHROW throw()
00170 # endif
00171 # define L4_BEGIN_DECLS extern "C" {
00172 #     define L4_END_DECLS }
00173 # if !defined __BEGIN_DECLS || defined DOXYGEN
00174 #     define __BEGIN_DECLS extern "C" {
00175 #     endif
00176 #     if !defined __END_DECLS || defined DOXYGEN
00177 #         define __END_DECLS }
00178 #     endif
00179 #     define L4_DEFAULT_PARAM(x) = x
00180 #endif /* __cplusplus */
00181
00182 /* Deprecation hints during compile -- remove later (2025+) */
00183 #ifndef EXTERN_C
00184 #define EXTERN_C DO_NOT_USE_EXTERN_C_ANY_MORE
00185 #endif
00186 #ifndef EXTERN_C_BEGIN
00187 #define EXTERN_C_BEGIN DO_NOT_USE_EXTERN_C_BEGIN_ANY_MORE__USE_L4_BEGIN_DECLS
00188 #endif
00189 #ifndef EXTERN_C_END
00190 #define EXTERN_C_END DO_NOT_USE_EXTERN_C_END_ANY_MORE__USE_L4_END_DECLS
00191 #endif
00192
00197 #if defined __cplusplus && __cplusplus >= 201402L
00198 # define L4_CONSTEXPR constexpr
00199 #else
00200 # define L4_CONSTEXPR
00201 #endif
00202
00207 #define L4_NORETURN __attribute__((noreturn))
00208
00209 #define L4_PURE __attribute__((pure))
00210
00215 #define L4_NOINSTRUMENT __attribute__((no_instrument_function))
00216 #ifndef L4_HIDDEN
00217 # define L4_HIDDEN __attribute__((visibility("hidden")))
00218 #endif
00219 #if !defined L4_EXPORT || defined DOXYGEN
00220 # define L4_EXPORT __attribute__((visibility("default")))
00221 #endif
00222 #ifndef L4_EXPORT_TYPE
00223 #     ifdef __cplusplus
00224 #         define L4_EXPORT_TYPE __attribute__((visibility("default")))
00225 #     else
00226 #         define L4_EXPORT_TYPE
00227 #     endif
00228 #endif
00229 #define L4_STRONG_ALIAS(name, aliasname) L4__STRONG_ALIAS(name, aliasname)
00230 #define L4_WEAK_ALIAS(name, aliasname) L4__WEAK_ALIAS(name, aliasname)
00231 #ifdef __clang__
00232 #define L4__STRONG_ALIAS(name, aliasname) \
00233     extern __typeof (name) aliasname __attribute__((alias (#name)));
00234 #define L4__WEAK_ALIAS(name, aliasname) \
00235     extern __typeof (name) aliasname __attribute__((weak, alias (#name)));
00236 #else
00237 #define L4__STRONG_ALIAS(name, aliasname) \
00238     extern __typeof (name) aliasname __attribute__((alias (#name), copy(name)));
00239 #define L4__WEAK_ALIAS(name, aliasname) \
00240     extern __typeof (name) aliasname __attribute__((weak, alias (#name), copy(name)));
00241 #endif
00242
00250 #if defined(__i386__) || defined(__amd64__) || \
00251     defined(__arm__) || defined(__aarch64__) || \
00252     defined(__mips__) || defined(__riscv__) || \
00253     defined(__powerpc__) || defined(__sparc__)
00254 # define L4_STACK_ALIGN __BIGGEST_ALIGNMENT__
00255 #else
00256 # error Define L4_STACK_ALIGN for this target!
00257 #endif
00258
00276 #if defined(__i386__) || defined(__amd64__)
00277 L4_INLINE unsigned long l4_align_stack_for_direct_fncall(unsigned long stack)
00278 {
00279     if ((stack & (L4_STACK_ALIGN - 1)) == (L4_STACK_ALIGN - sizeof(unsigned long)))
00280         return stack;
00281     return (stack & ~(L4_STACK_ALIGN)) - sizeof(unsigned long);
00282 }
00283 #else
00284 L4_INLINE unsigned long l4_align_stack_for_direct_fncall(unsigned long stack)
00285 {
00286     return stack & ~(L4_STACK_ALIGN);
00287 }
00288 #endif
00289
00290 #endif /* !__ASSEMBLY__ */

```

```

00291
00292 #include <l4/sys/linkage.h>
00293
00294 #define L4_LIKELY(x)    __builtin_expect((x),1)
00295 #define L4_UNLIKELY(x)  __builtin_expect((x),0)
00296
00297 /* Make sure that the function is not removed by optimization. Without the
00298  * "used" attribute, unreferenced static functions are removed. */
00299 #define L4_STICKY(x)     __attribute__((used)) x
00300 #define L4_DEPRECATED(s) __attribute__((deprecated(s)))
00301
00302 #ifndef static_assert
00303 # if !defined(__cplusplus)
00304 #  define static_assert _Static_assert
00305 # elif __cplusplus < 201103L
00306 #  define static_assert(x, y) \
00307    extern int l4_static_assert[-(!x)] __attribute__((unused))
00308 # endif
00309 #endif
00310
00311 #define L4_stringify_helper(x) #x
00312 #define L4_stringify(x)       L4_stringify_helper(x)
00313
00314 #ifdef __has_builtin
00315 #define L4_HAS_BUILTIN(def) __has_builtin(def)
00316 #else
00317 #define L4_HAS_BUILTIN(def) 0
00318 #endif
00319
00320 #ifndef __ASSEMBLER__
00324 L4_INLINE void l4_barrier(void);
00325
00329 L4_INLINE void l4_mb(void);
00330
00334 L4_INLINE void l4_wmb(void);
00335
00339 L4_INLINE L4_NORETURN void l4_infinite_loop(void);
00340
00341
00342 /* Implementations */
00343 L4_INLINE void l4_barrier(void)
00344 {
00345     __asm__ __volatile__ ("": : : "memory");
00346 }
00347
00348 L4_INLINE void l4_mb(void)
00349 {
00350     __asm__ __volatile__ ("": : : "memory");
00351 }
00352
00353 L4_INLINE void l4_wmb(void)
00354 {
00355     __asm__ __volatile__ ("": : : "memory");
00356 }
00357
00358 L4_INLINE L4_NORETURN void l4_infinite_loop(void)
00359 {
00360     while (1)
00361         l4_barrier();
00362 }
00363 #endif
00364
00366
00367 #endif /* !__L4_COMPILER_H__ */

```

16.466 amd64/l4/sys/arch/consts.h File Reference

Common [L4](#) constants, AMD64 version.

Macros

- `#define L4_PAGESHIFT 12`
Size of a page, log2-based.
- `#define L4_SUPERPAGESHIFT 21`
Size of a large page, log2-based.

16.466.1 Detailed Description

Common [L4](#) constants, AMD64 version.

Definition in file [consts.h](#).

16.467 consts.h

[Go to the documentation of this file.](#)

```

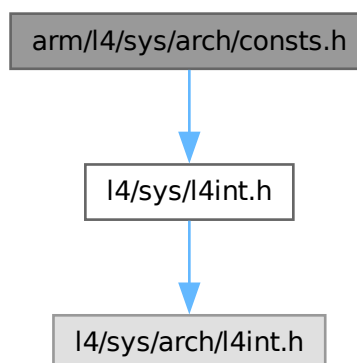
00001 /*****
00007 */
00008 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009 *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00010 *      Björn Döbel <doebel@os.inf.tu-dresden.de>,
00011 *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00012 *      economic rights: Technische Universität Dresden (Germany)
00013 *
00014 * License: see LICENSE.spdx (in this directory or the directories above)
00015 */
00016 /*****
00017 #ifndef __L4SYS__INCLUDE__ARCH_AMD64__CONSTS_H__
00018 #define __L4SYS__INCLUDE__ARCH_AMD64__CONSTS_H__
00019
00024 #define L4_PAGESHIFT    12
00025
00030 #define L4_SUPERPAGESHIFT 21
00031
00032 #endif /* ! __L4SYS__INCLUDE__ARCH_AMD64__CONSTS_H__ */

```

16.468 arm/l4/sys/arch/consts.h File Reference

Common [L4](#) constants, arm version.

#include <l4/sys/l4int.h>
 Include dependency graph for consts.h:



Macros

- `#define L4_PAGESHIFT 12`
Size of a page, log2-based.
- `#define L4_SUPERPAGESHIFT 21`
Size of a large page, log2-based.

16.468.1 Detailed Description

Common [L4](#) constants, arm version.

Definition in file [consts.h](#).

16.469 consts.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *           Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *           Björn Döbel <doebel@os.inf.tu-dresden.de>
00010  *           economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014 #ifndef _L4_SYS_CONSTS_H
00015 #define _L4_SYS_CONSTS_H
00016
00017 /* L4 includes */
00018 #include <l4/sys/l4int.h>
00026 #define L4_PAGESHIFT 12
00027
00031 #define L4_SUPERPAGESHIFT 21
00032
00034
00035 #endif /* !_L4_SYS_CONSTS_H */

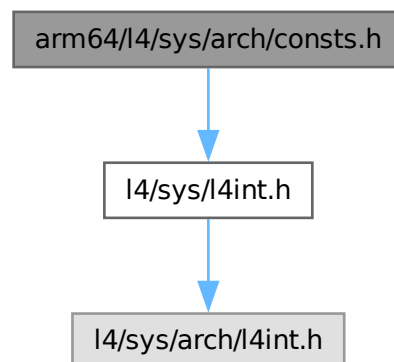
```

16.470 arm64/l4/sys/arch/consts.h File Reference

Common [L4](#) constants, arm version.

`#include <l4/sys/l4int.h>`

Include dependency graph for consts.h:



Macros

- `#define L4_PAGESHIFT 12`
Size of a page, log2-based.
- `#define L4_SUPERPAGESHIFT 21`
Size of a large page, log2-based.

16.470.1 Detailed Description

Common [L4](#) constants, arm version.

Definition in file [consts.h](#).

16.471 consts.h

[Go to the documentation of this file.](#)

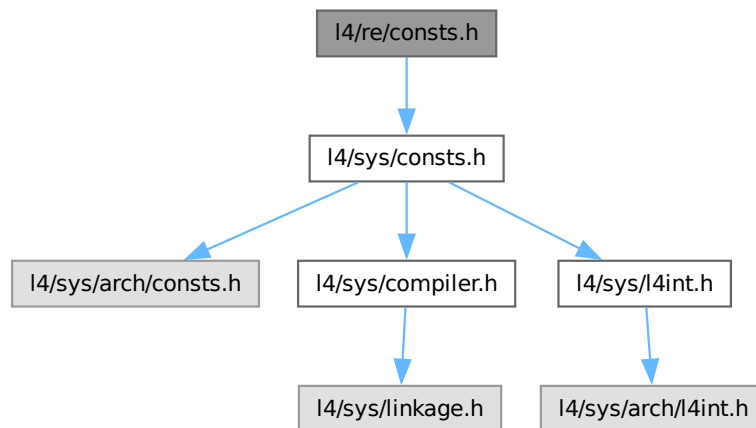
```
00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *      Björn Döbel <doebel@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014 #ifndef _L4_SYS_CONSTS_H
00015 #define _L4_SYS_CONSTS_H
00016
00017 /* L4 includes */
00018 #include <l4/sys/l4int.h>
00026 #define L4_PAGESHIFT    12
00027
00031 #define L4_SUPERPAGESHIFT 21
00032
00034
00035 #endif /* !_L4_SYS_CONSTS_H */
```

16.472 l4/re/consts.h File Reference

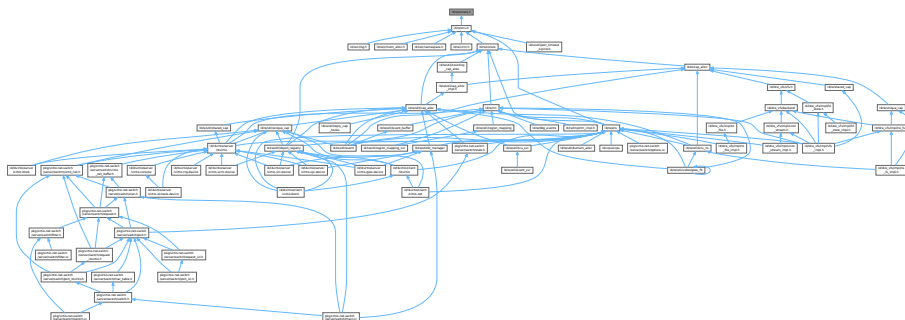
Constants.

```
#include <l4/sys/consts.h>
```

Include dependency graph for consts.h:



This graph shows which files directly or indirectly include this file:



Enumerations

- `enum`
Defaults for local thread priorities.

16.472.1 Detailed Description

Constants.

Definition in file [consts.h](#).

16.472.2 Enumeration Type Documentation

16.472.2.1 anonymous enum

anonymous enum

Defaults for local thread priorities.

Priorities are to be seen as local. These are used by the loader and libpthread. They are to be understood as 'local', which means the actual priority of the thread (as seen by the kernel) is the base priority as defined by the scheduler plus the local priority.

Definition at line 28 of file [consts.h](#).

16.473 consts.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011 #pragma once
00012
00013 #include <l4/sys/consts.h>
00014
00015 enum
00016 {
00017     L4RE_THIS_TASK_CAP = 1UL « L4_CAP_SHIFT,
00018 };
00019
00020 enum
00021 {
00030     L4RE_MAIN_THREAD_PRIO = 2, /* Priority of the main thread */
00031 };
00032

```

16.474 l4/sys/consts.h File Reference

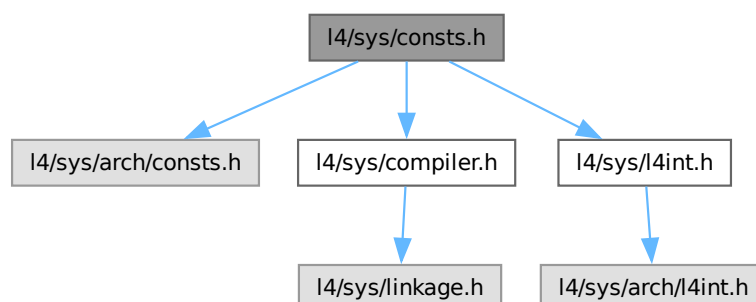
Common constants.

```
#include <l4/sys/arch/consts.h>
```

```
#include <l4/sys/compiler.h>
```

```
#include <l4/sys/l4int.h>
```

Include dependency graph for consts.h:



This graph shows which files directly or indirectly include this file:



Macros

- **#define L4_SYSF_NONE**
Capability selector flags.
- **#define L4_SYSF_SEND**
Send-phase flag.
- **#define L4_SYSF_RECV**
Receive-phase flag.
- **#define L4_SYSF_OPEN_WAIT**
Open-wait flag.
- **#define L4_SYSF_REPLY**
Reply flag.
- **#define L4_SYSF_CALL**
Call flags (combines send and receive).
- **#define L4_SYSF_WAIT**
Wait flags (combines receive and open wait).
- **#define L4_SYSF_SEND_AND_WAIT**
Send-and-wait flags.
- **#define L4_SYSF_REPLY_AND_WAIT**
Reply-and-wait flags.
- **#define L4_CAP_SHIFT**
Capability index shift.
- **#define L4_CAP_SIZE** (1UL << L4_CAP_SHIFT)
- **#define L4_CAP_OFFSET**
Offset of two consecutive capability selectors.
- **#define L4_CAP_MASK**
Mask to get only the relevant bits of an `l4_cap_idx_t`.
- **#define L4_INVALID_CAP**
Invalid capability selector.
- **#define L4_REPLY_CAP_BIT**
Mark this capability selector as index into the reply cap space instead of the regular capability space.
- **#define L4_INVALID_REPLY_CAP**
Invalid reply capability selector.
- **#define L4_PAGESIZE**
Minimal page size (in bytes).
- **#define L4_PAGEMASK**
Mask for the page number.
- **#define L4_LOG2_PAGESIZE**
Number of bits used for page offset.
- **#define L4_SUPERPAGESIZE**
Size of a large page.
- **#define L4_SUPERPAGEMASK**
Mask for the number of a large page.
- **#define L4_LOG2_SUPERPAGESIZE**
Number of bits used as offset for a large page.
- **#define L4_INVALID_PTR** ((void *)L4_INVALID_ADDR)
Invalid address as pointer type.

Enumerations

- enum [l4_unmap_flags_t](#) { [L4_FP_ALL_SPACES](#) , [L4_FP_DELETE_OBJ](#) , [L4_FP_OTHER_SPACES](#) }
Flags for the unmap operation.
- enum [l4_msg_item_consts_t](#) {
[L4_ITEM_MAP](#) = 8 , [L4_ITEM_CONT](#) = 1 , [L4_MAP_ITEM_GRANT](#) = 2 , [L4_MAP_ITEM_MAP](#) = 0 ,
[L4_RCV_ITEM_FORWARD_MAPPINGS](#) = 1 , [L4_RCV_ITEM_SINGLE_CAP](#) = [L4_ITEM_MAP](#) | 2 ,
[L4_RCV_ITEM_LOCAL_ID](#) = 4 }
Constants for message items.
- enum [l4_buffer_desc_consts_t](#) { [L4_BDR_MEM_SHIFT](#) = 0 , [L4_BDR_IO_SHIFT](#) = 5 , [L4_BDR_OBJ_SHIFT](#) = 10 , [L4_BDR_OFFSET_MASK](#) = (1UL << 20) - 1 }
Constants for buffer descriptors.
- enum [l4_default_caps_t](#) {
[L4_BASE_TASK_CAP](#) , [L4_BASE_FACTORY_CAP](#) , [L4_BASE_THREAD_CAP](#) , [L4_BASE_PAGER_CAP](#) ,
[L4_BASE_LOG_CAP](#) , [L4_BASE_ICU_CAP](#) , [L4_BASE_SCHEDULER_CAP](#) , [L4_BASE_IOMMU_CAP](#) ,
[L4_BASE_DEBUGGER_CAP](#) , [L4_BASE_ARM_SMCCC_CAP](#) , [L4_BASE_CAPS_LAST_P1](#) , [L4_BASE_CAPS_LAST](#)
= [L4_BASE_CAPS_LAST_P1](#) - 1 }
Default capabilities setup for the initial tasks.
- enum [l4_addr_consts_t](#) { [L4_INVALID_ADDR](#) = ~0UL }
Address related constants.

Functions

- [l4_addr_t l4_trunc_page](#) ([l4_addr_t](#) address) [L4_NOTHROW](#)
Round an address down to the next lower page boundary.
- [l4_addr_t l4_trunc_size](#) ([l4_addr_t](#) address, unsigned char bits) [L4_NOTHROW](#)
Round an address down to the next lower flexpage with size bits.
- [l4_addr_t l4_round_page](#) ([l4_addr_t](#) address) [L4_NOTHROW](#)
Round address up to the next page.
- [l4_addr_t l4_round_size](#) ([l4_addr_t](#) value, unsigned char bits) [L4_NOTHROW](#)
Round value up to the next alignment with bits size.
- unsigned [l4_bytes_to_mwords](#) (unsigned size) [L4_NOTHROW](#)
Determine how many machine words ([l4_umword_t](#)) are required to store a buffer of 'size' bytes.

16.474.1 Detailed Description

Common constants.

Definition in file [consts.h](#).

16.474.2 Macro Definition Documentation

16.474.2.1 L4_CAP_SIZE

```
#define L4_CAP_SIZE (1UL << L4\_CAP\_SHIFT)
```

Deprecated Superseded by [L4_CAP_OFFSET](#).

Definition at line 139 of file [consts.h](#).

16.474.2.2 L4_REPLY_CAP_BIT

```
#define L4_REPLY_CAP_BIT
```

Mark this capability selector as index into the reply cap space instead of the regular capability space.

Send phase: The sender wants to use an explicit reply cap slot from its reply cap space, instead of the otherwise used per-thread implicit reply cap slot.

Receive phase: The receiver wants to store the caller in an explicit reply cap slot from its reply cap space, instead of the otherwise used per-thread implicit reply cap slot.

Note

Only valid in combination with [L4_SYSF_REPLY](#), [L4_SYSF_WAIT](#) and [L4_SYSF_REPLY_AND_WAIT](#).

Definition at line [172](#) of file [consts.h](#).

16.474.2.3 L4_SYSF_CALL

```
#define L4_SYSF_CALL
```

Call flags (combines send and receive).

Combines [L4_SYSF_SEND](#) and [L4_SYSF_RECV](#).

Definition at line [107](#) of file [consts.h](#).

Referenced by [l4_ipc_call\(\)](#).

16.474.2.4 L4_SYSF_OPEN_WAIT

```
#define L4_SYSF_OPEN_WAIT
```

Open-wait flag.

This flag indicates that the receive operation (see [L4_SYSF_RECV](#)) shall be an *open wait*. *Open wait* means that the invoking thread shall wait for a message from any possible sender and *not* from the sender denoted by the capability.

Definition at line [90](#) of file [consts.h](#).

16.474.2.5 L4_SYSF_RECV

```
#define L4_SYSF_RECV
```

Receive-phase flag.

Setting this flag in a capability selector induces a receive phase, this means the invoking thread waits for a message from the object denoted by the capability. For a send phase see [L4_SYSF_SEND](#).

Definition at line [79](#) of file [consts.h](#).

Referenced by [l4_ipc_receive\(\)](#).

16.474.2.6 L4_SYSF_REPLY

```
#define L4_SYSF_REPLY
```

Reply flag.

This flag indicates that the send phase shall use the in-kernel reply capability instead of the capability denoted by the selector index.

Definition at line 99 of file [consts.h](#).

Referenced by [l4_ipc_reply\(\)](#).

16.474.2.7 L4_SYSF_REPLY_AND_WAIT

```
#define L4_SYSF_REPLY_AND_WAIT
```

Reply-and-wait flags.

Combines [L4_SYSF_SEND](#), [L4_SYSF_REPLY](#), and [L4_SYSF_WAIT](#).

Definition at line 131 of file [consts.h](#).

Referenced by [l4_ipc_reply_and_wait\(\)](#), and [L4::Server< LOOP_HOOKS >::reply_n_wait\(\)](#).

16.474.2.8 L4_SYSF_SEND

```
#define L4_SYSF_SEND
```

Send-phase flag.

Setting this flag in a capability selector induces a send phase, this means a message is sent to the object denoted by the capability. For receive phase see [L4_SYSF_RECV](#).

In [l4_vcpu_state_t::user_task](#) this flag means that the kernel has cached the user task capability internally, see [l4_thread_vcpu_resume_commit\(\)](#).

Definition at line 68 of file [consts.h](#).

Referenced by [l4_ipc_reply\(\)](#), and [l4_ipc_send\(\)](#).

16.474.2.9 L4_SYSF_SEND_AND_WAIT

```
#define L4_SYSF_SEND_AND_WAIT
```

Send-and-wait flags.

Combines [L4_SYSF_SEND](#) and [L4_SYSF_WAIT](#).

Definition at line 123 of file [consts.h](#).

Referenced by [l4_ipc_send_and_wait\(\)](#).

16.474.2.10 L4_SYSF_WAIT

```
#define L4_SYSF_WAIT
```

Wait flags (combines receive and open wait).

Combines [L4_SYSF_RECV](#) and [L4_SYSF_OPEN_WAIT](#).

Definition at line 115 of file [consts.h](#).

Referenced by [l4_ipc_wait\(\)](#), and [L4::Server< LOOP_HOOKS >::reply_n_wait\(\)](#).

16.475 consts.h

[Go to the documentation of this file.](#)

```
00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *      Björn Döbel <doebel@os.inf.tu-dresden.de>,
00010  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00011  *      economic rights: Technische Universität Dresden (Germany)
00012  *
00013  * License: see LICENSE.spdx (in this directory or the directories above)
00014  */
00015 #ifndef __L4_SYS__INCLUDE__CONSTS_H__
00016 #define __L4_SYS__INCLUDE__CONSTS_H__
00017
00018 #include <l4/sys/arch/consts.h>
00019 #include <l4/sys/compiler.h>
00020 #include <l4/sys/l4int.h>
00021
00055 #define L4_SYSF_NONE      0x00
00056
00068 #define L4_SYSF_SEND      0x01
00069
00079 #define L4_SYSF_RECV      0x02
00080
00090 #define L4_SYSF_OPEN_WAIT 0x04
00091
00099 #define L4_SYSF_REPLY     0x08
00100
00107 #define L4_SYSF_CALL      (L4_SYSF_SEND | L4_SYSF_RECV)
00108
00115 #define L4_SYSF_WAIT      (L4_SYSF_OPEN_WAIT | L4_SYSF_RECV)
00116
00123 #define L4_SYSF_SEND_AND_WAIT (L4_SYSF_OPEN_WAIT | L4_SYSF_CALL)
00124
00131 #define L4_SYSF_REPLY_AND_WAIT (L4_SYSF_WAIT | L4_SYSF_SEND | L4_SYSF_REPLY)
00132
00137 #define L4_CAP_SHIFT      12UL
00139 #define L4_CAP_SIZE        (1UL < L4_CAP_SHIFT)
00144 #define L4_CAP_OFFSET     (1UL < L4_CAP_SHIFT)
00150 #define L4_CAP_MASK        (~0UL < (L4_CAP_SHIFT - 2))
00152 #define L4_INVALID_CAP     (~0UL < (L4_CAP_SHIFT - 1))
00153
00154 #define L4_INVALID_CAP_BIT (1UL < (L4_CAP_SHIFT - 1))
00155
00172 #define L4_REPLY_CAP_BIT  (1UL < (L4_CAP_SHIFT - 2))
00173
00175 #define L4_INVALID_REPLY_CAP (~0UL < (L4_CAP_SHIFT - 2))
00176
00177 enum l4_sched_consts_t
00178 {
00179     L4_SCHED_MIN_PRIO = 1,
00180     L4_SCHED_MAX_PRIO = 255,
00181 };
00182
00188 enum l4_unmap_flags_t
00189 {
00202     L4_FP_ALL_SPACES    = 0x80000000UL,
00203
00216     L4_FP_DELETE_OBJ     = 0xc0000000UL,
00217
00224     L4_FP_OTHER_SPACES   = 0x0UL
```

```

00225 };
00226
00231 enum l4_msg_item_consts_t
00232 {
00233     L4_ITEM_MAP          = 8,
00234
00239     L4_ITEM_CONT        = 1,
00240
00241     // send
00264     L4_MAP_ITEM_GRANT    = 2,
00265
00266     L4_MAP_ITEM_MAP      = 0,
00267
00268     // receive
00279     L4_RCV_ITEM_FORWARD_MAPPINGS = 1,
00280
00294     L4_RCV_ITEM_SINGLE_CAP = L4_ITEM_MAP | 2,
00295
00315     L4_RCV_ITEM_LOCAL_ID   = 4,
00316 };
00317
00322 enum l4_buffer_desc_consts_t
00323 {
00324     L4_BDR_MEM_SHIFT      = 0,
00325     L4_BDR_IO_SHIFT       = 5,
00326     L4_BDR_OBJ_SHIFT      = 10,
00327     L4_BDR_OFFSET_MASK    = (1UL < 20) - 1,
00328 };
00329
00343 enum l4_default_caps_t
00344 {
00346     L4_BASE_TASK_CAP       = 1UL < L4_CAP_SHIFT,
00348     L4_BASE_FACTORY_CAP    = 2UL < L4_CAP_SHIFT,
00350     L4_BASE_THREAD_CAP     = 3UL < L4_CAP_SHIFT,
00358     L4_BASE_PAGER_CAP      = 4UL < L4_CAP_SHIFT,
00366     L4_BASE_LOG_CAP        = 5UL < L4_CAP_SHIFT,
00368     L4_BASE_ICU_CAP        = 6UL < L4_CAP_SHIFT,
00370     L4_BASE_SCHEDULER_CAP  = 7UL < L4_CAP_SHIFT,
00377     L4_BASE_IOMMU_CAP      = 8UL < L4_CAP_SHIFT,
00385     L4_BASE_DEBUGGER_CAP   = 10UL < L4_CAP_SHIFT,
00392     L4_BASE_ARM_SMCCC_CAP  = 11UL < L4_CAP_SHIFT,
00393
00395     L4_BASE_CAPS_LAST_P1,
00397     L4_BASE_CAPS_LAST = L4_BASE_CAPS_LAST_P1 - 1
00398 };
00399
00410 #define L4_PAGESIZE      (1UL < L4_PAGESHIFT)
00411
00419 #define L4_PAGEMASK      (~ (L4_PAGESIZE - 1))
00420
00428 #define L4_LOG2_PAGESIZE L4_PAGESHIFT
00429
00437 #define L4_SUPERPAGESIZE (1UL < L4_SUPERPAGESHIFT)
00438
00446 #define L4_SUPERPAGEMASK (~ (L4_SUPERPAGESIZE - 1))
00447
00454 #define L4_LOG2_SUPERPAGESIZE L4_SUPERPAGESHIFT
00455
00466 L4_INLINE l4_addr_t l4_trunc_page(l4_addr_t address) L4_NOTHROW;
00467 L4_INLINE l4_addr_t l4_trunc_page(l4_addr_t address) L4_NOTHROW
00468 { return address & L4_PAGEMASK; }
00469
00477 L4_INLINE l4_addr_t l4_trunc_size(l4_addr_t address, unsigned char bits) L4_NOTHROW;
00478 L4_INLINE l4_addr_t l4_trunc_size(l4_addr_t address, unsigned char bits) L4_NOTHROW
00479 { return address & (~0UL < bits); }
00480
00491 L4_INLINE l4_addr_t l4_round_page(l4_addr_t address) L4_NOTHROW;
00492 L4_INLINE l4_addr_t l4_round_page(l4_addr_t address) L4_NOTHROW
00493 { return (address + L4_PAGESIZE - 1) & L4_PAGEMASK; }
00494
00502 L4_INLINE l4_addr_t l4_round_size(l4_addr_t value, unsigned char bits) L4_NOTHROW;
00503 L4_INLINE l4_addr_t l4_round_size(l4_addr_t value, unsigned char bits) L4_NOTHROW
00504 { return (value + (1UL < bits) - 1) & (~0UL < bits); }
00505
00514 L4_INLINE unsigned l4_bytes_to_mwords(unsigned size) L4_NOTHROW;
00515 L4_INLINE unsigned l4_bytes_to_mwords(unsigned size) L4_NOTHROW
00516 { return (size + sizeof(l4_umword_t) - 1) / sizeof(l4_umword_t); }
00517
00522 enum l4_addr_consts_t {
00524     L4_INVALID_ADDR = ~0UL
00525 };
00526
00531 #define L4_INVALID_PTR ((void *)L4_INVALID_ADDR)
00532
00533 #ifndef NULL
00534 #ifndef __cplusplus
00535 # define NULL ((void *)0)

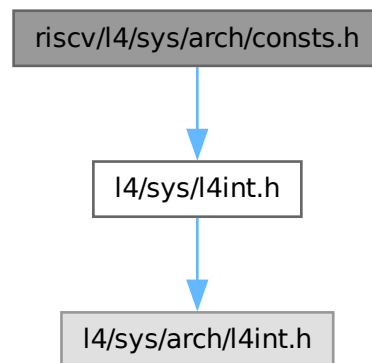
```

```
00539 #elif __cplusplus >= 201103L
00540 # define NULL nullptr
00541 #else
00542 # define NULL 0
00543 #endif
00544 #endif
00545
00546 #endif /* ! __L4_SYS__INCLUDE__CONSTS_H__ */
```

16.476 riscv/l4/sys/arch/consts.h File Reference

Common [L4](#) constants, RISC-V version.

```
#include <l4/sys/l4int.h>
Include dependency graph for consts.h:
```



Macros

- **#define L4_PAGESHIFT 12**
Sizeof a page in log2.
- **#define L4_SUPERPAGESHIFT 21**
Sizeof a large page in log2.

16.476.1 Detailed Description

Common [L4](#) constants, RISC-V version.

Definition in file [consts.h](#).

16.477 consts.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * Copyright (C) 2021, 2024 Kernkonzept GmbH.
00008  * Author(s): Georg Kotheimer <georg.kotheimer@kernkonzept.com>
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #ifndef __L4_SYS_CONSTS_H
00013 #define __L4_SYS_CONSTS_H
00014
00015 /* L4 includes */
00016 #include <l4/sys/l4int.h>
00017
00022 #define L4_PAGESHIFT 12 // 4K pages
00023
00028 #if __riscv_xlen == 32
00029 #define L4_SUPERPAGESHIFT 22
00030 #else
00031 #define L4_SUPERPAGESHIFT 21
00032 #endif
00033
00034 #endif /* !__L4_SYS_CONSTS_H */

```

16.478 x86/l4/sys/arch/consts.h File Reference

Common [L4](#) constants, x86 version.

Macros

- `#define L4_PAGESHIFT 12`
Size of a page log2-based.
- `#define L4_SUPERPAGESHIFT 22`
Size of a large page log2-based.

16.478.1 Detailed Description

Common [L4](#) constants, x86 version.

Definition in file [consts.h](#).

16.479 consts.h

[Go to the documentation of this file.](#)

```

00001 /*****
00007  */
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00010  *      Björn Döbel <doebel@os.inf.tu-dresden.de>,
00011  *      Lars Reuther <reuther@os.inf.tu-dresden.de>
00012  *      economic rights: Technische Universität Dresden (Germany)
00013  *
00014  * License: see LICENSE.spdx (in this directory or the directories above)
00015  */
00016 /*****
00017  */
00018 #ifndef __L4SYS__INCLUDE__ARCH_X86__CONSTS_H__
00019 #define __L4SYS__INCLUDE__ARCH_X86__CONSTS_H__
00019
00024 #define L4_PAGESHIFT 12
00025
00030 #define L4_SUPERPAGESHIFT 22
00031
00032 #endif /* !__L4SYS__INCLUDE__ARCH_X86__CONSTS_H__ */

```


16.480 capability.h

```

00001
00002 #pragma once
00003
00004 #include <l4/sys/consts.h>
00005 #include <l4/sys/types.h>
00006 #include <l4/sys/task.h>
00007
00008 namespace L4 {
00009
00010 class Task;
00011 class Kobject;
00012
00013 template< typename T > class L4_EXPORT Cap;
00014
00025 class L4_EXPORT Cap_base
00026 {
00027 public:
00029     enum No_init_type
00030     {
00034         No_init
00035     };
00036
00040     enum Cap_type
00041     {
00042         Invalid = L4_INVALID_CAP
00043     };
00044
00049     l4_cap_idx_t cap() const noexcept { return _c; }
00050
00057     bool is_valid() const noexcept { return !(_c & L4_INVALID_CAP_BIT); }
00058
00062     int invalid_cap_error() const noexcept { return _c & ~L4_INVALID_CAP_BIT; }
00063
00064     explicit operator bool () const noexcept
00065     { return !(_c & L4_INVALID_CAP_BIT); }
00066
00074     l4_fpage_t fpage(unsigned rights = L4_CAP_FPAGE_RWS) const noexcept
00075     { return l4_obj_fpage(_c, 0, rights); }
00076
00086     l4_umword_t snd_base(unsigned grant = L4_MAP_ITEM_MAP,
00087                          l4_cap_idx_t base = L4_INVALID_CAP) const noexcept
00088     {
00089         if (base == L4_INVALID_CAP)
00090             base = _c;
00091         return l4_map_obj_control(base, grant);
00092     }
00093
00094
00098     bool operator == (Cap_base const &o) const noexcept
00099     { return _c == o._c; }
00100
00104     bool operator != (Cap_base const &o) const noexcept
00105     { return _c != o._c; }
00106
00120     inline l4_msgtag_t validate(l4_utcb_t *u = l4_utcb()) const noexcept;
00121
00136     inline l4_msgtag_t validate(Cap<Task> task,
00137                                l4_utcb_t *u = l4_utcb()) const noexcept;
00138
00142     void invalidate() noexcept { _c = L4_INVALID_CAP; }
00143 protected:
00149     explicit Cap_base(l4_cap_idx_t c) noexcept : _c(c) {}
00153     explicit Cap_base(Cap_type cap) noexcept : _c(cap) {}
00154
00160     explicit Cap_base(l4_default_caps_t cap) noexcept : _c(cap) {}
00161
00165     explicit Cap_base() noexcept {}
00166
00176     void move(Cap_base const &src) const
00177     {
00178         if (!is_valid() || !src.is_valid())
00179             return;
00180
00181         l4_task_map(L4_BASE_TASK_CAP, L4_BASE_TASK_CAP, src.fpage(L4_CAP_FPAGE_RWS),
00182                    snd_base(L4_MAP_ITEM_GRANT) | L4_FPAGE_C_OBJ_RIGHTS);
00183     }
00184
00192     void copy(Cap_base const &src) const
00193     {
00194         if (!is_valid() || !src.is_valid())
00195             return;
00196
00197         l4_task_map(L4_BASE_TASK_CAP, L4_BASE_TASK_CAP, src.fpage(L4_CAP_FPAGE_RWS),
00198                    snd_base() | L4_FPAGE_C_OBJ_RIGHTS);

```

```

00199     }
00200
00203     l4_cap_idx_t _c;
00204 };
00205
00206
00222 template< typename T >
00223 class L4_EXPORT Cap : public Cap_base
00224 {
00225 private:
00226     friend class L4::Kobject;
00227
00239     explicit Cap(T const *p) noexcept
00240     : Cap_base(reinterpret_cast<l4_cap_idx_t>(p)) {}
00241
00242 public:
00243
00250     template< typename From >
00251     static void check_convertible_from() noexcept
00252     {
00253         using To = T;
00254         [[maybe_unused]] To* t = static_cast<From*>(nullptr);
00255     }
00256
00263     template< typename From >
00264     static void check_castable_from() noexcept
00265     {
00266         using To = T;
00267         [[maybe_unused]] To *t = static_cast<To *>(static_cast<From *>(nullptr));
00268     }
00269
00274     template< typename O >
00275     Cap(Cap<O> const &o) noexcept : Cap_base(o.cap())
00276     { check_convertible_from<O>(); }
00277
00282     Cap(Cap_type cap) noexcept : Cap_base(cap) {}
00283
00288     Cap(l4_default_caps_t cap) noexcept : Cap_base(cap) {}
00289
00294     explicit Cap(l4_cap_idx_t idx = L4_INVALID_CAP) noexcept : Cap_base(idx) {}
00295
00299     explicit Cap(No_init_type) noexcept {}
00300
00307     Cap move(Cap const &src) const
00308     {
00309         Cap_base::move(src);
00310         return *this;
00311     }
00312
00317     Cap copy(Cap const &src) const
00318     {
00319         Cap_base::copy(src);
00320         return *this;
00321     }
00322
00326     T *operator -> () const noexcept { return reinterpret_cast<T*>(_c); }
00327 };
00328
00329
00340 template<>
00341 class L4_EXPORT Cap<void> : public Cap_base
00342 {
00343 public:
00344
00345     explicit Cap(void const *p) noexcept
00346     : Cap_base(reinterpret_cast<l4_cap_idx_t>(p)) {}
00347
00351     Cap(Cap_type cap) noexcept : Cap_base(cap) {}
00352
00357     Cap(l4_default_caps_t cap) noexcept : Cap_base(cap) {}
00358
00363     explicit Cap(l4_cap_idx_t idx = L4_INVALID_CAP) noexcept : Cap_base(idx) {}
00364     explicit Cap(No_init_type) noexcept {}
00365
00366     template< typename From >
00367     static void check_convertible_from() noexcept {}
00368
00369     template< typename From >
00370     static void check_castable_from() noexcept {}
00371
00378     Cap move(Cap const &src) const
00379     {
00380         Cap_base::move(src);
00381         return *this;
00382     }
00383
00388     Cap copy(Cap const &src) const

```

```

00389 {
00390     Cap_base::copy(src);
00391     return *this;
00392 }
00393
00394 template< typename T >
00395 Cap(Cap<T> const &o) noexcept : Cap_base(o.cap()) {}
00396 };
00397
00414 template< typename T, typename F >
00415 inline
00416 Cap<T> cap_cast(Cap<F> const &c) noexcept
00417 {
00418     Cap<T>::template check_castable_from<F>();
00419     return Cap<T>(c.cap());
00420 }
00421
00422 // gracefully deal with L4::Kobject ambiguity
00423 template< typename T >
00424 inline
00425 Cap<T> cap_cast(Cap<L4::Kobject> const &c) noexcept
00426 {
00427     return Cap<T>(c.cap());
00428 }
00429
00445 template< typename T, typename F >
00446 inline
00447 Cap<T> cap_reinterpret_cast(Cap<F> const &c) noexcept
00448 {
00449     return Cap<T>(c.cap());
00450 }
00451
00458 class Reply_cap_idx
00459 {
00460     l4_cap_idx_t _i;
00461
00462 public:
00463     constexpr explicit Reply_cap_idx() noexcept : _i(L4_INVALID_REPLY_CAP) {}
00464     constexpr explicit Reply_cap_idx(l4_cap_idx_t i) noexcept
00465         : _i(i | L4_REPLY_CAP_BIT) {}
00466
00467     constexpr l4_cap_idx_t cap() const noexcept
00468     { return _i; }
00469
00470     explicit constexpr operator l4_cap_idx_t() const noexcept
00471     { return _i; }
00472
00473     explicit constexpr operator bool() const noexcept
00474     { return !(_i & L4_INVALID_CAP_BIT); }
00475
00476     constexpr bool operator==(Reply_cap_idx o) const noexcept
00477     { return _i == o._i; }
00478
00479     constexpr bool operator!=(Reply_cap_idx o) const noexcept
00480     { return _i != o._i; }
00481 };
00482
00483 class Reply_cap;
00484
00488 class Reply_cap_alloc
00489 {
00490     friend class Reply_cap;
00491
00492 public:
00493     constexpr Reply_cap_alloc() = default;
00494
00495     // Attention: do *not* define a destructor! We must keep the class trivially
00496     // destructible so that no global destructor is created for
00497     // L4Re::Util::reply_cap_alloc.
00498     // virtual ~Reply_cap_alloc() = default;
00499
00500     Reply_cap_alloc(Reply_cap_alloc const &) = delete;
00501     Reply_cap_alloc &operator = (Reply_cap_alloc const &) = delete;
00502
00508     virtual Reply_cap alloc() noexcept = 0;
00509
00510 protected:
00518     virtual void free(Reply_cap_idx cap) noexcept = 0;
00519 };
00520
00529 class Reply_cap
00530 {
00531 public:
00533     constexpr Reply_cap() noexcept = default;
00534
00544     constexpr Reply_cap(Reply_cap_idx cap, Reply_cap_alloc *alloc) noexcept
00545         : _cap(cap), _alloc(alloc)

```

```

00546 {}
00547
00554 ~Reply_cap()
00555 { reset(); }
00556
00557 // not copyable
00558 Reply_cap(Reply_cap const &) = delete;
00559 Reply_cap &operator=(Reply_cap const &) = delete;
00560
00561 constexpr Reply_cap(Reply_cap &&o) noexcept
00562 : _cap(o._cap), _alloc(o._alloc)
00563 {
00564     o._cap = Reply_cap_idx();
00565     o._alloc = nullptr;
00566 }
00567
00568 Reply_cap &operator=(Reply_cap &&o) noexcept
00569 {
00570     reset(o._cap, o._alloc);
00571     o._cap = Reply_cap_idx();
00572     o._alloc = nullptr;
00573     return *this;
00574 }
00575
00589 l4_ret_t reply(l4_msgtag_t tag, l4_utcb_t *utcb = l4_utcb()) noexcept
00590 {
00591     l4_umword_t err = L4_IPC_ENOT_EXISTENT;
00592     if (is_valid()) [[likely]]
00593     {
00594         err = l4_ipc_error(l4_ipc_reply(_cap.cap(), utcb, tag,
00595                                     L4_IPC_BOTH_TIMEOUT_0), utcb);
00596         _alloc->free(_cap);
00597         _cap = Reply_cap_idx();
00598     }
00599
00600     if (err) [[unlikely]]
00601         return l4_ipc_to_errno(err);
00602     else
00603         return 0;
00604 }
00605
00612 constexpr Reply_cap_idx get() const noexcept
00613 { return _cap; }
00614
00615 constexpr bool is_valid() const noexcept
00616 { return static_cast<bool>(_cap); }
00617
00618 explicit constexpr operator bool () const noexcept
00619 { return is_valid(); }
00620
00621 constexpr bool operator==(Reply_cap const &o) const noexcept
00622 { return _cap == o._cap; }
00623
00624 constexpr bool operator==(Reply_cap_idx o) const noexcept
00625 { return _cap == o; }
00626
00627 constexpr bool operator!=(Reply_cap const &o) const noexcept
00628 { return _cap != o._cap; }
00629
00630 constexpr bool operator!=(Reply_cap_idx o) const noexcept
00631 { return _cap != o; }
00632
00639 void reset(Reply_cap_idx cap = Reply_cap_idx(),
00640           Reply_cap_alloc *alloc = nullptr) noexcept
00641 {
00642     if (is_valid()) [[unlikely]]
00643     {
00644         // If the reply cap is still valid, reply to the caller that we've
00645         // dropped the reply. Should not happen in a well behaved server.
00646         auto tag = l4_msgtag(-L4_EDROPREPLY, 0, 0, 0);
00647         l4_ipc_reply(_cap.cap(), l4_utcb(), tag, L4_IPC_BOTH_TIMEOUT_0);
00648         _alloc->free(_cap);
00649     }
00650
00651     _cap = cap;
00652     _alloc = alloc;
00653 }
00654
00655 private:
00656 Reply_cap_idx _cap = Reply_cap_idx();
00657 Reply_cap_alloc *_alloc = nullptr;
00658 };
00659
00660 }

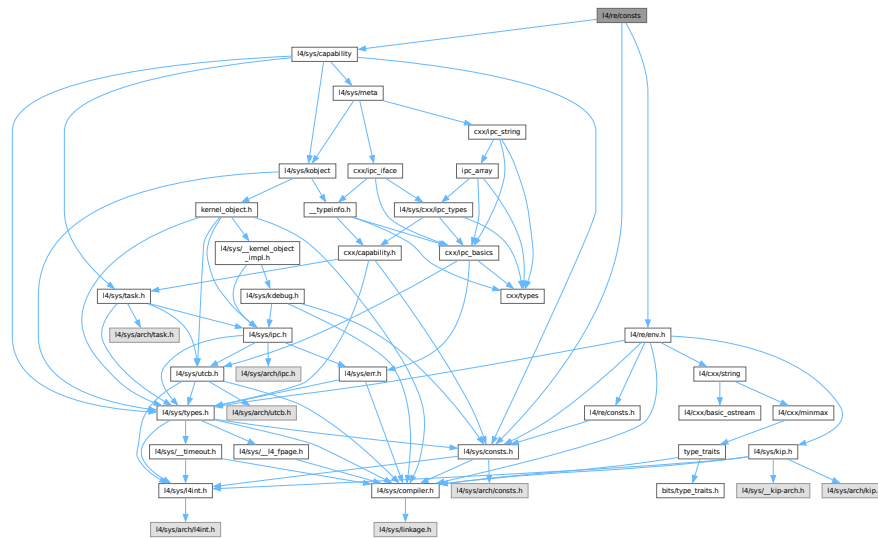
```

16.481 I4/re/consts File Reference

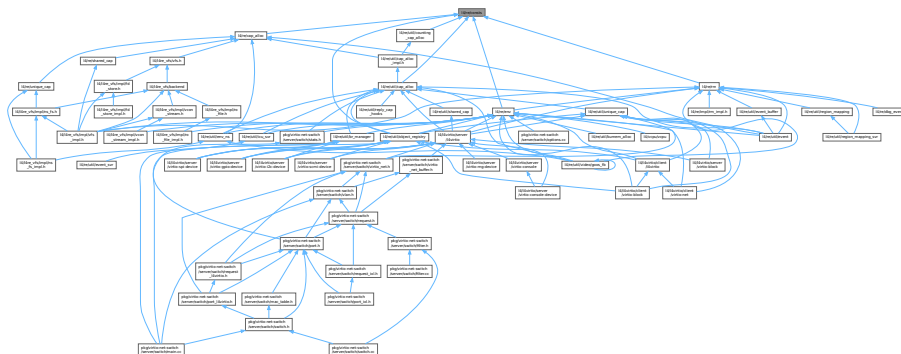
Constants.

```
#include <linux/sys/capability>
#include <linux/sys/consts.h>
#include <linux/re/env.h>
```

Include dependency graph for consts:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **L4Re**
L4Re C++ Interfaces.

16.481.1 Detailed Description

Constants.

Definition in file [consts](#).

16.482 consts

[Go to the documentation of this file.](#)

```
00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00003 /*
00004  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009 #pragma once
00010
00011 #include <l4/sys/capability>
00012 #include <l4/sys/consts.h>
00013 #include <l4/re/env.h>
00014
00015 namespace L4Re {
00016     static L4::Cap<L4::Task>::Cap_type const This_task
00017         = static_cast<L4::Cap<L4::Task>::Cap_type>(L4RE_THIS_TASK_CAP);
00018 }
```

16.483 consts

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002
00003 #pragma once
00004
00005 #include <l4/sys/consts.h>
00006
00007 namespace L4 {
00008
00009     template<typename T>
00010     constexpr T trunc_order(T val, unsigned char order)
00011     {
00012         return val & ((~T(0)) << order);
00013     }
00014
00015     template<typename T>
00016     constexpr T round_order(T val, unsigned char order)
00017     {
00018         return (val + (T(1) << order) - T(1)) & ((~T(0)) << order);
00019     }
00020
00021     template<typename T>
00022     constexpr T trunc_page(T val)
00023     {
00024         return trunc_order(val, L4_PAGESHIFT);
00025     }
00026
00027     template<typename T>
00028     constexpr T round_page(T val)
00029     {
00030         return round_order(val, L4_PAGESHIFT);
00031     }
00032
00033     template<typename T>
00034     inline unsigned char
00035     max_order(unsigned char order, T addr,
00036               T min_addr, T max_addr,
00037               T hotspot = T(0))
00038     {
00039         while (order < 30 /* limit to 1GB flexpages */)
00040         {
00041             T mask;
00042             T base = trunc_order(addr, order + 1);
00043             if (base < min_addr)
00044                 return order;
00045
00046             if (base + (T(1) << (order + 1)) - T(1) > max_addr - T(1))
00047                 return order;
00048
00049             mask = ~(~T(0) << (order + 1));
00050             if (hotspot == ~T(0) || ((addr ^ hotspot) & mask))
00051                 break;
00052
00053             ++order;
00054         }
00055         return order;
00056     }
```

```

00073     }
00074
00075 }

```

16.484 ipc_array

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include "types"
00010 #include "ipc_basics"
00011 #include "ipc_types"
00012
00013 namespace L4 { namespace Ipc L4_EXPORT {
00014
00015     using Array_len_default = unsigned short;
00016
00017     template<typename ELEM_TYPE, typename LEN_TYPE = Array_len_default >
00018     struct Array_ref
00019     {
00020         using ptr_type = ELEM_TYPE *;
00021         using len_type = LEN_TYPE;
00022
00023         len_type length;
00024         ptr_type data;
00025         Array_ref() = default;
00026         Array_ref(len_type length, ptr_type data)
00027             : length(length), data(data)
00028         {}
00029
00030         template<typename X> struct Non_const
00031         { using type = Array_ref<X, LEN_TYPE>; };
00032
00033         template<typename X> struct Non_const<X const>
00034         { using type = Array_ref<X, LEN_TYPE>; };
00035
00036         Array_ref(typename Non_const<ELEM_TYPE>::type const &other)
00037             : length(other.length), data(other.data)
00038         {}
00039
00040         Array_ref &operator = (typename Non_const<ELEM_TYPE>::type const &other)
00041         {
00042             this->length = other.length;
00043             this->data = other.data;
00044             return *this;
00045         }
00046     };
00047
00048     template<typename ELEM_TYPE, typename LEN_TYPE = Array_len_default>
00049     struct Array : Array_ref<ELEM_TYPE, LEN_TYPE>
00050     {
00051         Array() {}
00052         Array(LEN_TYPE length, ELEM_TYPE *data)
00053             : Array_ref<ELEM_TYPE, LEN_TYPE>(length, data)
00054         {}
00055
00056         template<typename X> struct Non_const
00057         { using type = Array<X, LEN_TYPE>; };
00058
00059         template<typename X> struct Non_const<X const>
00060         { using type = Array<X, LEN_TYPE>; };
00061
00062         Array(typename Non_const<ELEM_TYPE>::type const &other)
00063             : Array_ref<ELEM_TYPE, LEN_TYPE>(other.length, other.data)
00064         {}
00065
00066         Array &operator = (typename Non_const<ELEM_TYPE>::type const &other)
00067         {
00068             this->length = other.length;
00069             this->data = other.data;
00070             return *this;
00071         }
00072     };
00073
00074     template<typename ELEM_TYPE,
00075             typename LEN_TYPE = Array_len_default,
00076             LEN_TYPE MAX = (L4_UTCB_GENERIC_DATA_SIZE *
00077                             sizeof(l4_umword_t)) / sizeof(ELEM_TYPE) >

```

```

00126 struct Array_in_buf
00127 {
00128     using array = Array_ref<ELEM_TYPE, LEN_TYPE>;
00129     using const_array = Array_ref<ELEM_TYPE const, LEN_TYPE>;
00130
00132     ELEM_TYPE data[MAX];
00134     LEN_TYPE length;
00135
00137     void copy_in(const_array a)
00138     {
00139         length = a.length;
00140         if (length > MAX)
00141             length = MAX;
00142
00143         for (LEN_TYPE i = 0; i < length; ++i)
00144             data[i] = a.data[i];
00145     }
00146
00148     Array_in_buf(const_array a) { copy_in(a); }
00150     Array_in_buf(array a) { copy_in(a); }
00151 };
00152
00153 // implementation details for transmission
00154 namespace Msg {
00155
00157 template<typename A, typename LEN>
00158 struct Elem< Array<A, LEN> >
00159 {
00161     using arg_type = Array<A, LEN>;
00163     using svr_type = Array_ref<A, LEN>;
00164     using svr_arg_type = svr_type;
00165     enum { Is_optional = false };
00166 };
00167
00169 template<typename A, typename LEN>
00170 struct Elem< Array<A, LEN> & >
00171 {
00173     using arg_type = Array<A, LEN> &;
00175     using svr_type = Array_ref<A, LEN>;
00177     using svr_arg_type = svr_type &;
00178     enum { Is_optional = false };
00179 };
00180
00182 template<typename A, typename LEN>
00183 struct Elem< Array_ref<A, LEN> & >
00184 {
00186     using arg_type = Array_ref<A, LEN> &;
00188     using svr_type = Array_ref<L4::Types::Remove_const_t<A>, LEN>;
00190     using svr_arg_type = svr_type &;
00191     enum { Is_optional = false };
00192 };
00193
00194 template<typename A> struct Class<Array<A> > : Class<A>::type {};
00195 template<typename A> struct Class<Array_ref<A> > : Class<A>::type {};
00196
00197 namespace Detail {
00198
00199 template<typename A, typename LEN, typename ARRAY, bool REF>
00200 struct Clnt_val_ops_d_in : Clnt_noops<ARRAY>
00201 {
00202     using Clnt_noops<ARRAY>::to_msg;
00203     static int to_msg(char *msg, unsigned offset, unsigned limit,
00204                      ARRAY a, Dir_in, Cls_data)
00205     {
00206         offset = align_to<LEN>(offset);
00207         if (L4_UNLIKELY(!check_size<LEN>(offset, limit)))
00208             return -L4_MSGTOOLONG;
00209         *reinterpret_cast<LEN*>(msg + offset) = a.length;
00210         offset = align_to<A>(offset + sizeof(LEN));
00211         if (L4_UNLIKELY(!check_size<A>(offset, limit, a.length)))
00212             return -L4_MSGTOOLONG;
00213         using elem_type = L4::Types::Remove_const_t<A>;
00214         elem_type *data = reinterpret_cast<elem_type*>(msg + offset);
00215
00216         // we do not correctly handle overlaps
00217         if (!REF || data != a.data)
00218         {
00219             for (LEN i = 0; i < a.length; ++i)
00220                 data[i] = a.data[i];
00221         }
00222         return offset + a.length * sizeof(A);
00223     }
00224 };
00225 };
00226 } // namespace Detail
00227
00228 template<typename A, typename LEN>

```



```

00229 struct Clnt_val_ops<Array<A, LEN>, Dir_in, Cls_data> :
00230     Detail::Clnt_val_ops_d_in<A, LEN, Array<A, LEN>, false> {};
00231
00232 template<typename A, typename LEN>
00233 struct Clnt_val_ops<Array_ref<A, LEN>, Dir_in, Cls_data> :
00234     Detail::Clnt_val_ops_d_in<A, LEN, Array_ref<A, LEN>, true> {};
00235
00236 template<typename A, typename LEN, typename CLASS>
00237 struct Svr_val_ops<Array_ref<A, LEN>, Dir_in, CLASS> >
00238 : Svr_noops<Array_ref<A, LEN> >
00239 {
00240     using svr_type = Array_ref<A, LEN>;
00241
00242     using Svr_noops<svr_type>::to_svr;
00243     static int to_svr(char *msg, unsigned offset, unsigned limit,
00244         svr_type &a, Dir_in, Cls_data)
00245     {
00246         offset = align_to<LEN>(offset);
00247         if (L4_UNLIKELY(!check_size<LEN>(offset, limit)))
00248             return -L4_EMMSGTOOSHORT;
00249         a.length = *reinterpret_cast<LEN*>(msg + offset);
00250         offset = align_to<A>(offset + sizeof(LEN));
00251         if (L4_UNLIKELY(!check_size<A>(offset, limit, a.length)))
00252             return -L4_EMMSGTOOSHORT;
00253         a.data = reinterpret_cast<A*>(msg + offset);
00254         return offset + a.length * sizeof(A);
00255     }
00256 };
00257
00258 template<typename A, typename LEN>
00259 struct Svr_xmit<Array<A, LEN> > : Svr_xmit<Array_ref<A, LEN> > {};
00260
00261 template<typename A, typename LEN>
00262 struct Clnt_val_ops<Array<A, LEN>, Dir_out, Cls_data> : Clnt_noops<Array<A, LEN> >
00263 {
00264     using type = Array<A, LEN>;
00265
00266     using Clnt_noops<type>::from_msg;
00267     static int from_msg(char *msg, unsigned offset, unsigned limit, long,
00268         type &a, Dir_out, Cls_data)
00269     {
00270         offset = align_to<LEN>(offset);
00271         if (L4_UNLIKELY(!check_size<LEN>(offset, limit)))
00272             return -L4_EMMSGTOOSHORT;
00273
00274         LEN l = *reinterpret_cast<LEN*>(msg + offset);
00275
00276         offset = align_to<A>(offset + sizeof(LEN));
00277         if (L4_UNLIKELY(!check_size<A>(offset, limit, l)))
00278             return -L4_EMMSGTOOSHORT;
00279
00280         A *data = reinterpret_cast<A*>(msg + offset);
00281
00282         if (l > a.length)
00283             l = a.length;
00284         else
00285             a.length = l;
00286
00287         for (unsigned i = 0; i < l; ++i)
00288             a.data[i] = data[i];
00289
00290         return offset + l * sizeof(A);
00291     };
00292 };
00293
00294 template<typename A, typename LEN>
00295 struct Clnt_val_ops<Array_ref<A, LEN>, Dir_out, Cls_data> :
00296     Clnt_noops<Array_ref<A, LEN> >
00297 {
00298     using type = Array_ref<A, LEN>;
00299
00300     using Clnt_noops<type>::from_msg;
00301     static int from_msg(char *msg, unsigned offset, unsigned limit, long,
00302         type &a, Dir_out, Cls_data)
00303     {
00304         offset = align_to<LEN>(offset);
00305         if (L4_UNLIKELY(!check_size<LEN>(offset, limit)))
00306             return -L4_EMMSGTOOSHORT;
00307
00308         LEN l = *reinterpret_cast<LEN*>(msg + offset);
00309
00310         offset = align_to<A>(offset + sizeof(LEN));
00311         if (L4_UNLIKELY(!check_size<A>(offset, limit, l)))
00312             return -L4_EMMSGTOOSHORT;
00313
00314         a.data = reinterpret_cast<A*>(msg + offset);
00315         a.length = l;

```

```

00316     return offset + 1 * sizeof(A);
00317 };
00318 };
00319
00320 template<typename A, typename LEN, typename CLASS>
00321 struct Svr_val_ops<Array_ref<A, LEN>, Dir_out, CLASS> :
00322     Svr_noops<Array_ref<L4::Types::Remove_const_t<A>, LEN>
00323 {
00324     using elem_type = L4::Types::Remove_const_t<A>;
00325     using svr_type = Array_ref<elem_type, LEN>;
00326
00327     using Svr_noops<svr_type>::to_svr;
00328     static int to_svr(char *msg, unsigned offset, unsigned limit,
00329         svr_type &a, Dir_out, Cls_data)
00330     {
00331         offset = align_to<LEN>(offset);
00332         if (L4_UNLIKELY(!check_size<LEN>(offset, limit)))
00333             return -L4_MSGTOOLONG;
00334
00335         offset = align_to<A>(offset + sizeof(LEN));
00336         a.data = reinterpret_cast<elem_type *>(msg + offset);
00337         a.length = (limit - offset) / sizeof(A);
00338         return offset;
00339     }
00340
00341     using Svr_noops<svr_type>::from_svr;
00342     static int from_svr(char *msg, unsigned offset, unsigned limit, long,
00343         svr_type a, Dir_out, Cls_data)
00344     {
00345         offset = align_to<LEN>(offset);
00346         if (L4_UNLIKELY(!check_size<LEN>(offset, limit)))
00347             return -L4_MSGTOOLONG;
00348
00349         *reinterpret_cast<LEN *>(msg + offset) = a.length;
00350
00351         offset = align_to<A>(offset + sizeof(LEN));
00352         if (L4_UNLIKELY(!check_size<A>(offset, limit, a.length)))
00353             return -L4_MSGTOOLONG;
00354
00355         // In case of deferred replies, the Array_ref is pointing to a server
00356         // provided buffer instead of the UTCB.
00357         elem_type *data = reinterpret_cast<elem_type *>(msg + offset);
00358         if (data != a.data)
00359             for (LEN i = 0; i < a.length; ++i)
00360                 data[i] = a.data[i];
00361
00362         return offset + a.length * sizeof(A);
00363     }
00364 };
00365
00366 template<typename A, typename LEN>
00367 struct Svr_xmit<Array<A, LEN> &> : Svr_xmit<Array_ref<A, LEN> &> {};
00368
00369 // Pointer to array is not implemented.
00370 template<typename A, typename LEN>
00371 struct Is_valid_rpc_type<Array_ref<A, LEN> *> : L4::Types::False {};
00372
00373 // Optional input arrays are not implemented.
00374 template<typename A, typename LEN>
00375 struct Is_valid_rpc_type<Opt<Array_ref<A, LEN> > > : L4::Types::False {};
00376
00377 } // namespace Msg
00378
00379 }

```

16.485 ipc_basics

```

00001 // vi:set ft=c++ -- Mode: C++ --
00002 /*
00003  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include "types"
00010 #include <l4/sys/utcb.h>
00011 #include <l4/sys/err.h>
00012
00013 namespace L4 {
00014
00015 namespace Ipc {
00016

```

```

00019 namespace Msg {
00020
00021 using L4::Types::True;
00022 using L4::Types::False;
00023
00030 constexpr unsigned long align_to(unsigned long bytes, unsigned long align) noexcept
00031 { return (bytes + align - 1) & ~(align - 1); }
00032
00039 template<typename T>
00040 constexpr unsigned long align_to(unsigned long bytes) noexcept
00041 { return align_to(bytes, __alignof(T)); }
00042
00052 template<typename T>
00053 constexpr bool check_size(unsigned offset, unsigned limit) noexcept
00054 {
00055     return offset + sizeof(T) <= limit;
00056 }
00057
00070 template<typename T, typename CTYPE>
00071 inline bool check_size(unsigned offset, unsigned limit, CTYPE cnt) noexcept
00072 {
00073     if (L4_UNLIKELY(sizeof(CTYPE) <= sizeof(unsigned) &&
00074         ~0U / sizeof(T) <= static_cast<unsigned>(cnt)))
00075         return false;
00076
00077     if (L4_UNLIKELY(sizeof(CTYPE) > sizeof(unsigned) &&
00078         static_cast<CTYPE>(~0U / sizeof(T)) <= cnt))
00079         return false;
00080
00081     return sizeof(T) * cnt <= limit - offset;
00082 }
00083
00084
00085 enum
00086 {
00088     Word_bytes = sizeof(l4_umword_t),
00090     Item_words = 2,
00092     Item_bytes = Word_bytes * Item_words,
00094     Mr_words = L4_UTCB_GENERIC_DATA_SIZE,
00096     Mr_bytes = Word_bytes * Mr_words,
00098     Br_bytes = Word_bytes * L4_UTCB_GENERIC_BUFFERS_SIZE,
00099 };
00100
00101
00113 template<typename T>
00114 inline int msg_add(char *msg, unsigned offs, unsigned limit, T v) noexcept
00115 {
00116     offs = align_to<T>(offs);
00117     if (L4_UNLIKELY(!check_size<T>(offs, limit)))
00118         return -L4_EMMSGTOOLONG;
00119     *reinterpret_cast<L4::Types::Remove_const_t<T>*>(msg + offs) = v;
00120     return offs + sizeof(T);
00121 }
00122
00134 template<typename T>
00135 inline int msg_get(char *msg, unsigned offs, unsigned limit, T &v) noexcept
00136 {
00137     offs = align_to<T>(offs);
00138     if (L4_UNLIKELY(!check_size<T>(offs, limit)))
00139         return -L4_EMMSGTOOSHORT;
00140     v = *reinterpret_cast<T*>(msg + offs);
00141     return offs + sizeof(T);
00142 }
00143
00145 struct Dir_in { using type = Dir_in; using dir = Dir_in; };
00147 struct Dir_out { using type = Dir_out; using dir = Dir_out; };
00148
00150 struct Cls_data { using type = Cls_data; using cls = Cls_data; };
00152 struct Cls_item { using type = Cls_item; using cls = Cls_item; };
00154 struct Cls_buffer { using type = Cls_buffer; using cls = Cls_buffer; };
00155
00156 // Typical combinations
00158 struct Do_in_data : Dir_in, Cls_data {};
00160 struct Do_out_data : Dir_out, Cls_data {};
00162 struct Do_in_items : Dir_in, Cls_item {};
00164 struct Do_out_items : Dir_out, Cls_item {};
00166 struct Do_rcv_buffers : Dir_in, Cls_buffer {};
00167
00168 // implementation details
00169 namespace Detail {
00170
00171 template<typename T> struct _Plain
00172 {
00173     using type = T;
00174     static T deref(T v) noexcept { return v; }
00175 };
00176

```

```

00177 template<typename T> struct _Plain<T *>
00178 {
00179     using type = T;
00180     static T &deref(T *v) noexcept { return *v; }
00181 };
00182
00183 template<typename T> struct _Plain<T &>
00184 {
00185     using type = T;
00186     static T &deref(T &v) noexcept { return v; }
00187 };
00188
00189
00190 template<typename T> struct _Plain<T const &>
00191 {
00192     using type = T;
00193     static T const &deref(T const &v) noexcept { return v; }
00194 };
00195
00196 template<typename T> struct _Plain<T const *>
00197 {
00198     using type = T;
00199     static T const &deref(T const *v) noexcept { return *v; }
00200 };
00201
00202
00210 template<typename MTYPE, typename DIR, typename CLASS> struct Clnt_val_ops;
00211
00212 // Provides empty default implementations for the Dir/Cls combinations in which
00213 // the type does not participate (for the Do_in_items, Do_in_data, ... passes in
00214 // Rpc_inline_call::call).
00215 //
00216 // The Clnt_val_ops specializations derive from Clnt_noops, and pull in the
00217 // empty default implementations with 'using Clnt_noops::{to,from}_msg', to
00218 // handle the other Dir/Cls combinations.
00219 template<typename T> struct Clnt_noops
00220 {
00221     template<typename A, typename B>
00222     static constexpr int to_msg(char *, unsigned offset, unsigned, T, A, B) noexcept
00223     { return offset; }
00224
00225     template<typename A, typename B>
00226     static constexpr int from_msg(char *, unsigned offset, unsigned, long, T const &, A, B) noexcept
00227     { return offset; }
00228 };
00229
00230
00231 // Provides empty default implementations for the Dir/Class combinations in which
00232 // the type does not participate (for the Do_in_items, Do_in_data, ... passes in
00233 // handle_svr_obj_call).
00234 //
00235 // The Svr_val_ops specializations derive from Svr_noops, and pull in the
00236 // empty default implementations with 'using Svr_noops::{from,to}_svr', to
00237 // handle the other Dir/Cls combinations.
00238 template<typename T> struct Svr_noops
00239 {
00240     template<typename A, typename B>
00241     static constexpr int from_svr(char *, unsigned offset, unsigned, long, T, A, B) noexcept
00242     { return offset; }
00243
00244     template<typename A, typename B>
00245     static constexpr int to_svr(char *, unsigned offset, unsigned, T const &, A, B) noexcept
00246     { return offset; }
00247 };
00248
00249
00250 // Default implementation, via reinterpret_cast.
00251 // Override by partially specializing Clnt_val_ops.
00252 template<typename MTYPE, typename CLASS>
00253 struct Clnt_val_ops<MTYPE, Dir_in, CLASS> : Clnt_noops<MTYPE>
00254 {
00255     using Clnt_noops<MTYPE>::to_msg;
00256     static int to_msg(char *msg, unsigned offset, unsigned limit,
00257                     MTYPE arg, Dir_in, CLASS) noexcept
00258     { return msg_add<MTYPE>(msg, offset, limit, arg); }
00259 };
00260
00261
00262 // Default implementation, via reinterpret_cast.
00263 // Override by partially specializing Clnt_val_ops.
00264 template<typename MTYPE, typename CLASS>
00265 struct Clnt_val_ops<MTYPE, Dir_out, CLASS> : Clnt_noops<MTYPE>
00266 {
00267     using Clnt_noops<MTYPE>::from_msg;
00268     static int from_msg(char *msg, unsigned offset, unsigned limit, long,
00269                       MTYPE &arg, Dir_out, CLASS) noexcept
00270     { return msg_get<MTYPE>(msg, offset, limit, arg); }
00271 };
00272
00273
00281 template<typename MTYPE, typename DIR, typename CLASS> struct Svr_val_ops;

```

```

00282
00283 // Default implementation, via reinterpret_cast.
00284 // Override by partially specializing Clnt_val_ops.
00285 template<typename MTYPE, typename CLASS>
00286 struct Svr_val_ops<MTYPE, Dir_in, CLASS> : Svr_noops<MTYPE>
00287 {
00288     using Svr_noops<MTYPE>::to_svr;
00289     static int to_svr(char *msg, unsigned offset, unsigned limit,
00290                     MTYPE &arg, Dir_in, CLASS) noexcept
00291     { return msg_get<MTYPE>(msg, offset, limit, arg); }
00292 };
00293
00294
00295 // Default implementation, via reinterpret_cast.
00296 // Override by partially specializing Clnt_val_ops.
00297 template<typename MTYPE, typename CLASS>
00298 struct Svr_val_ops<MTYPE, Dir_out, CLASS> : Svr_noops<MTYPE>
00299 {
00300     using Svr_noops<MTYPE>::to_svr;
00301     static int to_svr(char *, unsigned offs, unsigned limit,
00302                     MTYPE &, Dir_out, CLASS) noexcept
00303     {
00304         offs = align_to<MTYPE>(offs);
00305         if (L4_UNLIKELY(!check_size<MTYPE>(offs, limit)))
00306             return -L4_MSGTOOLONG;
00307         return offs + sizeof(MTYPE);
00308     }
00309
00310     using Svr_noops<MTYPE>::from_svr;
00311     static int from_svr(char *msg, unsigned offset, unsigned limit, long,
00312                     MTYPE arg, Dir_out, CLASS) noexcept
00313     { return msg_add<MTYPE>(msg, offset, limit, arg); }
00314 };
00315
00316
00317 template<typename T> struct Elem
00318 {
00319     using arg_type = T;
00320     using svr_type = T;
00321     using svr_arg_type = T; // might be const & (depending on the size)
00322
00323     enum { Is_optional = false };
00324 };
00325
00326
00327 template<typename T> struct Elem<T &>
00328 {
00329     using arg_type = T &;
00330     using svr_type = T;
00331     using svr_arg_type = T &;
00332     enum { Is_optional = false };
00333 };
00334
00335 template<typename T> struct Elem<T const &>
00336 {
00337     using arg_type = T const &;
00338     using svr_type = T;
00339     // as the RPC uses a const reference we use it here too,
00340     // we could also use pass by value depending on the size
00341     using svr_arg_type = T const &;
00342     enum { Is_optional = false };
00343 };
00344
00345
00346 template<typename T> struct Elem<T *> : Elem<T &>
00347 {
00348     using arg_type = T *;
00349 };
00350
00351
00352 template<typename T> struct Elem<T const *> : Elem<T const &>
00353 {
00354     using arg_type = T const *;
00355 };
00356
00357
00358 template<typename T> struct Is_valid_rpc_type : L4::Types::True {};
00359
00360
00361 // Static assertions outside functions work only properly from C++11
00362 // onwards. On earlier version make sure the compiler fails on an ugly
00363 // undefined struct instead.
00364 template<typename T, bool B> struct Error_invalid_rpc_parameter_used;
00365 template<typename T> struct Error_invalid_rpc_parameter_used<T, true> {};
00366
00367
00368 #if __cplusplus >= 201103L
00369 template<typename T>
00370 struct _Elem : Elem<T>
00371 {
00372     static_assert(Is_valid_rpc_type<T>::value,
00373                 "L4::Rpc::Msg::_Elem<T>: type T is not a valid RPC parameter type.");
00374 };
00375 #else
00376 #endif

```

```

00377 template<typename T>
00378 struct _Elem : Elem<T>,
00379             Error_invalid_rpc_parameter_used<T, Is_valid_rpc_type<T>::value>
00380 {};
00381 #endif
00382
00383
00384 template<typename T> struct Class : Cls_data {};
00385 template<typename T> struct Direction : Dir_in {};
00386 template<typename T> struct Direction<T const &> : Dir_in {};
00387 template<typename T> struct Direction<T const *> : Dir_in {};
00388 template<typename T> struct Direction<T &> : Dir_out {};
00389 template<typename T> struct Direction<T *> : Dir_out {};
00390
00391 template<typename T> struct _Clnt_noops :
00392     Clnt_noops<typename Detail::_Plain<typename _Elem<T>::arg_type>::type>
00393 {};
00394
00395 namespace Detail {
00396
00397     template<typename T, typename DIR, typename CLASS>
00398     struct _Clnt_val_ops :
00399         Clnt_val_ops<typename Detail::_Plain<T>::type, DIR, CLASS> {};
00400
00401     template<typename T,
00402             typename ELEM = _Elem<T>,
00403             typename CLNT_OPS = _Clnt_val_ops<typename ELEM::arg_type,
00404                                             typename Direction<T>::type,
00405                                             typename Class<typename Detail::_Plain<T>::type>::type>
00406             >
00407     struct _Clnt_xmit : CLNT_OPS {};
00408
00409     template<typename T,
00410             typename ELEM = _Elem<T>,
00411             typename SVR_OPS = Svr_val_ops<typename ELEM::svr_type,
00412                                           typename Direction<T>::type,
00413                                           typename Class<typename Detail::_Plain<T>::type>::type>
00414             >
00415     struct _Svr_xmit : SVR_OPS {};
00416
00417 } //namespace Detail
00418
00419 // Specialize to alias a type X to T in client call signature.
00420 // Needs to be combined with a matching Elem< X > : Elem< T > specialization.
00421 // Or alternatively X needs to inherit from T.
00422 template<typename T> struct Clnt_xmit : Detail::_Clnt_xmit<T> {};
00423 // Specialize to alias a type X to T in server op... signature.
00424 // Needs to be combined with a matching Elem< X > : Elem< T > specialization.
00425 // Or alternatively X needs to inherit from T.
00426 template<typename T> struct Svr_xmit : Detail::_Svr_xmit<T> {};
00427
00428 }}} // namespace Msg, Ipc, L4
00429
00430

```

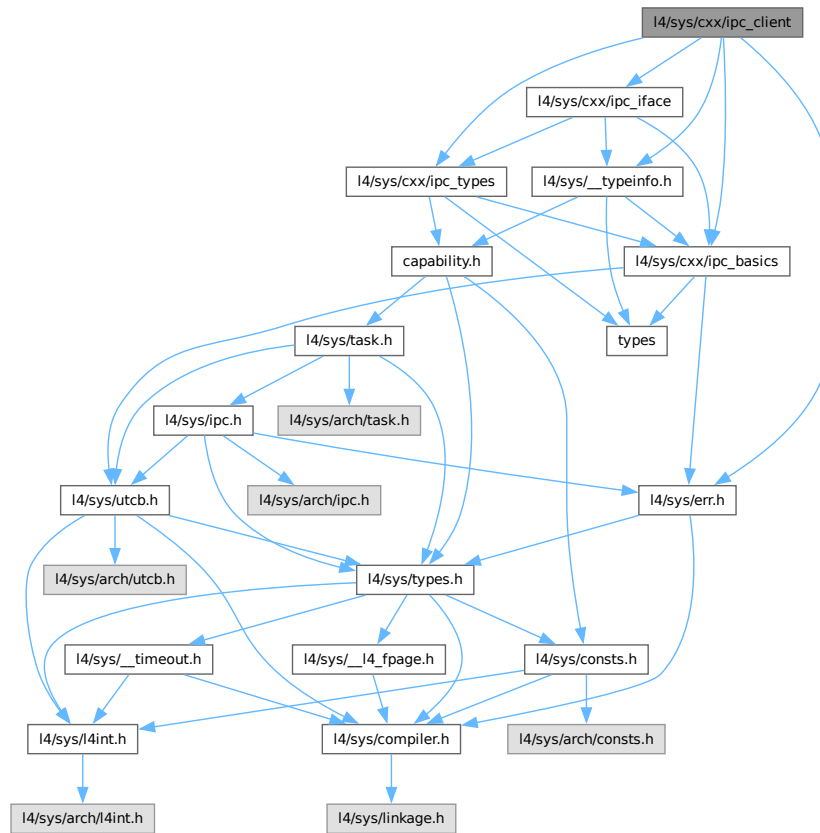
16.486 l4/sys/cxx/ipc_client File Reference

```

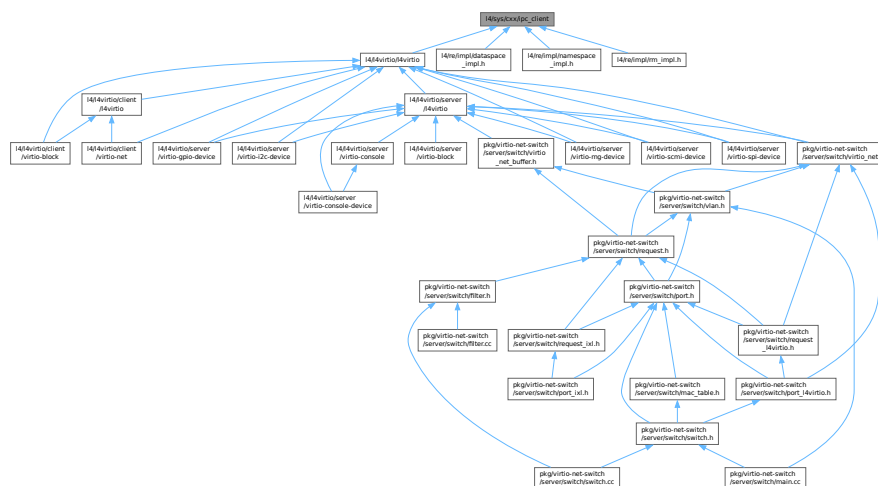
#include <l4/sys/cxx/ipc_basics>
#include <l4/sys/cxx/ipc_types>
#include <l4/sys/cxx/ipc_iface>
#include <l4/sys/__typeinfo.h>
#include <l4/sys/err.h>

```

Include dependency graph for ipc_client:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace L4

- [L4](#) *low-level kernel interface.*
- namespace [L4::lpc](#)
IPC related functionality.
- namespace [L4::lpc::Msg](#)
IPC Message related functionality.

Macros

- `#define L4_RPC_DEF(name)`
Generate the definition of an RPC stub.

16.486.1 Macro Definition Documentation

16.486.1.1 L4_RPC_DEF

```
#define L4_RPC_DEF(  
    name)
```

Value:

```
template struct L4::Ipc::Msg::Rpc_call \  
    <name##_t, name##_t::class_type, name##_t::ipc_type, name##_t::flags_type>
```

Generate the definition of an RPC stub.

Parameters

| | |
|-------------|--|
| <i>name</i> | The fully qualified method name to be implemented, this means <code>class::method</code> . |
|-------------|--|

This macro generates the definition (implementation) for the given RPC interface method.

Definition at line [32](#) of file [ipc_client](#).

16.487 ipc_client

[Go to the documentation of this file.](#)

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008 #pragma GCC system_header
00009
00010 #include <l4/sys/cxx/ipc_basics>
00011 #include <l4/sys/cxx/ipc_types>
00012 #include <l4/sys/cxx/ipc_iface>
00013 #include <l4/sys/__typeinfo.h>
00014 #include <l4/sys/err.h>
00015
00019
00020 namespace L4 { namespace Ipc { namespace Msg {
00021 //-----
00022
00032 #define L4_RPC_DEF(name) \  
00033     template struct L4::Ipc::Msg::Rpc_call \  

```



```

00034     <name##_t, name##_t::class_type, name##_t::ipc_type, name##_t::flags_type>
00035
00036
00037 //-----
00038 //Implementation of the RPC call
00039 template<typename OP, typename C, typename FLAGS, typename R, typename ...ARGS>
00040 R L4_EXPORT
00041 Rpc_call<OP, C, R (ARGS...), FLAGS>::
00042 call(L4::Cap<C> cap, typename _Elem<ARGS>::arg_type ...a, l4_utcb_t *utcb) noexcept
00043 {
00044     return Rpc_inline_call<OP, C, R (ARGS...), FLAGS>::call(cap, a..., utcb);
00045 }
00046
00047
00048 } // namespace Msg
00049 } // namespace Ipc
00050 } // namespace L4
00051
00052

```

16.488 ipc_epiface

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2014-2015 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008 #pragma GCC system_header
00009
00010 #include "capability.h"
00011 #include "ipc_server"
00012 #include "ipc_string"
00013 #include <l4/sys/types.h>
00014 #include <l4/sys/utcb.h>
00015 #include <l4/sys/__typeinfo.h>
00016 #include <l4/sys/meta>
00017 #include <l4/cxx/result>
00018 #include <l4/cxx/type_traits>
00019
00020 namespace L4 {
00021
00022 // forward for Irqep_t
00023 class Irq;
00024 class Rcv_endpoint;
00025
00026 namespace Ipc_svr {
00027
00028 class Timeout;
00029
00030 class Server_iface
00031 {
00032 private:
00033     Server_iface(Server_iface const &);
00034     Server_iface const &operator = (Server_iface const &);
00035
00036 public:
00037     class Mem_window final
00038     {
00039     public:
00040         class Allocator
00041         {
00042         public:
00043             virtual ~Allocator() = default;
00044
00045             virtual void dispose(l4_fpage_t fp) noexcept = 0;
00046         };
00047
00048         Mem_window() noexcept = default;
00049         Mem_window(l4_fpage_t fp, Allocator *allocator) noexcept
00050             : _fp(fp), _allocator(allocator)
00051         {}
00052
00053         ~Mem_window()
00054         { reset(); }
00055
00056         Mem_window(Mem_window const &) = delete;
00057         Mem_window &operator=(Mem_window const &) = delete;
00058
00059         Mem_window(Mem_window &&other) noexcept
00060             : _fp(other._fp), _allocator(other._allocator)
00061         {
00062             other._fp = l4_fpage_invalid();
00063             other._allocator = nullptr;
00064         }
00065     };
00066 }
00067

```

```

00077     }
00078
00079     Mem_window &operator=(Mem_window &&other) noexcept
00080     {
00081         if (this == &other)
00082             return *this;
00083
00084         reset();
00085         _fp = other._fp;
00086         _allocator = other._allocator;
00087
00088         other._fp = l4_fpage_invalid();
00089         other._allocator = nullptr;
00090
00091         return *this;
00092     }
00093
00094     void reset() noexcept
00095     {
00096         if (l4_is_fpage_valid(_fp))
00097         {
00098             _allocator->dispose(_fp);
00099             _fp = l4_fpage_invalid();
00100             _allocator = nullptr;
00101         }
00102     }
00103
00104     l4_fpage_t release() noexcept
00105     {
00106         l4_fpage_t ret = _fp;
00107         _fp = l4_fpage_invalid();
00108         _allocator = nullptr;
00109         return ret;
00110     }
00111
00112     explicit operator bool() const noexcept
00113     { return l4_is_fpage_valid(_fp); }
00114
00115     l4_fpage_t fp() const noexcept
00116     { return _fp; }
00117
00118     void *get() const noexcept
00119     { return reinterpret_cast<void *>(l4_fpage_memaddr(_fp)); }
00120
00121     unsigned order() const noexcept
00122     { return l4_fpage_size(_fp); }
00123
00124     l4_umword_t size() const noexcept
00125     { return l4_umword_t{1} « order(); }
00126
00127 private:
00128     l4_fpage_t _fp = l4_fpage_invalid();
00129     Allocator *_allocator = nullptr;
00130 };
00131
00132 using Demand = L4::Type_info::Demand;
00133
00134 Server_iface(Server_iface &&) = delete;
00135 Server_iface &operator = (Server_iface &&) = delete;
00136
00137 Server_iface() {}
00138
00139 // Destroy the server interface
00140 virtual ~Server_iface() = 0;
00141
00142 virtual int alloc_buffer_demand(Demand const &demand) = 0;
00143
00144 virtual L4::Cap<void> get_rcv_cap(int index) const = 0;
00145
00146 virtual int realloc_rcv_cap(int index) = 0;
00147
00148 virtual cxx::Result<L4::Reply_cap> take_reply_cap() noexcept
00149 { return cxx::Error(-L4_ENOSYS); }
00150
00151 virtual cxx::Result<Mem_window> get_rcv_mem() noexcept
00152 { return cxx::Error(-L4_ENOSYS); }
00153
00154 virtual int add_timeout(Timeout *timeout, l4_kernel_clock_t time) = 0;
00155
00156 virtual int remove_timeout(Timeout *timeout) = 0;
00157
00158 template<typename T>
00159 L4::Cap<T> rcv_cap(int index) const
00160 { return L4::cap_cast<T>(get_rcv_cap(index)); }
00161
00162 L4::Cap<void> rcv_cap(int index) const
00163 { return get_rcv_cap(index); }

```

```

00240 };
00241
00242 inline Server_iface::~Server_iface() {}
00243
00244 } // namespace Ipc_svr
00245
00257 struct Epiface
00258 {
00259     Epiface(Epiface const &) = delete;
00260     Epiface &operator = (Epiface const &) = delete;
00261
00263     using Server_iface = Ipc_svr::Server_iface;
00265     using Demand = Ipc_svr::Server_iface::Demand;
00266
00267     class Stored_cap : public Cap<void>
00268     {
00269     private:
00270         enum { Managed = 0x10 };
00271
00272     public:
00273         Stored_cap() = default;
00274         Stored_cap(Cap<void> const &c, bool managed = false)
00275             : Cap<void>((c.cap() & L4_CAP_MASK) | (managed ? Managed : 0))
00276         {
00277             static_assert (!(L4_CAP_MASK & Managed), "conflicting bits used...");
00278         }
00279
00280         bool managed() const { return cap() & Managed; }
00281     };
00282
00284     Epiface() : _data(0) {}
00285
00298     virtual l4_msgtag_t dispatch(l4_msgtag_t tag, unsigned rights,
00299                                 l4_utcb_t *utcb) = 0;
00300
00307     virtual Demand get_buffer_demand() const = 0; //{ return Demand(0); }
00308
00310     virtual ~Epiface() = 0;
00311
00318     Stored_cap obj_cap() const { return _cap; }
00319
00325     Server_iface *server_iface() const { return _data; }
00326
00336     int set_server(Server_iface *srv, Cap<void> cap, bool managed = false)
00337     {
00338         if ((srv && cap) || (!srv && !cap))
00339         {
00340             _data = srv;
00341             _cap = Stored_cap(cap, managed);
00342             return 0;
00343         }
00344
00345         return -L4_EINVAL;
00346     }
00347
00351     void set_obj_cap(Cap<void> const &cap) { _cap = cap; }
00352
00353 private:
00354     Server_iface *_data;
00355     Stored_cap _cap;
00356 };
00357
00358 inline Epiface::~Epiface() {}
00359
00367 template<typename RPC_IFACE, typename BASE = Epiface>
00368 struct Epiface_t0 : BASE
00369 {
00371     using Interface = RPC_IFACE;
00372
00374     typename Type_info::Demand get_buffer_demand() const
00375     { return typename Kobject_typeid<RPC_IFACE>::Demand(); }
00376
00381     Cap<RPC_IFACE> obj_cap() const
00382     { return L4::cap_cast<RPC_IFACE>(BASE::obj_cap()); }
00383 };
00384
00392 template<typename Derived, typename BASE = Epiface,
00393          bool = cxx::is_polymorphic<BASE>::value>
00394 struct Irqep_t : Epiface_t0<void, BASE>
00395 {
00396     l4_msgtag_t dispatch(l4_msgtag_t, unsigned, l4_utcb_t *) final
00397     {
00398         static_cast<Derived*>(this)->handle_irq();
00399         return l4_msgtag(~L4_ENOREPLY, 0, 0, 0);
00400     }
00401
00406     Cap<L4::Irq> obj_cap() const

```

```

00407 { return L4::cap_cast<L4::Irq>(BASE::obj_cap()); }
00408 };
00409
00410 template<typename Derived, typename BASE>
00411 struct Irqep_t<Derived, BASE, false> : Epiface_t0<void, BASE>
00412 {
00413     l4_msgtag_t dispatch(l4_msgtag_t, unsigned, l4_utcb_t *)
00414     {
00415         static_cast<Derived*>(this)->handle_irq();
00416         return l4_msgtag(~L4_ENOREPLY, 0, 0, 0);
00417     }
00418
00419     Cap<L4::Irq> obj_cap() const
00420     { return L4::cap_cast<L4::Irq>(BASE::obj_cap()); }
00421 };
00422
00423 class Registry_iface
00424 {
00425 public:
00426     virtual ~Registry_iface() = 0;
00427
00428     virtual L4::Cap<void>
00429     register_obj(L4::Epiface *o, char const *service) = 0;
00430
00431     virtual L4::Cap<void>
00432     register_obj(L4::Epiface *o) = 0;
00433
00434     virtual L4::Cap<L4::Irq> register_irq_obj(L4::Epiface *o) = 0;
00435
00436     virtual L4::Cap<L4::Rcv_endpoint>
00437     register_obj(L4::Epiface *o, L4::Cap<L4::Rcv_endpoint> ep) = 0;
00438
00439     virtual void
00440     unregister_obj(L4::Epiface *o, bool unmap = true) = 0;
00441 };
00442
00443 inline Registry_iface::~Registry_iface() {}
00444
00445 namespace Ipc {
00446 namespace Detail {
00447 using namespace L4::Typeid;
00448
00449 template<typename IFACE>
00450 struct Meta_svr
00451 {
00452     l4_ret_t op_num_interfaces(L4::Meta::Rights)
00453     {
00454         l4_ret_t ret = 0;
00455         auto iter = [&ret](L4::Type_info const *)
00456         {
00457             ++ret;
00458             return false;
00459         };
00460         _iter_interfaces(iter);
00461         return ret;
00462     }
00463
00464     l4_ret_t op_interface(L4::Meta::Rights, l4_umword_t idx, long &proto,
00465                           L4::Ipc::String<char> &name)
00466     {
00467         auto iter =
00468             [idx, &proto, &name, num = 0U](L4::Type_info const *t) mutable
00469             {
00470                 if (idx == num)
00471                 {
00472                     proto = t->proto();
00473                     if (auto *n = t->name())
00474                         name.copy_in(n);
00475                     else
00476                         name.copy_in("");
00477                     return true;
00478                 }
00479                 ++num;
00480                 return false;
00481             };
00482         return _iter_interfaces(iter) ? 0 : -L4_ERANGE;
00483     }
00484
00485     l4_ret_t op_supports(L4::Meta::Rights, l4_mword_t proto)
00486     { return L4::kobject_typeid<IFACE>()->has_proto(proto); }
00487 private:
00488     template<typename F>

```

```

00568 static bool _iter_interfaces(F &f)
00569 { return _iter_interfaces<F>(L4::kobject_typeid<IFACE>(), f); }
00570
00571 template<typename F>
00572 static bool _iter_interfaces(L4::Type_info const *t, F &f)
00573 {
00574     // Don't report L4::Kobject (no base) or L4::PROTO_EMPTY. The latter
00575     // interfaces are not dispatched and shall remain invisible.
00576     if (t->num_bases() && t->proto() != L4::PROTO_EMPTY)
00577         if (f(t))
00578             return true;
00579
00580     for (unsigned i = 0; i < t->num_bases(); ++i)
00581         if (_iter_interfaces(t->base(i), f))
00582             return true;
00583
00584     return false;
00585 }
00586 };
00587
00588 template<typename IFACE, typename LIST>
00589 struct _Dispatch;
00590
00591 // No match dispatcher found
00592 template<typename IFACE>
00593 struct _Dispatch<IFACE, Iface_list_end>
00594 {
00595     template< typename THIS, typename A1, typename A2 >
00596     static l4_msgtag_t f(THIS *, l4_msgtag_t, A1, A2 &)
00597     { return l4_msgtag(-L4_EBADPROTO, 0, 0, 0); }
00598 };
00599
00600 // call matching p_dispatch() function
00601 template<typename IFACE, typename I, typename LIST >
00602 struct _Dispatch<IFACE, Iface_list<I, LIST> >
00603 {
00604     // special handling for the meta protocol, to avoid 'using' murx
00605     template< typename THIS >
00606     static l4_msgtag_t _f(THIS *, l4_msgtag_t tag, unsigned r,
00607                          l4_utcb_t *utcb, True::type)
00608     {
00609         using L4::Ipcc::Msg::dispatch_call;
00610         using Meta = L4::Meta::Rpc;
00611         using Msvr = Meta_svr<IFACE>;
00612         return dispatch_call<Meta>(static_cast<Msvr *>(nullptr), utcb, tag, r);
00613     }
00614
00615     // normal dispatch to the op_<func> methods of \a self.
00616     template< typename THIS >
00617     static l4_msgtag_t _f(THIS *self, l4_msgtag_t t, unsigned r,
00618                          l4_utcb_t *utcb, False::type)
00619     {
00620         using L4::Ipcc::Msg::dispatch_call;
00621         return dispatch_call<typename I::iface_type::Rpc>(self, utcb, t, r);
00622     }
00623
00624     // dispatch function with switch for meta protocol
00625     template< typename THIS >
00626     static l4_msgtag_t f(THIS *self, l4_msgtag_t tag, unsigned r,
00627                         l4_utcb_t *utcb)
00628     {
00629         if (I::Proto == tag.label())
00630             return _f(self, tag, r, utcb,
00631                      Bool<I::Proto == static_cast<long>(L4_PROTO_META)>());
00632
00633         return _Dispatch<IFACE, typename LIST::type>::f(self, tag, r, utcb);
00634     }
00635 };
00636
00637 template<typename IFACE>
00638 struct Dispatch :
00639     _Dispatch<IFACE, typename L4::Kobject_typeid<IFACE>::Iface_list::type>
00640 {};
00641
00642 } // namespace Detail
00643
00644 template<typename EPIFACE>
00645 struct Dispatch : Detail::Dispatch<typename EPIFACE::Interface>
00646 {};
00647
00648 } // namespace Ipc
00649
00650 template<typename Derived, typename IFACE, typename BASE = L4::Epiface,
00651         bool = cxx::is_polymorphic<BASE>::value>
00652 struct Epiface_t : Epiface_t0<IFACE, BASE>
00653 {
00654     l4_msgtag_t

```

```

00661     dispatch(l4_msgtag_t tag, unsigned rights, l4_utcb_t *utcb) final
00662     {
00663         using Dispatch = Ipc::Dispatch<Derived>;
00664         return Dispatch::f(static_cast<Derived*>(this), tag, rights, utcb);
00665     }
00666 };
00667
00668 template<typename Derived, typename IFACE, typename BASE>
00669 struct Epiface_t<Derived, IFACE, BASE, false> : Epiface_t0<IFACE, BASE>
00670 {
00671     l4_msgtag_t
00672     dispatch(l4_msgtag_t tag, unsigned rights, l4_utcb_t *utcb)
00673     {
00674         using Dispatch = Ipc::Dispatch<Derived>;
00675         return Dispatch::f(static_cast<Derived*>(this), tag, rights, utcb);
00676     }
00677 };
00678
00684 class Basic_registry
00685 {
00686 public:
00687     using Value = Epiface;
00693     static Value *find(l4_umword_t label)
00694     { return reinterpret_cast<Value*>(label & ~3UL); }
00695
00709     static l4_msgtag_t dispatch(l4_msgtag_t tag, l4_umword_t label,
00710                                l4_utcb_t *utcb)
00711     {
00712         return find(label)->dispatch(tag, label, utcb);
00713     }
00714 };
00715
00716
00717 } // namespace L4

```

16.489 l4/sys/cxx/ipc_iface File Reference

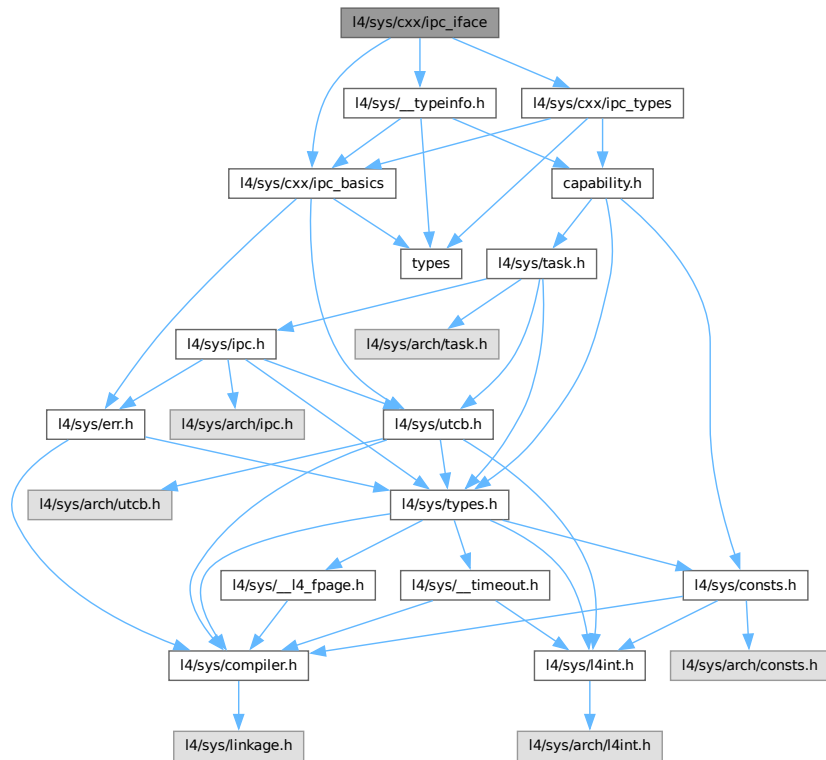
Interface Definition Language.

```

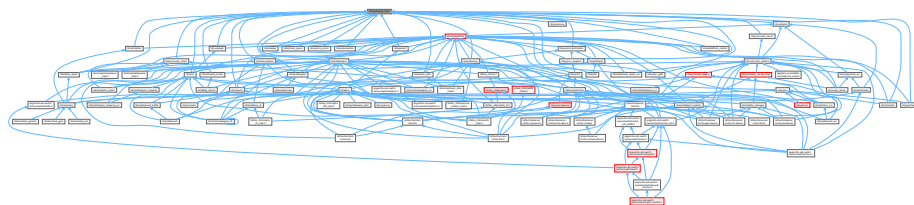
#include <l4/sys/cxx/ipc_basics>
#include <l4/sys/cxx/ipc_types>
#include <l4/sys/__typeinfo.h>

```

Include dependency graph for ipc_iface:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [L4::lpc::Call](#)
RPC attribute for a standard RPC call.
- struct [L4::lpc::Call_zero_send_timeout](#)
RPC attribute for an RPC call, with zero send timeout.
- struct [L4::lpc::Call_t< RIGHTS >](#)
RPC attribute for an RPC call with required rights.
- struct [L4::lpc::Send_only](#)
RPC attribute for a send-only RPC.

Namespaces

- namespace [L4](#)
L4 low-level kernel interface.
- namespace [L4::lpc](#)
IPC related functionality.
- namespace [L4::lpc::Msg](#)
IPC Message related functionality.

Macros

- `#define L4_INLINE_RPC_NF(res, name, args...)`
Define an inline RPC call type (the type only, no callable).
- `#define L4_INLINE_RPC_NF_OP(op, res, name, args...)`
Define an inline RPC call type with specific opcode (the type only, no callable).
- `#define L4_INLINE_RPC(res, name, args, attr...)`
Define an inline RPC call (type and callable).
- `#define L4_INLINE_RPC_OP(op, res, name, args, attr...)`
Define an inline RPC call with specific opcode (type and callable).
- `#define L4_RPC_NF(res, name, args...)`
Define an RPC call type (the type only, no callable).
- `#define L4_RPC_NF_OP(op, res, name, args...)`
Define an RPC call type with specific opcode (the type only, no callable).
- `#define L4_RPC(res, name, args, attr...)`
Define an RPC call (type and callable).
- `#define L4_RPC_OP(op, res, name, args, attr...)`
Define an RPC call with specific opcode (type and callable).

16.489.1 Detailed Description

Interface Definition Language.

See also

[L4_RPC](#), [L4_INLINE_RPC](#), [L4::lpc::Call](#) [L4::lpc::Send_only](#), [L4::lpc::Msg::Rpc_call](#), [L4::lpc::Msg::Rpc_↔
inline_call](#)

Definition in file [ipc_iface](#).

16.489.2 Macro Definition Documentation

16.489.2.1 L4_INLINE_RPC

```
#define L4_INLINE_RPC(  
    res,  
    name,  
    args,  
    attr...)
```

Value:

res name args

Define an inline RPC call (type and callable).

Parameters

| | |
|-------------|--|
| <i>res</i> | The result type of the RPC call |
| <i>name</i> | The name of the function (<i>name_t</i> is used for the type.) |
| <i>args</i> | The argument list of the RPC function. |
| <i>attr</i> | Optional RPC attributes (L4::Ipc::Call , L4::Ipc::Call_t etc.). |

Examples

[examples/clntsrv/src/shared.h](#).

Definition at line [483](#) of file [ipc_iface](#).

Referenced by [L4Re::Mem_alloc::info\(\)](#).

16.489.2.2 L4_INLINE_RPC_NF

```
#define L4_INLINE_RPC_NF(  
    res,  
    name,  
    args...)
```

Value:

```
struct name##_t : L4::Ipc::Msg::Rpc_inline_call<name##_t, Class, res args> \  
{                                                                 \  
    using type = L4::Ipc::Msg::Rpc_inline_call<name##_t, Class, res args>; \  
    L4_INLINE_RPC_SRV_FORWARD(name); \  
}
```

Define an inline RPC call type (the type only, no callable).

Parameters

| | |
|-------------|--|
| <i>res</i> | The result type of the RPC call |
| <i>name</i> | The name of the function (<i>name_t</i> is used for the type.) |
| <i>args</i> | The argument list of the RPC function, and RPC attributes (L4::Ipc::Call , L4::Ipc::Call_t etc.). |

Stubs generated by this macro can be used explicitly in custom wrapper methods that need to use the underlying RPC code and provide some higher level abstraction, for example with default arguments or extra argument conversion.

Definition at line [454](#) of file [ipc_iface](#).

16.489.2.3 L4_INLINE_RPC_NF_OP

```
#define L4_INLINE_RPC_NF_OP (
    op,
    res,
    name,
    args...)

```

Value:

```
struct name##_t : L4::Ipc::Msg::Rpc_inline_call<name##_t, Class, res args> \
{ \
    using type = L4::Ipc::Msg::Rpc_inline_call<name##_t, Class, res args>; \
    enum { Opcode = (op) }; \
    L4_INLINE_RPC_SRV_FORWARD(name); \
}

```

Define an inline RPC call type with specific opcode (the type only, no callable).

Parameters

| | |
|-------------|--|
| <i>op</i> | The opcode number for this function |
| <i>res</i> | The result type of the RPC call |
| <i>name</i> | The name of the function (<i>name_t</i> is used for the type.) |
| <i>args</i> | The argument list of the RPC function, and RPC attributes (L4::Ipc::Call , L4::Ipc::Call_t etc.). |

Stubs generated by this macro can be used explicitly in custom wrapper methods that need to use the underlying RPC code and provide some higher level abstraction, for example with default arguments or extra argument conversion.

Definition at line [467](#) of file [ipc_iface](#).

16.489.2.4 L4_INLINE_RPC_OP

```
#define L4_INLINE_RPC_OP (
    op,
    res,
    name,
    args,
    attr...)

```

Value:

```
res name args
```

Define an inline RPC call with specific opcode (type and callable).

Parameters

| | |
|-------------|--|
| <i>op</i> | The opcode number for this function |
| <i>res</i> | The result type of the RPC call |
| <i>name</i> | The name of the function (<i>name_t</i> is used for the type.) |
| <i>args</i> | The argument list of the RPC function. |
| <i>attr</i> | Optional RPC attributes (L4::Ipc::Call , L4::Ipc::Call_t etc.). |

Definition at line [498](#) of file [ipc_iface](#).

16.489.2.5 L4_RPC

```
#define L4_RPC(
    res,
    name,
    args,
    attr...)

```

Value:

res name args

Define an RPC call (type and callable).

Parameters

| | |
|-------------|--|
| <i>res</i> | The result type of the RPC call |
| <i>name</i> | The name of the function (<i>name_t</i> is used for the type.) |
| <i>args</i> | The argument list of the RPC function. |
| <i>attr</i> | Optional RPC attributes (L4::ipc::Call , L4::ipc::Call_t etc.). |

Definition at line 542 of file [ipc_iface](#).

Referenced by [L4Re::Dataspace::info\(\)](#).

16.489.2.6 L4_RPC_NF

```
#define L4_RPC_NF(
    res,
    name,
    args...)

```

Value:

```
struct name##_t : L4::Ipc::Msg::Rpc_call<name##_t, Class, res args>
{
    using type = L4::Ipc::Msg::Rpc_call<name##_t, Class, res args>;
    L4_INLINE_RPC_SRV_FORWARD(name);
}

```

Define an RPC call type (the type only, no callable).

Parameters

| | |
|-------------|--|
| <i>res</i> | The result type of the RPC call |
| <i>name</i> | The name of the function (<i>name_t</i> is used for the type.) |
| <i>args</i> | The argument list of the RPC function, and RPC attributes (L4::ipc::Call , L4::ipc::Call_t etc.). |

Definition at line 511 of file [ipc_iface](#).

Referenced by [L4Re::Dataspace::info\(\)](#).

16.489.2.7 L4_RPC_NF_OP

```
#define L4_RPC_NF_OP (
    op,
    res,
    name,
    args...)

```

Value:

```
struct name##_t : L4::Ipc::Msg::Rpc_call<name##_t, Class, res args>
{
    using type = L4::Ipc::Msg::Rpc_call<name##_t, Class, res args>;
    enum { Opcode = (op) };
    L4_INLINE_RPC_SRV_FORWARD(name);
}
```

Define an RPC call type with specific opcode (the type only, no callable).

Parameters

| | |
|-------------|--|
| <i>op</i> | The opcode number for this function |
| <i>res</i> | The result type of the RPC call |
| <i>name</i> | The name of the function (<i>name_t</i> is used for the type.) |
| <i>args</i> | The argument list of the RPC function, and RPC attributes (L4::Ipc::Call , L4::Ipc::Call_t etc.). |

Definition at line 526 of file [ipc_iface](#).

16.489.2.8 L4_RPC_OP

```
#define L4_RPC_OP (
    op,
    res,
    name,
    args,
    attr...)

```

Value:

```
res name args
```

Define an RPC call with specific opcode (type and callable).

Parameters

| | |
|-------------|--|
| <i>op</i> | The opcode number for this function |
| <i>res</i> | The result type of the RPC call |
| <i>name</i> | The name of the function (<i>name_t</i> is used for the type.) |
| <i>args</i> | The argument list of the RPC function. |
| <i>attr</i> | Optional RPC attributes (L4::Ipc::Call , L4::Ipc::Call_t etc.). |

Definition at line 557 of file [ipc_iface](#).

16.490 ipc_iface

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008 #pragma GCC system_header
00009
00010 #include <l4/sys/cxx/ipc_basics>
00011 #include <l4/sys/cxx/ipc_types>
00012 #include <l4/sys/__typeinfo.h>
00013
00022
00219
00220 // TODO: add some more documentation
00221 namespace L4 { namespace Ipc {
00222
00239 struct L4_EXPORT Call
00240 {
00241     enum { Is_call = true };
00242     enum { Rights = 0 };
00243     static l4_timeout_t timeout() { return L4_IPC_NEVER; }
00244 };
00245
00249 struct L4_EXPORT Call_zero_send_timeout : Call
00250 {
00251     static l4_timeout_t timeout() { return L4_IPC_SEND_TIMEOUT_0; }
00252 };
00253
00269 template<unsigned RIGHTS>
00270 struct L4_EXPORT Call_t : Call
00271 {
00272     enum { Rights = RIGHTS };
00273 };
00274
00287 struct L4_EXPORT Send_only
00288 {
00289     enum { Is_call = false };
00290     enum { Rights = 0 };
00291     static l4_timeout_t timeout() { return L4_IPC_NEVER; }
00292 };
00293
00294 namespace Msg {
00295
00306 template<typename OP, typename CLASS, typename SIG, typename FLAGS = Call>
00307 struct L4_EXPORT Rpc_inline_call;
00308
00313 template<typename OP, typename CLASS, typename FLAGS, typename R,
00314         typename ...ARGS>
00315 struct L4_EXPORT Rpc_inline_call<OP, CLASS, R (ARGS...), FLAGS>
00316 {
00317     template<typename T> struct Result { using result_type = T; };
00318     enum
00319     {
00320         Return_tag = L4::Types::Same_v<R, l4_msgtag_t>
00321     };
00322
00324     using type = Rpc_inline_call;
00326     using op_type = OP;
00328     using class_type = CLASS;
00330     using result_type = typename Result<R>::result_type;
00332     using ipc_type = R (ARGS...);
00334     using func_type = result_type (typename _Elem<ARGS>::arg_type...);
00335
00337     using flags_type = FLAGS;
00338
00339     template<typename RES>
00340     static L4::Types::Enable_if_t< Return_tag, RES >
00341     return_err(l4_ret_t err) noexcept { return l4_msgtag(err, 0, 0, 0); }
00342
00343     template<typename RES>
00344     static L4::Types::Enable_if_t< Return_tag, RES >
00345     return_ipc_err(l4_msgtag_t tag, l4_utcb_t const *) noexcept { return tag; }
00346
00347     template<typename RES>
00348     static L4::Types::Enable_if_t< Return_tag, RES >
00349     return_code(l4_msgtag_t tag) noexcept { return tag; }
00350
00351     template<typename RES>
00352     static L4::Types::Enable_if_t< !Return_tag, RES >
00353     return_err(long err) noexcept { return err; }

```

```

00354
00355     template<typename RES>
00356     static L4::Types::Enable_if_t< !Return_tag, RES >
00357     return_ipc_err(l4_msgtag_t, l4_utcb_t *utcb) noexcept
00358     { return l4_ipc_to_errno(l4_ipc_error_code(utcb)); }
00359
00360     template<typename RES>
00361     static L4::Types::Enable_if_t< !Return_tag, RES >
00362     return_code(l4_msgtag_t tag) noexcept { return tag.label(); }
00363
00364     static R call(L4::Cap<class_type> cap,
00365                  typename _Elem<ARGS>::arg_type ...a,
00366                  l4_utcb_t *utcb = l4_utcb()) noexcept;
00367 };
00368
00373 template<typename OP, typename CLASS, typename SIG, typename FLAGS = Call>
00374 struct L4_EXPORT Rpc_call;
00375
00383 template<typename IPC, typename SIG> struct _Call;
00384
00386 template<typename IPC, typename R, typename ...ARGS>
00387 struct _Call<IPC, R (ARGS...)>
00388 {
00389 public:
00390     using class_type = typename IPC::class_type;
00391     using result_type = typename IPC::result_type;
00392
00393 private:
00394     L4::Cap<class_type> cap() const noexcept
00395     {
00396         return L4::Cap<class_type>(reinterpret_cast<l4_cap_idx_t>(this)
00397                                     & L4_CAP_MASK);
00398     }
00399
00400 public:
00402     result_type operator () (ARGS ...a, l4_utcb_t *utcb = l4_utcb()) const noexcept
00403     { return IPC::call(cap(), a..., utcb); }
00404 };
00405
00412 template<typename IPC> struct Call : _Call<IPC, typename IPC::func_type> {};
00413
00418 template<typename OP,
00419          typename CLASS,
00420          typename FLAGS,
00421          typename R,
00422          typename ...ARGS>
00423 struct L4_EXPORT Rpc_call<OP, CLASS, R (ARGS...), FLAGS> :
00424     Rpc_inline_call<OP, CLASS, R (ARGS...), FLAGS>
00425 {
00426     static R call(L4::Cap<CLASS> cap,
00427                  typename _Elem<ARGS>::arg_type ...a,
00428                  l4_utcb_t *utcb = l4_utcb()) noexcept;
00429 };
00430
00431 #define L4_INLINE_RPC_SRV_FORWARD(name)
00432     template<typename OBJ> struct fwd
00433     {
00434         OBJ *o;
00435         fwd(OBJ *o) noexcept : o(o) {}
00436         template<typename ...ARGS>
00437         decltype(auto) call(ARGS ...a) noexcept(noexcept(o->op_##name(a...)))
00438         { return o->op_##name(a...); }
00439     }
00440
00441
00454 #define L4_INLINE_RPC_NF(res, name, args...)
00455     struct name##_t : L4::Ip::Msg::Rpc_inline_call<name##_t, Class, res args>
00456     {
00457         using type = L4::Ip::Msg::Rpc_inline_call<name##_t, Class, res args>;
00458         L4_INLINE_RPC_SRV_FORWARD(name);
00459     }
00460
00467 #define L4_INLINE_RPC_NF_OP(op, res, name, args...)
00468     struct name##_t : L4::Ip::Msg::Rpc_inline_call<name##_t, Class, res args>
00469     {
00470         using type = L4::Ip::Msg::Rpc_inline_call<name##_t, Class, res args>;
00471         enum { Opcode = (op) };
00472         L4_INLINE_RPC_SRV_FORWARD(name);
00473     }
00474
00475 #ifndef DOXYGEN
00483 #define L4_INLINE_RPC(res, name, args, attr...) res name args
00484 #else
00485 #define L4_INLINE_RPC(res, name, args...)
00486     L4_INLINE_RPC_NF(res, name, args); L4::Ip::Msg::Call<name##_t> name
00487 #endif
00488

```

```

00489 #ifndef DOXYGEN
00498 #define L4_INLINE_RPC_OP(op, res, name, args, attr...) res name args
00499 #else
00500 #define L4_INLINE_RPC_OP(op, res, name, args...) \
00501     L4_INLINE_RPC_NF_OP(op, res, name, args); L4::Ipc::Msg::Call<name##_t> name
00502 #endif
00503
00511 #define L4_RPC_NF(res, name, args...) \
00512     struct name##_t : L4::Ipc::Msg::Rpc_call<name##_t, Class, res args> \
00513     { \
00514         using type = L4::Ipc::Msg::Rpc_call<name##_t, Class, res args>; \
00515         L4_INLINE_RPC_SRV_FORWARD(name); \
00516     } \
00517
00526 #define L4_RPC_NF_OP(op, res, name, args...) \
00527     struct name##_t : L4::Ipc::Msg::Rpc_call<name##_t, Class, res args> \
00528     { \
00529         using type = L4::Ipc::Msg::Rpc_call<name##_t, Class, res args>; \
00530         enum { Opcode = (op) }; \
00531         L4_INLINE_RPC_SRV_FORWARD(name); \
00532     } \
00533
00534 #ifndef DOXYGEN
00542 #define L4_RPC(res, name, args, attr...) res name args
00543 #else
00544 #define L4_RPC(res, name, args...) \
00545     L4_RPC_NF(res, name, args); L4::Ipc::Msg::Call<name##_t> name
00546 #endif
00547
00548 #ifndef DOXYGEN
00557 #define L4_RPC_OP(op, res, name, args, attr...) res name args
00558 #else
00559 #define L4_RPC_OP(op, res, name, args...) \
00560     L4_RPC_NF_OP(op, res, name, args); L4::Ipc::Msg::Call<name##_t> name
00561 #endif
00562
00563 namespace Detail {
00564
00573 template<typename ...ARGS>
00574 struct Buf
00575 {
00576 public:
00577     template<typename DIR>
00578     static constexpr int write(char *, int offset, int) noexcept
00579     { return offset; }
00580
00581     template<typename DIR>
00582     static constexpr int read(char *, int offset, int, l4_ret_t) noexcept
00583     { return offset; }
00584
00585     using Base = void;
00586 };
00587
00588 template<typename A, typename ...M>
00589 struct Buf<A, M...> : Buf<M...>
00590 {
00591     using Base = Buf<M...>;
00592
00593     using xmit = Clnt_xmit<A>;
00594     using arg_type = typename _Elem<A>::arg_type;
00595     using plain = Detail::_Plain<arg_type>;
00596
00597     template<typename DIR>
00598     static int
00599     write(char *base, int offset, int limit,
00600           arg_type a, typename _Elem<M>::arg_type ...m) noexcept
00601     {
00602         offset = xmit::to_msg(base, offset, limit, plain::deref(a),
00603                               typename DIR::dir(), typename DIR::cls());
00604         return Base::template write<DIR>(base, offset, limit, m...);
00605     }
00606
00607     template<typename DIR>
00608     static int
00609     read(char *base, int offset, int limit, l4_ret_t ret,
00610          arg_type a, typename _Elem<M>::arg_type ...m) noexcept
00611     {
00612         int r = xmit::from_msg(base, offset, limit, ret, plain::deref(a),
00613                                typename DIR::dir(), typename DIR::cls());
00614         if (L4_LIKELY(r >= 0))
00615             return Base::template read<DIR>(base, r, limit, ret, m...);
00616
00617         if (_Elem<A>::Is_optional)
00618             return Base::template read<DIR>(base, offset, limit, ret, m...);
00619
00620         return r;

```

```

00621     }
00622 };
00623
00624 template <typename ...ARGS> struct _Part
00625 {
00626     using Data = Buf<ARGS...>;
00627
00628     template<typename DIR>
00629     static int write(void *b, int offset, int limit,
00630                     typename _Elem<ARGS>::arg_type ...m) noexcept
00631     {
00632         char *buf = static_cast<char *>(b);
00633         int r = Data::template write<DIR>(buf, offset, limit, m...);
00634         if (L4_LIKELY(r >= offset))
00635             return r - offset;
00636         return r;
00637     }
00638 }
00639
00640 template<typename DIR>
00641 static int read(void *b, int offset, int limit, l4_ret_t ret,
00642               typename _Elem<ARGS>::arg_type ...m) noexcept
00643 {
00644     char *buf = static_cast<char *>(b);
00645     int r = Data::template read<DIR>(buf, offset, limit, ret, m...);
00646     if (L4_LIKELY(r >= offset))
00647         return r - offset;
00648     return r;
00649 }
00650 };
00651
00652 template<typename IPC_TYPE, typename OPCODE = void>
00653 struct Part;
00654
00655 // The version without an op-code
00656 template<typename R, typename ...ARGS>
00657 struct Part<R (ARGS...), void> : _Part<ARGS...>
00658 {
00659     using Data = Buf<ARGS...>;
00660
00661     // write arguments, skipping the dummy opcode
00662     template<typename DIR>
00663     static int write_op(void *b, int offset, int limit,
00664                        int /*placeholder for op*/,
00665                        typename _Elem<ARGS>::arg_type ...m) noexcept
00666     {
00667         char *buf = static_cast<char *>(b);
00668         int r = Data::template write<DIR>(buf, offset, limit, m...);
00669         if (L4_LIKELY(r >= offset))
00670             return r - offset;
00671         return r;
00672     }
00673 }
00674 };
00675
00676 // Message part with additional opcode
00677 template<typename OPCODE, typename R, typename ...ARGS>
00678 struct Part<R (ARGS...), OPCODE> : _Part<ARGS...>
00679 {
00680     using opcode_type = OPCODE;
00681     using Data = Buf<opcode_type, ARGS...>;
00682
00683     // write arguments, including the opcode
00684     template<typename DIR>
00685     static int write_op(void *b, int offset, int limit,
00686                        opcode_type op, typename _Elem<ARGS>::arg_type ...m) noexcept
00687     {
00688         char *buf = static_cast<char *>(b);
00689         int r = Data::template write<DIR>(buf, offset, limit, op, m...);
00690         if (L4_LIKELY(r >= offset))
00691             return r - offset;
00692         return r;
00693     }
00694 }
00695 };
00696
00697 } // namespace Detail
00698
00699 //-----
00700 // Implementation of the RPC call
00701 // TODO: Add support for timeout via special RPC argument
00702 // TODO: Add support for passing the UTCB pointer as argument
00703 //
00704 template<typename OP, typename CLASS, typename FLAGS, typename R,
00705         typename ...ARGS>
00706 inline R
00707 Rpc_inline_call<OP, CLASS, R (ARGS...), FLAGS>::
00708     call(L4::Cap<CLASS> cap,
00709         typename _Elem<ARGS>::arg_type ...a,

```



```

00717         l4_utcb_t *utcb) noexcept
00718 {
00719     using namespace Ipc::Msg;
00720
00721     using Rpc = typename Kobject_typeid<CLASS>::Iface::Rpc;
00722     using Opt = typename Rpc::template Rpc<OP>;
00723     using Args = Detail::Part<ipc_type, typename Rpc::opcode_type>;
00724
00725     l4_msg_regs_t *mrs = l4_utcb_mr_u(utcb);
00726
00727     // handle in-data part of the arguments
00728     int send_bytes =
00729         Args::template write_op<Do_in_data>(mrs->mr, 0, Mr_bytes,
00730                                             Opt::Opcode, a...);
00731
00732     if (L4_UNLIKELY(send_bytes < 0))
00733         return return_err<R>(send_bytes);
00734
00735     send_bytes = align_to<l4_umword_t>(send_bytes);
00736     int const send_words = send_bytes / Word_bytes;
00737     // write the in-items part of the message if there is one
00738     int item_bytes =
00739         Args::template write<Do_in_items>(&mrs->mr[send_words], 0,
00740                                           Mr_bytes - send_bytes, a...);
00741
00742     if (L4_UNLIKELY(item_bytes < 0))
00743         return return_err<R>(item_bytes);
00744
00745     int send_items = item_bytes / Item_bytes;
00746
00747     {
00748         // setup the receive buffers for the RPC call
00749         l4_buf_regs_t *brs = l4_utcb_br_u(utcb);
00750         // XXX: we currently support only one type of receive buffers per call
00751         brs->bdr = 0; // we always start at br[0]
00752
00753         // the limit leaves us at least one register for the zero terminator
00754         // add the buffers given as arguments to the buffer registers
00755         int bytes =
00756             Args::template write<Do_rcv_buffers>(brs->br, 0, Br_bytes - Word_bytes,
00757                                                  a...);
00758
00759         if (L4_UNLIKELY(bytes < 0))
00760             return return_err<R>(bytes);
00761
00762         brs->br[bytes / Word_bytes] = 0;
00763     }
00764
00765     // here we do the actual IPC -----
00766     l4_msgtag_t t;
00767     t = l4_msgtag(CLASS::Protocol, send_words, send_items, 0);
00768     // do the call (Q: do we need support for timeouts?)
00769     if (flags_type::Is_call)
00770         t = l4_ipc_call(cap.cap(), utcb, t, flags_type::timeout());
00771     else
00772     {
00773         t = l4_ipc_send(cap.cap(), utcb, t, flags_type::timeout());
00774         if (L4_UNLIKELY(t.has_error()))
00775             return return_ipc_err<R>(t, utcb);
00776
00777         return return_code<R>(l4_msgtag(0, 0, 0, t.flags()));
00778     }
00779
00780     // unmarshalling starts here -----
00781
00782     // bail out early in the case of an IPC error
00783     if (L4_UNLIKELY(t.has_error()))
00784         return return_ipc_err<R>(t, utcb);
00785
00786     // take the label as return value
00787     l4_ret_t r = t.label();
00788
00789     // bail out on negative error codes too
00790     if (L4_UNLIKELY(r < 0))
00791         return return_err<R>(r);
00792
00793     int const rcv_bytes = t.words() * Word_bytes;
00794
00795     // read the static out-data values to the arguments
00796     int err = Args::template read<Do_out_data>(mrs->mr, 0, rcv_bytes, r, a...);
00797
00798     int const item_limit = t.items() * Item_bytes;
00799
00800     if (L4_UNLIKELY(err < 0 || item_limit > Mr_bytes))
00801         return return_err<R>(-L4_MSGTOOSHORT);
00802
00803

```

```

00804 // read the static out-items to the arguments
00805 err = Args::template read<Do_out_items>(&mrs->mr[t.words()], 0, item_limit,
00806                                         r, a...);
00807
00808 if (L4_UNLIKELY(err < 0))
00809     return return_err<R>(-L4_MSGTOOSHORT);
00810
00811 return return_code<R>(t);
00812 }
00813
00814 } // namespace Msg
00815 } // namespace Ipc
00816 } // namespace L4

```

16.491 ipc_legacy

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * Copyright (C) 2015, 2017, 2024 Kernkonzept GmbH.
00004  * Author(s): Alexander Warg <alexander.warg@kernkonzept.com>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include <l4/sys/cxx/ipc_epiface>
00011 #ifndef L4_RPC_DISABLE_LEGACY_DISPATCH
00012 #define L4_RPC_LEGACY_DISPATCH(IFACE)
00013     template<typename IOS>
00014     int dispatch(unsigned rights, IOS &ios)
00015     {
00016         using Dispatch = ::L4::Ipc::Detail::Dispatch<IFACE>;
00017         l4_msgtag_t r = Dispatch::f(this, ios.tag(), rights, ios.utcb());
00018         ios.set_ipc_params(r);
00019         return r.label();
00020     }
00021
00022     template<typename IOS>
00023     int p_dispatch(IFACE *, unsigned rights, IOS &ios)
00024     {
00025         using ::L4::Ipc::Msg::dispatch_call;
00026         l4_msgtag_t r;
00027         r = dispatch_call<typename IFACE::Rpcs>(this, ios.utcb(),
00028                                                ios.tag(), rights);
00029         ios.set_ipc_params(r);
00030         return r.label();
00031     }
00032
00033 #define L4_RPC_LEGACY_USING(IFACE) \
00034     using IFACE::p_dispatch
00035
00036 #else
00037 #define L4_RPC_LEGACY_DISPATCH(IFACE)
00038 #define L4_RPC_LEGACY_USING(IFACE)
00039 #endif

```

16.492 ipc_ret_array

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include "types"
00010 #include "ipc_basics"
00011
00012 namespace L4 { namespace Ipc L4_EXPORT {
00013
00014 // -----
00023 template<typename T> struct L4_EXPORT Ret_array
00024 {
00025     using ptr_type = T const **;
00026
00027     T *value = nullptr;
00028     unsigned max = 0;

```

```

00029 Ret_array() {}
00030 Ret_array(T *v, unsigned max) : value(v), max(max) {}
00031 };
00032
00033 namespace Msg {
00034
00035 template<typename A> struct Elem< Ret_array<A> >
00036 {
00037     enum { Is_optional = false };
00038     using type = Ret_array<A>;
00039     using arg_type = typename type::ptr_type;
00040     using svr_type = type;
00041     using svr_arg_type = type;
00042 };
00043
00044 template<typename A>
00045 struct Is_valid_rpc_type<Ret_array<A> *> : L4::Types::False {};
00046 template<typename A>
00047 struct Is_valid_rpc_type<Ret_array<A> &> : L4::Types::False {};
00048 template<typename A>
00049 struct Is_valid_rpc_type<Ret_array<A> const &> : L4::Types::False {};
00050 template<typename A>
00051 struct Is_valid_rpc_type<Ret_array<A> const *> : L4::Types::False {};
00052
00053 template<typename A> struct Class< Ret_array<A> > : Class<A>::type {};
00054 template<typename A> struct Direction< Ret_array<A> > : Dir_out {};
00055
00056 template<typename A, typename CLASS>
00057 struct Clnt_val_ops<A const *, Dir_out, CLASS> : Clnt_noops<A const *>
00058 {
00059     using Clnt_noops<A const *>::from_msg;
00060     static int from_msg(char *msg, unsigned offset, unsigned limit, long ret,
00061         A const *&arg, Dir_out, Cls_data)
00062     {
00063         offset = align_to<A>(offset);
00064         arg = reinterpret_cast<A const *>(msg + offset);
00065         if (L4_UNLIKELY(!check_size<A>(offset, limit, ret)))
00066             return -1;
00067         return offset + ret * sizeof(A);
00068     }
00069 };
00070 };
00071
00072 template<typename A, typename CLASS>
00073 struct Svr_val_ops<Ret_array<A>, Dir_out, CLASS> :
00074     Svr_noops<Ret_array<A> >
00075 {
00076     using ret_array = Ret_array<A>;
00077     using Svr_noops<ret_array>::from_svr;
00078     static int from_svr(char *msg, unsigned offset, unsigned limit, long ret,
00079         ret_array const &arg, Dir_out, CLASS)
00080     {
00081         offset = align_to<A>(offset);
00082         if (L4_UNLIKELY(!check_size<A>(offset, limit, ret)))
00083             return -1;
00084
00085         // In case of deferred replies, the Ret_array is pointing to a server
00086         // provided buffer instead of the UTCB.
00087         A *data = reinterpret_cast<A *>(msg + offset);
00088         if (data != arg.value)
00089             for (long i = 0; i < ret; ++i)
00090                 data[i] = arg.value[i];
00091         return offset + sizeof(A) * ret;
00092     }
00093 };
00094
00095 using Svr_noops<ret_array>::to_svr;
00096 static int to_svr(char *msg, unsigned offset, unsigned limit,
00097     ret_array &arg, Dir_out, CLASS)
00098 {
00099     // there can be actually no limit check here, as this
00100     // is variably sized output array
00101     // FIXME: we could somehow make sure that this is the last
00102     // output value...
00103     offset = align_to<A>(offset);
00104     arg = ret_array(reinterpret_cast<A*>(msg + offset),
00105         (limit - offset) / sizeof(A));
00106     // FIXME: we don't know the length of the array here so, cheat
00107     return offset;
00108 }
00109 };
00110 } // namespace Msg
00111
00112 }

```


16.494 ipc_server

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *           Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *           Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010  *           economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014 #pragma once
00015
00016 #include <l4/sys/capability>
00017 #include <l4/sys/typeinfo_svr>
00018 #include <l4/sys/err.h>
00019 #include <l4/cxx/ipc_stream>
00020 #include <l4/sys/cxx/ipc_epiface>
00021 #include <l4/sys/cxx/ipc_server_loop>
00022 #include <l4/cxx/type_traits>
00023 #include <l4/cxx/exceptions>
00024
00025 namespace L4 {
00026
00038 class Server_object : public Epiface
00039 {
00040 public:
00058     virtual int dispatch(unsigned long rights, Ipc::Iostream &ios) = 0;
00059
00060     l4_msgtag_t dispatch(l4_msgtag_t tag, unsigned rights, l4_utcb_t *utcb) override
00061     {
00062         L4::Ipc::Iostream ios(utcb);
00063         ios.tag() = tag;
00064         int r = dispatch(rights, ios);
00065         return ios.prepare_ipc(r);
00066     }
00067
00068     Cap<Kobject> obj_cap() const
00069     { return cap_cast<Kobject>(Epiface::obj_cap()); }
00070 };
00071
00079 template<typename IFACE, typename BASE = L4::Server_object>
00080 struct Server_object_t : BASE
00081 {
00083     typedef IFACE Interface;
00084
00086     typename BASE::Demand get_buffer_demand() const override
00087     { return typename L4::Kobject_typeid<IFACE>::Demand(); }
00088
00099     int dispatch_meta_request(L4::Ipc::Iostream &ios)
00100     { return L4::Util::handle_meta_request<IFACE>(ios); }
00101
00113     template<typename THIS>
00114     static int proto_dispatch(THIS *self, l4_umword_t rights, L4::Ipc::Iostream &ios)
00115     {
00116         l4_msgtag_t t;
00117         ios » t;
00118         return Kobject_typeid<IFACE>::proto_dispatch(self, t.label(), rights, ios);
00119     }
00120 };
00121
00142 template<typename Derived, typename IFACE, typename BASE = L4::Server_object>
00143 struct Server_object_x : Server_object_t<IFACE, BASE>
00144 {
00146     int dispatch(l4_umword_t r, L4::Ipc::Iostream &ios)
00147     {
00148         return Server_object_t<IFACE, BASE>::proto_dispatch(static_cast<Derived *>(this),
00149                                                             r, ios);
00150     }
00151 };
00152
00161 struct Irq_handler_object : Server_object_t<Kobject> {};
00162
00163 }

```

16.495 ipc_server

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*

```

```

00003  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008 #pragma GCC system_header
00009
00010 #include <l4/sys/cxx/ipc_basics>
00011 #include <l4/sys/cxx/ipc_iface>
00012 #include <l4/sys/cxx/limits>
00013 #include <l4/sys/___typeinfo.h>
00014 #include <l4/sys/assert.h>
00015 #include <l4/sys/err.h>
00016 #include <stddef.h>
00017
00018 namespace L4 {
00019 namespace Ipc {
00020 namespace Msg {
00021 namespace Detail {
00022
00023 template<typename T> struct Sizeof { enum { size = sizeof(T) }; };
00024 template<> struct Sizeof<void> { enum { size = 0 }; };
00025
00026 template<typename ...> struct Arg_pack
00027 {
00028     template<typename DIR>
00029     unsigned get(char *, unsigned offset, unsigned)
00030     { return offset; }
00031
00032     template<typename DIR>
00033     unsigned set(char *, unsigned offset, unsigned, l4_ret_t)
00034     { return offset; }
00035
00036     template<typename F, typename ...ARGS>
00037     decltype(auto) call(F f, ARGS ...args)
00038     { return f(args...); }
00039
00040     template<typename ...>
00041     void reply()
00042     {}
00043
00044     template<typename O, typename FUNC, typename ...ARGS>
00045     auto obj_call(O *o, ARGS ...args)
00046     -> decltype(typename FUNC::template fwd<O>(o).template call<ARGS...>(args...))
00047     {
00048         using Fwd = typename FUNC::template fwd<O>;
00049         using Idl_ret = typename FUNC::result_type;
00050         using Svr_ret = decltype(Fwd(o).template call<ARGS...>(args...));
00051         auto ret = Fwd(o).template call<ARGS...>(args...);
00052
00053         using L4::Types::numeric_limits;
00054         // Truncating a negative return value which is too small could result in a
00055         // positive value pretending "success" to the caller.
00056         if constexpr (numeric_limits<Svr_ret>::min() < numeric_limits<Idl_ret>::min())
00057         {
00058             if (L4_UNLIKELY(ret < numeric_limits<Idl_ret>::min()))
00059                 return -L4_MSGERRRANGE;
00060         }
00061         // Truncating a positive return value which is too large results either in a
00062         // positive value different from the original return value (but interpreted
00063         // as "success" by the caller) or in a negative value. Trigger an assertion
00064         // in these cases but also return an error in case of NDEBUG=y.
00065         if constexpr (numeric_limits<Svr_ret>::max() > numeric_limits<Idl_ret>::max())
00066         {
00067             l4_assert(ret <= numeric_limits<Idl_ret>::max());
00068             if (L4_UNLIKELY(ret > numeric_limits<Idl_ret>::max()))
00069                 return -L4_MSGERRRANGE;
00070         }
00071         // The following two checks are required in case the Idl_ret limits exceed
00072         // the limits of the 'label' part of l4_msgtag_t.
00073         if constexpr (numeric_limits<Svr_ret>::min() < numeric_limits<l4_msgtag_t>::min())
00074         {
00075             if (L4_UNLIKELY(ret < numeric_limits<l4_msgtag_t>::min()))
00076                 return -L4_MSGERRRANGE;
00077         }
00078         if constexpr (numeric_limits<Svr_ret>::max() > numeric_limits<l4_msgtag_t>::max())
00079         {
00080             l4_assert(ret <= numeric_limits<l4_msgtag_t>::max());
00081             if (L4_UNLIKELY(ret > numeric_limits<l4_msgtag_t>::max()))
00082                 return -L4_MSGERRRANGE;
00083         }
00084         return ret;
00085     }
00086 };
00087
00088 }
00089 }
00090 };
00091
00092 template<typename T, typename SVR_TYPE, typename ...M>

```

```

00096 struct Svr_arg : Svr_xmit<T>, Arg_pack<M...>
00097 {
00098     using Base = Arg_pack<M...>;
00099
00100     using svr_type = SVR_TYPE;
00101     using svr_arg_type = typename _Elem<T>::svr_arg_type;
00102
00103     svr_type v;
00104
00105     template<typename DIR>
00106     int get(char *msg, unsigned offset, unsigned limit)
00107     {
00108         using ct = Svr_xmit<T>;
00109         int r = ct::to_svr(msg, offset, limit, this->v,
00110             typename DIR::dir(), typename DIR::cls());
00111         if (L4_LIKELY(r >= 0))
00112             return Base::template get<DIR>(msg, r, limit);
00113
00114         if (_Elem<T>::Is_optional)
00115         {
00116             v = svr_type();
00117             return Base::template get<DIR>(msg, offset, limit);
00118         }
00119         return r;
00120     }
00121
00122     template<typename DIR>
00123     int set(char *msg, unsigned offset, unsigned limit, l4_ret_t ret)
00124     {
00125         using ct = Svr_xmit<T>;
00126         int r = ct::from_svr(msg, offset, limit, ret, this->v,
00127             typename DIR::dir(), typename DIR::cls());
00128         if (L4_UNLIKELY(r < 0))
00129             return r;
00130         return Base::template set<DIR>(msg, r, limit, ret);
00131     }
00132
00133     template<typename F, typename ...ARGS>
00134     decltype(auto) call(F f, ARGS ...args)
00135     {
00136         //As_arg<value_type> check;
00137         return Base::template
00138             call<F, ARGS..., svr_arg_type>(f, args..., this->v);
00139     }
00140
00141     template<typename ...ARGS>
00142     void reply(ARGS &&...args)
00143     {
00144         // Only consume out parameters or parameters of type In_out.
00145         if constexpr (L4::Types::Same_v<typename Direction<T>::type, Dir_out>
00146             || L4::Types::Same_template_v<T, In_out>)
00147             reply_consume_arg<ARGS...>(L4::Types::forward<ARGS>(args)...);
00148         else
00149             Base::template reply<ARGS...>(L4::Types::forward<ARGS>(args)...);
00150     }
00151
00152     template<typename O, typename FUNC, typename ...ARGS>
00153     decltype(auto) obj_call(O *o, ARGS ...args)
00154     {
00155         //As_arg<value_type> check;
00156         return Base::template
00157             obj_call<O, FUNC, ARGS..., svr_arg_type>(o, args..., this->v);
00158     }
00159
00160 private:
00161     template<typename A, typename ...ARGS>
00162     void reply_consume_arg(A &&a, ARGS &&...args)
00163     {
00164         this->v = L4::Types::move(a);
00165         Base::template reply<ARGS...>(L4::Types::forward<ARGS>(args)...);
00166     }
00167 };
00168
00174 template<typename T, typename ...M>
00175 struct Svr_arg<T, void, M...> : Arg_pack<M...>
00176 {};
00177
00178 template<typename A, typename ...M>
00179 struct Arg_pack<A, M...> : Svr_arg<A, typename _Elem<A>::svr_type, M...>
00180 {};
00181
00182 } // namespace Detail
00183
00184 //-----
00189 template<typename IPC_TYPE> struct Svr_arg_pack;
00190
00191 template<typename R, typename ...ARGS>

```

```

00192 struct Svr_arg_pack<R (ARGS...)> : Detail::Arg_pack<ARGS...>
00193 {
00194     using Base = Detail::Arg_pack<ARGS...>;
00195     template<typename DIR>
00196     int get(void *msg, unsigned offset, unsigned limit)
00197     {
00198         char *buf = static_cast<char *>(msg);
00199         return Base::template get<DIR>(buf, offset, limit);
00200     }
00201
00202     template<typename DIR>
00203     int set(void *msg, unsigned offset, unsigned limit, l4_ret_t ret)
00204     {
00205         char *buf = static_cast<char *>(msg);
00206         return Base::template set<DIR>(buf, offset, limit, ret);
00207     }
00208 };
00209
00213 template<typename IPC_TYPE, typename O, typename ...ARGS>
00214 static l4_msgtag_t
00215 handle_svr_obj_call(O *o, l4_utcb_t *utcb, l4_msgtag_t tag, ARGS ...args)
00216 {
00217     using Pack = Svr_arg_pack<typename IPC_TYPE::rpc::ipc_type>;
00218     enum
00219     {
00220         Do_reply = IPC_TYPE::rpc::flags_type::Is_call,
00221         Short_err = Do_reply ? -L4_MSGTOOSHORT : -L4_ENOREPLY,
00222     };
00223
00224     // XXX: send a reply or just do not reply in case of a cheating client
00225     if (L4_UNLIKELY(tag.words() + tag.items() * Item_words > Mr_words))
00226         return l4_msgtag(Short_err, 0, 0, 0);
00227
00228     // our whole arguments data structure
00229     Pack pack;
00230     l4_msg_regs_t *mrs = l4_utcb_mr_u(utcb);
00231
00232     int in_pos = Detail::Sizeof<typename IPC_TYPE::opcode_type>::size;
00233
00234     unsigned const in_bytes = tag.words() * Word_bytes;
00235
00236     in_pos = pack.template get<Do_in_data>(&mrs->mr[0], in_pos, in_bytes);
00237
00238     if (L4_UNLIKELY(in_pos < 0))
00239         return l4_msgtag(Short_err, 0, 0, 0);
00240
00241     if (L4_UNLIKELY(pack.template get<Do_out_data>(mrs->mr, 0, Mr_bytes) < 0))
00242         return l4_msgtag(Short_err, 0, 0, 0);
00243
00244     in_pos = pack.template get<Do_in_items>(&mrs->mr[tag.words()], 0,
00245                                         tag.items() * Item_bytes);
00246
00247     if (L4_UNLIKELY(in_pos < 0))
00248         return l4_msgtag(Short_err, 0, 0, 0);
00249
00250     asm volatile ("": "=m" (mrs->mr));
00251
00252     // call the server function
00253     auto ret = pack.template obj_call<O, typename IPC_TYPE::rpc, ARGS...>(o, args...);
00254
00255     if (!Do_reply)
00256         return l4_msgtag(-L4_ENOREPLY, 0, 0, 0);
00257
00258     // our convention says that negative return value means no
00259     // reply data
00260     if (L4_UNLIKELY(ret < 0))
00261         return l4_msgtag(ret, 0, 0, 0);
00262
00263     // reply with the reply data from the server function
00264     int bytes = pack.template set<Do_out_data>(mrs->mr, 0, Mr_bytes, ret);
00265     if (L4_UNLIKELY(bytes < 0))
00266         return l4_msgtag(-L4_MSGTOOLONG, 0, 0, 0);
00267
00268     unsigned words = (bytes + Word_bytes - 1) / Word_bytes;
00269     bytes = pack.template set<Do_out_items>(&mrs->mr[words], 0,
00270                                         Mr_bytes - words * Word_bytes,
00271                                         ret);
00272     if (L4_UNLIKELY(bytes < 0))
00273         return l4_msgtag(-L4_MSGTOOLONG, 0, 0, 0);
00274
00275     unsigned const items = bytes / Item_bytes;
00276     return l4_msgtag(ret, words, items, 0);
00277 }
00278
00281 template<typename RPC, typename ...OUT_ARGS>

```



```

00282 static l4_msgtag_t
00283 pack_svr_reply(l4_utcb_t *utcb, l4_ret_t ret, OUT_ARGS && ...args)
00284 {
00285     static_assert(RPC::flags_type::Is_call, "RPC to reply to must be a call.");
00286
00287     // our convention says that negative return value means no
00288     // reply data
00289     if (L4_UNLIKELY(ret < 0))
00290         return l4_msgtag(ret, 0, 0, 0);
00291
00292     // TODO: Could try to filter out Dir_in parameters already in the arg pack.
00293     using Pack = Svr_arg_pack<typename RPC::ipc_type>;
00294
00295     // our whole arguments data structure
00296     Pack pack;
00297     l4_msg_regs_t *mrs = l4_utcb_mr_u(utcb);
00298
00299     // Reserves spaces for all data output arguments.
00300     if (L4_UNLIKELY(pack.template get<Do_out_data>(mrs->mr, 0, Mr_bytes) < 0))
00301         return l4_msgtag(-L4_EMSTOOSHORT, 0, 0, 0);
00302
00303     // call the server reply function
00304     pack.template reply<OUT_ARGS...>(L4::Types::forward<OUT_ARGS>(args)...);
00305
00306     // reply with the reply data from the server function
00307     int bytes = pack.template set<Do_out_data>(mrs->mr, 0, Mr_bytes, ret);
00308     if (L4_UNLIKELY(bytes < 0))
00309         return l4_msgtag(-L4_EMSTOOLONG, 0, 0, 0);
00310
00311     unsigned words = (bytes + Word_bytes - 1) / Word_bytes;
00312     bytes = pack.template set<Do_out_items>(&mrs->mr[words], 0,
00313                                           Mr_bytes - words * Word_bytes,
00314                                           ret);
00315     if (L4_UNLIKELY(bytes < 0))
00316         return l4_msgtag(-L4_EMSTOOLONG, 0, 0, 0);
00317
00318     unsigned const items = bytes / Item_bytes;
00319     return l4_msgtag(ret, words, items, 0);
00320 }
00321
00335 template<typename RPC, typename ...OUT_ARGS>
00336 static l4_ret_t
00337 reply(L4::Reply_cap reply_cap, l4_ret_t res, OUT_ARGS && ...args)
00338 {
00339     l4_utcb_t *u = l4_utcb();
00340     return reply_cap.reply(
00341         L4::Ipc::Msg::pack_svr_reply<RPC, OUT_ARGS...>(
00342             u, res, L4::Types::forward<OUT_ARGS>(args)...),
00343         u);
00344 }
00345
00346 //-----
00347
00348 template<typename RPCS, typename OPCODE_TYPE>
00349 struct Dispatch_call;
00350
00351 template<typename CLASS>
00352 struct Dispatch_call<L4::Typeid::Raw_ipc<CLASS>, void>
00353 {
00354     template<typename OBJ, typename ...ARGS>
00355     static l4_msgtag_t
00356     call(OBJ *o, l4_utcb_t *utcb, l4_msgtag_t tag, ARGS ...a)
00357     {
00358         return o->op_dispatch(utcb, tag, a...);
00359     }
00360 };
00361
00362 template<typename RPCS>
00363 struct Dispatch_call<RPCS, void>
00364 {
00365     constexpr static unsigned rmask()
00366     { return RPCS::rpc::flags_type::Rights & 3UL; }
00367
00368     template<typename OBJ, typename ...ARGS>
00369     static l4_msgtag_t
00370     call(OBJ *o, l4_utcb_t *utcb, l4_msgtag_t tag, unsigned rights, ARGS ...a)
00371     {
00372         if ((rights & rmask()) != rmask())
00373             return l4_msgtag(-L4_EPERM, 0, 0, 0);
00374
00375         using Rights = L4::Typeid::Rights<typename RPCS::rpc::class_type>;
00376         return handle_svr_obj_call<RPCS>(o, utcb, tag,
00377                                           Rights(rights), a...);
00378     }
00379 };
00380 };
00381

```

```

00382 template<typename RPCS, typename OPCODE_TYPE>
00383 struct Dispatch_call
00384 {
00385     constexpr static unsigned rmask()
00386     { return RPCS::rpc::flags_type::Rights & 3UL; }
00387
00388     template<typename OBJ, typename ...ARGS>
00389     static l4_msgtag_t
00390     _call(OBJ *o, l4_utcb_t *utcb, l4_msgtag_t tag, unsigned rights, OPCODE_TYPE op, ARGS ...a)
00391     {
00392         if (L4::Types::Same_v<typename RPCS::opcode_type, void>
00393             || RPCS::Opcode == op)
00394         {
00395             if ((rights & rmask()) != rmask())
00396                 return l4_msgtag(-L4_EPERM, 0, 0, 0);
00397
00398             using Rights = L4::Typeid::Rights<typename RPCS::rpc::class_type>;
00399             return handle_svr_obj_call<RPCS>(o, utcb, tag,
00400                                             Rights(rights), a...);
00401         }
00402         return Dispatch_call<typename RPCS::next, OPCODE_TYPE>::template
00403             _call<OBJ, ARGS...>(o, utcb, tag, rights, op, a...);
00404     }
00405
00406     template<typename OBJ, typename ...ARGS>
00407     static l4_msgtag_t
00408     call(OBJ *o, l4_utcb_t *utcb, l4_msgtag_t tag, unsigned rights, ARGS ...a)
00409     {
00410         OPCODE_TYPE op;
00411         unsigned limit = tag.words() * Word_bytes;
00412         using S = Svr_xmit<OPCODE_TYPE>;
00413         l4_ret_t err = S::to_svr(reinterpret_cast<char *>(l4_utcb_mr_u(utcb)->mr), 0,
00414                                limit, op, Dir_in(), Cls_data());
00415         if (L4_UNLIKELY(err < 0))
00416             return l4_msgtag(-L4_EMMSGTOOSHORT, 0, 0, 0);
00417         return _call<OBJ, ARGS...>(o, utcb, tag, rights, op, a...);
00418     }
00419 };
00420
00421
00422 template<>
00423 struct Dispatch_call<Typeid::Detail::Rpc_end, void>
00424 {
00425     template<typename OBJ, typename ...ARGS>
00426     static l4_msgtag_t
00427     _call(OBJ *, l4_utcb_t *, l4_msgtag_t, unsigned, int, ARGS ...)
00428     { return l4_msgtag(-L4_ENOSYS, 0, 0, 0); }
00429
00430     template<typename OBJ, typename ...ARGS>
00431     static l4_msgtag_t
00432     call(OBJ *, l4_utcb_t *, l4_msgtag_t, unsigned, ARGS ...)
00433     { return l4_msgtag(-L4_ENOSYS, 0, 0, 0); }
00434 };
00435
00436 template<typename OPCODE_TYPE>
00437 struct Dispatch_call<Typeid::Detail::Rpc_end, OPCODE_TYPE> :
00438     Dispatch_call<Typeid::Detail::Rpc_end, void> {};
00439
00440 template<typename RPCS, typename OBJ, typename ...ARGS>
00441 static l4_msgtag_t
00442 dispatch_call(OBJ *o, l4_utcb_t *utcb, l4_msgtag_t tag, unsigned rights, ARGS ...a)
00443 {
00444     return Dispatch_call<typename RPCS::type, typename RPCS::opcode_type>::template
00445         call<OBJ, ARGS...>(o, utcb, tag, rights, a...);
00446 }
00447
00448 } // namespace Msg
00449 } // namespace Ipc
00450 } // namespace L4

```

16.496 ipc_server_loop

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * Copyright (C) 2015, 2017, 2019, 2021-2025 Kernkonzept GmbH.
00004  * Author(s): Alexander Warg <alexander.warg@kernkonzept.com>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include "ipc_epiface"
00011

```

```

00012 namespace L4 {
00013
00019
00031 namespace Ipc_svr {
00032
00046 enum Reply_mode
00047 {
00048     Reply_compound,
00049     Reply_separate
00050 };
00051
00057 struct Ignore_errors
00058 { static void error(l4_msgtag_t, l4_utcb_t *) {} };
00059
00065 struct Default_timeout
00066 { static l4_timeout_t timeout() { return L4_IPC_SEND_TIMEOUT_0; } };
00067
00073 struct Compound_reply
00074 {
00075     static Reply_mode before_reply(l4_msgtag_t, l4_utcb_t *)
00076     { return Reply_compound; }
00077 };
00078
00084 struct Default_setup_wait
00085 { static void setup_wait(l4_utcb_t *, Reply_mode) {} };
00086
00094 template< typename R >
00095 struct Direct_dispatch
00096 {
00097     R &r;
00099
00101     Direct_dispatch(R &r) : r(r) {}
00102
00104     l4_msgtag_t operator () (l4_msgtag_t tag, l4_umword_t obj, l4_utcb_t *utcb)
00105     { return r.dispatch(tag, obj, utcb); }
00106 };
00107
00115 template< typename R >
00116 struct Direct_dispatch<R*>
00117 {
00119     R *r;
00120
00122     Direct_dispatch(R *r) : r(r) {}
00123
00125     l4_msgtag_t operator () (l4_msgtag_t tag, l4_umword_t obj, l4_utcb_t *utcb)
00126     { return r->dispatch(tag, obj, utcb); }
00127 };
00128
00129 #ifdef __EXCEPTIONS
00139 template< typename R, typename Exc> // = L4::Runtime_error>
00140 struct Exc_dispatch : private Direct_dispatch<R>
00141 {
00143     Exc_dispatch(R r) : Direct_dispatch<R>(r) {}
00144
00148     l4_msgtag_t operator () (l4_msgtag_t tag, l4_umword_t obj, l4_utcb_t *utcb)
00149     {
00150         try
00151         {
00152             return Direct_dispatch<R>::operator () (tag, obj, utcb);
00153         }
00154         catch (Exc &e)
00155         {
00156             return l4_msgtag(e.err_no(), 0, 0, 0);
00157         }
00158         catch (int err)
00159         {
00160             return l4_msgtag(err, 0, 0, 0);
00161         }
00162         catch (long err)
00163         {
00164             return l4_msgtag(err, 0, 0, 0);
00165         }
00166     }
00167 };
00168
00183 template< typename R, typename Exc, typename Printer >
00184 struct Dbg_dispatch : private Direct_dispatch<R>
00185 {
00187     Dbg_dispatch(R r, Printer p) : Direct_dispatch<R>(r), _printer(p) {}
00188
00193     l4_msgtag_t operator () (l4_msgtag_t tag, l4_umword_t obj, l4_utcb_t *utcb)
00194     {
00195         try
00196         {
00197             return Direct_dispatch<R>::operator () (tag, obj, utcb);
00198         }
00199         catch (Exc &e)

```

```

00200     {
00201         _printer.printf("Error in handling IPC: %s: %s\n", e.str(),
00202             e.extra_str());
00203         return l4_msgtag(e.err_no(), 0, 0, 0);
00204     }
00205     catch (int err)
00206     {
00207         _printer.printf("Error in handling IPC: %d (%s)\n", err,
00208             l4sys_errtostr(err));
00209         return l4_msgtag(err, 0, 0, 0);
00210     }
00211     catch (long err)
00212     {
00213         _printer.printf("Error in handling IPC: %ld (%s)\n", err,
00214             l4sys_errtostr(err));
00215         return l4_msgtag(err, 0, 0, 0);
00216     }
00217 }
00218
00219 Printer _printer;
00220 };
00221 #endif
00222
00233 class Br_manager_no_buffers : public Server_iface
00234 {
00235 public:
00240     int alloc_buffer_demand(Demand const &demand) override
00241     {
00242         if (!demand.no_demand())
00243             return -L4_ENOMEM;
00244         return L4_EOK;
00245     }
00246
00248     L4::Cap<void> get_rcv_cap(int) const override
00249     { return L4::Cap<void>::Invalid; }
00250
00252     int realloc_rcv_cap(int) override
00253     { return -L4_ENOMEM; }
00254
00256     int add_timeout(Timeout *, l4_kernel_clock_t) override
00257     { return -L4_ENOSYS; }
00258
00260     int remove_timeout(Timeout *) override
00261     { return -L4_ENOSYS; }
00262
00263 protected:
00265     unsigned first_free_br() const
00266     { return 1; }
00267
00269     void setup_wait(l4_utcb_t *utcb, L4::Ipc_svr::Reply_mode)
00270     {
00271         l4_buf_regs_t *br = l4_utcb_br_u(utcb);
00272         br->bdr = 0;
00273         br->br[0] = 0;
00274     }
00275 };
00276
00285 struct Default_loop_hooks :
00286     public Ignore_errors, public Default_timeout, public Compound_reply,
00287     Br_manager_no_buffers
00288 {};
00289
00290 }
00291
00306 template< typename LOOP_HOOKS = Ipc_svr::Default_loop_hooks >
00307 class Server :
00308     public LOOP_HOOKS
00309 {
00310 public:
00317     /* Internal note: After all users have been converted, remove this
00318        * constructor. Also remove the constructor below then. */
00319     explicit Server(l4_utcb_t *)
00320         L4_DEPRECATED("Do not specify the UTCB with the constructor. "
00321             "Supply it on the loop function if needed.")
00322     {}
00323
00327     /* Internal note: Remove this constructor when the above deprecated
00328        * constructor with the UTCB pointer is also removed. */
00329     Server() {}
00330
00337     template< typename DISPATCH >
00338     inline L4_NORETURN void internal_loop(DISPATCH dispatch, l4_utcb_t *);
00339
00343     template< typename R >
00344     inline L4_NORETURN void loop_noexc(R r, l4_utcb_t *u = l4_utcb())
00345     { internal_loop(Ipc_svr::Direct_dispatch<R>(r), u); }
00346

```

```

00347 #ifndef __EXCEPTIONS
00354     template< typename EXC, typename R >
00355     inline L4_NORETURN void loop(R r, l4_utcb_t *u = l4_utcb())
00356     {
00357         internal_loop(Ipc_svr::Exc_dispatch<R, EXC>(r), u);
00358         // function will never return
00359     }
00360
00367     template< typename EXC, typename R, typename Printer >
00368     inline L4_NORETURN void loop_dbg(R r, Printer p, l4_utcb_t *u = l4_utcb())
00369     {
00370         internal_loop(Ipc_svr::Dbg_dispatch<R, EXC, Printer>(r, p), u);
00371         // function will never return
00372     }
00373 #endif
00374 protected:
00376     inline l4_msgtag_t reply_n_wait(l4_msgtag_t reply, l4_umword_t *p, l4_utcb_t *);
00377
00378 private:
00379     template<typename U> static auto _has_setup_reply_cap(int)
00380     -> decltype(&U::setup_reply_cap, L4::Types::True());
00381     template<typename> static L4::Types::False _has_setup_reply_cap(...);
00382
00383     // Backwards compatibility check via `if constexpr` in reply_n_wait() if
00384     // explicit reply cap mechanism is present. Once we move to C++20, this can
00385     // be replaced by `requires{Server::setup_reply_cap();}`.
00386     static constexpr bool Has_setup_reply_cap
00387     = L4::Types::Same_v<decltype(_has_setup_reply_cap<Server>()),
00388                        L4::Types::True>;
00389 };
00390
00391 template< typename L >
00392 inline l4_msgtag_t
00393 Server<L>::reply_n_wait(l4_msgtag_t reply, l4_umword_t *p, l4_utcb_t *utcb)
00394 {
00395     l4_cap_idx_t reply_cap = L4_INVALID_REPLY_CAP;
00396     if constexpr (Has_setup_reply_cap)
00397         reply_cap = this->setup_reply_cap().cap();
00398
00399     if (reply.label() != -L4_ENOREPLY)
00400     {
00401         Ipc_svr::Reply_mode m = this->before_reply(reply, utcb);
00402         if (m == Ipc_svr::Reply_compound)
00403         {
00404             this->setup_wait(utcb, m);
00405             return l4_ipc(reply_cap, utcb, L4_SYSF_REPLY_AND_WAIT, 0, reply, p,
00406                          this->timeout()); // l4_ipc_reply_and_wait
00407         }
00408         else
00409         {
00410             l4_msgtag_t res = l4_ipc_reply(reply_cap, utcb, reply,
00411                                           this->timeout());
00412             if (res.has_error())
00413                 return res;
00414         }
00415     }
00416     this->setup_wait(utcb, Ipc_svr::Reply_separate);
00417     return l4_ipc(reply_cap, utcb, L4_SYSF_WAIT, 0, l4_msgtag_t{0}, p,
00418                  this->timeout()); // l4_ipc_wait
00419 }
00420
00421 template< typename L >
00422 template< typename DISPATCH >
00423 inline L4_NORETURN void
00424 Server<L>::internal_loop(DISPATCH dispatch, l4_utcb_t *utcb)
00425 {
00426     l4_msgtag_t res;
00427     l4_umword_t p;
00428     l4_msgtag_t r = l4_msgtag(-L4_ENOREPLY, 0, 0, 0);
00429
00430     while (true)
00431     {
00432         res = reply_n_wait(r, &p, utcb);
00433         if (res.has_error())
00434         {
00435             this->error(res, utcb);
00436             r = l4_msgtag(-L4_ENOREPLY, 0, 0, 0);
00437             continue;
00438         }
00439
00440         r = dispatch(res, p, utcb);
00441     }
00442 }
00443
00444 } // namespace L4

```

16.497 ipc_string

```

00001 // vi:set ft=c++: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include "types"
00010 #include "ipc_basics"
00011 #include "ipc_array"
00012
00013 namespace L4 { namespace Ipc {
00014
00015     template<typename CHAR = char const, typename LEN = unsigned long>
00016     struct String : Array<CHAR, LEN>
00017     {
00018         static LEN strlenlength(CHAR *d) { LEN l = 0; while (d[l]) ++l; return l; }
00019         String() {}
00020         String(CHAR *d) : Array<CHAR, LEN>(strlenlength(d) + 1, d) {}
00021         String(LEN len, CHAR *d) : Array<CHAR, LEN>(len, d) {}
00022         void copy_in(CHAR const *s)
00023         {
00024             if (this->length < 1)
00025                 return;
00026
00027             LEN i;
00028             for (i = 0; i < this->length - 1 && s[i]; ++i)
00029                 this->data[i] = s[i];
00030             this->length = i + 1;
00031             this->data[i] = CHAR();
00032         }
00033     };
00034
00035     #if __cplusplus >= 201103L
00036     template< typename CHAR = char, typename LEN_TYPE = unsigned long,
00037             LEN_TYPE MAX = (L4_UTCB_GENERIC_DATA_SIZE *
00038                 sizeof(l4_umword_t)) / sizeof(CHAR) >
00039     using String_in_buf = Array_in_buf<CHAR, LEN_TYPE, MAX>;
00040     #endif
00041
00042     namespace Msg {
00043     template<typename A, typename LEN>
00044     struct Clnt_xmit< String<A, LEN> > : Clnt_xmit< Array<A, LEN> > {};
00045
00046     template<typename A, typename LEN, typename CLASS>
00047     struct Svr_val_ops< String<A, LEN>, Dir_in, CLASS >
00048     : Svr_val_ops< Array_ref<A, LEN>, Dir_in, CLASS >
00049     {
00050         using Base = Svr_val_ops< Array_ref<A, LEN>, Dir_in, CLASS >;
00051         using svr_type = typename Base::svr_type;
00052         using Base::to_svr;
00053         static int to_svr(char *msg, unsigned offset, unsigned limit,
00054             svr_type &a, Dir_in dir, Cls_data cls)
00055         {
00056             int r = Base::to_svr(msg, offset, limit, a, dir, cls);
00057             if (r < 0)
00058                 return r;
00059
00060             // correct clients send at least the zero terminator
00061             if (a.length < 1)
00062                 return -L4_MSGTOOSHORT;
00063
00064             using elem_type = L4::Types::Remove_const_t<A>;
00065             const_cast<elem_type*>(a.data)[a.length - 1] = A();
00066             return r;
00067         }
00068     };
00069
00070     template<typename A, typename LEN>
00071     struct Clnt_xmit<String<A, LEN> &> : Clnt_xmit<Array<A, LEN> &>
00072     {
00073         using type = Array<A, LEN> &;
00074
00075         using Clnt_xmit<type>::from_msg;
00076         static int from_msg(char *msg, unsigned offset, unsigned limit, long ret,
00077             Array<A, LEN> &a, Dir_out dir, Cls_data cls)
00078         {
00079             int r = Clnt_xmit<type>::from_msg(msg, offset, limit, ret, a, dir, cls);
00080             if (r < 0)
00081                 return r;
00082
00083             // check for a bad server
00084             if (a.length < 1)

```

```

00085         return -L4_EMMSGTOOSHORT;
00086
00087     a.data[a.length - 1] = A();
00088     return r;
00089 };
00090 };
00091
00092 template<typename A, typename LEN>
00093 struct Clnt_xmit<String<A, LEN> *> : Clnt_xmit<String<A, LEN> &> {};
00094
00095 template<typename A, typename LEN, typename CLASS>
00096 struct Svr_val_ops<String<A, LEN>, Dir_out, CLASS>
00097 : Svr_val_ops<Array_ref<A, LEN>, Dir_out, CLASS>
00098 {};
00099
00100 template<typename A, typename LEN>
00101 struct Is_valid_rpc_type<String<A, LEN> const *> : L4::Types::False {};
00102 template<typename A, typename LEN>
00103 struct Is_valid_rpc_type<String<A, LEN> const &> : L4::Types::False {};
00104
00105 } // namespace Msg
00106
00107 }

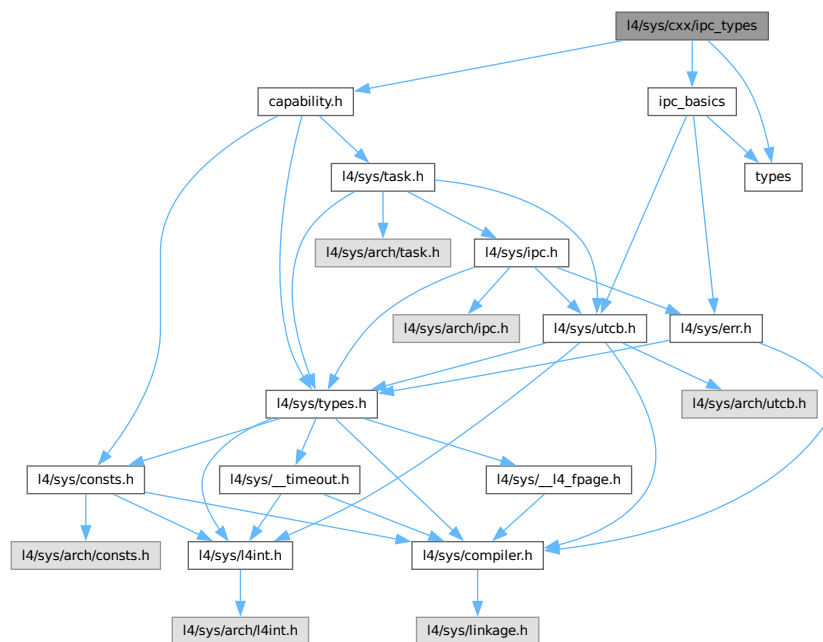
```

16.498 I4/sys/cxx/ipc_types File Reference

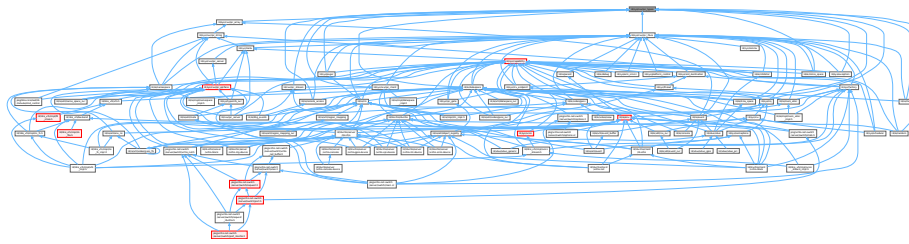
```

#include "capability.h"
#include "types"
#include "ipc_basics"
Include dependency graph for ipc_types:

```



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [L4::lpc::In_out< T >](#)
Mark an argument as in-out argument.
- struct [L4::lpc::As_value< T >](#)
Pass the argument as plain data value.
- struct [L4::lpc::Opt< T >](#)
Attribute for defining an optional RPC argument.
- class [L4::lpc::Small_buf](#)
A receive item for receiving a single object capability.
- class [L4::lpc::Gen_fpage](#)
Generic RPC base for typed message items.
- class [L4::lpc::Snd_fpage](#)
Send item or return item.
- class [L4::lpc::Rcv_fpage](#)
Non-small receive item.
- class [L4::lpc::Cap< T >](#)
Capability type for RPC interfaces (see [L4::Cap< T >](#)).

Namespaces

- namespace [L4](#)
L4 low-level kernel interface.
- namespace [L4::lpc](#)
IPC related functionality.
- namespace [L4::lpc::Msg](#)
IPC Message related functionality.

Functions

- template<typename T>
[Cap< T >](#) [L4::lpc::make_cap](#) ([L4::Cap< T >](#) cap, unsigned rights) noexcept
Make an [L4::lpc::Cap< T >](#) for the given capability and rights.
- template<typename T>
[Cap< T >](#) [L4::lpc::make_cap_rw](#) ([L4::Cap< T >](#) cap) noexcept
Make an [L4::lpc::Cap< T >](#) for the given capability with [L4_CAP_FPAGE_RW](#) rights.
- template<typename T>
[Cap< T >](#) [L4::lpc::make_cap_rws](#) ([L4::Cap< T >](#) cap) noexcept
Make an [L4::lpc::Cap< T >](#) for the given capability with [L4_CAP_FPAGE_RWS](#) rights.

- `template<typename T>`
`Cap< T > L4::ipc::make_cap_full (L4::Cap< T > cap) noexcept`
Make an L4::IPC::Cap<T> for the given capability with full fpage and object-specific rights.
- `template<typename T>`
`Cap< T > L4::ipc::make_cap_grant (L4::Cap< T > cap) noexcept`
Make an L4::IPC::Cap<T> for the given capability that shall be granted with full rights.

16.499 ipc_types

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include "capability.h"
00010 #include "types"
00011 #include "ipc_basics"
00012
00013 namespace L4 {
00014     using Opcode = int;
00015
00016     namespace Ipc {
00017         template<typename T> struct L4_EXPORT Out;
00018         template<typename T> struct L4_EXPORT In_out
00019         {
00020             T v;
00021             In_out() {}
00022             In_out(T v) : v(v) {}
00023             operator T () const { return v; }
00024             operator T & () { return v; }
00025         };
00026
00027         namespace Msg {
00028             template<typename A> struct Elem< In_out<A*> > : Elem<A*> {};
00029
00030             template<typename A>
00031             struct Svr_xmit< In_out<A*> > : Svr_xmit<A*>, Svr_xmit<A const*>
00032             {
00033                 using Svr_xmit<A*>::from_svr;
00034                 using Svr_xmit<A const*>::to_svr;
00035             };
00036
00037             template<typename A>
00038             struct Clnt_xmit< In_out<A*> > : Clnt_xmit<A*>, Clnt_xmit<A const*>
00039             {
00040                 using Clnt_xmit<A*>::from_msg;
00041                 using Clnt_xmit<A const*>::to_msg;
00042             };
00043
00044             template<typename A>
00045             struct Is_valid_rpc_type< In_out<A*> > : Is_valid_rpc_type<A*> {};
00046             template<typename A>
00047             struct Is_valid_rpc_type< In_out<A const*> > : L4::Types::False {};
00048
00049             #ifdef CONFIG_ALLOW_REFS
00050             template<typename A> struct Elem< In_out<A&> > : Elem<A&> {};
00051
00052             template<typename A>
00053             struct Svr_xmit< In_out<A&> > : Svr_xmit<A&>, Svr_xmit<A const&>
00054             {
00055                 using Svr_xmit<A&>::from_svr;
00056                 using Svr_xmit<A const&>::to_svr;
00057             };
00058
00059             template<typename A>
00060             struct Clnt_xmit< In_out<A&> > : Clnt_xmit<A&>, Clnt_xmit<A const&>
00061             {
00062                 using Clnt_xmit<A&>::from_msg;
00063                 using Clnt_xmit<A const&>::to_msg;
00064             };
00065
00066             #endif
00067         }
00068     }
00069 }

```

```

00087 };
00088
00089 template<typename A>
00090 struct Is_valid_rpc_type< In_out<A &> > : Is_valid_rpc_type<A &> {};
00091 template<typename A>
00092 struct Is_valid_rpc_type< In_out<A const &> > : L4::Types::False {};
00093
00094 #else
00095
00096 template<typename A>
00097 struct Is_valid_rpc_type< In_out<A &> > : L4::Types::False {};
00098
00099 #endif
00100
00101 // Value types don't make sense for output.
00102 template<typename A>
00103 struct Is_valid_rpc_type< In_out<A> > : L4::Types::False {};
00104
00105 }
00106
00107
00116 template<typename T> struct L4_EXPORT As_value
00117 {
00118     using value_type = T;
00119     T v;
00120     As_value() noexcept {}
00121     As_value(T v) noexcept : v(v) {}
00122     operator T () const noexcept { return v; }
00123     operator T & () noexcept { return v; }
00124 };
00125
00126 namespace Msg {
00127 template<typename T> struct Class< As_value<T> > : Cls_data {};
00128 template<typename T> struct Elem< As_value<T> > : Elem<T> {};
00129 template<typename T> struct Elem< As_value<T> *> : Elem<T *> {};
00130 }
00131
00132
00136 template<typename T> struct L4_EXPORT Opt
00137 {
00138     T _value;
00139     bool _valid;
00140
00142     Opt() noexcept : _valid(false) {}
00143
00145     Opt(T value) noexcept : _value(value), _valid(true) {}
00146
00148     Opt &operator = (T value) noexcept
00149     {
00150         this->_value = value;
00151         this->_valid = true;
00152         return *this;
00153     }
00154
00156     void set_valid(bool valid = true) noexcept { _valid = valid; }
00157
00159     T *operator -> () noexcept { return &this->_value; }
00161     T const *operator -> () const noexcept { return &this->_value; }
00163     T value() const noexcept { return this->_value; }
00165     T &value() noexcept { return this->_value; }
00167     bool is_valid() const noexcept { return this->_valid; }
00168 };
00169
00170 namespace Msg {
00171 template<typename T> struct Elem< Opt<T &> > : Elem<T &>
00172 {
00173     enum { Is_optional = true };
00174     using svr_arg_type = Opt<typename Elem<T &>::svr_type> &;
00175     using svr_type = Opt<typename Elem<T &>::svr_type>;
00176 };
00177
00178 template<typename T> struct Elem< Opt<T *> > : Elem<T *>
00179 {
00180     enum { Is_optional = true };
00181     using svr_arg_type = Opt<typename Elem<T *>::svr_type> &;
00182     using svr_type = Opt<typename Elem<T *>::svr_type>;
00183 };
00184
00185
00187 template<typename T, typename CLASS>
00188 struct Svr_val_ops<Opt<T>, Dir_out, CLASS> : Svr_noops< Opt<T> >
00189 {
00190     using svr_type = Opt<T>;
00191     using Native = Svr_val_ops<T, Dir_out, CLASS>;
00192
00193     using Svr_noops< svr_type >::to_svr;

```

```

00194     static int to_svr(char *msg, unsigned offset, unsigned limit,
00195                      svr_type &arg, Dir_out, CLASS) noexcept
00196     {
00197         return Native::to_svr(msg, offset, limit, arg.value(), Dir_out(), CLASS());
00198     }
00199
00200     using Svr_noops< svr_type >::from_svr;
00201     static int from_svr(char *msg, unsigned offset, unsigned limit, l4_ret_t ret,
00202                       svr_type const &arg, Dir_out, CLASS) noexcept
00203     {
00204         if (arg.is_valid())
00205             return Native::from_svr(msg, offset, limit, ret, arg.value(),
00206                                    Dir_out(), CLASS());
00207         return offset;
00208     }
00209 };
00210
00211 template<typename T> struct Elem< Opt<T> > : Elem<T>
00212 {
00213     enum { Is_optional = true };
00214     using arg_type = Opt<T>;
00215 };
00216
00217 template<typename T> struct Elem< Opt<T const *> > : Elem<T const *>
00218 {
00219     enum { Is_optional = true };
00220     using arg_type = Opt<T const *>;
00221 };
00222
00223 template<typename T>
00224 struct Is_valid_rpc_type< Opt<T const &> > : L4::Types::False {};
00225
00226 template<typename T, typename CLASS>
00227 struct Clnt_val_ops<Opt<T>, Dir_in, CLASS> : Clnt_noops< Opt<T> >
00228 {
00229     using arg_type = Opt<T>;
00230     using Native = Detail::_Clnt_val_ops<typename Elem<T>::arg_type, Dir_in, CLASS>;
00231
00232     using Clnt_noops< arg_type >::to_msg;
00233     static int to_msg(char *msg, unsigned offset, unsigned limit,
00234                      arg_type arg, Dir_in, CLASS) noexcept
00235     {
00236         if (arg.is_valid())
00237             return Native::to_msg(msg, offset, limit,
00238                                  Detail::_Plain<T>::deref(arg.value()),
00239                                  Dir_in(), CLASS());
00240         return offset;
00241     }
00242 };
00243
00244 template<typename T> struct Class< Opt<T> > :
00245     Class< typename Detail::_Plain<T>::type > {};
00246 template<typename T> struct Direction< Opt<T> > : Direction<T> {};
00247 }
00248
00257 class L4_EXPORT Small_buf
00258 {
00259 public:
00267     explicit Small_buf(L4::Cap<void> cap, unsigned long flags = 0) noexcept
00268         : _data(cap.cap() | L4_RCV_ITEM_SINGLE_CAP | flags) {}
00269
00274     explicit Small_buf(l4_cap_idx_t cap, unsigned long flags = 0) noexcept
00275         : _data(cap | L4_RCV_ITEM_SINGLE_CAP | flags) {}
00276
00278     l4_umword_t raw() const noexcept { return _data; }
00279 private:
00280     l4_umword_t _data;
00281 };
00282
00286 class L4_EXPORT Gen_fpage
00287 {
00288 public:
00290     enum Type
00291     {
00292         Special = L4_FPAGE_SPECIAL << 4,
00293         Memory  = L4_FPAGE_MEMORY << 4,
00294         Io      = L4_FPAGE_IO << 4,
00295         Obj     = L4_FPAGE_OBJ << 4
00296     };
00297
00299     Gen_fpage(l4_umword_t base, l4_umword_t data) noexcept
00300         : _base(base), _data(data)
00301     {}
00302
00304     l4_umword_t data() const noexcept { return _data; }
00306     l4_umword_t base_x() const noexcept { return _base; }
00307

```

```

00308 protected:
00309     l4_umword_t _base;
00310     l4_umword_t _data;
00311 };
00312
00323 class Snd_fpage : public Gen_fpage
00324 {
00325 public:
00326     enum Map_type
00327     {
00330         Map = L4_MAP_ITEM_MAP,
00331         Grant = L4_MAP_ITEM_GRANT,
00332     };
00333
00336     enum Cacheopt
00337     {
00338         None = 0,
00339         Cached = L4_FPAGE_CACHEABLE << 4,
00340         Buffered = L4_FPAGE_BUFFERABLE << 4,
00341         Uncached = L4_FPAGE_UNCACHEABLE << 4
00342     };
00343
00346     enum Continue
00347     {
00348         Single = 0,
00349         Last = 0,
00350         More = L4_ITEM_CONT,
00351         Compound = L4_ITEM_CONT,
00352     };
00353
00355     Snd_fpage(l4_umword_t base = 0, l4_umword_t data = 0) noexcept
00356     : Gen_fpage(base, data)
00357     {}
00358
00370     Snd_fpage(l4_fpage_t const &fp, l4_addr_t snd_base = 0,
00371               Map_type map_type = Map,
00372               Cacheopt cache = None, Continue cont = Last) noexcept
00373     : Gen_fpage(L4_ITEM_MAP | (snd_base & (~0UL << 12)) | l4_umword_t(map_type)
00374               | l4_umword_t(cache) | l4_umword_t(cont),
00375               fp.raw)
00376     {}
00377
00386     Snd_fpage(L4::Cap<void> cap, unsigned rights, Map_type map_type = Map) noexcept
00387     : Gen_fpage(L4_ITEM_MAP | l4_umword_t(map_type) | (rights & 0xf0),
00388               cap.fpage(rights).raw)
00389     {}
00390
00402     static Snd_fpage obj(l4_cap_idx_t base, int order,
00403                         unsigned char rights,
00404                         l4_addr_t snd_base = 0,
00405                         Map_type map_type = Map,
00406                         Continue cont = Last) noexcept
00407     {
00408         return Snd_fpage(l4_obj_fpage(base, order, rights), snd_base,
00409                         map_type, None, cont);
00410     }
00411
00424     static Snd_fpage mem(l4_addr_t base, int order,
00425                         unsigned char rights,
00426                         l4_addr_t snd_base = 0,
00427                         Map_type map_type = Map,
00428                         Cacheopt cache = None, Continue cont = Last) noexcept
00429     {
00430         return Snd_fpage(l4_fpage(base, order, rights), snd_base, map_type, cache,
00431                         cont);
00432     }
00433
00445     static Snd_fpage io(unsigned long base, int order,
00446                        unsigned char rights,
00447                        l4_addr_t snd_base = 0,
00448                        Map_type map_type = Map,
00449                        Continue cont = Last) noexcept
00450     {
00451         return Snd_fpage(l4_fpage_set_rights(l4_iofpage(base, order), rights),
00452                         snd_base, map_type, None, cont);
00453     }
00454
00457     unsigned order() const noexcept { return (_data >> 6) & 0x3f; }
00458
00461     unsigned snd_order() const noexcept { return (_data >> 6) & 0x3f; }
00462
00465     unsigned rcv_order() const noexcept { return (_base >> 6) & 0x3f; }
00466
00469     l4_addr_t base() const noexcept { return _data & (~0UL << 12); }
00470
00473     l4_addr_t snd_base() const noexcept { return _base & (~0UL << 12); }
00474

```

```

00477 void snd_base(l4_addr_t b) noexcept { _base = (_base & ~(~0UL << 12)) | (b & (~0UL << 12)); }
00478
00480 bool is_valid() const noexcept { return _base & L4_ITEM_MAP; }
00481
00496 bool cap_received() const noexcept { return (_base & 0x3e) == 0x38; }
00512 bool id_received() const noexcept { return (_base & 0x3e) == 0x3c; }
00528 bool local_id_received() const noexcept { return (_base & 0x3e) == 0x3e; }
00535 bool is_compound() const noexcept { return _base & 1; }
00536 };
00537
00544 class Rcv_fpage : public Gen_fpage
00545 {
00546 public:
00550 Rcv_fpage() noexcept : Gen_fpage(0, 0), _rcv_task(L4_INVALID_CAP) {}
00551
00561 Rcv_fpage(l4_fpage_t const &fp, l4_addr_t snd_base = 0,
00562           l4_cap_idx_t rcv_task = L4_INVALID_CAP) noexcept
00563 : Gen_fpage(L4_ITEM_MAP | (snd_base & (~0UL << 12))
00564             | (l4_is_valid_cap(rcv_task) ? L4_RCV_ITEM_FORWARD_MAPPINGS : 0),
00565             fp.raw),
00566   _rcv_task(rcv_task)
00567 {}
00568
00578 static Rcv_fpage obj(l4_cap_idx_t base, int order, l4_addr_t snd_base = 0,
00579                     L4::Cap<void> rcv_task = L4::Cap<void>::Invalid) noexcept
00580 {
00581     return Rcv_fpage(l4_obj_fpage(base, order, 0), snd_base,
00582                     rcv_task.cap());
00583 }
00584
00594 static Rcv_fpage mem(l4_addr_t base, int order, l4_addr_t snd_base = 0,
00595                      L4::Cap<void> rcv_task = L4::Cap<void>::Invalid) noexcept
00596 {
00597     return Rcv_fpage(l4_fpage(base, order, 0), snd_base, rcv_task.cap());
00598 }
00599
00609 static Rcv_fpage io(unsigned long base, int order, l4_addr_t snd_base = 0,
00610                    L4::Cap<void> rcv_task = L4::Cap<void>::Invalid) noexcept
00611 {
00612     return Rcv_fpage(l4_io_fpage(base, order), snd_base, rcv_task.cap());
00613 }
00614
00620 l4_cap_idx_t rcv_task() const { return _rcv_task; }
00621
00625 bool forward_mappings() const noexcept
00626 { return _base & L4_RCV_ITEM_FORWARD_MAPPINGS; }
00627
00628 protected:
00629     l4_cap_idx_t _rcv_task;
00630 };
00631
00632
00633 namespace Msg {
00634
00635 // Snd_fpage are out items
00636 template<> struct Class<L4::Ipc::Snd_fpage> : Cls_item {};
00637
00638 // Rcv_fpage are buffer items
00639 template<> struct Class<L4::Ipc::Rcv_fpage> : Cls_buffer {};
00640
00641 template<>
00642 struct Clnt_val_ops<L4::Ipc::Rcv_fpage, Dir_in, Cls_buffer>
00643 : Clnt_noops<L4::Ipc::Rcv_fpage>
00644 {
00645     using Clnt_noops<L4::Ipc::Rcv_fpage>::to_msg;
00646
00647     static int to_msg(char *msg, unsigned offs, unsigned limit,
00648                      L4::Ipc::Rcv_fpage arg, Dir_in, Cls_buffer) noexcept
00649     {
00650         offs = align_to<l4_umword_t>(offs);
00651         unsigned words = arg.forward_mappings() ? 3 : 2;
00652         if (L4_UNLIKELY(!check_size<l4_umword_t>(offs, limit, words)))
00653             return -L4_EMMSGTOOLONG;
00654         auto *buf = reinterpret_cast<l4_umword_t*>(msg + offs);
00655         *buf++ = arg.base_x();
00656         *buf++ = arg.data();
00657         if (arg.forward_mappings())
00658             *buf++ = arg.rcv_task();
00659         return offs + sizeof(l4_umword_t) * words;
00660     }
00661 };
00662
00663
00664 // Remove receive buffers from server-side arguments
00665 template<> struct Elem<L4::Ipc::Rcv_fpage>
00666 {
00667     using arg_type = L4::Ipc::Rcv_fpage;

```

```

00668     using svr_type = void;
00669     using svr_arg_type = void;
00670     enum { Is_optional = false };
00671 };
00672
00673 // Small_buf are buffer items
00674 template<> struct Class<L4::Ipc::Small_buf> : Cls_buffer {};
00675
00676 // Remove receive buffers from server-side arguments
00677 template<> struct Elem<L4::Ipc::Small_buf>
00678 {
00679     using arg_type = L4::Ipc::Small_buf;
00680     using svr_type = void;
00681     using svr_arg_type = void;
00682     enum { Is_optional = false };
00683 };
00684 } // namespace Msg
00685
00686 // L4::Cap<> handling
00687
00698 template<typename T> class Cap
00699 {
00700     template<typename O> friend class Cap;
00701     l4_umword_t _cap_n_rights;
00702
00703 public:
00704     enum
00705     {
00711         Rights_mask = 0xff,
00712
00716         Map_type_mask = 0x100,
00717
00722         Cap_mask = L4_CAP_MASK
00723     };
00724
00726     enum Map_type
00727     {
00729         Map = 0,
00731         Grant = 0x100,
00732     };
00733
00735     template<typename O>
00736     Cap(Cap<O> const &o) noexcept : _cap_n_rights(o._cap_n_rights)
00737     {
00738         L4::Cap<T>::template check_convertible_from<O>();
00739     }
00740
00742     Cap(L4::Cap<T> cap) noexcept
00743     : _cap_n_rights((cap.cap() & Cap_mask) | (cap ? L4_CAP_FPAGE_R : 0))
00744     {}
00745
00747     template<typename O>
00748     Cap(L4::Cap<O> cap) noexcept
00749     : _cap_n_rights((cap.cap() & Cap_mask) | (cap ? L4_CAP_FPAGE_R : 0))
00750     {
00751         L4::Cap<T>::template check_convertible_from<O>();
00752     }
00753
00755     Cap() noexcept : _cap_n_rights(L4_INVALID_CAP) {}
00756
00764     Cap(L4::Cap<T> cap, unsigned char rights) noexcept
00765     : _cap_n_rights((cap.cap() & Cap_mask) | (rights & Rights_mask)) {}
00766
00775     Cap(L4::Cap<T> cap, unsigned char rights, Map_type map_type) noexcept
00776     : _cap_n_rights((cap.cap() & Cap_mask) | (rights & Rights_mask)
00777         | (static_cast<l4_umword_t>(map_type) & Map_type_mask))
00778     {}
00779
00785     static Cap from_ci(l4_cap_idx_t c) noexcept
00786     { return Cap(L4::Cap<T>(c & Cap_mask), c & Rights_mask); }
00787
00789     L4::Cap<T> cap() const noexcept
00790     { return L4::Cap<T>(_cap_n_rights & Cap_mask); }
00791
00793     unsigned rights() const noexcept
00794     { return _cap_n_rights & Rights_mask; }
00795
00796     Map_type map_type() const noexcept
00797     { return static_cast<Map_type>(_cap_n_rights & Map_type_mask); }
00798
00800     L4::Ipc::Snd_fpage fpage() const noexcept
00801     {
00802         return L4::Ipc::Snd_fpage(cap(), rights(), map_type() == Map
00803             ? Snd_fpage::Map
00804             : Snd_fpage::Grant);
00805     }
00806

```

```

00808 bool is_valid() const noexcept
00809 { return !(_cap_n_rights & L4_INVALID_CAP_BIT); }
00810 };
00811
00812 template<typename T>
00819 Cap<T> make_cap(L4::Cap<T> cap, unsigned rights) noexcept
00820 { return Cap<T>(cap, rights); }
00821
00822 template<typename T>
00829 Cap<T> make_cap_rw(L4::Cap<T> cap) noexcept
00830 { return Cap<T>(cap, L4_CAP_FPAGE_RW); }
00831
00832 template<typename T>
00839 Cap<T> make_cap_rws(L4::Cap<T> cap) noexcept
00840 { return Cap<T>(cap, L4_CAP_FPAGE_RWS); }
00841
00857 template<typename T>
00858 Cap<T> make_cap_full(L4::Cap<T> cap) noexcept
00859 { return Cap<T>(cap, L4_CAP_FPAGE_RWS | L4_FPAGE_C_OBJ_RIGHTS); }
00860
00870 template<typename T>
00871 Cap<T> make_cap_grant(L4::Cap<T> cap) noexcept
00872 {
00873     return Cap<T>(cap, L4_CAP_FPAGE_RWS | L4_FPAGE_C_OBJ_RIGHTS, Cap<T>::Grant);
00874 }
00875
00876 // caps are special, they have an invalid representation
00877 template<typename T> struct L4_EXPORT Opt< Cap<T> >
00878 {
00879     Cap<T> _value;
00880     Opt() noexcept {}
00881     Opt(Cap<T> value) noexcept : _value(value) {}
00882     Opt(L4::Cap<T> value) noexcept : _value(value) {}
00883     Opt &operator = (Cap<T> value) noexcept
00884     { this->_value = value; return *this; }
00885     Opt &operator = (L4::Cap<T> value) noexcept
00886     { this->_value = value; return *this; }
00887
00888     Cap<T> value() const noexcept { return this->_value; }
00889     bool is_valid() const noexcept { return this->_value.is_valid(); }
00890 };
00891
00892 namespace Msg {
00893 // prohibit L4::Cap as argument
00894 template<typename A>
00895 struct Is_valid_rpc_type< L4::Cap<A> > : L4::Types::False {};
00896
00897 template<typename A> struct Class< Cap<A> > : Cls_item {};
00898 template<typename A> struct Elem< Cap<A> >
00899 {
00900     enum { Is_optional = false };
00901     using arg_type = Cap<A>;
00902     using svr_type = L4::Ipc::Snd_fpage;
00903     using svr_arg_type = L4::Ipc::Snd_fpage;
00904 };
00905
00906 template<typename A, typename CLASS>
00907 struct Svr_val_ops<Cap<A>, Dir_in, CLASS> :
00908     // Uses default Svr_val_ops implementation with L4::Ipc::Snd_fpage.
00909     Svr_val_ops<L4::Ipc::Snd_fpage, Dir_in, CLASS>
00910 {}
00911
00912 template<typename A, typename CLASS>
00913 struct Clnt_val_ops<Cap<A>, Dir_in, CLASS> :
00914     Clnt_noops< Cap<A> >
00915 {
00916     using Clnt_noops< Cap<A> >::to_msg;
00917
00918     static int to_msg(char *msg, unsigned offset, unsigned limit,
00919         Cap<A> arg, Dir_in, Cls_item) noexcept
00920     {
00921         // passing an invalid cap as mandatory argument is an error
00922         // XXX: This checks for a client calling error, we could
00923         // also just ignore this for performance reasons and
00924         // let the client fail badly (Alex: I'd prefer this)
00925         if (L4_UNLIKELY(!arg.is_valid()))
00926             return -L4_EMSGMISSARG;
00927
00928         return msg_add(msg, offset, limit, arg.fpage());
00929     }
00930 }
00931
00932 template<typename A>
00933 struct Elem<Out<L4::Cap<A> > >
00934 {

```

```

00937     enum { Is_optional = false };
00938     using arg_type = L4::Cap<A>;
00939     using svr_type = Ipc::Cap<A>;
00940     using svr_arg_type = svr_type &;
00941 };
00942
00943 template<typename A> struct Direction< Out< L4::Cap<A> > > : Dir_out {};
00944 template<typename A> struct Class< Out< L4::Cap<A> > > : Cls_item {};
00945
00946 template<typename A>
00947 struct Clnt_val_ops< L4::Cap<A>, Dir_out, Cls_item > :
00948     Clnt_noops< L4::Cap<A> >
00949 {
00950     using Clnt_noops< L4::Cap<A> >::to_msg;
00951     static int to_msg(char *msg, unsigned offset, unsigned limit,
00952         L4::Cap<A> arg, Dir_in, Cls_buffer) noexcept
00953     {
00954         if (L4_UNLIKELY(!arg.is_valid()))
00955             return -L4_MSGMISSARG; // no buffer inserted
00956         return msg_add(msg, offset, limit, Small_buf(arg));
00957     }
00958 };
00959
00960 template<typename A>
00961 struct Svr_val_ops< L4::Ipc::Cap<A>, Dir_out, Cls_item > :
00962     Svr_noops<Cap<A>»
00963 {
00964     using Svr_noops<Cap<A>>::from_svr;
00965     static int from_svr(char *msg, unsigned offset, unsigned limit, long,
00966         Cap<A> arg, Dir_out, Cls_item) noexcept
00967     {
00968         if (L4_UNLIKELY(!arg.is_valid()))
00969             // do not map anything
00970             return msg_add(msg, offset, limit, L4::Ipc::Snd_fpage(arg.cap(), 0));
00971         return msg_add(msg, offset, limit, arg.fpage());
00972     }
00973 };
00974 };
00975
00976 // prohibit a UTCB pointer as normal RPC argument
00977 template<> struct Is_valid_rpc_type<l4_utcb_t *> : L4::Types::False {};
00978
00979 } // namespace Msg
00980 } // namespace Ipc
00981 } // namespace L4

```

16.500 ipc_varg

```

00001 // vi:set ft=c++: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008 #pragma GCC system_header
00009
00010 #include "types"
00011 #include "ipc_basics"
00012
00013 namespace L4 { namespace Ipc L4_EXPORT {
00014
00015     template< typename T, template <typename X> class B >
00016     struct Generic_va_type : B<T>
00017     {
00018         enum { Id = B<T>::Id };
00019         using ID = B<T>;
00020         using Ret_value = T const &;
00021         using Value = T;
00022
00023         static Ret_value value(void const *d)
00024         { return *reinterpret_cast<Value const *>(d); }
00025
00026         static void const *addr_of(Value const &v) { return &v; }
00027
00028         static unsigned size(void const *) { return sizeof(T); }
00029
00030         static L4_varg_type unsigned_id()
00031         {
00032             return static_cast<L4_varg_type>(Id & ~L4_VARG_TYPE_SIGN);
00033         }
00034
00035         static L4_varg_type signed_id()

```



```

00036 {
00037     return static_cast<L4_varg_type>(Id | L4_VARG_TYPE_SIGN);
00038 }
00039
00040 static L4_varg_type id()
00041 {
00042     return static_cast<L4_varg_type>(Id);
00043 }
00044 };
00045
00046 template< typename T > struct Va_type_id;
00047 template<> struct Va_type_id<l4_umword_t> { enum { Id = L4_VARG_TYPE_UMWORD }; };
00048 template<> struct Va_type_id<l4_mword_t> { enum { Id = L4_VARG_TYPE_MWORD }; };
00049 template<> struct Va_type_id<l4_fpage_t> { enum { Id = L4_VARG_TYPE_FPAGE }; };
00050 template<> struct Va_type_id<void> { enum { Id = L4_VARG_TYPE_NIL }; };
00051 template<> struct Va_type_id<char const *> { enum { Id = L4_VARG_TYPE_STRING }; };
00052
00053 template< typename T > struct Va_type;
00054
00055 template<> struct Va_type<l4_umword_t> : Generic_va_type<l4_umword_t, Va_type_id> {};
00056 template<> struct Va_type<l4_mword_t> : Generic_va_type<l4_mword_t, Va_type_id> {};
00057 template<> struct Va_type<l4_fpage_t> : Generic_va_type<l4_fpage_t, Va_type_id> {};
00058
00059 template<> struct Va_type<void>
00060 {
00061     using Ret_value = void;
00062     using Value = void;
00063
00064     static void const *addr_of(void) { return 0; }
00065
00066     static void value(void const *) {}
00067     static L4_varg_type id() { return L4_VARG_TYPE_NIL; }
00068     static unsigned size(void const *) { return 0; }
00069 };
00070
00071 template<> struct Va_type<char const *>
00072 {
00073     using Ret_value = char const *;
00074     using Value = char const *;
00075
00076     static void const *addr_of(Value v) { return v; }
00077
00078     static L4_varg_type id() { return L4_VARG_TYPE_STRING; }
00079     static unsigned size(void const *s)
00080     {
00081         char const *_s = reinterpret_cast<char const *>(s);
00082         int l = 1;
00083         while (*_s)
00084         {
00085             ++_s; ++l;
00086         }
00087         return l;
00088     }
00089
00090     static Ret_value value(void const *d) { return static_cast<char const *>(d); }
00091 };
00092
00096 class Varg
00097 {
00098 private:
00099     enum { Direct_data = 0x8000 };
00100     l4_umword_t _tag;
00101     char const *_d;
00102
00103 public:
00104
00106     using Tag = l4_umword_t;
00107
00109     L4_varg_type type() const { return static_cast<L4_varg_type>(_tag & 0xff); }
00114     unsigned length() const { return _tag >> 16; }
00116     Tag tag() const { return _tag & ~Direct_data; }
00118     void tag(Tag tag) { _tag = tag; }
00120     void data(char const *d) { _d = d; }
00121
00123     char const *data() const
00124     {
00125         if (_tag & Direct_data)
00126         {
00127             union T { char const *d; char v[sizeof(char const *)]; };
00128             return reinterpret_cast<T const *>(&_d)->v;
00129         }
00130         return _d;
00131     }
00132
00134     #if __cplusplus >= 201103L
00135     Varg() = default;
00136     #else

```

```

00137     Varg() {}
00138 #endif
00139
00141     Varg(L4_varg_type t, void const *v, int len)
00142     : _tag(t | (static_cast<L4_mword_t>(len) << 16)),
00143       _d(static_cast<char const *>(v))
00144     {}
00145
00146     static Varg nil() { return Varg(L4_VARG_TYPE_NIL, 0, 0); }
00147
00154     template< typename V >
00155     typename Va_type<V>::Ret_value value() const
00156     {
00157         if (_tag & Direct_data)
00158         {
00159             union X { char const *d; V v; };
00160             return reinterpret_cast<X const &>(_d).v;
00161         }
00162
00163         return Va_type<V>::value(_d);
00164     }
00165
00166
00168     template< typename T >
00169     bool is_of() const { return Va_type<T>::id() == type(); }
00170
00172     bool is_nil() const { return is_of<void>(); }
00173
00175     bool is_of_int() const
00176     { return (type() & ~L4_VARG_TYPE_SIGN) == L4_VARG_TYPE_UMWORD; }
00177
00184     template< typename T >
00185     bool get_value(typename Va_type<T>::Value *v) const
00186     {
00187         if (!is_of<T>())
00188             return false;
00189
00190         *v = this->value<T>();
00191         return true;
00192     }
00193
00195     template< typename T >
00196     void set_value(void const *d)
00197     {
00198         using Vt = Va_type<T>;
00199         _tag = Vt::id() | (Vt::size(d) << 16);
00200         _d = static_cast<char const *>(d);
00201     }
00202
00204     template<typename T>
00205     void set_direct_value(T val, L4::Types::Enable_if_t<sizeof(T) <= sizeof(char const *)>, bool> = true)
00206     {
00207         static_assert(sizeof(T) <= sizeof(char const *), "direct Varg value too big");
00208         using Vt = Va_type<T>;
00209         _tag = Vt::id() | (sizeof(T) << 16) | Direct_data;
00210         union X { char const *d; T v; };
00211         reinterpret_cast<X &>(_d).v = val;
00212     }
00213
00215     template<typename T> explicit
00216     Varg(T const *data) { set_value<T>(data); }
00218     Varg(char const *data) { set_value<char const *>(data); }
00219
00221     template<typename T> explicit
00222     Varg(T data, L4::Types::Enable_if_t<sizeof(T) <= sizeof(char const *)>, bool> = true)
00223     { set_direct_value<T>(data); }
00224 };
00225
00226
00227     template<typename T>
00228     class Varg_t : public Varg
00229     {
00230     public:
00231         using Value = typename Va_type<T>::Value;
00232         explicit Varg_t(Value v) : Varg()
00233         { _data = v; set_value<T>(Va_type<T>::addr_of(_data)); }
00234
00235     private:
00236         Value _data;
00237     };
00238
00239     template<unsigned MAX = L4_UTCB_GENERIC_DATA_SIZE>
00240     class Varg_list;
00241
00253     class Varg_list_ref
00254     {
00255     private:

```

```

00256     template<unsigned T>
00257     friend class Varg_list;
00258
00260     class Iter_state
00261     {
00262     private:
00263         using M = l4_umword_t;
00264         using Mp = M const *;
00265         Mp _c;
00266         Mp _e;
00267
00269         Mp next_arg(Varg const &a) const
00270         {
00271             return _c + 1 + (Msg::align_to<M>(a.length()) / sizeof(M));
00272         }
00273
00274     public:
00276         Iter_state() : _c(nullptr), _e(nullptr) {}
00277
00279         Iter_state(Mp c, Mp e) : _c(c), _e(e)
00280         {}
00281
00283         bool valid() const
00284         { return _c && _c < _e; }
00285
00287         Mp begin() const { return _c; }
00288
00290         Mp end() const { return _e; }
00291
00296         Varg pop()
00297         {
00298             if (!valid())
00299                 return Varg::nil();
00300
00301             Varg a;
00302             a.tag(_c[0]);
00303             a.data(reinterpret_cast<char const *>(&_c[1]));
00304             _c = next_arg(a);
00305             if (_c > _e)
00306                 return Varg::nil();
00307
00308             return a;
00309         }
00310
00312         bool operator == (Iter_state const &o) const
00313         { return _c == o._c; }
00314
00316         bool operator != (Iter_state const &o) const
00317         { return _c != o._c; }
00318     };
00319
00320     Iter_state _s;
00321
00322 public:
00324     Varg_list_ref() = default;
00325
00332     Varg_list_ref(void const *start, void const *end)
00333     : _s(reinterpret_cast<l4_umword_t const *>(start),
00334         reinterpret_cast<l4_umword_t const *>(end))
00335     {}
00336
00338     class Iterator
00339     {
00340     private:
00341         Iter_state _s;
00342         Varg _a;
00343
00344     public:
00346         Iterator(Iter_state const &s)
00347         : _s(s)
00348         {
00349             _a = _s.pop();
00350         }
00351
00353         explicit operator bool () const
00354         { return !_a.is_nil(); }
00355
00357         Iterator &operator ++ ()
00358         {
00359             if (!_a.is_nil())
00360                 _a = _s.pop();
00361
00362             return *this;
00363         }
00364
00366         Varg operator * () const
00367         { return _a; }

```

```

00368
00370     bool equals(Iterator const &o) const
00371     {
00372         if (_a.is_nil() && o._a.is_nil())
00373             return true;
00374
00375         return _s == o._s;
00376     }
00377
00378     bool operator == (Iterator const &o) const
00379     { return equals(o); }
00380
00381     bool operator != (Iterator const &o) const
00382     { return !equals(o); }
00383 };
00384
00386 Varg pop_front()
00387 { return _s.pop(); }
00388
00390 Varg next()
00391     L4_DEPRECATED("Use range for or pop_front.")
00392 { return _s.pop(); }
00393
00395 Iterator begin() const
00396 { return Iterator(_s); }
00397
00399 Iterator end() const
00400 { return Iterator(Iter_state()); }
00401 };
00402
00410 template<unsigned MAX>
00411 class Varg_list : public Varg_list_ref
00412 {
00413     l4_umword_t data[MAX];
00414     Varg_list(Varg_list const &);
00415
00416 public:
00418     Varg_list(Varg_list_ref const &r)
00419     {
00420         if (!r._s.valid())
00421             return;
00422
00423         l4_umword_t const *rs = r._s.begin();
00424         unsigned c = r._s.end() - rs;
00425         for (unsigned i = 0; i < c; ++i)
00426             data[i] = rs[i];
00427
00428         this->_s = Iter_state(data, data + c);
00429     }
00430 };
00431
00432
00433 namespace Msg {
00434     template<> struct Elem<Varg const *>
00435     {
00436         using arg_type = Varg const *;
00437         using svr_type = Varg_list_ref;
00438         using svr_arg_type = Varg_list_ref;
00439         enum { Is_optional = false };
00440     };
00441
00442     template<> struct Is_valid_rpc_type<Varg> : L4::Types::False {};
00443     template<> struct Is_valid_rpc_type<Varg *> : L4::Types::False {};
00444     template<> struct Is_valid_rpc_type<Varg &> : L4::Types::False {};
00445     template<> struct Is_valid_rpc_type<Varg const &> : L4::Types::False {};
00446
00447     template<> struct Direction<Varg const *> : Dir_in {};
00448     template<> struct Class<Varg const *> : Cls_data {};
00449
00450     template<typename DIR, typename CLASS>
00451     struct Clnt_val_ops<Varg, DIR, CLASS>
00452     {
00453     template<>
00454     struct Clnt_val_ops<Varg, Dir_in, Cls_data> :
00455     Clnt_noops<Varg const &>
00456     {
00457         using Clnt_noops<Varg const &>::to_msg;
00458         static int to_msg(char *msg, unsigned offs, unsigned limit,
00459             Varg const &a, Dir_in, Cls_data)
00460         {
00461             for (Varg const *i = &a; i->tag(); ++i)
00462             {
00463                 offs = align_to<l4_umword_t>(offs);
00464                 if (L4_UNLIKELY(!check_size<l4_umword_t>(offs, limit)))
00465                     return -L4_MSGTOOLONG;
00466                 *reinterpret_cast<l4_umword_t*>(msg + offs) = i->tag();
00467                 offs += sizeof(l4_umword_t);

```

```

00468         if (L4_UNLIKELY(!check_size<char>(offs, limit, i->length())))
00469             return -L4_MSGTOOLONG;
00470         char const *d = i->data();
00471         for (unsigned x = 0; x < i->length(); ++x)
00472             msg[offs++] = *d++;
00473     }
00474
00475     return offs;
00476 }
00477 };
00478
00479 template<
00480 struct Svr_val_ops<Varg_list_ref, Dir_in, Cls_data> :
00481     Svr_noops<Varg_list_ref>
00482 {
00483     using Svr_noops<Varg_list_ref>::to_svr;
00484     static int to_svr(char *msg, unsigned offset, unsigned limit,
00485                     Varg_list_ref &a, Dir_in, Cls_data)
00486     {
00487         unsigned start = align_to<l4_umword_t>(offset);
00488         unsigned offs;
00489         for (offs = start; offs < limit;)
00490         {
00491             unsigned noffs = align_to<l4_umword_t>(offs);
00492             if (L4_UNLIKELY(!check_size<l4_umword_t>(noffs, limit)))
00493                 break;
00494
00495             offs = noffs;
00496             Varg arg;
00497             arg.tag(*reinterpret_cast<l4_umword_t*>(msg + offs));
00498
00499             if (!arg.tag())
00500                 break;
00501
00502             offs += sizeof(l4_umword_t);
00503
00504             if (L4_UNLIKELY(!check_size<char>(offs, limit, arg.length())))
00505                 return -L4_MSGTOOLONG;
00506             offs += arg.length();
00507         }
00508
00509         a = Varg_list_ref(msg + start, msg + align_to<l4_umword_t>(offs));
00510         return offs;
00511     }
00512 };
00513 }
00514 }

```

16.501 limits

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * Copyright (C) 2026 Kernkonzept GmbH.
00004  * Author(s): Frank Mehnert <frank.mehnert@kernkonzept.com>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008
00009 #pragma once
00010 #pragma GCC system_header
00011
00012 #include <stddef.h>
00013 #include <limits.h>
00014
00015 // unfortunately we cannot include <limits>
00016 namespace L4 { namespace Types {
00017
00018     template< typename T > struct numeric_limits;
00019     template<> struct numeric_limits<bool>
00020     {
00021         static constexpr bool min() { return false; }
00022         static constexpr bool max() { return true; }
00023     };
00024     template<> struct numeric_limits<char>
00025     {
00026         static constexpr signed char min() { return SCHAR_MIN; }
00027         static constexpr signed char max() { return SCHAR_MAX; }
00028     };
00029     template<> struct numeric_limits<unsigned char>
00030     {
00031         static constexpr unsigned char min() { return 0; }
00032         static constexpr unsigned char max() { return UCHAR_MAX; }
00033     };

```

```

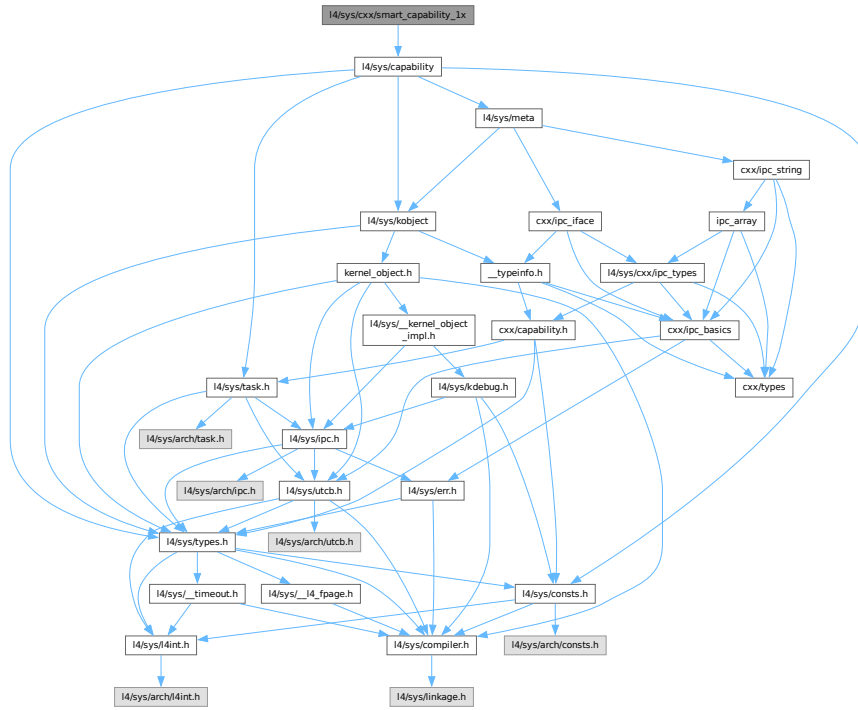
00034     template<> struct numeric_limits<signed short int>
00035     {
00036         static constexpr signed short int min() { return SHRT_MIN; }
00037         static constexpr signed short int max() { return SHRT_MAX; }
00038     };
00039     template<> struct numeric_limits<unsigned short int>
00040     {
00041         static constexpr unsigned short int min() { return 0; }
00042         static constexpr unsigned short int max() { return USHRT_MAX; }
00043     };
00044     template<> struct numeric_limits<signed int>
00045     {
00046         static constexpr signed int min() { return INT_MIN; }
00047         static constexpr signed int max() { return INT_MAX; }
00048     };
00049     template<> struct numeric_limits<unsigned int>
00050     {
00051         static constexpr unsigned int min() { return 0U; }
00052         static constexpr unsigned int max() { return UINT_MAX; }
00053     };
00054     template<> struct numeric_limits<signed long int>
00055     {
00056         static constexpr signed long min() { return LONG_MIN; }
00057         static constexpr signed long max() { return LONG_MAX; }
00058     };
00059     template<> struct numeric_limits<unsigned long int>
00060     {
00061         static constexpr unsigned long int min() { return 0UL; }
00062         static constexpr unsigned long int max() { return ULONG_MAX; }
00063     };
00064     template<> struct numeric_limits<signed long long int>
00065     {
00066         static constexpr signed long long min() { return LLONG_MIN; }
00067         static constexpr signed long long max() { return LLONG_MAX; }
00068     };
00069     template<> struct numeric_limits<unsigned long long int>
00070     {
00071         static constexpr unsigned long long min() { return 0ULL; }
00072         static constexpr unsigned long long max() { return ULLONG_MAX; }
00073     };
00074     template<> struct numeric_limits<l4_msgtag_t>
00075     {
00076         static constexpr long min() { return -max() - 1L; }
00077         static constexpr long max() { return numeric_limits<long>::max() >> 16; }
00078     };
00079
00080 }} // namespace Types, namespace L4
00081

```

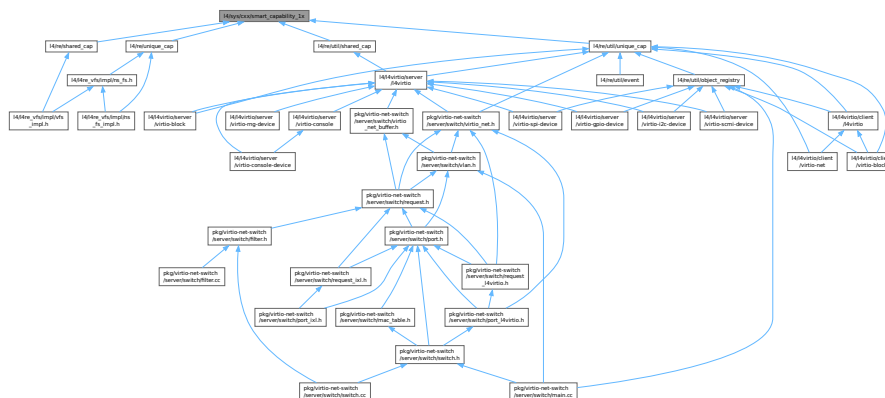
16.502 l4/sys/cxx/smart_capability_1x File Reference

```
#include <l4/sys/capability>
```

Include dependency graph for smart_capability_1x:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace [L4](#)
L4 low-level kernel interface.

16.503 smart_capability_1x

[Go to the documentation of this file.](#)

```

00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2017 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00008 #pragma once
00009
00010 #include <l4/sys/capability>
00011
00012 namespace L4 { namespace Detail {
00013
00014     template< typename T, typename IMPL >
00015     class Smart_cap_base : public Cap_base, protected IMPL
00016     {
00017     protected:
00018         template<typename X>
00019         static IMPL &impl(Smart_cap_base<X, IMPL> &o) { return o; }
00020
00021         template<typename X>
00022         static IMPL const &impl(Smart_cap_base<X, IMPL> const &o) { return o; }
00023
00024     public:
00025         template<typename X, typename I>
00026         friend class ::L4::Detail::Smart_cap_base;
00027
00028         Smart_cap_base(Smart_cap_base const &) = delete;
00029         Smart_cap_base &operator = (Smart_cap_base const &) = delete;
00030
00031         Smart_cap_base() noexcept : Cap_base(Invalid) {}
00032
00033         explicit Smart_cap_base(Cap_base::Cap_type t) noexcept
00034         : Cap_base(t)
00035         {}
00036
00037         template<typename O>
00038         explicit constexpr Smart_cap_base(Cap<O> c) noexcept
00039         : Cap_base(c.cap())
00040         {}
00041
00042         template<typename O>
00043         explicit constexpr Smart_cap_base(Cap<O> c, IMPL const &impl) noexcept
00044         : Cap_base(c.cap()), IMPL(impl)
00045         {}
00046
00047         Cap<T> release() noexcept
00048         {
00049             l4_cap_idx_t c = this->cap();
00050             IMPL::invalidate(*this);
00051             return Cap<T>(c);
00052         }
00053
00054         void reset()
00055         { IMPL::free(*this); }
00056
00057         Cap<T> operator -> () const noexcept { return Cap<T>(this->cap()); }
00058         Cap<T> get() const noexcept { return Cap<T>(this->cap()); }
00059         ~Smart_cap_base() noexcept { IMPL::free(*this); }
00060     };
00061
00062     template< typename T, typename IMPL >
00063     class Unique_cap_impl final : public Smart_cap_base<T, IMPL>
00064     {
00065     private:
00066         using Base = Smart_cap_base<T, IMPL>;
00067
00068     public:
00069         using Base::Base;
00070         Unique_cap_impl() noexcept = default;
00071
00072         Unique_cap_impl(Unique_cap_impl &&o) noexcept
00073         : Base(o.release(), Base::impl(o))
00074         {}
00075
00076         template<typename O>
00077         Unique_cap_impl(Unique_cap_impl<O, IMPL> &&o) noexcept
00078         : Base(o.release(), Base::impl(o))
00079         { Cap<T>::template check_convertible_from<O>(); }
00080
00081         Unique_cap_impl &operator = (Unique_cap_impl &&o) noexcept

```



```

00087 {
00088     if (&o == this)
00089         return *this;
00090
00091     IMPL::free(*this);
00092     this->_c = o.release().cap();
00093     this->IMPL::operator = (Base::impl(o));
00094     return *this;
00095 }
00096
00097 template<typename O>
00098 Unique_cap_impl &operator = (Unique_cap_impl<O, IMPL> &o) noexcept
00099 {
00100     Cap<T>::template check_convertible_from<O>();
00101
00102     IMPL::free(*this);
00103     this->_c = o.release().cap();
00104     this->IMPL::operator = (Base::impl(o));
00105     return *this;
00106 }
00107 };
00108
00109 template<typename T, typename IMPL>
00110 class Shared_cap_impl final : public Smart_cap_base<T, IMPL>
00111 {
00112 private:
00113     using Base = Smart_cap_base<T, IMPL>;
00114
00115 public:
00116     using Base::Base;
00117     Shared_cap_impl() noexcept = default;
00118
00119     Shared_cap_impl(Shared_cap_impl &o) noexcept
00120     : Base(o.release(), Base::impl(o))
00121     {}
00122
00123     template<typename O>
00124     Shared_cap_impl(Shared_cap_impl<O, IMPL> &o) noexcept
00125     : Base(o.release(), Base::impl(o))
00126     { Cap<T>::template check_convertible_from<O>(); }
00127
00128     template<typename O>
00129     Shared_cap_impl(Shared_cap_impl<O, IMPL> &o, Cap<T> cap) noexcept
00130     : Base(cap, Base::impl(o))
00131     { o.release(); }
00132
00133     Shared_cap_impl &operator = (Shared_cap_impl &o) noexcept
00134     {
00135         if (&o == this)
00136             return *this;
00137
00138         IMPL::free(*this);
00139         this->_c = o.release().cap();
00140         this->IMPL::operator = (Base::impl(o));
00141         return *this;
00142     }
00143
00144     template<typename O>
00145     Shared_cap_impl &operator = (Shared_cap_impl<O, IMPL> &o) noexcept
00146     {
00147         Cap<T>::template check_convertible_from<O>();
00148
00149         IMPL::free(*this);
00150         this->_c = o.release().cap();
00151         this->IMPL::operator = (Base::impl(o));
00152         return *this;
00153     }
00154
00155     Shared_cap_impl(Shared_cap_impl const &o) noexcept
00156     : Base()
00157     {
00158         this->IMPL::operator = (Base::impl(o));
00159         this->_c = IMPL::copy(o).cap();
00160     }
00161
00162     template<typename O>
00163     Shared_cap_impl(Shared_cap_impl<O, IMPL> const &o) noexcept
00164     : Base()
00165     {
00166         Cap<T>::template check_convertible_from<O>();
00167         this->IMPL::operator = (Base::impl(o));
00168         this->_c = IMPL::copy(o).cap();
00169     }
00170
00171     template<typename O>
00172     Shared_cap_impl(Shared_cap_impl<O, IMPL> const &o, Cap<T> cap) noexcept
00173     : Base()

```

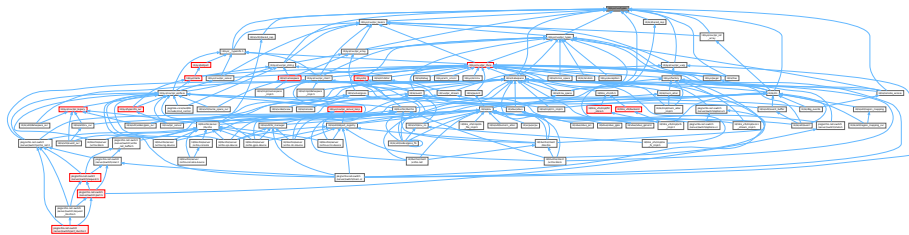
```

00174 {
00175     this->IMPL::operator = (Base::impl(o));
00176     this->_c = IMPL::copy(cap).cap();
00177 }
00178
00179 Shared_cap_impl &operator = (Shared_cap_impl const &o) noexcept
00180 {
00181     if (&o == this)
00182         return *this;
00183
00184     IMPL::free(*this);
00185     this->IMPL::operator = (static_cast<IMPL const &>(o));
00186     this->_c = this->IMPL::copy(o).cap();
00187     return *this;
00188 }
00189
00190 template<typename O>
00191 Shared_cap_impl &operator = (Shared_cap_impl<O, IMPL> const &o) noexcept
00192 {
00193     Cap<T>::template check_convertible_from<O>();
00194     IMPL::free(*this);
00195     this->IMPL::operator = (static_cast<IMPL const &>(o));
00196     this->_c = this->IMPL::copy(o).cap();
00197     return *this;
00198 }
00199 };
00200
00201 } // L4::Detail

```

16.504 I4/sys/cxx/types File Reference

This graph shows which files directly or indirectly include this file:



Data Structures

- class [L4::Types::Flags< BITS_ENUM, UNDERLYING >](#)
Template for defining typical *Flags* bitmaps.
- struct [L4::Types::Int_for_type< T >](#)
Metafunctor to get an integral type of the same size as *T*.
- struct [L4::Types::Flags_ops_t< DT >](#)
Mixin class to define a set of friend bitwise operators on *DT*.
- struct [L4::Types::Flags_t< DT, T >](#)
Template type to define a flags type with bitwise operations.
- struct [L4::Types::__Add_rvalue_reference_helper< T, typename >](#)
Helper template for *Add_rvalue_reference*.
- struct [L4::Types::__Add_rvalue_reference_helper< T, Void< T && > >](#)
Helper template for *Add_rvalue_reference*.
- struct [L4::Types::Add_rvalue_reference< T >](#)
Create an rvalue reference of the given type.
- struct [L4::Types::Bool< V >](#)
Boolean meta type.

- struct [L4::Types::False](#)
False meta value.
- struct [L4::Types::True](#)
True meta value.
- struct [L4::Types::Integral_constant< T, Value >](#)
Wrapper for a static constant of the given type.
- struct [L4::Types::Is_enum< T >](#)
Check whether the given type is an enumeration type.
- struct [L4::Types::__Underlying_type_helper< T, bool >](#)
Helper template for [Underlying_type](#).
- struct [L4::Types::__Underlying_type_helper< T, false >](#)
Helper template for [Underlying_type](#).
- struct [L4::Types::Underlying_type< T >](#)
Get an underlying type of an enumeration type.
- struct [L4::Types::Same< A, B >](#)
Compare two data types for equality.
- struct [L4::Types::Same_template< T, Template >](#)
Check if a type T is an instantiation of a given template.
- struct [Enum_bitops::Has_marker< typename, typename >](#)
Marker for the opt-in ADL function.
- struct [Enum_bitops::Has_marker< T, Void< decltype\(enum_bitops_enable\(declval< T >\(\)\)\)> >](#)
Marker for the opt-in ADL function.
- struct [Enum_bitops::Enable< T >](#)
Check whether the given enum type opts in for the bitwise operators.

Namespaces

- namespace [L4](#)
L4 low-level kernel interface.
- namespace [L4::Types](#)
L4 basic type helpers for C++.
- namespace [Enum_bitops](#)
Mechanism to opt-in for enum bitwise operators.
- namespace [Enum_bitops_impl](#)
Bitwise operators on enumeration types.

Typedefs

- `template<typename...>`
`using L4::Types::Void = void`
Map a sequence of any types to the void type.
- `template<typename T>`
`using L4::Types::Add_rvalue_reference_t = typename Add_rvalue_reference<T>::type`
Helper type for the [Add_rvalue_reference](#).
- `template<typename T>`
`using L4::Types::Underlying_type_t = typename Underlying_type<T>::type`
Helper type for [Underlying_type](#).
- `template<bool Condition, typename T = void>`
`using L4::Types::Enable_if_t = typename Enable_if<Condition, T>::type`
Helper type for [Enable_if](#).

Functions

- `template<typename T>
Add_value_reference_t< T > L4::Types::declval () noexcept`
Template for writing typed expressions in unevaluated contexts.
- `template<typename T, typename = L4::Types::Enable_if_t<L4::Types::Is_enum<T>::value>>
constexpr L4::Types::Underlying_type_t< T > Enum_bitops_impl::to_underlying (T const arg) noexcept`
Convert enum value to its underlying type value.
- `template<typename T, typename = L4::Types::Enable_if_t<Enum_bitops::Enable<T>::value>>
constexpr T Enum_bitops_impl::operator~ (T const a) noexcept`
Negate enum value.
- `template<typename T, typename = L4::Types::Enable_if_t<Enum_bitops::Enable<T>::value>>
constexpr T Enum_bitops_impl::operator& (T l, T r) noexcept`
Intersect enum values.
- `template<typename T, typename = L4::Types::Enable_if_t<Enum_bitops::Enable<T>::value>>
constexpr T & Enum_bitops_impl::operator&= (T &a, T const b) noexcept`
Intersect and assign enum values.
- `template<typename T, typename = L4::Types::Enable_if_t<Enum_bitops::Enable<T>::value>>
constexpr T Enum_bitops_impl::operator| (T l, T r) noexcept`
Union enum values.
- `template<typename T, typename = L4::Types::Enable_if_t<Enum_bitops::Enable<T>::value>>
constexpr T & Enum_bitops_impl::operator|= (T &a, T const b) noexcept`
Union and assign enum values.
- `template<typename T, typename = L4::Types::Enable_if_t<Enum_bitops::Enable<T>::value>>
constexpr T Enum_bitops_impl::operator- (T l, T r) noexcept`
Difference (intersect with negation, clear bits) enum values.
- `template<typename T, typename = L4::Types::Enable_if_t<Enum_bitops::Enable<T>::value>>
constexpr T & Enum_bitops_impl::operator-= (T &a, T const b) noexcept`
Difference (intersect with negation, clear bits) and assign enum values.

16.505 types

[Go to the documentation of this file.](#)

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00008
00009
00010 #pragma once
00011
00012 // very simple type traits for basic L4 functions, for a more complete set
00013 // use <l4/cxx/type_traits> or the standard <type_traits>.
00014
00015 namespace L4 {
00016
00020 namespace Types {
00021
00051     template<typename BITS_ENUM, typename UNDERLYING = unsigned long>
00052     class Flags
00053     {
00054     public:
00056         using value_type = UNDERLYING;
00058         using bits_enum_type = BITS_ENUM;
00060         using type = Flags<BITS_ENUM, UNDERLYING>;
00061
00062     private:
00063         value_type _v;
00064         explicit constexpr Flags(value_type v) noexcept : _v(v) {}
00065
00066     public:
```

```

00068     enum None_type { None };
00069
00078     constexpr Flags(None_type) noexcept : _v(0) {}
00079
00081     constexpr Flags() noexcept : _v(0) {}
00082
00091     constexpr Flags(BITS_ENUM e) noexcept
00092     : _v(static_cast<value_type>(1) << e)
00093     {}
00094
00100     static constexpr type from_raw(value_type v) noexcept { return type(v); }
00101
00103     explicit constexpr operator bool () const noexcept
00104     { return _v != 0; }
00105
00107     constexpr bool operator ! () const noexcept { return _v == 0; }
00108
00110     friend constexpr type operator | (type lhs, type rhs) noexcept
00111     { return type(lhs._v | rhs._v); }
00112
00114     friend constexpr type operator | (type lhs, bits_enum_type rhs) noexcept
00115     { return lhs | type(rhs); }
00116
00118     friend constexpr type operator & (type lhs, type rhs) noexcept
00119     { return type(lhs._v & rhs._v); }
00120
00122     friend constexpr type operator & (type lhs, bits_enum_type rhs) noexcept
00123     { return lhs & type(rhs); }
00124
00126     constexpr type &operator |= (type rhs) noexcept
00127     { _v |= rhs._v; return *this; }
00128
00130     constexpr type &operator |= (bits_enum_type rhs)
00131     { return operator |= (type(rhs)); }
00132
00134     constexpr type &operator &= (type rhs) { _v &= rhs._v; return *this; }
00135
00137     constexpr type &operator &= (bits_enum_type rhs)
00138     { return operator &= (type(rhs)); }
00139
00141     constexpr type operator ~ () const noexcept { return type(~_v); }
00142
00149     constexpr type &clear(bits_enum_type flag) noexcept
00150     { return operator &= (~type(flag)); }
00151
00153     constexpr value_type as_value() const noexcept { return _v; }
00154 };
00155
00161     template<unsigned SIZE, bool = true> struct Int_for_size;
00162
00163     template<> struct Int_for_size<sizeof(unsigned char), true>
00164     { using type = unsigned char; };
00165
00166     template<> struct Int_for_size<sizeof(unsigned short),
00167                                   (sizeof(unsigned short) > sizeof(unsigned char))>
00168     { using type = unsigned short; };
00169
00170     template<> struct Int_for_size<sizeof(unsigned),
00171                                   (sizeof(unsigned) > sizeof(unsigned short))>
00172     { using type = unsigned; };
00173
00174     template<> struct Int_for_size<sizeof(unsigned long),
00175                                   (sizeof(unsigned long) > sizeof(unsigned))>
00176     { using type = unsigned long; };
00177
00178     template<> struct Int_for_size<sizeof(unsigned long long),
00179                                   (sizeof(unsigned long long) > sizeof(unsigned long))>
00180     { using type = unsigned long long; };
00181
00188     template<typename T> struct Int_for_type
00189     {
00193         using type = typename Int_for_size<sizeof(T)>::type;
00194     };
00195
00202     template<typename DT>
00203     struct Flags_ops_t
00204     {
00206         friend constexpr DT operator | (DT l, DT r)
00207         { return DT(l.raw | r.raw); }
00208
00210         friend constexpr DT operator & (DT l, DT r)
00211         { return DT(l.raw & r.raw); }
00212
00214         friend constexpr DT operator ~ (DT l, DT r)
00215         { return DT(l.raw & ~r.raw); }
00216
00218         friend constexpr bool operator == (DT l, DT r)

```

```

00219     { return l.raw == r.raw; }
00220
00222     friend constexpr bool operator != (DT l, DT r)
00223     { return l.raw != r.raw; }
00224
00226     constexpr DT &operator |= (DT const r)
00227     {
00228         static_cast<DT *>(this)->raw |= r.raw;
00229         return *static_cast<DT *>(this);
00230     }
00231
00233     constexpr DT &operator &= (DT const r)
00234     {
00235         static_cast<DT *>(this)->raw &= r.raw;
00236         return *static_cast<DT *>(this);
00237     }
00238
00240     constexpr DT &operator -= (DT const r)
00241     {
00242         static_cast<DT *>(this)->raw &= ~r.raw;
00243         return *static_cast<DT *>(this);
00244     }
00245
00247     explicit constexpr operator bool () const
00248     { return static_cast<DT const *>(this)->raw != 0; }
00249
00251     constexpr DT operator ~ () const
00252     { return DT(~static_cast<DT const *>(this)->raw); }
00253 };
00254
00263     template<typename DT, typename T>
00264     struct Flags_t : Flags_ops_t<Flags_t<DT, T>
00265     {
00266         T raw = 0;
00269         constexpr Flags_t() = default;
00271         constexpr Flags_t(DT f) : raw(static_cast<T>(f)) {}
00273         explicit constexpr Flags_t(T f) : raw(f) {}
00274
00276         static constexpr Flags_t None = Flags_t();
00277     };
00278
00280     template<typename...> using Void = void;
00281
00283     template<typename T, typename = void> struct __Add_rvalue_reference_helper
00284     { using type = T; };
00285
00287     template<typename T> struct __Add_rvalue_reference_helper<T, Void<T &&>>
00288     { using type = T && };
00289
00291     template<typename T> struct Add_rvalue_reference
00292     { using type = typename __Add_rvalue_reference_helper<T>::type; };
00293
00295     template<typename T> using Add_rvalue_reference_t
00296     = typename Add_rvalue_reference<T>::type;
00297
00306     template<typename T> Add_rvalue_reference_t<T> declval() noexcept;
00307
00313     template<bool V > struct Bool
00314     {
00315         using type = Bool<V>;
00316         enum { value = V };
00317     };
00318
00321     struct False : Bool<false> {};
00322
00325     struct True : Bool<true> {};
00326
00328     template<typename T, T Value>
00329     struct Integral_constant
00330     {
00331         static T const value = Value;
00332
00333         typedef T value_type;
00334         typedef Integral_constant<T, Value> type;
00335     };
00336
00348     template<typename T>
00349     struct Is_enum : Integral_constant<bool, __is_enum(T)> {};
00350
00362     template<typename T, bool = Is_enum<T>::value>
00363     struct __Underlying_type_helper { using type = __underlying_type(T); };
00364
00366     template<typename T> struct __Underlying_type_helper<T, false> {};
00367
00369     template<typename T> struct Underlying_type
00370     : public __Underlying_type_helper<T> {};
00371

```

```

00373     template<typename T> using Underlying_type_t
00374         = typename Underlying_type<T>::type;
00375
00376     /*****
00385     template<typename A, typename B>
00386     struct Same : False {};
00387
00388     template<typename A>
00389     struct Same<A, A> : True {};
00390
00391     template< typename T1, typename T2 >
00392     inline constexpr bool Same_v = Same<T1, T2>::value;
00393
00395     template <typename T, template <typename...> typename Template>
00396     struct Same_template : False {};
00397
00398     template <template <typename...> typename Template, typename... Args>
00399     struct Same_template<Template<Args...>, Template> : True {};
00400
00401     template <typename T, template <typename...> typename Template>
00402     inline constexpr bool Same_template_v = Same_template<T, Template>::value;
00403
00404     template<bool, typename = void> struct Enable_if {};
00405     template<typename T> struct Enable_if<true, T> { using type = T; };
00406
00408     template<bool Condition, typename T = void>
00409         using Enable_if_t = typename Enable_if<Condition, T>::type;
00410
00411     template<typename T1, typename T2, typename T = void>
00412     struct Enable_if_same : Enable_if<Same_v<T1, T2>, T> {};
00413
00414     template<typename T> struct Remove_const { using type = T; };
00415     template<typename T> struct Remove_const<T const> { using type = T; };
00416     template<typename T> using Remove_const_t = typename Remove_const<T>::type;
00417
00418     template<typename T> struct Remove_volatile { using type = T; };
00419     template<typename T> struct Remove_volatile<T volatile> { using type = T; };
00420     template<typename T> using Remove_volatile_t = typename Remove_volatile<T>::type;
00421
00422     template<typename T> struct Remove_cv
00423     { using type = Remove_const_t<Remove_volatile_t<T>>; };
00424     template<typename T> using Remove_cv_t = typename Remove_cv<T>::type;
00425
00426     template<typename T> struct Remove_pointer { using type = T; };
00427     template<typename T> struct Remove_pointer<T*> { using type = T; };
00428     template<typename T> using Remove_pointer_t = typename Remove_pointer<T>::type;
00429
00430     template<typename T> struct Remove_reference { using type = T; };
00431     template<typename T> struct Remove_reference<T&> { using type = T; };
00432     template<typename T> using Remove_reference_t = typename Remove_reference<T>::type;
00433
00434     template<typename T> struct Remove_pr { using type = T; };
00435     template<typename T> struct Remove_pr<T&> { using type = T; };
00436     template<typename T> struct Remove_pr<T*> { using type = T; };
00437     template<typename T> using Remove_pr_t = typename Remove_pr<T>::type;
00438
00439     template<typename T>
00440     constexpr T &&
00441     forward(Remove_reference_t<T> &t)
00442     { return static_cast<T &&>(t); }
00443
00444     template<typename T>
00445     constexpr T &&
00446     forward(Remove_reference_t<T> &&t)
00447     { return static_cast<T &&>(t); }
00448
00449     template<typename T>
00450     constexpr Remove_reference_t<T> &&
00451     move(T &&t) { return static_cast<Remove_reference_t<T> &&>(t); }
00452
00453     template< typename... > using Void_t = void;
00454 } // Types
00455
00456 } // L4
00457
00469 namespace Enum_bitops {
00470     using namespace L4::Types;
00471
00473     template<typename, typename = void> struct Has_marker : False {};
00474
00476     template<typename T>
00477     struct Has_marker<T, Void<decltype(enum_bitops_enable(declval<T>()))>>
00478         : True {};
00479
00481     template<typename T>
00482     struct Enable
00483         : Integral_constant<bool, Is_enum<T>::value && Has_marker<T>::value> {};

```

```

00484 } // Enum_bitops
00485
00497 inline namespace Enum_bitops_impl {
00499     template<typename T,
00500             typename = L4::Types::Enable_if_t<L4::Types::Is_enum<T>::value>
00501             constexpr L4::Types::Underlying_type_t<T> to_underlying(T const arg) noexcept
00502             { return static_cast<L4::Types::Underlying_type_t<T>>(arg); }
00503
00505     template<typename T,
00506             typename = L4::Types::Enable_if_t<Enum_bitops::Enable<T>::value>
00507             constexpr T operator ~ (T const a) noexcept
00508             { return static_cast<T>(~to_underlying(a)); }
00509
00511     template<typename T,
00512             typename = L4::Types::Enable_if_t<Enum_bitops::Enable<T>::value>
00513             constexpr T operator & (T l, T r) noexcept
00514             { return static_cast<T>(to_underlying(l) & to_underlying(r)); }
00515
00517     template<typename T,
00518             typename = L4::Types::Enable_if_t<Enum_bitops::Enable<T>::value>
00519             constexpr T &operator &= (T &a, T const b) noexcept
00520             {
00521                 a = a & b;
00522                 return a;
00523             }
00524
00526     template<typename T,
00527             typename = L4::Types::Enable_if_t<Enum_bitops::Enable<T>::value>
00528             constexpr T operator | (T l, T r) noexcept
00529             { return static_cast<T>(to_underlying(l) | to_underlying(r)); }
00530
00532     template<typename T,
00533             typename = L4::Types::Enable_if_t<Enum_bitops::Enable<T>::value>
00534             constexpr T &operator |= (T &a, T const b) noexcept
00535             {
00536                 a = a | b;
00537                 return a;
00538             }
00539
00541     template<typename T,
00542             typename = L4::Types::Enable_if_t<Enum_bitops::Enable<T>::value>
00543             constexpr T operator - (T l, T r) noexcept
00544             { return static_cast<T>(to_underlying(l) & ~to_underlying(r)); }
00545
00547     template<typename T,
00548             typename = L4::Types::Enable_if_t<Enum_bitops::Enable<T>::value>
00549             constexpr T &operator -= (T &a, T const b) noexcept
00550             {
00551                 a = a - b;
00552                 return a;
00553             }
00554 } // Enum_bitops_impl

```

16.506 l4/sys/debugger File Reference

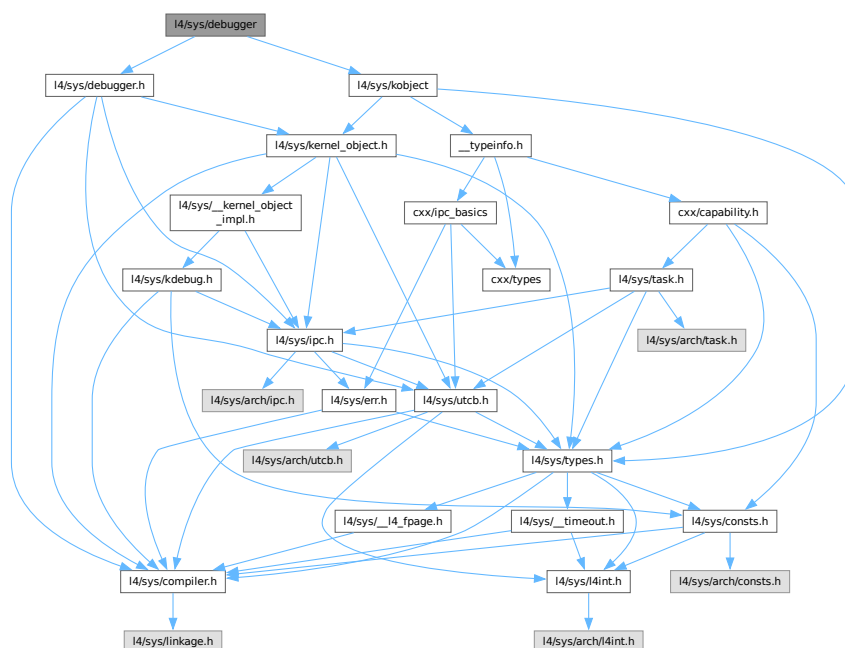
The debugger interface specifies common debugging related definitions.

```

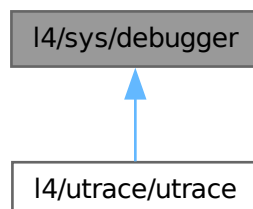
#include <l4/sys/debugger.h>
#include <l4/sys/kobject>

```


Include dependency graph for debugger:



This graph shows which files directly or indirectly include this file:



Data Structures

- class [L4::Debugger](#)
C++ kernel debugger API.

Namespaces

- namespace [L4](#)
L4 low-level kernel interface.

16.506.1 Detailed Description

The debugger interface specifies common debugging related definitions.

Definition in file [debugger](#).

16.507 debugger

[Go to the documentation of this file.](#)

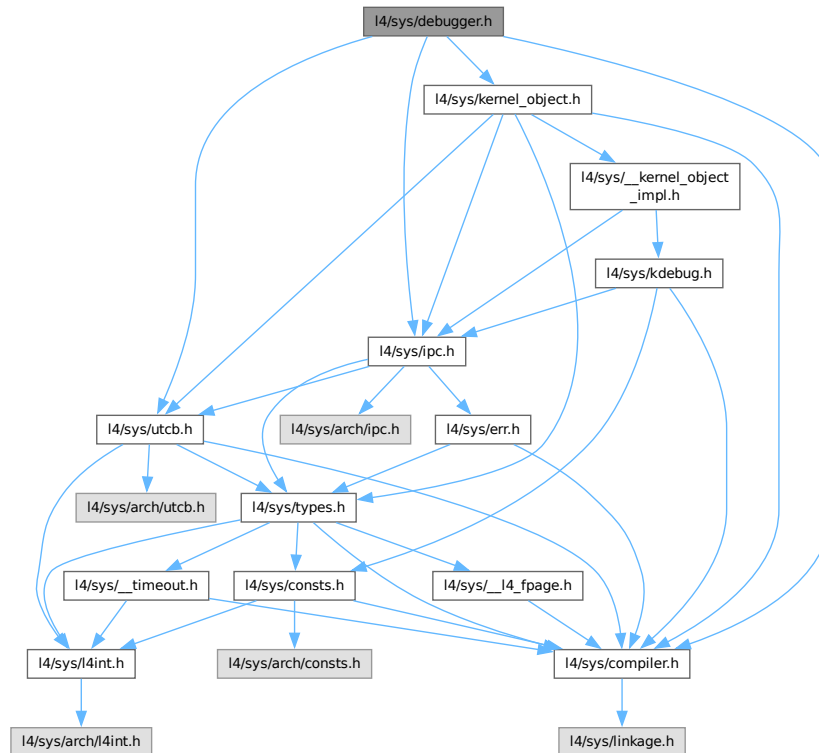
```

00001 // vi:set ft=cpp: -- Mode: C++ --
00006 /*
00007  * (c) 2010-2011 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013 #pragma once
00014
00015 #include <l4/sys/debugger.h>
00016 #include <l4/sys/kobject>
00017
00018 namespace L4 {
00019
00042 class Debugger : public Kobject_t<Debugger, Kobject, L4_PROTO_DEBUGGER>
00043 {
00044 public:
00045     enum
00046     {
00047         Switch_log_on  = L4_DEBUGGER_SWITCH_LOG_ON,
00048         Switch_log_off = L4_DEBUGGER_SWITCH_LOG_OFF,
00049     };
00050
00059     l4_msgtag_t set_object_name(const char *name,
00060                                l4_utcb_t *utcb = l4_utcb()) noexcept
00061     { return l4_debugger_set_object_name_u(cap(), name, utcb); }
00062
00071     unsigned long global_id(l4_utcb_t *utcb = l4_utcb()) noexcept
00072     { return l4_debugger_global_id_u(cap(), utcb); }
00073
00083     unsigned long kobj_to_id(l4_addr_t kobjp,
00084                              l4_utcb_t *utcb = l4_utcb()) noexcept
00085     { return l4_debugger_kobj_to_id_u(cap(), kobjp, utcb); }
00086
00097     l4_ret_t query_log_typeid(const char *name, unsigned idx,
00098                               l4_utcb_t *utcb = l4_utcb()) noexcept
00099     { return l4_debugger_query_log_typeid_u(cap(), name, idx, utcb); }
00100
00116     l4_ret_t query_log_name(unsigned idx,
00117                              char *name, unsigned namelen,
00118                              char *shortname, unsigned shortnamelen,
00119                              l4_utcb_t *utcb = l4_utcb()) noexcept
00120     {
00121         return l4_debugger_query_log_name_u(cap(), idx, name, namelen,
00122                                             shortname, shortnamelen, utcb);
00123     }
00124
00133     l4_msgtag_t switch_log(const char *name, unsigned on_off,
00134                             l4_utcb_t *utcb = l4_utcb()) noexcept
00135     { return l4_debugger_switch_log_u(cap(), name, on_off, utcb); }
00136
00148     l4_msgtag_t query_object_name(unsigned id, char *name, unsigned size,
00149                                    l4_utcb_t *utcb = l4_utcb()) noexcept
00150     { return l4_debugger_query_object_name_u(cap(), id, name, size, utcb); }
00151
00162     l4_msgtag_t get_object_name(char *name, unsigned size,
00163                                 l4_utcb_t *utcb = l4_utcb()) noexcept
00164     { return l4_debugger_get_object_name_u(cap(), name, size, utcb); }
00165
00175     l4_msgtag_t add_image_info(l4_addr_t base, const char *name,
00176                                l4_utcb_t *utcb = l4_utcb()) noexcept
00177     { return l4_debugger_add_image_info_u(cap(), base, name, utcb); }
00178 };
00179 }
```

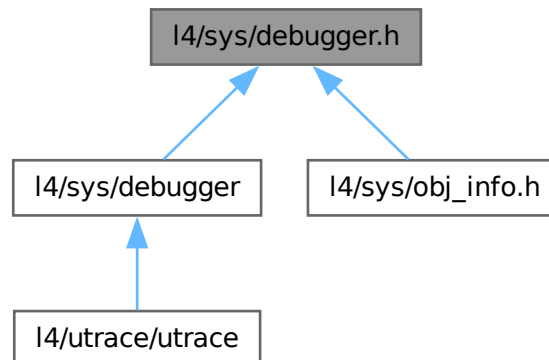
16.508 l4/sys/debugger.h File Reference

Debugger related definitions.

```
#include <l4/sys/compiler.h>
#include <l4/sys/utcb.h>
#include <l4/sys/ipc.h>
#include <l4/sys/kernel_object.h>
Include dependency graph for debugger.h:
```



This graph shows which files directly or indirectly include this file:



Enumerations

- enum {
[L4_DEBUGGER_KOBJ_SET_NAME_OP](#) = 0UL , [L4_DEBUGGER_KOBJ_GET_GLOBAL_ID_OP](#) = 1UL ,
[L4_DEBUGGER_KOBJ_PTR_GET_GLOBAL_ID_OP](#) = 2UL , [L4_DEBUGGER_LOG_QUERY_TYPEID_OP](#)
 = 3UL ,
[L4_DEBUGGER_LOG_SWITCH_OP](#) = 4UL , [L4_DEBUGGER_GLOBAL_ID_GET_NAME_OP](#) = 5UL ,
[L4_DEBUGGER_LOG_QUERY_NAME_OP](#) = 6UL , [L4_DEBUGGER_TASK_ADD_IMAGE_INFO_OP](#) =
 7UL ,
[L4_DEBUGGER_KOBJ_GET_NAME_OP](#) = 8UL , [L4_DEBUGGER_OBJ_INFO_OP](#) = 16UL }

Functions

- [l4_msgtag_t l4_debugger_set_object_name](#) ([l4_cap_idx_t](#) cap, const char *name) [L4_NOTHROW](#)
 Set the name of a kernel object.
- [l4_msgtag_t l4_debugger_query_object_name](#) ([l4_cap_idx_t](#) cap, unsigned id, char *name, unsigned size)
[L4_NOTHROW](#)
 Get name of the kernel object with Id *id*.
- [l4_msgtag_t l4_debugger_get_object_name](#) ([l4_cap_idx_t](#) cap, char *name, unsigned size) [L4_NOTHROW](#)
 Get name of a kernel object.
- unsigned long [l4_debugger_global_id](#) ([l4_cap_idx_t](#) cap) [L4_NOTHROW](#)
 Get the globally unique ID of the object behind a capability.
- unsigned long [l4_debugger_kobj_to_id](#) ([l4_cap_idx_t](#) cap, [l4_addr_t](#) kobjp) [L4_NOTHROW](#)
 Get the globally unique ID of the object behind the kobject pointer.
- [l4_ret_t l4_debugger_query_log_typeid](#) ([l4_cap_idx_t](#) cap, const char *name, unsigned idx) [L4_NOTHROW](#)
 Query the log-id for a log type.
- [l4_ret_t l4_debugger_query_log_name](#) ([l4_cap_idx_t](#) cap, unsigned idx, char *name, unsigned namelen,
 char *shortname, unsigned shortnamelen) [L4_NOTHROW](#)
 Query the name of a log type given the ID.
- [l4_msgtag_t l4_debugger_switch_log](#) ([l4_cap_idx_t](#) cap, const char *name, int on_off) [L4_NOTHROW](#)
 Set or unset log.
- [l4_msgtag_t l4_debugger_add_image_info](#) ([l4_cap_idx_t](#) cap, [l4_addr_t](#) base, const char *name)
[L4_NOTHROW](#)
 Add loaded image information for a task.

16.508.1 Detailed Description

Debugger related definitions.

Definition in file [debugger.h](#).

16.508.2 Enumeration Type Documentation

16.508.2.1 anonymous enum

anonymous enum

Enumerator

| | |
|---------------------------------------|--|
| L4_DEBUGGER_KOBJ_SET_NAME_OP | Set debug name of kernel object. |
| L4_DEBUGGER_KOBJ_GET_GLOBAL_ID_OP | Get debug ID of kernel object. |
| L4_DEBUGGER_KOBJ_PTR_GET_GLOBAL_ID_OP | Get debug ID of kernel object by pointer. |
| L4_DEBUGGER_LOG_QUERY_TYPEID_OP | Query log-id for log type. |
| L4_DEBUGGER_LOG_SWITCH_OP | Enable / disable log. |
| L4_DEBUGGER_GLOBAL_ID_GET_NAME_OP | Get debug name of kernel object by debug ID. |
| L4_DEBUGGER_LOG_QUERY_NAME_OP | Get name of log type for given log-id. |
| L4_DEBUGGER_TASK_ADD_IMAGE_INFO_OP | Add image information for task. |
| L4_DEBUGGER_KOBJ_GET_NAME_OP | Get debug name of kernel object. |
| L4_DEBUGGER_OBJ_INFO_OP | Query information about all kernel objects. |

Definition at line 234 of file [debugger.h](#).

16.509 debugger.h

[Go to the documentation of this file.](#)

```

00001 #pragma once
00007 /*
00008  * (c) 2008-2011 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014
00015 #include <l4/sys/compiler.h>
00016 #include <l4/sys/utcb.h>
00017 #include <l4/sys/ipc.h>
00018
00031
00042 L4_INLINE l4_msgtag_t
00043 l4_debugger_set_object_name(l4_cap_idx_t cap, const char *name) L4_NOTHROW;
00044
00048 L4_INLINE l4_msgtag_t
00049 l4_debugger_set_object_name_u(l4_cap_idx_t cap, const char *name, l4_utcb_t *utcb) L4_NOTHROW;
00050
00063 L4_INLINE l4_msgtag_t
00064 l4_debugger_query_object_name(l4_cap_idx_t cap, unsigned id,
00065                               char *name, unsigned size) L4_NOTHROW;
00066
00070 L4_INLINE l4_msgtag_t
00071 l4_debugger_query_object_name_u(l4_cap_idx_t cap, unsigned id,

```

```

00072             char *name, unsigned size,
00073             l4_utcb_t *utcb) L4_NOTHROW;
00074
00086 L4_INLINE l4_msgtag_t
00087 l4_debugger_get_object_name(l4_cap_idx_t cap,
00088             char *name, unsigned size) L4_NOTHROW;
00089
00093 L4_INLINE l4_msgtag_t
00094 l4_debugger_get_object_name_u(l4_cap_idx_t cap,
00095             char *name, unsigned size,
00096             l4_utcb_t *utcb) L4_NOTHROW;
00097
00109 L4_INLINE unsigned long
00110 l4_debugger_global_id(l4_cap_idx_t cap) L4_NOTHROW;
00111
00115 L4_INLINE unsigned long
00116 l4_debugger_global_id_u(l4_cap_idx_t cap, l4_utcb_t *utcb) L4_NOTHROW;
00117
00130 L4_INLINE unsigned long
00131 l4_debugger_kobj_to_id(l4_cap_idx_t cap, l4_addr_t kobjp) L4_NOTHROW;
00132
00136 L4_INLINE unsigned long
00137 l4_debugger_kobj_to_id_u(l4_cap_idx_t cap, l4_addr_t kobjp, l4_utcb_t *utcb) L4_NOTHROW;
00138
00151 L4_INLINE l4_ret_t
00152 l4_debugger_query_log_typeid(l4_cap_idx_t cap, const char *name,
00153             unsigned idx) L4_NOTHROW;
00154
00158 L4_INLINE l4_ret_t
00159 l4_debugger_query_log_typeid_u(l4_cap_idx_t cap, const char *name,
00160             unsigned idx, l4_utcb_t *utcb) L4_NOTHROW;
00161
00178 L4_INLINE l4_ret_t
00179 l4_debugger_query_log_name(l4_cap_idx_t cap, unsigned idx,
00180             char *name, unsigned namelen,
00181             char *shortname, unsigned shortnamelen) L4_NOTHROW;
00182
00186 L4_INLINE l4_ret_t
00187 l4_debugger_query_log_name_u(l4_cap_idx_t cap, unsigned idx,
00188             char *name, unsigned namelen,
00189             char *shortname, unsigned shortnamelen,
00190             l4_utcb_t *utcb) L4_NOTHROW;
00191
00202 L4_INLINE l4_msgtag_t
00203 l4_debugger_switch_log(l4_cap_idx_t cap, const char *name,
00204             int on_off) L4_NOTHROW;
00205
00209 L4_INLINE l4_msgtag_t
00210 l4_debugger_switch_log_u(l4_cap_idx_t cap, const char *name, int on_off,
00211             l4_utcb_t *utcb) L4_NOTHROW;
00212
00223 L4_INLINE l4_msgtag_t
00224 l4_debugger_add_image_info(l4_cap_idx_t cap, l4_addr_t base,
00225             const char *name) L4_NOTHROW;
00226
00230 L4_INLINE l4_msgtag_t
00231 l4_debugger_add_image_info_u(l4_cap_idx_t cap, l4_addr_t base, const char *name,
00232             l4_utcb_t *utcb) L4_NOTHROW;
00233
00234 enum
00235 {
00237     L4_DEBUGGER_KOBJ_SET_NAME_OP = 0UL,
00239     L4_DEBUGGER_KOBJ_GET_GLOBAL_ID_OP = 1UL,
00241     L4_DEBUGGER_KOBJ_PTR_GET_GLOBAL_ID_OP = 2UL,
00243     L4_DEBUGGER_LOG_QUERY_TYPEID_OP = 3UL,
00245     L4_DEBUGGER_LOG_SWITCH_OP = 4UL,
00247     L4_DEBUGGER_GLOBAL_ID_GET_NAME_OP = 5UL,
00249     L4_DEBUGGER_LOG_QUERY_NAME_OP = 6UL,
00251     L4_DEBUGGER_TASK_ADD_IMAGE_INFO_OP = 7UL,
00253     L4_DEBUGGER_KOBJ_GET_NAME_OP = 8UL,
00254
00256     L4_DEBUGGER_OBJ_INFO_OP = 16UL,
00257 };
00258
00259 enum
00260 {
00261     L4_DEBUGGER_SWITCH_LOG_ON = 1,
00262     L4_DEBUGGER_SWITCH_LOG_OFF = 0,
00263 };
00264
00265 /* IMPLEMENTATION ----- */
00266
00267 #include <l4/sys/kernel_object.h>
00268
00282 L4_INLINE unsigned
00283 __strcpy_maxlen(char *dst, char const *src, unsigned maxlen)
00284 {

```

```

00285     unsigned i;
00286     if (!maxlen)
00287         return 0;
00288
00289     for (i = 0; i < maxlen - 1 && src[i]; ++i)
00290         dst[i] = src[i];
00291     dst[i] = '\0';
00292
00293     return i + 1;
00294 }
00295
00296 L4_INLINE l4_msgtag_t
00297 l4_debugger_set_object_name_u(l4_cap_idx_t cap,
00298                               const char *name, l4_utcb_t *utcb) L4_NOTHROW
00299 {
00300     unsigned i;
00301     l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_KOBJ_SET_NAME_OP;
00302     i = __strcpy_maxlen((char *)&l4_utcb_mr_u(utcb)->mr[1], name,
00303                        (L4_UTCB_GENERIC_DATA_SIZE - 2) * sizeof(l4_umword_t));
00304     i = l4_bytes_to_mwords(i);
00305     return l4_invoke_debugger(cap, l4_msgtag(0, 1 + i, 0, 0), utcb);
00306 }
00307
00308 L4_INLINE unsigned long
00309 l4_debugger_global_id_u(l4_cap_idx_t cap, l4_utcb_t *utcb) L4_NOTHROW
00310 {
00311     l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_KOBJ_GET_GLOBAL_ID_OP;
00312     if (l4_error_u(l4_invoke_debugger(cap, l4_msgtag(0, 1, 0, 0), utcb), utcb))
00313         return ~0UL;
00314     return l4_utcb_mr_u(utcb)->mr[0];
00315 }
00316
00317 L4_INLINE unsigned long
00318 l4_debugger_kobj_to_id_u(l4_cap_idx_t cap, l4_addr_t kobjp, l4_utcb_t *utcb) L4_NOTHROW
00319 {
00320     l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_KOBJ_PTR_GET_GLOBAL_ID_OP;
00321     l4_utcb_mr_u(utcb)->mr[1] = kobjp;
00322     if (l4_error_u(l4_invoke_debugger(cap, l4_msgtag(0, 2, 0, 0), utcb), utcb))
00323         return ~0UL;
00324     return l4_utcb_mr_u(utcb)->mr[0];
00325 }
00326
00327 L4_INLINE l4_ret_t
00328 l4_debugger_query_log_typeid_u(l4_cap_idx_t cap, const char *name,
00329                                unsigned idx,
00330                                l4_utcb_t *utcb) L4_NOTHROW
00331 {
00332     unsigned i;
00333     l4_ret_t e;
00334     l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_LOG_QUERY_TYPEID_OP;
00335     l4_utcb_mr_u(utcb)->mr[1] = idx;
00336     i = __strcpy_maxlen((char *)&l4_utcb_mr_u(utcb)->mr[2], name, 32);
00337     i = l4_bytes_to_mwords(i);
00338     e = l4_error_u(l4_invoke_debugger(cap, l4_msgtag(0, 2 + i, 0, 0), utcb), utcb);
00339     if (e < 0)
00340         return e;
00341     return l4_utcb_mr_u(utcb)->mr[0];
00342 }
00343
00344 L4_INLINE l4_ret_t
00345 l4_debugger_query_log_name_u(l4_cap_idx_t cap, unsigned idx,
00346                              char *name, unsigned namelen,
00347                              char *shortname, unsigned shortnamelen,
00348                              l4_utcb_t *utcb) L4_NOTHROW
00349 {
00350     l4_ret_t e;
00351     char const *n;
00352     l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_LOG_QUERY_NAME_OP;
00353     l4_utcb_mr_u(utcb)->mr[1] = idx;
00354     e = l4_error_u(l4_invoke_debugger(cap, l4_msgtag(0, 2, 0, 0), utcb), utcb);
00355     if (e < 0)
00356         return e;
00357     n = (char const *)&l4_utcb_mr_u(utcb)->mr[0];
00358     __strcpy_maxlen(name, n, namelen);
00359     __strcpy_maxlen(shortname, n + __builtin_strlen(n) + 1, shortnamelen);
00360     return 0;
00361 }
00362
00363
00364 L4_INLINE l4_msgtag_t
00365 l4_debugger_switch_log_u(l4_cap_idx_t cap, const char *name, int on_off,
00366                          l4_utcb_t *utcb) L4_NOTHROW
00367 {
00368     unsigned i;
00369     l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_LOG_SWITCH_OP;
00370     l4_utcb_mr_u(utcb)->mr[1] = on_off;
00371     i = __strcpy_maxlen((char *)&l4_utcb_mr_u(utcb)->mr[2], name, 32);

```

```

00372     i = l4_bytes_to_mwords(i);
00373     return l4_invoke_debugger(cap, l4_msgtag(0, 2 + i, 0, 0), utcb);
00374 }
00375
00376 L4_INLINE l4_msgtag_t
00377 l4_debugger_query_object_name_u(l4_cap_idx_t cap, unsigned id,
00378                                char *name, unsigned size,
00379                                l4_utcb_t *utcb) L4_NOTHROW
00380 {
00381     l4_msgtag_t t;
00382     l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_GLOBAL_ID_GET_NAME_OP;
00383     l4_utcb_mr_u(utcb)->mr[1] = id;
00384     t = l4_invoke_debugger(cap, l4_msgtag(0, 2, 0, 0), utcb);
00385     __strcpy_maxlen(name, (char const *)&l4_utcb_mr_u(utcb)->mr[0], size);
00386     return t;
00387 }
00388
00389 L4_INLINE l4_msgtag_t
00390 l4_debugger_get_object_name_u(l4_cap_idx_t cap,
00391                               char *name, unsigned size,
00392                               l4_utcb_t *utcb) L4_NOTHROW
00393 {
00394     l4_msgtag_t t;
00395     l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_KOBJ_GET_NAME_OP;
00396     t = l4_invoke_debugger(cap, l4_msgtag(0, 1, 0, 0), utcb);
00397     __strcpy_maxlen(name, (char const *)&l4_utcb_mr_u(utcb)->mr[0], size);
00398     return t;
00399 }
00400
00401 L4_INLINE l4_msgtag_t
00402 l4_debugger_add_image_info_u(l4_cap_idx_t cap, l4_addr_t base,
00403                             const char *name, l4_utcb_t *utcb) L4_NOTHROW
00404 {
00405     unsigned i;
00406     l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_TASK_ADD_IMAGE_INFO_OP;
00407     l4_utcb_mr_u(utcb)->mr[1] = base;
00408     i = __strcpy_maxlen((char *)&l4_utcb_mr_u(utcb)->mr[2], name,
00409                        (L4_UTCB_GENERIC_DATA_SIZE - 3) * sizeof(l4_umword_t));
00410     i = l4_bytes_to_mwords(i);
00411     return l4_invoke_debugger(cap, l4_msgtag(0, 2 + i, 0, 0), utcb);
00412 }
00413
00414
00415 L4_INLINE l4_msgtag_t
00416 l4_debugger_set_object_name(l4_cap_idx_t cap,
00417                             const char *name) L4_NOTHROW
00418 {
00419     return l4_debugger_set_object_name_u(cap, name, l4_utcb());
00420 }
00421
00422 L4_INLINE unsigned long
00423 l4_debugger_global_id(l4_cap_idx_t cap) L4_NOTHROW
00424 {
00425     return l4_debugger_global_id_u(cap, l4_utcb());
00426 }
00427
00428 L4_INLINE unsigned long
00429 l4_debugger_kobj_to_id(l4_cap_idx_t cap, l4_addr_t kobjp) L4_NOTHROW
00430 {
00431     return l4_debugger_kobj_to_id_u(cap, kobjp, l4_utcb());
00432 }
00433
00434 L4_INLINE l4_ret_t
00435 l4_debugger_query_log_typeid(l4_cap_idx_t cap, const char *name,
00436                              unsigned idx) L4_NOTHROW
00437 {
00438     return l4_debugger_query_log_typeid_u(cap, name, idx, l4_utcb());
00439 }
00440
00441 L4_INLINE l4_ret_t
00442 l4_debugger_query_log_name(l4_cap_idx_t cap, unsigned idx,
00443                            char *name, unsigned namelen,
00444                            char *shortname, unsigned shortnamelen) L4_NOTHROW
00445 {
00446     return l4_debugger_query_log_name_u(cap, idx, name, namelen,
00447                                         shortname, shortnamelen, l4_utcb());
00448 }
00449
00450 L4_INLINE l4_msgtag_t
00451 l4_debugger_switch_log(l4_cap_idx_t cap, const char *name,
00452                       int on_off) L4_NOTHROW
00453 {
00454     return l4_debugger_switch_log_u(cap, name, on_off, l4_utcb());
00455 }
00456
00457 L4_INLINE l4_msgtag_t
00458 l4_debugger_query_object_name(l4_cap_idx_t cap, unsigned id,

```



```

00459             char *name, unsigned size) L4_NOTHROW
00460 {
00461     return l4_debugger_query_object_name_u(cap, id, name, size, l4_utcb());
00462 }
00463
00464 L4_INLINE l4_msgtag_t
00465 l4_debugger_get_object_name(l4_cap_idx_t cap,
00466                             char *name, unsigned size) L4_NOTHROW
00467 {
00468     return l4_debugger_get_object_name_u(cap, name, size, l4_utcb());
00469 }
00470
00471 L4_INLINE l4_msgtag_t
00472 l4_debugger_add_image_info(l4_cap_idx_t cap, l4_addr_t base,
00473                           const char *name) L4_NOTHROW
00474 {
00475     return l4_debugger_add_image_info_u(cap, base, name, l4_utcb());
00476 }

```

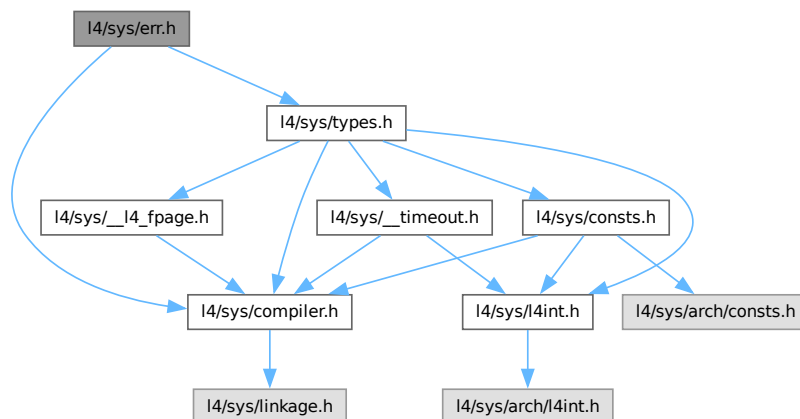
16.510 l4/sys/err.h File Reference

Error codes.

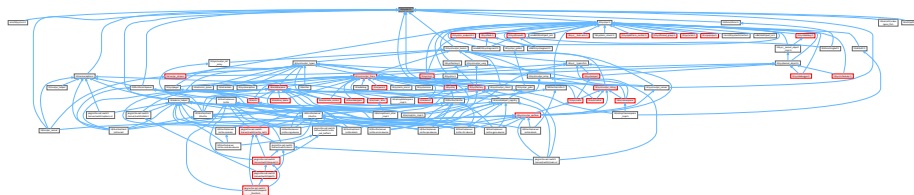
```
#include <l4/sys/compiler.h>
```

```
#include <l4/sys/types.h>
```

Include dependency graph for err.h:



This graph shows which files directly or indirectly include this file:



Enumerations

- enum `l4_error_code_t` {
`L4_EOK` = 0 , `L4_EPERM` = 1 , `L4_ENOENT` = 2 , `L4_EIO` = 5 ,
`L4_ENXIO` = 6 , `L4_E2BIG` = 7 , `L4_EAGAIN` = 11 , `L4_ENOMEM` = 12 ,
`L4_EACCESS` = 13 , `L4_EFAULT` = 14 , `L4_EBUSY` = 16 , `L4_EEXIST` = 17 ,
`L4_ENODEV` = 19 , `L4_ENOTDIR` = 20 , `L4_EINVAL` = 22 , `L4_ENOSPC` = 28 ,
`L4_ERANGE` = 34 , `L4_ENAMETOOLONG` = 36 , `L4_ENOSYS` = 38 , `L4_EBADPROTO` = 39 ,
`L4_EADDRNOTAVAIL` = 99 , `L4_ERRNOMAX` = 100 , `L4_ENOREPLY` = 1000 , `L4_MSGTOOSHORT` =
1001 ,
`L4_MSGTOOLONG` = 1002 , `L4_MSGMISSARG` = 1003 , `L4_MSGERRRANGE` = 1004 ,
`L4_EDROPREPLY` = 1005 ,
`L4_EIPC_LO` = 2000 , `L4_EIPC_HI` = 2000 + 0x1f }
L4 error codes.

16.510.1 Detailed Description

Error codes.

Definition in file [err.h](#).

16.511 err.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #pragma once
00013
00014 #include <l4/sys/compiler.h>
00015 #include <l4/sys/types.h>
00016
00024
00031 enum l4_error_code_t
00032 {
00033     L4_EOK                = 0,
00034     L4_EPERM              = 1,
00035     L4_ENOENT             = 2,
00036     L4_EIO                = 5,
00037     L4_ENXIO              = 6,
00038     L4_E2BIG              = 7,
00039     L4_EAGAIN             = 11,
00040     L4_ENOMEM             = 12,
00041     L4_EACCESS            = 13,
00042     L4_EFAULT             = 14,
00043     L4_EBUSY              = 16,
00044     L4_EEXIST             = 17,
00045     L4_ENODEV             = 19,
00046     L4_ENOTDIR            = 20,
00047     L4_EINVAL             = 22,
00048     L4_ENOSPC             = 28,
00049     L4_ERANGE             = 34,
00050     L4_ENAMETOOLONG       = 36,
00051     L4_ENOSYS             = 38,
00052     L4_EBADPROTO          = 39,
00053     L4_EADDRNOTAVAIL      = 99,
00054     L4_ERRNOMAX           = 100,
00055
00056     L4_ENOREPLY           = 1000,
00057     L4_MSGTOOSHORT        = 1001,
00058     L4_MSGTOOLONG         = 1002,
00059     L4_MSGMISSARG         = 1003,
00060     L4_MSGERRRANGE        = 1004,

```

16.512 I4/sys/exception File Reference

```
#include <l4/sys/capability>
#include <l4/sys/types.h>
#include <l4/sys/cxx/ipc_types>
#include <l4/sys/cxx/ipc_iface>
Include dependency graph for exception:
```



- Generated for L4Re by Doxygen

Namespaces

- namespace [L4](#)
[L4](#) low-level kernel interface.

16.512.1 Detailed Description

Exception C++ interface.

Definition in file [exception](#).

16.513 exception

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011
00012 #pragma once
00013
00014 #include <l4/sys/capability>
00015 #include <l4/sys/types.h>
00016 #include <l4/sys/cxx/ipc_types>
00017 #include <l4/sys/cxx/ipc_iface>
00018
00019 namespace L4 {
00020
00031 class L4_EXPORT Exception :
00032     public Kobject_0t<Exception, L4_PROTO_EXCEPTION>
00033 {
00034 public:
00035     // TODO: pass a reference/pointer to the UTCB not copy the regs
00046     L4_INLINE_RPC(
00047         l4_msgtag_t, exception, (L4::Ipc::In_out<l4_exc_regs_t *> regs,
00048                                 L4::Ipc::Rcv_fpage rwin,
00049                                 L4::Ipc::Opt<L4::Ipc::Snd_fpage &> fp));
00050
00051     typedef L4::Typeid::Rpc_nocode<exception_t> Rpcs;
00052 };
00053
00054 }
```

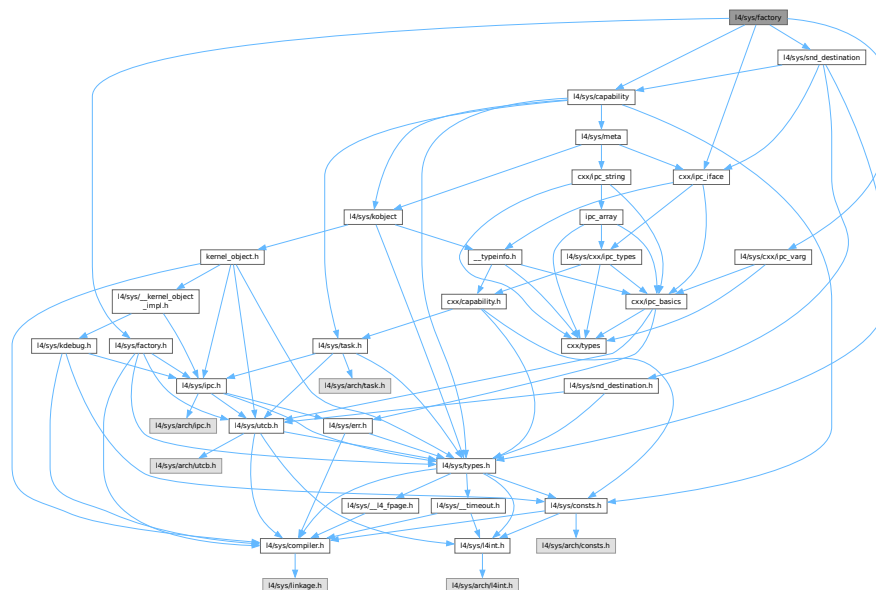
16.514 l4/sys/factory File Reference

Common factory related definitions.

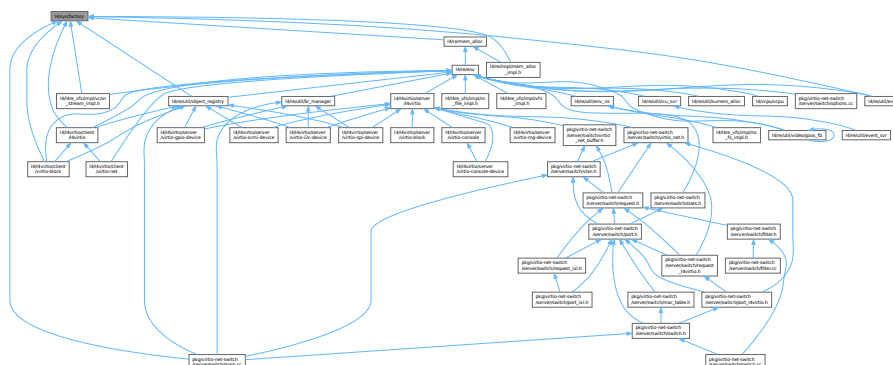
```

#include <l4/sys/factory.h>
#include <l4/sys/capability>
#include <l4/sys/snd_destination>
#include <l4/sys/cxx/ipc_iface>
```

```
#include <l4/sys/cxx/ipc_varg>
```



This graph shows which files directly or indirectly include this file:



Data Structures

- class `L4::Factory`
C++ Factory interface, see `Factory` for the C interface.
- struct `L4::Factory::Nil`
Special type to add a void argument into the factory create stream.
- struct `L4::Factory::Lstr`
Special type to add a pascal string into the factory create stream.
- class `L4::Factory::S`
Stream class for the `create()` argument stream.

Namespaces

- namespace L4
L4 low-level kernel interface.

16.514.1 Detailed Description

Common factory related definitions.

Definition in file [factory](#).

16.515 factory

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013
00014 #pragma once
00015
00016 #include <l4/sys/factory.h>
00017 #include <l4/sys/capability>
00018 #include <l4/sys/snd_destination>
00019 #include <l4/sys/cxx/ipc_iface>
00020 #include <l4/sys/cxx/ipc_varg>
00021
00022 namespace L4 {
00023
00038 class Factory : public Kobject_t<Factory, Kobject, L4_PROTO_FACTORY>
00039 {
00040 public:
00041
00042     typedef l4_mword_t Proto;
00043
00047     struct Nil {};
00048
00054     struct Lstr
00055     {
00059         char const *s;
00060
00064         unsigned len;
00065
00070         Lstr(char const *s, unsigned len) noexcept : s(s), len(len) {}
00071     };
00072
00079     class S
00080     {
00081     private:
00082         l4_utcb_t *u;
00083         l4_msgtag_t t;
00084         l4_cap_idx_t f;
00085
00086         template<typename T>
00087         static T &&_move(T &c) { return static_cast<T &&>(c); }
00088
00089     public:
00090         S(S const &) = delete;
00091         S &operator = (S const &) &= delete;
00092
00098         S(S &&o) noexcept
00099         : u(o.u), t(o.t), f(o.f)
00100         { o.t.raw = 0; }
00101
00102         S &operator = (S &&o) & noexcept
00103         {
00104             u = o.u;
00105             t = o.t;
00106             f = o.f;
00107             o.t.raw = 0;
00108             return *this;
00109         }
00110
00125         S(l4_cap_idx_t f, long obj, L4::Cap<void> target,
00126           l4_utcb_t *utcb) noexcept
00127         : u(utcb), t(l4_factory_create_start_u(obj, target.cap(), u)), f(f)
00128         {}
00129
00139         ~S() noexcept

```

```

00140     {
00141         if (t.raw)
00142     l4_factory_create_commit_u(f, t, u);
00143     }
00144
00157     operator l4_msgtag_t () noexcept
00158     {
00159         l4_msgtag_t r = l4_factory_create_commit_u(f, t, u);
00160         t.raw = 0;
00161         return r;
00162     }
00163
00169     void put(l4_mword_t i) noexcept
00170     {
00171         l4_factory_create_add_int_u(i, &t, u);
00172     }
00173
00179     void put(l4_umword_t i) noexcept
00180     {
00181         l4_factory_create_add_uint_u(i, &t, u);
00182     }
00183
00191     void put(char const *s) & noexcept
00192     {
00193         l4_factory_create_add_str_u(s, &t, u);
00194     }
00195
00205     void put(Lstr const &s) & noexcept
00206     {
00207         l4_factory_create_add_lstr_u(s.s, s.len, &t, u);
00208     }
00209
00213     void put(Nil) & noexcept
00214     {
00215         l4_factory_create_add_nil_u(&t, u);
00216     }
00217
00223     void put(l4_fpage_t d) & noexcept
00224     {
00225         l4_factory_create_add_fpage_u(d, &t, u);
00226     }
00227
00236     template<typename T>
00237     S &operator « (T const &d) & noexcept
00238     {
00239         put(d);
00240         return *this;
00241     }
00242
00251     template<typename T>
00252     S &&operator « (T const &d) && noexcept
00253     {
00254         put(d);
00255         return _move(*this);
00256     }
00257 };
00258
00259 public:
00260
00261
00292     S create(Cap<void> target, long obj, l4_utcb_t *utcb = l4_utcb()) noexcept
00293     {
00294         return S(cap(), obj, target, utcb);
00295     }
00296
00328     template<typename OBJ>
00329     S create(Cap<OBJ> target, l4_utcb_t *utcb = l4_utcb()) noexcept
00330     {
00331         return S(cap(), OBJ::Protocol, target, utcb);
00332     }
00333
00334     L4_INLINE_RPC_NF(
00335         l4_msgtag_t, create, (L4::Ipc::Out<L4::Cap<void> > target, l4_mword_t obj,
00336                             L4::Ipc::Varg const *args),
00337         L4::Ipc::Call_t<L4_CAP_FPAGE_S>);
00338
00370     l4_msgtag_t create_task(Cap<Task> const & target_cap,
00371                             l4_fpage_t *utcb_area,
00372                             l4_utcb_t *utcb = l4_utcb()) noexcept
00373     { return l4_factory_create_task_u(cap(), target_cap.cap(), utcb_area, utcb); }
00374
00404     l4_msgtag_t create_factory(Cap<Factory> const &target_cap,
00405                                unsigned long limit,
00406                                l4_utcb_t *utcb = l4_utcb()) noexcept
00407     { return l4_factory_create_factory_u(cap(), target_cap.cap(), limit, utcb); }
00408
00440     l4_msgtag_t create_gate(Cap<void> const &target_cap,

```

```

00441         Cap<Snd_destination> const &snd_dst_cap,
00442         l4_umword_t label,
00443         l4_utcb_t *utcb = l4_utcb()) noexcept
00444     {
00445         return l4_factory_create_gate_u(cap(), target_cap.cap(), snd_dst_cap.cap(),
00446                                         label, utcb);
00447     }
00448
00475 l4_msgtag_t create_thread_group(Cap<Thread_group> const &target_cap,
00476                                unsigned policy,
00477                                l4_utcb_t *utcb = l4_utcb()) noexcept
00478     {
00479         return l4_factory_create_thread_group_u(cap(), target_cap.cap(),
00480                                                 policy, utcb);
00481     }
00482
00483 typedef L4::Typeid::Rpc_nocode<create_t> Rpccs;
00484 };
00485
00486 }

```

16.516 l4/sys/factory.h File Reference

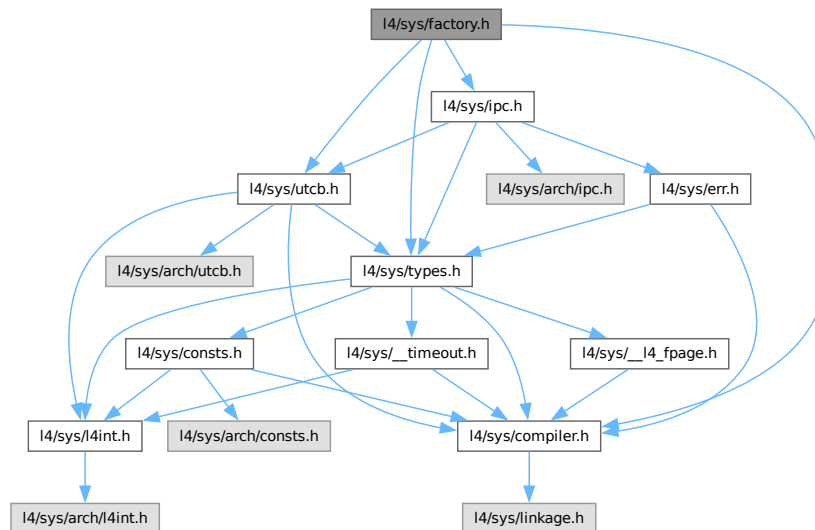
Common factory related definitions.

```

#include <l4/sys/compiler.h>
#include <l4/sys/types.h>
#include <l4/sys/utcb.h>
#include <l4/sys/ipc.h>

```

Include dependency graph for factory.h:



- `l4_msgtag_t l4_factory_create_task (l4_cap_idx_t factory, l4_cap_idx_t target_cap, l4_fpage_t *utcb_area) L4_NOTHROW`
Create a new task.
- `l4_msgtag_t l4_factory_create_thread (l4_cap_idx_t factory, l4_cap_idx_t target_cap) L4_NOTHROW`
Create a new thread.
- `l4_msgtag_t l4_factory_create_factory (l4_cap_idx_t factory, l4_cap_idx_t target_cap, unsigned long limit) L4_NOTHROW`
Create a new factory.
- `l4_msgtag_t l4_factory_create_gate (l4_cap_idx_t factory, l4_cap_idx_t target_cap, l4_cap_idx_t snd_dst_ ← cap, l4_umword_t label) L4_NOTHROW`
Create a new IPC gate, optionally bound to a send destination (a thread or thread group).
- `l4_msgtag_t l4_factory_create_irq (l4_cap_idx_t factory, l4_cap_idx_t target_cap) L4_NOTHROW`
Create a new IRQ sender.
- `l4_msgtag_t l4_factory_create_vm (l4_cap_idx_t factory, l4_cap_idx_t target_cap) L4_NOTHROW`
Create a new virtual machine.
- `l4_msgtag_t l4_factory_create_vcpu_context (l4_cap_idx_t factory, l4_cap_idx_t target_cap) L4_NOTHROW`
Create a new vCPU context.
- `l4_msgtag_t l4_factory_create_thread_group (l4_cap_idx_t factory, l4_cap_idx_t target_cap, unsigned policy) L4_NOTHROW`
Create a new thread group.
- `l4_msgtag_t l4_factory_create (l4_cap_idx_t factory, long obj, l4_cap_idx_t target) L4_NOTHROW`
Create a new object.

Definition in file [factory.h](#).

16.517 factory.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *      Björn Döbel <doebel@os.inf.tu-dresden.de>,
00010  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>,
00011  *      Henning Schild <hschild@os.inf.tu-dresden.de>
00012  *      economic rights: Technische Universität Dresden (Germany)
00013  *
00014  * License: see LICENSE.spdx (in this directory or the directories above)
00015  */
00016 #pragma once
00017
00018 #include <l4/sys/compiler.h>
00019 #include <l4/sys/types.h>
00020 #include <l4/sys/utcb.h>
00021
00050
00056
00086 L4_INLINE l4_msgtag_t
00087 l4_factory_create_task(l4_cap_idx_t factory, l4_cap_idx_t target_cap,
00088                      l4_fpage_t *utcb_area) L4_NOTHROW;
00089
00094 L4_INLINE l4_msgtag_t
00095 l4_factory_create_task_u(l4_cap_idx_t factory, l4_cap_idx_t target_cap,
00096                        l4_fpage_t *utcb_area, l4_utcb_t *utcb) L4_NOTHROW;
00097
00116 L4_INLINE l4_msgtag_t
00117 l4_factory_create_thread(l4_cap_idx_t factory,
00118                         l4_cap_idx_t target_cap) L4_NOTHROW;
00119
00124 L4_INLINE l4_msgtag_t
00125 l4_factory_create_thread_u(l4_cap_idx_t factory,
00126                           l4_cap_idx_t target_cap, l4_utcb_t *utcb) L4_NOTHROW;
00127
00153 L4_INLINE l4_msgtag_t
00154 l4_factory_create_factory(l4_cap_idx_t factory, l4_cap_idx_t target_cap,
00155                          unsigned long limit) L4_NOTHROW;
00156
00161 L4_INLINE l4_msgtag_t
00162 l4_factory_create_factory_u(l4_cap_idx_t factory, l4_cap_idx_t target_cap,
00163                           unsigned long limit, l4_utcb_t *utcb) L4_NOTHROW;
00164
00194 L4_INLINE l4_msgtag_t
00195 l4_factory_create_gate(l4_cap_idx_t factory,
00196                       l4_cap_idx_t target_cap,
00197                       l4_cap_idx_t snd_dst_cap, l4_umword_t label) L4_NOTHROW;
00198
00203 L4_INLINE l4_msgtag_t
00204 l4_factory_create_gate_u(l4_cap_idx_t factory,
00205                          l4_cap_idx_t target_cap,
00206                          l4_cap_idx_t snd_dst_cap, l4_umword_t label,
00207                          l4_utcb_t *utcb) L4_NOTHROW;
00208
00225 L4_INLINE l4_msgtag_t
00226 l4_factory_create_irq(l4_cap_idx_t factory,
00227                      l4_cap_idx_t target_cap) L4_NOTHROW;
00228
00233 L4_INLINE l4_msgtag_t
00234 l4_factory_create_irq_u(l4_cap_idx_t factory,
00235                        l4_cap_idx_t target_cap, l4_utcb_t *utcb) L4_NOTHROW;
00236
00255 L4_INLINE l4_msgtag_t
00256 l4_factory_create_vm(l4_cap_idx_t factory,
00257                     l4_cap_idx_t target_cap) L4_NOTHROW;
00258
00278 L4_INLINE l4_msgtag_t
00279 l4_factory_create_vcpu_context(l4_cap_idx_t factory,
00280                               l4_cap_idx_t target_cap) L4_NOTHROW;
00281
00309 L4_INLINE l4_msgtag_t
00310 l4_factory_create_thread_group(l4_cap_idx_t factory,
00311                               l4_cap_idx_t target_cap,
00312                               unsigned policy) L4_NOTHROW;
00313
00318 L4_INLINE l4_msgtag_t
00319 l4_factory_create_vm_u(l4_cap_idx_t factory,
00320                       l4_cap_idx_t target_cap, l4_utcb_t *utcb) L4_NOTHROW;
00321
00326 L4_INLINE l4_msgtag_t
00327 l4_factory_create_vcpu_context_u(l4_cap_idx_t factory,
00328                                  l4_cap_idx_t target_cap,

```

```

00329                                     l4_utcb_t *utcb) L4_NOTHROW;
00330
00335 L4_INLINE l4_msgtag_t
00336 l4_factory_create_thread_group_u(l4_cap_idx_t factory,
00337                                   l4_cap_idx_t target_cap,
00338                                   unsigned policy,
00339                                   l4_utcb_t *u) L4_NOTHROW;
00340
00345 L4_INLINE l4_msgtag_t
00346 l4_factory_create_start_u(long obj, l4_cap_idx_t target,
00347                           l4_utcb_t *utcb) L4_NOTHROW;
00348
00353 L4_INLINE int
00354 l4_factory_create_add_fpage_u(l4_fpage_t d, l4_msgtag_t *tag,
00355                               l4_utcb_t *utcb) L4_NOTHROW;
00356
00361 L4_INLINE int
00362 l4_factory_create_add_int_u(l4_mword_t d, l4_msgtag_t *tag,
00363                             l4_utcb_t *utcb) L4_NOTHROW;
00364
00369 L4_INLINE int
00370 l4_factory_create_add_uint_u(l4_umword_t d, l4_msgtag_t *tag,
00371                              l4_utcb_t *utcb) L4_NOTHROW;
00372
00377 L4_INLINE int
00378 l4_factory_create_add_str_u(char const *s, l4_msgtag_t *tag,
00379                             l4_utcb_t *utcb) L4_NOTHROW;
00380
00385 L4_INLINE int
00386 l4_factory_create_add_lstr_u(char const *s, unsigned len, l4_msgtag_t *tag,
00387                              l4_utcb_t *utcb) L4_NOTHROW;
00388
00393 L4_INLINE int
00394 l4_factory_create_add_nil_u(l4_msgtag_t *tag, l4_utcb_t *utcb) L4_NOTHROW;
00395
00400 L4_INLINE l4_msgtag_t
00401 l4_factory_create_commit_u(l4_cap_idx_t factory, l4_msgtag_t tag,
00402                            l4_utcb_t *utcb) L4_NOTHROW;
00403
00408 L4_INLINE l4_msgtag_t
00409 l4_factory_create_u(l4_cap_idx_t factory, long obj, l4_cap_idx_t target,
00410                    l4_utcb_t *utcb) L4_NOTHROW;
00411
00412
00430 L4_INLINE l4_msgtag_t
00431 l4_factory_create(l4_cap_idx_t factory, long obj,
00432                  l4_cap_idx_t target) L4_NOTHROW;
00433
00434 /* IMPLEMENTATION -----*/
00435
00436 #include <l4/sys/ipc.h>
00437
00438 L4_INLINE l4_msgtag_t
00439 l4_factory_create_task_u(l4_cap_idx_t factory,
00440                         l4_cap_idx_t target_cap, l4_fpage_t *utcb_area,
00441                         l4_utcb_t *u) L4_NOTHROW
00442 {
00443     l4_msgtag_t t;
00444     t = l4_factory_create_start_u(L4_PROTO_TASK, target_cap, u);
00445     l4_factory_create_add_fpage_u(*utcb_area, &t, u);
00446     t = l4_factory_create_commit_u(factory, t, u);
00447     if (!l4_msgtag_has_error(t))
00448     {
00449         l4_msg_regs_t *v = l4_utcb_mr_u(u);
00450         utcb_area->raw = v->mr[0];
00451     }
00452     return t;
00453 }
00454
00455 L4_INLINE l4_msgtag_t
00456 l4_factory_create_thread_u(l4_cap_idx_t factory,
00457                            l4_cap_idx_t target_cap, l4_utcb_t *u) L4_NOTHROW
00458 {
00459     return l4_factory_create_u(factory, L4_PROTO_THREAD, target_cap, u);
00460 }
00461
00462 L4_INLINE l4_msgtag_t
00463 l4_factory_create_factory_u(l4_cap_idx_t factory,
00464                             l4_cap_idx_t target_cap, unsigned long limit,
00465                             l4_utcb_t *u) L4_NOTHROW
00466 {
00467     l4_msgtag_t t;
00468     t = l4_factory_create_start_u(L4_PROTO_FACTORY, target_cap, u);
00469     l4_factory_create_add_uint_u(limit, &t, u);
00470     return l4_factory_create_commit_u(factory, t, u);
00471 }
00472

```

```

00473 L4_INLINE l4_msgtag_t
00474 l4_factory_create_gate_u(l4_cap_idx_t factory,
00475                          l4_cap_idx_t target_cap,
00476                          l4_cap_idx_t snd_dst_cap, l4_umword_t label,
00477                          l4_utcb_t *u) L4_NOTHROW
00478 {
00479     l4_msgtag_t t;
00480     l4_msg_regs_t *v;
00481     int items = 0;
00482     t = l4_factory_create_start_u(0, target_cap, u);
00483     l4_factory_create_add_uint_u(label, &t, u);
00484     v = l4_utcb_mr_u(u);
00485     if (!(snd_dst_cap & L4_INVALID_CAP_BIT))
00486     {
00487         items = 1;
00488         v->mr[3] = l4_map_obj_control(0,0);
00489         v->mr[4] = l4_obj_fpage(snd_dst_cap, 0, L4_CAP_FPAGE_RWS).raw;
00490     }
00491     t = l4_msgtag(l4_msgtag_label(t), l4_msgtag_words(t), items, l4_msgtag_flags(t));
00492     return l4_factory_create_commit_u(factory, t, u);
00493 }
00494
00495 L4_INLINE l4_msgtag_t
00496 l4_factory_create_irq_u(l4_cap_idx_t factory,
00497                        l4_cap_idx_t target_cap, l4_utcb_t *u) L4_NOTHROW
00498 {
00499     return l4_factory_create_u(factory, L4_PROTO_IRQ_SENDER, target_cap, u);
00500 }
00501
00502 L4_INLINE l4_msgtag_t
00503 l4_factory_create_vm_u(l4_cap_idx_t factory,
00504                      l4_cap_idx_t target_cap,
00505                      l4_utcb_t *u) L4_NOTHROW
00506 {
00507     return l4_factory_create_u(factory, L4_PROTO_VM, target_cap, u);
00508 }
00509
00510 L4_INLINE l4_msgtag_t
00511 l4_factory_create_vcpu_context_u(l4_cap_idx_t factory,
00512                                l4_cap_idx_t target_cap,
00513                                l4_utcb_t *u) L4_NOTHROW
00514 {
00515     return l4_factory_create_u(factory, L4_PROTO_VCPU_CONTEXT, target_cap, u);
00516 }
00517
00518 L4_INLINE l4_msgtag_t
00519 l4_factory_create_thread_group_u(l4_cap_idx_t factory,
00520                                l4_cap_idx_t target_cap,
00521                                unsigned policy,
00522                                l4_utcb_t *u) L4_NOTHROW
00523 {
00524     l4_msgtag_t t = l4_factory_create_start_u(L4_PROTO_THREAD_GROUP, target_cap, u);
00525     l4_factory_create_add_uint_u(policy, &t, u);
00526     return l4_factory_create_commit_u(factory, t, u);
00527 }
00528
00529
00530 L4_INLINE l4_msgtag_t
00531 l4_factory_create_task(l4_cap_idx_t factory,
00532                      l4_cap_idx_t target_cap, l4_fpage_t *utcb_area) L4_NOTHROW
00533 {
00534     return l4_factory_create_task_u(factory, target_cap, utcb_area, l4_utcb());
00535 }
00536
00537 L4_INLINE l4_msgtag_t
00538 l4_factory_create_thread(l4_cap_idx_t factory,
00539                        l4_cap_idx_t target_cap) L4_NOTHROW
00540 {
00541     return l4_factory_create_thread_u(factory, target_cap, l4_utcb());
00542 }
00543
00544 L4_INLINE l4_msgtag_t
00545 l4_factory_create_factory(l4_cap_idx_t factory,
00546                        l4_cap_idx_t target_cap, unsigned long limit) L4_NOTHROW
00547 {
00548     return l4_factory_create_factory_u(factory, target_cap, limit, l4_utcb());
00549 }
00550
00551
00552 L4_INLINE l4_msgtag_t
00553 l4_factory_create_gate(l4_cap_idx_t factory,
00554                      l4_cap_idx_t target_cap,
00555                      l4_cap_idx_t snd_dst_cap, l4_umword_t label) L4_NOTHROW
00556 {
00557     return l4_factory_create_gate_u(factory, target_cap, snd_dst_cap, label, l4_utcb());
00558 }
00559

```

```

00560 L4_INLINE l4_msgtag_t
00561 l4_factory_create_irq(l4_cap_idx_t factory,
00562                      l4_cap_idx_t target_cap) L4_NOTHROW
00563 {
00564     return l4_factory_create_irq_u(factory, target_cap, l4_utcb());
00565 }
00566
00567 L4_INLINE l4_msgtag_t
00568 l4_factory_create_vm(l4_cap_idx_t factory,
00569                    l4_cap_idx_t target_cap) L4_NOTHROW
00570 {
00571     return l4_factory_create_vm_u(factory, target_cap, l4_utcb());
00572 }
00573
00574 L4_INLINE l4_msgtag_t
00575 l4_factory_create_vcpu_context(l4_cap_idx_t factory,
00576                              l4_cap_idx_t target_cap) L4_NOTHROW
00577 {
00578     return l4_factory_create_vcpu_context_u(factory, target_cap, l4_utcb());
00579 }
00580
00581 L4_INLINE l4_msgtag_t
00582 l4_factory_create_thread_group(l4_cap_idx_t factory,
00583                               l4_cap_idx_t target_cap,
00584                               unsigned policy) L4_NOTHROW
00585 {
00586     return l4_factory_create_thread_group_u(factory, target_cap, policy, l4_utcb());
00587 }
00588
00589
00590 L4_INLINE l4_msgtag_t
00591 l4_factory_create_start_u(long obj, l4_cap_idx_t target_cap,
00592                          l4_utcb_t *u) L4_NOTHROW
00593 {
00594     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00595     l4_buf_regs_t *b = l4_utcb_br_u(u);
00596     v->mr[0] = obj;
00597     b->bdr = 0;
00598     b->br[0] = target_cap | L4_RCV_ITEM_SINGLE_CAP;
00599     return l4_msgtag(L4_PROTO_FACTORY, 1, 0, 0);
00600 }
00601
00602 L4_INLINE int
00603 l4_factory_create_add_fpage_u(l4_fpage_t d, l4_msgtag_t *tag,
00604                             l4_utcb_t *u) L4_NOTHROW
00605 {
00606     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00607     int w = l4_msgtag_words(*tag);
00608     if (w + 2 > L4_UTCB_GENERIC_DATA_SIZE)
00609         return 0;
00610     v->mr[w] = L4_VARG_TYPE_FPAGE | (sizeof(l4_fpage_t) << 16);
00611     v->mr[w + 1] = d.raw;
00612     w += 2;
00613     tag->raw = (tag->raw & ~0x3fUL) | (w & 0x3f);
00614     return 1;
00615 }
00616
00617 L4_INLINE int
00618 l4_factory_create_add_int_u(l4_mword_t d, l4_msgtag_t *tag,
00619                          l4_utcb_t *u) L4_NOTHROW
00620 {
00621     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00622     int w = l4_msgtag_words(*tag);
00623     if (w + 2 > L4_UTCB_GENERIC_DATA_SIZE)
00624         return 0;
00625     v->mr[w] = L4_VARG_TYPE_MWORD | (sizeof(l4_mword_t) << 16);
00626     v->mr[w + 1] = d;
00627     w += 2;
00628     tag->raw = (tag->raw & ~0x3fUL) | (w & 0x3f);
00629     return 1;
00630 }
00631
00632 L4_INLINE int
00633 l4_factory_create_add_uint_u(l4_umword_t d, l4_msgtag_t *tag,
00634                             l4_utcb_t *u) L4_NOTHROW
00635 {
00636     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00637     int w = l4_msgtag_words(*tag);
00638     if (w + 2 > L4_UTCB_GENERIC_DATA_SIZE)
00639         return 0;
00640     v->mr[w] = L4_VARG_TYPE_UMWORD | (sizeof(l4_umword_t) << 16);
00641     v->mr[w + 1] = d;
00642     w += 2;
00643     tag->raw = (tag->raw & ~0x3fUL) | (w & 0x3f);
00644     return 1;
00645 }
00646

```

```

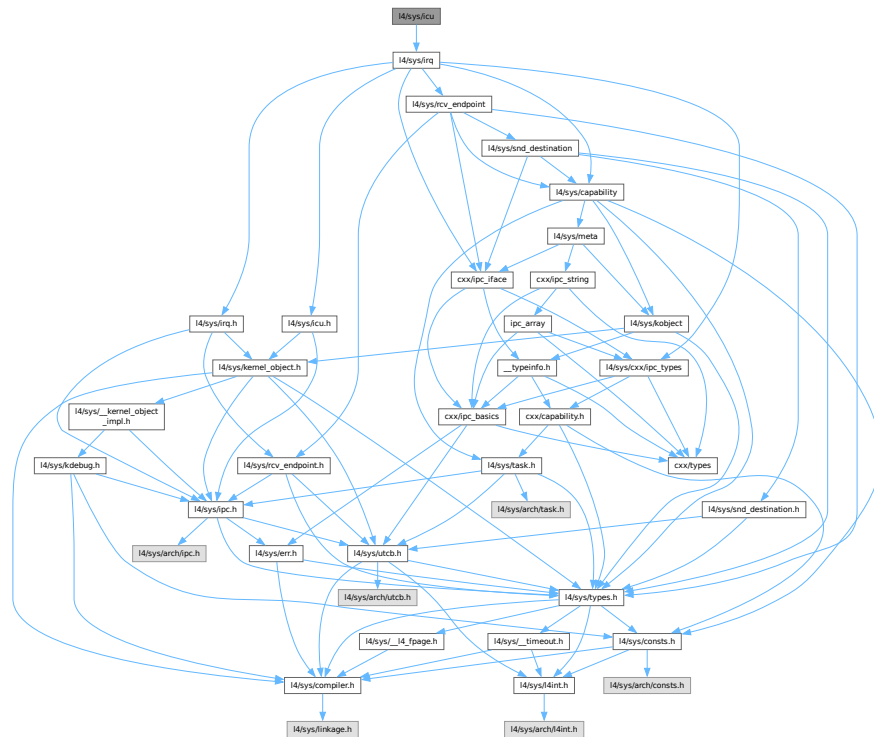
00647 L4_INLINE int
00648 l4_factory_create_add_str_u(char const *s, l4_msgtag_t *tag,
00649                             l4_utcb_t *u) L4_NOTHROW
00650 {
00651     return l4_factory_create_add_lstr_u(s, __builtin_strlen(s) + 1, tag, u);
00652 }
00653
00654 L4_INLINE int
00655 l4_factory_create_add_lstr_u(char const *s, unsigned len, l4_msgtag_t *tag,
00656                             l4_utcb_t *u) L4_NOTHROW
00657 {
00658
00659     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00660     unsigned w = l4_msgtag_words(*tag);
00661     char *c;
00662     unsigned i;
00663
00664     if (w + 1 + l4_bytes_to_mwords(len) > L4_UTCB_GENERIC_DATA_SIZE)
00665         return 0;
00666
00667     v->mr[w] = L4_VARG_TYPE_STRING | (len << 16);
00668     c = (char*)&v->mr[w + 1];
00669     for (i = 0; i < len; ++i)
00670         *c++ = *s++;
00671
00672     w = w + 1 + l4_bytes_to_mwords(len);
00673
00674     tag->raw = (tag->raw & ~0x3fUL) | (w & 0x3f);
00675     return 1;
00676 }
00677
00678 L4_INLINE int
00679 l4_factory_create_add_nil_u(l4_msgtag_t *tag, l4_utcb_t *utcb) L4_NOTHROW
00680 {
00681     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00682     int w = l4_msgtag_words(*tag);
00683     v->mr[w] = L4_VARG_TYPE_NIL;
00684     ++w;
00685     tag->raw = (tag->raw & ~0x3fUL) | (w & 0x3f);
00686     return 1;
00687 }
00688
00689
00690 L4_INLINE l4_msgtag_t
00691 l4_factory_create_commit_u(l4_cap_idx_t factory, l4_msgtag_t tag,
00692                             l4_utcb_t *u) L4_NOTHROW
00693 {
00694     return l4_ipc_call(factory, u, tag, L4_IPC_NEVER);
00695 }
00696
00697 L4_INLINE l4_msgtag_t
00698 l4_factory_create_u(l4_cap_idx_t factory, long obj, l4_cap_idx_t target,
00699                     l4_utcb_t *utcb) L4_NOTHROW
00700 {
00701     l4_msgtag_t t = l4_factory_create_start_u(obj, target, utcb);
00702     return l4_factory_create_commit_u(factory, t, utcb);
00703 }
00704
00705
00706 L4_INLINE l4_msgtag_t
00707 l4_factory_create(l4_cap_idx_t factory, long obj,
00708                   l4_cap_idx_t target) L4_NOTHROW
00709 {
00710     return l4_factory_create_u(factory, obj, target, l4_utcb());
00711 }

```

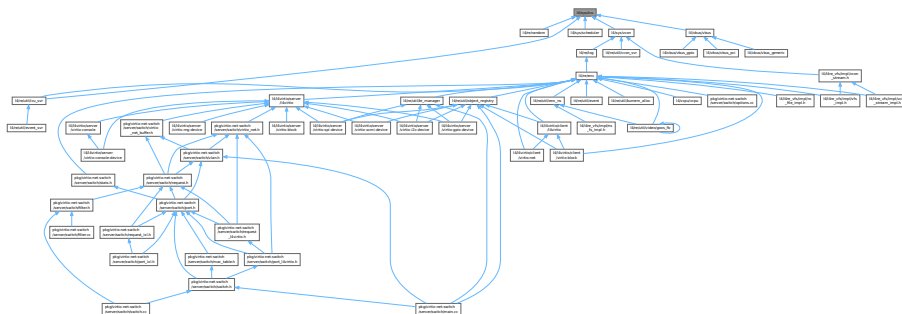
16.518 I4/sys/icu File Reference

Interrupt controller.

```
#include <linux/sys/irq>
```



This graph shows which files directly or indirectly include this file:



16.518.1 Detailed Description

Interrupt controller.

Definition in file [icu](#).

16.519 icu

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00007 /*
00008  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013 #pragma once
00014
00015 #include <l4/sys/irq>

```

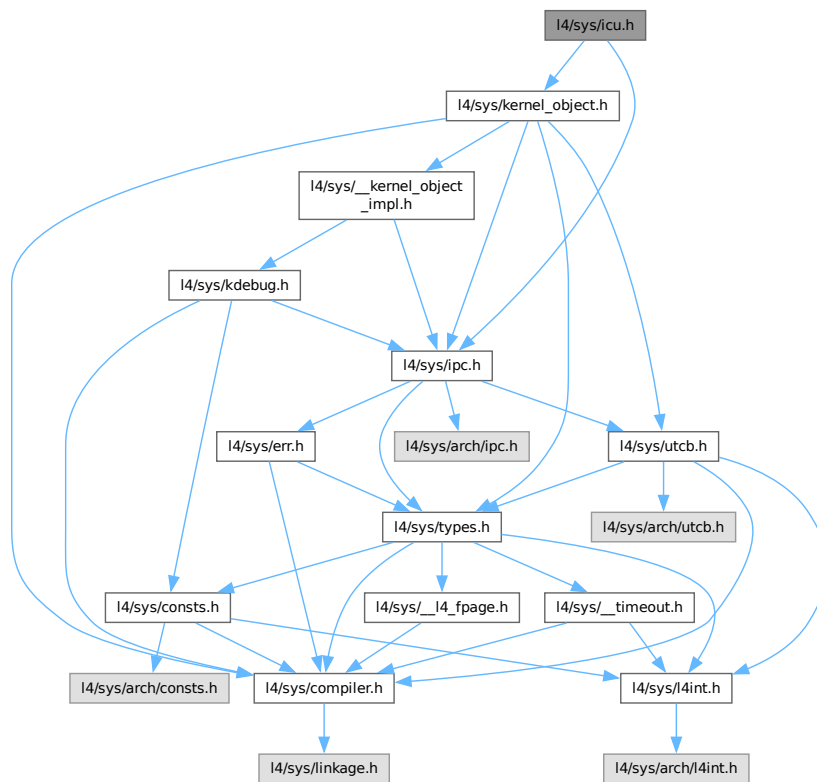
16.520 I4/sys/icu.h File Reference

Interrupt controller.

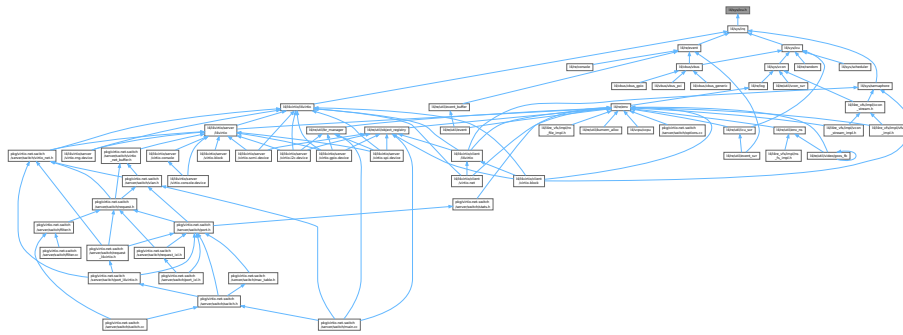
```
#include <l4/sys/kernel_object.h>
```

```
#include <l4/sys/ipc.h>
```

Include dependency graph for icu.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [l4_icu_info_t](#)
Info structure for an ICU.
- struct [l4_icu_msi_info_t](#)
Info to use for a specific MSI.

Typedefs

- typedef struct [l4_icu_info_t](#) [l4_icu_info_t](#)
Info structure for an ICU.
- typedef struct [l4_icu_msi_info_t](#) [l4_icu_msi_info_t](#)
Info to use for a specific MSI.

Enumerations

- enum [L4_icu_flags](#) { [L4_ICU_FLAG_MSI](#) }
Flags for IRQ numbers used for the ICU.
- enum [L4_irq_mode](#) {
[L4_IRQ_F_NONE](#) = 0 , [L4_IRQ_F_SET_MODE](#) = 0x1 , [L4_IRQ_F_LEVEL](#) = 0x2 , [L4_IRQ_F_EDGE](#) = 0x0 ,
[L4_IRQ_F_POS](#) = 0x0 , [L4_IRQ_F_NEG](#) = 0x4 , [L4_IRQ_F_BOTH](#) = 0x8 , [L4_IRQ_F_LEVEL_HIGH](#) =
[L4_IRQ_F_SET_MODE](#) | [L4_IRQ_F_LEVEL](#) | [L4_IRQ_F_POS](#) ,
[L4_IRQ_F_LEVEL_LOW](#) = [L4_IRQ_F_SET_MODE](#) | [L4_IRQ_F_LEVEL](#) | [L4_IRQ_F_NEG](#) , [L4_IRQ_F_POS_EDGE](#)
= [L4_IRQ_F_SET_MODE](#) | [L4_IRQ_F_EDGE](#) | [L4_IRQ_F_POS](#) , [L4_IRQ_F_NEG_EDGE](#) = [L4_IRQ_F_](#)
[SET_MODE](#) | [L4_IRQ_F_EDGE](#) | [L4_IRQ_F_NEG](#) , [L4_IRQ_F_BOTH_EDGE](#) = [L4_IRQ_F_SET_MODE](#) |
[L4_IRQ_F_EDGE](#) | [L4_IRQ_F_BOTH](#) ,
[L4_IRQ_F_MASK](#) = 0xf , [L4_IRQ_F_SET_WAKEUP](#) = 0x10 , [L4_IRQ_F_CLEAR_WAKEUP](#) = 0x20 }
Interrupt attributes.
- enum [L4_icu_opcode](#) {
[L4_ICU_OP_BIND](#) , [L4_ICU_OP_UNBIND](#) , [L4_ICU_OP_INFO](#) , [L4_ICU_OP_MSI_INFO](#) ,
[L4_ICU_OP_UNMASK](#) , [L4_ICU_OP_MASK](#) , [L4_ICU_OP_SET_MODE](#) }
Opcodes to the ICU interface.

Functions

- [l4_msgtag_t l4_icu_bind](#) ([l4_cap_idx_t](#) icu, unsigned irqnum, [l4_cap_idx_t](#) irq) [L4_NOTHROW](#)
Bind an interrupt line of an interrupt controller to an interrupt object.
- [l4_msgtag_t l4_icu_bind_u](#) ([l4_cap_idx_t](#) icu, unsigned irqnum, [l4_cap_idx_t](#) irq, [l4_utcb_t](#) *utcb) [L4_NOTHROW](#)
Bind an interrupt line of an interrupt controller to an interrupt object.
- [l4_msgtag_t l4_icu_unbind](#) ([l4_cap_idx_t](#) icu, unsigned irqnum, [l4_cap_idx_t](#) irq) [L4_NOTHROW](#)
Remove binding of an interrupt line from the interrupt controller object.
- [l4_msgtag_t l4_icu_unbind_u](#) ([l4_cap_idx_t](#) icu, unsigned irqnum, [l4_cap_idx_t](#) irq, [l4_utcb_t](#) *utcb) [L4_NOTHROW](#)
Remove binding of an interrupt line from the interrupt controller object.
- [l4_msgtag_t l4_icu_set_mode](#) ([l4_cap_idx_t](#) icu, unsigned irqnum, [l4_umword_t](#) mode) [L4_NOTHROW](#)
Set interrupt mode.
- [l4_msgtag_t l4_icu_set_mode_u](#) ([l4_cap_idx_t](#) icu, unsigned irqnum, [l4_umword_t](#) mode, [l4_utcb_t](#) *utcb) [L4_NOTHROW](#)
Set interrupt mode.
- [l4_msgtag_t l4_icu_info](#) ([l4_cap_idx_t](#) icu, [l4_icu_info_t](#) *info) [L4_NOTHROW](#)
Get information about the ICU features.
- [l4_msgtag_t l4_icu_info_u](#) ([l4_cap_idx_t](#) icu, [l4_icu_info_t](#) *info, [l4_utcb_t](#) *utcb) [L4_NOTHROW](#)
Get information about the ICU features.
- [l4_msgtag_t l4_icu_msi_info](#) ([l4_cap_idx_t](#) icu, unsigned irqnum, [l4_uint64_t](#) source, [l4_icu_msi_info_t](#) *msi_info) [L4_NOTHROW](#)
Get MSI info about IRQ.
- [l4_msgtag_t l4_icu_msi_info_u](#) ([l4_cap_idx_t](#) icu, unsigned irqnum, [l4_uint64_t](#) source, [l4_icu_msi_info_t](#) *msi_info, [l4_utcb_t](#) *utcb) [L4_NOTHROW](#)
Get MSI info about IRQ.
- [l4_msgtag_t l4_icu_unmask](#) ([l4_cap_idx_t](#) icu, unsigned irqnum, [l4_umword_t](#) *label, [l4_timeout_t](#) to) [L4_NOTHROW](#)
Unmask an IRQ line.
- [l4_msgtag_t l4_icu_unmask_u](#) ([l4_cap_idx_t](#) icu, unsigned irqnum, [l4_umword_t](#) *label, [l4_timeout_t](#) to, [l4_utcb_t](#) *utcb) [L4_NOTHROW](#)
Unmask the given interrupt line.
- [l4_msgtag_t l4_icu_mask](#) ([l4_cap_idx_t](#) icu, unsigned irqnum, [l4_umword_t](#) *label, [l4_timeout_t](#) to) [L4_NOTHROW](#)
Mask an IRQ line.
- [l4_msgtag_t l4_icu_mask_u](#) ([l4_cap_idx_t](#) icu, unsigned irqnum, [l4_umword_t](#) *label, [l4_timeout_t](#) to, [l4_utcb_t](#) *utcb) [L4_NOTHROW](#)
Mask an IRQ line.

16.520.1 Detailed Description

Interrupt controller.

Definition in file [icu.h](#).

16.521 icu.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014 #pragma once
00015
00016 #include <l4/sys/kernel_object.h>
00017 #include <l4/sys/ipc.h>
00018
00047
00048
00053 enum L4_icu_flags
00054 {
00062     L4_ICU_FLAG_MSI = 0x80000000,
00063 };
00064
00065
00070 enum L4_irq_mode
00071 {
00073     L4_IRQ_F_NONE           = 0,
00074     L4_IRQ_F_SET_MODE      = 0x1,
00075     L4_IRQ_F_LEVEL         = 0x2,
00076     L4_IRQ_F_EDGE         = 0x0,
00077     L4_IRQ_F_POS           = 0x0,
00078     L4_IRQ_F_NEG           = 0x4,
00079     L4_IRQ_F_BOTH          = 0x8,
00080     L4_IRQ_F_LEVEL_HIGH   = L4_IRQ_F_SET_MODE | L4_IRQ_F_LEVEL | L4_IRQ_F_POS,
00081     L4_IRQ_F_LEVEL_LOW    = L4_IRQ_F_SET_MODE | L4_IRQ_F_LEVEL | L4_IRQ_F_NEG,
00082     L4_IRQ_F_POS_EDGE     = L4_IRQ_F_SET_MODE | L4_IRQ_F_EDGE | L4_IRQ_F_POS,
00083     L4_IRQ_F_NEG_EDGE     = L4_IRQ_F_SET_MODE | L4_IRQ_F_EDGE | L4_IRQ_F_NEG,
00084     L4_IRQ_F_BOTH_EDGE    = L4_IRQ_F_SET_MODE | L4_IRQ_F_EDGE | L4_IRQ_F_BOTH,
00085     L4_IRQ_F_MASK         = 0xf,
00086
00088     L4_IRQ_F_SET_WAKEUP    = 0x10,
00089     L4_IRQ_F_CLEAR_WAKEUP = 0x20,
00090 };
00091
00092
00097 enum L4_icu_opcode
00098 {
00104     L4_ICU_OP_BIND = 0,
00105
00111     L4_ICU_OP_UNBIND = 1,
00112
00118     L4_ICU_OP_INFO = 2,
00119
00125     L4_ICU_OP_MSI_INFO = 3,
00126
00132     L4_ICU_OP_UNMASK = 4,
00133
00139     L4_ICU_OP_MASK = 5,
00140
00146     L4_ICU_OP_SET_MODE = 6,
00147 };
00148
00149 enum L4_icu_ctl_op
00150 {
00151     L4_ICU_CTL_UNMASK = 0,
00152     L4_ICU_CTL_MASK = 1
00153 };
00154
00155
00163 typedef struct l4_icu_info_t
00164 {
00170     unsigned features;
00171
00175     unsigned nr_irqs;
00176
00180     unsigned nr_msis;
00181 } l4_icu_info_t;
00182
00184 typedef struct l4_icu_msi_info_t
00185 {
00187     l4_uint64_t msi_addr;
00189     l4_uint32_t msi_data;
00190 } l4_icu_msi_info_t;
00191

```

```

00219 L4_INLINE l4_msgtag_t
00220 l4_icu_bind(l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq) L4_NOTHROW;
00221
00222 L4_INLINE l4_msgtag_t
00223 l4_icu_bind_u(l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq,
00230 l4_utcb_t *utcb) L4_NOTHROW;
00231
00242 L4_INLINE l4_msgtag_t
00243 l4_icu_unbind(l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq) L4_NOTHROW;
00244
00251 L4_INLINE l4_msgtag_t
00252 l4_icu_unbind_u(l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq,
00253 l4_utcb_t *utcb) L4_NOTHROW;
00254
00265 L4_INLINE l4_msgtag_t
00266 l4_icu_set_mode(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t mode) L4_NOTHROW;
00267
00274 L4_INLINE l4_msgtag_t
00275 l4_icu_set_mode_u(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t mode,
00276 l4_utcb_t *utcb) L4_NOTHROW;
00277
00286 L4_INLINE l4_msgtag_t
00287 l4_icu_info(l4_cap_idx_t icu, l4_icu_info_t *info) L4_NOTHROW;
00288
00295 L4_INLINE l4_msgtag_t
00296 l4_icu_info_u(l4_cap_idx_t icu, l4_icu_info_t *info,
00297 l4_utcb_t *utcb) L4_NOTHROW;
00298
00305 L4_INLINE l4_msgtag_t
00306 l4_icu_msi_info(l4_cap_idx_t icu, unsigned irqnum, l4_uint64_t source,
00307 l4_icu_msi_info_t *msi_info) L4_NOTHROW;
00308
00315 L4_INLINE l4_msgtag_t
00316 l4_icu_msi_info_u(l4_cap_idx_t icu, unsigned irqnum, l4_uint64_t source,
00317 l4_icu_msi_info_t *msi_info, l4_utcb_t *utcb) L4_NOTHROW;
00318
00319
00337 L4_INLINE l4_msgtag_t
00338 l4_icu_unmask(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label,
00339 l4_timeout_t to) L4_NOTHROW;
00340
00347 L4_INLINE l4_msgtag_t
00348 l4_icu_unmask_u(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label,
00349 l4_timeout_t to, l4_utcb_t *utcb) L4_NOTHROW;
00350
00368 L4_INLINE l4_msgtag_t
00369 l4_icu_mask(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label,
00370 l4_timeout_t to) L4_NOTHROW;
00371
00378 L4_INLINE l4_msgtag_t
00379 l4_icu_mask_u(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label,
00380 l4_timeout_t to, l4_utcb_t *utcb) L4_NOTHROW;
00381
00385 L4_INLINE l4_msgtag_t
00386 l4_icu_control_u(l4_cap_idx_t icu, unsigned irqnum, unsigned op,
00387 l4_umword_t *label, l4_timeout_t to,
00388 l4_utcb_t *utcb) L4_NOTHROW;
00389
00390
00391 /*****
00392 * Implementations
00393 */
00394
00395 L4_INLINE l4_msgtag_t
00396 l4_icu_bind_u(l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq,
00397 l4_utcb_t *utcb) L4_NOTHROW
00398 {
00399     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00400     m->mr[0] = L4_ICU_OP_BIND;
00401     m->mr[1] = irqnum;
00402     m->mr[2] = l4_map_obj_control(0, 0);
00403     m->mr[3] = l4_obj_fpage(irq, 0, L4_CAP_FPAGE_RWS).raw;
00404     return l4_ipc_call(icu, utcb, l4_msgtag(L4_PROTO_IRQ, 2, 1, 0), L4_IPC_NEVER);
00405 }
00406
00407 L4_INLINE l4_msgtag_t
00408 l4_icu_unbind_u(l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq,
00409 l4_utcb_t *utcb) L4_NOTHROW
00410 {
00411     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00412     m->mr[0] = L4_ICU_OP_UNBIND;
00413     m->mr[1] = irqnum;
00414     m->mr[2] = l4_map_obj_control(0, 0);
00415     m->mr[3] = l4_obj_fpage(irq, 0, L4_CAP_FPAGE_RWS).raw;
00416     return l4_ipc_call(icu, utcb, l4_msgtag(L4_PROTO_IRQ, 2, 1, 0), L4_IPC_NEVER);
00417 }
00418

```

```

00419 L4_INLINE l4_msgtag_t
00420 l4_icu_info_u(l4_cap_idx_t icu, l4_icu_info_t *info,
00421              l4_utcb_t *utcb) L4_NOTHROW
00422 {
00423     l4_msgtag_t res;
00424     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00425     m->mr[0] = L4_ICU_OP_INFO;
00426     res = l4_ipc_call(icu, utcb, l4_msgtag(L4_PROTO_IRQ, 1, 0, 0), L4_IPC_NEVER);
00427     info->features = m->mr[0];
00428     info->nr_irqs = m->mr[1];
00429     info->nr_msis = m->mr[2];
00430     return res;
00431 }
00432
00433 L4_INLINE l4_msgtag_t
00434 l4_icu_msi_info_u(l4_cap_idx_t icu, unsigned irqnum, l4_uint64_t source,
00435                  l4_icu_msi_info_t *msi_info, l4_utcb_t *utcb) L4_NOTHROW
00436 {
00437     l4_msgtag_t res;
00438     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00439     m->mr[0] = L4_ICU_OP_MSI_INFO;
00440     m->mr[1] = irqnum;
00441     m->mr64[l4_utcb_mr64_idx(2)] = source;
00442     res = l4_ipc_call(icu, utcb, l4_msgtag(L4_PROTO_IRQ,
00443                                           2 + 1 * sizeof(l4_uint64_t)
00444                                           / sizeof(l4_umword_t),
00445                                           0, 0), L4_IPC_NEVER);
00446     if (L4_UNLIKELY(l4_msgtag_has_error(res)))
00447         return res;
00448
00449     if (L4_UNLIKELY(l4_msgtag_words(res) * sizeof(l4_umword_t) < sizeof(*msi_info)))
00450         return res;
00451
00452     __builtin_memcpy(msi_info, &m->mr[0], sizeof(*msi_info));
00453     return res;
00454 }
00455
00456 L4_INLINE l4_msgtag_t
00457 l4_icu_set_mode_u(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t mode,
00458                  l4_utcb_t *utcb) L4_NOTHROW
00459 {
00460     l4_msg_regs_t *mr = l4_utcb_mr_u(utcb);
00461     mr->mr[0] = L4_ICU_OP_SET_MODE;
00462     mr->mr[1] = irqnum;
00463     mr->mr[2] = mode;
00464     return l4_ipc_call(icu, utcb, l4_msgtag(L4_PROTO_IRQ, 3, 0, 0), L4_IPC_NEVER);
00465 }
00466
00467 L4_INLINE l4_msgtag_t
00468 l4_icu_control_u(l4_cap_idx_t icu, unsigned irqnum, unsigned op,
00469                  l4_umword_t *label, l4_timeout_t to,
00470                  l4_utcb_t *utcb) L4_NOTHROW
00471 {
00472     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00473     m->mr[0] = L4_ICU_OP_UNMASK + op;
00474     m->mr[1] = irqnum;
00475     if (label)
00476         return l4_ipc_send_and_wait(icu, utcb, l4_msgtag(L4_PROTO_IRQ, 2, 0, 0),
00477                                     label, to);
00478     else
00479         return l4_ipc_send(icu, utcb, l4_msgtag(L4_PROTO_IRQ, 2, 0, 0), to);
00480 }
00481
00482 L4_INLINE l4_msgtag_t
00483 l4_icu_mask_u(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label,
00484               l4_timeout_t to, l4_utcb_t *utcb) L4_NOTHROW
00485 { return l4_icu_control_u(icu, irqnum, L4_ICU_CTL_MASK, label, to, utcb); }
00486
00487 L4_INLINE l4_msgtag_t
00488 l4_icu_unmask_u(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label,
00489                 l4_timeout_t to, l4_utcb_t *utcb) L4_NOTHROW
00490 { return l4_icu_control_u(icu, irqnum, L4_ICU_CTL_UNMASK, label, to, utcb); }
00491
00492
00493
00494
00495 L4_INLINE l4_msgtag_t
00496 l4_icu_bind(l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq) L4_NOTHROW
00497 { return l4_icu_bind_u(icu, irqnum, irq, l4_utcb()); }
00498
00499 L4_INLINE l4_msgtag_t
00500 l4_icu_unbind(l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq) L4_NOTHROW
00501 { return l4_icu_unbind_u(icu, irqnum, irq, l4_utcb()); }
00502
00503 L4_INLINE l4_msgtag_t
00504 l4_icu_info(l4_cap_idx_t icu, l4_icu_info_t *info) L4_NOTHROW
00505 { return l4_icu_info_u(icu, info, l4_utcb()); }

```

```

00506
00507 L4_INLINE l4_msgtag_t
00508 l4_icu_msi_info(l4_cap_idx_t icu, unsigned irqnum, l4_uint64_t source,
00509                l4_icu_msi_info_t *msi_info) L4_NOTHROW
00510 { return l4_icu_msi_info_u(icu, irqnum, source, msi_info, l4_utcb()); }
00511
00512 L4_INLINE l4_msgtag_t
00513 l4_icu_unmask(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label,
00514              l4_timeout_t to) L4_NOTHROW
00515 { return l4_icu_control_u(icu, irqnum, L4_ICU_CTL_UNMASK, label, to, l4_utcb()); }
00516
00517 L4_INLINE l4_msgtag_t
00518 l4_icu_mask(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label,
00519            l4_timeout_t to) L4_NOTHROW
00520 { return l4_icu_control_u(icu, irqnum, L4_ICU_CTL_MASK, label, to, l4_utcb()); }
00521
00522 L4_INLINE l4_msgtag_t
00523 l4_icu_set_mode(l4_cap_idx_t icu, unsigned irqnum, l4_umword_t mode) L4_NOTHROW
00524 {
00525     return l4_icu_set_mode_u(icu, irqnum, mode, l4_utcb());
00526 }

```

16.522 iommu

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /* \file
00003  * IO-MMU interface description.
00004  */
00005 #pragma once
00006
00007 #include <l4/sys/cxx/ipc_iface>
00008
00009 namespace L4 {
00021 class Iommu :
00022     public Kobject_x<Iommu, Proto_t<L4_PROTO_IOMMU>, Type_info::Demand_t<1> >
00023 {
00024 public:
00037     L4_INLINE_RPC(
00038         l4_msgtag_t, bind, (l4_uint64_t src_id, Ipc::Cap<Task> dma_space));
00039
00050     L4_INLINE_RPC(
00051         l4_msgtag_t, unbind, (l4_uint64_t src_id, Ipc::Cap<Task> dma_space));
00052
00053     typedef Typeid::Rpc_code<l4_umword_t>::F<bind_t, unbind_t> Rpcs;
00054 };
00055
00056 }

```

16.523 ipc.h

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *      Frank Mehnert <fm3@os.inf.tu-dresden.de>,
00010  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00011  *      economic rights: Technische Universität Dresden (Germany)
00012  *
00013  * License: see LICENSE.spdx (in this directory or the directories above)
00014  */
00015 #ifndef __L4SYS__INCLUDE__ARCH_AMD64__L4API_L4F__IPC_H__
00016 #define __L4SYS__INCLUDE__ARCH_AMD64__L4API_L4F__IPC_H__
00017
00018 L4_INLINE l4_msgtag_t
00019 l4_ipc(l4_cap_idx_t dest, l4_utcb_t *utcb,
00020        l4_umword_t flags,
00021        l4_umword_t slabel,
00022        l4_msgtag_t tag,
00023        l4_umword_t *rlabel,
00024        l4_timeout_t timeout) L4_NOTHROW
00025 {
00026     l4_umword_t dummy, dummy2;
00027     register l4_umword_t to __asm__("r8") = timeout.raw;
00028
00029     (void)utcb;
00030
00031     __asm__ __volatile__
00032     ("syscall"

```

```

00033     :
00034     "=d" (dummy2),
00035     "=S" (slabel),
00036     "=D" (dummy),
00037     "=a" (tag.raw)
00038     :
00039     "S" (slabel),
00040     "r" (to),
00041     "a" (tag.raw),
00042     "d" (dest | flags)
00043     :
00044     "memory", "cc", "rcx", "r11", "r15"
00045     );
00046
00047     if (rlabel)
00048         *rlabel = slabel;
00049
00050     return tag;
00051 }
00052
00053 #endif /* ! __L4SYS__INCLUDE__ARCH_AMD64__L4API_L4F__IPC_H__ */

```

16.524 ipc.h

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013 #pragma once
00014
00015 #include <l4/sys/compiler.h>
00016 #include <l4/sys/syscall_defs.h>
00017
00018 L4_INLINE l4_msgtag_t
00019 l4_ipc(l4_cap_idx_t dest, l4_utcb_t *utcb,
00020        l4_umword_t flags,
00021        l4_umword_t slabel,
00022        l4_msgtag_t tag,
00023        l4_umword_t *rlabel,
00024        l4_timeout_t timeout) L4_NOTHROW
00025 {
00026     register l4_umword_t _dest    __asm__("r2") = dest | flags;
00027     register l4_umword_t _timeout __asm__("r3") = timeout.raw;
00028     register l4_mword_t _tag      __asm__("r0") = tag.raw;
00029     register l4_umword_t _label   __asm__("r4") = slabel;
00030     (void)utcb;
00031
00032     __asm__ __volatile__
00033     ("type __l4_sys_syscall, #function\n"
00034      "mov r5, %[sc]          \n"
00035      "blx __l4_sys_syscall  \n"
00036      :
00037      "+r" (_dest),
00038      "+r" (_timeout),
00039      "+r" (_label),
00040      "+r" (_tag)
00041      :
00042      [sc] "i" (L4_SYSCALL_INVOKE)
00043      :
00044      "cc", "memory", "r5", "ip", "lr");
00045
00046     if (rlabel)
00047         *rlabel = _label;
00048     tag.raw = _tag;
00049
00050     return tag;
00051 }

```

16.525 ipc.h

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>

```

```

00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013 #pragma once
00014
00015 #include <l4/sys/compiler.h>
00016
00017 L4_BEGIN_DECLS
00018
00019 struct __l4_sys_syscall_res
00020 {
00021     l4_mword_t tag;
00022     l4_umword_t label;
00023 };
00024
00025 extern struct __l4_sys_syscall_res
00026 __l4_sys_syscall(l4_mword_t tag,
00027                 l4_umword_t slabel,
00028                 l4_umword_t dest,
00029                 l4_umword_t timeout) L4_NOTHROW;
00030
00031 L4_END_DECLS
00032
00033 L4_INLINE l4_msgtag_t
00034 l4_ipc(l4_cap_idx_t dest, l4_utcb_t *utcb,
00035        l4_umword_t flags,
00036        l4_umword_t slabel,
00037        l4_msgtag_t tag,
00038        l4_umword_t *rlabel,
00039        l4_timeout_t timeout) L4_NOTHROW
00040 {
00041     // No need for memory clobbers. The compiler has to assume that all global
00042     // data is read/written because __l4_sys_syscall is implemented in a
00043     // different translation unit.
00044     struct __l4_sys_syscall_res res
00045         = __l4_sys_syscall(tag.raw, slabel, dest | flags, timeout.raw);
00046
00047     (void)utcb;
00048
00049     if (rlabel)
00050         *rlabel = res.label;
00051     tag.raw = res.tag;
00052
00053     return tag;
00054 }

```

16.526 l4/sys/ipc.h File Reference

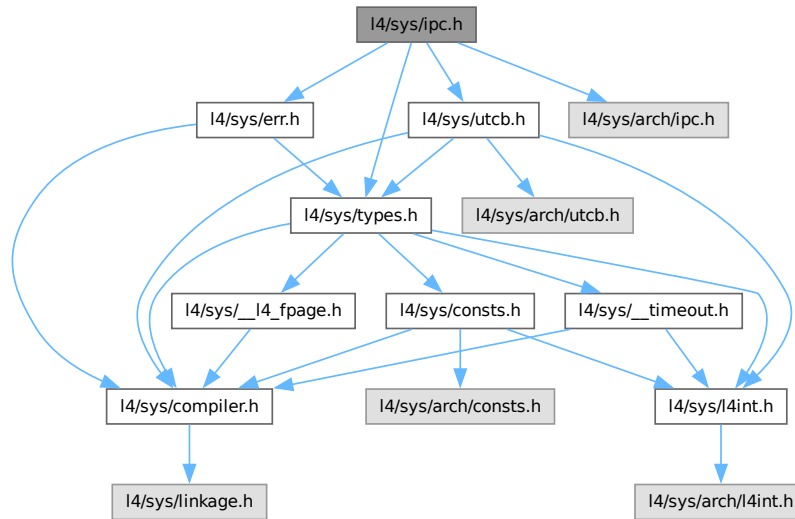
Common IPC interface.

```

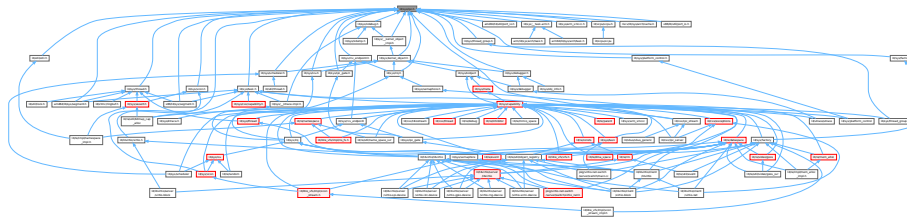
#include <l4/sys/types.h>
#include <l4/sys/utcb.h>
#include <l4/sys/err.h>
#include <l4/sys/arch/ipc.h>

```


Include dependency graph for ipc.h:



This graph shows which files directly or indirectly include this file:



Enumerations

- enum `l4_ipc_tcr_error_t` {
`L4_IPC_ERROR_MASK` = 0x1F , `L4_IPC_SND_ERR_MASK` = 0x01 , `L4_IPC_ENOT_EXISTENT` = 0x04 ,
`L4_IPC_RETIMEOUT` = 0x03 ,
`L4_IPC_SETIMEOUT` = 0x02 , `L4_IPC_RECANCELED` = 0x07 , `L4_IPC_SECANCELED` = 0x06 ,
`L4_IPC_REMAPFAILED` = 0x11 ,
`L4_IPC_SEMAPFAILED` = 0x10 , `L4_IPC_RESNDPFTO` = 0x0b , `L4_IPC_SESNDPFTO` = 0x0a ,
`L4_IPC_RERCVPFTO` = 0x0d ,
`L4_IPC_SERCVPFTO` = 0x0c , `L4_IPC_REABORTED` = 0x0f , `L4_IPC_SEABORTED` = 0x0e ,
`L4_IPC_REMSGCUT` = 0x09 ,
`L4_IPC_SEMSGCUT` = 0x08 , `L4_IPC_NO_REPLY_CAP` = 0x13 }
Error codes in the error TCR.

Functions

- `l4_umword_t l4_ipc_error (l4_msgtag_t tag, l4_utcb_t *utcb) L4_NOTHROW`
Get the IPC error code for an IPC operation.
- `l4_ret_t l4_error (l4_msgtag_t tag) L4_NOTHROW`

- Get IPC error code if any or message tag label otherwise for an IPC call.*

 - `int l4_ipc_is_snd_error (l4_utcb_t *utcb) L4_NOTHROW`
Returns whether an error occurred in send phase of an invocation.
 - `int l4_ipc_is_rcv_error (l4_utcb_t *utcb) L4_NOTHROW`
Returns whether an error occurred in receive phase of an invocation.
 - `int l4_ipc_error_code (l4_utcb_t *utcb) L4_NOTHROW`
Get the error condition of the last invocation from the TCR.
 - `L4_CONSTEXPR l4_ret_t l4_ipc_to_errno (unsigned long ipc_error_code) L4_NOTHROW`
Get a negative error code for the given IPC error code.
 - `l4_msgtag_t l4_ipc_send (l4_cap_idx_t dest, l4_utcb_t *utcb, l4_msgtag_t tag, l4_timeout_t timeout) L4_NOTHROW`
*Send a message to an object (do **not** wait for a reply).*
 - `l4_msgtag_t l4_ipc_wait (l4_utcb_t *utcb, l4_umword_t *label, l4_timeout_t timeout) L4_NOTHROW`
Wait for an incoming message from any possible sender.
 - `l4_msgtag_t l4_ipc_receive (l4_cap_idx_t object, l4_utcb_t *utcb, l4_timeout_t timeout) L4_NOTHROW`
Wait for a message from a specific source.
 - `l4_msgtag_t l4_ipc_call (l4_cap_idx_t object, l4_utcb_t *utcb, l4_msgtag_t tag, l4_timeout_t timeout) L4_NOTHROW`
Object call (usual invocation).
 - `l4_msgtag_t l4_ipc_reply (l4_cap_idx_t reply_cap, l4_utcb_t *utcb, l4_msgtag_t tag, l4_timeout_t timeout) L4_NOTHROW`
Reply operation (uses a reply capability).
 - `l4_msgtag_t l4_ipc_reply_and_wait (l4_utcb_t *utcb, l4_msgtag_t tag, l4_umword_t *label, l4_timeout_t timeout) L4_NOTHROW`
Reply and wait operation (uses the reply capability).
 - `l4_msgtag_t l4_ipc_send_and_wait (l4_cap_idx_t dest, l4_utcb_t *utcb, l4_msgtag_t tag, l4_umword_t *label, l4_timeout_t timeout) L4_NOTHROW`
Send a message and do an open wait.
 - `l4_msgtag_t l4_ipc (l4_cap_idx_t dest, l4_utcb_t *utcb, l4_umword_t flags, l4_umword_t slabel, l4_msgtag_t tag, l4_umword_t *rlabel, l4_timeout_t timeout) L4_NOTHROW`
Generic L4 object invocation.
 - `l4_msgtag_t l4_ipc_sleep (l4_timeout_t timeout) L4_NOTHROW`
Sleep for an amount of time.
 - `l4_msgtag_t l4_ipc_sleep_ms (l4_uint32_t ms) L4_NOTHROW`
Sleep for a certain amount of milliseconds.
 - `l4_msgtag_t l4_ipc_sleep_us (l4_uint64_t us) L4_NOTHROW`
Sleep for a certain amount of microseconds.
 - `int l4_sndfpage_add (l4_fpage_t const snd_fpage, unsigned long snd_base, l4_msgtag_t *tag) L4_NOTHROW`
Add a flexpage to be sent to the UTCB.

16.526.1 Detailed Description

Common IPC interface.

Definition in file [ipc.h](#).

16.526.2 Function Documentation

16.526.2.1 l4_ipc_to_errno()

```
L4_CONSTEXPR l4_ret_t l4_ipc_to_errno (
    unsigned long ipc_error_code) [inline]
```

Get a negative error code for the given IPC error code.

Parameters

| | |
|-----------------------|---|
| <i>ipc_error_code</i> | IPC error code as delivered by the kernel. (or returned by the l4_ipc_error_code() function). |
|-----------------------|---|

Returns

negative error code in the range of [L4_EIPC_LO](#) to [L4_EIPC_HI](#).

Definition at line 595 of file [ipc.h](#).

References [L4_EIPC_LO](#), and [L4_NOTHROW](#).

Referenced by [L4::Reply_cap::reply\(\)](#).

Here is the caller graph for this function:



16.527 ipc.h

[Go to the documentation of this file.](#)

```
00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *               Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *               Björn Döbel <doebel@os.inf.tu-dresden.de>,
00010  *               Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00011  *               economic rights: Technische Universität Dresden (Germany)
00012  *
00013  * License: see LICENSE.spdx (in this directory or the directories above)
00014  */
00015 #ifndef __L4SYS__INCLUDE__L4API_FIASCO_IPC_H__
00016 #define __L4SYS__INCLUDE__L4API_FIASCO_IPC_H__
00017
00018 #include <l4/sys/types.h>
00019 #include <l4/sys/utcb.h>
00020 #include <l4/sys/err.h>
00021
00061
00062 /*****
00063  *** IPC result checking
00064  *****/
```

```

00065
00073
00081 enum l4_ipc_tcr_error_t
00082 {
00083     L4_IPC_ERROR_MASK          = 0x1F,
00084     L4_IPC_SND_ERR_MASK       = 0x01,
00085
00086     L4_IPC_ENOT_EXISTENT      = 0x04,
00089     L4_IPC_RETIMEOUT         = 0x03,
00092     L4_IPC_SETTIMEOUT        = 0x02,
00095     L4_IPC_RECANCELED        = 0x07,
00098     L4_IPC_SECANCELED        = 0x06,
00101     L4_IPC_REMAPFAILED       = 0x11,
00105     L4_IPC_SEMAPFAILED       = 0x10,
00108     L4_IPC_RESNDPFTO        = 0x0b,
00112     L4_IPC_SESNDPFTO        = 0x0a,
00116     L4_IPC_RERCVPFTO        = 0x0d,
00120     L4_IPC_SERCVPFTO        = 0x0c,
00124     L4_IPC_REABORTED        = 0x0f,
00127     L4_IPC_SEABORTED        = 0x0e,
00136     L4_IPC_REMSGCUT         = 0x09,
00137
00143     L4_IPC_SEMSGCUT          = 0x08,
00144
00150     L4_IPC_NO_REPLY_CAP      = 0x13,
00151 };
00152
00153
00164 L4_INLINE l4_umword_t
00165 l4_ipc_error(l4_msgtag_t tag, l4_utcb_t *utcb) L4_NOTHROW;
00166
00167
00184 L4_INLINE l4_ret_t
00185 l4_error(l4_msgtag_t tag) L4_NOTHROW;
00186
00187 L4_INLINE l4_ret_t
00188 l4_error_u(l4_msgtag_t tag, l4_utcb_t *utcb) L4_NOTHROW;
00189
00190 /*****
00191 *** IPC results
00192 *****/
00193
00203 L4_INLINE int l4_ipc_is_snd_error(l4_utcb_t *utcb) L4_NOTHROW;
00204
00214 L4_INLINE int l4_ipc_is_rcv_error(l4_utcb_t *utcb) L4_NOTHROW;
00215
00225 L4_INLINE int l4_ipc_error_code(l4_utcb_t *utcb) L4_NOTHROW;
00226
00233 L4_CONSTEXPR L4_INLINE l4_ret_t
00234 l4_ipc_to_errno(unsigned long ipc_error_code) L4_NOTHROW;
00235
00236
00237 /*****
00238 *** IPC calls
00239 *****/
00240
00269 L4_INLINE l4_msgtag_t
00270 l4_ipc_send(l4_cap_idx_t dest, l4_utcb_t *utcb, l4_msgtag_t tag,
00271             l4_timeout_t timeout) L4_NOTHROW;
00272
00273
00300 L4_INLINE l4_msgtag_t
00301 l4_ipc_wait(l4_utcb_t *utcb, l4_umword_t *label,
00302             l4_timeout_t timeout) L4_NOTHROW;
00303
00304
00331 L4_INLINE l4_msgtag_t
00332 l4_ipc_receive(l4_cap_idx_t object, l4_utcb_t *utcb,
00333                l4_timeout_t timeout) L4_NOTHROW;
00334
00365 L4_INLINE l4_msgtag_t
00366 l4_ipc_call(l4_cap_idx_t object, l4_utcb_t *utcb, l4_msgtag_t tag,
00367             l4_timeout_t timeout) L4_NOTHROW;
00368
00389 L4_INLINE l4_msgtag_t
00390 l4_ipc_reply(l4_cap_idx_t reply_cap, l4_utcb_t *utcb, l4_msgtag_t tag,
00391             l4_timeout_t timeout) L4_NOTHROW;
00392
00418 L4_INLINE l4_msgtag_t
00419 l4_ipc_reply_and_wait(l4_utcb_t *utcb, l4_msgtag_t tag,
00420                       l4_umword_t *label, l4_timeout_t timeout) L4_NOTHROW;
00421
00450 L4_INLINE l4_msgtag_t
00451 l4_ipc_send_and_wait(l4_cap_idx_t dest, l4_utcb_t *utcb, l4_msgtag_t tag,
00452                      l4_umword_t *label, l4_timeout_t timeout) L4_NOTHROW;
00453
00459

```

```

00460 #if 0
00471 L4_INLINE l4_msgtag_t
00472 l4_ipc_wait_next_period(l4_utcb_t *utcb,
00473                         l4_umword_t *label,
00474                         l4_timeout_t timeout);
00475
00476 #endif
00477
00497 L4_ALWAYS_INLINE l4_msgtag_t
00498 l4_ipc(l4_cap_idx_t dest,
00499        l4_utcb_t *utcb,
00500        l4_umword_t flags,
00501        l4_umword_t slabel,
00502        l4_msgtag_t tag,
00503        l4_umword_t *rlabel,
00504        l4_timeout_t timeout) L4_NOTHROW;
00505
00522 L4_INLINE l4_msgtag_t
00523 l4_ipc_sleep(l4_timeout_t timeout) L4_NOTHROW;
00524
00541 L4_INLINE l4_msgtag_t
00542 l4_ipc_sleep_ms(l4_uint32_t ms) L4_NOTHROW;
00543
00560 L4_INLINE l4_msgtag_t
00561 l4_ipc_sleep_us(l4_uint64_t us) L4_NOTHROW;
00562
00577 L4_INLINE int
00578 l4_sndfpage_add(l4_fpage_t const snd_fpage, unsigned long snd_base,
00579                 l4_msgtag_t *tag) L4_NOTHROW;
00580
00581 /*
00582  * \internal
00583  * \ingroup l4_ipc_api
00584  */
00585 L4_INLINE int
00586 l4_sndfpage_add_u(l4_fpage_t const snd_fpage, unsigned long snd_base,
00587                  l4_msgtag_t *tag, l4_utcb_t *utcb) L4_NOTHROW;
00588
00589
00590 /*****
00591  * Implementations
00592  *****/
00593
00594 L4_CONSTEXPR L4_INLINE l4_ret_t
00595 l4_ipc_to_errno(unsigned long ipc_error_code) L4_NOTHROW
00596 { return -(L4_EIPC_LO + ipc_error_code); }
00597
00598 L4_INLINE l4_msgtag_t
00599 l4_ipc_call(l4_cap_idx_t object, l4_utcb_t *utcb, l4_msgtag_t tag,
00600            l4_timeout_t timeout) L4_NOTHROW
00601 {
00602     return l4_ipc(object, utcb, L4_SYSF_CALL, 0, tag, 0, timeout);
00603 }
00604
00605 L4_INLINE l4_msgtag_t
00606 l4_ipc_reply(l4_cap_idx_t reply_cap, l4_utcb_t *utcb, l4_msgtag_t tag,
00607            l4_timeout_t timeout) L4_NOTHROW
00608 {
00609     return l4_ipc(reply_cap, utcb, L4_SYSF_REPLY | L4_SYSF_SEND, 0, tag, 0,
00610                  timeout);
00611 }
00612
00613 L4_INLINE l4_msgtag_t
00614 l4_ipc_reply_and_wait(l4_utcb_t *utcb, l4_msgtag_t tag,
00615                      l4_umword_t *label, l4_timeout_t timeout) L4_NOTHROW
00616 {
00617     return l4_ipc(L4_INVALID_CAP, utcb, L4_SYSF_REPLY_AND_WAIT, 0, tag, label, timeout);
00618 }
00619
00620 L4_INLINE l4_msgtag_t
00621 l4_ipc_send_and_wait(l4_cap_idx_t dest, l4_utcb_t *utcb, l4_msgtag_t tag,
00622                     l4_umword_t *label, l4_timeout_t timeout) L4_NOTHROW
00623 {
00624     return l4_ipc(dest, utcb, L4_SYSF_SEND_AND_WAIT, 0, tag, label, timeout);
00625 }
00626
00627 L4_INLINE l4_msgtag_t
00628 l4_ipc_send(l4_cap_idx_t dest, l4_utcb_t *utcb, l4_msgtag_t tag,
00629            l4_timeout_t timeout) L4_NOTHROW
00630 {
00631     return l4_ipc(dest, utcb, L4_SYSF_SEND, 0, tag, 0, timeout);
00632 }
00633
00634 L4_INLINE l4_msgtag_t
00635 l4_ipc_wait(l4_utcb_t *utcb, l4_umword_t *label,
00636            l4_timeout_t timeout) L4_NOTHROW
00637 {

```

```

00638     l4_msgtag_t t;
00639     t.raw = 0;
00640     return l4_ipc(L4_INVALID_CAP, utcb, L4_SYSF_WAIT, 0, t, label, timeout);
00641 }
00642
00643 L4_INLINE l4_msgtag_t
00644 l4_ipc_receive(l4_cap_idx_t object, l4_utcb_t *utcb,
00645               l4_timeout_t timeout) L4_NOTHROW
00646 {
00647     l4_msgtag_t t;
00648     t.raw = 0;
00649     return l4_ipc(object, utcb, L4_SYSF_RECV, 0, t, 0, timeout);
00650 }
00651
00652 L4_INLINE l4_msgtag_t
00653 l4_ipc_sleep(l4_timeout_t timeout) L4_NOTHROW
00654 { return l4_ipc_receive(L4_INVALID_CAP, NULL, timeout); }
00655
00656 L4_INLINE l4_msgtag_t
00657 l4_ipc_sleep_ms(l4_uint32_t ms) L4_NOTHROW
00658 {
00659     l4_uint64_t us = ms * 1000ULL; // cannot overflow because ms < 2^32
00660     return l4_ipc_sleep(l4_timeout(L4_IPC_TIMEOUT_NEVER, l4_timeout_from_us(us)));
00661 }
00662
00663 L4_INLINE l4_msgtag_t
00664 l4_ipc_sleep_us(l4_uint64_t us) L4_NOTHROW
00665 {
00666     return l4_ipc_sleep(l4_timeout(L4_IPC_TIMEOUT_NEVER,
00667                                   l4_timeout_from_us(us)));
00668 }
00669
00670 L4_INLINE l4_umword_t
00671 l4_ipc_error(l4_msgtag_t tag, l4_utcb_t *utcb) L4_NOTHROW
00672 {
00673     if (L4_LIKELY(!l4_msgtag_has_error(tag)))
00674         return 0;
00675     return l4_utcb_tcr_u(utcb)->error & L4_IPC_ERROR_MASK;
00676 }
00677
00678 L4_INLINE l4_ret_t
00679 l4_error_u(l4_msgtag_t tag, l4_utcb_t *u) L4_NOTHROW
00680 {
00681     if (L4_UNLIKELY(l4_msgtag_has_error(tag)))
00682         return l4_ipc_to_errno(l4_utcb_tcr_u(u)->error & L4_IPC_ERROR_MASK);
00683
00684     return l4_msgtag_label(tag);
00685 }
00686
00687 L4_INLINE l4_ret_t
00688 l4_error(l4_msgtag_t tag) L4_NOTHROW
00689 {
00690     return l4_error_u(tag, l4_utcb());
00691 }
00692
00693
00694 L4_INLINE int l4_ipc_is_snd_error(l4_utcb_t *u) L4_NOTHROW
00695 { return (l4_utcb_tcr_u(u)->error & 1) == 0; }
00696
00697 L4_INLINE int l4_ipc_is_rcv_error(l4_utcb_t *u) L4_NOTHROW
00698 { return l4_utcb_tcr_u(u)->error & 1; }
00699
00700 L4_INLINE int l4_ipc_error_code(l4_utcb_t *u) L4_NOTHROW
00701 { return l4_utcb_tcr_u(u)->error & L4_IPC_ERROR_MASK; }
00702
00703
00704 /*
00705  * \internal
00706  * \ingroup l4_ipc_api
00707  */
00708 L4_INLINE int
00709 l4_sndfpage_add_u(l4_fpage_t const snd_fpage, unsigned long snd_base,
00710                  l4_msgtag_t *tag, l4_utcb_t *utcb) L4_NOTHROW
00711 {
00712     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00713     int i = l4_msgtag_words(*tag) + 2 * l4_msgtag_items(*tag);
00714
00715     if (i >= L4_UTCB_GENERIC_DATA_SIZE - 1)
00716         return -L4_ENOMEM;
00717
00718     v->mr[i] = snd_base | L4_ITEM_MAP | L4_ITEM_CONT;
00719     v->mr[i + 1] = snd_fpage.raw;
00720
00721     *tag = l4_msgtag(l4_msgtag_label(*tag), l4_msgtag_words(*tag),
00722                     l4_msgtag_items(*tag) + 1, l4_msgtag_flags(*tag));
00723     return 0;
00724 }

```

```

00725
00726 L4_INLINE int
00727 l4_sndfpage_add(l4_fpage_t const snd_fpage, unsigned long snd_base,
00728                 l4_msgtag_t *tag) L4_NOTHROW
00729 {
00730     return l4_sndfpage_add_u(snd_fpage, snd_base, tag, l4_utcb());
00731 }
00732
00733 #include <l4/sys/arch/ipc.h>
00734
00735 #endif /* ! __L4SYS__INCLUDE__L4API_FIASCO__IPC_H__ */

```

16.528 ipc.h

```

00001
00006 /*
00007  * Copyright (C) 2021, 2024-2025 Kernkonzept GmbH.
00008  * Author(s): Georg Kotheimer <georg.kotheimer@kernkonzept.com>
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #pragma once
00013
00014 #include <l4/sys/compiler.h>
00015
00016 L4_INLINE l4_msgtag_t
00017 l4_ipc(l4_cap_idx_t dest, l4_utcb_t *utcb,
00018        l4_umword_t flags,
00019        l4_umword_t slabel,
00020        l4_msgtag_t tag,
00021        l4_umword_t *rlabel,
00022        l4_timeout_t timeout) L4_NOTHROW
00023 {
00024     register l4_mword_t _tag      __asm__("a0") = tag.raw;
00025     register l4_umword_t _dest    __asm__("a1") = dest | flags;
00026     register l4_umword_t _timeout __asm__("a2") = timeout.raw;
00027     register l4_umword_t _label   __asm__("a3") = slabel;
00028     (void)utcb;
00029
00030     __asm__ __volatile__
00031     ("ecall"
00032      :
00033      : "+r" (_tag),
00034      : "+r" (_dest),
00035      : "+r" (_timeout),
00036      : "+r" (_label)
00037      :
00038      :
00039      : "memory");
00040
00041     if (rlabel)
00042         *rlabel = _label;
00043     tag.raw = _tag;
00044
00045     return tag;
00046 }

```

16.529 x86/l4/sys/arch/ipc.h File Reference

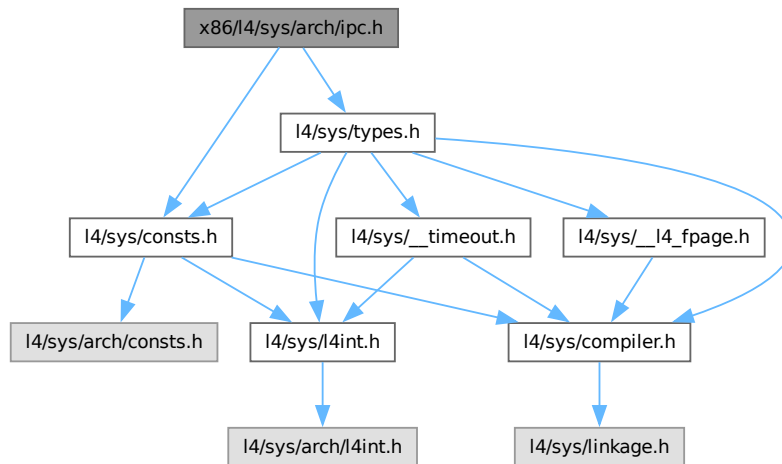
[L4 IPC System Calls, x86.](#)

```

#include <l4/sys/types.h>
#include <l4/sys/consts.h>

```

Include dependency graph for ipc.h:



Functions

- [l4_msgtag_t l4_ipc](#) ([l4_cap_idx_t](#) dest, [l4_utcb_t](#) *u, [l4_umword_t](#) flags, [l4_umword_t](#) slabel, [l4_msgtag_t](#) tag, [l4_umword_t](#) *rlabel, [l4_timeout_t](#) timeout) [L4_NOTHROW](#)

Generic L4 object invocation.

16.529.1 Detailed Description

[L4 IPC System Calls](#), x86.

Definition in file [ipc.h](#).

16.530 ipc.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00009  *      Lars Reuther <reuther@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014 #ifndef __L4_IPC_H__
00015 #define __L4_IPC_H__
00016
00017 #include <l4/sys/types.h>
00018 #include <l4/sys/consts.h>
00019
00020 #ifdef __PIC__
00021 # define L4S_PIC_SAVE "push %%ebx; "
00022 # define L4S_PIC_RESTORE "pop %%ebx; "
00023 # define L4S_PIC_CLOBBER
00024 # define L4S_PIC_SYSCALL , [func] "m" (__l4sys_invoke_indirect)
00025 extern void (*__l4sys_invoke_indirect)(void);

```



```

00026 # define IPC_SYSENTER      "# indirect sys invoke \n\t" \
00027                               "call *%[func] \n\t"
00028 # define IPC_SYSENTER_ASM    call __l4sys_invoke_direct@plt
00029 #else
00034 #define IPC_SYSENTER        "call __l4sys_invoke_direct \n\t"
00039 #define IPC_SYSENTER_ASM    call __l4sys_invoke_direct
00044 # define L4S_PIC_SAVE
00049 # define L4S_PIC_RESTORE
00054 # define L4S_PIC_CLOBBER , "ebx"
00055 # define L4S_PIC_SYSCALL
00056
00057 #endif
00058
00059 L4_INLINE l4_msgtag_t
00060 l4_ipc(l4_cap_idx_t dest, l4_utcb_t *u,
00061        l4_umword_t flags,
00062        l4_umword_t slabel,
00063        l4_msgtag_t tag,
00064        l4_umword_t *rlabel,
00065        l4_timeout_t timeout) L4_NOTHROW
00066 {
00067     l4_umword_t dummy, dummy1, dummy2;
00068
00069     (void)u;
00070
00071     __asm__ __volatile__
00072     (L4S_PIC_SAVE "push %%ebp; " IPC_SYSENTER " pop %%ebp; " L4S_PIC_RESTORE
00073      :
00074      "=d" (dummy2),
00075      "=S" (slabel),
00076      "=c" (dummy1),
00077      "=D" (dummy),
00078      "=a" (tag.raw)
00079      :
00080      "S" (slabel),
00081      "c" (timeout),
00082      "a" (tag.raw),
00083      "d" (dest | flags)
00084      : L4S_PIC_SYSCALL
00085      :
00086      "memory", "cc" L4S_PIC_CLOBBER
00087      );
00088
00089     if (rlabel)
00090         *rlabel = slabel;
00091
00092     return tag;
00093 }
00094
00095 #endif /* !__L4_IPC_H__ */

```

16.531 l4/sys/ipc_gate File Reference

The C++ IPC gate interface.

```

#include <l4/sys/ipc_gate.h>
#include <l4/sys/capability>
#include <l4/sys/rcv_endpoint>
#include <l4/sys/cxx/ipc_iface>

```


16.532 ipc_gate

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2009-2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013 #pragma once
00014
00015 #include <l4/sys/ipc_gate.h>
00016 #include <l4/sys/capability>
00017 #include <l4/sys/rcv_endpoint>
00018 #include <l4/sys/cxx/ipc_iface>
00019
00020 namespace L4 {
00021
00022 class Thread;
00023
00085 class L4_EXPORT Ipc_gate :
00086     public Kobject_t<Ipc_gate, Rcv_endpoint, L4_PROTO_KOBJECT,
00087         Type_info::Demand_t<1> >
00088 {
00089 public:
00103     L4_INLINE_RPC_OP(L4_IPC_GATE_GET_INFO_OP,
00104         l4_msgtag_t, get_infos, (l4_umword_t *label));
00105
00106     typedef L4::Typeid::Rpcsys<bind_thread_t, get_infos_t> Rpcsys;
00107 };
00108
00109 }

```

16.533 l4/sys/ipc_gate.h File Reference

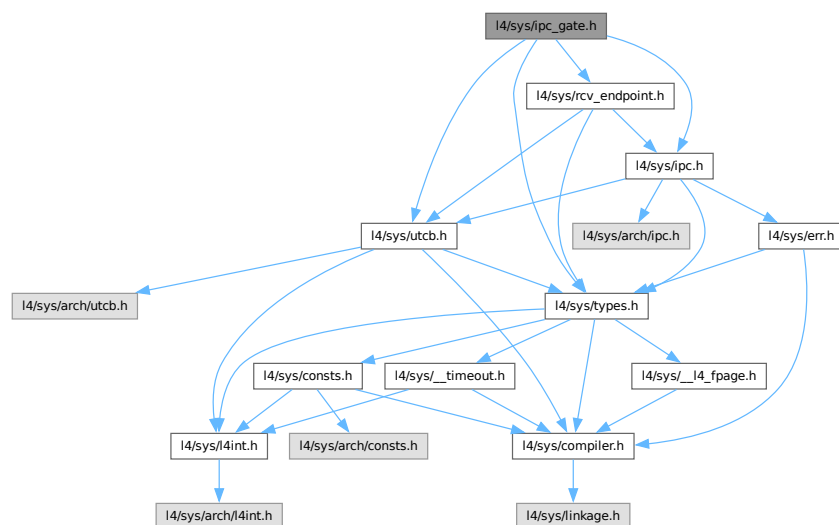
The C IPC gate interface, see [L4::ipc_gate](#) for the C++ interface.

```

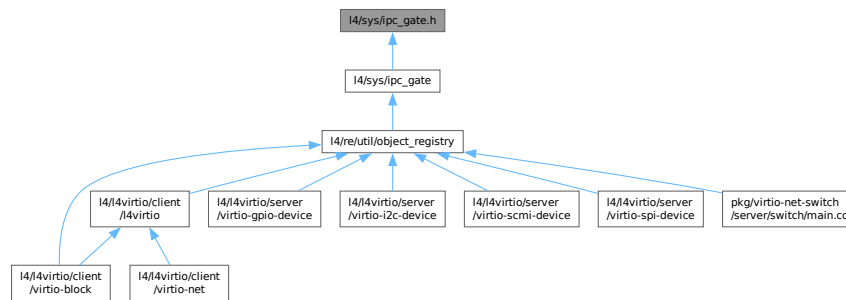
#include <l4/sys/utcb.h>
#include <l4/sys/types.h>
#include <l4/sys/rcv_endpoint.h>
#include <l4/sys/ipc.h>

```

Include dependency graph for ipc_gate.h:



This graph shows which files directly or indirectly include this file:



Enumerations

- enum `L4_ipc_gate_ops` { `L4_IPC_GATE_BIND_OP` = 0x10 , `L4_IPC_GATE_GET_INFO_OP` = 0x11 }
- Operations on the IPC-gate.*

Functions

- `l4_msgtag_t l4_ipc_gate_get_infos` (`l4_cap_idx_t` gate, `l4_umword_t *`label)
- Get information about the IPC-gate.*

16.533.1 Detailed Description

The C IPC gate interface, see [L4::ipc_gate](#) for the C++ interface.

IPC gates are used to create secure communication channels between protection domains. An IPC gate can be created using the [Factory](#) interface.

Depending on the permissions of the capability used, an IPC gate forwards IPC to the [Thread](#) the IPC gate is *bound* to (cf. [l4_rcv_ep_bind_thread\(\)](#) and [l4_rcv_ep_bind_snd_destination\(\)](#)). If the capability has the [L4_FPAGE_C_IPCGATE_SVR](#) permission, only IPC using a protocol different from the [L4_PROTO_KOBJECT](#) protocol is forwarded. Without the [L4_FPAGE_C_IPCGATE_SVR](#) permission, all IPC is forwarded. The latter is the usual case for a client in a client/server scenario. When not bound to a thread or thread group yet, the forwarded IPC blocks until the IPC gate is bound to a thread or thread group, or the IPC times out.

Forwarded IPC is always forwarded to the userland of the thread the IPC gate is bound to, either directly or indirectly using a thread group. That means, the [Thread](#) interface of that thread is not accessible via an IPC gate. The [IPC-Gate API](#) of an IPC gate is only accessible if the capability used has the [L4_FPAGE_C_IPCGATE_SVR](#) permission (cf. previous paragraph). Conversely that means, if the capability used lacks the [L4_FPAGE_C_IPCGATE_SVR](#) permission, [IPC-Gate API](#) calls are forwarded to the thread or thread group the IPC gate is bound to instead of being processed by the IPC gate itself. In a client/server scenario, a client should only get IPC gate capabilities without [L4_FPAGE_C_IPCGATE_SVR](#) permission so the client cannot tamper with the IPC gate.

When binding an IPC gate to a thread or thread group, a user-defined, kernel protected, machine-word sized payload called the IPC gate's *label* is assigned to the IPC gate (note that the two least significant bits of the label must be zero; cf. [l4_rcv_ep_bind_thread\(\)](#) and [l4_rcv_ep_bind_snd_destination\(\)](#)). When a send-only IPC or call IPC is forwarded via an IPC gate, the label provided by the sender is ignored and replaced by the IPC gate's label where the two least significant bits are set to the [L4_CAP_FPAGE_S](#) and [L4_CAP_FPAGE_W](#) permissions of the capability used. The replaced label is only visible to the thread the IPC gate is bound to upon receive (or to the

selected thread from the thread group the IPC gate is bound to). However, the configured label of an IPC gate can also be queried via [l4_ipc_gate_get_infos\(\)](#) if the capability used has the [L4_FPAGE_C_IPCGATE_SVR](#) permission.

When deleting an IPC gate or when unbinding it from a thread or thread group, the label of IPC already in flight won't be changed. To ensure that no IPC from this IPC gate is received by a thread with an unexpected label, [l4_thread_modify_sender_start\(\)](#) shall be used to change the labels of every pending IPC to that gate. This is also required if the label of an already bound IPC gate is changed. It is not necessary after binding the IPC gate to a thread or thread group for the first time.

When binding a currently bound IPC gate to a new thread or thread group, the same label should be used that was used with the old thread. Otherwise the old and the new thread need to synchronize to avoid IPC messages with unexpected labels.

Include File

```
#include <l4/sys/ipc_gate.h>
```

For the C++ interface refer to the [L4::ipc_gate](#) documentation.

See also

[Object Invocation](#)

Definition in file [ipc_gate.h](#).

16.534 ipc_gate.h

[Go to the documentation of this file.](#)

```
00001 /*
00002  * (c) 2009-2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008
00009 #pragma once
00010
00011 #include <l4/sys/utcb.h>
00012 #include <l4/sys/types.h>
00013 #include <l4/sys/rcv_endpoint.h>
00014
00015 L4_INLINE l4_msgtag_t
00016 l4_ipc_gate_get_infos(l4_cap_idx_t gate, l4_umword_t *label);
00017
00018 L4_INLINE l4_msgtag_t
00019 l4_ipc_gate_get_infos_u(l4_cap_idx_t gate, l4_umword_t *label, l4_utcb_t *utcb);
00020
00021 enum l4_ipc_gate_ops
00022 {
00023     L4_IPC_GATE_BIND_OP      = 0x10,
00024     L4_IPC_GATE_GET_INFO_OP = 0x11,
00025 };
00026
00027 /* IMPLEMENTATION -----*/
00028 #include <l4/sys/ipc.h>
00029
00030 L4_INLINE l4_msgtag_t
00031 l4_ipc_gate_bind_thread_u(l4_cap_idx_t gate,
00032                          l4_cap_idx_t thread, l4_umword_t label,
00033                          l4_utcb_t *utcb)
00034 {
00035     return l4_rcv_ep_bind_thread_u(gate, thread, label, utcb);
00036 }
```

```

00126
00127 L4_INLINE l4_msgtag_t
00128 l4_ipc_gate_bind_snd_destination_u(l4_cap_idx_t gate,
00129                                   l4_cap_idx_t snd_dst, l4_umword_t label,
00130                                   l4_utcb_t *utcb)
00131 {
00132     return l4_rcv_ep_bind_snd_destination_u(gate, snd_dst, label, utcb);
00133 }
00134
00135 L4_INLINE l4_msgtag_t
00136 l4_ipc_gate_get_infos_u(l4_cap_idx_t gate, l4_umword_t *label, l4_utcb_t *utcb)
00137 {
00138     l4_msgtag_t tag;
00139     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00140     m->mr[0] = L4_IPC_GATE_GET_INFO_OP;
00141     tag = l4_ipc_call(gate, utcb, l4_msgtag(L4_PROTO_KOBJECT, 1, 0, 0),
00142                     L4_IPC_NEVER);
00143     if (!l4_msgtag_has_error(tag) && l4_msgtag_label(tag) >= 0)
00144         *label = m->mr[0];
00145     return tag;
00146 }
00147
00148
00149
00150
00151 L4_INLINE l4_msgtag_t
00152 l4_ipc_gate_bind_thread(l4_cap_idx_t gate, l4_cap_idx_t thread,
00153                        l4_umword_t label)
00154 {
00155     return l4_rcv_ep_bind_thread_u(gate, thread, label, l4_utcb());
00156 }
00157
00158 L4_INLINE l4_msgtag_t
00159 l4_ipc_gate_bind_snd_destination(l4_cap_idx_t gate, l4_cap_idx_t snd_dst,
00160                                 l4_umword_t label)
00161 {
00162     return l4_rcv_ep_bind_snd_destination_u(gate, snd_dst, label, l4_utcb());
00163 }
00164
00165 L4_INLINE l4_msgtag_t
00166 l4_ipc_gate_get_infos(l4_cap_idx_t gate, l4_umword_t *label)
00167 {
00168     return l4_ipc_gate_get_infos_u(gate, label, l4_utcb());
00169 }

```

16.535 l4/sys/irq File Reference

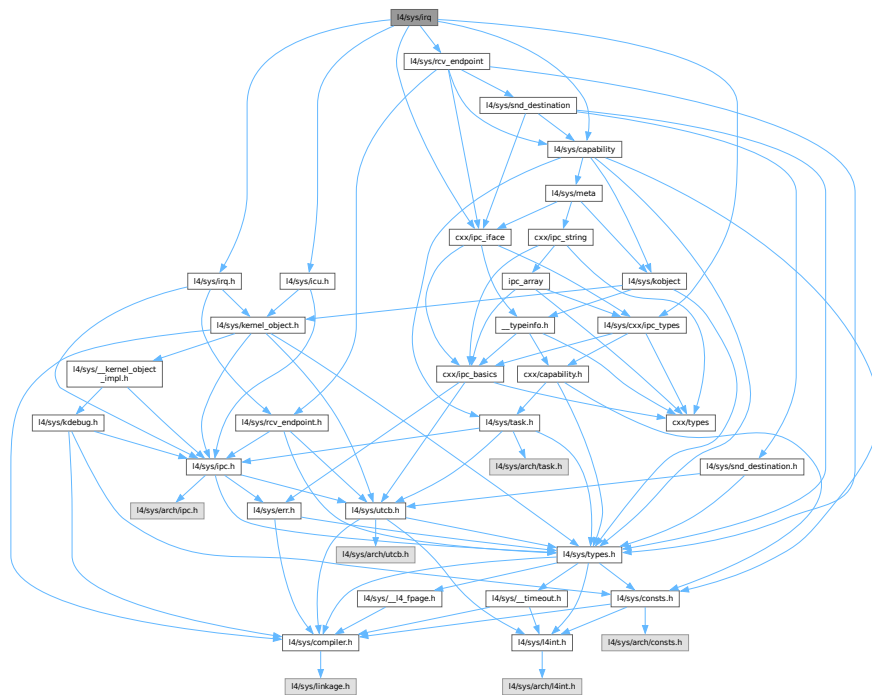
C++ Irq interface.

```

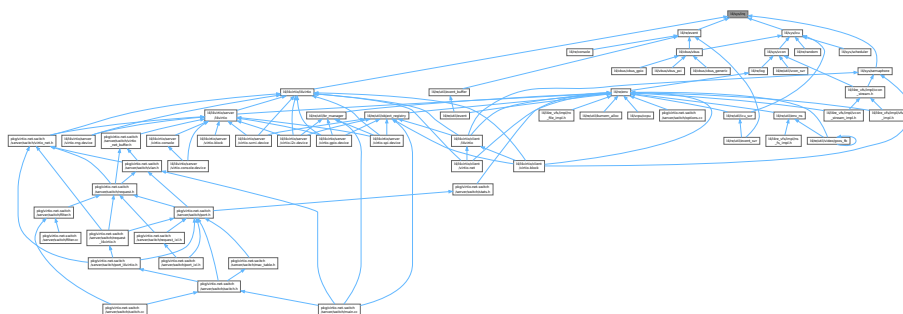
#include <l4/sys/icu.h>
#include <l4/sys/irq.h>
#include <l4/sys/capability>
#include <l4/sys/rcv_endpoint>
#include <l4/sys/cxx/ipc_iface>
#include <l4/sys/cxx/ipc_types>

```

Include dependency graph for irq:



This graph shows which files directly or indirectly include this file:



Data Structures

- class [L4::Irq_eoi](#)
Interface for sending an unmask message to an object.
- struct [L4::Triggerable](#)
Interface that allows an object to be triggered by some source.
- class [L4::Irq](#)
C++ *Irq* interface, see *IRQs* for the C interface.
- class [L4::Lcu](#)
C++ *lcu* interface, see *Interrupt controller* for the C interface.
- class [L4::Lcu::Info](#)
This class encapsulates information about an ICU.

Namespaces

- namespace [L4](#)
L4 low-level kernel interface.

16.535.1 Detailed Description

C++ Irq interface.

Definition in file [irq](#).

16.536 irq

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013
00014 #pragma once
00015
00016 #include <l4/sys/icu.h>
00017 #include <l4/sys/irq.h>
00018 #include <l4/sys/capability>
00019 #include <l4/sys/rcv_endpoint>
00020 #include <l4/sys/cxx/ipc_iface>
00021 #include <l4/sys/cxx/ipc_types>
00022
00023 namespace L4 {
00024
00037 class Irq_eoi : public Kobject_0t<Irq_eoi, L4::PROTO_EMPTY>
00038 {
00039 public:
00064     l4_msgtag_t unmask(unsigned irqnum, l4_umword_t *label = 0,
00065                        l4_timeout_t to = L4_IPC_NEVER,
00066                        l4_utcb_t *utcb = l4_utcb()) noexcept
00067     {
00068         return l4_icu_control_u(cap(), irqnum, L4_ICU_CTL_UNMASK, label, to, utcb);
00069     }
00070 };
00071
00079 struct Triggerable : Kobject_t<Triggerable, Irq_eoi, L4_PROTO_IRQ>
00080 {
00091     l4_msgtag_t trigger(l4_utcb_t *utcb = l4_utcb()) noexcept
00092     { return l4_irq_trigger_u(cap(), utcb); }
00093 };
00094
00120 class Irq : public Kobject_2t<Irq, Triggerable, Rcv_endpoint, L4_PROTO_IRQ_SENDER>
00121 {
00122 public:
00123     using Triggerable::unmask;
00124
00158     l4_msgtag_t bind_vcpu(L4::Cap<Thread> const &thread, l4_umword_t cfg,
00159                          l4_utcb_t *utcb = l4_utcb()) noexcept
00160     { return l4_irq_bind_vcpu_u(cap(), thread.cap(), cfg, utcb); }
00161
00176     l4_msgtag_t detach(l4_utcb_t *utcb = l4_utcb()) noexcept
00177     { return l4_irq_detach_u(cap(), utcb); }
00178
00179
00191     l4_msgtag_t receive(l4_timeout_t timeout = L4_IPC_NEVER,
00192                        l4_utcb_t *utcb = l4_utcb()) noexcept
00193     { return l4_irq_receive_u(cap(), timeout, utcb); }
00194
00204     l4_msgtag_t wait(l4_umword_t *label, l4_timeout_t timeout = L4_IPC_NEVER,
00205                     l4_utcb_t *utcb = l4_utcb()) noexcept
00206     { return unmask(-1, label, timeout, utcb); }
00207
00221     l4_msgtag_t unmask(l4_utcb_t *utcb = l4_utcb()) noexcept

```



```

00222 { return unmask(-1, 0, L4_IPC_NEVER, utcb); }
00223 };
00224
00250 class Icu :
00251     public Kobject_t<Icu, Irq_eoi, L4_PROTO_IRQ,
00252         Type_info::Demand_t<1> >
00253 {
00254 public:
00255     enum Mode
00256     {
00257         F_none           = L4_IRQ_F_NONE,
00258         F_level_high     = L4_IRQ_F_LEVEL_HIGH,
00259         F_level_low      = L4_IRQ_F_LEVEL_LOW,
00260         F_pos_edge       = L4_IRQ_F_POS_EDGE,
00261         F_neg_edge       = L4_IRQ_F_NEG_EDGE,
00262         F_both_edge      = L4_IRQ_F_BOTH_EDGE,
00263         F_mask           = L4_IRQ_F_MASK,
00264
00265         F_set_wakeup     = L4_IRQ_F_SET_WAKEUP,
00266         F_clear_wakeup   = L4_IRQ_F_CLEAR_WAKEUP,
00267     };
00268
00269     enum Flags
00270     {
00271         F_msi = L4_ICU_FLAG_MSI
00272     };
00273
00277     class Info : public l4_icu_info_t
00278     {
00279     public:
00281         bool supports_msi() const noexcept { return features & F_msi; }
00282     };
00283
00310     l4_msgtag_t bind(unsigned irqnum, L4::Cap<Triggerable> irq,
00311         l4_utcb_t *utcb = l4_utcb()) noexcept
00312     { return l4_icu_bind_u(cap(), irqnum, irq.cap(), utcb); }
00313
00314     L4_RPC_NF_OP(
00315         L4_ICU_OP_BIND,
00316         l4_msgtag_t, bind, (l4_umword_t irqnum, Ipc::Cap<Irq> irq)
00317     );
00318
00328     l4_msgtag_t unbind(unsigned irqnum, L4::Cap<Triggerable> irq,
00329         l4_utcb_t *utcb = l4_utcb()) noexcept
00330     { return l4_icu_unbind_u(cap(), irqnum, irq.cap(), utcb); }
00331
00332     L4_RPC_NF_OP(
00333         L4_ICU_OP_UNBIND,
00334         l4_msgtag_t, unbind, (l4_umword_t irqnum, Ipc::Cap<Irq> irq)
00335     );
00336
00345     l4_msgtag_t info(l4_icu_info_t *info, l4_utcb_t *utcb = l4_utcb()) noexcept
00346     { return l4_icu_info_u(cap(), info, utcb); }
00347
00348     struct _Info { l4_umword_t features, nr_irqs, nr_msis; };
00349     L4_RPC_NF_OP(L4_ICU_OP_INFO, l4_msgtag_t, info, (_Info *info));
00350
00363     L4_INLINE_RPC_OP(L4_ICU_OP_MSI_INFO,
00364         l4_msgtag_t, msi_info, (l4_umword_t irqnum, l4_uint64_t source,
00365             l4_icu_msi_info_t *msi_info));
00366
00370     l4_msgtag_t control(unsigned irqnum, unsigned op, l4_umword_t *label,
00371         l4_timeout_t to, l4_utcb_t *utcb = l4_utcb()) noexcept
00372     { return l4_icu_control_u(cap(), irqnum, op, label, to, utcb); }
00373
00393     l4_msgtag_t mask(unsigned irqnum,
00394         l4_umword_t *label = 0,
00395         l4_timeout_t to = L4_IPC_NEVER,
00396         l4_utcb_t *utcb = l4_utcb()) noexcept
00397     { return l4_icu_mask_u(cap(), irqnum, label, to, utcb); }
00398
00399     L4_RPC_NF_OP(
00400         L4_ICU_OP_MASK,
00401         l4_msgtag_t, mask, (l4_umword_t irqnum),
00402         L4::Ipc::Send_only
00403     );
00404
00405
00406     L4_RPC_NF_OP(
00407         L4_ICU_OP_UNMASK,
00408         l4_msgtag_t, unmask, (l4_umword_t irqnum),
00409         L4::Ipc::Send_only
00410     );
00411
00421     l4_msgtag_t set_mode(unsigned irqnum, l4_umword_t mode,
00422         l4_utcb_t *utcb = l4_utcb()) noexcept
00423     { return l4_icu_set_mode_u(cap(), irqnum, mode, utcb); }

```

```

00424
00425 L4_RPC_NF_OP (
00426     L4_ICU_OP_SET_MODE,
00427     l4_msgtag_t, set_mode, (l4_umword_t irqnum, l4_umword_t mode)
00428 );
00429
00430 typedef L4::Typeid::Rpcsys<
00431     bind_t, unbind_t, info_t, msi_info_t, unmask_t, mask_t, set_mode_t
00432 > Rpcsys;
00433 };
00434
00435 }

```

16.537 I4/sys/kdebug.h File Reference

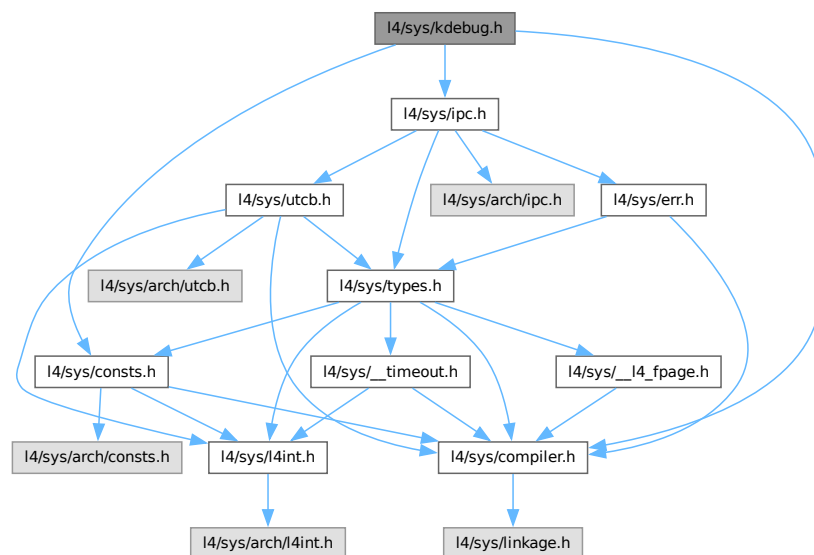
Functionality for invoking the kernel debugger.

```

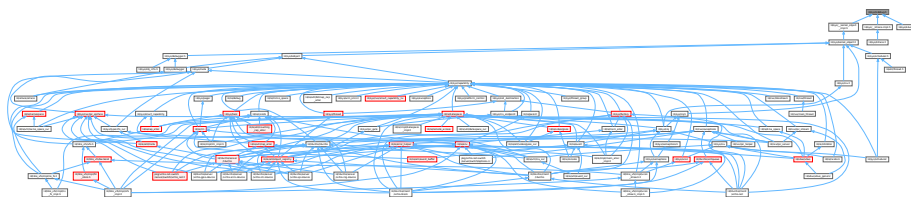
#include <l4/sys/compiler.h>
#include <l4/sys/consts.h>
#include <l4/sys/ipc.h>

```

Include dependency graph for kdebug.h:



This graph shows which files directly or indirectly include this file:



Enumerations

- enum [l4_kdebug_group_t](#)
Opcode groups for operations that can be invoked on the base debugger capability.
- enum [l4_kdebug_ops_t](#)
Op-codes for operations that can be invoked on the base debugger capability.

Functions

- void [l4_kd_enter](#) (char const *text) [L4_NOTHROW](#)
Enter the kernel debugger.
- [l4_msgtag_t](#) [__l4_kdebug_op](#) (unsigned op) [L4_NOTHROW](#)
Invoke a nullary operation on the base debugger capability.
- [l4_msgtag_t](#) [__l4_kdebug_text](#) (unsigned op, char const *text, unsigned len) [L4_NOTHROW](#)
Invoke a text output operation on the base debugger capability.
- [l4_msgtag_t](#) [__l4_kdebug_3_text](#) (unsigned op, char const *text, unsigned len, [l4_umword_t](#) v1, [l4_umword_t](#) v2, [l4_umword_t](#) v3) [L4_NOTHROW](#)
Invoke a text output operation with 3 additional machine word arguments on the base debugger capability.
- [l4_msgtag_t](#) [__l4_kdebug_op_1](#) (unsigned op, [l4_mword_t](#) val) [L4_NOTHROW](#)
Invoke an unary operation on the base debugger capability.
- void [l4_kd_outnstring](#) (char const *text, unsigned len)
Output a fixed-length string via the kernel debugger.
- void [l4_kd_outstring](#) (char const *text)
Output a string via the kernel debugger.
- void [l4_kd_outchar](#) (char c)
Output a single character via the kernel debugger.
- void [l4_kd_outumword](#) ([l4_umword_t](#) number)
Output a hexadecimal unsigned machine word via the kernel debugger.
- void [l4_kd_outhex64](#) ([l4_uint64_t](#) number)
Output a 64-bit unsigned hexadecimal number via the kernel debugger.
- void [l4_kd_outhex32](#) ([l4_uint32_t](#) number)
Output a 32-bit unsigned hexadecimal number via the kernel debugger.
- void [l4_kd_outhex20](#) ([l4_uint32_t](#) number)
Output a 20-bit unsigned hexadecimal number via the kernel debugger.
- void [l4_kd_outhex16](#) ([l4_uint16_t](#) number)
Output a 16-bit unsigned hexadecimal number via the kernel debugger.
- void [l4_kd_outhex12](#) ([l4_uint16_t](#) number)
Output a 12-bit unsigned hexadecimal number via the kernel debugger.
- void [l4_kd_outhex8](#) ([l4_uint8_t](#) number)
Output an 8-bit unsigned hexadecimal number via the kernel debugger.
- void [l4_kd_outdec](#) ([l4_mword_t](#) number)
Output a decimal unsigned machine word via the kernel debugger.

16.537.1 Detailed Description

Functionality for invoking the kernel debugger.

Definition in file [kdebug.h](#).

16.537.2 Enumeration Type Documentation

16.537.2.1 l4_kdebug_ops_t

```
enum l4_kdebug_ops_t
```

Op-codes for operations that can be invoked on the base debugger capability.

See also [__ktrace-impl.h](#) for additional op-codes.

Definition at line 44 of file [kdebug.h](#).

16.537.3 Function Documentation

16.537.3.1 __l4_kdebug_3_text()

```
l4_msgtag_t __l4_kdebug_3_text (
    unsigned op,
    char const * text,
    unsigned len,
    l4_umword_t v1,
    l4_umword_t v2,
    l4_umword_t v3) [inline]
```

Invoke a text output operation with 3 additional machine word arguments on the base debugger capability.

Parameters

| | |
|-------------|---|
| <i>op</i> | Text output operation code from l4_kdebug_ops_t or a value above 0x200 used by the kernel trace buffer implementation (__ktrace-impl.h). |
| <i>text</i> | Output string. |
| <i>len</i> | Length of the output string. The maximum length is limited to L4_UTCB_GENERIC_DATA_SIZE - 5 machine words. Output strings longer than this limit will be cropped. |
| <i>v1</i> | First machine word argument. |
| <i>v2</i> | Second machine word argument. |
| <i>v3</i> | Third machine word argument. |

Return values

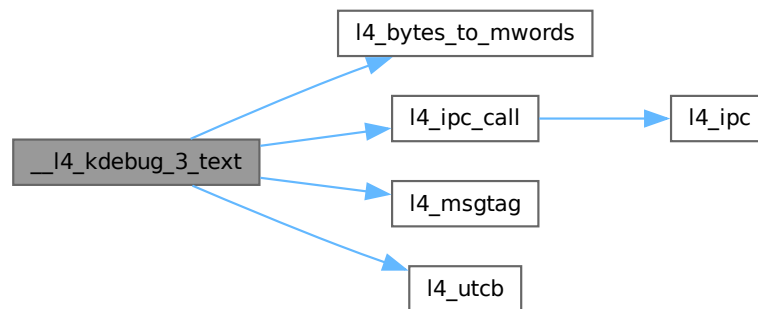
| | |
|----------------|--|
| <i>Message</i> | tag returned from the IPC on the base debugger capability. |
|----------------|--|

Definition at line 139 of file [kdebug.h](#).

References [L4_BASE_DEBUGGER_CAP](#), [l4_bytes_to_mwords\(\)](#), [l4_ipc_call\(\)](#), [L4_IPC_NEVER](#), [l4_msgtag\(\)](#), [L4_NOTHROW](#), [L4_PROTO_DEBUGGER](#), [l4_utcb\(\)](#), and [l4_msg_regs_t::mr](#).

Referenced by [fiasco_tbuf_log_3val\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



16.537.3.2 __l4_kdebug_op()

```
l4_msgtag_t __l4_kdebug_op (
    unsigned op) [inline]
```

Invoke a nullary operation on the base debugger capability.

Parameters

| | |
|-----------|--|
| <i>op</i> | Nullary operation code from l4_kdebug_ops_t or a value above 0x200 used by the kernel trace buffer implementation (__ktrace-impl.h). |
|-----------|--|

Return values

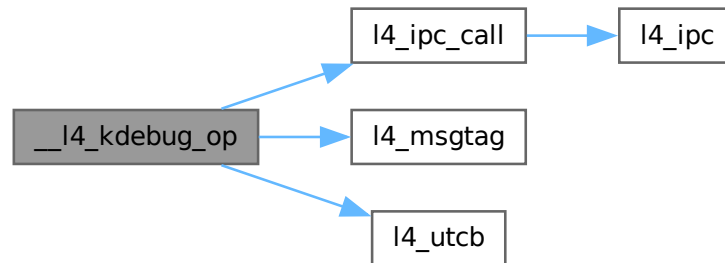
| | |
|----------------|--|
| <i>Message</i> | tag returned from the IPC on the base debugger capability. |
|----------------|--|

Definition at line 68 of file [kdebug.h](#).

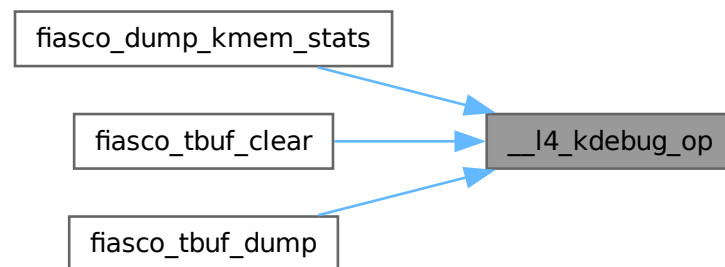
References [L4_BASE_DEBUGGER_CAP](#), [l4_ipc_call\(\)](#), [L4_IPC_NEVER](#), [l4_msgtag\(\)](#), [L4_NOTHROW](#), [L4_PROTO_DEBUGGER](#), [l4_utcb\(\)](#), and [l4_msg_regs_t::mr](#).

Referenced by [fiasco_dump_kmem_stats\(\)](#), [fiasco_tbuf_clear\(\)](#), and [fiasco_tbuf_dump\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



16.537.3.3 __l4_kdebug_op_1()

```

l4_msgtag_t __l4_kdebug_op_1 (
    unsigned op,
    l4_mword_t val) [inline]
  
```

Invoke an unary operation on the base debugger capability.

Parameters

| | |
|------------|--|
| <i>op</i> | Unary operation code from l4_kdebug_ops_t or a value above 0x200 used by the kernel trace buffer implementation (__ktrace-impl.h). |
| <i>val</i> | Machine word argument to the unary operation. |

Return values

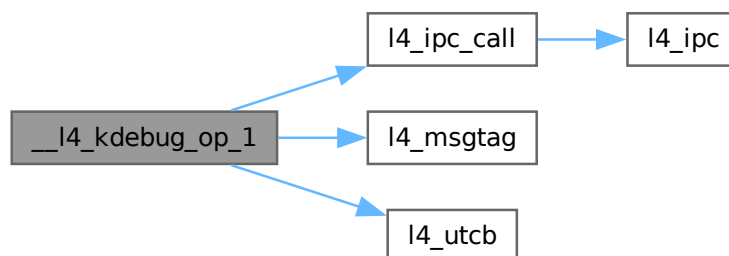
| | |
|----------------|--|
| <i>Message</i> | tag returned from the IPC on the base debugger capability. |
|----------------|--|

Definition at line 176 of file [kdebug.h](#).

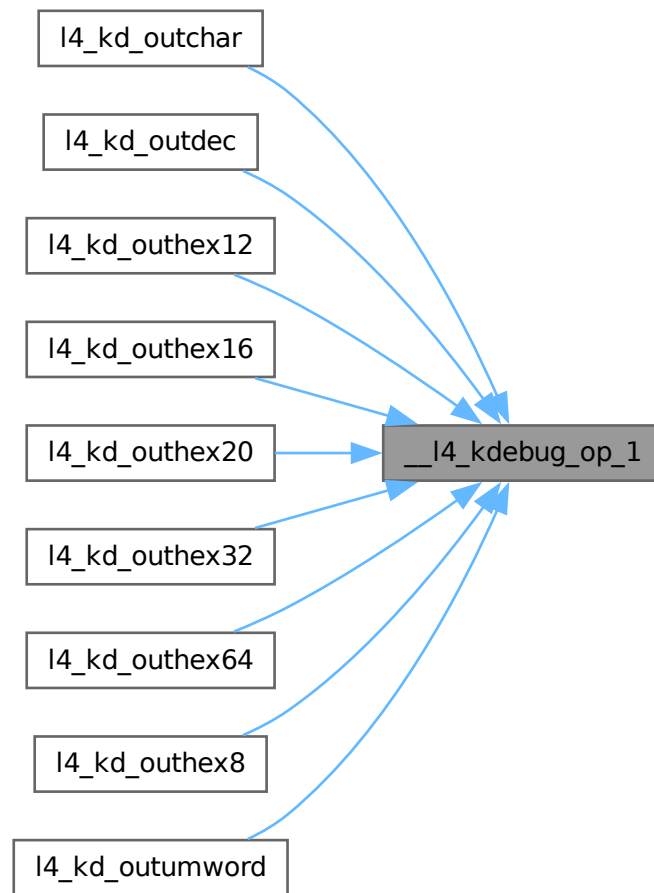
References [L4_BASE_DEBUGGER_CAP](#), [l4_ipc_call\(\)](#), [L4_IPC_NEVER](#), [l4_msgtag\(\)](#), [L4_NOTHROW](#), [L4_PROTO_DEBUGGER](#), [l4_utcb\(\)](#), and [l4_msg_regs_t::mr](#).

Referenced by [l4_kd_outchar\(\)](#), [l4_kd_outdec\(\)](#), [l4_kd_outhex12\(\)](#), [l4_kd_outhex16\(\)](#), [l4_kd_outhex20\(\)](#), [l4_kd_outhex32\(\)](#), [l4_kd_outhex64\(\)](#), [l4_kd_outhex8\(\)](#), and [l4_kd_outumword\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



16.537.3.4 __l4_kdebug_text()

```

l4_msgtag_t __l4_kdebug_text (
    unsigned op,
    char const * text,
    unsigned len) [inline]

```

Invoke a text output operation on the base debugger capability.

Parameters

| | |
|-------------|--|
| <i>op</i> | Text output operation code from <code>l4_kdebug_ops_t</code> or a value above 0x200 used by the kernel trace buffer implementation (<code>__ktrace-impl.h</code>). |
| <i>text</i> | Output string. |

| | |
|------------|---|
| <i>len</i> | Length of the output string. The maximum length is limited to L4_UTCB_GENERIC_DATA_SIZE - 2 machine words. Output strings longer than this limit will be cropped. |
|------------|---|

Return values

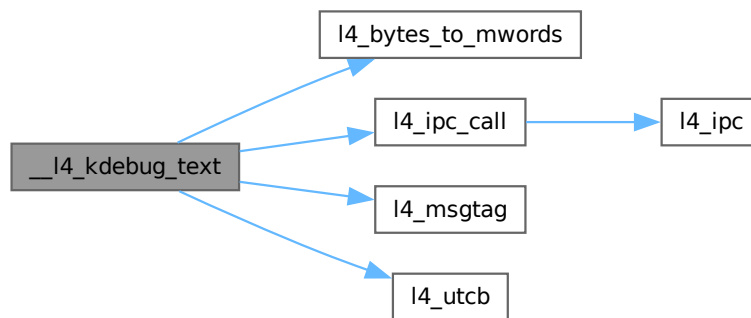
| | |
|----------------|--|
| <i>Message</i> | tag returned from the IPC on the base debugger capability. |
|----------------|--|

Definition at line 98 of file [kdebug.h](#).

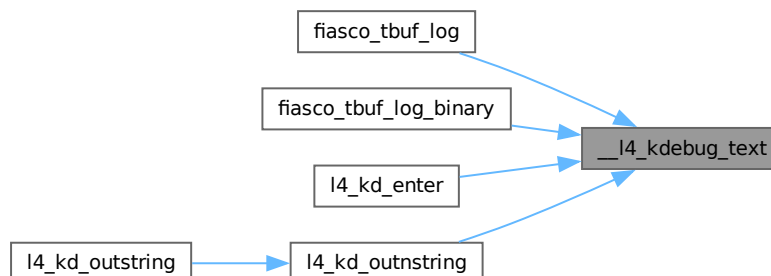
References [L4_BASE_DEBUGGER_CAP](#), [l4_bytes_to_mwords\(\)](#), [l4_ipc_call\(\)](#), [L4_IPC_NEVER](#), [l4_msgtag\(\)](#), [L4_NOTHROW](#), [L4_PROTO_DEBUGGER](#), [l4_utcb\(\)](#), and [l4_msg_regs_t::mr](#).

Referenced by [fiasco_tbuf_log\(\)](#), [fiasco_tbuf_log_binary\(\)](#), [l4_kd_enter\(\)](#), and [l4_kd_outstring\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



16.537.3.5 l4_kd_enter()

```
void l4_kd_enter (
    char const * text) [inline]
```

Enter the kernel debugger.

Parameters

| | |
|-------------|---|
| <i>text</i> | Optional message displayed by the kernel debugger when entered. |
|-------------|---|

Enter the kernel debugger, if configured. An optional message can be passed to the kernel debugger which is printed upon the entering of the debugger.

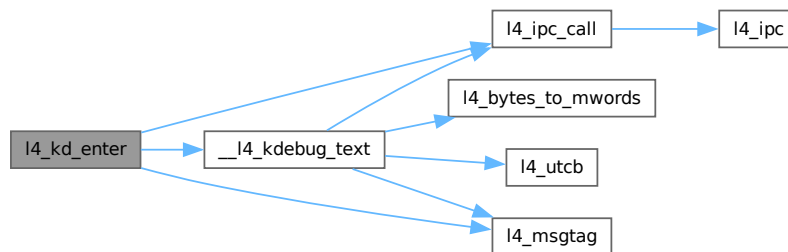
Examples

[examples/sys/singlestep/main.c](#).

Definition at line 204 of file [kdebug.h](#).

References [__l4_kdebug_text\(\)](#), [L4_BASE_DEBUGGER_CAP](#), [L4_INLINE](#), [l4_ipc_call\(\)](#), [L4_IPC_NEVER](#), [l4_msgtag\(\)](#), [L4_NOTHROW](#), and [L4_PROTO_DEBUGGER](#).

Here is the call graph for this function:



16.537.3.6 l4_kd_outchar()

```
void l4_kd_outchar (
    char c) [inline]
```

Output a single character via the kernel debugger.

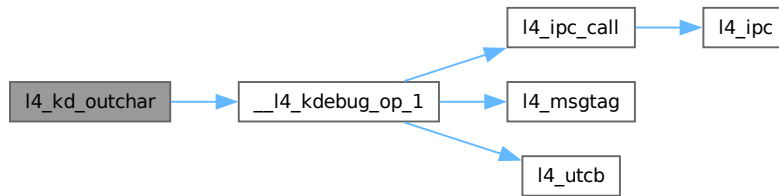
Parameters

| | |
|----------|-------------------|
| <i>c</i> | Output character. |
|----------|-------------------|

Definition at line 245 of file [kdebug.h](#).

References [__l4_kdebug_op_1\(\)](#), and [L4_INLINE](#).

Here is the call graph for this function:



16.537.3.7 l4_kd_outdec()

```
void l4_kd_outdec (
    l4_mword_t number) [inline]
```

Output a decimal unsigned machine word via the kernel debugger.

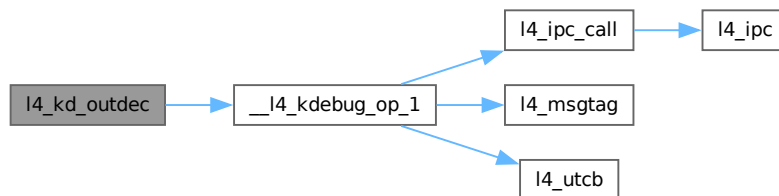
Parameters

| | |
|---------------|----------------------|
| <i>number</i> | Output machine word. |
|---------------|----------------------|

Definition at line 334 of file [kdebug.h](#).

References [__l4_kdebug_op_1\(\)](#), and [L4_INLINE](#).

Here is the call graph for this function:



16.537.3.8 l4_kd_outhex12()

```
void l4_kd_outhex12 (
    l4_uint16_t number) [inline]
```

Output a 12-bit unsigned hexadecimal number via the kernel debugger.

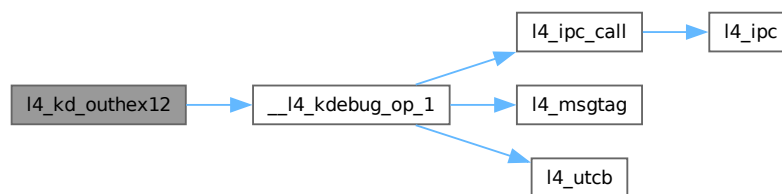
Parameters

| | |
|---------------|--|
| <i>number</i> | Output 12-bit number. Only the 12 LSB bits are used. |
|---------------|--|

Definition at line 314 of file [kdebug.h](#).

References [__l4_kdebug_op_1\(\)](#), and [L4_INLINE](#).

Here is the call graph for this function:



16.537.3.9 l4_kd_outhex16()

```
void l4_kd_outhex16 (
    l4_uint16_t number) [inline]
```

Output a 16-bit unsigned hexadecimal number via the kernel debugger.

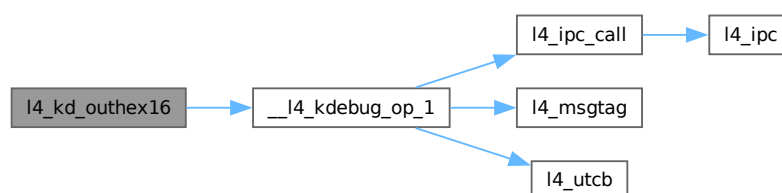
Parameters

| | |
|---------------|-----------------------|
| <i>number</i> | Output 16-bit number. |
|---------------|-----------------------|

Definition at line 304 of file [kdebug.h](#).

References [__l4_kdebug_op_1\(\)](#), and [L4_INLINE](#).

Here is the call graph for this function:



16.537.3.10 l4_kd_outhex20()

```
void l4_kd_outhex20 (  
    l4_uint32_t number) [inline]
```

Output a 20-bit unsigned hexadecimal number via the kernel debugger.

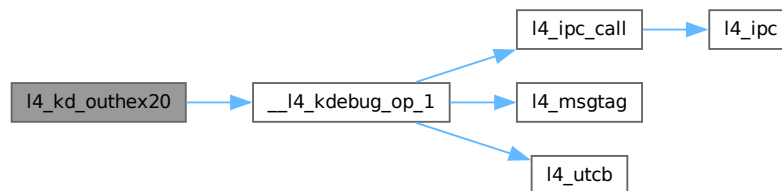
Parameters

| | |
|---------------|--|
| <i>number</i> | Output 20-bit number. Only the 20 LSB bits are used. |
|---------------|--|

Definition at line 294 of file [kdebug.h](#).

References [__l4_kdebug_op_1\(\)](#), and [L4_INLINE](#).

Here is the call graph for this function:



16.537.3.11 l4_kd_outhex32()

```
void l4_kd_outhex32 (  
    l4_uint32_t number) [inline]
```

Output a 32-bit unsigned hexadecimal number via the kernel debugger.

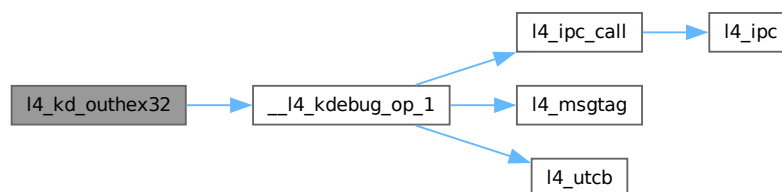
Parameters

| | |
|---------------|-----------------------|
| <i>number</i> | Output 32-bit number. |
|---------------|-----------------------|

Definition at line 284 of file [kdebug.h](#).

References [__l4_kdebug_op_1\(\)](#), and [L4_INLINE](#).

Here is the call graph for this function:



16.537.3.12 l4_kd_outhex64()

```
void l4_kd_outhex64 (  
    l4_uint64_t number) [inline]
```

Output a 64-bit unsigned hexadecimal number via the kernel debugger.

Parameters

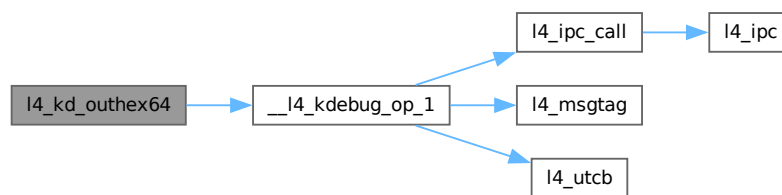
| | |
|---------------|-----------------------|
| <i>number</i> | Output 64-bit number. |
|---------------|-----------------------|

The two 32-bit halves are printed non-atomically.

Definition at line 273 of file [kdebug.h](#).

References [__l4_kdebug_op_1\(\)](#), and [L4_INLINE](#).

Here is the call graph for this function:



16.537.3.13 l4_kd_outhex8()

```
void l4_kd_outhex8 (  
    l4_uint8_t number) [inline]
```

Output an 8-bit unsigned hexadecimal number via the kernel debugger.

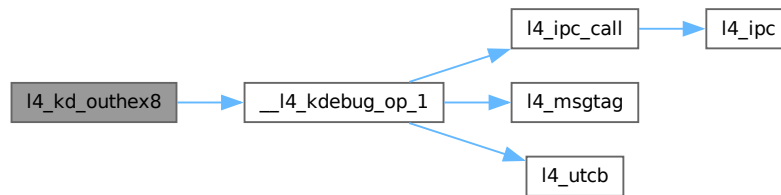
Parameters

| | |
|---------------|----------------------|
| <i>number</i> | Output 8-bit number. |
|---------------|----------------------|

Definition at line 324 of file [kdebug.h](#).

References [__l4_kdebug_op_1\(\)](#), and [L4_INLINE](#).

Here is the call graph for this function:



16.537.3.14 l4_kd_outnstring()

```
void l4_kd_outnstring (
    char const * text,
    unsigned len) [inline]
```

Output a fixed-length string via the kernel debugger.

Parameters

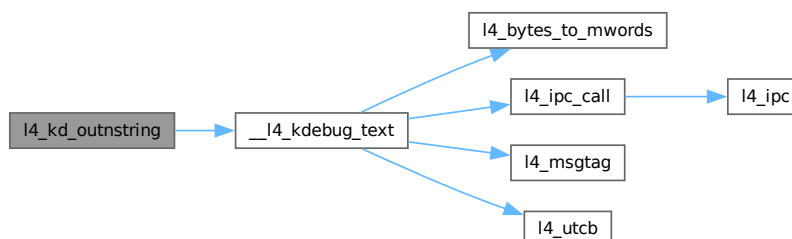
| | |
|-------------|---|
| <i>text</i> | Beginning of the output string. |
| <i>len</i> | Length of the output string. The maximum length is limited to L4_UTCB_GENERIC_DATA_SIZE - 2 machine words. Output strings longer than this limit will be cropped. |

Definition at line 226 of file [kdebug.h](#).

References [__l4_kdebug_text\(\)](#), and [L4_INLINE](#).

Referenced by [l4_kd_outstring\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



16.537.3.15 l4_kd_outstring()

```
void l4_kd_outstring (
    char const * text) [inline]
```

Output a string via the kernel debugger.

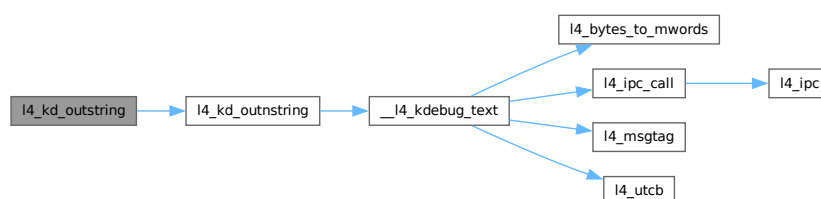
Parameters

| | |
|-------------|---|
| <i>text</i> | Beginning of the output string. The maximum length of the output string is limited to L4_UTCB_GENERIC_DATA_SIZE - 2 machine words. Output strings longer than this limit will be cropped. |
|-------------|---|

Definition at line 237 of file [kdebug.h](#).

References [L4_INLINE](#), and [l4_kd_outnstring\(\)](#).

Here is the call graph for this function:



16.537.3.16 l4_kd_outumword()

```
void l4_kd_outumword (
    l4_umword_t number) [inline]
```

Output a hexadecimal unsigned machine word via the kernel debugger.

Parameters

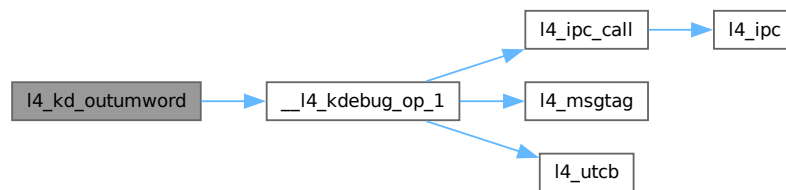
| | |
|---------------|----------------------|
| <i>number</i> | Output machine word. |
|---------------|----------------------|

If the machine word is 64 bits long, it is printed non-atomically as two 32-bit numbers.

Definition at line 258 of file [kdebug.h](#).

References [__l4_kdebug_op_1\(\)](#), and [L4_INLINE](#).

Here is the call graph for this function:



16.538 kdebug.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00003  *     economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00008 #pragma once
00009
00010
00011 #ifndef __KDEBUG_H__
00012 #define __KDEBUG_H__
00013
00014 #include <l4/sys/compiler.h>
00015 #include <l4/sys/consts.h>
00016 #include <l4/sys/ipc.h>
00017
00018 L4_INLINE void
00019 l4_kd_enter(char const *text) L4_NOTHROW;
00020
00021 enum l4_kdebug_group_t
00022 {
00023     L4_KDEBUG_GROUP_JDB      = 0x000,
00024     L4_KDEBUG_GROUP_KOBJ     = 0x100, // see __kernel_object_impl.h
00025     L4_KDEBUG_GROUP_TRACE    = 0x200, // see __ktrace-impl.h
00026     L4_KDEBUG_GROUP_COV      = 0x400,
00027     L4_KDEBUG_GROUP_DUMP     = 0x500, // see kdump.h
00028 };
00029
00030 enum l4_kdebug_ops_t
00031 {
00032     L4_KDEBUG_ENTER          = L4_KDEBUG_GROUP_JDB + 0,
00033     L4_KDEBUG_OUTCHAR        = L4_KDEBUG_GROUP_JDB + 1,
00034     L4_KDEBUG_OUTNSTRING     = L4_KDEBUG_GROUP_JDB + 2,
00035     L4_KDEBUG_OUTHEX32       = L4_KDEBUG_GROUP_JDB + 3,
00036     L4_KDEBUG_OUTHEX20       = L4_KDEBUG_GROUP_JDB + 4,
00037     L4_KDEBUG_OUTHEX16       = L4_KDEBUG_GROUP_JDB + 5,
00038     L4_KDEBUG_OUTHEX12       = L4_KDEBUG_GROUP_JDB + 6,
00039     L4_KDEBUG_OUTHEX8        = L4_KDEBUG_GROUP_JDB + 7,
00040 };

```

```

00054 L4_KDEBUG_OUTDEC      = L4_KDEBUG_GROUP_JDB + 8,
00055 };
00056
00057
00067 L4_INLINE l4_msgtag_t
00068 __l4_kdebug_op(unsigned op) L4_NOTHROW
00069 {
00070     l4_msgtag_t res;
00071     l4_utcb_t *u = l4_utcb();
00072     l4_msg_regs_t *mr = l4_utcb_mr_u(u);
00073     l4_umword_t mr0 = mr->mr[0];
00074
00075     mr->mr[0] = op;
00076     res = l4_ipc_call(L4_BASE_DEBUGGER_CAP, u,
00077                     l4_msgtag(L4_PROTO_DEBUGGER, 1, 0, 0),
00078                     L4_IPC_NEVER);
00079     mr->mr[0] = mr0;
00080     return res;
00081 }
00082
00097 L4_INLINE l4_msgtag_t
00098 __l4_kdebug_text(unsigned op, char const *text, unsigned len) L4_NOTHROW
00099 {
00100     l4_msg_regs_t store;
00101     l4_msgtag_t res;
00102     l4_utcb_t *u = l4_utcb();
00103     l4_msg_regs_t *mr = l4_utcb_mr_u(u);
00104
00105     if (len > (sizeof(store) - (2 * sizeof(l4_umword_t))))
00106         len = sizeof(store) - (2 * sizeof(l4_umword_t));
00107
00108     __builtin_memcpy(&store, mr, sizeof(store));
00109     mr->mr[0] = op;
00110     mr->mr[1] = len;
00111     __builtin_memcpy(&mr->mr[2], text, len);
00112     res = l4_ipc_call(L4_BASE_DEBUGGER_CAP, u,
00113                     l4_msgtag(L4_PROTO_DEBUGGER,
00114                             l4_bytes_to_mwords(len) + 2, 0, 0),
00115                     L4_IPC_NEVER);
00116     __builtin_memcpy(mr, &store, sizeof(*mr));
00117     return res;
00118 }
00119
00138 L4_INLINE l4_msgtag_t
00139 __l4_kdebug_3_text(unsigned op, char const *text, unsigned len,
00140                   l4_umword_t v1, l4_umword_t v2, l4_umword_t v3) L4_NOTHROW
00141 {
00142     l4_msg_regs_t store;
00143     l4_msgtag_t res;
00144     l4_utcb_t *u = l4_utcb();
00145     l4_msg_regs_t *mr = l4_utcb_mr_u(u);
00146
00147     if (len > (sizeof(store) - (5 * sizeof(l4_umword_t))))
00148         len = sizeof(store) - (5 * sizeof(l4_umword_t));
00149
00150     __builtin_memcpy(&store, mr, sizeof(store));
00151     mr->mr[0] = op;
00152     mr->mr[1] = v1;
00153     mr->mr[2] = v2;
00154     mr->mr[3] = v3;
00155     mr->mr[4] = len;
00156     __builtin_memcpy(&mr->mr[5], text, len);
00157     res = l4_ipc_call(L4_BASE_DEBUGGER_CAP, u,
00158                     l4_msgtag(L4_PROTO_DEBUGGER,
00159                             l4_bytes_to_mwords(len) + 5, 0, 0),
00160                     L4_IPC_NEVER);
00161     __builtin_memcpy(mr, &store, sizeof(*mr));
00162     return res;
00163 }
00164
00175 L4_INLINE l4_msgtag_t
00176 __l4_kdebug_op_1(unsigned op, l4_mword_t val) L4_NOTHROW
00177 {
00178     l4_umword_t m[2];
00179     l4_msgtag_t res;
00180     l4_utcb_t *u = l4_utcb();
00181     l4_msg_regs_t *mr = l4_utcb_mr_u(u);
00182
00183     m[0] = mr->mr[0];
00184     m[1] = mr->mr[1];
00185     mr->mr[0] = op;
00186     mr->mr[1] = val;
00187     res = l4_ipc_call(L4_BASE_DEBUGGER_CAP, u,
00188                     l4_msgtag(L4_PROTO_DEBUGGER, 2, 0, 0),
00189                     L4_IPC_NEVER);
00190     mr->mr[0] = m[0];
00191     mr->mr[1] = m[1];

```

```

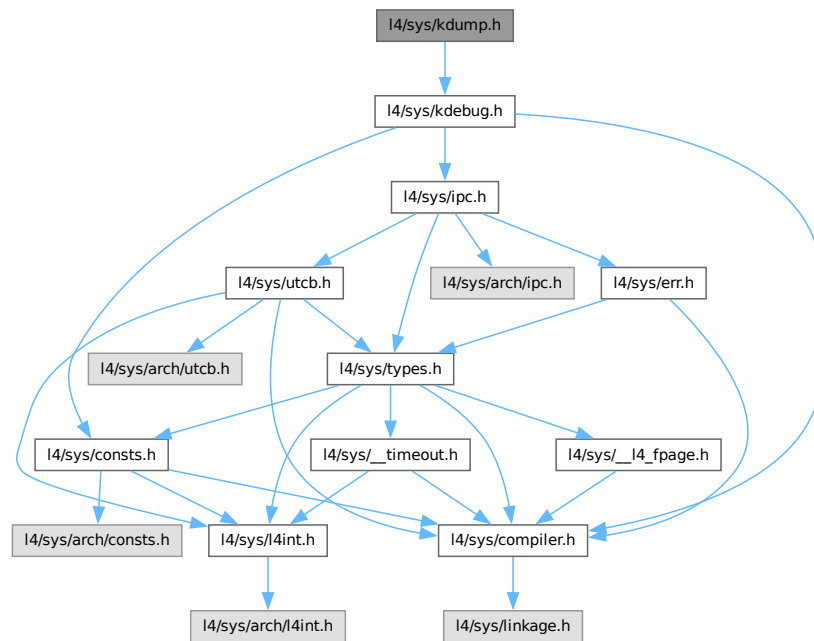
00192     return res;
00193 }
00194
00204 L4_INLINE void l4_kd_enter(char const *text) L4_NOTHROW
00205 {
00206     /* special case, enter without any text and use of the UTCB */
00207     if (!text)
00208     {
00209         l4_ipc_call(L4_BASE_DEBUGGER_CAP, 0,
00210                    l4_msgtag(L4_PROTO_DEBUGGER, 0, 0, 0),
00211                    L4_IPC_NEVER);
00212         return;
00213     }
00214
00215     __l4_kdebug_text(L4_KDEBUG_ENTER, text, __builtin_strlen(text));
00216 }
00217
00226 L4_INLINE void l4_kd_outnstring(char const *text, unsigned len)
00227 { __l4_kdebug_text(L4_KDEBUG_OUTNSTRING, text, len); }
00228
00237 L4_INLINE void l4_kd_outstring(char const *text)
00238 { l4_kd_outnstring(text, __builtin_strlen(text)); }
00239
00245 L4_INLINE void l4_kd_outchar(char c)
00246 {
00247     __l4_kdebug_op_1(L4_KDEBUG_OUTCHAR, c);
00248 }
00249
00258 L4_INLINE void l4_kd_outumword(l4_umword_t number)
00259 {
00260     if (sizeof(l4_umword_t) == sizeof(l4_uint64_t))
00261         __l4_kdebug_op_1(L4_KDEBUG_OUTHEX32, (l4_uint64_t)number >> 32);
00262
00263     __l4_kdebug_op_1(L4_KDEBUG_OUTHEX32, number);
00264 }
00265
00273 L4_INLINE void l4_kd_outhex64(l4_uint64_t number)
00274 {
00275     __l4_kdebug_op_1(L4_KDEBUG_OUTHEX32, number >> 32);
00276     __l4_kdebug_op_1(L4_KDEBUG_OUTHEX32, number);
00277 }
00278
00284 L4_INLINE void l4_kd_outhex32(l4_uint32_t number)
00285 {
00286     __l4_kdebug_op_1(L4_KDEBUG_OUTHEX32, number);
00287 }
00288
00294 L4_INLINE void l4_kd_outhex20(l4_uint32_t number)
00295 {
00296     __l4_kdebug_op_1(L4_KDEBUG_OUTHEX20, number);
00297 }
00298
00304 L4_INLINE void l4_kd_outhex16(l4_uint16_t number)
00305 {
00306     __l4_kdebug_op_1(L4_KDEBUG_OUTHEX16, number);
00307 }
00308
00314 L4_INLINE void l4_kd_outhex12(l4_uint16_t number)
00315 {
00316     __l4_kdebug_op_1(L4_KDEBUG_OUTHEX12, number);
00317 }
00318
00324 L4_INLINE void l4_kd_outhex8(l4_uint8_t number)
00325 {
00326     __l4_kdebug_op_1(L4_KDEBUG_OUTHEX8, number);
00327 }
00328
00334 L4_INLINE void l4_kd_outdec(l4_mword_t number)
00335 {
00336     __l4_kdebug_op_1(L4_KDEBUG_OUTDEC, number);
00337 }
00338
00339 #endif //__KDEBUG_H__

```

16.539 l4/sys/kdump.h File Reference

Functionality for dumping kernel information.

```
#include <l4/sys/kdebug.h>
Include dependency graph for kdump.h:
```



Functions

- long [fiasco_dump_kmem_stats](#) (void)
Dump kernel memory statistics on console.

16.539.1 Detailed Description

Functionality for dumping kernel information.

Definition in file [kdump.h](#).

16.539.2 Function Documentation

16.539.2.1 fiasco_dump_kmem_stats()

```
long fiasco_dump_kmem_stats (
    void ) [inline]
```

Dump kernel memory statistics on console.

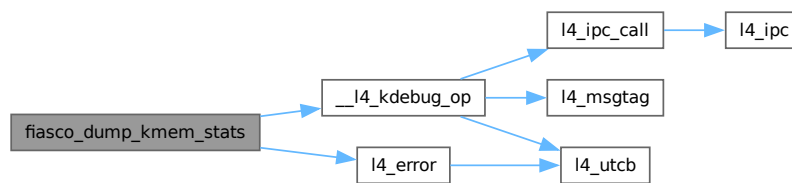
Return values

| | |
|------------|----------------------------|
| 0 | Success. |
| -L4_ENOSYS | Not implemented by kernel. |

Definition at line 38 of file [kdump.h](#).

References [__l4_kdebug_op\(\)](#), and [l4_error\(\)](#).

Here is the call graph for this function:



16.540 kdump.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (C) 2024-2025 Kernkonzept GmbH.
00003  * Author(s): Jan Klötzke <jan.kloetzke@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00008 #pragma once
00009
00014
00025
00026 #include <l4/sys/kdebug.h>
00027
00034 L4_INLINE long
00035 fiasco_dump_kmem_stats(void);
00036
00037 L4_INLINE long
00038 fiasco_dump_kmem_stats(void)
00039 {
00040     enum { DUMP_KMEM_STATS = L4_KDEBUG_GROUP_DUMP + 0x00 };
00041     return l4_error(__l4_kdebug_op(DUMP_KMEM_STATS));
00042 }

```

16.541 l4/sys/kernel_object.h File Reference

Kernel object system calls.

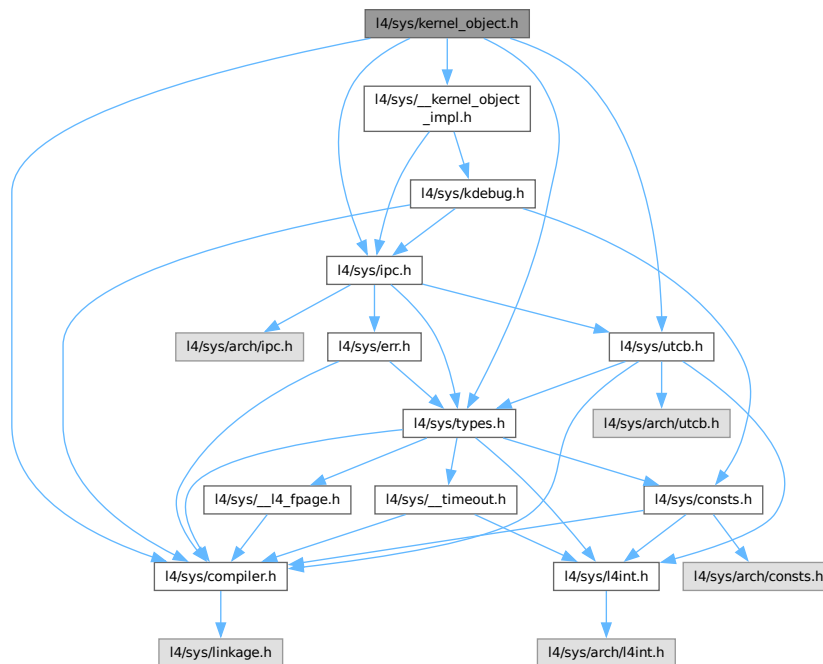
```

#include <l4/sys/types.h>
#include <l4/sys/compiler.h>
#include <l4/sys/utcb.h>
#include <l4/sys/__kernel_object_impl.h>

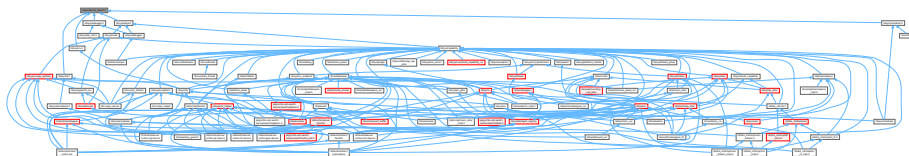
```

```
#include <l4/sys/ipc.h>
```

Include dependency graph for kernel_object.h:



This graph shows which files directly or indirectly include this file:



16.541.1 Detailed Description

Kernel object system calls.

Definition in file [kernel_object.h](#).

16.542 kernel_object.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00008  *      Björn Döbel <doebel@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)

```

```

00012  */
00013  #ifndef __L4SYS__KERNEL_OBJECT_H__
00014  #define __L4SYS__KERNEL_OBJECT_H__
00015
00016  #include <l4/sys/types.h>
00017  #include <l4/sys/compiler.h>
00018  #include <l4/sys/utcb.h>
00019
00027
00038  L4_INLINE l4_msgtag_t
00039  l4_invoke_debugger(l4_cap_idx_t obj, l4_msgtag_t tag, l4_utcb_t *utcb) L4_NOTHROW;
00040
00041
00042  /*****
00043   * Implementation
00044   *****/
00045
00046  #include <l4/sys/__kernel_object_impl.h>
00047  #include <l4/sys/ipc.h>
00048
00049  enum l4_kobject_op {
00050      L4_KOBJECT_OP_DEC_REFCNT = 0,
00051      L4_KOBJECT_OP_REGISTER_IRQ,
00052  };
00053
00054  L4_INLINE l4_msgtag_t
00055  l4_kobject_dec_refcnt_u(l4_cap_idx_t obj, l4_mword_t diff, l4_utcb_t *u) L4_NOTHROW;
00056
00057  L4_INLINE l4_msgtag_t
00058  l4_kobject_dec_refcnt(l4_cap_idx_t obj, l4_mword_t diff) L4_NOTHROW;
00059
00060  L4_INLINE l4_msgtag_t
00061  l4_kobject_dec_refcnt_u(l4_cap_idx_t obj, l4_mword_t diff, l4_utcb_t *u) L4_NOTHROW
00062  {
00063      l4_msg_regs_t *m = l4_utcb_mr_u(u);
00064      m->mr[0] = L4_KOBJECT_OP_DEC_REFCNT;
00065      m->mr[1] = diff;
00066      return l4_ipc_call(obj, u, l4_msgtag(L4_PROTO_KOBJECT, 2, 0, 0), L4_IPC_NEVER);
00067  }
00068
00069  L4_INLINE l4_msgtag_t
00070  l4_kobject_dec_refcnt(l4_cap_idx_t obj, l4_mword_t diff) L4_NOTHROW
00071  {
00072      return l4_kobject_dec_refcnt_u(obj, diff, l4_utcb());
00073  }
00074
00075  #endif /* ! __L4SYS__KERNEL_OBJECT_H__ */

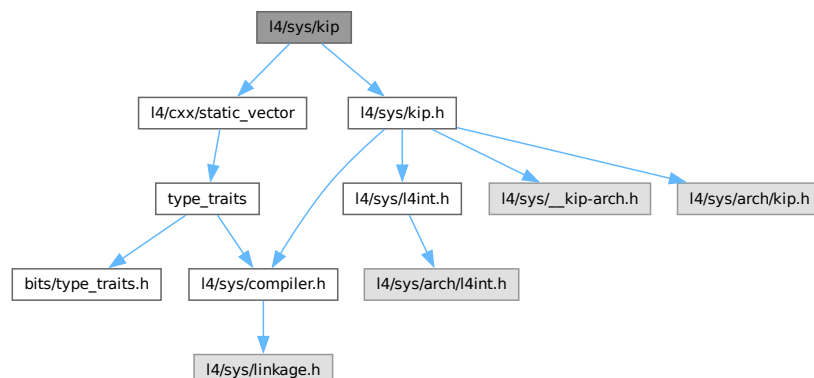
```

16.543 I4/sys/kip File Reference

```
#include <l4/cxx/static_vector>
```

```
#include <l4/sys/kip.h>
```

Include dependency graph for kip:



Data Structures

- class [L4::Kip::Mem_desc](#)

Memory descriptors stored in the kernel interface page.

Namespaces

- namespace [L4](#)

[L4](#) low-level kernel interface.

16.543.1 Detailed Description

L4::Kip class, memory descriptors.

Author

Alexander Warg alexander.warg@os.inf.tu-dresden.de

Definition in file [kip](#).

16.544 kip

[Go to the documentation of this file.](#)

```

00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00010 /*
00011  * (c) 2008-2009 Author(s)
00012  *      economic rights: Technische Universität Dresden (Germany)
00013  *
00014  * License: see LICENSE.spdx (in this directory or the directories above)
00015  */
00016 #pragma once
00017
00018 #include <l4/cxx/static_vector>
00019 #include <l4/sys/kip.h>
00020
00021 /* C++ version of memory descriptors */
00022
00031
00032 namespace L4
00033 {
00034     namespace Kip
00035     {
00042         class Mem_desc
00043         {
00044         public:
00048             enum Mem_type
00049             {
00050                 Undefined      = 0x0,
00051                 Conventional   = 0x1,
00052                 Reserved       = 0x2,
00053                 Dedicated      = 0x3,
00054                 Shared         = 0x4,
00055
00056                 Info           = 0xd,
00057                 Bootloader     = 0xe,
00058                 Arch           = 0xf
00059             };
00060
00064             enum Info_sub_type
00065             {
00066                 Info_acpi_rsdp = 0,
00067
00068                 Reserved_kernel = 0,
00069                 Reserved_heap   = 1,
00070                 Reserved_mmio   = 2,

```



```

00071     };
00072
00076     enum Arch_sub_type_common
00077     {
00078         Arch_acpi_tables = 3,
00079         Arch_acpi_nvs    = 4,
00080     };
00081
00082     private:
00083         unsigned long _l, _h;
00084
00085     public:
00093         static Mem_desc *first(l4_kernel_info_t *kip) noexcept
00094         {
00095             return reinterpret_cast<Mem_desc *>(reinterpret_cast<char *>(kip) + kip->mem_descs);
00096         }
00097
00098         static Mem_desc const *first(l4_kernel_info_t const *kip) noexcept
00099         {
00100             char const *addr = reinterpret_cast<char const *>(kip) + kip->mem_descs;
00101             return reinterpret_cast<Mem_desc const *>(addr);
00102         }
00103
00111         static unsigned long count(l4_kernel_info_t const *kip) noexcept
00112         {
00113             return kip->mem_descs_num;
00114         }
00115
00122         static void count(l4_kernel_info_t *kip, unsigned count) noexcept
00123         {
00124             kip->mem_descs_num = count;
00125         }
00126
00132         static inline cxx::static_vector<Mem_desc const> all(l4_kernel_info_t const *kip)
00133         {
00134             return cxx::static_vector<Mem_desc const>(Mem_desc::first(kip),
00135                                                         Mem_desc::count(kip));
00136         }
00137
00143         static inline cxx::static_vector<Mem_desc> all(l4_kernel_info_t *kip)
00144         {
00145             return cxx::static_vector<Mem_desc>(Mem_desc::first(kip),
00146                                                  Mem_desc::count(kip));
00147         }
00148
00162         Mem_desc(unsigned long start, unsigned long end,
00163                  Mem_type t, unsigned char st = 0, bool virt = false,
00164                  bool eager = false) noexcept
00165         : _l((start & ~0x3ffUL) | (t & 0x0f) | ((st << 4) & 0x0f0)
00166            | (virt ? 0x0200 : 0x0) | (eager ? 0x100 : 0x0)), _h(end | 0x3ffUL)
00167         {}
00168
00174         unsigned long start() const noexcept { return _l & ~0x3ffUL; }
00175
00181         unsigned long end() const noexcept { return _h | 0x3ffUL; }
00182
00188         unsigned long size() const noexcept { return end() + 1 - start(); }
00189
00195         Mem_type type() const noexcept
00196         {
00197             return static_cast<Mem_type>(_l & 0x0f);
00198         }
00199
00205         unsigned char sub_type() const noexcept { return (_l >> 4) & 0x0f; }
00206
00213         unsigned is_virtual() const noexcept { return _l & 0x200; }
00214
00219         unsigned eager_map() const noexcept { return _l & 0x100; }
00220
00233         void set(unsigned long start, unsigned long end,
00234                  Mem_type t, unsigned char st = 0, bool virt = false,
00235                  bool eager = false) noexcept
00236         {
00237             _l = (start & ~0x3ffUL) | (t & 0x0f) | ((st << 4) & 0x0f0)
00238                 | (virt ? 0x0200 : 0x0) | (eager ? 0x0100 : 0x0);
00239
00240             _h = end | 0x3ffUL;
00241         }
00242     };
00243 };
00244 };
00245 };

```

16.545 kobject

```

00001 // vi:set ft=c++: -*- Mode: C++ -*-
00002 /* \file
00003  * Kobject C++ interface.
00004  */
00005 /*
00006  * Copyright (C) 2015-2017, 2019, 2021, 2023-2024 Kernkonzept GmbH.
00007  * Author(s): Alexander Warg <alexander.warg@kernkonzept.com>
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011 #pragma once
00012
00013 #include "kernel_object.h"
00014 #include "types.h"
00015 #include "__typeinfo.h"
00016
00017 namespace L4 {
00018
00036 class L4_EXPORT Kobject
00037 {
00038 private:
00039     Kobject();
00040     Kobject(Kobject const &);
00041     Kobject &operator = (Kobject const &);
00042
00043     template<typename T> friend struct Kobject_typeid;
00044
00045 protected:
00046     typedef Typeid::Iface<L4_PROTO_META, Kobject> __Iface;
00047     typedef Typeid::Iface_list<__Iface> __Iface_list;
00048
00055     struct __Kobject_typeid
00056     {
00057         typedef Type_info::Demand_t<> Demand;
00058         static Type_info const _m;
00059     };
00060
00069     l4_cap_idx_t cap() const noexcept { return _c(); }
00070
00071 private:
00072
00077     l4_cap_idx_t _c() const noexcept
00078     { return reinterpret_cast<l4_cap_idx_t>(this) & L4_CAP_MASK; }
00079
00080 public:
00100     l4_msgtag_t dec_refcnt(l4_mword_t diff, l4_utcb_t *utcb = l4_utcb())
00101     { return l4_kobject_dec_refcnt_u(cap(), diff, utcb); }
00102 };
00103
00104 template<typename Derived, l4_proto_t PROTO = L4::PROTO_ANY,
00105         typename S_DEMAND = Type_info::Demand_t<> >
00106 struct Kobject_0t : Kobject_t<Derived, L4::Kobject, PROTO, S_DEMAND> {};
00107
00108 }
00109

```

16.546 l4/sys/ktrace.h File Reference

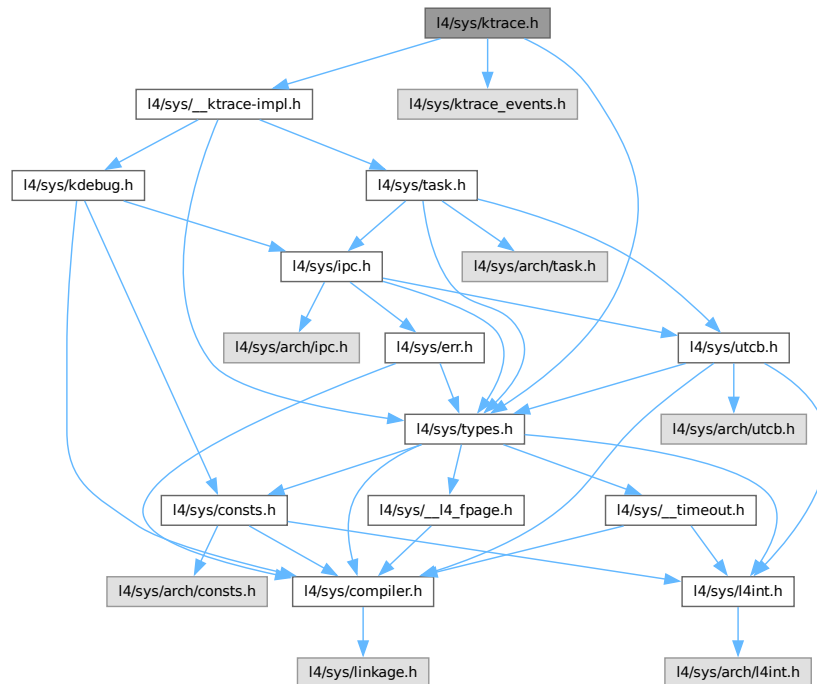
L4 kernel event tracing.

```

#include <l4/sys/types.h>
#include <l4/sys/ktrace_events.h>
#include <l4/sys/__ktrace-impl.h>

```

Include dependency graph for ktrace.h:



Functions

- [l4_msgtag_t fiasco_tbuf_validate](#) (void)
Validate the kernel base debugger capability.
- [l4_umword_t fiasco_tbuf_log](#) (const char *text)
Create new trace-buffer entry with describing <text>.
- [l4_umword_t fiasco_tbuf_log_3val](#) (const char *text, [l4_umword_t](#) v1, [l4_umword_t](#) v2, [l4_umword_t](#) v3)
Create new trace-buffer entry with describing <text> and three additional values.
- [l4_umword_t fiasco_tbuf_log_binary](#) (const unsigned char *data)
Create new trace-buffer entry with binary data.
- void [fiasco_tbuf_clear](#) (void)
Clear trace-buffer.
- void [fiasco_tbuf_dump](#) (void)
Dump trace-buffer to kernel console.
- [l4_msgtag_t fiasco_tbuf_map_status](#) ([l4_fpage_t](#) *ku_mem)
Map kernel trace-buffer status page.
- [l4_msgtag_t fiasco_tbuf_map_slots](#) ([l4_fpage_t](#) *ku_mem)
Map kernel trace-buffer slots.

16.546.1 Detailed Description

[L4](#) kernel event tracing.

Definition in file [ktrace.h](#).

16.547 ktrace.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *          Björn Döbel <doebel@os.inf.tu-dresden.de>,
00009  *          Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010  *          2015 Adam Lackorzynski <adam@l4re.org>
00011  *          economic rights: Technische Universität Dresden (Germany)
00012  *
00013  * License: see LICENSE.spdx (in this directory or the directories above)
00014  */
00015 /*****
00016 #ifndef __L4_KTRACE_H__
00017 #define __L4_KTRACE_H__
00018
00019 #include <l4/sys/types.h>
00020 #include <l4/sys/ktrace_events.h>
00021
00033
00041 L4_INLINE l4_msgtag_t
00042 fiasco_tbuf_validate(void);
00043
00051 L4_INLINE l4_umword_t
00052 fiasco_tbuf_log(const char *text);
00053
00065 L4_INLINE l4_umword_t
00066 fiasco_tbuf_log_3val(const char *text, l4_umword_t v1, l4_umword_t v2, l4_umword_t v3);
00067
00075 L4_INLINE l4_umword_t
00076 fiasco_tbuf_log_binary(const unsigned char *data);
00077
00082 L4_INLINE void
00083 fiasco_tbuf_clear(void);
00084
00089 L4_INLINE void
00090 fiasco_tbuf_dump(void);
00091
00101 L4_INLINE l4_msgtag_t
00102 fiasco_tbuf_map_status(l4_fpage_t *ku_mem);
00103
00113 L4_INLINE l4_msgtag_t
00114 fiasco_tbuf_map_slots(l4_fpage_t *ku_mem);
00115
00116 #include <l4/sys/__ktrace-impl.h>
00117
00118 #endif

```

16.548 amd64/l4/sys/arch/l4int.h File Reference

Fixed sized integer types, AMD64 version.

Macros

- `#define L4_MWORD_BITS 64`
Size of machine words in bits.

Typedefs

- typedef unsigned long `l4_size_t`
Unsigned size type.
- typedef signed long `l4_ssize_t`
Signed size type.

16.548.1 Detailed Description

Fixed sized integer types, AMD64 version.

Definition in file [l4int.h](#).

16.549 l4int.h

[Go to the documentation of this file.](#)

```

00001 /*****/
00007 /*
00008  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014 #pragma once
00015
00020
00021 #define L4_MWORD_BITS      64
00022
00023 typedef unsigned long      l4_size_t;
00024 typedef signed long       l4_ssize_t;
00026

```

16.550 arm/l4/sys/arch/l4int.h File Reference

Fixed sized integer types, arm version.

Macros

- `#define L4_MWORD_BITS 32`
Size of machine words in bits.

Typedefs

- `typedef unsigned int l4_size_t`
Unsigned size type.
- `typedef signed int l4_ssize_t`
Signed size type.

16.550.1 Detailed Description

Fixed sized integer types, arm version.

Definition in file [l4int.h](#).

16.551 I4int.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013 #pragma once
00014
00019
00020 #define L4_MWORD_BITS          32
00021
00022 typedef unsigned int           l4_size_t;
00023 typedef signed int            l4_ssize_t;
00025

```

16.552 arm64/l4/sys/arch/l4int.h File Reference

Fixed sized integer types, arm version.

Macros

- `#define L4_MWORD_BITS 64`
Size of machine words in bits.

Typedefs

- `typedef unsigned long l4_size_t`
Unsigned size type.
- `typedef signed long l4_ssize_t`
Signed size type.

16.552.1 Detailed Description

Fixed sized integer types, arm version.

Definition in file [l4int.h](#).

16.553 I4int.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013 #pragma once
00014
00019
00020 #define L4_MWORD_BITS          64
00021
00022 typedef unsigned long          l4_size_t;
00023 typedef signed long            l4_ssize_t;
00025

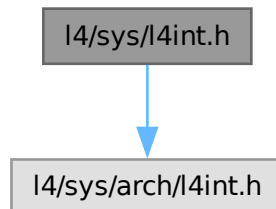
```

16.554 l4/sys/l4int.h File Reference

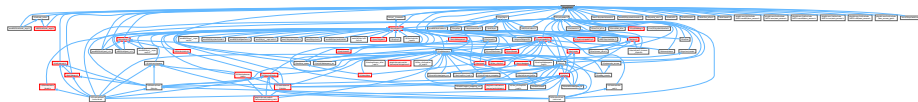
Fixed sized integer types, generic version.

```
#include <l4/sys/arch/l4int.h>
```

Include dependency graph for l4int.h:



This graph shows which files directly or indirectly include this file:



Typedefs

- typedef signed char **l4_int8_t**
Signed 8bit value.
- typedef unsigned char **l4_uint8_t**
Unsigned 8bit value.
- typedef signed short int **l4_int16_t**
Signed 16bit value.
- typedef unsigned short int **l4_uint16_t**
Unsigned 16bit value.
- typedef signed int **l4_int32_t**
Signed 32bit value.
- typedef unsigned int **l4_uint32_t**
Unsigned 32bit value.
- typedef signed long long **l4_int64_t**
Signed 64bit value.
- typedef unsigned long long **l4_uint64_t**
Unsigned 64bit value.
- typedef unsigned long **l4_addr_t**
Address type.
- typedef signed long **l4_mword_t**
Signed machine word.
- typedef unsigned long **l4_umword_t**

Unsigned machine word.

- typedef [l4_uint64_t](#) [l4_cpu_time_t](#)

CPU clock type.

- typedef [l4_uint64_t](#) [l4_kernel_clock_t](#)

Kernel clock type.

16.554.1 Detailed Description

Fixed sized integer types, generic version.

Definition in file [l4int.h](#).

16.555 l4int.h

[Go to the documentation of this file.](#)

```

00001
00007
00013 /*
00014  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00015  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00016  *      economic rights: Technische Universität Dresden (Germany)
00017  *
00018  * License: see LICENSE.spdx (in this directory or the directories above)
00019  */
00020 #ifndef __L4_SYS_L4INT_H__
00021 #define __L4_SYS_L4INT_H__
00022
00023 /* fixed sized data types */
00024 typedef signed char      l4_int8_t;
00025 typedef unsigned char    l4_uint8_t;
00026 typedef signed short int l4_int16_t;
00027 typedef unsigned short int l4_uint16_t;
00028 typedef signed int       l4_int32_t;
00029 typedef unsigned int     l4_uint32_t;
00030 typedef signed long long l4_int64_t;
00031 typedef unsigned long long l4_uint64_t;
00032
00033 /* some common data types */
00034 typedef unsigned long    l4_addr_t;
00035
00036
00037 typedef signed long      l4_mword_t;
00040 typedef unsigned long    l4_umword_t;
00047 typedef l4_uint64_t l4_cpu_time_t;
00048
00053 typedef l4_uint64_t l4_kernel_clock_t;
00054
00055 #include <l4/sys/arch/l4int.h>
00056
00057 #endif /* !__L4_SYS_L4INT_H__ */

```

16.556 riscv/l4/sys/arch/l4int.h File Reference

Fixed sized integer types, RISC-V version.

Macros

- #define [L4_MWORD_BITS](#) [__riscv_xlen](#)

Size of machine words in bits.

Typedefs

- typedef unsigned int **l4_size_t**
Unsigned size type.
- typedef signed int **l4_ssize_t**
Signed size type.

16.556.1 Detailed Description

Fixed sized integer types, RISC-V version.

Definition in file [l4int.h](#).

16.557 l4int.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * Copyright (C) 2021, 2024 Kernkonzept GmbH.
00008  * Author(s): Georg Kotheimer <georg.kotheimer@kernkonzept.com>
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #pragma once
00013
00018
00019 #define L4_MWORD_BITS          __riscv_xlen
00020
00021 #if __riscv_xlen == 64
00022 typedef unsigned long          l4_size_t;
00023 typedef signed long            l4_ssize_t;
00024 #else
00025 typedef unsigned int           l4_size_t;
00026 typedef signed int             l4_ssize_t;
00027 #endif

```

16.558 x86/l4/sys/arch/l4int.h File Reference

Fixed sized integer types, x86 version.

Macros

- #define **L4_MWORD_BITS** 32
Size of machine words in bits.

Typedefs

- typedef unsigned int **l4_size_t**
Unsigned size type.
- typedef signed int **l4_ssize_t**
Signed size type.

16.558.1 Detailed Description

Fixed sized integer types, x86 version.

Definition in file [l4int.h](#).

16.559 l4int.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013 #pragma once
00014
00019
00020 #define L4_MWORD_BITS      32
00021
00022 typedef unsigned int      l4_size_t;
00023 typedef signed int       l4_ssize_t;
00025

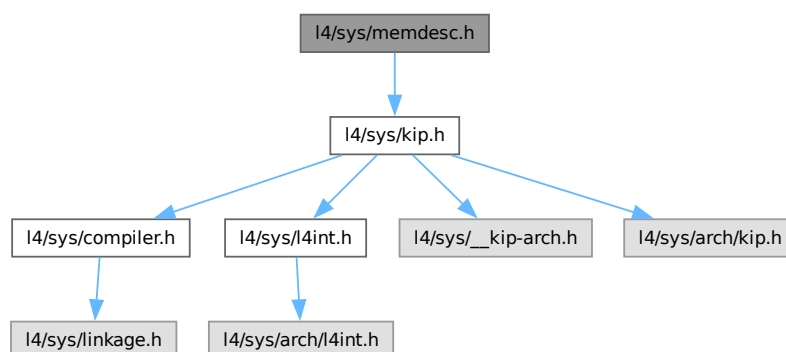
```

16.560 l4/sys/memdesc.h File Reference

Memory description functions.

```
#include <l4/sys/kip.h>
```

Include dependency graph for memdesc.h:



Data Structures

- struct [l4_kernel_info_mem_desc_t](#)
Memory descriptor data structure.

Typedefs

- typedef struct l4_kernel_info_mem_desc_t [l4_kernel_info_mem_desc_t](#)
Memory descriptor data structure.

Enumerations

- enum [l4_mem_type_t](#) {
 [l4_mem_type_undefined](#) = 0x0 , [l4_mem_type_conventional](#) = 0x1 , [l4_mem_type_reserved](#) = 0x2 ,
 [l4_mem_type_dedicated](#) = 0x3 ,
 [l4_mem_type_shared](#) = 0x4 , [l4_mem_type_info](#) = 0xd , [l4_mem_type_bootloader](#) = 0xe , [l4_mem_type_archspecific](#)
 = 0xf }
Type of a memory descriptor.
- enum [l4_mem_info_sub_type_t](#) { [l4_mem_info_acpi_rsdp](#) = 0 , [l4_mem_reserved_kernel](#) = 0 ,
 [l4_mem_reserved_heap](#) = 1 , [l4_mem_reserved_mmio](#) = 2 }
Memory sub types for l4_mem_type_info descriptors.
- enum [l4_mem_archspecific_sub_type_common_t](#) { [l4_mem_archspecific_acpi_tables](#) = 3 , [l4_mem_archspecific_acpi_nvs](#)
 = 4 }
Memory sub types for l4_mem_type_archspecific descriptors.

Functions

- [l4_kernel_info_mem_desc_t](#) * [l4_kernel_info_get_mem_descs](#) ([l4_kernel_info_t](#) *kip) [L4_NOTHROW](#)
Get pointer to memory descriptors from KIP.
- unsigned [l4_kernel_info_get_num_mem_descs](#) ([l4_kernel_info_t](#) *kip) [L4_NOTHROW](#)
Get number of memory descriptors in KIP.
- void [l4_kernel_info_set_mem_desc](#) ([l4_kernel_info_mem_desc_t](#) *md, [l4_addr_t](#) start, [l4_addr_t](#) end, unsigned type, unsigned virt, unsigned sub_type) [L4_NOTHROW](#)
Populate a memory descriptor.
- [l4_umword_t](#) [l4_kernel_info_get_mem_desc_start](#) ([l4_kernel_info_mem_desc_t](#) *md) [L4_NOTHROW](#)
Get start address of the region described by the memory descriptor.
- [l4_umword_t](#) [l4_kernel_info_get_mem_desc_end](#) ([l4_kernel_info_mem_desc_t](#) *md) [L4_NOTHROW](#)
Get end address of the region described by the memory descriptor.
- [l4_umword_t](#) [l4_kernel_info_get_mem_desc_type](#) ([l4_kernel_info_mem_desc_t](#) *md) [L4_NOTHROW](#)
Get type of the memory region.
- [l4_umword_t](#) [l4_kernel_info_get_mem_desc_subtype](#) ([l4_kernel_info_mem_desc_t](#) *md) [L4_NOTHROW](#)
Get sub-type of memory region.
- [l4_umword_t](#) [l4_kernel_info_get_mem_desc_is_virtual](#) ([l4_kernel_info_mem_desc_t](#) *md) [L4_NOTHROW](#)
Get virtual flag of the memory descriptor.

16.560.1 Detailed Description

Memory description functions.

Definition in file [memdesc.h](#).

16.561 memdesc.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2007-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013 #ifndef __L4SYS__MEMDESC_H__
00014 #define __L4SYS__MEMDESC_H__
00015
00016 #include <l4/sys/kip.h>
00017
00028
00033 enum l4_mem_type_t
00034 {
00035     l4_mem_type_undefined    = 0x0,
00036     l4_mem_type_conventional = 0x1,
00037     l4_mem_type_reserved     = 0x2,
00038     l4_mem_type_dedicated    = 0x3,
00039     l4_mem_type_shared       = 0x4,
00040
00041     l4_mem_type_info         = 0xd,
00042     l4_mem_type_bootloader   = 0xe,
00043     l4_mem_type_archspecific = 0xf,
00044 };
00045
00050 enum l4_mem_info_sub_type_t
00051 {
00052     l4_mem_info_acpi_rsdp = 0,
00053
00054     l4_mem_reserved_kernel = 0,
00055     l4_mem_reserved_heap   = 1,
00056     l4_mem_reserved_mmio   = 2,
00057 };
00058
00063 enum l4_mem_archspecific_sub_type_common_t
00064 {
00065     l4_mem_archspecific_acpi_tables = 3,
00066     l4_mem_archspecific_acpi_nvs    = 4,
00067 };
00068
00069
00077 typedef struct l4_kernel_info_mem_desc_t
00078 {
00080     l4_umword_t l;
00082     l4_umword_t h;
00083 } l4_kernel_info_mem_desc_t;
00084
00085
00090 L4_INLINE
00091 l4_kernel_info_mem_desc_t *
00092 l4_kernel_info_get_mem_descs(l4_kernel_info_t *kip) L4_NOTHROW;
00093
00100 L4_INLINE
00101 unsigned
00102 l4_kernel_info_get_num_mem_descs(l4_kernel_info_t *kip) L4_NOTHROW;
00103
00115 L4_INLINE
00116 void
00117 l4_kernel_info_set_mem_desc(l4_kernel_info_mem_desc_t *md,
00118                             l4_addr_t start,
00119                             l4_addr_t end,
00120                             unsigned type,
00121                             unsigned virt,
00122                             unsigned sub_type) L4_NOTHROW;
00123
00130 L4_INLINE
00131 l4_umword_t
00132 l4_kernel_info_get_mem_desc_start(l4_kernel_info_mem_desc_t *md) L4_NOTHROW;
00133
00140 L4_INLINE
00141 l4_umword_t
00142 l4_kernel_info_get_mem_desc_end(l4_kernel_info_mem_desc_t *md) L4_NOTHROW;
00143
00150 L4_INLINE
00151 l4_umword_t
00152 l4_kernel_info_get_mem_desc_type(l4_kernel_info_mem_desc_t *md) L4_NOTHROW;
00153
00163 L4_INLINE
00164 l4_umword_t
00165 l4_kernel_info_get_mem_desc_subtype(l4_kernel_info_mem_desc_t *md) L4_NOTHROW;

```

```

00166
00173 L4_INLINE
00174 l4_umword_t
00175 l4_kernel_info_get_mem_desc_is_virtual(l4_kernel_info_mem_desc_t *md) L4_NOTHROW;
00176
00177 /*****
00178  * Implementations
00179  *****/
00180
00181 L4_INLINE
00182 l4_kernel_info_mem_desc_t *
00183 l4_kernel_info_get_mem_descs(l4_kernel_info_t *kip) L4_NOTHROW
00184 {
00185     return (l4_kernel_info_mem_desc_t *) ((l4_addr_t)kip + kip->mem_descs);
00186 }
00187
00188 L4_INLINE
00189 unsigned
00190 l4_kernel_info_get_num_mem_descs(l4_kernel_info_t *kip) L4_NOTHROW
00191 {
00192     return kip->mem_descs_num;
00193 }
00194
00195 L4_INLINE
00196 void
00197 l4_kernel_info_set_mem_desc(l4_kernel_info_mem_desc_t *md,
00198                             l4_addr_t start,
00199                             l4_addr_t end,
00200                             unsigned type,
00201                             unsigned virt,
00202                             unsigned sub_type) L4_NOTHROW
00203 {
00204     md->l = (start & ~0x3ffUL) | (type & 0x0f) | ((sub_type < 4) & 0x0f0)
00205             | (virt ? 0x200 : 0x0);
00206     md->h = end;
00207 }
00208
00209
00210 L4_INLINE
00211 l4_umword_t
00212 l4_kernel_info_get_mem_desc_start(l4_kernel_info_mem_desc_t *md) L4_NOTHROW
00213 {
00214     return md->l & ~0x3ffUL;
00215 }
00216
00217 L4_INLINE
00218 l4_umword_t
00219 l4_kernel_info_get_mem_desc_end(l4_kernel_info_mem_desc_t *md) L4_NOTHROW
00220 {
00221     return md->h | 0x3ffUL;
00222 }
00223
00224 L4_INLINE
00225 l4_umword_t
00226 l4_kernel_info_get_mem_desc_type(l4_kernel_info_mem_desc_t *md) L4_NOTHROW
00227 {
00228     return md->l & 0xf;
00229 }
00230
00231 L4_INLINE
00232 l4_umword_t
00233 l4_kernel_info_get_mem_desc_subtype(l4_kernel_info_mem_desc_t *md) L4_NOTHROW
00234 {
00235     return (md->l & 0xf0) >> 4;
00236 }
00237
00238 L4_INLINE
00239 l4_umword_t
00240 l4_kernel_info_get_mem_desc_is_virtual(l4_kernel_info_mem_desc_t *md) L4_NOTHROW
00241 {
00242     return md->l & 0x200;
00243 }
00244
00245 #endif /* ! __L4SYS__MEMDESC_H__ */

```

16.562 meta

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002
00003 #pragma once
00004
00005 #include <l4/sys/meta>
00006 #include <l4/sys/typeinfo_svr>

```

```

00007
00008 namespace L4Re { namespace Util {
00009 using L4::Util::handle_meta_request;
00010 }}

```

16.563 I4/sys/meta File Reference

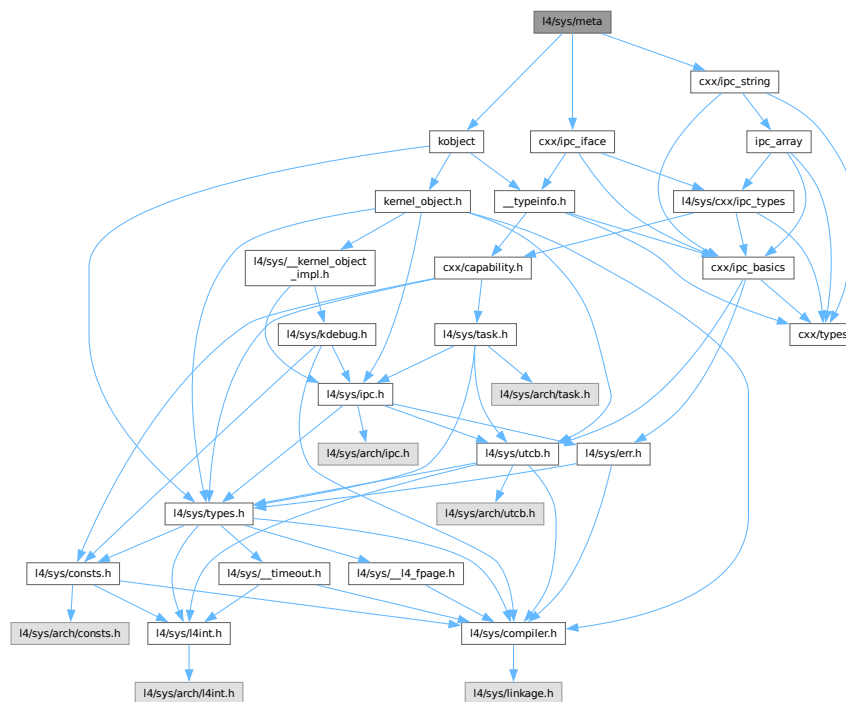
Meta interface for getting dynamic type information about objects behind capabilities.

```

#include "kobject"
#include "cxx/ipc_iface"
#include "cxx/ipc_string"

```

Include dependency graph for meta:



This graph shows which files directly or indirectly include this file:



Data Structures

- class [L4::Meta](#)

Meta interface that shall be implemented by each [L4Re](#) object and gives access to the dynamic type information for [L4Re](#) objects.

Namespaces

- namespace [L4](#)
L4 low-level kernel interface.

16.563.1 Detailed Description

Meta interface for getting dynamic type information about objects behind capabilities.

Definition in file [meta](#).

16.564 meta

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00007 /*
00008  * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013
00014 #pragma once
00015
00016 #include "kobject"
00017 #include "cxx/ipc_iface"
00018 #include "cxx/ipc_string"
00019
00020 namespace L4 {
00021
00026 class Meta : public Kobject_t<Meta, Kobject, L4_PROTO_META>
00027 {
00028 public:
00035     L4_INLINE_RPC(l4_msgtag_t, num_interfaces, ());
00036
00050     L4_INLINE_RPC(l4_msgtag_t, interface, (l4_umword_t idx, long *proto,
00051                                           L4::Ipc::String<char> *name));
00052
00064     L4_INLINE_RPC(l4_msgtag_t, supports, (l4_mword_t protocol));
00065
00066     typedef L4::Typeid::Rpc<num_interfaces_t, interface_t, supports_t> Rpc;
00067 };
00068
00069 }
```

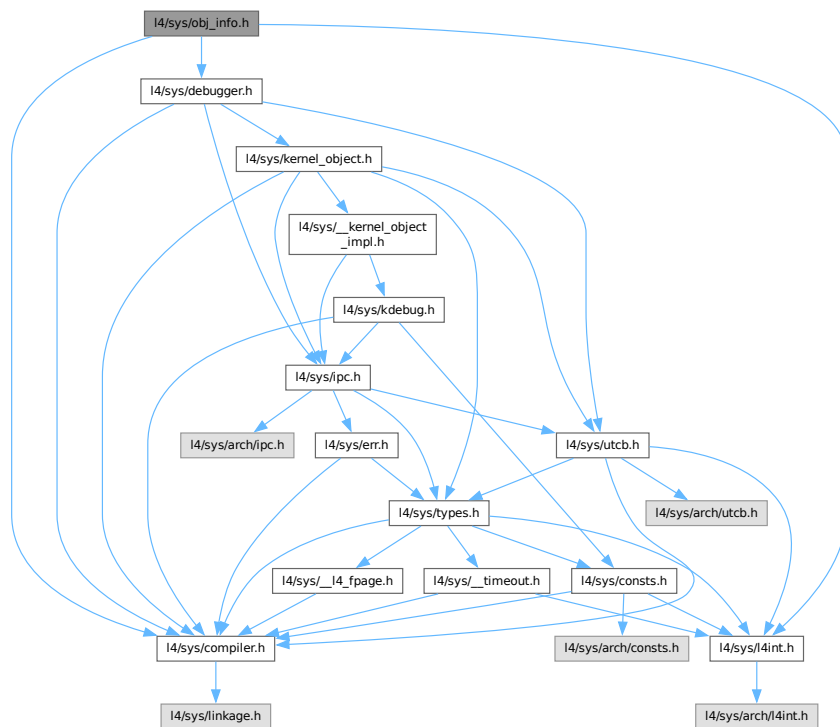
16.565 l4/sys/obj_info.h File Reference

Debugger related functions.

```

#include <l4/sys/compiler.h>
#include <l4/sys/debugger.h>
```

```
#include <l4/sys/l4int.h>
Include dependency graph for obj_info.h:
```



Functions

- [l4_msgtag_t l4_debugger_query_obj_infos](#) ([l4_cap_idx_t](#) cap, [l4_addr_t](#) ku_mem_addr, [l4_size_t](#) ku_mem_size, [l4_umword_t](#) skip, [l4_umword_t](#) *result_cnt, [l4_umword_t](#) *result_all) [L4_NOTHROW](#)

Retrieve information from the kernel about all objects in the mapping database and write data to the passed KU memory.

16.565.1 Detailed Description

Debugger related functions.

Attention

This API is subject to change!

Definition in file [obj_info.h](#).

16.565.2 Function Documentation

16.565.2.1 l4_debugger_query_obj_infos()

```
l4_msgtag_t l4_debugger_query_obj_infos (
    l4_cap_idx_t cap,
    l4_addr_t ku_mem_addr,
    l4_size_t ku_mem_size,
    l4_umword_t skip,
    l4_umword_t * result_cnt,
    l4_umword_t * result_all) [inline]
```

Retrieve information from the kernel about all objects in the mapping database and write data to the passed KU memory.

Parameters

| | | |
|-----|--------------------|---|
| | <i>cap</i> | Capability of the debugger object. |
| | <i>ku_mem_addr</i> | Address of the KU memory for writing the information. |
| | <i>ku_mem_size</i> | Size of the KU memory for writing the information. |
| | <i>skip</i> | Number of objects to skip. |
| out | <i>result_cnt</i> | Number of objects in the mapping database. |
| out | <i>result_all</i> | Number of objects written to the KU memory. |

Note

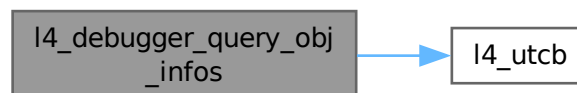
The kernel will only write a number of object information which fits to the passed KU memory. To retrieve missing object information, repeat the call and adapt the *skip* parameter accordingly.

If this system call is performed several times, the number of kernel objects might have changed in the meantime.

Definition at line 177 of file [obj_info.h](#).

References [L4_NOTHROW](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



16.566 obj_info.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (C) 2023, 2025 Kernkonzept GmbH.
00003  * Author(s): Frank Mehnert <frank.mehnert@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00014
00015 #pragma once
00016
00017 #include <l4/sys/compiler.h>
00018 #include <l4/sys/debugger.h>
00019 #include <l4/sys/l4int.h>
00020
00021 struct L4_kobj_info
00022 {
00023     // Type_mapping: See Jdb_mapdb::info_obj_mapping().
00024     struct Mapping
00025     {
00026         enum { Type = 0 };
00027         l4_uint64_t mapping_ptr;
00028         char space_name[16];
00029         l4_uint32_t cap_idx;
00030         l4_uint16_t entry_rights;
00031         l4_uint16_t entry_flags;
00032         l4_uint64_t entry_ptr;
00033     };
00034
00035     // Type_thread: See Jdb_tcb::info_kobject().
00036     struct Thread
00037     {
00038         enum { Type = 1 };
00039         bool is_kernel;
00040         bool is_current;
00041         bool in_ready_list;
00042         bool is_kernel_task;
00043         l4_uint32_t home_cpu;
00044         l4_uint32_t current_cpu;
00045         l4_int64_t ref_cnt;
00046         l4_uint64_t space_id;
00047     };
00048
00049     // Type_space: See Jdb_space::info_kobject().
00050     struct Space
00051     {
00052         enum { Type = 2 };
00053         bool is_kernel;
00054         l4_int64_t ref_cnt;
00055     };
00056
00057     // Type_vm: See Jdb_vm::info_kobject().
00058     struct Vm
00059     {
00060         enum { Type = 3 };
00061         l4_uint64_t utcb;
00062         l4_uint64_t pc;
00063     };
00064
00065     // Type_ipc_gate: See Jdb_ipc_gate::info_kobject().
00066     struct Ipc_gate
00067     {
00068         enum { Type = 4 };
00069         l4_uint64_t label;
00070         l4_uint64_t thread_id;
00071     };
00072
00073     // Type_irq: See Jdb_kobject_irq::info_kobject().
00074     struct Irq_sender
00075     {
00076         enum { Type = 5 };
00077         char chip_type[10];
00078         l4_uint16_t flags;
00079         l4_uint32_t pin;
00080         l4_uint64_t label;
00081         l4_uint64_t target_id;
00082         l4_int64_t queued;
00083     };
00084
00085     // Type_irq: See Jdb_kobject_irq::info_kobject().
00086     struct Irq_semaphore
00087     {
00088         enum { Type = 6 };

```

```

00089     char chip_type[10];
00090     l4_uint16_t flags;
00091     l4_uint32_t pin;
00092     l4_uint64_t sender_id;
00093     l4_uint64_t target_id;
00094     l4_int64_t queued;
00095 };
00096
00097 // Type_factory: See Jdb_factory::info_kobject().
00098 struct Factory
00099 {
00100     enum { Type = 7 };
00101     l4_uint64_t current;
00102     l4_uint64_t limit;
00103 };
00104
00105 struct Jdb          { enum { Type = 8 }; };
00106 struct Scheduler    { enum { Type = 9 }; };
00107 struct Vlog         { enum { Type = 10 }; };
00108 struct Pfc          { enum { Type = 11 }; };
00109 struct Dmar_space   { enum { Type = 12 }; };
00110 struct Iommu        { enum { Type = 13 }; };
00111 struct Smmu         { enum { Type = 14 }; };
00112
00113 l4_uint64_t type;
00114 l4_uint64_t id;
00115 l4_uint64_t mapping_ptr;
00116 l4_uint64_t ref_cnt;
00117 union
00118 {
00119     Thread thread;
00120     Space space;
00121     Vm vm;
00122     Ipc_gate ipc_gate;
00123     Irq_sender irq_sender;
00124     Irq_semaphore irq_semaphore;
00125     Factory factory;
00126     Mapping mapping;
00127     l4_uint64_t raw[5];
00128 };
00129 };
00130
00131 static_assert(sizeof(L4_kobj_info) == 64, "Size of Jobj_info");
00132
00151 L4_INLINE l4_msgtag_t
00152 l4_debugger_query_obj_infos(l4_cap_idx_t cap, l4_addr_t ku_mem_addr,
00153                             l4_size_t ku_mem_size, l4_umword_t skip,
00154                             l4_umword_t *result_cnt, l4_umword_t *result_all)
00155     L4_NOTHROW;
00156
00157 L4_INLINE l4_msgtag_t
00158 l4_debugger_query_obj_infos_u(l4_cap_idx_t cap, l4_addr_t ku_mem_addr,
00159                              l4_size_t ku_mem_size, l4_umword_t skip,
00160                              l4_umword_t *result_cnt, l4_umword_t *result_all,
00161                              l4_utcb_t *utcb) L4_NOTHROW
00162 {
00163     l4_utcb_mr()->mr[0] = L4_DEBUGGER_OBJ_INFO_OP;
00164     l4_utcb_mr()->mr[1] = ku_mem_addr;
00165     l4_utcb_mr()->mr[2] = ku_mem_size;
00166     l4_utcb_mr()->mr[3] = skip;
00167
00168     l4_msgtag_t tag = l4_invoke_debugger(cap, l4_msgtag(0, 4, 0, 0), utcb);
00169
00170     *result_cnt = l4_utcb_mr()->mr[0];
00171     *result_all = l4_utcb_mr()->mr[1];
00172
00173     return tag;
00174 }
00175
00176 L4_INLINE l4_msgtag_t
00177 l4_debugger_query_obj_infos(l4_cap_idx_t cap, l4_addr_t ku_mem_addr,
00178                             l4_size_t ku_mem_size, l4_umword_t skip,
00179                             l4_umword_t *result_cnt, l4_umword_t *result_all)
00180     L4_NOTHROW
00181 {
00182     return l4_debugger_query_obj_infos_u(cap, ku_mem_addr, ku_mem_size, skip,
00183                                           result_cnt, result_all, l4_utcb());
00184 }

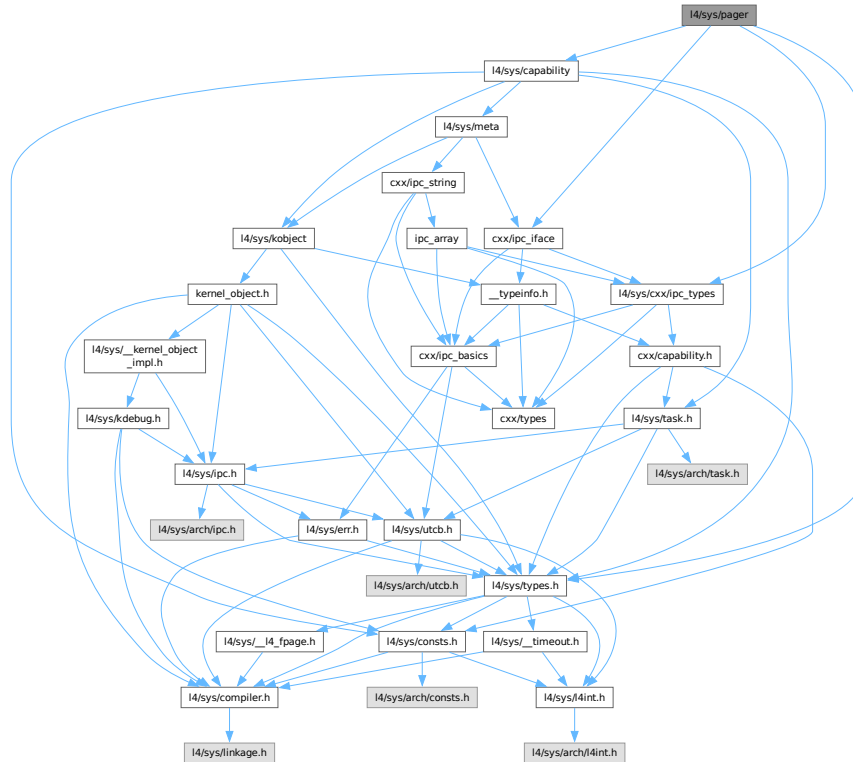
```

16.567 l4/sys/pager File Reference

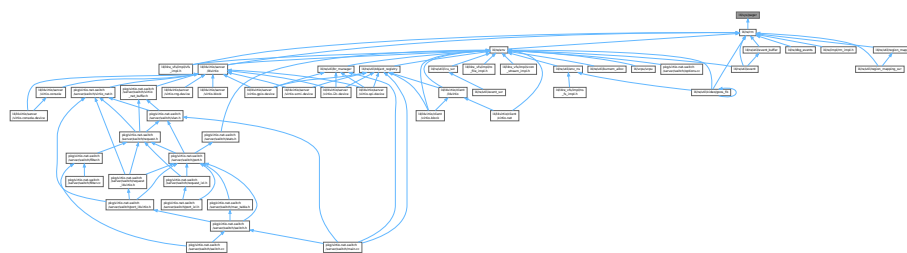
Pager and lo_pager C++ interface.

```
#include <l4/sys/capability>
#include <l4/sys/types.h>
#include <l4/sys/cxx/ipc_types>
#include <l4/sys/cxx/ipc_iface>
```

Include dependency graph for pager:



This graph shows which files directly or indirectly include this file:



Data Structures

- class [L4::lo_pager](#)
lo_pager interface.
- class [L4::Pager](#)
Pager interface including the *lo_pager* interface.

Namespaces

- namespace [L4](#)
L4 low-level kernel interface.

16.567.1 Detailed Description

Pager and Io_pager C++ interface.

Definition in file [pager](#).

16.568 pager

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00008 #pragma once
00009
00010 #include <l4/sys/capability>
00011 #include <l4/sys/types.h>
00012 #include <l4/sys/cxx/ipc_types>
00013 #include <l4/sys/cxx/ipc_iface>
00014
00015 namespace L4 {
00016
00017 class L4_EXPORT Io_pager :
00018     public Kobject_0t<Io_pager, L4_PROTO_IO_PAGE_FAULT>
00019 {
00020 public:
00021     L4_INLINE_RPC(
00022         l4_msgtag_t, io_page_fault, (l4_fpage_t io_pfa, l4_umword_t pc,
00023                                     L4::Ipc::Rcv_fpage rwin,
00024                                     L4::Ipc::Opt<L4::Ipc::Snd_fpage &> fp));
00025
00026     typedef L4::Typeid::Rpc_nocode<io_page_fault_t> Rpcs;
00027 };
00028
00029 class L4_EXPORT Pager :
00030     public Kobject_t<Pager, Io_pager, L4_PROTO_PAGE_FAULT>
00031 {
00032 public:
00033     L4_INLINE_RPC(
00034         l4_msgtag_t, page_fault, (l4_umword_t pfa, l4_umword_t pc,
00035                                   L4::Ipc::Rcv_fpage rwin,
00036                                   L4::Ipc::Opt<L4::Ipc::Snd_fpage &> fp));
00037
00038     typedef L4::Typeid::Rpc_nocode<page_fault_t> Rpcs;
00039 };
00040
00041 }
00042

```

16.569 l4/sys/platform_control File Reference

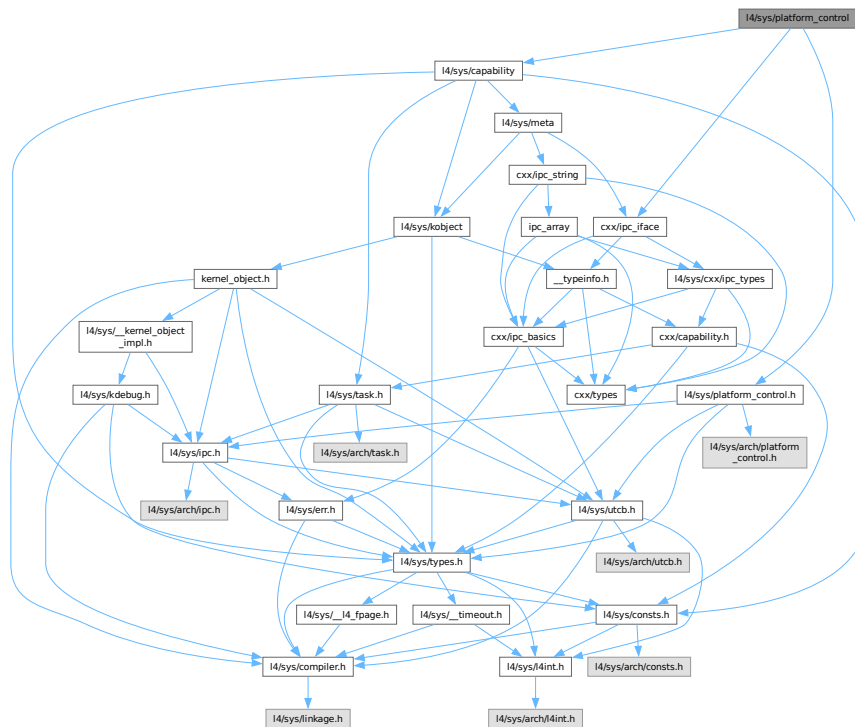
Platform control object.

```

#include <l4/sys/capability>
#include <l4/sys/platform_control.h>

```

```
#include <l4/sys/cxx/ipc_iface>
Include dependency graph for platform_control:
```



Data Structures

- class [L4::Platform_control](#)

[L4](#) C++ interface for controlling platform-wide properties, see [Platform Control C API](#) for the C interface.

Namespaces

- namespace [L4](#)

[L4](#) low-level kernel interface.

16.569.1 Detailed Description

Platform control object.

Definition in file [platform_control](#).

16.570 platform_control

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2014 Steffen Liebergeld <steffen.liebergeld@kernkonzept.com>
00004  *      Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00005  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00006  *      economic rights: Technische Universität Dresden (Germany)
00007  *
00008  * License: see LICENSE.spdx (in this directory or the directories above)
00009  */
00010
00011 #pragma once
00012
00013 #include <l4/sys/capability>
00014 #include <l4/sys/platform_control.h>
00015 #include <l4/sys/cxx/ipc_iface>
00016
00017 namespace L4 {
00018
00019 class L4_EXPORT Platform_control
00020 : public Kobject_t<Platform_control, Kobject, L4_PROTO_PLATFORM_CTL>
00021 {
00022 public:
00023     L4_INLINE_RPC_OP(L4_PLATFORM_CTL_SYS_SUSPEND_OP,
00024                     l4_msgtag_t, system_suspend, (l4_umword_t extras));
00025
00026     L4_INLINE_RPC_OP(L4_PLATFORM_CTL_SYS_SHUTDOWN_OP,
00027                     l4_msgtag_t, system_shutdown, (l4_umword_t reboot));
00028
00029     L4_INLINE_RPC_OP(L4_PLATFORM_CTL_CPU_ALLOW_SHUTDOWN_OP,
00030                     l4_msgtag_t, cpu_allow_shutdown,
00031                     (l4_umword_t phys_id, l4_umword_t enable));
00032
00033     L4_INLINE_RPC_OP(L4_PLATFORM_CTL_CPU_ENABLE_OP,
00034                     l4_msgtag_t, cpu_enable, (l4_umword_t phys_id));
00035
00036     L4_INLINE_RPC_OP(L4_PLATFORM_CTL_CPU_DISABLE_OP,
00037                     l4_msgtag_t, cpu_disable, (l4_umword_t phys_id));
00038
00039     typedef L4::Typeid::Rpcsys<system_suspend_t, system_shutdown_t,
00040                               cpu_allow_shutdown_t, cpu_enable_t,
00041                               cpu_disable_t> Rpcsys;
00042 };
00043
00044 }
00045
00046
00047
00048

```

16.571 platform_control.h

```

00001 /*
00002  * Copyright (C) 2026 Kernkonzept GmbH.
00003  * Author(s): Jan Klötzke <jan.kloetzke@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once

```

16.572 platform_control.h

```

00001 /*
00002  * Copyright (C) 2024 Kernkonzept GmbH.
00003  * Author(s): Jan Klötzke <jan.kloetzke@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include <l4/sys/__platform_control-arm.h>

```

16.573 platform_control.h

```

00001 /*
00002  * Copyright (C) 2024 Kernkonzept GmbH.
00003  * Author(s): Jan Klötzke <jan.kloetzke@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include <l4/sys/__platform_control-arm.h>

```

16.574 l4/sys/platform_control.h File Reference

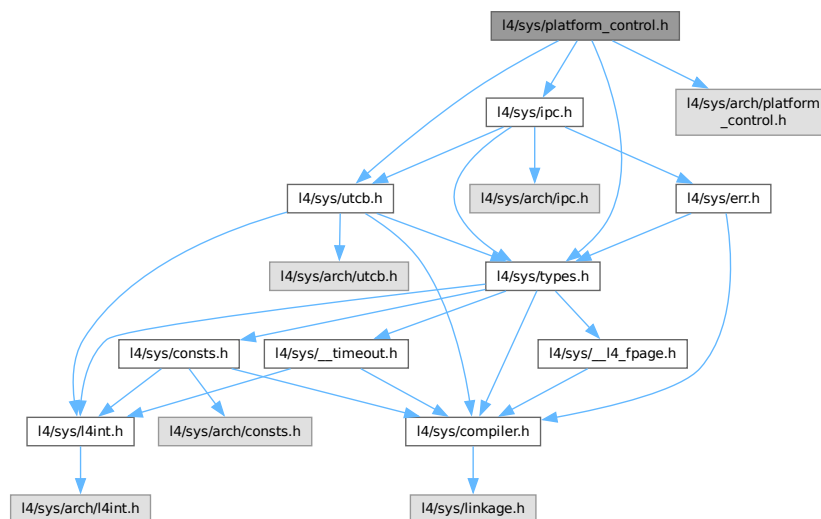
Platform control object.

```

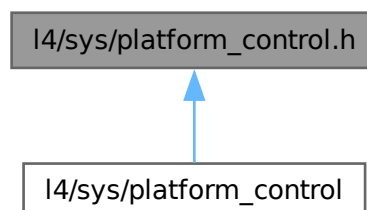
#include <l4/sys/types.h>
#include <l4/sys/utcb.h>
#include <l4/sys/ipc.h>
#include <l4/sys/arch/platform_control.h>

```

Include dependency graph for platform_control.h:



This graph shows which files directly or indirectly include this file:



Enumerations

- enum `L4_platform_ctl_ops` {
`L4_PLATFORM_CTL_SYS_SUSPEND_OP = 0UL` , `L4_PLATFORM_CTL_SYS_SHUTDOWN_OP = 1UL` ,
`L4_PLATFORM_CTL_CPU_ALLOW_SHUTDOWN_OP = 2UL` , `L4_PLATFORM_CTL_CPU_ENABLE_OP = 3UL` ,
`L4_PLATFORM_CTL_CPU_DISABLE_OP = 4UL` , `L4_PLATFORM_CTL_SET_TASK_ASID_OP = 0x10UL` }
Operations on platform-control objects.
- enum `L4_platform_ctl_proto` { `L4_PROTO_PLATFORM_CTL = 0` }
Predefined protocol type for messages to platform-control objects.

Functions

- `l4_msgtag_t l4_platform_ctl_system_suspend (l4_cap_idx_t pfc, l4_umword_t extras) L4_NOTHROW`
Enter suspend to RAM.
- `l4_msgtag_t l4_platform_ctl_system_shutdown (l4_cap_idx_t pfc, l4_umword_t reboot) L4_NOTHROW`
Shutdown or reboot the system.
- `l4_msgtag_t l4_platform_ctl_cpu_allow_shutdown (l4_cap_idx_t pfc, l4_umword_t phys_id, l4_umword_t enable) L4_NOTHROW`
Allow a CPU to be shut down.
- `l4_msgtag_t l4_platform_ctl_cpu_enable (l4_cap_idx_t pfc, l4_umword_t phys_id) L4_NOTHROW`
Enable an offline CPU.
- `l4_msgtag_t l4_platform_ctl_cpu_disable (l4_cap_idx_t pfc, l4_umword_t phys_id) L4_NOTHROW`
Disable an online CPU.

16.574.1 Detailed Description

Platform control object.

Definition in file [platform_control.h](#).

16.575 platform_control.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00007  *
00008  * License: see LICENSE.spdx (in this directory or the directories above)
00009  */
00010
00011 #pragma once
00012
00013 #include <l4/sys/types.h>
00014 #include <l4/sys/utcb.h>
00015
00030
00031
00049 L4_INLINE l4_msgtag_t
00050 l4_platform_ctl_system_suspend(l4_cap_idx_t pfc,
00051                               l4_umword_t extras) L4_NOTHROW;
00052
00056 L4_INLINE l4_msgtag_t
00057 l4_platform_ctl_system_suspend_u(l4_cap_idx_t pfc,
00058                                  l4_umword_t extras,
00059                                  l4_utcb_t *utcb) L4_NOTHROW;
00060
00061
00070 L4_INLINE l4_msgtag_t

```

```

00071 l4_platform_ctl_system_shutdown(l4_cap_idx_t pfc,
00072                                l4_umword_t reboot) L4_NOTHROW;
00073
00077 L4_INLINE l4_msgtag_t
00078 l4_platform_ctl_system_shutdown_u(l4_cap_idx_t pfc,
00079                                   l4_umword_t reboot,
00080                                   l4_utcb_t *utcb) L4_NOTHROW;
00081
00091 L4_INLINE l4_msgtag_t
00092 l4_platform_ctl_cpu_allow_shutdown(l4_cap_idx_t pfc,
00093                                    l4_umword_t phys_id,
00094                                    l4_umword_t enable) L4_NOTHROW;
00095
00099 L4_INLINE l4_msgtag_t
00100 l4_platform_ctl_cpu_allow_shutdown_u(l4_cap_idx_t pfc,
00101                                     l4_umword_t phys_id,
00102                                     l4_umword_t enable,
00103                                     l4_utcb_t *utcb) L4_NOTHROW;
00104
00114 L4_INLINE l4_msgtag_t
00115 l4_platform_ctl_cpu_enable(l4_cap_idx_t pfc,
00116                            l4_umword_t phys_id) L4_NOTHROW;
00117
00121 L4_INLINE l4_msgtag_t
00122 l4_platform_ctl_cpu_enable_u(l4_cap_idx_t pfc,
00123                              l4_umword_t phys_id,
00124                              l4_utcb_t *utcb) L4_NOTHROW;
00125
00136 L4_INLINE l4_msgtag_t
00137 l4_platform_ctl_cpu_disable(l4_cap_idx_t pfc,
00138                             l4_umword_t phys_id) L4_NOTHROW;
00139
00143 L4_INLINE l4_msgtag_t
00144 l4_platform_ctl_cpu_disable_u(l4_cap_idx_t pfc,
00145                               l4_umword_t phys_id,
00146                               l4_utcb_t *utcb) L4_NOTHROW;
00147 /* ends l4_platform_control_api group */
00149
00150
00159 enum L4_platform_ctl_ops
00160 {
00161     L4_PLATFORM_CTL_SYS_SUSPEND_OP      = 0UL,
00162     L4_PLATFORM_CTL_SYS_SHUTDOWN_OP     = 1UL,
00163     L4_PLATFORM_CTL_CPU_ALLOW_SHUTDOWN_OP = 2UL,
00164     L4_PLATFORM_CTL_CPU_ENABLE_OP       = 3UL,
00165     L4_PLATFORM_CTL_CPU_DISABLE_OP      = 4UL,
00166
00167     L4_PLATFORM_CTL_SET_TASK_ASID_OP     = 0x10UL,
00168 };
00169
00174 enum L4_platform_ctl_proto
00175 {
00181     L4_PROTO_PLATFORM_CTL = 0
00182 };
00183
00184 /* IMPLEMENTATION -----*/
00185
00186 #include <l4/sys/ipc.h>
00187
00188 L4_INLINE l4_msgtag_t
00189 l4_platform_ctl_system_suspend_u(l4_cap_idx_t pfc,
00190                                  l4_umword_t extras,
00191                                  l4_utcb_t *utcb) L4_NOTHROW
00192 {
00193     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00194     v->mr[0] = L4_PLATFORM_CTL_SYS_SUSPEND_OP;
00195     v->mr[1] = extras;
00196     return l4_ipc_call(pfc, utcb, l4_msgtag(L4_PROTO_PLATFORM_CTL, 2, 0, 0),
00197                       L4_IPC_NEVER);
00198 }
00199
00200 L4_INLINE l4_msgtag_t
00201 l4_platform_ctl_system_shutdown_u(l4_cap_idx_t pfc,
00202                                   l4_umword_t reboot,
00203                                   l4_utcb_t *utcb) L4_NOTHROW
00204 {
00205     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00206     v->mr[0] = L4_PLATFORM_CTL_SYS_SHUTDOWN_OP;
00207     v->mr[1] = reboot;
00208     return l4_ipc_call(pfc, utcb, l4_msgtag(L4_PROTO_PLATFORM_CTL, 2, 0, 0),
00209                       L4_IPC_NEVER);
00210 }
00211
00212
00213 L4_INLINE l4_msgtag_t
00214 l4_platform_ctl_system_suspend(l4_cap_idx_t pfc,
00215                                l4_umword_t extras) L4_NOTHROW
00216 {

```

```

00217     return l4_platform_ctl_system_suspend_u(pfc, extras, l4_utcb());
00218 }
00219
00220 L4_INLINE l4_msgtag_t
00221 l4_platform_ctl_system_shutdown(l4_cap_idx_t pfc,
00222                                l4_umword_t reboot) L4_NOTHROW
00223 {
00224     return l4_platform_ctl_system_shutdown_u(pfc, reboot, l4_utcb());
00225 }
00226
00227 L4_INLINE l4_msgtag_t
00228 l4_platform_ctl_cpu_allow_shutdown_u(l4_cap_idx_t pfc,
00229                                     l4_umword_t phys_id,
00230                                     l4_umword_t enable,
00231                                     l4_utcb_t *utcb) L4_NOTHROW
00232 {
00233     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00234     v->mr[0] = L4_PLATFORM_CTL_CPU_ALLOW_SHUTDOWN_OP;
00235     v->mr[1] = phys_id;
00236     v->mr[2] = enable;
00237     return l4_ipc_call(pfc, utcb, l4_msgtag(L4_PROTO_PLATFORM_CTL, 3, 0, 0),
00238                       L4_IPC_NEVER);
00239 }
00240
00241 L4_INLINE l4_msgtag_t
00242 l4_platform_ctl_cpu_allow_shutdown(l4_cap_idx_t pfc,
00243                                   l4_umword_t phys_id,
00244                                   l4_umword_t enable) L4_NOTHROW
00245 {
00246     return l4_platform_ctl_cpu_allow_shutdown_u(pfc, phys_id, enable, l4_utcb());
00247 }
00248
00249 L4_INLINE l4_msgtag_t
00250 l4_platform_ctl_cpu_enable_u(l4_cap_idx_t pfc,
00251                              l4_umword_t phys_id,
00252                              l4_utcb_t *utcb) L4_NOTHROW
00253 {
00254     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00255     v->mr[0] = L4_PLATFORM_CTL_CPU_ENABLE_OP;
00256     v->mr[1] = phys_id;
00257     return l4_ipc_call(pfc, utcb, l4_msgtag(L4_PROTO_PLATFORM_CTL, 2, 0, 0),
00258                       L4_IPC_NEVER);
00259 }
00260
00261 L4_INLINE l4_msgtag_t
00262 l4_platform_ctl_cpu_disable_u(l4_cap_idx_t pfc,
00263                               l4_umword_t phys_id,
00264                               l4_utcb_t *utcb) L4_NOTHROW
00265 {
00266     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00267     v->mr[0] = L4_PLATFORM_CTL_CPU_DISABLE_OP;
00268     v->mr[1] = phys_id;
00269     return l4_ipc_call(pfc, utcb, l4_msgtag(L4_PROTO_PLATFORM_CTL, 2, 0, 0),
00270                       L4_IPC_NEVER);
00271 }
00272
00273 L4_INLINE l4_msgtag_t
00274 l4_platform_ctl_cpu_enable(l4_cap_idx_t pfc,
00275                             l4_umword_t phys_id) L4_NOTHROW
00276 {
00277     return l4_platform_ctl_cpu_enable_u(pfc, phys_id, l4_utcb());
00278 }
00279
00280 L4_INLINE l4_msgtag_t
00281 l4_platform_ctl_cpu_disable(l4_cap_idx_t pfc,
00282                              l4_umword_t phys_id) L4_NOTHROW
00283 {
00284     return l4_platform_ctl_cpu_disable_u(pfc, phys_id, l4_utcb());
00285 }
00286
00287 #include <l4/sys/arch/platform_control.h>

```

16.576 platform_control.h

```

00001 /*
00002  * Copyright (C) 2026 Kernkonzept GmbH.
00003  * Author(s): Jan Klötzke <jan.kloetzke@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once

```


Data Structures

- class [L4::Rcv_endpoint](#)

Interface for kernel objects that allow to receive IPC from them.

Namespaces

- namespace [L4](#)

[L4](#) low-level kernel interface.

16.578.1 Detailed Description

The C++ Receive endpoint interface.

Definition in file [rcv_endpoint](#).

16.579 rcv_endpoint

[Go to the documentation of this file.](#)

```

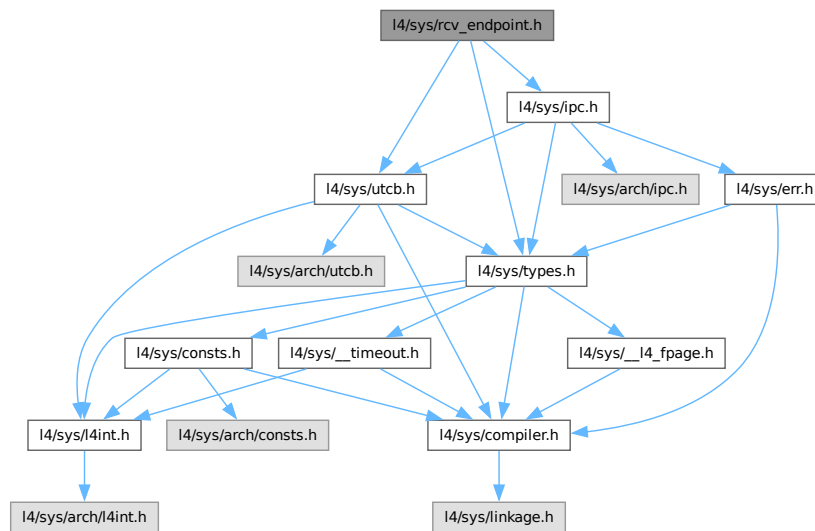
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2017 Alexander Warg <alexander.warg@kernkonzept.com>
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011 #pragma once
00012
00013 #include <l4/sys/rcv_endpoint.h>
00014 #include <l4/sys/snd_destination>
00015 #include <l4/sys/types.h>
00016 #include <l4/sys/capability>
00017 #include <l4/sys/cxx/ipc_iface>
00018
00019 namespace L4 {
00020
00021 class Thread;
00022
00030 class L4_EXPORT Rcv_endpoint :
00031     public Kobject_t<Rcv_endpoint, Kobject, L4_PROTO_KOBJECT,
00032         Type_info::Demand_t<1> >
00033 {
00034 public:
00066     L4_INLINE_RPC_OP(L4_RCV_EP_BIND_OP,
00067         l4_msgtag_t, bind_thread, (Ipc::Cap<Thread> t, l4_umword_t label));
00068
00101     l4_msgtag_t bind_snd_destination(Cap<Snd_destination> snd_dst, l4_umword_t label)
00102     {
00103         return l4_rcv_ep_bind_snd_destination(cap(), snd_dst.cap(), label);
00104     }
00105
00106     typedef L4::Typeid::Rpcsys_sys<bind_thread_t> Rpcsys;
00107 };
00108
00109 }
```

16.580 I4/sys/rcv_endpoint.h File Reference

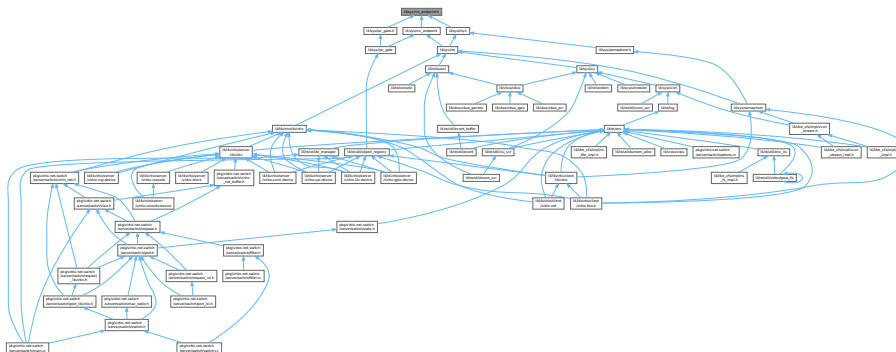
Receive endpoint C interface.

```
#include <l4/sys/utcb.h>
#include <l4/sys/types.h>
#include <l4/sys/ipc.h>
```

Include dependency graph for rcv_endpoint.h:



This graph shows which files directly or indirectly include this file:



Enumerations

- enum [L4_rcv_ep_ops](#) { [L4_RCV_EP_BIND_OP](#) = 0x10 }
- Receive endpoint operations.

Functions

- [l4_msgtag_t l4_rcv_ep_bind_thread](#) ([l4_cap_idx_t](#) ep, [l4_cap_idx_t](#) thread, [l4_umword_t](#) label)
Bind the IPC receive endpoint to a thread.
- [l4_msgtag_t l4_rcv_ep_bind_snd_destination](#) ([l4_cap_idx_t](#) ep, [l4_cap_idx_t](#) snd_dst, [l4_umword_t](#) label)
Bind the IPC receive endpoint to a send destination (a thread).

16.580.1 Detailed Description

Receive endpoint C interface.

Definition in file [rcv_endpoint.h](#).

16.580.2 Enumeration Type Documentation

16.580.2.1 L4_rcv_ep_ops

enum [L4_rcv_ep_ops](#)

Receive endpoint operations.

Enumerator

| | |
|-------------------|---------------------------------|
| L4_RCV_EP_BIND_OP | Bind to thread or thread group. |
|-------------------|---------------------------------|

Definition at line 93 of file [rcv_endpoint.h](#).

16.581 rcv_endpoint.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2017 Alexander Warg <alexander.warg@kernkonzept.com>
00007  *
00008  * License: see LICENSE.spdx (in this directory or the directories above)
00009  */
00010 #pragma once
00011
00012 #include <l4/sys/utcb.h>
00013 #include <l4/sys/types.h>
00014
00045 L4_INLINE l4_msgtag_t
00046 l4_rcv_ep_bind_thread(l4_cap_idx_t ep, l4_cap_idx_t thread,
00047                      l4_umword_t label);
00048
00076 L4_INLINE l4_msgtag_t
00077 l4_rcv_ep_bind_snd_destination(l4_cap_idx_t ep, l4_cap_idx_t snd_dst,
00078                               l4_umword_t label);
00079
00084 L4_INLINE l4_msgtag_t
00085 l4_rcv_ep_bind_thread_u(l4_cap_idx_t ep, l4_cap_idx_t thread,
00086                        l4_umword_t label, l4_utcb_t *utcb);
00087
00088 L4_INLINE l4_msgtag_t
00089 l4_rcv_ep_bind_snd_destination_u(l4_cap_idx_t ep, l4_cap_idx_t snd_dst,
00090                                 l4_umword_t label, l4_utcb_t *utcb);
00091
00093 enum L4_rcv_ep_ops
00094 {
00095     L4_RCV_EP_BIND_OP      = 0x10,
00096 };
00097
00098 /* IMPLEMENTATION -----*/
00099
00100 #include <l4/sys/ipc.h>
00101
00102 L4_INLINE l4_msgtag_t
00103 l4_rcv_ep_bind_thread_u(l4_cap_idx_t ep,
00104                        l4_cap_idx_t thread, l4_umword_t label,
00105                        l4_utcb_t *utcb)
00106 {
00107     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);

```

```

00108     m->mr[0] = L4_RCV_EP_BIND_OP;
00109     m->mr[1] = label;
00110     m->mr[2] = l4_map_obj_control(0, 0);
00111     m->mr[3] = l4_obj_fpage(thread, 0, L4_CAP_FPAGE_RWS).raw;
00112     return l4_ipc_call(ep, utcb, l4_msgtag(L4_PROTO_KOBJECT, 2, 1, 0),
00113                       L4_IPC_NEVER);
00114 }
00115
00116 L4_INLINE l4_msgtag_t
00117 l4_rcv_ep_bind_snd_destination_u(l4_cap_idx_t ep,
00118                                  l4_cap_idx_t snd_dst, l4_umword_t label,
00119                                  l4_utcb_t *utcb)
00120 {
00121     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00122     m->mr[0] = L4_RCV_EP_BIND_OP;
00123     m->mr[1] = label;
00124     m->mr[2] = l4_map_obj_control(0, 0);
00125     m->mr[3] = l4_obj_fpage(snd_dst, 0, L4_CAP_FPAGE_RWS).raw;
00126     return l4_ipc_call(ep, utcb, l4_msgtag(L4_PROTO_KOBJECT, 2, 1, 0),
00127                       L4_IPC_NEVER);
00128 }
00129
00130 L4_INLINE l4_msgtag_t
00131 l4_rcv_ep_bind_thread(l4_cap_idx_t ep, l4_cap_idx_t thread,
00132                       l4_umword_t label)
00133 {
00134     return l4_rcv_ep_bind_thread_u(ep, thread, label, l4_utcb());
00135 }
00136
00137 L4_INLINE l4_msgtag_t
00138 l4_rcv_ep_bind_snd_destination(l4_cap_idx_t ep, l4_cap_idx_t snd_dst,
00139                                 l4_umword_t label)
00140 {
00141     return l4_rcv_ep_bind_snd_destination_u(ep, snd_dst, label, l4_utcb());
00142 }

```

16.582 l4/sys/scheduler File Reference

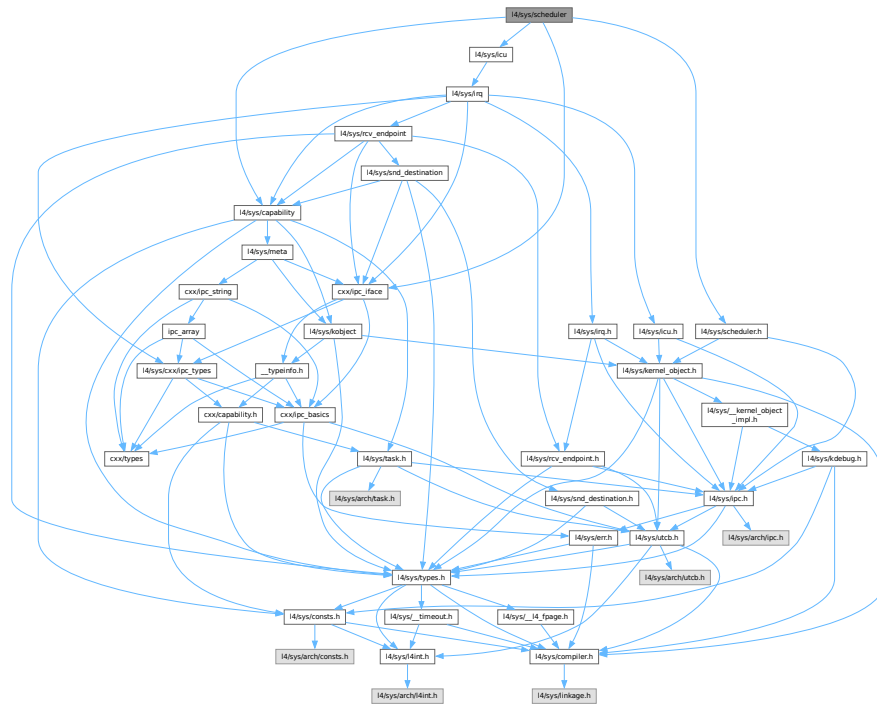
Scheduler object functions.

```

#include <l4/sys/icu>
#include <l4/sys/scheduler.h>
#include <l4/sys/capability>
#include <l4/sys/cxx/ipc_iface>

```


Include dependency graph for scheduler:



Data Structures

- class [L4::Scheduler](#)

C++ interface of the [Scheduler](#) kernel object, see [Scheduler](#) for the C interface.

Namespaces

- namespace [L4](#)

[L4](#) low-level kernel interface.

16.582.1 Detailed Description

Scheduler object functions.

Definition in file [scheduler](#).

16.583 scheduler

[Go to the documentation of this file.](#)

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  * Alexander Warg <warg@os.inf.tu-dresden.de>
00009  * economic rights: Technische Universität Dresden (Germany)
00010  *
```

```

00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013  #pragma once
00014
00015  #include <l4/sys/icu>
00016  #include <l4/sys/scheduler.h>
00017  #include <l4/sys/capability>
00018  #include <l4/sys/cxx/ipc_iface>
00019
00020  namespace L4 {
00021
00046  class L4_EXPORT Scheduler :
00047      public Kobject_t<Scheduler, Icu, L4_PROTO_SCHEDULER,
00048          Type_info::Demand_t<1> >
00049  {
00050  public:
00051      // ABI function for 'info' call
00052      L4_INLINE_RPC_NF_OP(L4_SCHEDULER_INFO_OP,
00053          l4_msgtag_t, info, (l4_umword_t gran_offset, l4_umword_t *map,
00054              l4_umword_t *cpu_max, l4_umword_t *sched_classes));
00055
00074      l4_msgtag_t info(l4_umword_t *cpu_max, l4_sched_cpu_set_t *cpus,
00075          l4_umword_t *sched_classes = nullptr,
00076          l4_utcb_t *utcb = l4_utcb()) const noexcept
00077      {
00078          l4_umword_t max = 0;
00079          l4_umword_t sc = 0;
00080          l4_msgtag_t t =
00081              info_t::call(c(), cpus->gran_offset, &cpus->map, &max, &sc, utcb);
00082          if (cpu_max)
00083              *cpu_max = max;
00084          if (sched_classes)
00085              *sched_classes = sc;
00086          return t;
00087      }
00088
00112      L4_INLINE_RPC_OP(L4_SCHEDULER_RUN_THREAD_OP,
00113          l4_msgtag_t, run_thread, (Ipc::Cap<Thread> thread, l4_sched_param_t const &sp));
00114
00141      L4_INLINE_RPC_OP(L4_SCHEDULER_IDLE_TIME_OP,
00142          l4_msgtag_t, idle_time, (l4_sched_cpu_set_t const &cpus,
00143              l4_kernel_clock_t *us));
00144
00154      bool is_online(l4_umword_t cpu, l4_utcb_t *utcb = l4_utcb()) const noexcept
00155      { return l4_scheduler_is_online_u(cap(), cpu, utcb); }
00156
00157      typedef L4::Typeid::Rpcsys<info_t, run_thread_t, idle_time_t> Rpcsys;
00158  };
00159  }

```

16.584 l4/sys/scheduler.h File Reference

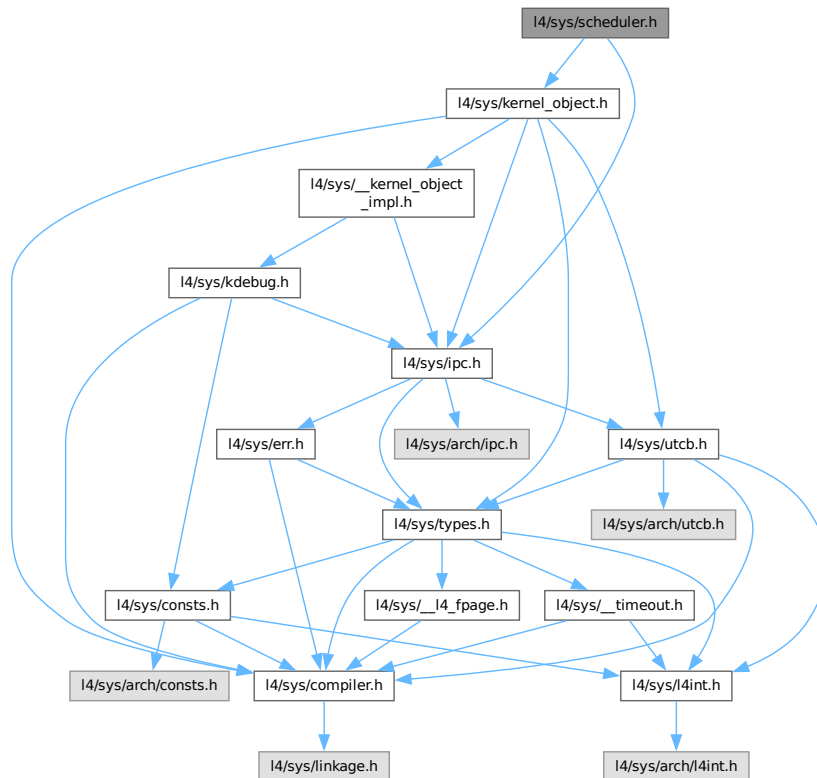
Scheduler object functions.

```

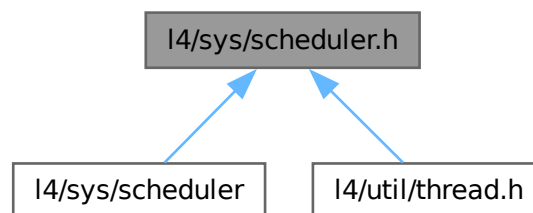
#include <l4/sys/kernel_object.h>
#include <l4/sys/ipc.h>

```

Include dependency graph for scheduler.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [l4_sched_cpu_set_t](#)
CPU sets.
- struct [l4_sched_param_t](#)
Scheduler parameter set.

Typedefs

- typedef struct l4_sched_cpu_set_t **l4_sched_cpu_set_t**
CPU sets.
- typedef struct l4_sched_param_t **l4_sched_param_t**
Scheduler parameter set.

Enumerations

- enum **L4_scheduler_classes** { **L4_SCHEDULER_CLASS_FIXED_PRIO** = 1UL << 1, **L4_SCHEDULER_CLASS_WFQ** = 1UL << 2 }
Supported scheduler classes.
- enum **L4_scheduler_ops** { **L4_SCHEDULER_INFO_OP** = 0UL, **L4_SCHEDULER_RUN_THREAD_OP** = 1UL, **L4_SCHEDULER_IDLE_TIME_OP** = 2UL }
Operations on the Scheduler object.

Functions

- **l4_sched_cpu_set_t l4_sched_cpu_set** (**l4_umword_t** offset, unsigned char granularity, **l4_umword_t** map=1) **L4_NOTHROW**
- **l4_msgtag_t l4_scheduler_info** (**l4_cap_idx_t** scheduler, **l4_umword_t** *cpu_max, **l4_sched_cpu_set_t** *cpus) **L4_NOTHROW**)
Get scheduler information.
- **l4_msgtag_t l4_scheduler_info_with_classes** (**l4_cap_idx_t** scheduler, **l4_umword_t** *cpu_max, **l4_sched_cpu_set_t** *cpus, **l4_umword_t** *sched_classes) **L4_NOTHROW**)
Get scheduler information.
- **l4_sched_param_t l4_sched_param** (unsigned prio, **l4_umword_t** quantum=0) **L4_NOTHROW**
Construct scheduler parameter.
- **l4_msgtag_t l4_scheduler_run_thread** (**l4_cap_idx_t** scheduler, **l4_cap_idx_t** thread, **l4_sched_param_t** const *sp) **L4_NOTHROW**)
Run a thread on a Scheduler.
- **l4_msgtag_t l4_scheduler_idle_time** (**l4_cap_idx_t** scheduler, **l4_sched_cpu_set_t** const *cpus, **l4_kernel_clock_t** *us) **L4_NOTHROW**)
Query the idle time (in μ s) of a CPU.
- int **l4_scheduler_is_online** (**l4_cap_idx_t** scheduler, **l4_umword_t** cpu) **L4_NOTHROW**
Query if a CPU is online.

16.584.1 Detailed Description

Scheduler object functions.

Definition in file [scheduler.h](#).

16.585 scheduler.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #pragma once
00013
00014 #include <l4/sys/kernel_object.h>
00015 #include <l4/sys/ipc.h>
00016
00040
00046 enum L4_scheduler_classes
00047 {
00049     L4_SCHEDULER_CLASS_FIXED_PRIO = 1UL < 1,
00051     L4_SCHEDULER_CLASS_WFQ        = 1UL < 2,
00052 };
00053
00058 typedef struct l4_sched_cpu_set_t
00059 {
00072     l4_umword_t gran_offset;
00073
00077     l4_umword_t map;
00078
00079 #ifdef __cplusplus
00081     unsigned char granularity() const { return gran_offset >> 24; }
00083     unsigned offset() const { return gran_offset & 0x0ffffff; }
00090     void set(unsigned char granularity, unsigned offset)
00091     {
00092         gran_offset = (static_cast<l4_umword_t>(granularity) << 24)
00093             | (offset & 0x0ffffff);
00094     }
00095 #endif
00096 } l4_sched_cpu_set_t;
00097
00108 L4_INLINE l4_sched_cpu_set_t
00109 l4_sched_cpu_set(l4_umword_t offset, unsigned char granularity,
00110                 l4_umword_t map L4_DEFAULT_PARAM(1)) L4_NOTHROW;
00111
00128 L4_INLINE l4_msgtag_t
00129 l4_scheduler_info(l4_cap_idx_t scheduler, l4_umword_t *cpu_max,
00130                  l4_sched_cpu_set_t *cpus)
00131     L4_NOTHROW __attribute__((nonnull (3)));
00132
00154 L4_INLINE l4_msgtag_t
00155 l4_scheduler_info_with_classes(l4_cap_idx_t scheduler, l4_umword_t *cpu_max,
00156                               l4_sched_cpu_set_t *cpus,
00157                               l4_umword_t *sched_classes)
00158     L4_NOTHROW __attribute__((nonnull (3)));
00159
00163 L4_INLINE l4_msgtag_t
00164 l4_scheduler_info_u(l4_cap_idx_t scheduler, l4_umword_t *cpu_max,
00165                    l4_sched_cpu_set_t *cpus, l4_umword_t *sched_classes,
00166                    l4_utcb_t *utcb) L4_NOTHROW __attribute__((nonnull (3, 5)));
00167
00168
00173 typedef struct l4_sched_param_t
00174 {
00176     l4_sched_cpu_set_t affinity;
00182     l4_umword_t prio;
00184     l4_umword_t quantum;
00185 } l4_sched_param_t;
00186
00195 L4_INLINE l4_sched_param_t
00196 l4_sched_param(unsigned prio,
00197                l4_umword_t quantum L4_DEFAULT_PARAM(0)) L4_NOTHROW;
00198
00206 L4_INLINE l4_msgtag_t
00207 l4_scheduler_run_thread(l4_cap_idx_t scheduler,
00208                        l4_cap_idx_t thread, l4_sched_param_t const *sp)
00209     L4_NOTHROW __attribute__((nonnull));
00210
00214 L4_INLINE l4_msgtag_t
00215 l4_scheduler_run_thread_u(l4_cap_idx_t scheduler, l4_cap_idx_t thread,
00216                          l4_sched_param_t const *sp, l4_utcb_t *utcb)
00217     L4_NOTHROW __attribute__((nonnull));
00218
00226 L4_INLINE l4_msgtag_t
00227 l4_scheduler_idle_time(l4_cap_idx_t scheduler, l4_sched_cpu_set_t const *cpus,
00228                       l4_kernel_clock_t *us)

```

```

00229             L4_NOTHROW __attribute__((nonnull));
00230
00234 L4_INLINE l4_msgtag_t
00235 l4_scheduler_idle_time_u(l4_cap_idx_t scheduler, l4_sched_cpu_set_t const *cpus,
00236                         l4_kernel_clock_t *us, l4_utcb_t *utcb)
00237             L4_NOTHROW __attribute__((nonnull));
00238
00239
00240
00251 L4_INLINE int
00252 l4_scheduler_is_online(l4_cap_idx_t scheduler, l4_umword_t cpu) L4_NOTHROW;
00253
00257 L4_INLINE int
00258 l4_scheduler_is_online_u(l4_cap_idx_t scheduler, l4_umword_t cpu,
00259                         l4_utcb_t *utcb) L4_NOTHROW __attribute__((nonnull));
00260
00261
00262
00269 enum l4_scheduler_ops
00270 {
00271     L4_SCHEDULER_INFO_OP      = 0UL,
00272     L4_SCHEDULER_RUN_THREAD_OP = 1UL,
00273     L4_SCHEDULER_IDLE_TIME_OP = 2UL,
00274 };
00275
00276 /***** Implementations *****/
00277
00278 L4_INLINE l4_sched_cpu_set_t
00279 l4_sched_cpu_set(l4_umword_t offset, unsigned char granularity,
00280                 l4_umword_t map) L4_NOTHROW
00281 {
00282     l4_sched_cpu_set_t cs;
00283     cs.gran_offset = ((l4_umword_t)granularity < 24) | (offset & 0x00ffffff);
00284     cs.map         = map;
00285     return cs;
00286 }
00287
00288 L4_INLINE l4_sched_param_t
00289 l4_sched_param(unsigned prio, l4_umword_t quantum) L4_NOTHROW
00290 {
00291     l4_sched_param_t sp;
00292     sp.prio          = prio;
00293     sp.quantum       = quantum;
00294     sp.affinity      = l4_sched_cpu_set(0, ~0, 1);
00295     return sp;
00296 }
00297
00298
00299 L4_INLINE l4_msgtag_t
00300 l4_scheduler_info_u(l4_cap_idx_t scheduler, l4_umword_t *cpu_max,
00301                   l4_sched_cpu_set_t *cpus, l4_umword_t *sched_classes,
00302                   l4_utcb_t *utcb) L4_NOTHROW
00303 {
00304     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00305     l4_msgtag_t res;
00306
00307     m->mr[0] = L4_SCHEDULER_INFO_OP;
00308     m->mr[1] = cpus->gran_offset;
00309
00310     res = l4_ipc_call(scheduler, utcb, l4_msgtag(L4_PROTO_SCHEDULER, 2, 0, 0), L4_IPC_NEVER);
00311
00312     if (l4_msgtag_has_error(res))
00313         return res;
00314
00315     cpus->map = m->mr[0];
00316
00317     if (cpu_max)
00318         *cpu_max = m->mr[1];
00319
00320     if (sched_classes)
00321         *sched_classes = m->mr[2];
00322
00323     return res;
00324 }
00325
00326 L4_INLINE l4_msgtag_t
00327 l4_scheduler_run_thread_u(l4_cap_idx_t scheduler, l4_cap_idx_t thread,
00328                          l4_sched_param_t const *sp, l4_utcb_t *utcb) L4_NOTHROW
00329 {
00330     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00331     m->mr[0] = L4_SCHEDULER_RUN_THREAD_OP;
00332     m->mr[1] = sp->affinity.gran_offset;
00333     m->mr[2] = sp->affinity.map;
00334     m->mr[3] = sp->prio;
00335     m->mr[4] = sp->quantum;
00336     m->mr[5] = l4_map_obj_control(0, 0);
00337     m->mr[6] = l4_obj_fpage(thread, 0, L4_CAP_FPAGE_RWS).raw;

```

```

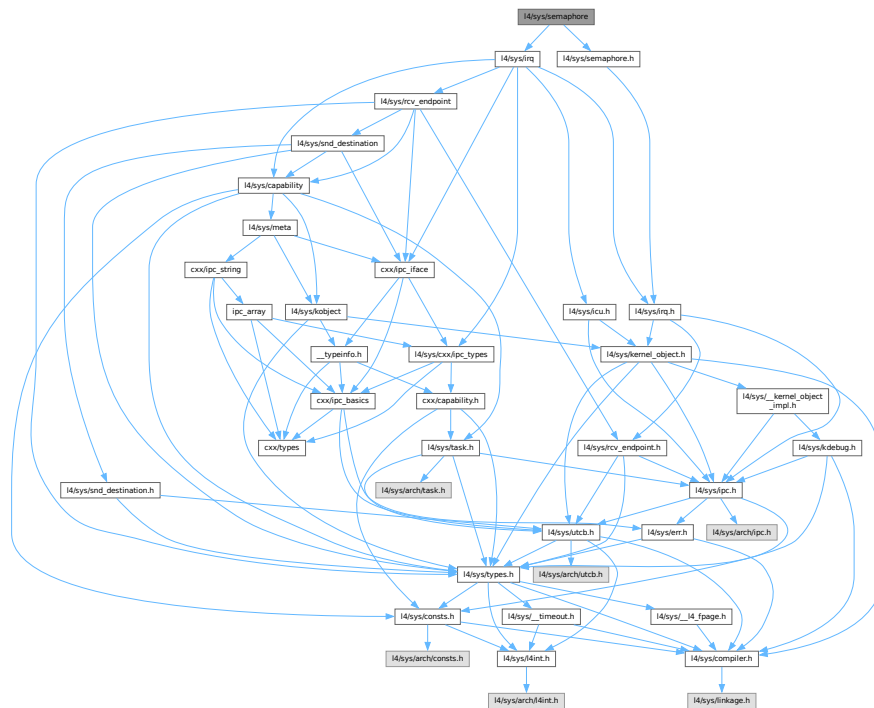
00338
00339     return l4_ipc_call(scheduler, utcb, l4_msgtag(L4_PROTO_SCHEDULER, 5, 1, 0), L4_IPC_NEVER);
00340 }
00341
00342 L4_INLINE l4_msgtag_t
00343 l4_scheduler_idle_time_u(l4_cap_idx_t scheduler, l4_sched_cpu_set_t const *cpus,
00344                        l4_kernel_clock_t *us, l4_utcb_t *utcb) L4_NOTHROW
00345 {
00346     l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00347     l4_msgtag_t res;
00348
00349     v->mr[0] = L4_SCHEDULER_IDLE_TIME_OP;
00350     v->mr[1] = cpus->gran_offset;
00351     v->mr[2] = cpus->map;
00352
00353     res = l4_ipc_call(scheduler, utcb,
00354                      l4_msgtag(L4_PROTO_SCHEDULER, 3, 0, 0), L4_IPC_NEVER);
00355
00356     if (l4_msgtag_has_error(res))
00357         return res;
00358
00359     *us = v->mr64[l4_utcb_mr64_idx(0)];
00360
00361     return res;
00362 }
00363
00364
00365 L4_INLINE int
00366 l4_scheduler_is_online_u(l4_cap_idx_t scheduler, l4_umword_t cpu,
00367                        l4_utcb_t *utcb) L4_NOTHROW
00368 {
00369     l4_sched_cpu_set_t s;
00370     l4_msgtag_t r;
00371     s.gran_offset = cpu;
00372     r = l4_scheduler_info_u(scheduler, NULL, &s, NULL, utcb);
00373     if (l4_msgtag_has_error(r) || l4_msgtag_label(r) < 0)
00374         return 0;
00375
00376     return s.map & 1;
00377 }
00378
00379
00380 L4_INLINE l4_msgtag_t
00381 l4_scheduler_info(l4_cap_idx_t scheduler, l4_umword_t *cpu_max,
00382                  l4_sched_cpu_set_t *cpus) L4_NOTHROW
00383 {
00384     return l4_scheduler_info_u(scheduler, cpu_max, cpus, NULL, l4_utcb());
00385 }
00386
00387 L4_INLINE l4_msgtag_t
00388 l4_scheduler_info_with_classes(l4_cap_idx_t scheduler, l4_umword_t *cpu_max,
00389                               l4_sched_cpu_set_t *cpus,
00390                               l4_umword_t *sched_classes) L4_NOTHROW
00391 {
00392     return l4_scheduler_info_u(scheduler, cpu_max, cpus, sched_classes, l4_utcb());
00393 }
00394
00395 L4_INLINE l4_msgtag_t
00396 l4_scheduler_run_thread(l4_cap_idx_t scheduler,
00397                        l4_cap_idx_t thread, l4_sched_param_t const *sp) L4_NOTHROW
00398 {
00399     return l4_scheduler_run_thread_u(scheduler, thread, sp, l4_utcb());
00400 }
00401
00402 L4_INLINE l4_msgtag_t
00403 l4_scheduler_idle_time(l4_cap_idx_t scheduler, l4_sched_cpu_set_t const *cpus,
00404                        l4_kernel_clock_t *us) L4_NOTHROW
00405 {
00406     return l4_scheduler_idle_time_u(scheduler, cpus, us, l4_utcb());
00407 }
00408
00409 L4_INLINE int
00410 l4_scheduler_is_online(l4_cap_idx_t scheduler, l4_umword_t cpu) L4_NOTHROW
00411 {
00412     return l4_scheduler_is_online_u(scheduler, cpu, l4_utcb());
00413 }

```

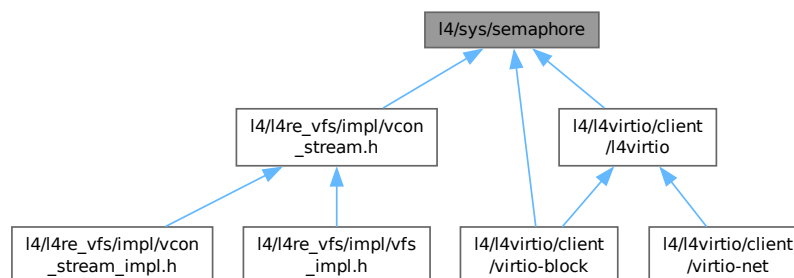
16.586 l4/sys/semaphore File Reference

Semaphore class definition.

```
#include <l4/sys/irq>
#include <l4/sys/semaphore.h>
Include dependency graph for semaphore:
```



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [L4::Semaphore](#)

C++ Kernel-provided semaphore interface, see [Kernel-provided semaphore](#) for the C interface.

Namespaces

- namespace [L4](#)

[L4](#) low-level kernel interface.

16.586.1 Detailed Description

Semaphore class definition.

Definition in file [semaphore](#).

16.587 semaphore

[Go to the documentation of this file.](#)

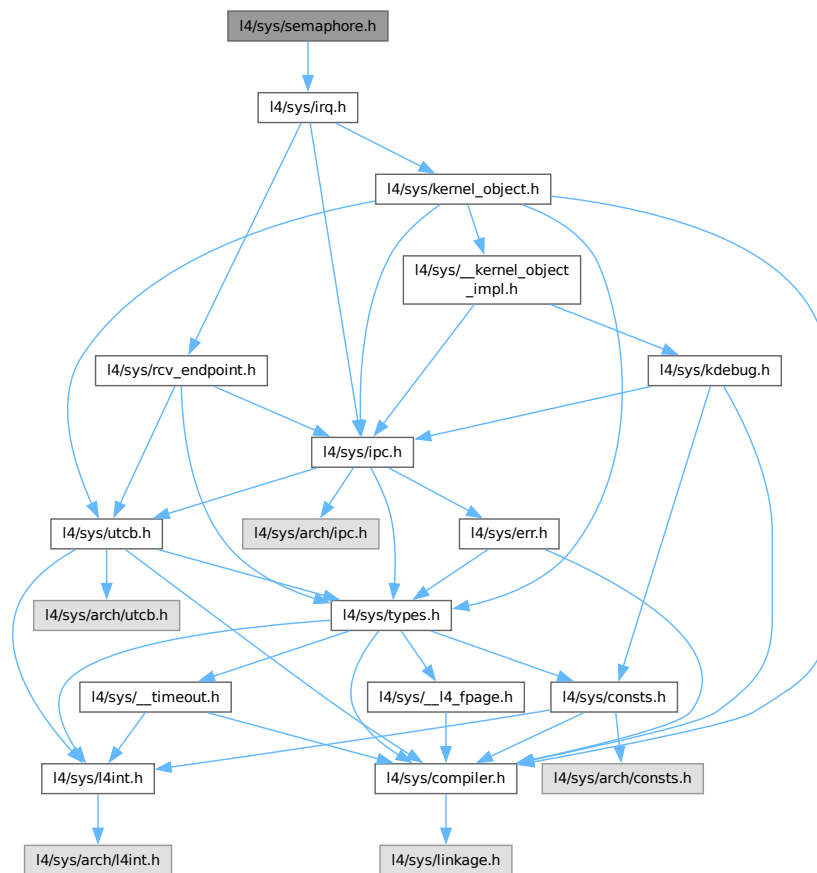
```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2015 Alexander Warg <alexander.warg@kernkonzept.com>
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011
00012 #pragma once
00013
00014 #include <l4/sys/irq>
00015 #include <l4/sys/semaphore.h>
00016
00017 namespace L4 {
00018
00051 struct Semaphore : Kobject_t<Semaphore, Triggerable, L4_PROTO_SEMAPHORE>
00052 {
00067     l4_msgtag_t up(l4_utcb_t *utcb = l4_utcb()) noexcept
00068     { return trigger(utcb); }
00069
00089     l4_msgtag_t down(l4_timeout_t timeout = L4_IPC_NEVER,
00090                     l4_utcb_t *utcb = l4_utcb()) noexcept
00091     { return l4_semaphore_down_u(cap(), timeout, utcb); }
00092 };
00093
00094 }
```

16.588 l4/sys/semaphore.h File Reference

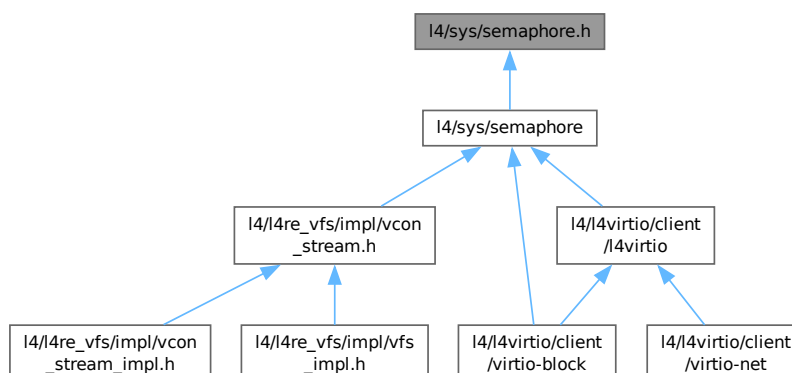
C semaphore interface.

```
#include <l4/sys/irq.h>
```

Include dependency graph for semaphore.h:



This graph shows which files directly or indirectly include this file:



Functions

- [l4_msgtag_t l4_semaphore_up\(l4_cap_idx_t sem\)](#) [L4_NOTHROW](#)

Semaphore up operation (wrapper for trigger()).

- `l4_msgtag_t l4_semaphore_down (l4_cap_idx_t sem, l4_timeout_t timeout) L4_NOTHROW`

Semaphore down operation.

16.588.1 Detailed Description

C semaphore interface.

Definition in file [semaphore.h](#).

16.589 semaphore.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2015 Alexander Warg <alexander.warg@kernkonzept.com>
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011
00012 #pragma once
00013
00014 #include <l4/sys/irq.h>
00015
00024
00025 enum l4_semaphore_op
00026 {
00027     L4_SEMAPHORE_OP_DOWN = 0,
00028     // semaphore up is IRQ_OP_TRIGGER with IRQ/Triggerable protocol
00029 };
00030
00044 L4_INLINE l4_msgtag_t
00045 l4_semaphore_up(l4_cap_idx_t sem) L4_NOTHROW
00046 {
00047     return l4_irq_trigger(sem);
00048 }
00049
00053 L4_INLINE l4_msgtag_t
00054 l4_semaphore_up_u(l4_cap_idx_t sem, l4_utcb_t *utcb) L4_NOTHROW
00055 {
00056     return l4_irq_trigger_u(sem, utcb);
00057 }
00058
00078 L4_INLINE l4_msgtag_t
00079 l4_semaphore_down(l4_cap_idx_t sem, l4_timeout_t timeout) L4_NOTHROW;
00080
00084 L4_INLINE l4_msgtag_t
00085 l4_semaphore_down_u(l4_cap_idx_t sem, l4_timeout_t to,
00086                    l4_utcb_t *utcb) L4_NOTHROW;
00087
00088
00089 L4_INLINE l4_msgtag_t
00090 l4_semaphore_down_u(l4_cap_idx_t sem, l4_timeout_t to,
00091                    l4_utcb_t *utcb) L4_NOTHROW
00092 {
00093     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00094     m->mr[0] = L4_SEMAPHORE_OP_DOWN;
00095     return l4_ipc_call(sem, utcb, l4_msgtag(L4_PROTO_SEMAPHORE, 1, 0, 0), to);
00096 }
00097
00098
00099 L4_INLINE l4_msgtag_t
00100 l4_semaphore_down(l4_cap_idx_t sem, l4_timeout_t to) L4_NOTHROW
00101 {
00102     return l4_semaphore_down_u(sem, to, l4_utcb());
00103 }
00104

```


Namespaces

- namespace [L4](#)
L4 low-level kernel interface.

Functions

- `template<typename T, typename F, typename SMART>`
`Smart_cap< T, SMART > L4::cap_cast (Smart_cap< F, SMART > const &c) noexcept`
static_cast for (smart) capabilities.
- `template<typename T, typename F, typename SMART>`
`Smart_cap< T, SMART > L4::cap_reinterpret_cast (Smart_cap< F, SMART > const &c) noexcept`
reinterpret_cast for (smart) capabilities.

16.590.1 Detailed Description

L4::Capability class.

Author

Alexander Warg alexander.warg@os.inf.tu-dresden.de

Definition in file [smart_capability](#).

16.591 smart_capability

[Go to the documentation of this file.](#)

```
00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00009 /*
00010  * (c) 2008-2009 Author(s)
00011  *      economic rights: Technische Universität Dresden (Germany)
00012  *
00013  * License: see LICENSE.spdx (in this directory or the directories above)
00014  */
00015 #pragma once
00016
00017 #include <l4/sys/capability>
00018
00019 namespace L4 {
00020
00021 template< typename T, typename SMART >
00022 class Smart_cap : public Cap_base, private SMART
00023 {
00024 public:
00025     SMART const &smart() const noexcept { return *this; }
00026
00027     void _delete() noexcept
00028     {
00029         SMART::free(const_cast<Smart_cap<T, SMART>&>(*this));
00030     }
00031
00032     Cap<T> release() const noexcept
00033     {
00034         l4_cap_idx_t r = cap();
00035         SMART::invalidate(const_cast<Smart_cap<T, SMART>&>(*this));
00036
00037         return Cap<T>(r);
00038     }
00039
00040     void reset() noexcept
00041     {
00042         _c = L4_INVALID_CAP;
00043     }
00044 }
```

```

00047     }
00048
00049     Smart_cap() noexcept : Cap_base(Invalid) {}
00050
00051     Smart_cap(Cap_base::Cap_type t) noexcept : Cap_base(t) {}
00052
00061     template< typename O >
00062     Smart_cap(Cap<O> const &p) noexcept : Cap_base(p.cap())
00063     { Cap<T>::template check_convertible_from<O>(); }
00064
00065     template< typename O >
00066     Smart_cap(Cap<O> const &p, SMART const &smart) noexcept
00067     : Cap_base(p.cap()), SMART(smart)
00068     { Cap<T>::template check_convertible_from<O>(); }
00069
00070     template< typename O >
00071     Smart_cap(Smart_cap<O, SMART> const &o) noexcept
00072     : Cap_base(SMART::copy(o)), SMART(o.smart())
00073     { Cap<T>::template check_convertible_from<O>(); }
00074
00075     Smart_cap(Smart_cap const &o) noexcept
00076     : Cap_base(SMART::copy(o)), SMART(o.smart())
00077     { }
00078
00079     template< typename O >
00080     Smart_cap(typename Cap<O>::Cap_type cap) noexcept : Cap_base(cap)
00081     { Cap<T>::template check_convertible_from<O>(); }
00082
00083     void operator = (typename Cap<T>::Cap_type cap) noexcept
00084     {
00085         _delete();
00086         _c = cap;
00087     }
00088
00089     template< typename O >
00090     void operator = (Smart_cap<O, SMART> const &o) noexcept
00091     {
00092         _delete();
00093         _c = this->SMART::copy(o).cap();
00094         this->SMART::operator = (o.smart());
00095         // return *this;
00096     }
00097
00098     Smart_cap const &operator = (Smart_cap const &o) noexcept
00099     {
00100         if (&o == this)
00101             return *this;
00102
00103         _delete();
00104         _c = this->SMART::copy(o).cap();
00105         this->SMART::operator = (o.smart());
00106         return *this;
00107     }
00108
00109 #if __cplusplus >= 201103L
00110     template< typename O >
00111     Smart_cap(Smart_cap<O, SMART> &&o) noexcept
00112     : Cap_base(o.release()), SMART(o.smart())
00113     { Cap<T>::template check_convertible_from<O>(); }
00114
00115     Smart_cap(Smart_cap &&o) noexcept
00116     : Cap_base(o.release()), SMART(o.smart())
00117     { }
00118
00119     template< typename O >
00120     void operator = (Smart_cap<O, SMART> &&o) noexcept
00121     {
00122         _delete();
00123         _c = o.release().cap();
00124         this->SMART::operator = (o.smart());
00125         // return *this;
00126     }
00127
00128     Smart_cap const &operator = (Smart_cap &&o) noexcept
00129     {
00130         if (&o == this)
00131             return *this;
00132
00133         _delete();
00134         _c = o.release().cap();
00135         this->SMART::operator = (o.smart());
00136         return *this;
00137     }
00138 #endif
00139
00143     Cap<T> operator -> () const noexcept { return Cap<T>(_c); }
00144

```

```

00145     Cap<T> get() const noexcept { return Cap<T>(_c); }
00146
00147     ~Smart_cap() noexcept { _delete(); }
00148 };
00149
00150 template< typename T >
00151 class Weak_cap : public Cap_base
00152 {
00153 public:
00154     Weak_cap() noexcept : Cap_base(Invalid) {}
00155
00156     template< typename O >
00157     Weak_cap(typename Cap<O>::Cap_type t) noexcept : Cap_base(t)
00158     { Cap<T>::template check_convertible_from<O>(); }
00159
00160     template< typename O, typename S >
00161     Weak_cap(Smart_cap<O, S> const &c) noexcept : Cap_base(c.cap())
00162     { Cap<T>::template check_convertible_from<O>(); }
00163
00164     Weak_cap(Weak_cap const &o) noexcept : Cap_base(o) {}
00165
00166     template< typename O >
00167     Weak_cap(Weak_cap<O> const &o) noexcept : Cap_base(o)
00168     { Cap<T>::template check_convertible_from<O>(); }
00169 };
00170
00171 namespace Cap_traits {
00172     template< typename T1, typename T2 >
00173     struct Type { enum { Equal = false }; };
00174
00175     template< typename T1 >
00176     struct Type<T1,T1> { enum { Equal = true }; };
00177 };
00178
00179 template< typename T, typename F, typename SMART >
00180 inline
00181 Smart_cap<T, SMART> cap_cast(Smart_cap<F, SMART> const &c) noexcept
00182 {
00183     Cap<T>::template check_castable_from<F>();
00184     return Smart_cap<T, SMART>(Cap<T>(SMART::copy(c).cap()));
00185 }
00186
00187 template< typename T, typename F, typename SMART >
00188 inline
00189 Smart_cap<T, SMART> cap_reinterpret_cast(Smart_cap<F, SMART> const &c) noexcept
00190 {
00191     return Smart_cap<T, SMART>(Cap<T>(SMART::copy(c).cap()));
00192 }
00193
00194 template< typename T, typename F, typename SMART >
00195 inline
00196 Smart_cap<T, SMART> cap_reinterpret_cast(Smart_cap<F, SMART> const &c) noexcept
00197 {
00198     return Smart_cap<T, SMART>(Cap<T>(SMART::copy(c).cap()));
00199 }
00200
00201 template< typename T, typename F, typename SMART >
00202 inline
00203 Smart_cap<T, SMART> cap_reinterpret_cast(Smart_cap<F, SMART> const &c) noexcept
00204 {
00205     return Smart_cap<T, SMART>(Cap<T>(SMART::copy(c).cap()));
00206 }
00207 }

```

16.592 l4/sys/snd_destination File Reference

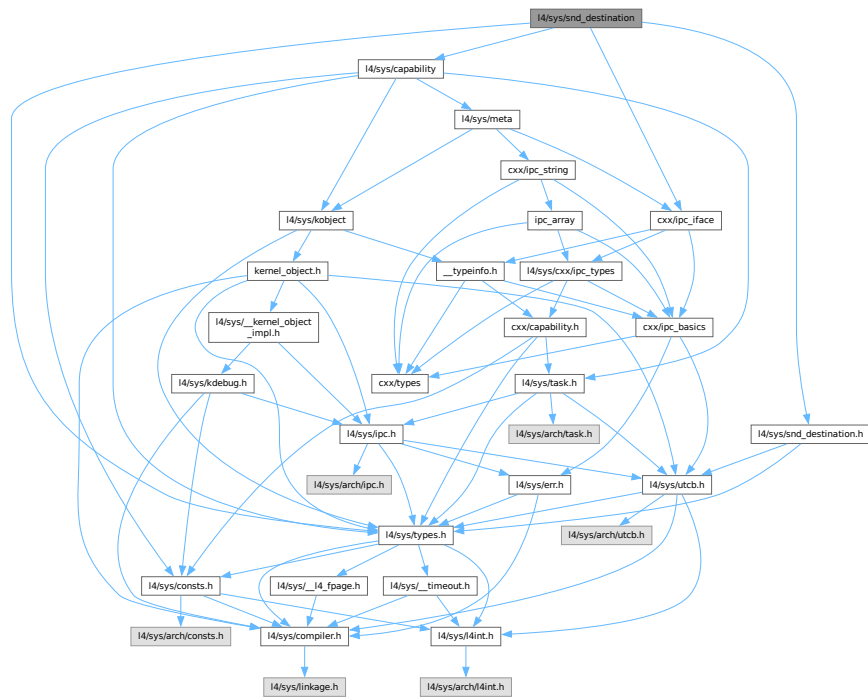
The C++ Sender destination interface.

```

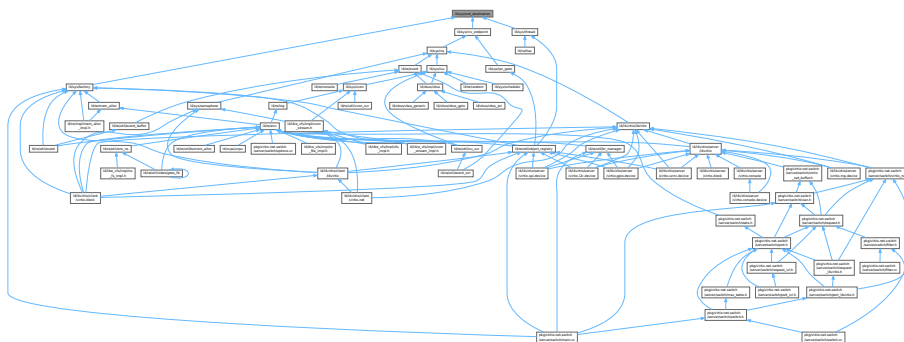
#include <l4/sys/snd_destination.h>
#include <l4/sys/types.h>
#include <l4/sys/capability>
#include <l4/sys/cxx/ipc_iface>

```

Include dependency graph for snd_destination:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **L4**
L4 low-level kernel interface.

16.592.1 Detailed Description

The C++ Sender destination interface.

Definition in file [snd_destination](#).

16.593 snd_destination

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2025 Frank Mehnert <frank.mehnert@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include <l4/sys/snd_destination.h>
00010 #include <l4/sys/types.h>
00011 #include <l4/sys/capability>
00012 #include <l4/sys/cxx/ipc_iface>
00013
00014 namespace L4 {
00015
00016 class L4_EXPORT Snd_destination :
00017     public Kobject_t<Snd_destination, Kobject, L4_PROTO_KOBJECT>
00018 {
00019 };
00020
00021 }

```

16.594 l4/sys/snd_destination.h File Reference

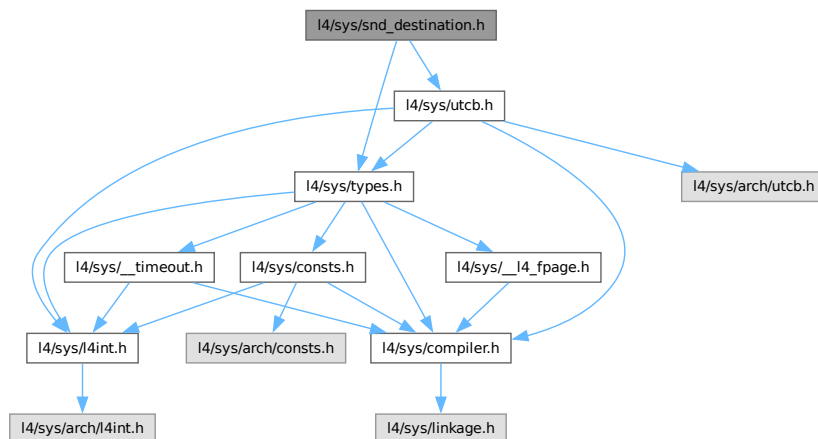
Sender destination endpoint C interface.

```

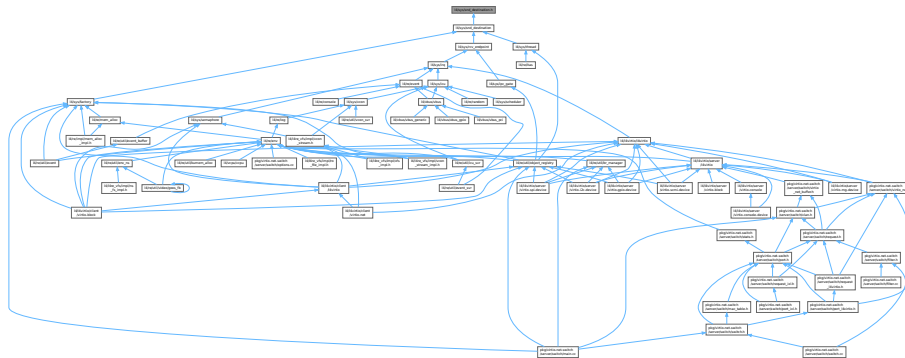
#include <l4/sys/utcb.h>
#include <l4/sys/types.h>

```

Include dependency graph for snd_destination.h:



This graph shows which files directly or indirectly include this file:



16.594.1 Detailed Description

Sender destination endpoint C interface.

Definition in file [snd_destination.h](#).

16.595 snd_destination.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2025 Frank Mehnert <frank.mehnert@kernkonzept.com>
00007  *
00008  * License: see LICENSE.spdx (in this directory or the directories above)
00009  */
00010 #pragma once
00011
00012 #include <l4/sys/utcb.h>
00013 #include <l4/sys/types.h>

```

16.596 l4/sys/task File Reference

Common task related definitions.

```

#include <l4/sys/task.h>
#include <l4/sys/capability>

```

[illegible]

- class `L4::Task`
C++ interface of the `Task` kernel object, see `Task` for the C interface.

- namespace L4
L4 low-level kernel interface.

16.596.1 Detailed Description

Common task related definitions.

Definition in file [task](#).

16.597 task

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009
00010 #pragma once
00011
00012 #include <l4/sys/task.h>
00013 #include <l4/sys/capability>
00014
00015 namespace L4 {
00016
00017 class Task :
00018     public Kobject_t<Task, Kobject, L4_PROTO_TASK,
00019         Type_info::Demand_t<2> >
00020 {
00021 public:
00022     l4_msgtag_t map(Cap<Task> const &src_task,
00023         l4_fpage_t const &snd_fpage, l4_umword_t snd_base,
00024         l4_utcb_t *utcb = l4_utcb()) noexcept
00025     { return l4_task_map_u(cap(), src_task.cap(), snd_fpage, snd_base, utcb); }
00026
00027     l4_msgtag_t unmap(l4_fpage_t const &fpage,
00028         l4_umword_t map_mask,
00029         l4_utcb_t *utcb = l4_utcb()) noexcept
00030     { return l4_task_unmap_u(cap(), fpage, map_mask, utcb); }
00031
00032     l4_msgtag_t unmap_batch(l4_fpage_t const *fpages,
00033         unsigned num_fpages,
00034         l4_umword_t map_mask,
00035         l4_utcb_t *utcb = l4_utcb()) noexcept
00036     { return l4_task_unmap_batch_u(cap(), fpages, num_fpages, map_mask, utcb); }
00037
00038     l4_msgtag_t delete_obj(L4::Cap<void> obj,
00039         l4_utcb_t *utcb = l4_utcb()) noexcept
00040     { return l4_task_delete_obj_u(cap(), obj.cap(), utcb); }
00041
00042     l4_msgtag_t release_cap(L4::Cap<void> cap,
00043         l4_utcb_t *utcb = l4_utcb()) noexcept
00044     { return l4_task_release_cap_u(this->cap(), cap.cap(), utcb); }
00045
00046     l4_msgtag_t cap_valid(Cap<void> const &cap,
00047         l4_utcb_t *utcb = l4_utcb()) noexcept
00048     { return l4_task_cap_valid_u(this->cap(), cap.cap(), utcb); }
00049
00050     l4_msgtag_t cap_equal(Cap<void> const &cap_a,
00051         Cap<void> const &cap_b,
00052         l4_utcb_t *utcb = l4_utcb()) noexcept
00053     { return l4_task_cap_equal_u(cap(), cap_a.cap(), cap_b.cap(), utcb); }
00054
00055     l4_msgtag_t add_ku_mem(l4_fpage_t *fpage,
00056         l4_utcb_t *utcb = l4_utcb()) noexcept
00057     { return l4_task_add_ku_mem_u(cap(), fpage, utcb); }
00058 };
00059
00060
00061
00062

```

16.598 task.h

```

00001 /*
00002  * Copyright (C) 2026 Kernkonzept GmbH.
00003  * Author(s): Jan Klötzke <jan.kloetzke@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once

```

16.599 task.h

```

00001 /*
00002  * (c) 2018 Adam Lackorzynski <adam@l4re.org>
00003  *
00004  * License: see LICENSE.spdx (in this directory or the directories above)
00005  */
00006 #pragma once
00007
00008 #include <l4/sys/__task-arm.h>

```

16.600 task.h

```

00001 /*
00002  * (c) 2018 Adam Lackorzynski <adam@l4re.org>
00003  *
00004  * License: see LICENSE.spdx (in this directory or the directories above)
00005  */
00006 #pragma once
00007
00008 #include <l4/sys/__task-arm.h>

```

16.601 l4/sys/task.h File Reference

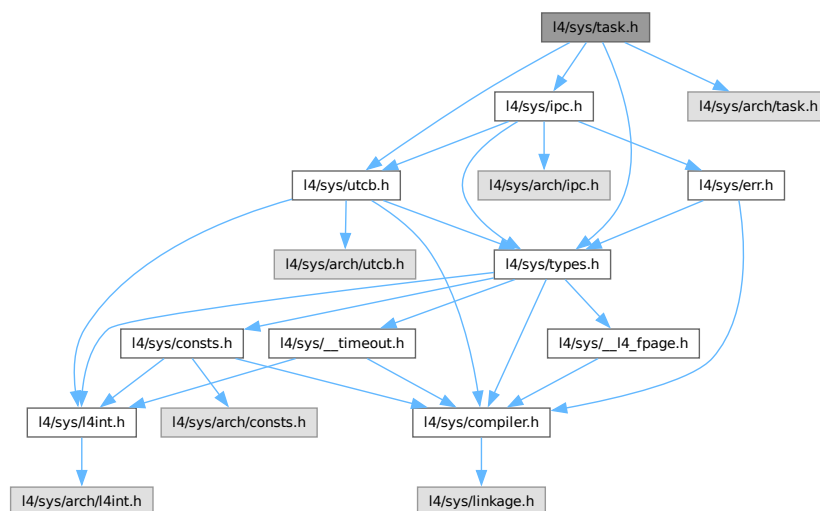
Common task related definitions.

```

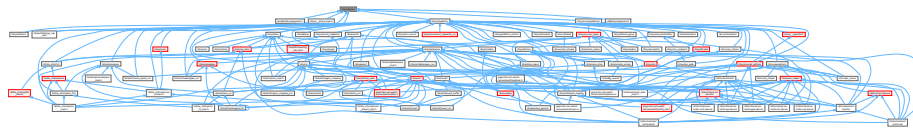
#include <l4/sys/types.h>
#include <l4/sys/utcb.h>
#include <l4/sys/ipc.h>
#include <l4/sys/arch/task.h>

```

Include dependency graph for task.h:



This graph shows which files directly or indirectly include this file:



Enumerations

- enum `L4_task_ops` {
`L4_TASK_MAP_OP` = 0UL , `L4_TASK_UNMAP_OP` = 1UL , `L4_TASK_CAP_INFO_OP` = 2UL ,
`L4_TASK_ADD_KU_MEM_OP` = 3UL ,
`L4_TASK_LDT_SET_X86_OP` = 0x11UL , `L4_TASK_MAP_VGICC_ARM_OP` = 0x12UL }

Operations on task objects.

Functions

- `l4_msgtag_t l4_task_map` (`l4_cap_idx_t` dst_task, `l4_cap_idx_t` src_task, `l4_fpage_t` snd_fpage, `l4_umword_t` snd_base) `L4_NOTHROW`
Map resources available in the source task to a destination task.
- `l4_msgtag_t l4_task_unmap` (`l4_cap_idx_t` task, `l4_fpage_t` fpage, `l4_umword_t` map_mask) `L4_NOTHROW`
Revoke rights from the task.
- `l4_msgtag_t l4_task_unmap_batch` (`l4_cap_idx_t` task, `l4_fpage_t` const *fpages, unsigned num_fpages, `l4_umword_t` map_mask) `L4_NOTHROW`
Revoke rights from a task.
- `l4_msgtag_t l4_task_delete_obj` (`l4_cap_idx_t` task, `l4_cap_idx_t` obj) `L4_NOTHROW`
Release capability and delete object.
- `l4_msgtag_t l4_task_release_cap` (`l4_cap_idx_t` task, `l4_cap_idx_t` cap) `L4_NOTHROW`
Release object capability.
- `l4_msgtag_t l4_task_cap_valid` (`l4_cap_idx_t` task, `l4_cap_idx_t` cap) `L4_NOTHROW`
Check whether a capability is present (refers to an object).
- `l4_msgtag_t l4_task_cap_equal` (`l4_cap_idx_t` task, `l4_cap_idx_t` cap_a, `l4_cap_idx_t` cap_b) `L4_NOTHROW`
Test whether two capabilities point to the same object with the same permissions (only considering selected permissions).
- `l4_msgtag_t l4_task_add_ku_mem` (`l4_cap_idx_t` task, `l4_fpage_t` *ku_mem) `L4_NOTHROW`
Add kernel-user memory.

16.601.1 Detailed Description

Common task related definitions.

Definition in file [task.h](#).

16.602 task.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00008  *      Björn Döbel <doebel@os.inf.tu-dresden.de>,
00009  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  *
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014 #pragma once
00015 #include <l4/sys/types.h>
00016 #include <l4/sys/utcb.h>
00017
00031
00079 L4_INLINE l4_msgtag_t
00080 l4_task_map(l4_cap_idx_t dst_task, l4_cap_idx_t src_task,
00081            l4_fpage_t snd_fpage, l4_umword_t snd_base) L4_NOTHROW;
00082
00086 L4_INLINE l4_msgtag_t
00087 l4_task_map_u(l4_cap_idx_t dst_task, l4_cap_idx_t src_task,
00088              l4_fpage_t snd_fpage, l4_umword_t snd_base, l4_utcb_t *utcb) L4_NOTHROW;
00089
00134 L4_INLINE l4_msgtag_t
00135 l4_task_unmap(l4_cap_idx_t task, l4_fpage_t fpage,
00136              l4_umword_t map_mask) L4_NOTHROW;
00137
00141 L4_INLINE l4_msgtag_t
00142 l4_task_unmap_u(l4_cap_idx_t task, l4_fpage_t fpage,
00143                 l4_umword_t map_mask, l4_utcb_t *utcb) L4_NOTHROW;
00144
00164 L4_INLINE l4_msgtag_t
00165 l4_task_unmap_batch(l4_cap_idx_t task, l4_fpage_t const *fpages,
00166                    unsigned num_fpages, l4_umword_t map_mask) L4_NOTHROW;
00167
00171 L4_INLINE l4_msgtag_t
00172 l4_task_unmap_batch_u(l4_cap_idx_t task, l4_fpage_t const *fpages,
00173                       unsigned num_fpages, l4_umword_t map_mask,
00174                       l4_utcb_t *u) L4_NOTHROW;
00175
00197 L4_INLINE l4_msgtag_t
00198 l4_task_delete_obj(l4_cap_idx_t task, l4_cap_idx_t obj) L4_NOTHROW;
00199
00203 L4_INLINE l4_msgtag_t
00204 l4_task_delete_obj_u(l4_cap_idx_t task, l4_cap_idx_t obj,
00205                      l4_utcb_t *u) L4_NOTHROW;
00206
00225 L4_INLINE l4_msgtag_t
00226 l4_task_release_cap(l4_cap_idx_t task, l4_cap_idx_t cap) L4_NOTHROW;
00227
00231 L4_INLINE l4_msgtag_t
00232 l4_task_release_cap_u(l4_cap_idx_t task, l4_cap_idx_t cap,
00233                       l4_utcb_t *u) L4_NOTHROW;
00234
00235
00253 L4_INLINE l4_msgtag_t
00254 l4_task_cap_valid(l4_cap_idx_t task, l4_cap_idx_t cap) L4_NOTHROW;
00255
00259 L4_INLINE l4_msgtag_t
00260 l4_task_cap_valid_u(l4_cap_idx_t task, l4_cap_idx_t cap, l4_utcb_t *utcb) L4_NOTHROW;
00261
00289 L4_INLINE l4_msgtag_t
00290 l4_task_cap_equal(l4_cap_idx_t task, l4_cap_idx_t cap_a,
00291                  l4_cap_idx_t cap_b) L4_NOTHROW;
00292
00296 L4_INLINE l4_msgtag_t
00297 l4_task_add_ku_mem_u(l4_cap_idx_t task, l4_fpage_t *ku_mem,
00298                     l4_utcb_t *u) L4_NOTHROW;
00299
00326 L4_INLINE l4_msgtag_t
00327 l4_task_add_ku_mem(l4_cap_idx_t task, l4_fpage_t *ku_mem) L4_NOTHROW;
00328
00329
00333 L4_INLINE l4_msgtag_t
00334 l4_task_cap_equal_u(l4_cap_idx_t task, l4_cap_idx_t cap_a,
00335                    l4_cap_idx_t cap_b, l4_utcb_t *utcb) L4_NOTHROW;
00336
00341 enum L4_task_ops
00342 {
00343     L4_TASK_MAP_OP          = 0UL,
00344     L4_TASK_UNMAP_OP       = 1UL,
00345     L4_TASK_CAP_INFO_OP    = 2UL,

```

```

00346     L4_TASK_ADD_KU_MEM_OP      = 3UL,
00347     L4_TASK_LDT_SET_X86_OP      = 0x11UL,
00348     L4_TASK_MAP_VGICC_ARM_OP    = 0x12UL,
00349 };
00350
00351
00352 /* IMPLEMENTATION ----- */
00353
00354 #include <l4/sys/ipc.h>
00355
00356
00357 L4_INLINE l4_msgtag_t
00358 l4_task_map_u(l4_cap_idx_t dst_task, l4_cap_idx_t src_task,
00359              l4_fpage_t snd_fpage, l4_umword_t snd_base, l4_utcb_t *u) L4_NOTHROW
00360 {
00361     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00362     v->mr[0] = L4_TASK_MAP_OP;
00363     v->mr[3] = l4_map_obj_control(0,0);
00364     v->mr[4] = l4_obj_fpage(src_task, 0, L4_CAP_FPAGE_RWS).raw;
00365     v->mr[1] = snd_base;
00366     v->mr[2] = snd_fpage.raw;
00367     return l4_ipc_call(dst_task, u, l4_msgtag(L4_PROTO_TASK, 3, 1, 0), L4_IPC_NEVER);
00368 }
00369
00370 L4_INLINE l4_msgtag_t
00371 l4_task_unmap_u(l4_cap_idx_t task, l4_fpage_t fpage,
00372                l4_umword_t map_mask, l4_utcb_t *u) L4_NOTHROW
00373 {
00374     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00375     v->mr[0] = L4_TASK_UNMAP_OP;
00376     v->mr[1] = map_mask;
00377     v->mr[2] = fpage.raw;
00378     return l4_ipc_call(task, u, l4_msgtag(L4_PROTO_TASK, 3, 0, 0), L4_IPC_NEVER);
00379 }
00380
00381 L4_INLINE l4_msgtag_t
00382 l4_task_unmap_batch_u(l4_cap_idx_t task, l4_fpage_t const *fpages,
00383                      unsigned num_fpages, l4_umword_t map_mask,
00384                      l4_utcb_t *u) L4_NOTHROW
00385 {
00386     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00387     v->mr[0] = L4_TASK_UNMAP_OP;
00388     v->mr[1] = map_mask;
00389     __builtin_memcpy(&v->mr[2], fpages, num_fpages * sizeof(l4_fpage_t));
00390     return l4_ipc_call(task, u, l4_msgtag(L4_PROTO_TASK, 2 + num_fpages, 0, 0), L4_IPC_NEVER);
00391 }
00392
00393 L4_INLINE l4_msgtag_t
00394 l4_task_cap_valid_u(l4_cap_idx_t task, l4_cap_idx_t cap, l4_utcb_t *u) L4_NOTHROW
00395 {
00396     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00397     v->mr[0] = L4_TASK_CAP_INFO_OP;
00398     v->mr[1] = cap & ~1UL;
00399     return l4_ipc_call(task, u, l4_msgtag(L4_PROTO_TASK, 2, 0, 0), L4_IPC_NEVER);
00400 }
00401
00402 L4_INLINE l4_msgtag_t
00403 l4_task_cap_equal_u(l4_cap_idx_t task, l4_cap_idx_t cap_a,
00404                    l4_cap_idx_t cap_b, l4_utcb_t *u) L4_NOTHROW
00405 {
00406     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00407     v->mr[0] = L4_TASK_CAP_INFO_OP;
00408     v->mr[1] = cap_a;
00409     v->mr[2] = cap_b;
00410     return l4_ipc_call(task, u, l4_msgtag(L4_PROTO_TASK, 3, 0, 0), L4_IPC_NEVER);
00411 }
00412
00413 L4_INLINE l4_msgtag_t
00414 l4_task_add_ku_mem_u(l4_cap_idx_t task, l4_fpage_t *ku_mem,
00415                     l4_utcb_t *u) L4_NOTHROW
00416 {
00417     l4_msg_regs_t *v = l4_utcb_mr_u(u);
00418     l4_msgtag_t ret;
00419     v->mr[0] = L4_TASK_ADD_KU_MEM_OP;
00420     v->mr[1] = ku_mem->raw;
00421     ret = l4_ipc_call(task, u, l4_msgtag(L4_PROTO_TASK, 2, 0, 0), L4_IPC_NEVER);
00422     if (!l4_msgtag_has_error(ret))
00423     {
00424         l4_msg_regs_t *v = l4_utcb_mr_u(u);
00425         ku_mem->raw = v->mr[0];
00426     }
00427     return ret;
00428 }
00429
00430
00431
00432 L4_INLINE l4_msgtag_t

```



```

00433 l4_task_map(l4_cap_idx_t dst_task, l4_cap_idx_t src_task,
00434             l4_fpage_t snd_fpage, l4_umword_t snd_base) L4_NOTHROW
00435 {
00436     return l4_task_map_u(dst_task, src_task, snd_fpage, snd_base, l4_utcb());
00437 }
00438
00439 L4_INLINE l4_msgtag_t
00440 l4_task_unmap(l4_cap_idx_t task, l4_fpage_t fpage,
00441             l4_umword_t map_mask) L4_NOTHROW
00442 {
00443     return l4_task_unmap_u(task, fpage, map_mask, l4_utcb());
00444 }
00445
00446 L4_INLINE l4_msgtag_t
00447 l4_task_unmap_batch(l4_cap_idx_t task, l4_fpage_t const *fpages,
00448                   unsigned num_fpages, l4_umword_t map_mask) L4_NOTHROW
00449 {
00450     return l4_task_unmap_batch_u(task, fpages, num_fpages, map_mask,
00451                                 l4_utcb());
00452 }
00453
00454 L4_INLINE l4_msgtag_t
00455 l4_task_delete_obj_u(l4_cap_idx_t task, l4_cap_idx_t obj,
00456                   l4_utcb_t *u) L4_NOTHROW
00457 {
00458     return l4_task_unmap_u(task, l4_obj_fpage(obj, 0, L4_CAP_FPAGE_RWSD),
00459                           L4_FP_DELETE_OBJ, u);
00460 }
00461
00462 L4_INLINE l4_msgtag_t
00463 l4_task_delete_obj(l4_cap_idx_t task, l4_cap_idx_t obj) L4_NOTHROW
00464 {
00465     return l4_task_delete_obj_u(task, obj, l4_utcb());
00466 }
00467
00468 L4_INLINE l4_msgtag_t
00469 l4_task_release_cap_u(l4_cap_idx_t task, l4_cap_idx_t cap,
00470                   l4_utcb_t *u) L4_NOTHROW
00471 {
00472     return l4_task_unmap_u(task, l4_obj_fpage(cap, 0, L4_CAP_FPAGE_RWSD),
00473                           L4_FP_ALL_SPACES, u);
00474 }
00475
00476 L4_INLINE l4_msgtag_t
00477 l4_task_release_cap(l4_cap_idx_t task, l4_cap_idx_t cap) L4_NOTHROW
00478 {
00479     return l4_task_release_cap_u(task, cap, l4_utcb());
00480 }
00481
00482 L4_INLINE l4_msgtag_t
00483 l4_task_cap_valid(l4_cap_idx_t task, l4_cap_idx_t cap) L4_NOTHROW
00484 {
00485     return l4_task_cap_valid_u(task, cap, l4_utcb());
00486 }
00487
00488 L4_INLINE l4_msgtag_t
00489 l4_task_cap_equal(l4_cap_idx_t task, l4_cap_idx_t cap_a,
00490                 l4_cap_idx_t cap_b) L4_NOTHROW
00491 {
00492     return l4_task_cap_equal_u(task, cap_a, cap_b, l4_utcb());
00493 }
00494
00495 L4_INLINE l4_msgtag_t
00496 l4_task_add_ku_mem(l4_cap_idx_t task, l4_fpage_t *ku_mem) L4_NOTHROW
00497 {
00498     return l4_task_add_ku_mem_u(task, ku_mem, l4_utcb());
00499 }
00500
00501
00502 #include <l4/sys/arch/task.h>

```

16.603 task.h

```

00001 /*
00002  * Copyright (C) 2026 Kernkonzept GmbH.
00003  * Author(s): Jan Klötzke <jan.kloetzke@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once

```

16.604 task.h

```

00001 /*
00002  * Copyright (C) 2026 Kernkonzept GmbH.
00003  * Author(s): Jan Klötzke <jan.kloetzke@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once

```

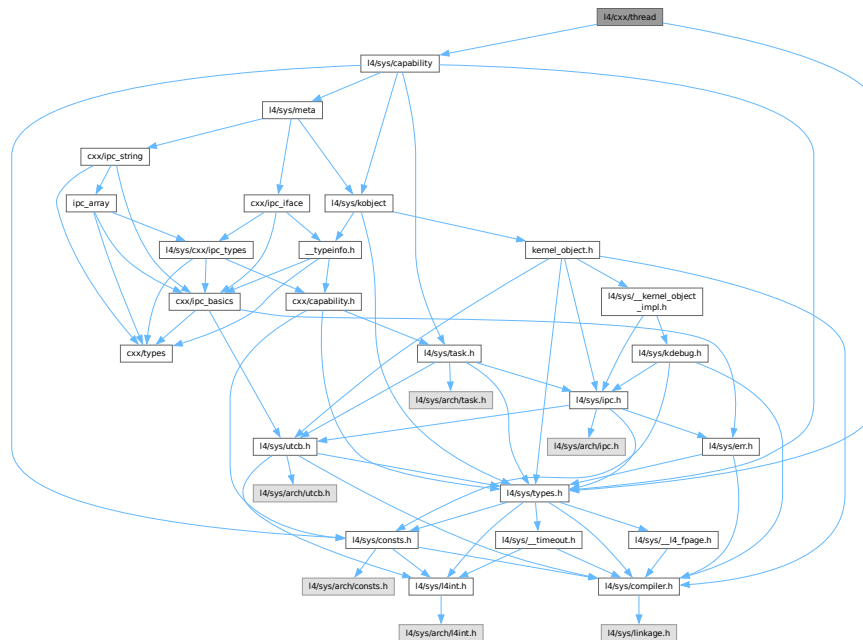
16.605 I4/cxx/thread File Reference

Thread implementation.

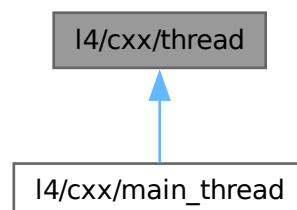
```
#include <l4/sys/capability>
```

```
#include <l4/sys/types.h>
```

Include dependency graph for thread:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace `cxx`
Our C++ library.

16.605.1 Detailed Description

Thread implementation.

Definition in file [thread](#).

16.606 thread

[Go to the documentation of this file.](#)

```
00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2004-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *     economic rights: Technische Universität Dresden (Germany)
00005  * This file is part of TUD:OS and distributed under the terms of the
00006  * GNU Lesser General Public License 2.1.
00007  * Please see the COPYING-LGPL-2.1 file for details.
00008  */
00009
00010 #ifndef CXX_THREAD_H__
00011 #define CXX_THREAD_H__
00012
00013 #include <l4/sys/capability>
00014 #include <l4/sys/types.h>
00015
00016 namespace cxx {
00017
00018     class Thread
00019     {
00020     public:
00021
00022         enum State
00023         {
00024             Dead      = 0,
00025             Running   = 1,
00026             Stopped   = 2,
00027         };
00028
00029         Thread(bool initiate);
00030         Thread(void *stack);
00031         Thread(void *stack, L4::Cap<L4::Thread> const &cap);
00032         virtual ~Thread();
00033         void execute() asm ("L4_Thread_execute");
00034         virtual void run() = 0;
00035         virtual void shutdown() asm ("L4_Thread_shutdown");
00036         void start();
00037         void stop();
00038
00039         L4::Cap<L4::Thread> self() const throw()
00040         { return _cap; }
00041
00042         State state() const
00043         { return _state; }
00044
00045         static void start_cxx_thread(Thread *_this)
00046             asm ("L4_Thread_start_cxx_thread");
00047
00048         static void kill_cxx_thread(Thread *_this)
00049             asm ("L4_Thread_kill_cxx_thread");
00050
00051         static void set_pager(L4::Cap<void>const &p) throw()
00052         { _pager = p; }
00053
00054     private:
00055         int create();
00056
00057         L4::Cap<L4::Thread> _cap;
00058         State _state;
00059     };
00060
00061     L4::Cap<L4::Thread> _cap;
00062     State _state;
00063 }
```


Data Structures

- class [L4::Thread](#)
C++ [L4](#) kernel thread interface, see [Thread](#) for the C interface.
- class [L4::Thread::Attr](#)
[Thread](#) attributes used for [control\(\)](#).
- class [L4::Thread::Modify_senders](#)
[Class](#) wrapping a list of rules which modify the sender label of IPC messages inbound to this thread.

Namespaces

- namespace [L4](#)
[L4](#) low-level kernel interface.

16.607.1 Detailed Description

Common thread related definitions.

Definition in file [thread](#).

16.608 thread

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013
00014 #pragma once
00015
00016 #include <l4/sys/capability>
00017 #include <l4/sys/snd_destination>
00018 #include <l4/sys/thread.h>
00019
00020 namespace L4 {
00021
00052 class Thread :
00053     public Kobject_t<Thread, Snd_destination, L4_PROTO_THREAD,
00054         Type_info::Demand_t<1> >
00055 {
00056 public:
00084     l4_msgtag_t ex_regs(l4_addr_t ip, l4_addr_t sp,
00085                         l4_umword_t flags,
00086                         l4_utcb_t *utcb = l4_utcb()) noexcept
00087     { return l4_thread_ex_regs_u(cap(), ip, sp, flags, utcb); }
00088
00119     l4_msgtag_t ex_regs(l4_addr_t *ip, l4_addr_t *sp,
00120                         l4_umword_t *flags,
00121                         l4_utcb_t *utcb = l4_utcb()) noexcept
00122     { return l4_thread_ex_regs_ret_u(cap(), ip, sp, flags, utcb); }
00123
00124
00137     class Attr
00138     {
00139     private:
00140         friend class L4::Thread;
00141         l4_utcb_t *_u;
00142
00143     public:
00151         explicit Attr(l4_utcb_t *utcb = l4_utcb()) noexcept : _u(utcb)
00152         { l4_thread_control_start_u(utcb); }

```

```

00153
00161 void pager(Cap<void> const &pager) noexcept
00162 { l4_thread_control_pager_u(pager.cap(), _u); }
00163
00170 Cap<void> pager() noexcept
00171 { return Cap<void>(l4_utcb_mr_u(_u)->mr[1]); }
00172
00180 void exc_handler(Cap<void> const &exc_handler) noexcept
00181 { l4_thread_control_exc_handler_u(exc_handler.cap(), _u); }
00182
00189 Cap<void> exc_handler() noexcept
00190 { return Cap<void>(l4_utcb_mr_u(_u)->mr[2]); }
00191
00218 void bind(l4_utcb_t *thread_utcb, Cap<Task> const &task) noexcept
00219 { l4_thread_control_bind_u(thread_utcb, task.cap(), _u); }
00220
00224 void alien(int on) noexcept
00225 { l4_thread_control_alien_u(_u, on); }
00226 };
00227
00243 l4_msgtag_t control(Attr const &attr) noexcept
00244 { return l4_thread_control_commit_u(cap(), attr._u); }
00245
00253 l4_msgtag_t switch_to(l4_utcb_t *utcb = l4_utcb()) noexcept
00254 { return l4_thread_switch_u(cap(), utcb); }
00255
00264 l4_msgtag_t stats_time(l4_kernel_clock_t *us,
00265                        l4_utcb_t *utcb = l4_utcb()) noexcept
00266 { return l4_thread_stats_time_u(cap(), us, utcb); }
00267
00283 l4_msgtag_t vcpu_resume_start(l4_utcb_t *utcb = l4_utcb()) noexcept
00284 { return l4_thread_vcpu_resume_start_u(utcb); }
00285
00334 l4_msgtag_t vcpu_resume_commit(l4_msgtag_t tag,
00335                                l4_utcb_t *utcb = l4_utcb()) noexcept
00336 { return l4_thread_vcpu_resume_commit_u(cap(), tag, utcb); }
00337
00358 l4_msgtag_t vcpu_control(l4_addr_t vcpu_state, l4_utcb_t *utcb = l4_utcb())
00359 noexcept
00360 { return l4_thread_vcpu_control_u(cap(), vcpu_state, utcb); }
00361
00398 l4_msgtag_t vcpu_control_ext(l4_addr_t ext_vcpu_state,
00399                              l4_utcb_t *utcb = l4_utcb()) noexcept
00400 { return l4_thread_vcpu_control_ext_u(cap(), ext_vcpu_state, utcb); }
00401
00427 l4_msgtag_t register_del_irq(Cap<Irq> irq, l4_utcb_t *u = l4_utcb()) noexcept
00428 { return l4_thread_register_del_irq_u(cap(), irq.cap(), u); }
00429
00448 class Modify_senders
00449 {
00450 private:
00451     friend class Thread;
00452     l4_utcb_t *utcb;
00453     unsigned cnt;
00454
00455 public:
00456     explicit Modify_senders(l4_utcb_t *u = l4_utcb()) noexcept
00457     : utcb(u), cnt(1)
00458     {
00459         l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_MODIFY_SENDER_OP;
00460     }
00461
00481 int add(l4_umword_t match_mask, l4_umword_t match,
00482        l4_umword_t del_bits, l4_umword_t add_bits) noexcept
00483 {
00484     l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00485     if (cnt >= L4_UTCB_GENERIC_DATA_SIZE - 4)
00486         return -L4_ENOMEM;
00487     m->mr[cnt++] = match_mask;
00488     m->mr[cnt++] = match;
00489     m->mr[cnt++] = del_bits;
00490     m->mr[cnt++] = add_bits;
00491     return 0;
00492 }
00493 };
00494
00525 l4_msgtag_t modify_senders(Modify_senders const &todo) noexcept
00526 {
00527     return l4_ipc_call(cap(), todo.utcb, l4_msgtag(L4_PROTO_THREAD, todo.cnt, 0, 0), L4_IPC_NEVER);
00528 }
00529
00553 l4_msgtag_t register_doorbell_irq(Cap<Irq> irq, l4_utcb_t *u = l4_utcb()) noexcept
00554 { return l4_thread_register_doorbell_irq_u(cap(), irq.cap(), u); }
00555 };
00556 }

```



```

00017 namespace L4 {
00018
00034 class L4_EXPORT Thread_group :
00035     public Kobject_t<Thread_group, Snd_destination, L4_PROTO_THREAD_GROUP,
00036         Type_info::Demand_t<1>»
00037 {
00038 public:
00053     l4_msgtag_t add(Cap<Thread> thread, l4_utcb_t *utcb = l4_utcb()) noexcept
00054     { return l4_thread_group_add_u(cap(), thread.cap(), utcb); }
00055
00067     l4_msgtag_t remove(Cap<Thread> thread, l4_utcb_t *utcb = l4_utcb()) noexcept
00068     { return l4_thread_group_remove_u(cap(), thread.cap(), utcb); }
00069 };
00070
00071 }

```

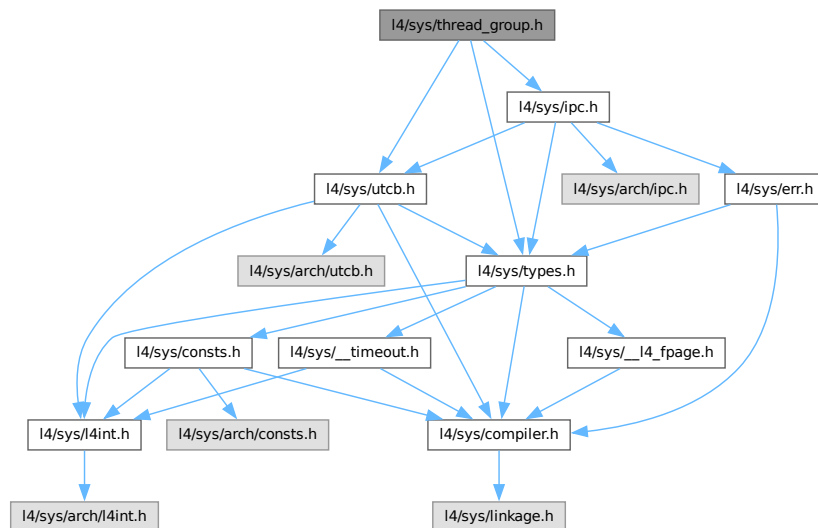
16.611 I4/sys/thread_group.h File Reference

```
#include <l4/sys/types.h>
```

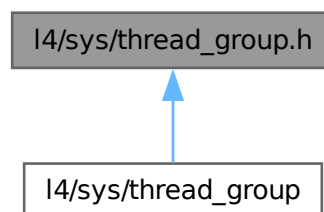
```
#include <l4/sys/utcb.h>
```

```
#include <l4/sys/ipc.h>
```

Include dependency graph for thread_group.h:



This graph shows which files directly or indirectly include this file:



Functions

- [l4_msgtag_t l4_thread_group_add \(l4_cap_idx_t tg, l4_cap_idx_t thread\) L4_NOTHROW](#)
Add thread to a thread group.
- [l4_msgtag_t l4_thread_group_remove \(l4_cap_idx_t tg, l4_cap_idx_t thread\) L4_NOTHROW](#)
Remove thread from a thread group.

16.612 thread_group.h

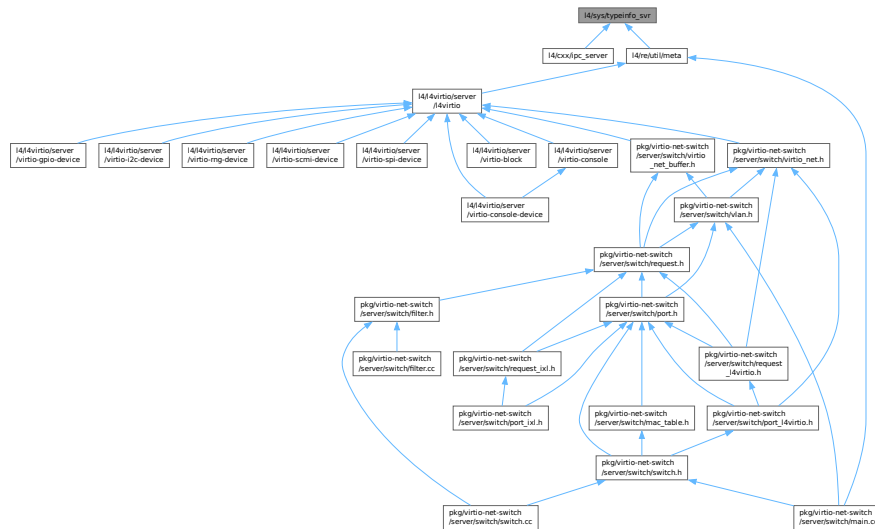
[Go to the documentation of this file.](#)

```

00001  /*
00002  * Copyright (C) 2022-2025 Kernkonzept GmbH.
00003  * Author(s): Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00004  *           Frank Mehnert <frank.mehnert@kernkonzept.com>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00011  #pragma once
00012
00013  #include <l4/sys/types.h>
00014  #include <l4/sys/utcb.h>
00015
00033
00034  enum l4_thread_group_ops
00035  {
00036      L4_THREAD_GROUP_ADD_OP = 2UL,
00037      L4_THREAD_GROUP_REMOVE_OP = 3UL,
00038  };
00039
00040  enum l4_thread_group_policy
00041  {
00042      L4_THREAD_GROUP_POLICY_STRICT_CORE_LOCAL = 0,
00043      L4_THREAD_GROUP_POLICY_SOFT_CORE_LOCAL = 1,
00044  };
00045
00052  L4_INLINE l4_msgtag_t
00053  l4_thread_group_add(l4_cap_idx_t tg,
00054                     l4_cap_idx_t thread) L4_NOTHROW;
00055
00059  L4_INLINE l4_msgtag_t
00060  l4_thread_group_add_u(l4_cap_idx_t tg,
00061                       l4_cap_idx_t thread,
00062                       l4_utcb_t *utcb) L4_NOTHROW;
00063
00070  L4_INLINE l4_msgtag_t
00071  l4_thread_group_remove(l4_cap_idx_t tg,
00072                        l4_cap_idx_t thread) L4_NOTHROW;
00073
00077  L4_INLINE l4_msgtag_t
00078  l4_thread_group_remove_u(l4_cap_idx_t tg,
00079                           l4_cap_idx_t thread,
00080                           l4_utcb_t *utcb) L4_NOTHROW;
00081
00082
00083  /* IMPLEMENTATION ----- */
00084
00085  #include <l4/sys/ipc.h>
00086
00087  L4_INLINE l4_msgtag_t
00088  l4_thread_group_add_u(l4_cap_idx_t tg,
00089                       l4_cap_idx_t thread,
00090                       l4_utcb_t *utcb) L4_NOTHROW
00091  {
00092      l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00093      v->mr[0] = L4_THREAD_GROUP_ADD_OP;
00094      v->mr[1] = l4_map_obj_control(0, 0);
00095      v->mr[2] = l4_obj_fpage(thread, 0, L4_CAP_FPAGE_RWS).raw;
00096      return l4_ipc_call(tg, utcb,
00097                        l4_msgtag(L4_PROTO_THREAD_GROUP, 1, 1, 0), L4_IPC_NEVER);
00098  }
00099
00100  L4_INLINE l4_msgtag_t
00101  l4_thread_group_remove_u(l4_cap_idx_t tg,
00102                           l4_cap_idx_t thread,
00103                           l4_utcb_t *utcb) L4_NOTHROW
00104  {
00105      l4_msg_regs_t *v = l4_utcb_mr_u(utcb);

```


This graph shows which files directly or indirectly include this file:



Namespaces

- namespace [L4](#)
L4 low-level kernel interface.

16.613.1 Detailed Description

Type information server template.

Definition in file [typeinfo_svr](#).

16.614 typeinfo_svr

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -- Mode: C++ --
00006 /*
00007  * (c) 2010 Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *     economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012
00013 #pragma once
00014
00015 #include <l4/sys/meta>
00016 #include <l4/sys/cxx/ipc_epiface>
00017
00018 namespace L4 { namespace Util {
00019
00020 template<typename KO, typename IOS>
00021 long handle_meta_request(IOS &ios)
00022 {
00023     using L4::IpC::Msg::dispatch_call;
00024     typedef L4::IpC::Detail::Meta_svr<KO> Msvr;
00025     typedef L4::Meta::Rpcs Rpcs;
00026     Msvr *svr = nullptr;
00027     l4_msgtag_t tag = dispatch_call<Rpcs>(svr, ios.utcb(), ios.tag(), 0);
00028     ios.set_ipc_params(tag);
00029     return tag.label();
00030 }
00031
00032 }}

```

16.615 types.h

```

00001  /*
00002  * (c) 2008-2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00003  *     economic rights: Technische Universität Dresden (Germany)
00004  * License: see LICENSE.spdx (in this directory or the directories above)
00005  */
00006  #pragma once
00007
00008  #include <l4/vbus/vbus_types.h>
00009
00014  enum l4io_iomem_flags_t
00015  {
00016      L4IO_MEM_NONCACHED = 0,
00017      L4IO_MEM_CACHED    = 1,
00018      L4IO_MEM_USE_MTRR   = 2,
00019      L4IO_MEM_ATTR_MASK = 0xf,
00020
00021      // combinations
00022      L4IO_MEM_WRITE_COMBINED = L4IO_MEM_USE_MTRR | L4IO_MEM_CACHED,
00023
00024
00027      L4IO_MEM_USE_RESERVED_AREA = 0x40 « 8,
00029      L4IO_MEM_EAGER_MAP         = 0x80 « 8,
00030  };
00031
00036  enum l4io_device_types_t {
00037      L4IO_DEVICE_INVALID = 0,
00038      L4IO_DEVICE_PCI,
00039      L4IO_DEVICE_USB,
00040      L4IO_DEVICE_OTHER,
00041      L4IO_DEVICE_ANY = ~0
00042  };
00043
00048  enum l4io_resource_types_t {
00049      L4IO_RESOURCE_INVALID = L4VBUS_RESOURCE_INVALID,
00050      L4IO_RESOURCE_IRQ     = L4VBUS_RESOURCE_IRQ,
00051      L4IO_RESOURCE_MEM     = L4VBUS_RESOURCE_MEM,
00052      L4IO_RESOURCE_PORT    = L4VBUS_RESOURCE_PORT,
00053      L4IO_RESOURCE_ANY     = ~0
00054  };
00055
00056
00057  typedef l4vbus_device_handle_t l4io_device_handle_t;
00058  typedef unsigned l4io_resource_handle_t;
00059
00067  typedef l4vbus_resource_t l4io_resource_t;
00068
00072  typedef l4vbus_device_t l4io_device_t;

```

16.616 l4/sys/types.h File Reference

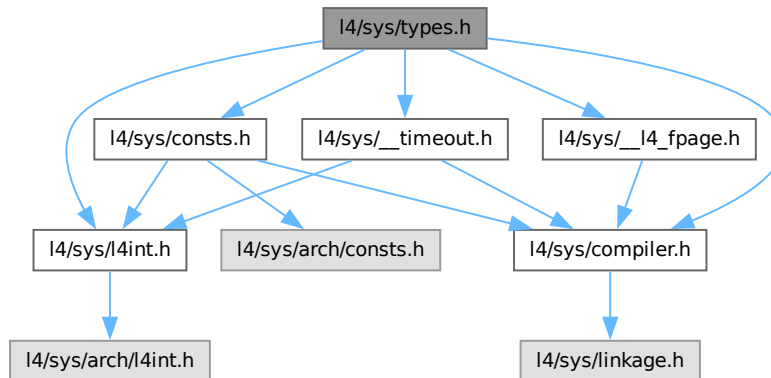
Common [L4](#) ABI Data Types.

```

#include <l4/sys/l4int.h>
#include <l4/sys/compiler.h>
#include <l4/sys/consts.h>
#include <l4/sys/__l4_fpage.h>
#include <l4/sys/__timeout.h>

```

Include dependency graph for types.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [l4_msgtag_t](#)
Message tag data structure.

Typedefs

- typedef [l4_int16_t](#) [l4_ret_t](#)
Return value of an IPC call as well as an RPC call.
- typedef [l4_int16_t](#) [l4_proto_t](#)
Prototype value for RPC calls.
- typedef unsigned long [l4_cap_idx_t](#)
Capability selector type.

Enumerations

- enum [L4_msgtag_protocol](#) {
[L4_PROTO_NONE](#) = 0 , [L4_PROTO_ALLOW_SYSCALL](#) = 1 , [L4_PROTO_PF_EXCEPTION](#) = 1 ,
[L4_PROTO_IRQ](#) = -1L ,
[L4_PROTO_PAGE_FAULT](#) = -2L , [L4_PROTO_EXCEPTION](#) = -5L , [L4_PROTO_SIGMA0](#) = -6L ,
[L4_PROTO_IO_PAGE_FAULT](#) = -8L ,
[L4_PROTO_THREAD_GROUP](#) = -9L , [L4_PROTO_KOBJECT](#) = -10L , [L4_PROTO_TASK](#) = -11L ,
[L4_PROTO_THREAD](#) = -12L ,
[L4_PROTO_LOG](#) = -13L , [L4_PROTO_SCHEDULER](#) = -14L , [L4_PROTO_FACTORY](#) = -15L ,
[L4_PROTO_VM](#) = -16L ,
[L4_PROTO_DMA_SPACE](#) = -17L , [L4_PROTO_IRQ_SENDER](#) = -18L , [L4_PROTO_SEMAPHORE](#) = -20L ,
[L4_PROTO_META](#) = -21L ,
[L4_PROTO_IOMMU](#) = -22L , [L4_PROTO_DEBUGGER](#) = -23L , [L4_PROTO_SMCCC](#) = -24L ,
[L4_PROTO_VCPU_CONTEXT](#) = -25L }

Message tag for IPC operations.

- enum [L4_msgtag_flags](#) { [L4_MSGTAG_ERROR](#) , [L4_MSGTAG_TRANSFER_FPU](#) , [L4_MSGTAG_SCHEDULE](#) , [L4_MSGTAG_PROPAGATE](#) = 0x4000 , [L4_MSGTAG_FLAGS](#) }

Flags for message tags.

Functions

- long [l4_msgtag_label](#) ([l4_msgtag_t](#) t) [L4_NOTHROW](#)
Get the protocol of tag.
- unsigned [l4_msgtag_words](#) ([l4_msgtag_t](#) t) [L4_NOTHROW](#)
Get the number of untyped words.
- unsigned [l4_msgtag_items](#) ([l4_msgtag_t](#) t) [L4_NOTHROW](#)
Get the number of typed items.
- unsigned [l4_msgtag_flags](#) ([l4_msgtag_t](#) t) [L4_NOTHROW](#)
Get the flags.
- unsigned [l4_msgtag_has_error](#) ([l4_msgtag_t](#) t) [L4_NOTHROW](#)
Test for error indicator flag.
- unsigned [l4_msgtag_is_page_fault](#) ([l4_msgtag_t](#) t) [L4_NOTHROW](#)
Test for page-fault protocol.
- unsigned [l4_msgtag_is_exception](#) ([l4_msgtag_t](#) t) [L4_NOTHROW](#)
Test for exception protocol.
- unsigned [l4_msgtag_is_sigma0](#) ([l4_msgtag_t](#) t) [L4_NOTHROW](#)
Test for sigma0 protocol.
- unsigned [l4_msgtag_is_io_page_fault](#) ([l4_msgtag_t](#) t) [L4_NOTHROW](#)
Test for IO-page-fault protocol.
- [l4_msgtag_t](#) [l4_msgtag](#) (long label, unsigned words, unsigned items, unsigned flags) [L4_NOTHROW](#)
Create a message tag from the specified values.
- unsigned [l4_is_invalid_cap](#) ([l4_cap_idx_t](#) c) [L4_NOTHROW](#)
Test if a capability selector is the invalid capability.
- unsigned [l4_is_valid_cap](#) ([l4_cap_idx_t](#) c) [L4_NOTHROW](#)
Test if a capability selector is a valid selector.
- unsigned [l4_capability_equal](#) ([l4_cap_idx_t](#) c1, [l4_cap_idx_t](#) c2) [L4_NOTHROW](#)
Test if the capability indices of two capability selectors are equal.
- [l4_cap_idx_t](#) [l4_capability_next](#) ([l4_cap_idx_t](#) c) [L4_NOTHROW](#)
Get the next capability selector after c.

16.616.1 Detailed Description

Common [L4](#) ABI Data Types.

Definition in file [types.h](#).

16.616.2 Typedef Documentation

16.616.2.1 l4_proto_t

```
typedef l4\_int16\_t l4\_proto\_t
```

Prototype value for RPC calls.

This type must not be larger than the label part of [l4_msgtag_t](#).

Definition at line 35 of file [types.h](#).

16.616.2.2 l4_ret_t

```
typedef l4_int16_t l4_ret_t
```

Return value of an IPC call as well as an RPC call.

This type must not be larger than the label part of [l4_msgtag_t](#).

Definition at line 28 of file [types.h](#).

16.616.3 Function Documentation

16.616.3.1 l4_capability_next()

```
l4_cap_idx_t l4_capability_next (
    l4_cap_idx_t c) [inline]
```

Get the next capability selector after *c*.

Parameters

| | |
|----------|--|
| <i>c</i> | The capability selector for which the next selector shall be computed. |
|----------|--|

Returns

The next capability selector after *c*.

Definition at line 478 of file [types.h](#).

References [L4_CAP_OFFSET](#), [L4_INLINE](#), and [L4_NOTHROW](#).

16.617 types.h

[Go to the documentation of this file.](#)

```
00001 /*****
00002  */
00003  * (c) 2008-2013 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  * Alexander Warg <warg@os.inf.tu-dresden.de>,
00005  * Björn Döbel <doebel@os.inf.tu-dresden.de>,
00006  * Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00007  * economic rights: Technische Universität Dresden (Germany)
00008  *
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011 /*****
00012  #pragma once
00013  */
00014 #include <l4/sys/l4int.h>
00015 #include <l4/sys/compiler.h>
00016 #include <l4/sys/consts.h>
00017
00018 typedef l4_int16_t l4_ret_t;
00019
00020 typedef l4_int16_t l4_proto_t;
00021
00022
00023 enum l4_msgtag_protocol
```

```

00053 {
00054     L4_PROTO_NONE = 0,
00055     L4_PROTO_ALLOW_SYSCALL = 1,
00056     L4_PROTO_PF_EXCEPTION = 1,
00057
00058     L4_PROTO_IRQ = -1L,
00059     L4_PROTO_PAGE_FAULT = -2L,
00060     // -3L unused
00061     // -4L unused
00062     L4_PROTO_EXCEPTION = -5L,
00063     L4_PROTO_SIGMA0 = -6L,
00064     L4_PROTO_IO_PAGE_FAULT = -8L,
00065     L4_PROTO_THREAD_GROUP = -9L,
00066     L4_PROTO_KOBJECT = -10L,
00067     L4_PROTO_TASK = -11L,
00068     L4_PROTO_THREAD = -12L,
00069     L4_PROTO_LOG = -13L,
00070     L4_PROTO_SCHEDULER = -14L,
00071     L4_PROTO_FACTORY = -15L,
00072     L4_PROTO_VM = -16L,
00073     L4_PROTO_DMA_SPACE = -17L,
00074     L4_PROTO_IRQ_SENDER = -18L,
00075     // -19L unusable, used for L4::PROTO_EMPTY
00076     L4_PROTO_SEMAPHORE = -20L,
00077     L4_PROTO_META = -21L,
00078     L4_PROTO_IOMMU = -22L,
00079     L4_PROTO_DEBUGGER = -23L,
00080     L4_PROTO_SMCCC = -24L,
00081     L4_PROTO_VCPU_CONTEXT = -25L,
00082 };
00083
00084 enum L4_varg_type
00085 {
00086     L4_VARG_TYPE_NIL = 0x00,
00087     L4_VARG_TYPE_UMWORD = 0x01,
00088     L4_VARG_TYPE_MWORD = 0x81,
00089     L4_VARG_TYPE_STRING = 0x02,
00090     L4_VARG_TYPE_FPAGE = 0x03,
00091
00092     L4_VARG_TYPE_SIGN = 0x80,
00093 };
00094
00095
00100 enum L4_msgtag_flags
00101 {
00102     // flags for received IPC
00107     L4_MSGTAG_ERROR = 0x8000,
00108
00109     // flags for sending IPC
00119     L4_MSGTAG_TRANSFER_FPU = 0x1000,
00128     L4_MSGTAG_SCHEDULE = 0x2000,
00141     L4_MSGTAG_PROPAGATE = 0x4000,
00142
00147     L4_MSGTAG_FLAGS = 0xf000,
00148 };
00149
00150 typedef struct l4_msgtag_t l4_msgtag_t;
00151
00160 L4_INLINE long l4_msgtag_label(l4_msgtag_t t) L4_NOTHROW;
00161
00170 L4_INLINE unsigned l4_msgtag_words(l4_msgtag_t t) L4_NOTHROW;
00171
00180 L4_INLINE unsigned l4_msgtag_items(l4_msgtag_t t) L4_NOTHROW;
00181
00192 L4_INLINE unsigned l4_msgtag_flags(l4_msgtag_t t) L4_NOTHROW;
00193
00206 L4_INLINE unsigned l4_msgtag_has_error(l4_msgtag_t t) L4_NOTHROW;
00207
00216 L4_INLINE unsigned l4_msgtag_is_page_fault(l4_msgtag_t t) L4_NOTHROW;
00217
00226 L4_INLINE unsigned l4_msgtag_is_exception(l4_msgtag_t t) L4_NOTHROW;
00227
00236 L4_INLINE unsigned l4_msgtag_is_sigma0(l4_msgtag_t t) L4_NOTHROW;
00237
00246 L4_INLINE unsigned l4_msgtag_is_io_page_fault(l4_msgtag_t t) L4_NOTHROW;
00247
00248
00265 struct l4_msgtag_t
00266 {
00267     l4_mword_t raw;
00268 #ifdef __cplusplus
00270     long label() const L4_NOTHROW { return l4_msgtag_label(*this); }
00272     void label(long v) L4_NOTHROW { raw = (raw & 0xffff) | ((l4_umword_t)v << 16); }
00274     unsigned words() const L4_NOTHROW { return l4_msgtag_words(*this); }
00276     unsigned items() const L4_NOTHROW { return l4_msgtag_items(*this); }
00283     unsigned flags() const L4_NOTHROW { return l4_msgtag_flags(*this); }
00285     bool is_page_fault() const L4_NOTHROW

```



```

00286     { return l4_msgtag_is_page_fault(*this); }
00288     bool is_exception() const L4_NOTHROW
00289     { return l4_msgtag_is_exception(*this); }
00291     bool is_sigma0() const L4_NOTHROW { return l4_msgtag_is_sigma0(*this); }
00293     bool is_io_page_fault() const L4_NOTHROW
00294     { return l4_msgtag_is_io_page_fault(*this); }
00299     bool has_error() const L4_NOTHROW { return l4_msgtag_has_error(*this); }
00300 #endif
00301 };
00302
00303 // The following declaration should be _after_ the definition of struct
00304 // l4_msgtag_t. Circumvents a clang warning about a *potentially*
00305 // incompatibility with C if the type is incomplete when the function is
00306 // declared.
00307
00319 L4_INLINE l4_msgtag_t l4_msgtag(long label, unsigned words, unsigned items,
00320                                unsigned flags) L4_NOTHROW;
00321
00322
00323
00324
00357 typedef unsigned long l4_cap_idx_t;
00358
00368 L4_INLINE unsigned l4_is_invalid_cap(l4_cap_idx_t c) L4_NOTHROW;
00369
00379 L4_INLINE unsigned l4_is_valid_cap(l4_cap_idx_t c) L4_NOTHROW;
00380
00394 L4_INLINE unsigned l4_capability_equal(l4_cap_idx_t c1, l4_cap_idx_t c2) L4_NOTHROW;
00395
00404 L4_INLINE l4_cap_idx_t l4_capability_next(l4_cap_idx_t c) L4_NOTHROW;
00405
00406 /* ***** Implementation ***** */
00407 /* Implementation */
00408
00409 L4_INLINE unsigned
00410 l4_is_invalid_cap(l4_cap_idx_t c) L4_NOTHROW
00411 { return c & L4_INVALID_CAP_BIT; }
00412
00413 L4_INLINE unsigned
00414 l4_is_valid_cap(l4_cap_idx_t c) L4_NOTHROW
00415 { return !(c & L4_INVALID_CAP_BIT); }
00416
00417 L4_INLINE unsigned
00418 l4_capability_equal(l4_cap_idx_t c1, l4_cap_idx_t c2) L4_NOTHROW
00419 { return (c1 >> L4_CAP_SHIFT) == (c2 >> L4_CAP_SHIFT); }
00420
00421
00425 L4_INLINE
00426 l4_msgtag_t l4_msgtag(long label, unsigned words, unsigned items,
00427                      unsigned flags) L4_NOTHROW
00428 {
00429     return (l4_msgtag_t){ (l4_mword_t)((l4_umword_t)label << 16)
00430                          | (l4_mword_t)(words & 0x3f)
00431                          | (l4_mword_t)((items & 0x3f) << 6)
00432                          | (l4_mword_t)(flags & 0xf000)};
00433 }
00434
00435
00436
00437 L4_INLINE
00438 long l4_msgtag_label(l4_msgtag_t t) L4_NOTHROW
00439 {
00440     #if defined(__cplusplus) && (__cplusplus >= 202002L)
00441         return t.raw >> 16;
00442     #else
00443         return t.raw < 0 ? ~(~t.raw >> 16) : t.raw >> 16;
00444     #endif
00445 }
00446
00447 L4_INLINE
00448 unsigned l4_msgtag_words(l4_msgtag_t t) L4_NOTHROW
00449 { return t.raw & 0x3f; }
00450
00451 L4_INLINE
00452 unsigned l4_msgtag_items(l4_msgtag_t t) L4_NOTHROW
00453 { return (t.raw >> 6) & 0x3f; }
00454
00455 L4_INLINE
00456 unsigned l4_msgtag_flags(l4_msgtag_t t) L4_NOTHROW
00457 { return t.raw & 0xf000; }
00458
00459
00460 L4_INLINE
00461 unsigned l4_msgtag_has_error(l4_msgtag_t t) L4_NOTHROW
00462 { return t.raw & L4_MSGTAG_ERROR; }
00463
00464
00465

```

```

00466 L4_INLINE unsigned l4_msgtag_is_page_fault(l4_msgtag_t t) L4_NOTHROW
00467 { return l4_msgtag_label(t) == L4_PROTO_PAGE_FAULT; }
00468
00469 L4_INLINE unsigned l4_msgtag_is_exception(l4_msgtag_t t) L4_NOTHROW
00470 { return l4_msgtag_label(t) == L4_PROTO_EXCEPTION; }
00471
00472 L4_INLINE unsigned l4_msgtag_is_sigma0(l4_msgtag_t t) L4_NOTHROW
00473 { return l4_msgtag_label(t) == L4_PROTO_SIGMA0; }
00474
00475 L4_INLINE unsigned l4_msgtag_is_io_page_fault(l4_msgtag_t t) L4_NOTHROW
00476 { return l4_msgtag_label(t) == L4_PROTO_IO_PAGE_FAULT; }
00477
00478 L4_INLINE l4_cap_idx_t l4_capability_next(l4_cap_idx_t c) L4_NOTHROW
00479 { return c + L4_CAP_OFFSET; }
00480
00481 #include <l4/sys/__l4_fpage.h>
00482 #include <l4/sys/__timeout.h>

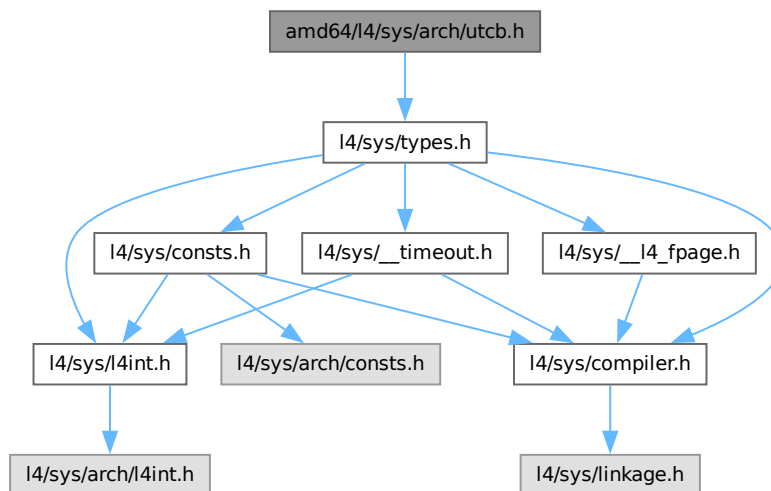
```

16.618 amd64/l4/sys/arch/utcb.h File Reference

UTCB definitions for AMD64.

```
#include <l4/sys/types.h>
```

Include dependency graph for utcb.h:



Data Structures

- struct `l4_exc_regs_t`
UTCB structure for exceptions.

Functions

- `l4_umword_t l4_utcb_exc_pc(l4_exc_regs_t const *u) L4_NOTHROW`
Access function to get the program counter of the exception state.
- `void l4_utcb_exc_pc_set(l4_exc_regs_t *u, l4_addr_t pc) L4_NOTHROW`
Set the program counter register in the exception state.

- [l4_umword_t l4_utcb_exc_typeval \(l4_exc_regs_t const *u\) L4_NOTHROW](#)
Get the value out of an exception UTCB that describes the type of exception.
- [int l4_utcb_exc_is_pf \(l4_exc_regs_t const *u\) L4_NOTHROW](#)
Check whether an exception IPC is a page fault.
- [l4_addr_t l4_utcb_exc_pfa \(l4_exc_regs_t const *u\) L4_NOTHROW](#)
Function to get the L4 style page fault address out of an exception.
- [int l4_utcb_exc_is_ex_regs_exception \(l4_exc_regs_t const *u\) L4_NOTHROW](#)
Check whether an exception IPC was triggered via [l4_thread_ex_regs\(\)](#).

16.618.1 Detailed Description

UTCB definitions for AMD64.

Definition in file [utcb.h](#).

16.619 utcb.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013 /*****
00014  *ifndef __L4_SYS__INCLUDE__ARCH_AMD64__UTCB_H__
00015  *define __L4_SYS__INCLUDE__ARCH_AMD64__UTCB_H__
00016  *
00017  *include <l4/sys/types.h>
00018  *
00023
00028 struct l4_exc_regs_t
00029 {
00030     l4_umword_t r15;
00031     l4_umword_t r14;
00032     l4_umword_t r13;
00033     l4_umword_t r12;
00034     l4_umword_t r11;
00035     l4_umword_t r10;
00036     l4_umword_t r9;
00037     l4_umword_t r8;
00038     l4_umword_t rdi;
00039     l4_umword_t rsi;
00040     l4_umword_t rbp;
00041     l4_umword_t pfa;
00042     l4_umword_t rbx;
00043     l4_umword_t rdx;
00044     l4_umword_t rcx;
00045     l4_umword_t rax;
00046
00047     l4_umword_t trapno;
00048     l4_umword_t err;
00049     l4_umword_t ip;
00050     l4_umword_t dummy1;
00051     l4_umword_t flags;
00052     l4_umword_t sp;
00053     l4_umword_t ss;
00054     l4_umword_t fs_base;
00055     l4_umword_t gs_base;
00056     l4_uint16_t ds, es, fs, gs;
00057 };
00058
00059 #define L4_UTCB_EXCEPTION_REGS_SIZE    (sizeof(l4_exc_regs_t) / sizeof(l4_umword_t))
00060 #define L4_UTCB_GENERIC_DATA_SIZE      63
00061 #define L4_UTCB_GENERIC_BUFFERS_SIZE   58
00062
00063 #define L4_UTCB_MSG_REGS_OFFSET        0
00064 #define L4_UTCB_BUF_REGS_OFFSET        (64 * sizeof(l4_umword_t))

```

```

00065 #define L4_UTCB_THREAD_REGS_OFFSET      (123 * sizeof(l4_umword_t))
00066
00067 #define L4_UTCB_INHERIT_FPU                (1UL < 24)
00068 #define L4_UTCB_OFFSET                    1024
00069
00070 /*
00071  * =====
00072  * Implementations.
00073  */
00074
00075 L4_INLINE l4_utcb_t *l4_utcb_direct(void) L4_NOTHROW
00076 {
00077     l4_utcb_t *res;
00078     __asm__ ( "mov %%gs:0, %0 \n" : "=r"(res));
00079     return res;
00080 }
00081
00082 L4_INLINE l4_umword_t l4_utcb_exc_pc(l4_exc_regs_t const *u) L4_NOTHROW
00083 {
00084     return u->ip;
00085 }
00086
00087 L4_INLINE void l4_utcb_exc_pc_set(l4_exc_regs_t *u, l4_addr_t pc) L4_NOTHROW
00088 {
00089     u->ip = pc;
00090 }
00091
00092 L4_INLINE l4_umword_t l4_utcb_exc_typeval(l4_exc_regs_t const *u) L4_NOTHROW
00093 {
00094     return u->trapno;
00095 }
00096
00097 L4_INLINE int l4_utcb_exc_is_pf(l4_exc_regs_t const *u) L4_NOTHROW
00098 {
00099     return u->trapno == 14;
00100 }
00101
00102 L4_INLINE l4_addr_t l4_utcb_exc_pfa(l4_exc_regs_t const *u) L4_NOTHROW
00103 {
00104     return (u->pfa & ~7UL) | (u->err & 2);
00105 }
00106
00107 L4_INLINE int l4_utcb_exc_is_ex_regs_exception(l4_exc_regs_t const *u) L4_NOTHROW
00108 {
00109     return l4_utcb_exc_typeval(u) == 0xff;
00110 }
00111
00112 #endif /* ! __L4_SYS__INCLUDE__ARCH_AMD64__UTCB_H__ */

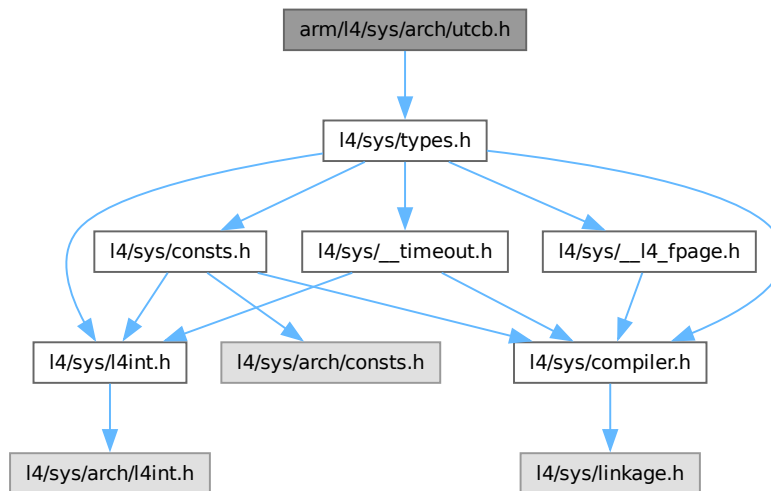
```

16.620 arm/l4/sys/arch/utcb.h File Reference

UTCB definitions for ARM.

```
#include <l4/sys/types.h>
```

Include dependency graph for utcb.h:



Data Structures

- struct [l4_exc_regs_t](#)
UTCB structure for exceptions.

Functions

- [l4_umword_t l4_utcb_exc_pc](#) ([l4_exc_regs_t](#) const *u) [L4_NOTHROW](#)
Access function to get the program counter of the exception state.
- void [l4_utcb_exc_pc_set](#) ([l4_exc_regs_t](#) *u, [l4_addr_t](#) pc) [L4_NOTHROW](#)
Set the program counter register in the exception state.
- [l4_umword_t l4_utcb_exc_typeval](#) ([l4_exc_regs_t](#) const *u) [L4_NOTHROW](#)
Get the value out of an exception UTCB that describes the type of exception.
- int [l4_utcb_exc_is_pf](#) ([l4_exc_regs_t](#) const *u) [L4_NOTHROW](#)
Check whether an exception IPC is a page fault.
- [l4_addr_t l4_utcb_exc_pfa](#) ([l4_exc_regs_t](#) const *u) [L4_NOTHROW](#)
Function to get the L4 style page fault address out of an exception.
- int [l4_utcb_exc_is_ex_regs_exception](#) ([l4_exc_regs_t](#) const *u) [L4_NOTHROW](#)
Check whether an exception IPC was triggered via [l4_thread_ex_regs\(\)](#).

16.620.1 Detailed Description

UTCB definitions for ARM.

Definition in file [utcb.h](#).

16.621 utcb.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013 #ifndef __L4_SYS__INCLUDE__ARCH_ARM__UTCB_H__
00014 #define __L4_SYS__INCLUDE__ARCH_ARM__UTCB_H__
00015
00016 #include <l4/sys/types.h>
00017
00022
00027 struct l4_exc_regs_t
00028 {
00029     l4_umword_t pfa;
00030     l4_umword_t err;
00031
00032     l4_umword_t r[13];
00033     l4_umword_t sp;
00034     l4_umword_t ulr;
00035     l4_umword_t _dummy1;
00036     union { l4_umword_t ip; l4_umword_t pc; };
00037     l4_umword_t cpsr;
00038     l4_umword_t tpidruro;
00039     l4_umword_t tpidrurw;
00040 };
00041
00042 #define L4_UTCB_EXCEPTION_REGS_SIZE    (sizeof(l4_exc_regs_t) / sizeof(l4_umword_t))
00043 #define L4_UTCB_GENERIC_DATA_SIZE      63
00044 #define L4_UTCB_GENERIC_BUFFERS_SIZE   58
00045
00046 #define L4_UTCB_MSG_REGS_OFFSET         0
00047 #define L4_UTCB_BUF_REGS_OFFSET         (64 * sizeof(l4_umword_t))
00048 #define L4_UTCB_THREAD_REGS_OFFSET      (123 * sizeof(l4_umword_t))
00049
00050 #define L4_UTCB_INHERIT_FPU              (1UL << 24)
00051
00052 #define L4_UTCB_OFFSET                   512
00053
00054 /*
00055  * =====
00056  * Implementations.
00057  */
00058
00059 #ifdef __GNUC__
00060 L4_INLINE l4_utcb_t *l4_utcb_direct(void) L4_NOTHROW
00061 {
00062     # if defined(__ARM_ARCH) && __ARM_ARCH >= 7
00063         l4_utcb_t *utcb;
00064         __asm__ ("mrc p15, 0, %0, c13, c0, 2" : "=r" (utcb)); // TPIDRURW
00065     #else
00066         register l4_utcb_t *utcb __asm__ ("r0");
00067         __asm__ ("mov lr, pc\n"
00068                 "mvn pc, #0xff\n"
00069                 : "=r" (utcb) : : "lr"); // write 0xffffffff00 to pc
00070     #endif
00071     return utcb;
00072 }
00073 #endif
00074
00075 L4_INLINE l4_umword_t l4_utcb_exc_pc(l4_exc_regs_t const *u) L4_NOTHROW
00076 {
00077     return u->pc;
00078 }
00079
00080 L4_INLINE void l4_utcb_exc_pc_set(l4_exc_regs_t *u, l4_addr_t pc) L4_NOTHROW
00081 {
00082     u->pc = pc;
00083 }
00084
00085 L4_INLINE l4_umword_t l4_utcb_exc_typeval(l4_exc_regs_t const *u) L4_NOTHROW
00086 {
00087     return u->err >> 26;
00088 }
00089
00090 L4_INLINE int l4_utcb_exc_is_pf(l4_exc_regs_t const *u) L4_NOTHROW
00091 {
00092     return ((u->err >> 26) & 0x30) == 0x20;
00093 }
00094

```

```

00095 L4_INLINE l4_addr_t l4_utcb_exc_pfa(l4_exc_regs_t const *u) L4_NOTHROW
00096 {
00097     return (u->pfa & ~7UL) | ((u->err >> 5) & 2);
00098 }
00099
00100 L4_INLINE int l4_utcb_exc_is_ex_regs_exception(l4_exc_regs_t const *u) L4_NOTHROW
00101 {
00102     return l4_utcb_exc_typeval(u) == 0x3e;
00103 }
00104
00105 #endif /* ! __L4_SYS__INCLUDE__ARCH_ARM__UTCB_H__ */

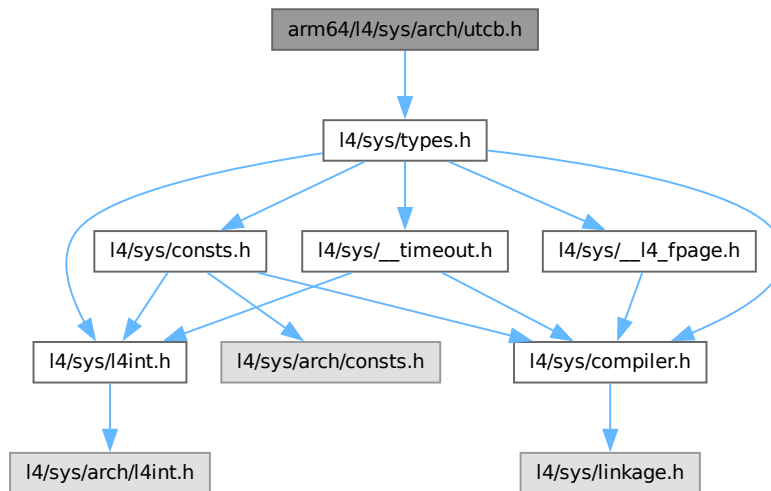
```

16.622 arm64/l4/sys/arch/utcb.h File Reference

UTCB definitions for ARM64.

```
#include <l4/sys/types.h>
```

Include dependency graph for utcb.h:



Data Structures

- struct `l4_exc_regs_t`
UTCB structure for exceptions.

Functions

- `l4_umword_t l4_utcb_exc_pc(l4_exc_regs_t const *u) L4_NOTHROW`
Access function to get the program counter of the exception state.
- `void l4_utcb_exc_pc_set(l4_exc_regs_t *u, l4_addr_t pc) L4_NOTHROW`
Set the program counter register in the exception state.
- `l4_umword_t l4_utcb_exc_typeval(l4_exc_regs_t const *u) L4_NOTHROW`
Get the value out of an exception UTCB that describes the type of exception.
- `int l4_utcb_exc_is_pf(l4_exc_regs_t const *u) L4_NOTHROW`
Check whether an exception IPC is a page fault.
- `l4_addr_t l4_utcb_exc_pfa(l4_exc_regs_t const *u) L4_NOTHROW`
Function to get the L4 style page fault address out of an exception.
- `int l4_utcb_exc_is_ex_regs_exception(l4_exc_regs_t const *u) L4_NOTHROW`
Check whether an exception IPC was triggered via `l4_thread_ex_regs()`.

16.622.1 Detailed Description

UTCB definitions for ARM64.

Definition in file [utcb.h](#).

16.623 utcb.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013 #ifndef __L4_SYS__INCLUDE__ARCH_ARM64__UTCB_H__
00014 #define __L4_SYS__INCLUDE__ARCH_ARM64__UTCB_H__
00015
00016 #include <l4/sys/types.h>
00017
00022
00027 struct l4_exc_regs_t
00028 {
00029     l4_umword_t eret_work;
00030     l4_umword_t r[31];
00031     l4_umword_t reserved;
00032     l4_umword_t err;
00033
00034     l4_umword_t pfa;
00035     l4_umword_t sp;
00036     union { l4_umword_t ip; l4_umword_t pc; }; /* aliases for PC */
00037     union { l4_umword_t flags; l4_umword_t pstate; }; /* aliases for PSTATE (PSR) */
00038     l4_umword_t tpidrro;
00039     l4_umword_t tpidrurw;
00040 };
00041
00042 #define L4_UTCB_EXCEPTION_REGS_SIZE    (sizeof(l4_exc_regs_t) / sizeof(l4_umword_t))
00043 #define L4_UTCB_GENERIC_DATA_SIZE     63
00044 #define L4_UTCB_GENERIC_BUFFERS_SIZE  58
00045
00046 #define L4_UTCB_MSG_REGS_OFFSET        0
00047 #define L4_UTCB_BUF_REGS_OFFSET        (64 * sizeof(l4_umword_t))
00048 #define L4_UTCB_THREAD_REGS_OFFSET     (123 * sizeof(l4_umword_t))
00049
00050 #define L4_UTCB_INHERIT_FPU             (1UL << 24)
00051
00052 #define L4_UTCB_OFFSET                  1024
00053
00054 /*
00055  * =====
00056  * Implementations.
00057  */
00058
00059 #ifndef __GNUC__
00060 L4_INLINE l4_utcb_t *l4_utcb_direct(void) L4_NOTHROW
00061 {
00062     l4_utcb_t *utcb;
00063     __asm__ ("mrs %0, TPIDRRO_EL0" : "=r" (utcb));
00064     return utcb;
00065 }
00066 #endif
00067
00068 L4_INLINE l4_umword_t l4_utcb_exc_pc(l4_exc_regs_t const *u) L4_NOTHROW
00069 {
00070     return u->pc;
00071 }
00072
00073 L4_INLINE void l4_utcb_exc_pc_set(l4_exc_regs_t *u, l4_addr_t pc) L4_NOTHROW
00074 {
00075     u->pc = pc;
00076 }
00077
00078 L4_INLINE l4_umword_t l4_utcb_exc_typeval(l4_exc_regs_t const *u) L4_NOTHROW
00079 {
00080     return u->err >> 26;
00081 }

```



```

00082
00083 L4_INLINE int l4_utcb_exc_is_pf(l4_exc_regs_t const *u) L4_NOTHROW
00084 {
00085     return ((u->err >> 26) & 0x30) == 0x20;
00086 }
00087
00088 L4_INLINE l4_addr_t l4_utcb_exc_pfa(l4_exc_regs_t const *u) L4_NOTHROW
00089 {
00090     return (u->pfa & ~7UL) | ((u->err >> 5) & 2);
00091 }
00092
00093 L4_INLINE int l4_utcb_exc_is_ex_regs_exception(l4_exc_regs_t const *u) L4_NOTHROW
00094 {
00095     return (u->err >> 26) == 0x3e;
00096 }
00097
00098 #endif /* ! __L4_SYS__INCLUDE__ARCH_ARM64__UTCB_H__ */

```

16.624 l4/sys/utcb.h File Reference

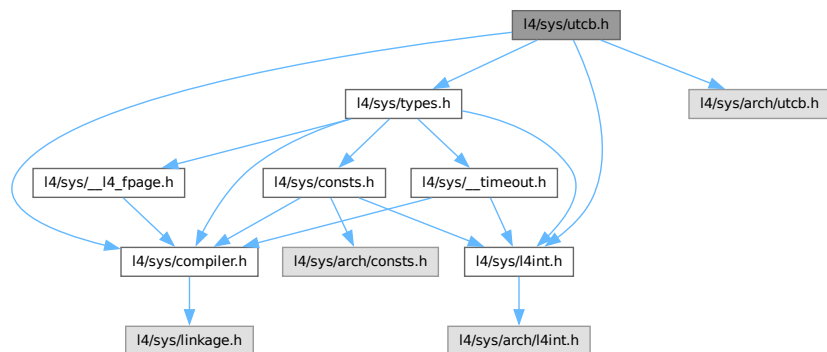
UTCB definitions.

```

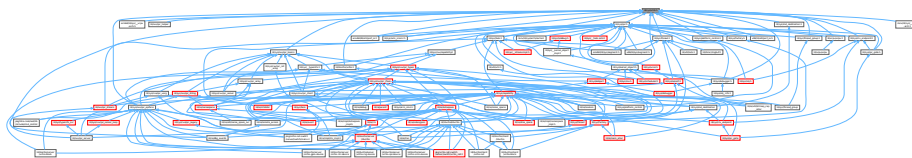
#include <l4/sys/types.h>
#include <l4/sys/compiler.h>
#include <l4/sys/l4int.h>
#include <l4/sys/arch/utcb.h>

```

Include dependency graph for utcb.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- union [l4_msg_regs_t](#)
Encapsulation of the message-register block in the UTCB.
- struct [l4_buf_regs_t](#)
Encapsulation of the buffer-registers block in the UTCB.
- struct [l4_thread_regs_t](#)
Encapsulation of the thread-control-register block of the UTCB.

Typedefs

- typedef struct [l4_utcb_t](#) [l4_utcb_t](#)
Opaque type for the UTCB.
- typedef union [l4_msg_regs_t](#) [l4_msg_regs_t](#)
Encapsulation of the message-register block in the UTCB.
- typedef struct [l4_buf_regs_t](#) [l4_buf_regs_t](#)
Encapsulation of the buffer-registers block in the UTCB.
- typedef struct [l4_thread_regs_t](#) [l4_thread_regs_t](#)
Encapsulation of the thread-control-register block of the UTCB.

Functions

- [l4_umword_t](#) [l4_utcb_exc_pc](#) ([l4_exc_regs_t](#) const *u) [L4_NOTHROW](#) [L4_PURE](#)
Access function to get the program counter of the exception state.
- void [l4_utcb_exc_pc_set](#) ([l4_exc_regs_t](#) *u, [l4_addr_t](#) pc) [L4_NOTHROW](#)
Set the program counter register in the exception state.
- unsigned long [l4_utcb_exc_typeval](#) ([l4_exc_regs_t](#) const *u) [L4_NOTHROW](#) [L4_PURE](#)
Get the value out of an exception UTCB that describes the type of exception.
- int [l4_utcb_exc_is_pf](#) ([l4_exc_regs_t](#) const *u) [L4_NOTHROW](#) [L4_PURE](#)
Check whether an exception IPC is a page fault.
- [l4_addr_t](#) [l4_utcb_exc_pfa](#) ([l4_exc_regs_t](#) const *u) [L4_NOTHROW](#) [L4_PURE](#)
Function to get the L4 style page fault address out of an exception.
- int [l4_utcb_exc_is_ex_regs_exception](#) ([l4_exc_regs_t](#) const *u) [L4_NOTHROW](#) [L4_PURE](#)
Check whether an exception IPC was triggered via [l4_thread_ex_regs\(\)](#).
- [l4_utcb_t](#) * [l4_utcb](#) (void) [L4_NOTHROW](#) [L4_PURE](#)
Get the UTCB address.
- [l4_msg_regs_t](#) * [l4_utcb_mr](#) (void) [L4_NOTHROW](#) [L4_PURE](#)
Get the message-register block of a UTCB.
- [l4_buf_regs_t](#) * [l4_utcb_br](#) (void) [L4_NOTHROW](#) [L4_PURE](#)
Get the buffer-register block of a UTCB.
- [l4_thread_regs_t](#) * [l4_utcb_tcr](#) (void) [L4_NOTHROW](#) [L4_PURE](#)
Get the thread-control-register block of a UTCB.
- [l4_exc_regs_t](#) * [l4_utcb_exc](#) (void) [L4_NOTHROW](#) [L4_PURE](#)
Get the message-register block of a UTCB (for an exception IPC).
- void [l4_utcb_inherit_fpu](#) (int switch_on) [L4_NOTHROW](#)
Enable or disable inheritance of FPU state to receiver.
- [l4_umword_t](#) [l4_bdr](#) ([l4_umword_t](#) mem, [l4_umword_t](#) io, [l4_umword_t](#) obj, [l4_umword_t](#) flags) [L4_NOTHROW](#)
Create a buffer descriptor.
- [l4_timeout_s](#) [l4_timeout_abs](#) ([l4_kernel_clock_t](#) pint, int br) [L4_NOTHROW](#)
Set an absolute timeout.
- unsigned [l4_utcb_mr64_idx](#) (unsigned idx) [L4_NOTHROW](#)
Get index into 64bit message registers alias from native-sized index.

16.624.1 Detailed Description

UTCB definitions.

Definition in file [utcb.h](#).

16.625 utcb.h

[Go to the documentation of this file.](#)

```

00001 /*****
00002  */
00003 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004 *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00005 *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00006 *      economic rights: Technische Universität Dresden (Germany)
00007 *
00008 * License: see LICENSE.spdx (in this directory or the directories above)
00009 */
00010 /*****
00011  */
00012 #ifndef _L4_SYS_UTCB_H
00013 #define _L4_SYS_UTCB_H
00014
00015 #include <l4/sys/types.h>
00016 #include <l4/sys/compiler.h>
00017 #include <l4/sys/l4int.h>
00018
00019 typedef struct l4_utcb_t l4_utcb_t;
00020
00021 typedef struct l4_exc_regs_t l4_exc_regs_t;
00022
00023 L4_INLINE l4_utcb_t *l4_utcb_direct(void) L4_NOTHROW L4_PURE;
00024
00025 L4_INLINE l4_umword_t l4_utcb_exc_pc(l4_exc_regs_t const *u) L4_NOTHROW L4_PURE;
00026
00027 L4_INLINE void l4_utcb_exc_pc_set(l4_exc_regs_t *u, l4_addr_t pc) L4_NOTHROW;
00028
00029 L4_INLINE unsigned long l4_utcb_exc_typeval(l4_exc_regs_t const *u) L4_NOTHROW L4_PURE;
00030
00031 L4_INLINE int l4_utcb_exc_is_pf(l4_exc_regs_t const *u) L4_NOTHROW L4_PURE;
00032
00033 L4_INLINE l4_addr_t l4_utcb_exc_pfa(l4_exc_regs_t const *u) L4_NOTHROW L4_PURE;
00034
00035 L4_INLINE int l4_utcb_exc_is_ex_regs_exception(l4_exc_regs_t const *u) L4_NOTHROW L4_PURE;
00036
00037 #include <l4/sys/arch/utcb.h>
00038
00039 typedef union l4_msg_regs_t
00040 {
00041     l4_umword_t mr[L4_UTCB_GENERIC_DATA_SIZE];
00042     l4_uint64_t mr64[L4_UTCB_GENERIC_DATA_SIZE / (sizeof(l4_uint64_t)/sizeof(l4_umword_t))];
00043 } l4_msg_regs_t;
00044
00045 typedef struct l4_buf_regs_t
00046 {
00047     l4_umword_t bdr;
00048
00049     l4_umword_t br[L4_UTCB_GENERIC_BUFFERS_SIZE];
00050 } l4_buf_regs_t;
00051
00052 typedef struct l4_thread_regs_t
00053 {
00054     l4_umword_t error;
00055
00056     l4_umword_t free_marker;
00057
00058     l4_umword_t user[3];
00059 } l4_thread_regs_t;
00060
00061 L4_BEGIN_DECLS
00062
00063 L4_CV l4_utcb_t *l4_utcb_wrap(void) L4_NOTHROW L4_PURE;
00064
00065 L4_INLINE l4_utcb_t *l4_utcb(void) L4_NOTHROW L4_PURE;
00066
00067 L4_INLINE l4_msg_regs_t *l4_utcb_mr(void) L4_NOTHROW L4_PURE;
00068
00069 L4_INLINE l4_msg_regs_t *l4_utcb_mr_u(l4_utcb_t *u) L4_NOTHROW L4_PURE;
00070
00071 L4_INLINE l4_buf_regs_t *l4_utcb_br(void) L4_NOTHROW L4_PURE;
00072
00073 L4_INLINE l4_buf_regs_t *l4_utcb_br_u(l4_utcb_t *u) L4_NOTHROW L4_PURE;
00074
00075 L4_INLINE l4_thread_regs_t *l4_utcb_tcr(void) L4_NOTHROW L4_PURE;
00076
00077 L4_INLINE l4_thread_regs_t *l4_utcb_tcr_u(l4_utcb_t *u) L4_NOTHROW L4_PURE;
00078
00079 L4_INLINE l4_exc_regs_t *l4_utcb_exc(void) L4_NOTHROW L4_PURE;
00080
00081

```

```

00269 L4_INLINE l4_exc_regs_t *l4_utcb_exc_u(l4_utcb_t *u) L4_NOTHROW L4_PURE;
00270
00275 L4_INLINE void l4_utcb_inherit_fpu(int switch_on) L4_NOTHROW;
00276
00280 L4_INLINE void l4_utcb_inherit_fpu_u(l4_utcb_t *u, int switch_on) L4_NOTHROW;
00281
00292 L4_INLINE l4_umword_t l4_bdr(l4_umword_t mem, l4_umword_t io, l4_umword_t obj,
00293                             l4_umword_t flags) L4_NOTHROW;
00294
00309 L4_INLINE
00310 l4_timeout_s l4_timeout_abs_u(l4_kernel_clock_t pint, int br,
00311                               l4_utcb_t *utcb) L4_NOTHROW;
00325 L4_INLINE
00326 l4_timeout_s l4_timeout_abs(l4_kernel_clock_t pint, int br) L4_NOTHROW;
00327
00335 L4_INLINE
00336 unsigned l4_utcb_mr64_idx(unsigned idx) L4_NOTHROW;
00337
00338 /*****
00339  * Implementations
00340  *****/
00341
00342 L4_INLINE l4_msg_regs_t *l4_utcb_mr_u(l4_utcb_t *u) L4_NOTHROW
00343 { return (l4_msg_regs_t*)((char*)u + L4_UTCB_MSG_REGS_OFFSET); }
00344
00345 L4_INLINE l4_buf_regs_t *l4_utcb_br_u(l4_utcb_t *u) L4_NOTHROW
00346 { return (l4_buf_regs_t*)((char*)u + L4_UTCB_BUF_REGS_OFFSET); }
00347
00348 L4_INLINE l4_thread_regs_t *l4_utcb_tcr_u(l4_utcb_t *u) L4_NOTHROW
00349 { return (l4_thread_regs_t*)((char*)u + L4_UTCB_THREAD_REGS_OFFSET); }
00350
00351 L4_INLINE l4_exc_regs_t *l4_utcb_exc_u(l4_utcb_t *u) L4_NOTHROW
00352 { return (l4_exc_regs_t*)((char*)u + L4_UTCB_MSG_REGS_OFFSET); }
00353
00354 L4_INLINE void l4_utcb_inherit_fpu_u(l4_utcb_t *u, int switch_on) L4_NOTHROW
00355 {
00356     if (switch_on)
00357         l4_utcb_br_u(u)->bdr |= L4_UTCB_INHERIT_FPU;
00358     else
00359         l4_utcb_br_u(u)->bdr &= ~L4_UTCB_INHERIT_FPU;
00360 }
00361
00362 L4_INLINE l4_umword_t l4_bdr(l4_umword_t mem, l4_umword_t io, l4_umword_t obj,
00363                             l4_umword_t flags) L4_NOTHROW
00364 {
00365     // flags (bits 24..31) are already shifted!
00366     return (mem < 0) | (io < 5) | (obj < 10) | flags;
00367 }
00368
00369 L4_INLINE l4_utcb_t *l4_utcb(void) L4_NOTHROW
00370 {
00371     #ifdef L4SYS_USE_UTCB_WRAP
00372         return l4_utcb_wrap();
00373     #else
00374         return l4_utcb_direct();
00375     #endif
00376 }
00377
00378
00379
00380
00381 L4_INLINE l4_msg_regs_t *l4_utcb_mr(void) L4_NOTHROW
00382 { return l4_utcb_mr_u(l4_utcb()); }
00383
00384 L4_INLINE l4_buf_regs_t *l4_utcb_br(void) L4_NOTHROW
00385 { return l4_utcb_br_u(l4_utcb()); }
00386
00387 L4_INLINE l4_thread_regs_t *l4_utcb_tcr(void) L4_NOTHROW
00388 { return l4_utcb_tcr_u(l4_utcb()); }
00389
00390 L4_INLINE l4_exc_regs_t *l4_utcb_exc(void) L4_NOTHROW
00391 { return l4_utcb_exc_u(l4_utcb()); }
00392
00393 L4_INLINE void l4_utcb_inherit_fpu(int switch_on) L4_NOTHROW
00394 { l4_utcb_inherit_fpu_u(l4_utcb(), switch_on); }
00395
00396 L4_INLINE
00397 l4_timeout_s l4_timeout_abs_u(l4_kernel_clock_t val, int pos,
00398                               l4_utcb_t *utcb) L4_NOTHROW
00399 {
00400     union T
00401     {
00402         l4_kernel_clock_t t;
00403         l4_umword_t m[sizeof(l4_kernel_clock_t)/sizeof(l4_umword_t)];
00404     };
00405     l4_timeout_s to;
00406     to.t = 0x8000 | pos;

```

```

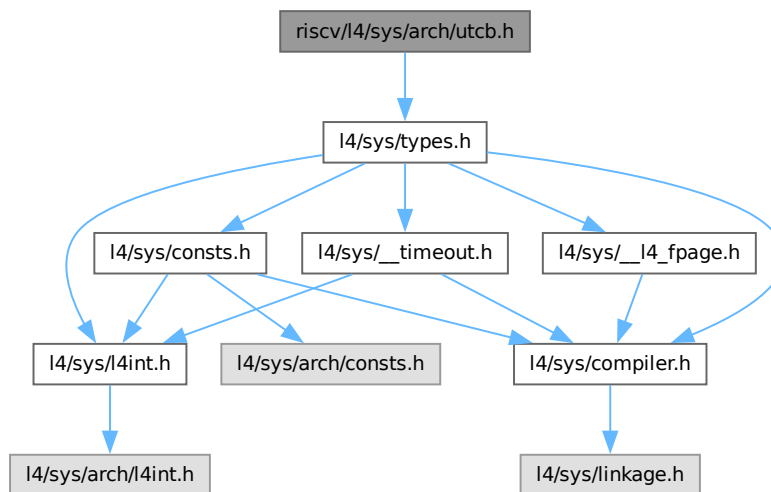
00407  ((union T*) (l4_utcb_br_u(utcb)->br + pos))->t = val;
00408  return to;
00409 }
00410
00411 L4_INLINE
00412 l4_timeout_s l4_timeout_abs(l4_kernel_clock_t val, int pos) L4_NOTHROW
00413 { return l4_timeout_abs_u(val, pos, l4_utcb()); }
00414
00415 L4_INLINE unsigned l4_utcb_mr64_idx(unsigned idx) L4_NOTHROW
00416 { return idx / (sizeof(l4_uint64_t) / sizeof(l4_umword_t)); }
00417
00418 L4_END_DECLS
00419
00420 #endif /* ! _L4_SYS_UTCB_H */

```

16.626 riscv/l4/sys/arch/utcb.h File Reference

UTCB definitions for RISC-V.

```
#include <l4/sys/types.h>
Include dependency graph for utcb.h:
```



Data Structures

- struct `l4_exc_regs_t`
UTCB structure for exceptions.

Functions

- `l4_umword_t l4_utcb_exc_pc(l4_exc_regs_t const *u) L4_NOTHROW`
Access function to get the program counter of the exception state.
- `void l4_utcb_exc_pc_set(l4_exc_regs_t *u, l4_addr_t pc) L4_NOTHROW`
Set the program counter register in the exception state.
- `l4_umword_t l4_utcb_exc_typeval(l4_exc_regs_t const *u) L4_NOTHROW`

Get the value out of an exception UTCB that describes the type of exception.

- `int l4_utcb_exc_is_pf (l4_exc_regs_t const *u) L4_NOTHROW`

Check whether an exception IPC is a page fault.

- `l4_addr_t l4_utcb_exc_pfa (l4_exc_regs_t const *u) L4_NOTHROW`

Function to get the L4 style page fault address out of an exception.

- `int l4_utcb_exc_is_ex_regs_exception (l4_exc_regs_t const *u) L4_NOTHROW`

Check whether an exception IPC was triggered via `l4_thread_ex_regs()`.

16.626.1 Detailed Description

UTCB definitions for RISC-V.

Definition in file `utcb.h`.

16.627 utcb.h

[Go to the documentation of this file.](#)

```
00001
00006 /*
00007  * Copyright (C) 2021, 2024 Kernkonzept GmbH.
00008  * Author(s): Georg Kotheimer <georg.kotheimer@kernkonzept.com>
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #pragma once
00013
00014 #include <l4/sys/types.h>
00015
00020
00021 enum L4_riscv_exc_cause
00022 {
00023     L4_riscv_exc_inst_misaligned      = 0,
00024     L4_riscv_exc_inst_access          = 1,
00025     L4_riscv_exc_illegal_inst         = 2,
00026     L4_riscv_exc_breakpoint           = 3,
00027     L4_riscv_exc_load_access          = 5,
00028     L4_riscv_exc_store_access         = 7,
00029     L4_riscv_exc_ecall                = 8,
00030     L4_riscv_exc_hcall                = 10,
00031     L4_riscv_exc_inst_page_fault      = 12,
00032     L4_riscv_exc_load_page_fault      = 13,
00033     L4_riscv_exc_store_page_fault     = 15,
00034     L4_riscv_exc_guest_inst_page_fault = 20,
00035     L4_riscv_exc_guest_load_page_fault = 21,
00036     L4_riscv_exc_virtual_inst         = 22,
00037     L4_riscv_exc_guest_store_page_fault = 23,
00038
00039     L4_riscv_ec_l4_ipc_upcall          = 0x18,
00040     L4_riscv_ec_l4_exregs_exception    = 0x19,
00041     L4_riscv_ec_l4_debug_ipi           = 0x1a,
00042     L4_riscv_ec_l4_alien_after_syscall = 0x1b,
00043 };
00044
00049 struct l4_exc_regs_t
00050 {
00051     l4_umword_t eret_work;
00052     union
00053     {
00054         l4_umword_t r[31];
00055         struct
00056         {
00057             l4_umword_t ra;
00058             l4_umword_t sp;
00059             l4_umword_t gp;
00060             l4_umword_t tp;
00061             l4_umword_t t0, t1, t2;
00062             l4_umword_t s0, s1;
00063             l4_umword_t a0, a1, a2, a3, a4, a5, a6, a7;
00064             l4_umword_t s2, s3, s4, s5, s6, s7, s8, s9, s10, s11;
00065             l4_umword_t t3, t4, t5, t6;
```

```

00066     };
00067 };
00068
00069 union { l4_umword_t pc; l4_umword_t ip; };
00070 l4_umword_t status;
00071 l4_umword_t cause;
00072 union { l4_umword_t tval; l4_umword_t pfa; };
00073
00074 l4_umword_t hstatus; // only if virtualization is enabled
00075 };
00076
00077 #define L4_UTCB_EXCEPTION_REGS_SIZE    (sizeof(l4_exc_regs_t) / sizeof(l4_umword_t))
00078 #define L4_UTCB_GENERIC_DATA_SIZE      63
00079 #define L4_UTCB_GENERIC_BUFFERS_SIZE   58
00080
00081 #define L4_UTCB_MSG_REGS_OFFSET         0
00082 #define L4_UTCB_BUF_REGS_OFFSET        (64 * sizeof(l4_umword_t))
00083 #define L4_UTCB_THREAD_REGS_OFFSET     (123 * sizeof(l4_umword_t))
00084
00085 #define L4_UTCB_INHERIT_FPU             (1UL < 24)
00086
00087 #define L4_UTCB_OFFSET                  (128 * sizeof(l4_umword_t))
00088
00089 /*
00090 * =====
00091 * Implementations.
00092 */
00093
00094 L4_INLINE l4_utcb_t *l4_utcb_direct(void) L4_NOTHROW
00095 {
00096     // We store the UTCB pointer immediately before the TCB.
00097     l4_addr_t tp;
00098     __asm__ ("mv %0, tp" : "=r" (tp));
00099     return *(l4_utcb_t **) (tp - 3 * sizeof(void*));
00100 }
00101
00102 L4_INLINE l4_umword_t l4_utcb_exc_pc(l4_exc_regs_t const *u) L4_NOTHROW
00103 {
00104     return u->pc;
00105 }
00106
00107 L4_INLINE void l4_utcb_exc_pc_set(l4_exc_regs_t *u, l4_addr_t pc) L4_NOTHROW
00108 {
00109     u->pc = pc;
00110 }
00111
00112 L4_INLINE l4_umword_t l4_utcb_exc_typeval(l4_exc_regs_t const *u) L4_NOTHROW
00113 {
00114     return u->cause;
00115 }
00116
00117 L4_INLINE int l4_utcb_exc_is_pf(l4_exc_regs_t const *u) L4_NOTHROW
00118 {
00119     switch(u->cause)
00120     {
00121         case L4_riscv_exc_inst_access:
00122         case L4_riscv_exc_load_access:
00123         case L4_riscv_exc_store_access:
00124         case L4_riscv_exc_inst_page_fault:
00125         case L4_riscv_exc_load_page_fault:
00126         case L4_riscv_exc_store_page_fault:
00127             return 1;
00128         default:
00129             return 0;
00130     }
00131 }
00132
00133 L4_INLINE l4_addr_t l4_utcb_exc_pfa(l4_exc_regs_t const *u) L4_NOTHROW
00134 {
00135     return u->tval | (u->cause == L4_riscv_exc_store_page_fault ? 2 : 0);
00136 }
00137
00138 L4_INLINE int l4_utcb_exc_is_ex_regs_exception(l4_exc_regs_t const *u) L4_NOTHROW
00139 {
00140     return l4_utcb_exc_typeval(u) == L4_riscv_ec_l4_exregs_exception;
00141 }

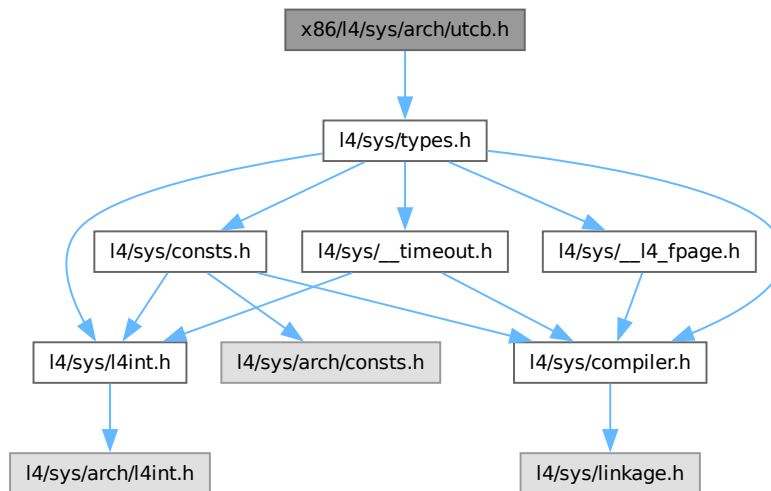
```

16.628 x86/i4/sys/arch/utcb.h File Reference

UTCB definitions for x86.

```
#include <l4/sys/types.h>
```

Include dependency graph for utcb.h:



Data Structures

- struct [l4_exc_regs_t](#)
UTCB structure for exceptions.

Macros

- #define **L4_UTCB_EXCEPTION_REGS_SIZE** (sizeof([l4_exc_regs_t](#)) / sizeof([l4_umword_t](#)))
Number of message registers used for exception IPC.
- #define **L4_UTCB_GENERIC_DATA_SIZE** 63
Total number of message register (MRs) available.
- #define **L4_UTCB_GENERIC_BUFFERS_SIZE** 58
Total number of buffer registers (BRs) available.
- #define **L4_UTCB_MSG_REGS_OFFSET** 0
Offset of MR[0] relative to the UTCB pointer.
- #define **L4_UTCB_BUF_REGS_OFFSET** (64 * sizeof([l4_umword_t](#)))
Offset of BR[0] relative to the UTCB pointer.
- #define **L4_UTCB_THREAD_REGS_OFFSET** (123 * sizeof([l4_umword_t](#)))
Offset of TCR[0] relative to the UTCB pointer.
- #define **L4_UTCB_INHERIT_FPU** (1UL << 24)
BDR flag to accept reception of FPU state.
- #define **L4_UTCB_OFFSET** 512
Offset of two consecutive UTCBs.

Functions

- `l4_umword_t l4_utcb_exc_pc (l4_exc_regs_t const *u) L4_NOTHROW`
Access function to get the program counter of the exception state.
- `void l4_utcb_exc_pc_set (l4_exc_regs_t *u, l4_addr_t pc) L4_NOTHROW`
Set the program counter register in the exception state.
- `l4_umword_t l4_utcb_exc_typeval (l4_exc_regs_t const *u) L4_NOTHROW`
Get the value out of an exception UTCB that describes the type of exception.
- `int l4_utcb_exc_is_pf (l4_exc_regs_t const *u) L4_NOTHROW`
Check whether an exception IPC is a page fault.
- `l4_addr_t l4_utcb_exc_pfa (l4_exc_regs_t const *u) L4_NOTHROW`
Function to get the L4 style page fault address out of an exception.
- `int l4_utcb_exc_is_ex_regs_exception (l4_exc_regs_t const *u) L4_NOTHROW`
Check whether an exception IPC was triggered via `l4_thread_ex_regs()`.

16.628.1 Detailed Description

UTCB definitions for x86.

Definition in file [utcb.h](#).

16.629 utcb.h

[Go to the documentation of this file.](#)

```

00001 /*****
00007 */
00008 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009 *      Alexander Warg <warg@os.inf.tu-dresden.de>
00010 *      economic rights: Technische Universität Dresden (Germany)
00011 *
00012 * License: see LICENSE.spdx (in this directory or the directories above)
00013 */
00014 /*****
00015 #ifndef __L4_SYS__INCLUDE__ARCH_X86__UTCB_H__
00016 #define __L4_SYS__INCLUDE__ARCH_X86__UTCB_H__
00017
00018 #include <l4/sys/types.h>
00019
00024
00029 struct l4_exc_regs_t
00030 {
00031     l4_umword_t es;
00032     l4_umword_t ds;
00033     l4_umword_t gs;
00034     l4_umword_t fs;
00035
00036     l4_umword_t edi;
00037     l4_umword_t esi;
00038     l4_umword_t ebp;
00039     l4_umword_t pfa;
00040     l4_umword_t ebx;
00041     l4_umword_t edx;
00042     l4_umword_t ecx;
00043     l4_umword_t eax;
00044
00045     l4_umword_t trapno;
00046     l4_umword_t err;
00047
00048     l4_umword_t ip;
00049     l4_umword_t dummy1;
00050     l4_umword_t flags;
00051     l4_umword_t sp;
00052     l4_umword_t ss;
00053 };
00054
00056 #define L4_UTCB_EXCEPTION_REGS_SIZE    (sizeof(l4_exc_regs_t) / sizeof(l4_umword_t))

```

```

00057
00059 #define L4_UTCB_GENERIC_DATA_SIZE      63
00060
00062 #define L4_UTCB_GENERIC_BUFFERS_SIZE    58
00063
00065 #define L4_UTCB_MSG_REGS_OFFSET         0
00066
00068 #define L4_UTCB_BUF_REGS_OFFSET          (64 * sizeof(l4_umword_t))
00069
00071 #define L4_UTCB_THREAD_REGS_OFFSET       (123 * sizeof(l4_umword_t))
00072
00074 #define L4_UTCB_INHERIT_FPU              (1UL << 24)
00075
00077 #define L4_UTCB_OFFSET                   512
00078
00079 /*
00080 * =====
00081 * Implementations.
00082 */
00083
00084 L4_INLINE l4_utcb_t *l4_utcb_direct(void) L4_NOTHROW
00085 {
00086     l4_utcb_t *utcb;
00087     __asm__ ("mov %%fs:0, %0" : "=r" (utcb));
00088     return utcb;
00089 }
00090
00091 L4_INLINE l4_umword_t l4_utcb_exc_pc(l4_exc_regs_t const *u) L4_NOTHROW
00092 {
00093     return u->ip;
00094 }
00095
00096 L4_INLINE void l4_utcb_exc_pc_set(l4_exc_regs_t *u, l4_addr_t pc) L4_NOTHROW
00097 {
00098     u->ip = pc;
00099 }
00100
00101 L4_INLINE void l4_utcb_exc_sp_set(l4_exc_regs_t *u, l4_addr_t sp) L4_NOTHROW
00102 {
00103     u->sp = sp;
00104 }
00105
00106 L4_INLINE l4_umword_t l4_utcb_exc_typeval(l4_exc_regs_t const *u) L4_NOTHROW
00107 {
00108     return u->trapno;
00109 }
00110
00111 L4_INLINE int l4_utcb_exc_is_pf(l4_exc_regs_t const *u) L4_NOTHROW
00112 {
00113     return u->trapno == 14;
00114 }
00115
00116 L4_INLINE l4_addr_t l4_utcb_exc_pfa(l4_exc_regs_t const *u) L4_NOTHROW
00117 {
00118     return (u->pfa & ~7UL) | (u->err & 2);
00119 }
00120
00121 L4_INLINE int l4_utcb_exc_is_ex_regs_exception(l4_exc_regs_t const *u) L4_NOTHROW
00122 {
00123     return l4_utcb_exc_typeval(u) == 0xff;
00124 }
00125
00126 #endif /* ! __L4_SYS__INCLUDE__ARCH_X86__UTCB_H__ */

```

16.630 l4/sys/vcon File Reference

C++ Virtual console interface.

```

#include <l4/sys/icu>
#include <l4/sys/vcon.h>
#include <l4/sys/capability>

```

[illegible]

- class [L4::Vcon](#)
C++ L4 Vcon interface, see [Virtual Console](#) for the C interface.

- namespace **L4**
L4 low-level kernel interface.

16.630.1 Detailed Description

C++ Virtual console interface.

Definition in file [vcon](#).

16.631 vcon

[Go to the documentation of this file.](#)

```

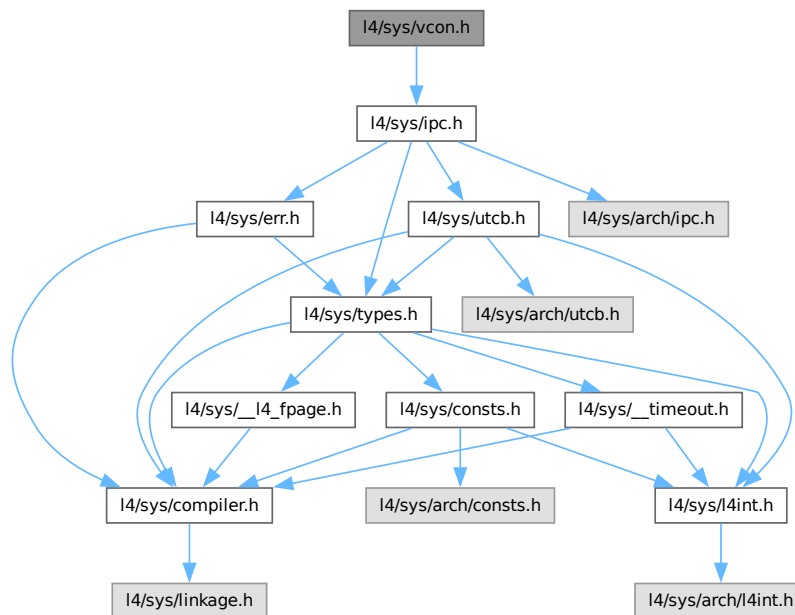
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00005  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00006  *      economic rights: Technische Universität Dresden (Germany)
00007  *
00008  * License: see LICENSE.spdx (in this directory or the directories above)
00009  */
00010 #pragma once
00011
00012 #include <l4/sys/icu>
00013 #include <l4/sys/vcon.h>
00014 #include <l4/sys/capability>
00015
00016 namespace L4 {
00017
00018 class Vcon :
00019     public Kobject<Vcon, Icu, L4_PROTO_LOG>
00020 {
00021 public:
00022     l4_msgtag_t
00023     send(char const *buf, unsigned size, l4_utcb_t *utcb = l4_utcb()) const noexcept
00024     { return l4_vcon_send_u(cap(), buf, size, utcb); }
00025
00026     long
00027     write(char const *buf, unsigned size, l4_utcb_t *utcb = l4_utcb()) const noexcept
00028     { return l4_vcon_write_u(cap(), buf, size, utcb); }
00029
00030     int
00031     read(char *buf, unsigned size, l4_utcb_t *utcb = l4_utcb()) const noexcept
00032     { return l4_vcon_read_u(cap(), buf, size, utcb); }
00033
00034     int
00035     read_with_flags(char *buf, unsigned size, l4_utcb_t *utcb = l4_utcb()) const noexcept
00036     { return l4_vcon_read_with_flags_u(cap(), buf, size, utcb); }
00037
00038     l4_msgtag_t
00039     set_attr(l4_vcon_attr_t const *attr, l4_utcb_t *utcb = l4_utcb()) const noexcept
00040     { return l4_vcon_set_attr_u(cap(), attr, utcb); }
00041
00042     l4_msgtag_t
00043     get_attr(l4_vcon_attr_t *attr, l4_utcb_t *utcb = l4_utcb()) const noexcept
00044     { return l4_vcon_get_attr_u(cap(), attr, utcb); }
00045
00046     typedef L4::Typeid::Raw_ipc<Vcon> Rpcs;
00047 };
00048
00049 }
```

16.632 l4/sys/vcon.h File Reference

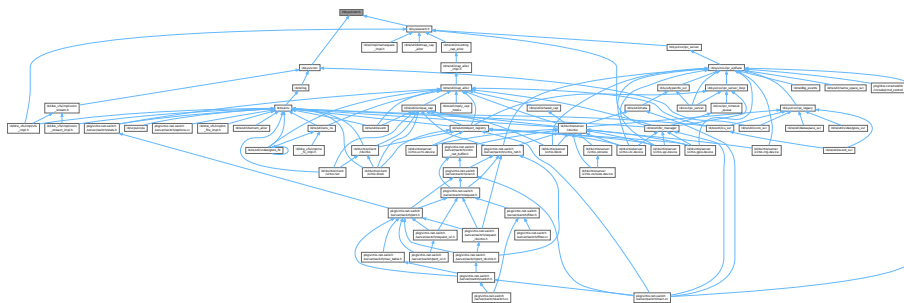
Virtual console interface.

```
#include <l4/sys/ipc.h>
```

Include dependency graph for vcon.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct `l4_vcon_attr_t`
Vcon attribute structure.

Typedefs

- typedef struct `l4_vcon_attr_t` `l4_vcon_attr_t`
Vcon attribute structure.

Enumerations

- enum `L4_vcon_size_consts` { `L4_VCON_WRITE_SIZE` = (L4_UTCB_GENERIC_DATA_SIZE - 2) * sizeof(l4_umword_t) , `L4_VCON_READ_SIZE` = (L4_UTCB_GENERIC_DATA_SIZE - 1) * sizeof(l4_umword_t) }
Size constants.
- enum `L4_vcon_read_flags` { `L4_VCON_READ_SIZE_MASK` = 0x3ffffff , `L4_VCON_READ_STAT_BREAK` = 1 << 30 , `L4_VCON_READ_STAT_DONE` = 1 << 31 }
Vcon read flags.
- enum `L4_vcon_i_flags` { `L4_VCON_INLCR` = 000100 , `L4_VCON_IGNCR` = 000200 , `L4_VCON_ICRNL` = 000400 }
Input flags.
- enum `L4_vcon_o_flags` { `L4_VCON_ONLCR` = 000004 , `L4_VCON_OCRNL` = 000010 , `L4_VCON_ONLRET` = 000040 }
Output flags.
- enum `L4_vcon_l_flags` { `L4_VCON_ICANON` = 000002 , `L4_VCON_ECHO` = 000010 }
Local flags.
- enum `L4_vcon_ops` { `L4_VCON_WRITE_OP` = 0UL , `L4_VCON_READ_OP` = 1UL , `L4_VCON_SET_ATTR_OP` = 2UL , `L4_VCON_GET_ATTR_OP` = 3UL }
Operations on vcon objects.

Functions

- `l4_msgtag_t l4_vcon_send (l4_cap_idx_t vcon, char const *buf, unsigned size)` `L4_NOTHROW`
Send data to virtual console.
- `l4_msgtag_t l4_vcon_send_u (l4_cap_idx_t vcon, char const *buf, unsigned size, l4_utcb_t *utcb)` `L4_NOTHROW`
Send data to *this* virtual console.
- `long l4_vcon_write (l4_cap_idx_t vcon, char const *buf, unsigned size)` `L4_NOTHROW`
Write data to virtual console.
- `long l4_vcon_write_u (l4_cap_idx_t vcon, char const *buf, unsigned size, l4_utcb_t *utcb)` `L4_NOTHROW`
Write data to *this* virtual console.
- `int l4_vcon_read (l4_cap_idx_t vcon, char *buf, unsigned size)` `L4_NOTHROW`
Read data from virtual console.
- `int l4_vcon_read_u (l4_cap_idx_t vcon, char *buf, unsigned size, l4_utcb_t *utcb)` `L4_NOTHROW`
Read data from *this* virtual console.
- `int l4_vcon_read_with_flags (l4_cap_idx_t vcon, char *buf, unsigned size)` `L4_NOTHROW`
Read data from virtual console, extended version including flags.
- `l4_msgtag_t l4_vcon_set_attr (l4_cap_idx_t vcon, l4_vcon_attr_t const *attr)` `L4_NOTHROW`
Set attributes of a Vcon.
- `l4_msgtag_t l4_vcon_set_attr_u (l4_cap_idx_t vcon, l4_vcon_attr_t const *attr, l4_utcb_t *utcb)` `L4_NOTHROW`
Set the attributes of *this* virtual console.
- `l4_msgtag_t l4_vcon_get_attr (l4_cap_idx_t vcon, l4_vcon_attr_t *attr)` `L4_NOTHROW`
Get attributes of a Vcon.
- `l4_msgtag_t l4_vcon_get_attr_u (l4_cap_idx_t vcon, l4_vcon_attr_t *attr, l4_utcb_t *utcb)` `L4_NOTHROW`
Get attributes of *this* virtual console.
- `void l4_vcon_set_attr_raw (l4_vcon_attr_t *attr)` `L4_NOTHROW`
Set terminal attributes to disable all special processing.

16.632.1 Detailed Description

Virtual console interface.

Definition in file [vcon.h](#).

16.632.2 Enumeration Type Documentation

16.632.2.1 L4_vcon_read_flags

```
enum L4_vcon_read_flags
```

Vcon read flags.

Enumerator

| | |
|-------------------------|-----------------------|
| L4_VCON_READ_SIZE_MASK | Size mask. |
| L4_VCON_READ_STAT_BREAK | Break condition flag. |
| L4_VCON_READ_STAT_DONE | Done condition flag. |

Definition at line 170 of file [vcon.h](#).

16.633 vcon.h

[Go to the documentation of this file.](#)

```
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00008  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013 #pragma once
00014
00015 #include <l4/sys/ipc.h>
00016
00039
00056 L4_INLINE l4_msgtag_t
00057 l4_vcon_send(l4_cap_idx_t vcon, char const *buf, unsigned size) L4_NOTHROW;
00058
00065 L4_INLINE l4_msgtag_t
00066 l4_vcon_send_u(l4_cap_idx_t vcon, char const *buf, unsigned size, l4_utcb_t *utcb) L4_NOTHROW;
00067
00079 L4_INLINE long
00080 l4_vcon_write(l4_cap_idx_t vcon, char const *buf, unsigned size) L4_NOTHROW;
00081
00088 L4_INLINE long
00089 l4_vcon_write_u(l4_cap_idx_t vcon, char const *buf, unsigned size, l4_utcb_t *utcb) L4_NOTHROW;
00090
00095 enum L4_vcon_size_consts
00096 {
00098     L4_VCON_WRITE_SIZE = (L4_UTCB_GENERIC_DATA_SIZE - 2) * sizeof(l4_umword_t),
00100     L4_VCON_READ_SIZE  = (L4_UTCB_GENERIC_DATA_SIZE - 1) * sizeof(l4_umword_t),
00101 };
00102
00119 L4_INLINE int
00120 l4_vcon_read(l4_cap_idx_t vcon, char *buf, unsigned size) L4_NOTHROW;
00121
00128 L4_INLINE int
00129 l4_vcon_read_u(l4_cap_idx_t vcon, char *buf, unsigned size, l4_utcb_t *utcb) L4_NOTHROW;
```

```

00130
00157 L4_INLINE int
00158 l4_vcon_read_with_flags(l4_cap_idx_t vcon, char *buf, unsigned size) L4_NOTHROW;
00159
00163 L4_INLINE int
00164 l4_vcon_read_with_flags_u(l4_cap_idx_t vcon, char *buf, unsigned size,
00165                           l4_utcb_t *utcb) L4_NOTHROW;
00166
00170 enum L4_vcon_read_flags
00171 {
00172     L4_VCON_READ_SIZE_MASK = 0x3fffffff,
00173     L4_VCON_READ_STAT_BREAK = 1 << 30,
00174     L4_VCON_READ_STAT_DONE = 1 << 31,
00175 };
00176
00187 typedef struct l4_vcon_attr_t
00188 {
00189     l4_umword_t i_flags;
00190     l4_umword_t o_flags;
00191     l4_umword_t l_flags;
00192
00193 #ifdef __cplusplus
00200     inline void set_raw();
00201 #endif
00202 } l4_vcon_attr_t;
00203
00208 enum L4_vcon_i_flags
00209 {
00210     L4_VCON_INLCR = 000100,
00211     L4_VCON_IGNCR = 000200,
00212     L4_VCON_ICRNL = 000400,
00213 };
00214
00219 enum L4_vcon_o_flags
00220 {
00221     L4_VCON_ONLCR = 000004,
00222     L4_VCON_OCRNL = 000010,
00223     L4_VCON_ONLRET = 000040,
00224 };
00225
00230 enum L4_vcon_l_flags
00231 {
00232     L4_VCON_ICANON = 000002,
00233     L4_VCON_ECHO = 000010,
00234 };
00235
00244 L4_INLINE l4_msgtag_t
00245 l4_vcon_set_attr(l4_cap_idx_t vcon, l4_vcon_attr_t const *attr) L4_NOTHROW;
00246
00253 L4_INLINE l4_msgtag_t
00254 l4_vcon_set_attr_u(l4_cap_idx_t vcon, l4_vcon_attr_t const *attr,
00255                   l4_utcb_t *utcb) L4_NOTHROW;
00256
00265 L4_INLINE l4_msgtag_t
00266 l4_vcon_get_attr(l4_cap_idx_t vcon, l4_vcon_attr_t *attr) L4_NOTHROW;
00267
00274 L4_INLINE l4_msgtag_t
00275 l4_vcon_get_attr_u(l4_cap_idx_t vcon, l4_vcon_attr_t *attr,
00276                   l4_utcb_t *utcb) L4_NOTHROW;
00277
00283 L4_INLINE void
00284 l4_vcon_set_attr_raw(l4_vcon_attr_t *attr) L4_NOTHROW;
00285
00286
00291 enum L4_vcon_ops
00292 {
00293     L4_VCON_WRITE_OP = 0UL,
00294     L4_VCON_READ_OP = 1UL,
00295     L4_VCON_SET_ATTR_OP = 2UL,
00296     L4_VCON_GET_ATTR_OP = 3UL,
00297 };
00298
00299 /***** Implementations *****/
00300
00301 L4_INLINE l4_msgtag_t
00302 l4_vcon_send_u(l4_cap_idx_t vcon, char const *buf, unsigned size, l4_utcb_t *utcb) L4_NOTHROW
00303 {
00304     l4_msg_regs_t *mr = l4_utcb_mr_u(utcb);
00305     mr->mr[0] = L4_VCON_WRITE_OP;
00306     mr->mr[1] = size;
00307     __builtin_memcpy(&mr->mr[2], buf, size);
00308     return l4_ipc_send(vcon, utcb,
00309                       l4_msgtag(L4_PROTO_LOG, 2 + l4_bytes_to_mwords(size),
00310                                0, L4_MSGTAG_SCHEDULE),
00311                       L4_IPC_NEVER);
00312 }
00313

```



```

00314 L4_INLINE l4_msgtag_t
00315 l4_vcon_send(l4_cap_idx_t vcon, char const *buf, unsigned size) L4_NOTHROW
00316 {
00317     return l4_vcon_send_u(vcon, buf, size, l4_utcb());
00318 }
00319
00320 L4_INLINE long
00321 l4_vcon_write_u(l4_cap_idx_t vcon, char const *buf, unsigned size, l4_utcb_t *utcb) L4_NOTHROW
00322 {
00323     l4_msgtag_t t;
00324
00325     if (size > L4_VCON_WRITE_SIZE)
00326         size = L4_VCON_WRITE_SIZE;
00327
00328     t = l4_vcon_send_u(vcon, buf, size, utcb);
00329     if (l4_msgtag_has_error(t))
00330         return l4_error(t);
00331
00332     return (long) size;
00333 }
00334
00335 L4_INLINE long
00336 l4_vcon_write(l4_cap_idx_t vcon, char const *buf, unsigned size) L4_NOTHROW
00337 {
00338     return l4_vcon_write_u(vcon, buf, size, l4_utcb());
00339 }
00340
00341 L4_INLINE int
00342 l4_vcon_read_with_flags_u(l4_cap_idx_t vcon, char *buf, unsigned size,
00343                          l4_utcb_t *utcb) L4_NOTHROW
00344 {
00345     int ret;
00346     unsigned r;
00347     l4_msg_regs_t *mr;
00348
00349     mr = l4_utcb_mr_u(utcb);
00350     mr->mr[0] = (size << 16) | L4_VCON_READ_OP;
00351
00352     ret = l4_error_u(l4_ipc_call(vcon, utcb,
00353                                l4_msgtag(L4_PROTO_LOG, 1, 0, 0),
00354                                L4_IPC_NEVER),
00355                    utcb);
00356     if (ret < 0)
00357         return ret;
00358
00359     r = mr->mr[0] & L4_VCON_READ_SIZE_MASK;
00360
00361     if (!(mr->mr[0] & L4_VCON_READ_STAT_DONE)) // !eof
00362         ret = size + 1;
00363     else if (r < size)
00364         ret = r;
00365     else
00366         ret = size;
00367
00368     if (L4_LIKELY(buf != NULL))
00369         __builtin_memcpy(buf, &mr->mr[1], r < size ? r : size);
00370
00371     return ret | (mr->mr[0] & ~(L4_VCON_READ_STAT_DONE | L4_VCON_READ_SIZE_MASK));
00372 }
00373
00374 L4_INLINE int
00375 l4_vcon_read_with_flags(l4_cap_idx_t vcon, char *buf, unsigned size) L4_NOTHROW
00376 {
00377     return l4_vcon_read_with_flags_u(vcon, buf, size, l4_utcb());
00378 }
00379
00380 L4_INLINE int
00381 l4_vcon_read_u(l4_cap_idx_t vcon, char *buf, unsigned size, l4_utcb_t *utcb) L4_NOTHROW
00382 {
00383     int r = l4_vcon_read_with_flags_u(vcon, buf, size, utcb);
00384     if (r < 0)
00385         return r;
00386
00387     return r & L4_VCON_READ_SIZE_MASK;
00388 }
00389
00390 L4_INLINE int
00391 l4_vcon_read(l4_cap_idx_t vcon, char *buf, unsigned size) L4_NOTHROW
00392 {
00393     return l4_vcon_read_u(vcon, buf, size, l4_utcb());
00394 }
00395
00396 L4_INLINE l4_msgtag_t
00397 l4_vcon_set_attr_u(l4_cap_idx_t vcon, l4_vcon_attr_t const *attr,
00398                  l4_utcb_t *utcb) L4_NOTHROW
00399 {
00400     l4_msg_regs_t *mr = l4_utcb_mr_u(utcb);

```

```

00401
00402     mr->mr[0] = L4_VCON_SET_ATTR_OP;
00403     __builtin_memcpy(&mr->mr[1], attr, sizeof(*attr));
00404
00405     return l4_ipc_call(vcon, utcb,
00406                       l4_msgtag(L4_PROTO_LOG, 4, 0, 0),
00407                       L4_IPC_NEVER);
00408 }
00409
00410 L4_INLINE l4_msgtag_t
00411 l4_vcon_set_attr(l4_cap_idx_t vcon, l4_vcon_attr_t const *attr) L4_NOTHROW
00412 {
00413     return l4_vcon_set_attr_u(vcon, attr, l4_utcb());
00414 }
00415
00416 L4_INLINE l4_msgtag_t
00417 l4_vcon_get_attr_u(l4_cap_idx_t vcon, l4_vcon_attr_t *attr,
00418                   l4_utcb_t *utcb) L4_NOTHROW
00419 {
00420     l4_msgtag_t res;
00421     l4_msg_regs_t *mr = l4_utcb_mr_u(utcb);
00422
00423     mr->mr[0] = L4_VCON_GET_ATTR_OP;
00424
00425     res = l4_ipc_call(vcon, utcb,
00426                      l4_msgtag(L4_PROTO_LOG, 1, 0, 0),
00427                      L4_IPC_NEVER);
00428     if (l4_error_u(res, utcb) >= 0)
00429         __builtin_memcpy(attr, &mr->mr[1], sizeof(*attr));
00430
00431     return res;
00432 }
00433
00434 L4_INLINE l4_msgtag_t
00435 l4_vcon_get_attr(l4_cap_idx_t vcon, l4_vcon_attr_t *attr) L4_NOTHROW
00436 {
00437     return l4_vcon_get_attr_u(vcon, attr, l4_utcb());
00438 }
00439
00440 L4_INLINE void
00441 l4_vcon_set_attr_raw(l4_vcon_attr_t *attr) L4_NOTHROW
00442 {
00443     attr->i_flags = 0;
00444     attr->o_flags = 0;
00445     attr->l_flags = 0;
00446 }
00447
00448 #ifdef __cplusplus
00449 inline void
00450 l4_vcon_attr_t::set_raw()
00451 { l4_vcon_set_attr_raw(this); }
00452 #endif

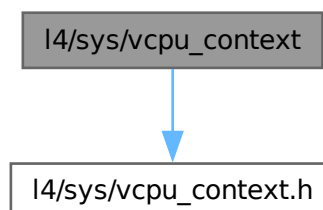
```

16.634 l4/sys/vcpu_context File Reference

Hardware vCPU context interface.

```
#include <l4/sys/vcpu_context.h>
```

Include dependency graph for vcpu_context:



Namespaces

- namespace [L4](#)
[L4](#) low-level kernel interface.

16.634.1 Detailed Description

Hardware vCPU context interface.

Definition in file [vcpu_context](#).

16.635 vcpu_context

[Go to the documentation of this file.](#)

```
00001 // vi:set ft=c++: -*- Mode: C++ -*-
00006
00007 #pragma once
00008
00009 #include <l4/sys/vcpu_context.h>
00010
00011 namespace L4 {
00012
00013 class Vcpu_context :
00014     public Kobject_t<Vcpu_context, Kobject, L4_PROTO_VCPU_CONTEXT>
00015 {
00016 public:
00017     Vcpu_context(Vcpu_context const &) = delete;
00018     void operator = (Vcpu_context const &) = delete;
00019
00020 protected:
00021     Vcpu_context();
00022 };
00023
00024 };
```

16.636 vcpu_context.h

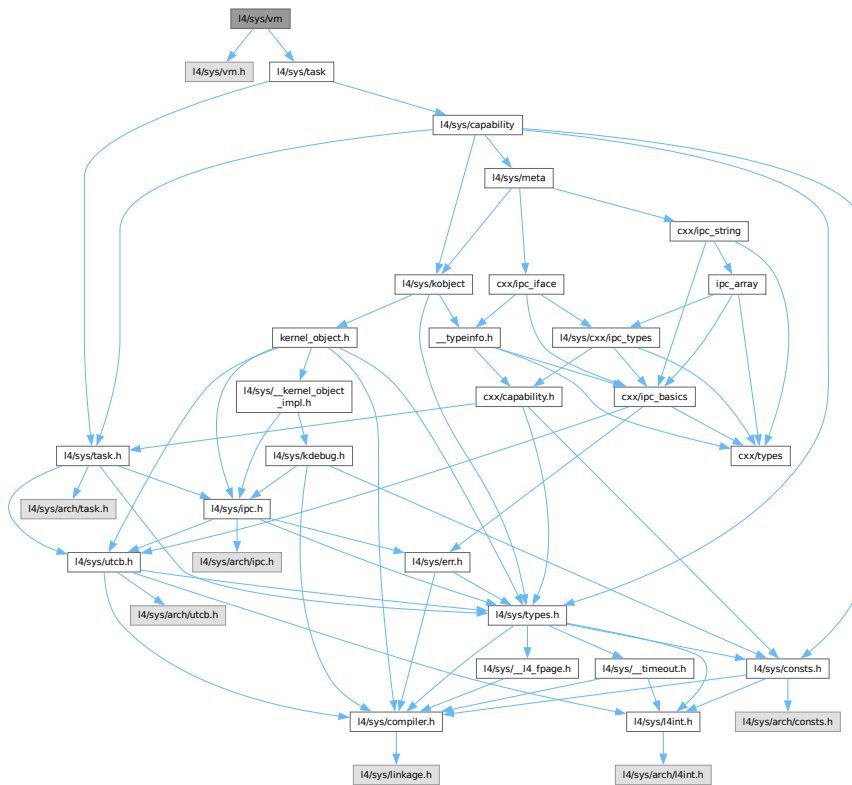
```
00001
00006
00007 #pragma once
```

16.637 l4/sys/vm File Reference

Virtualization interface.

```
#include <l4/sys/vm.h>
#include <l4/sys/task>
```

Include dependency graph for `vm`:



Data Structures

- class [L4::Vm](#)
Virtual machine host address space.

Namespaces

- namespace [L4](#)
[L4](#) low-level kernel interface.

16.637.1 Detailed Description

Virtualization interface.

Definition in file [vm](#).

16.638 vm

[Go to the documentation of this file.](#)

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2008-2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009
00010 #pragma once
00011
00012 #include <l4/sys/vm.h>
00013 #include <l4/sys/task>
00014
00015 namespace L4 {
00016
00017 class Vm : public Kobject_t<Vm, Task, L4_PROTO_VM>
00018 {
00019 public:
00020     #if defined(__arm__) || defined(__aarch64__)
00021         l4_msgtag_t vgicc_map(l4_fpage_t const vgicc_fpage,
00022                               l4_utcb_t *utcb = l4_utcb()) noexcept
00023         { return l4_task_vgicc_map_u(cap(), vgicc_fpage, utcb); }
00024     #endif
00025
00026 protected:
00027     Vm();
00028
00029 private:
00030     Vm(Vm const &);
00031     void operator = (Vm const &);
00032 };
00033
00034
00035
00036
00037
00038
00039
00040
00041
00042
00043
00044
00045
00046
00047
00048
00049
00050
00051
00052
00053
00054
00055
00056
00057

```

16.639 l4/umalloc/umalloc.h File Reference

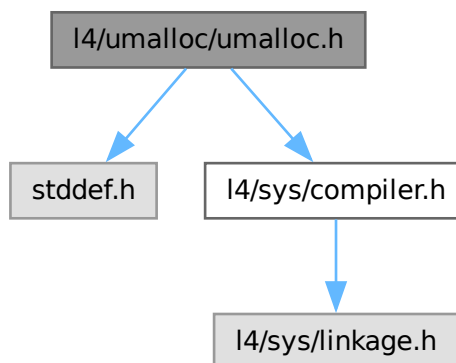
Public API for the basic next-fit memory allocator.

```

#include <stddef.h>
#include <l4/sys/compiler.h>

```

Include dependency graph for umalloc.h:



Functions

- `void * umalloc_area_create (size_t area_size) L4_NOTHROW`
Create a new heap area.

Variables

- `L4_BEGIN_DECLS size_t umalloc_area_granularity`
Heap area granularity.

16.639.1 Detailed Description

Public API for the basic next-fit memory allocator.

The user of this allocator is required to provide the implementation of the `umalloc_area_create()` function and provide the `umalloc_area_granularity` symbol with a value.

Note

The current implementation is NOT thread-safe. If the user wants to use this allocator concurrently, they need to deploy their custom mutual exclusion mechanism around the public calls to the allocator.

Definition in file `umalloc.h`.

16.639.2 Function Documentation

16.639.2.1 `umalloc_area_create()`

```
void * umalloc_area_create (
    size_t area_size)
```

Create a new heap area.

Parameters

| | |
|------------------------|---|
| <code>area_size</code> | Requested heap area size. Always a multiple of the heap area granularity as reported by <code>umalloc_area_granularity()</code> . |
|------------------------|---|

Returns

Pointer to the successfully created heap area of (at least) the requested size. It is assumed that the pointer satisfies at least the `umalloc::Base_alignment` alignment. If the value is `nullptr`, then it is assumed that the creation of the heap area failed.

References `L4_END_DECLS`, and `L4_NOTHROW`.

16.639.3 Variable Documentation

16.639.3.1 umalloc_area_granularity

`L4_BEGIN_DECLS` `size_t` `umalloc_area_granularity` `[extern]`

Heap area granularity.

The allocator only requests the creation of heap areas with this granularity.

16.640 umalloc.h

[Go to the documentation of this file.](#)

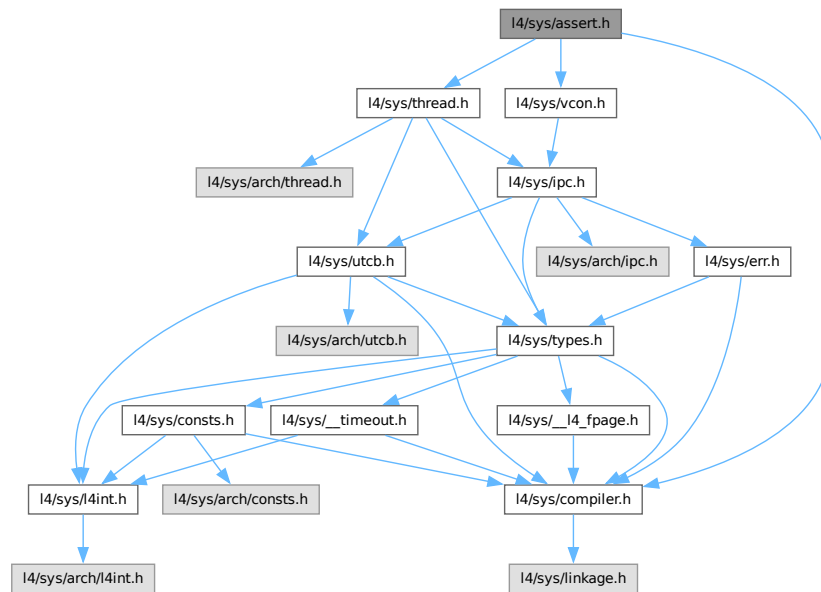
```
00001 /*
00002  * Copyright (C) 2025 Kernkonzept GmbH.
00003  * Author(s): Martin Decky <martin.decky@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00020
00021 #pragma once
00022
00023 #include <stddef.h>
00024 #include <l4/sys/compiler.h>
00025
00026 L4_BEGIN_DECLS
00027
00033 extern size_t umalloc_area_granularity;
00034
00047 void *umalloc_area_create(size_t area_size) L4_NOTHROW;
00048
00049 L4_END_DECLS
```

16.641 l4/sys/assert.h File Reference

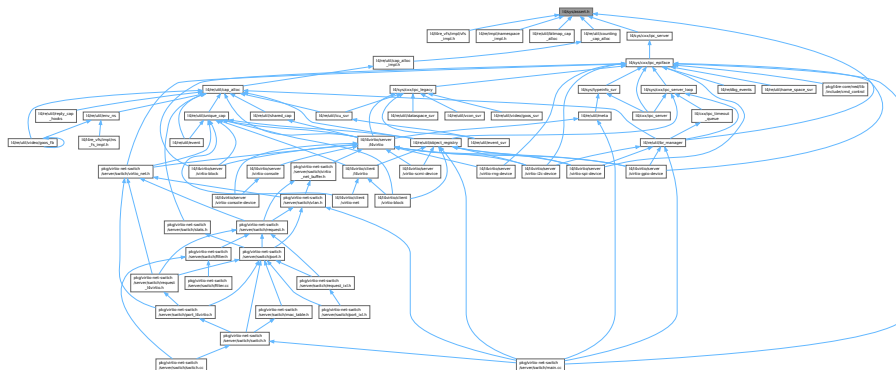
Low-level assert implementation.

```
#include <l4/sys/compiler.h>
#include <l4/sys/thread.h>
#include <l4/sys/vcon.h>
```

Include dependency graph for assert.h:



This graph shows which files directly or indirectly include this file:



Macros

- `#define l4_assert(expr)`
Low-level assert.

16.641.1 Detailed Description

Low-level assert implementation.

Definition in file [assert.h](#).

16.641.2 Macro Definition Documentation

16.641.2.1 l4_assert

```
#define l4_assert(  
    expr)
```

Value:

```
l4_assert_fn(!(expr), __FILE__ ":" L4_stringify(__LINE__) ": Assertion \"" \
    L4_stringify(expr) "\" failed.\n")
```

Low-level assert.

Parameters

| | |
|-------------|---|
| <i>expr</i> | Expression to be evaluated for the assertion. |
|-------------|---|

This assertion is a low-level implementation that directly uses kernel primitives. Only use `l4_assert()` when the standard `assert()` functionality is not available.

Definition at line 32 of file [assert.h](#).

Referenced by [L4Re::Util::Cap_alloc_base::free\(\)](#), [L4Re::Util::Counting_cap_alloc< COUNTERTYPE, Dbg >::free\(\)](#), [L4Re::Util::Counting_cap_alloc< COUNTERTYPE, Dbg >::release\(\)](#), and [L4Re::Util::Br_manager::set_rcv_cap_flags\(\)](#).

16.642 assert.h

[Go to the documentation of this file.](#)

```
00001
00005 /*
00006  * (c) 2015 Adam Lackorzynski <adam@l4re.org>
00007  *
00008  * License: see LICENSE.spdx (in this directory or the directories above)
00009  */
00010 #pragma once
00011
00012 #ifdef NDEBUG
00013
00014 #define l4_assert(x) do { } while (0)
00015 #define l4_check(x) do { (void)(x); } while (0)
00016
00017 #else
00018
00019 #include <l4/sys/compiler.h>
00020 #include <l4/sys/thread.h>
00021 #include <l4/sys/vcon.h>
00022
00032 #define l4_assert(expr) \
00033     l4_assert_fn(!(expr), __FILE__ ":" L4_stringify(__LINE__) ": Assertion \"" \
00034         L4_stringify(expr) "\" failed.\n")
00035
00036 #define l4_check(expr) l4_assert(expr)
00037
00041 L4_ALWAYS_INLINE
00042 void l4_assert_fn(unsigned expr, const char *text) L4_NOTHROW;
00043
00047 L4_INLINE L4_NORETURN
00048 void l4_assert_abort(const char *text) L4_NOTHROW;
00049
00050
00051 /* IMPLEMENTATION ----- */
00052
00053 L4_INLINE L4_NORETURN
00054 void l4_assert_abort(const char *text) L4_NOTHROW
00055 {
00056     l4_vcon_write(L4_BASE_LOG_CAP, text, __builtin_strlen(text));
```

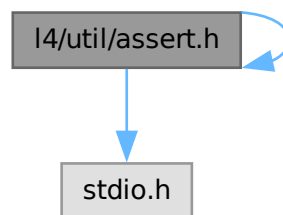
```
00057     for (;;)
00058     {
00059         l4_thread_ex_regs(L4_INVALID_CAP, ~0UL, ~0UL,
00060                          L4_THREAD_EX_REGS_TRIGGER_EXCEPTION);
00061     }
00062     L4_ALWAYS_INLINE
00063 void l4_assert_fn(unsigned expr, const char *text) L4_NOTHROW
00064 {
00065     if (L4_LIKELY(expr))
00066         return;
00067     l4_assert_abort(text);
00068 }
00069 #endif /* NDEBUG */
```

16.643 l4/util/assert.h File Reference

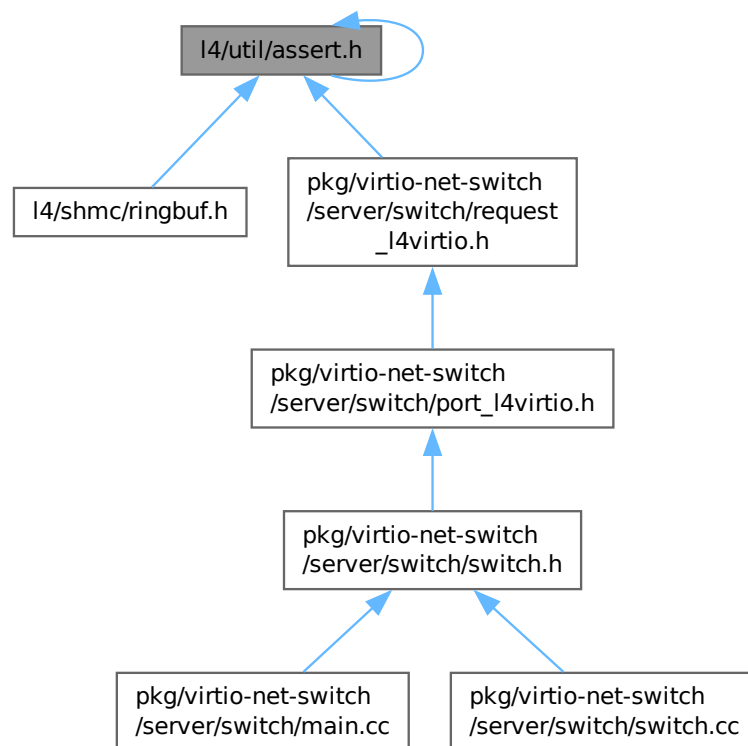
Some useful assert-style macros.

```
#include <stdio.h>
#include <assert.h>
```

Include dependency graph for assert.h:



This graph shows which files directly or indirectly include this file:



16.643.1 Detailed Description

Some useful assert-style macros.

Date

09/2009

Author

Bjoern Doebel doebel@tudos.org

Definition in file [assert.h](#).

16.644 assert.h

[Go to the documentation of this file.](#)

```

00001  /*****
00009  /*
00010  * (c) 2009 Author(s)

```

```

00011 *      economic rights: Technische Universität Dresden (Germany)
00012 * License: see LICENSE.spdx (in this directory or the directories above)
00013 */
00014
00015 /*****
00016 #pragma once
00017
00018 #ifndef NDEBUG
00019
00020 #define DO_NOTHING          do {} while (0)
00021 #define ASSERT_ASSERT(x)    DO_NOTHING
00022 #define ASSERT_VALID(c)     DO_NOTHING
00023 #define ASSERT_EQUAL(a,b)   DO_NOTHING
00024 #define ASSERT_NOT_EQUAL(a,b) DO_NOTHING
00025 #define ASSERT_LOWER_EQ(a,b) DO_NOTHING
00026 #define ASSERT_GREATER_EQ(a,b) DO_NOTHING
00027 #define ASSERT_BETWEEN(a,b,c) DO_NOTHING
00028 #define ASSERT_IPC_OK(i)    DO_NOTHING
00029 #define ASSERT_OK(e)         do { (void)e; } while (0)
00030 #define ASSERT_NOT_NULL(p)   DO_NOTHING
00031 #ifndef assert
00032 #define assert(cond)          DO_NOTHING
00033 #endif
00034
00035 #else // NDEBUG
00036
00037 #ifndef ASSERT_PRINTF
00038 #include <stdio.h>
00039 #define ASSERT_PRINTF printf
00040 #endif
00041 #ifndef ASSERT_ASSERT
00042 #include <assert.h>
00043 #define ASSERT_ASSERT(x) assert(x)
00044 #endif
00045
00046 #define ASSERT_VALID(cap) \
00047     do { \
00048         typeof(cap) _cap = cap; \
00049         if (!l4_is_invalid_cap(_cap)) { \
00050             ASSERT_PRINTF("%s: Cap invalid.\n", __func__); \
00051             ASSERT_ASSERT(!l4_is_invalid_cap(_cap)); \
00052         } \
00053     } while (0)
00054
00055 #define ASSERT_EQUAL(a, b) \
00056     do { \
00057         typeof(a) _a = a; \
00058         typeof(b) _b = b; \
00059         if (_a != _b) { \
00060             ASSERT_PRINTF("%s:\n", __func__); \
00061             ASSERT_PRINTF("    #a" (%lx) != "#b" (%lx)\n", (unsigned long)_a, (unsigned long)_b); \
00062             ASSERT_ASSERT(_a == _b); \
00063         } \
00064     } while (0)
00065
00066 #define ASSERT_NOT_EQUAL(a, b) \
00067     do { \
00068         typeof(a) _a = a; \
00069         typeof(b) _b = b; \
00070         if (_a == _b) { \
00071             ASSERT_PRINTF("%s:\n", __func__); \
00072             ASSERT_PRINTF("    #a" (%lx) == "#b" (%lx)\n", (unsigned long)_a, (unsigned long)_b); \
00073             ASSERT_ASSERT(_a != _b); \
00074         } \
00075     } while (0)
00076
00077 #define ASSERT_LOWER_EQ(val, max) \
00078     do { \
00079         typeof(val) _val = val; \
00080         typeof(max) _max = max; \
00081         if (_val > _max) { \
00082             ASSERT_PRINTF("%s:\n", __func__); \
00083             ASSERT_PRINTF("    #val" (%lx) > "#max" (%lx)\n", (unsigned long)_val, (unsigned long)_max); \
00084             ASSERT_ASSERT(_val <= _max); \
00085         } \
00086     } while (0)
00087
00088 #define ASSERT_GREATER_EQ(val, min) \
00089     do { \
00090         typeof(val) _val = val; \
00091         typeof(min) _min = min; \
00092         if (_val < _min) { \
00093             ASSERT_PRINTF("%s:\n", __func__); \

```

```

00098     ASSERT_PRINTF("    "#val" (%lx) < "#min" (%lx)\n", (unsigned long)_val, (unsigned long)_min); \
00099     ASSERT_ASSERT(_val >= _min); \
00100     } \
00101     } while (0)
00102
00103
00104 #define ASSERT_BETWEEN(val, min, max) \
00105     ASSERT_LOWER_EQ((val), (max)); \
00106     ASSERT_GREATER_EQ((val), (min));
00107
00108
00109 #define ASSERT_IPC_OK(msgtag) \
00110     do { \
00111         int _r = l4_ipc_error(msgtag, l4_utcb()); \
00112         if (_r) { \
00113             ASSERT_PRINTF("%s: IPC Error: %lx\n", __func__, _r); \
00114             ASSERT_ASSERT(_r == 0); \
00115         } \
00116     } while (0)
00117
00118 #define ASSERT_OK(val)          ASSERT_EQUAL((val), 0)
00119 #define ASSERT_NOT_NULL(ptr)    ASSERT_NOT_EQUAL((ptr), (void *)0)
00120
00121 #endif // NDEBUG

```

16.645 atomic.h

```

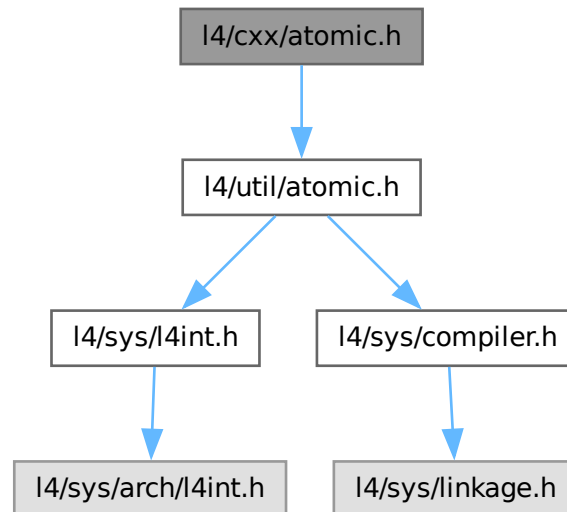
00001
00006 /*
00007  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013 #pragma once
00014
00015 #include <l4/sys/compiler.h>
00016
00017 L4_BEGIN_DECLS
00018
00019 long int
00020 l4_atomic_add(volatile long int* mem, long int offset) L4_NOTHROW L4_LONG_CALL;
00021
00022 long int
00023 l4_atomic_xchg(volatile long int* mem, long int newval) L4_NOTHROW L4_LONG_CALL;
00024
00025 long int
00026 l4_atomic_cmpxchg(volatile long int* mem, long int oldval, long int newval) L4_NOTHROW L4_LONG_CALL;
00027
00028 L4_END_DECLS

```

16.646 l4/cxx/atomic.h File Reference

Atomic template.

```
#include <l4/util/atomic.h>
Include dependency graph for atomic.h:
```



Namespaces

- namespace [L4](#)
[L4](#) low-level kernel interface.

16.646.1 Detailed Description

Atomic template.

Definition in file [atomic.h](#).

16.647 atomic.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2004-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  *
00010  * License: see LICENSE.spdx (in this directory or the directories above)
00011  */
00012 #pragma once
00013
00014 #include <l4/util/atomic.h>
00015
00016 extern "C" void  ____error_compare_and_swap_does_not_support_3_bytes____();
00017 extern "C" void  ____error_compare_and_swap_does_not_support_more_than_4_bytes____();
00018
00019 namespace L4
```

```

00020 {
00021     template< typename X >
00022     inline int compare_and_swap(X volatile *dst, X old_val, X new_val)
00023     {
00024         switch (sizeof(X))
00025         {
00026             case 1:
00027                 return l4util_cmpxchg8((l4_uint8_t volatile*)dst, old_val, new_val);
00028             case 2:
00029                 return l4util_cmpxchg16((l4_uint16_t volatile *)dst, old_val, new_val);
00030             case 3: ____error_compare_and_swap_does_not_support_3_bytes____();
00031             case 4:
00032                 return l4util_cmpxchg32((l4_uint32_t volatile*)dst, old_val, new_val);
00033             default:
00034                 ____error_compare_and_swap_does_not_support_more_than_4_bytes____();
00035         }
00036         return 0;
00037     }
00038 }

```

16.648 l4/util/atomic.h File Reference

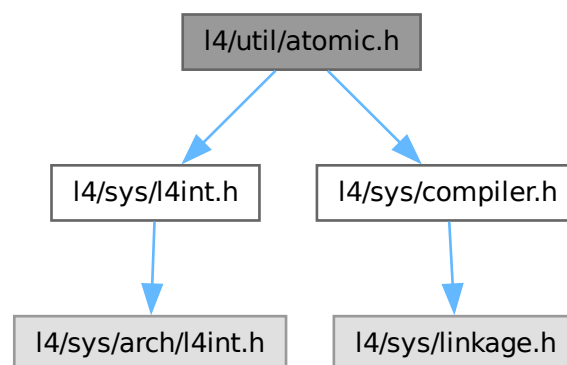
atomic operations header and generic implementations

```

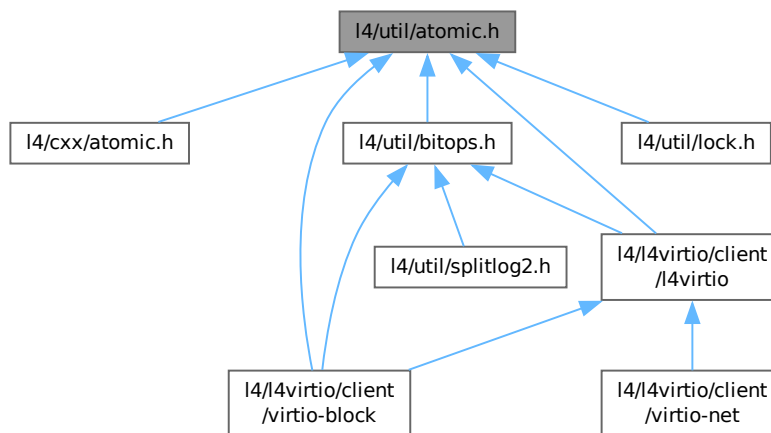
#include <l4/sys/l4int.h>
#include <l4/sys/compiler.h>

```

Include dependency graph for atomic.h:



This graph shows which files directly or indirectly include this file:



Functions

- `int l4util_cmpxchg32 (volatile l4_uint32_t *dest, l4_uint32_t cmp_val, l4_uint32_t new_val)`
Atomic compare and exchange (32 bit version).
- `int l4util_cmpxchg16 (volatile l4_uint16_t *dest, l4_uint16_t cmp_val, l4_uint16_t new_val)`
Atomic compare and exchange (16 bit version).
- `int l4util_cmpxchg8 (volatile l4_uint8_t *dest, l4_uint8_t cmp_val, l4_uint8_t new_val)`
Atomic compare and exchange (8 bit version).
- `int l4util_cmpxchg (volatile l4_umword_t *dest, l4_umword_t cmp_val, l4_umword_t new_val)`
Atomic compare and exchange (machine wide fields).
- `l4_uint32_t l4util_xchg32 (volatile l4_uint32_t *dest, l4_uint32_t val)`
Atomic exchange (32 bit version).
- `l4_uint16_t l4util_xchg16 (volatile l4_uint16_t *dest, l4_uint16_t val)`
Atomic exchange (16 bit version).
- `l4_uint8_t l4util_xchg8 (volatile l4_uint8_t *dest, l4_uint8_t val)`
Atomic exchange (8 bit version).
- `l4_umword_t l4util_xchg (volatile l4_umword_t *dest, l4_umword_t val)`
Atomic exchange (machine wide fields).
- `void l4util_atomic_add (volatile long *dest, long val)`
Atomic add.
- `void l4util_atomic_inc (volatile long *dest)`
Atomic increment.

Atomic add/sub/and/or (8,16,32 bit version) without result

- `void l4util_add8 (volatile l4_uint8_t *dest, l4_uint8_t val)`
- `void l4util_add16 (volatile l4_uint16_t *dest, l4_uint16_t val)`
- `void l4util_add32 (volatile l4_uint32_t *dest, l4_uint32_t val)`
- `void l4util_sub8 (volatile l4_uint8_t *dest, l4_uint8_t val)`
- `void l4util_sub16 (volatile l4_uint16_t *dest, l4_uint16_t val)`
- `void l4util_sub32 (volatile l4_uint32_t *dest, l4_uint32_t val)`
- `void l4util_and8 (volatile l4_uint8_t *dest, l4_uint8_t val)`

- void [l4util_and16](#) (volatile [l4_uint16_t](#) *dest, [l4_uint16_t](#) val)
- void [l4util_and32](#) (volatile [l4_uint32_t](#) *dest, [l4_uint32_t](#) val)
- void [l4util_or8](#) (volatile [l4_uint8_t](#) *dest, [l4_uint8_t](#) val)
- void [l4util_or16](#) (volatile [l4_uint16_t](#) *dest, [l4_uint16_t](#) val)
- void [l4util_or32](#) (volatile [l4_uint32_t](#) *dest, [l4_uint32_t](#) val)

Atomic add/sub/and/or operations (8,16,32 bit) with result

- [l4_uint8_t l4util_add8_res](#) (volatile [l4_uint8_t](#) *dest, [l4_uint8_t](#) val)
- [l4_uint16_t l4util_add16_res](#) (volatile [l4_uint16_t](#) *dest, [l4_uint16_t](#) val)
- [l4_uint32_t l4util_add32_res](#) (volatile [l4_uint32_t](#) *dest, [l4_uint32_t](#) val)
- [l4_uint8_t l4util_sub8_res](#) (volatile [l4_uint8_t](#) *dest, [l4_uint8_t](#) val)
- [l4_uint16_t l4util_sub16_res](#) (volatile [l4_uint16_t](#) *dest, [l4_uint16_t](#) val)
- [l4_uint32_t l4util_sub32_res](#) (volatile [l4_uint32_t](#) *dest, [l4_uint32_t](#) val)
- [l4_uint8_t l4util_and8_res](#) (volatile [l4_uint8_t](#) *dest, [l4_uint8_t](#) val)
- [l4_uint16_t l4util_and16_res](#) (volatile [l4_uint16_t](#) *dest, [l4_uint16_t](#) val)
- [l4_uint32_t l4util_and32_res](#) (volatile [l4_uint32_t](#) *dest, [l4_uint32_t](#) val)
- [l4_uint8_t l4util_or8_res](#) (volatile [l4_uint8_t](#) *dest, [l4_uint8_t](#) val)
- [l4_uint16_t l4util_or16_res](#) (volatile [l4_uint16_t](#) *dest, [l4_uint16_t](#) val)
- [l4_uint32_t l4util_or32_res](#) (volatile [l4_uint32_t](#) *dest, [l4_uint32_t](#) val)

Atomic inc/dec (8,16,32 bit) without result

- void [l4util_inc8](#) (volatile [l4_uint8_t](#) *dest)
- void [l4util_inc16](#) (volatile [l4_uint16_t](#) *dest)
- void [l4util_inc32](#) (volatile [l4_uint32_t](#) *dest)
- void [l4util_dec8](#) (volatile [l4_uint8_t](#) *dest)
- void [l4util_dec16](#) (volatile [l4_uint16_t](#) *dest)
- void [l4util_dec32](#) (volatile [l4_uint32_t](#) *dest)

Atomic inc/dec (8,16,32 bit) with result

- [l4_uint8_t l4util_inc8_res](#) (volatile [l4_uint8_t](#) *dest)
- [l4_uint16_t l4util_inc16_res](#) (volatile [l4_uint16_t](#) *dest)
- [l4_uint32_t l4util_inc32_res](#) (volatile [l4_uint32_t](#) *dest)
- [l4_uint8_t l4util_dec8_res](#) (volatile [l4_uint8_t](#) *dest)
- [l4_uint16_t l4util_dec16_res](#) (volatile [l4_uint16_t](#) *dest)
- [l4_uint32_t l4util_dec32_res](#) (volatile [l4_uint32_t](#) *dest)

16.648.1 Detailed Description

atomic operations header and generic implementations

Date

10/20/2000

Author

Lars Reuther reuther@os.inf.tu-dresden.de, Jork Loeser jork@os.inf.tu-dresden.de

Definition in file [atomic.h](#).

16.649 atomic.h

[Go to the documentation of this file.](#)

```

00001 /*****
00010 */
00011 * (c) 2000-2009 Author(s)
00012 *     economic rights: Technische Universität Dresden (Germany)
00013 * License: see LICENSE.spdx (in this directory or the directories above)
00014 */
00015
00016 /*****
00017 #ifndef __L4UTIL__INCLUDE__ATOMIC_H__
00018 #define __L4UTIL__INCLUDE__ATOMIC_H__
00019
00020 #include <l4/sys/l4int.h>
00021 #include <l4/sys/compiler.h>
00022
00023 /*****
00024 *** Prototypes
00025 *****/
00026
00027 L4_BEGIN_DECLS
00028
00029
00030 #if __SIZEOF_LONG__ == 8
00031
00032 L4_INLINE int
00033 l4util_cmpxchg64(volatile l4_uint64_t * dest,
00034                 l4_uint64_t cmp_val, l4_uint64_t new_val);
00035
00036 #endif
00037
00038 L4_INLINE int
00039 l4util_cmpxchg32(volatile l4_uint32_t * dest,
00040                 l4_uint32_t cmp_val, l4_uint32_t new_val);
00041
00042 L4_INLINE int
00043 l4util_cmpxchg16(volatile l4_uint16_t * dest,
00044                 l4_uint16_t cmp_val, l4_uint16_t new_val);
00045
00046 L4_INLINE int
00047 l4util_cmpxchg8(volatile l4_uint8_t * dest,
00048                l4_uint8_t cmp_val, l4_uint8_t new_val);
00049
00050 L4_INLINE int
00051 l4util_cmpxchg(volatile l4_umword_t * dest,
00052               l4_umword_t cmp_val, l4_umword_t new_val);
00053
00054 L4_INLINE l4_uint32_t
00055 l4util_xchg32(volatile l4_uint32_t * dest, l4_uint32_t val);
00056
00057 L4_INLINE l4_uint16_t
00058 l4util_xchg16(volatile l4_uint16_t * dest, l4_uint16_t val);
00059
00060 L4_INLINE l4_uint8_t
00061 l4util_xchg8(volatile l4_uint8_t * dest, l4_uint8_t val);
00062
00063 L4_INLINE l4_umword_t
00064 l4util_xchg(volatile l4_umword_t * dest, l4_umword_t val);
00065
00066 L4_INLINE void
00067 l4util_add8(volatile l4_uint8_t *dest, l4_uint8_t val);
00068 L4_INLINE void
00069 l4util_add16(volatile l4_uint16_t *dest, l4_uint16_t val);
00070 L4_INLINE void
00071 l4util_add32(volatile l4_uint32_t *dest, l4_uint32_t val);
00072 L4_INLINE void
00073 l4util_sub8(volatile l4_uint8_t *dest, l4_uint8_t val);
00074 L4_INLINE void
00075 l4util_sub16(volatile l4_uint16_t *dest, l4_uint16_t val);
00076 L4_INLINE void
00077 l4util_sub32(volatile l4_uint32_t *dest, l4_uint32_t val);
00078 L4_INLINE void
00079 l4util_and8(volatile l4_uint8_t *dest, l4_uint8_t val);
00080 L4_INLINE void
00081 l4util_and16(volatile l4_uint16_t *dest, l4_uint16_t val);
00082 L4_INLINE void
00083 l4util_and32(volatile l4_uint32_t *dest, l4_uint32_t val);
00084 L4_INLINE void
00085 l4util_or8(volatile l4_uint8_t *dest, l4_uint8_t val);
00086 L4_INLINE void
00087 l4util_or16(volatile l4_uint16_t *dest, l4_uint16_t val);
00088 L4_INLINE void
00089 l4util_or32(volatile l4_uint32_t *dest, l4_uint32_t val);

```

```

00220
00222
00229 L4_INLINE l4_uint8_t
00230 l4util_add8_res(volatile l4_uint8_t *dest, l4_uint8_t val);
00232 L4_INLINE l4_uint16_t
00233 l4util_add16_res(volatile l4_uint16_t *dest, l4_uint16_t val);
00235 L4_INLINE l4_uint32_t
00236 l4util_add32_res(volatile l4_uint32_t *dest, l4_uint32_t val);
00238 L4_INLINE l4_uint8_t
00239 l4util_sub8_res(volatile l4_uint8_t *dest, l4_uint8_t val);
00241 L4_INLINE l4_uint16_t
00242 l4util_sub16_res(volatile l4_uint16_t *dest, l4_uint16_t val);
00244 L4_INLINE l4_uint32_t
00245 l4util_sub32_res(volatile l4_uint32_t *dest, l4_uint32_t val);
00247 L4_INLINE l4_uint8_t
00248 l4util_and8_res(volatile l4_uint8_t *dest, l4_uint8_t val);
00250 L4_INLINE l4_uint16_t
00251 l4util_and16_res(volatile l4_uint16_t *dest, l4_uint16_t val);
00253 L4_INLINE l4_uint32_t
00254 l4util_and32_res(volatile l4_uint32_t *dest, l4_uint32_t val);
00256 L4_INLINE l4_uint8_t
00257 l4util_or8_res(volatile l4_uint8_t *dest, l4_uint8_t val);
00259 L4_INLINE l4_uint16_t
00260 l4util_or16_res(volatile l4_uint16_t *dest, l4_uint16_t val);
00262 L4_INLINE l4_uint32_t
00263 l4util_or32_res(volatile l4_uint32_t *dest, l4_uint32_t val);
00265
00267
00272 L4_INLINE void
00273 l4util_inc8(volatile l4_uint8_t *dest);
00275 L4_INLINE void
00276 l4util_inc16(volatile l4_uint16_t *dest);
00278 L4_INLINE void
00279 l4util_inc32(volatile l4_uint32_t *dest);
00281 L4_INLINE void
00282 l4util_dec8(volatile l4_uint8_t *dest);
00284 L4_INLINE void
00285 l4util_dec16(volatile l4_uint16_t *dest);
00287 L4_INLINE void
00288 l4util_dec32(volatile l4_uint32_t *dest);
00290
00292
00298 L4_INLINE l4_uint8_t
00299 l4util_inc8_res(volatile l4_uint8_t *dest);
00301 L4_INLINE l4_uint16_t
00302 l4util_inc16_res(volatile l4_uint16_t *dest);
00304 L4_INLINE l4_uint32_t
00305 l4util_inc32_res(volatile l4_uint32_t *dest);
00307 L4_INLINE l4_uint8_t
00308 l4util_dec8_res(volatile l4_uint8_t *dest);
00310 L4_INLINE l4_uint16_t
00311 l4util_dec16_res(volatile l4_uint16_t *dest);
00313 L4_INLINE l4_uint32_t
00314 l4util_dec32_res(volatile l4_uint32_t *dest);
00316
00324 L4_INLINE void
00325 l4util_atomic_add(volatile long *dest, long val);
00326
00333 L4_INLINE void
00334 l4util_atomic_inc(volatile long *dest);
00335
00336 L4_END_DECLS
00337
00338 /*****
00339  * IMPLEMENTAION *
00340  *****/
00341
00342 #if __SIZEOF_LONG__ == 8
00343
00344 L4_INLINE int
00345 l4util_cmpxchg64(volatile l4_uint64_t * dest,
00346                  l4_uint64_t cmp_val, l4_uint64_t new_val)
00347 {
00348     return __atomic_compare_exchange_n(dest, &cmp_val, new_val, 0,
00349                                         __ATOMIC_SEQ_CST, __ATOMIC_SEQ_CST);
00350 }
00351
00352 #endif
00353
00354 L4_INLINE int
00355 l4util_cmpxchg32(volatile l4_uint32_t * dest,
00356                  l4_uint32_t cmp_val, l4_uint32_t new_val)
00357 {
00358     return __atomic_compare_exchange_n(dest, &cmp_val, new_val, 0,
00359                                         __ATOMIC_SEQ_CST, __ATOMIC_SEQ_CST);
00360 }
00361

```

```

00362 L4_INLINE int
00363 l4util_cmpxchg16(volatile l4_uint16_t * dest,
00364                  l4_uint16_t cmp_val, l4_uint16_t new_val)
00365 {
00366     return __atomic_compare_exchange_n(dest, &cmp_val, new_val, 0,
00367                                         __ATOMIC_SEQ_CST, __ATOMIC_SEQ_CST);
00368 }
00369
00370 L4_INLINE int
00371 l4util_cmpxchg8(volatile l4_uint8_t * dest,
00372                 l4_uint8_t cmp_val, l4_uint8_t new_val)
00373 {
00374     return __atomic_compare_exchange_n(dest, &cmp_val, new_val, 0,
00375                                         __ATOMIC_SEQ_CST, __ATOMIC_SEQ_CST);
00376 }
00377
00378 L4_INLINE int
00379 l4util_cmpxchg4(volatile l4_umword_t * dest,
00380                 l4_umword_t cmp_val, l4_umword_t new_val)
00381 {
00382     return __atomic_compare_exchange_n(dest, &cmp_val, new_val, 0,
00383                                         __ATOMIC_SEQ_CST, __ATOMIC_SEQ_CST);
00384 }
00385
00386 L4_INLINE l4_uint32_t
00387 l4util_xchg32(volatile l4_uint32_t * dest, l4_uint32_t val)
00388 {
00389     return __atomic_exchange_n(dest, val, __ATOMIC_SEQ_CST);
00390 }
00391
00392 L4_INLINE l4_uint16_t
00393 l4util_xchg16(volatile l4_uint16_t * dest, l4_uint16_t val)
00394 {
00395     return __atomic_exchange_n(dest, val, __ATOMIC_SEQ_CST);
00396 }
00397
00398 L4_INLINE l4_uint8_t
00399 l4util_xchg8(volatile l4_uint8_t * dest, l4_uint8_t val)
00400 {
00401     return __atomic_exchange_n(dest, val, __ATOMIC_SEQ_CST);
00402 }
00403
00404 L4_INLINE l4_umword_t
00405 l4util_xchg4(volatile l4_umword_t * dest, l4_umword_t val)
00406 {
00407     return __atomic_exchange_n(dest, val, __ATOMIC_SEQ_CST);
00408 }
00409
00410 L4_INLINE void
00411 l4util_inc8(volatile l4_uint8_t *dest)
00412 { __atomic_fetch_add(dest, 1, __ATOMIC_SEQ_CST); }
00413
00414 L4_INLINE void
00415 l4util_inc16(volatile l4_uint16_t *dest)
00416 { __atomic_fetch_add(dest, 1, __ATOMIC_SEQ_CST); }
00417
00418 L4_INLINE void
00419 l4util_inc32(volatile l4_uint32_t *dest)
00420 { __atomic_fetch_add(dest, 1, __ATOMIC_SEQ_CST); }
00421
00422 L4_INLINE void
00423 l4util_atomic_inc(volatile long *dest)
00424 { __atomic_fetch_add(dest, 1, __ATOMIC_SEQ_CST); }
00425
00426 L4_INLINE void
00427 l4util_dec8(volatile l4_uint8_t *dest)
00428 { __atomic_fetch_sub(dest, 1, __ATOMIC_SEQ_CST); }
00429
00430 L4_INLINE void
00431 l4util_dec16(volatile l4_uint16_t *dest)
00432 { __atomic_fetch_sub(dest, 1, __ATOMIC_SEQ_CST); }
00433
00434 L4_INLINE void
00435 l4util_dec32(volatile l4_uint32_t *dest)
00436 { __atomic_fetch_sub(dest, 1, __ATOMIC_SEQ_CST); }
00437
00438
00439 L4_INLINE l4_uint8_t
00440 l4util_inc8_res(volatile l4_uint8_t *dest)
00441 { return __atomic_add_fetch(dest, 1, __ATOMIC_SEQ_CST); }
00442
00443 L4_INLINE l4_uint16_t
00444 l4util_inc16_res(volatile l4_uint16_t *dest)
00445 { return __atomic_add_fetch(dest, 1, __ATOMIC_SEQ_CST); }
00446
00447 L4_INLINE l4_uint32_t
00448 l4util_inc32_res(volatile l4_uint32_t *dest)

```

```

00449 { return __atomic_add_fetch(dest, 1, __ATOMIC_SEQ_CST); }
00450
00451 L4_INLINE l4_uint8_t
00452 l4util_dec8_res(volatile l4_uint8_t *dest)
00453 { return __atomic_sub_fetch(dest, 1, __ATOMIC_SEQ_CST); }
00454
00455 L4_INLINE l4_uint16_t
00456 l4util_dec16_res(volatile l4_uint16_t *dest)
00457 { return __atomic_sub_fetch(dest, 1, __ATOMIC_SEQ_CST); }
00458
00459 L4_INLINE l4_uint32_t
00460 l4util_dec32_res(volatile l4_uint32_t *dest)
00461 { return __atomic_sub_fetch(dest, 1, __ATOMIC_SEQ_CST); }
00462
00463 L4_INLINE l4_umword_t
00464 l4util_dec_res(volatile l4_umword_t *dest)
00465 { return __atomic_sub_fetch(dest, 1, __ATOMIC_SEQ_CST); }
00466
00467 L4_INLINE void
00468 l4util_add8(volatile l4_uint8_t *dest, l4_uint8_t val)
00469 { __atomic_fetch_add(dest, val, __ATOMIC_SEQ_CST); }
00470
00471 L4_INLINE void
00472 l4util_add16(volatile l4_uint16_t *dest, l4_uint16_t val)
00473 { __atomic_fetch_add(dest, val, __ATOMIC_SEQ_CST); }
00474
00475 L4_INLINE void
00476 l4util_add32(volatile l4_uint32_t *dest, l4_uint32_t val)
00477 { __atomic_fetch_add(dest, val, __ATOMIC_SEQ_CST); }
00478
00479 L4_INLINE void
00480 l4util_atomic_add(volatile long *dest, long val)
00481 { __atomic_fetch_add(dest, val, __ATOMIC_SEQ_CST); }
00482
00483 L4_INLINE void
00484 l4util_sub8(volatile l4_uint8_t *dest, l4_uint8_t val)
00485 { __atomic_fetch_sub(dest, val, __ATOMIC_SEQ_CST); }
00486
00487 L4_INLINE void
00488 l4util_sub16(volatile l4_uint16_t *dest, l4_uint16_t val)
00489 { __atomic_fetch_sub(dest, val, __ATOMIC_SEQ_CST); }
00490
00491 L4_INLINE void
00492 l4util_sub32(volatile l4_uint32_t *dest, l4_uint32_t val)
00493 { __atomic_fetch_sub(dest, val, __ATOMIC_SEQ_CST); }
00494
00495 L4_INLINE void
00496 l4util_and8(volatile l4_uint8_t *dest, l4_uint8_t val)
00497 { __atomic_fetch_and(dest, val, __ATOMIC_SEQ_CST); }
00498
00499 L4_INLINE void
00500 l4util_and16(volatile l4_uint16_t *dest, l4_uint16_t val)
00501 { __atomic_fetch_and(dest, val, __ATOMIC_SEQ_CST); }
00502
00503 L4_INLINE void
00504 l4util_and32(volatile l4_uint32_t *dest, l4_uint32_t val)
00505 { __atomic_fetch_and(dest, val, __ATOMIC_SEQ_CST); }
00506
00507 L4_INLINE void
00508 l4util_or8(volatile l4_uint8_t *dest, l4_uint8_t val)
00509 { __atomic_fetch_or(dest, val, __ATOMIC_SEQ_CST); }
00510
00511 L4_INLINE void
00512 l4util_or16(volatile l4_uint16_t *dest, l4_uint16_t val)
00513 { __atomic_fetch_or(dest, val, __ATOMIC_SEQ_CST); }
00514
00515 L4_INLINE void
00516 l4util_or32(volatile l4_uint32_t *dest, l4_uint32_t val)
00517 { __atomic_fetch_or(dest, val, __ATOMIC_SEQ_CST); }
00518
00519 L4_INLINE l4_uint8_t
00520 l4util_add8_res(volatile l4_uint8_t *dest, l4_uint8_t val)
00521 { return __atomic_add_fetch(dest, val, __ATOMIC_SEQ_CST); }
00522
00523 L4_INLINE l4_uint16_t
00524 l4util_add16_res(volatile l4_uint16_t *dest, l4_uint16_t val)
00525 { return __atomic_add_fetch(dest, val, __ATOMIC_SEQ_CST); }
00526
00527 L4_INLINE l4_uint32_t
00528 l4util_add32_res(volatile l4_uint32_t *dest, l4_uint32_t val)
00529 { return __atomic_add_fetch(dest, val, __ATOMIC_SEQ_CST); }
00530
00531 L4_INLINE l4_uint8_t
00532 l4util_sub8_res(volatile l4_uint8_t *dest, l4_uint8_t val)
00533 { return __atomic_sub_fetch(dest, val, __ATOMIC_SEQ_CST); }
00534
00535 L4_INLINE l4_uint16_t

```

```

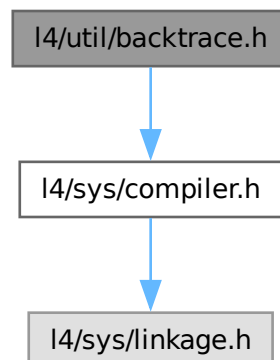
00536 l4util_sub16_res(volatile l4_uint16_t *dest, l4_uint16_t val)
00537 { return __atomic_sub_fetch(dest, val, __ATOMIC_SEQ_CST); }
00538
00539 L4_INLINE l4_uint32_t
00540 l4util_sub32_res(volatile l4_uint32_t *dest, l4_uint32_t val)
00541 { return __atomic_sub_fetch(dest, val, __ATOMIC_SEQ_CST); }
00542
00543 L4_INLINE l4_uint8_t
00544 l4util_and8_res(volatile l4_uint8_t *dest, l4_uint8_t val)
00545 { return __atomic_and_fetch(dest, val, __ATOMIC_SEQ_CST); }
00546
00547 L4_INLINE l4_uint16_t
00548 l4util_and16_res(volatile l4_uint16_t *dest, l4_uint16_t val)
00549 { return __atomic_and_fetch(dest, val, __ATOMIC_SEQ_CST); }
00550
00551 L4_INLINE l4_uint32_t
00552 l4util_and32_res(volatile l4_uint32_t *dest, l4_uint32_t val)
00553 { return __atomic_and_fetch(dest, val, __ATOMIC_SEQ_CST); }
00554
00555 L4_INLINE l4_uint8_t
00556 l4util_or8_res(volatile l4_uint8_t *dest, l4_uint8_t val)
00557 { return __atomic_or_fetch(dest, val, __ATOMIC_SEQ_CST); }
00558
00559 L4_INLINE l4_uint16_t
00560 l4util_or16_res(volatile l4_uint16_t *dest, l4_uint16_t val)
00561 { return __atomic_or_fetch(dest, val, __ATOMIC_SEQ_CST); }
00562
00563 L4_INLINE l4_uint32_t
00564 l4util_or32_res(volatile l4_uint32_t *dest, l4_uint32_t val)
00565 { return __atomic_or_fetch(dest, val, __ATOMIC_SEQ_CST); }
00566
00567 #endif /* ! __L4UTIL__INCLUDE__ATOMIC_H__ */

```

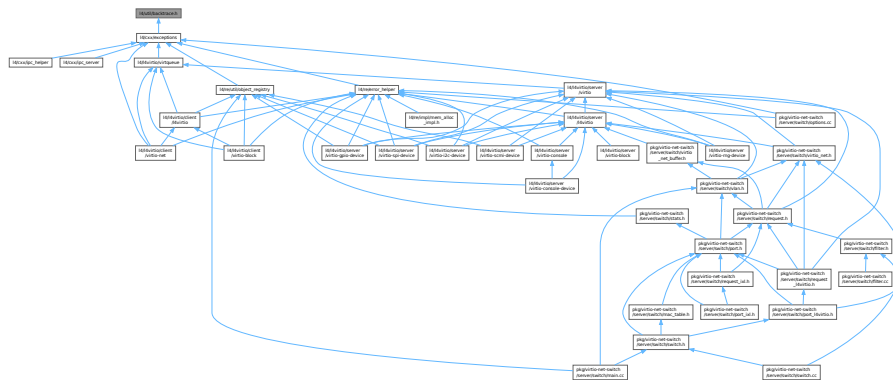
16.650 l4/util/backtrace.h File Reference

Backtrace.

#include <l4/sys/compiler.h>
 Include dependency graph for backtrace.h:



This graph shows which files directly or indirectly include this file:



Functions

- [L4_BEGIN_DECLS](#) `int l4util_backtrace (void **pc_array, int max_len)`
Fill backtrace structure.

16.650.1 Detailed Description

Backtrace.

Definition in file [backtrace.h](#).

16.650.2 Function Documentation

16.650.2.1 l4util_backtrace()

```
L4_BEGIN_DECLS int l4util_backtrace (
    void ** pc_array,
    int max_len)
```

Fill backtrace structure.

Parameters

| | |
|-----------------|--------------------------------|
| <i>pc_array</i> | Array of instruction pointers. |
| <i>max_len</i> | Length of array. |

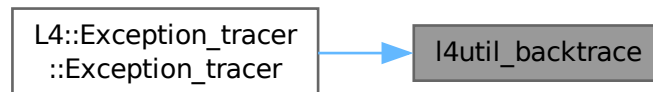
Returns

Number of entries

References [L4_END_DECLS](#).

Referenced by [L4::Exception_tracer::Exception_tracer\(\)](#).

Here is the caller graph for this function:



16.651 backtrace.h

[Go to the documentation of this file.](#)

```

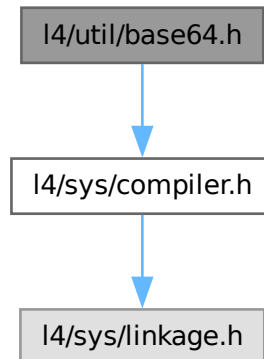
00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011 #pragma once
00012
00013 #include <l4/sys/compiler.h>
00014
00015 L4\_BEGIN\_DECLS
00016
00024 int l4util_backtrace(void **pc_array, int max_len);
00025
00026 L4\_END\_DECLS
  
```

16.652 l4/util/base64.h File Reference

base 64 encoding and decoding functions adapted from Bob Trower 08/04/01


```
#include <l4/sys/compiler.h>
```

Include dependency graph for base64.h:



Functions

- void **base64_encode** (const char *infile, unsigned int in_size, char **outfile)
base-64-encode string infile
- void **base64_decode** (const char *infile, unsigned int in_size, char **outfile)
decode base-64-encoded string infile

16.652.1 Detailed Description

base 64 encoding and decoding functions adapted from Bob Trower 08/04/01

Date

04/26/2002

Author

Joerg Nothnagel jn6@os.inf.tu-dresden.de

Definition in file [base64.h](#).

16.653 base64.h

[Go to the documentation of this file.](#)

```

00001
00009 /*
00010  * (c) 2008-2009 Author(s)
00011  *     economic rights: Technische Universität Dresden (Germany)
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014
00015 #ifndef B64_EN_DECODE
00016 #define B64_EN_DECODE
00017
00018 #include <l4/sys/compiler.h>
00019
00020 L4_BEGIN_DECLS
00021
00027
00038 L4_CV void base64_encode( const char *infile, unsigned int in_size, char **outfile);
00039
00050 L4_CV void base64_decode(const char *infile, unsigned int in_size, char **outfile);
00051
00052 L4_END_DECLS
00053
00055 #endif //B64_EN_DECODE

```

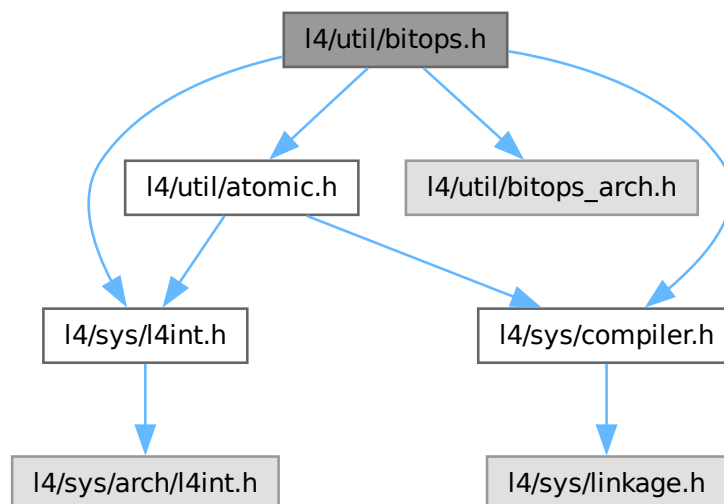
16.654 l4/util/bitops.h File Reference

bit manipulation functions

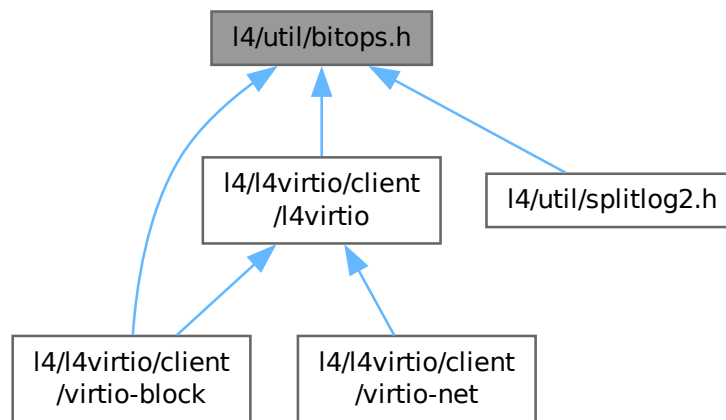
```

#include <l4/sys/l4int.h>
#include <l4/sys/compiler.h>
#include <l4/util/bitops_arch.h>
#include <l4/util/atomic.h>
Include dependency graph for bitops.h:

```



This graph shows which files directly or indirectly include this file:



Macros

- `#define l4util_test_and_clear_bit(b, dest)`
define some more usual names

Functions

- void `l4util_set_bit` (int b, volatile `l4_umword_t` *dest)
Set bit in memory.
- void `l4util_clear_bit` (int b, volatile `l4_umword_t` *dest)
Clear bit in memory.
- void `l4util_complement_bit` (int b, volatile `l4_umword_t` *dest)
Complement bit in memory.
- int `l4util_test_bit` (int b, const volatile `l4_umword_t` *dest)
Test bit (return value of bit).
- int `l4util_bts` (int b, volatile `l4_umword_t` *dest)
Bit test and set.
- int `l4util_btr` (int b, volatile `l4_umword_t` *dest)
Bit test and reset.
- int `l4util_btc` (int b, volatile `l4_umword_t` *dest)
Bit test and complement.
- int `l4util_bsr` (`l4_umword_t` word)
Bit scan reverse.
- int `l4util_bsf` (`l4_umword_t` word)
Bit scan forward.
- int `l4util_find_first_set_bit` (const void *dest, `l4_size_t` size)
Find the first set bit in a memory region.
- int `l4util_find_first_zero_bit` (const void *dest, `l4_size_t` size)
Find the first zero bit in a memory region.
- int `l4util_next_power2` (unsigned long val)
Find the next power of 2 for a given number.

16.654.1 Detailed Description

bit manipulation functions

Date

07/03/2001

Author

Lars Reuther reuther@os.inf.tu-dresden.de

Definition in file [bitops.h](#).

16.655 bitops.h

[Go to the documentation of this file.](#)

```

00001  /*****
00009  */
00010  * (c) 2000-2009 Author(s)
00011  *     economic rights: Technische Universität Dresden (Germany)
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014
00015  /*****
00016  #ifndef __L4UTIL__INCLUDE__BITOPS_H__
00017  #define __L4UTIL__INCLUDE__BITOPS_H__
00018
00019  /* L4 includes */
00020  #include <l4/sys/l4int.h>
00021  #include <l4/sys/compiler.h>
00022
00024  #define l4util_test_and_clear_bit(b, dest)  l4util_btr(b, dest)
00025  #define l4util_test_and_set_bit(b, dest)    l4util_bts(b, dest)
00026  #define l4util_test_and_change_bit(b, dest) l4util_btc(b, dest)
00027  #define l4util_log2(word)                  l4util_bsr(word)
00028
00029  /*****
00030  *** Prototypes
00031  *****/
00032
00033  L4_BEGIN_DECLS
00034
00039
00047  L4_INLINE void
00048  l4util_set_bit(int b, volatile l4_umword_t * dest);
00049
00057  L4_INLINE void
00058  l4util_clear_bit(int b, volatile l4_umword_t * dest);
00059
00067  L4_INLINE void
00068  l4util_complement_bit(int b, volatile l4_umword_t * dest);
00069
00079  L4_INLINE int
00080  l4util_test_bit(int b, const volatile l4_umword_t * dest);
00081
00093  L4_INLINE int
00094  l4util_bts(int b, volatile l4_umword_t * dest);
00095
00107  L4_INLINE int
00108  l4util_btr(int b, volatile l4_umword_t * dest);
00109
00121  L4_INLINE int
00122  l4util_btc(int b, volatile l4_umword_t * dest);
00123
00135  L4_INLINE int
00136  l4util_bsr(l4_umword_t word);
00137
00149  L4_INLINE int
00150  l4util_bsf(l4_umword_t word);
00151
00163  L4_INLINE int

```

```

00164 l4util_find_first_set_bit(const void * dest, l4_size_t size);
00165
00177 L4_INLINE int
00178 l4util_find_first_zero_bit(const void * dest, l4_size_t size);
00179
00180
00189 L4_INLINE int
00190 l4util_next_power2(unsigned long val);
00191
00192 L4_END_DECLS
00193
00194 /*****
00195 *** Implementation of specific version
00196 *****/
00197
00198 #include <l4/util/bitops_arch.h>
00199
00200 /*****
00201 *** Generic implementations
00202 *****/
00203
00204 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_SET_BIT
00205 #include <l4/util/atomic.h>
00206 L4_INLINE void
00207 l4util_set_bit(int b, volatile l4_umword_t * dest)
00208 {
00209     l4_umword_t oldval, newval;
00210
00211     dest += b / (sizeof(*dest) * 8); /* advance dest to the proper element */
00212     b    &= sizeof(*dest) * 8 - 1; /* modulo; cut off all upper bits */
00213
00214     do
00215     {
00216         oldval = *dest;
00217         newval = oldval | (1UL << b);
00218     }
00219     while (!l4util_cmpxchg(dest, oldval, newval));
00220 }
00221 #endif
00222
00223 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_CLEAR_BIT
00224 #include <l4/util/atomic.h>
00225 L4_INLINE void
00226 l4util_clear_bit(int b, volatile l4_umword_t * dest)
00227 {
00228     l4_umword_t oldval, newval;
00229
00230     dest += b / (sizeof(*dest) * 8);
00231     b    &= sizeof(*dest) * 8 - 1;
00232
00233     do
00234     {
00235         oldval = *dest;
00236         newval = oldval & ~(1UL << b);
00237     }
00238     while (!l4util_cmpxchg(dest, oldval, newval));
00239 }
00240 #endif
00241
00242 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_TEST_BIT
00243 L4_INLINE int
00244 l4util_test_bit(int b, const volatile l4_umword_t * dest)
00245 {
00246     dest += b / (sizeof(*dest) * 8);
00247     b    &= sizeof(*dest) * 8 - 1;
00248
00249     return (*dest >> b) & 1;
00250 }
00251 #endif
00252
00253 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_SET
00254 #include <l4/util/atomic.h>
00255 L4_INLINE int
00256 l4util_bts(int b, volatile l4_umword_t * dest)
00257 {
00258     l4_umword_t oldval, newval;
00259
00260     dest += b / (sizeof(*dest) * 8);
00261     b    &= sizeof(*dest) * 8 - 1;
00262
00263     do
00264     {
00265         oldval = *dest;
00266         newval = oldval | (1UL << b);
00267     }
00268     while (!l4util_cmpxchg(dest, oldval, newval));
00269

```

```

00270  /* Return old bit */
00271  return (oldval >> b) & 1;
00272 }
00273 #endif
00274
00275 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_RESET
00276 #include <l4/util/atomic.h>
00277 L4_INLINE int
00278 l4util_btr(int b, volatile l4_umword_t * dest)
00279 {
00280     l4_umword_t oldval, newval;
00281
00282     dest += b / (sizeof(*dest) * 8);
00283     b    &= sizeof(*dest) * 8 - 1;
00284
00285     do
00286     {
00287         oldval = *dest;
00288         newval = oldval & ~(1UL << b);
00289     }
00290     while (!l4util_cmpxchg(dest, oldval, newval));
00291
00292     /* Return old bit */
00293     return (oldval >> b) & 1;
00294 }
00295 #endif
00296
00297 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_BIT_SCAN_REVERSE
00298 L4_INLINE int
00299 l4util_bsr(l4_umword_t word)
00300 {
00301     int i;
00302
00303     if (!word)
00304         return -1;
00305
00306     for (i = 8 * sizeof(word) - 1; i >= 0; i--)
00307         if ((1UL << i) & word)
00308             return i;
00309
00310     __builtin_unreachable();
00311 }
00312 #endif
00313
00314 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_BIT_SCAN_FORWARD
00315 L4_INLINE int
00316 l4util_bsf(l4_umword_t word)
00317 {
00318     unsigned int i;
00319
00320     if (!word)
00321         return -1;
00322
00323     for (i = 0; i < sizeof(word) * 8; i++)
00324         if ((1UL << i) & word)
00325             return i;
00326
00327     __builtin_unreachable();
00328 }
00329 #endif
00330
00331 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_FIND_FIRST_ZERO_BIT
00332 L4_INLINE int
00333 l4util_find_first_zero_bit(const void * dest, l4_size_t size)
00334 {
00335     l4_size_t i, j;
00336     unsigned long *v = (unsigned long*)dest;
00337
00338     if (!size)
00339         return 0;
00340
00341     size = (size + 31) & ~0x1f; /* Grmbl: adapt to x86 implementation... */
00342
00343     for (i = j = 0; i < size; i++, j++)
00344     {
00345         if (j >= sizeof(*v) * 8)
00346         {
00347             j = 0;
00348             v++;
00349         }
00350         if (!((1UL << j) & *v))
00351             return i;
00352     }
00353     return size + 1;
00354 }
00355 #endif
00356

```

```

00357 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_COMPLEMENT_BIT
00358 L4_INLINE void
00359 l4util_complement_bit(int b, volatile l4_umword_t * dest)
00360 {
00361     dest += b / (sizeof(*dest) * 8);
00362     b    &= sizeof(*dest) * 8 - 1;
00363
00364     *dest ^= 1UL << b;
00365 }
00366 #endif
00367
00368 /*
00369  * Adapted from:
00370  * http://en.wikipedia.org/wiki/Power\_of\_two#Algorithm\_to\_find\_the\_next-highest\_power\_of\_two
00371  */
00372 L4_INLINE int
00373 l4util_next_power2(unsigned long val)
00374 {
00375     unsigned i;
00376
00377     if (val == 0)
00378         return 1;
00379
00380     val--;
00381     for (i=1; i < sizeof(unsigned long)*8; i<=1)
00382         val = val | val >> i;
00383
00384     return val+1;
00385 }
00386
00387
00388 /* Non-implemented version, catch with a linker warning */
00389
00390 extern int __this_l4util_bitops_function_is_not_implemented_for_this_arch__sorry(void);
00391
00392 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_COMPLEMENT
00393 L4_INLINE int
00394 l4util_btc(int b, volatile l4_umword_t * dest)
00395 { (void)b; (void)dest; __this_l4util_bitops_function_is_not_implemented_for_this_arch__sorry(); return
00396 0; }
00397 #endif
00398
00399 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_FIND_FIRST_SET_BIT
00400 L4_INLINE int
00400 l4util_find_first_set_bit(const void * dest, l4_size_t size)
00401 { (void)dest; (void)size; __this_l4util_bitops_function_is_not_implemented_for_this_arch__sorry();
00402 return 0; }
00403 #endif
00404 #endif /* ! __L4UTIL__INCLUDE__BITOPS_H__ */

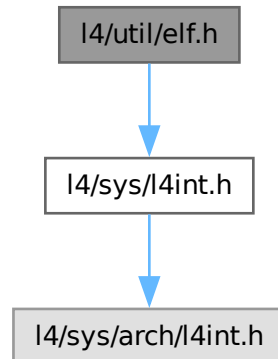
```

16.656 l4/util/elf.h File Reference

ELF definition.

```
#include <l4/sys/l4int.h>
```

Include dependency graph for elf.h:



Data Structures

- struct [Elf32_Ehdr](#)
ELF32 header.
- struct [Elf64_Ehdr](#)
ELF64 header.
- struct [Elf32_Shdr](#)
ELF32 section header.
- struct [Elf64_Shdr](#)
ELF64 section header.
- struct [Elf32_Phdr](#)
ELF32 program header.
- struct [Elf64_Phdr](#)
ELF64 program header.
- struct [Elf32_Dyn](#)
ELF32 dynamic entry.
- struct [Elf64_Dyn](#)
ELF64 dynamic entry.
- struct [Elf32_Rel](#)
ELF32 relocation entry w/o addend.
- struct [Elf32_Rela](#)
ELF32 relocation entry w/ addend.
- struct [Elf64_Rel](#)
ELF64 relocation entry w/o addend.
- struct [Elf64_Rela](#)
ELF64 relocation entry w/ addend.
- struct [Elf32_Sym](#)
ELF32 symbol table entry.
- struct [Elf64_Sym](#)
ELF64 symbol table entry.

- struct [Elf32_Auxv](#)
Auxiliary vector (32-bit).
- struct [Elf64_Auxv](#)
Auxiliary vector (64-bit).

Macros

- #define **ElfW**(type)
Use 64 or 32 bits types depending on the target architecture.
- #define **ELF32_R_SYM**(i)
Symbol table index.
- #define **ELF32_R_TYPE**(i)
- #define **ELF32_R_INFO**(s, t)
Create info from symbol table index + type.
- #define **ELF64_R_SYM**(i)
Symbol table index.
- #define **ELF64_R_TYPE**(i)
- #define **ELF64_R_INFO**(s, t)
Create info from symbol table index + type.
- #define **ELF32_ST_BIND**(i)
- #define **ELF32_ST_TYPE**(i)
- #define **ELF32_ST_INFO**(b, t)
Make info from bind + type.
- #define **ELF64_ST_BIND**(i)
- #define **ELF64_ST_TYPE**(i)
- #define **ELF64_ST_INFO**(b, t)
Make info from bind + type.

Typedefs

- typedef struct Elf32_Auxv **Elf32_Auxv**
Auxiliary vector (32-bit).
- typedef struct Elf64_Auxv **Elf64_Auxv**
Auxiliary vector (64-bit).

ELF types

- typedef [I4_uint32_t](#) **Elf32_Addr**
size 4 align 4
- typedef [I4_uint32_t](#) **Elf32_Off**
size 4 align 4
- typedef [I4_uint16_t](#) **Elf32_Half**
size 2 align 2
- typedef [I4_uint32_t](#) **Elf32_Word**
size 4 align 4
- typedef [I4_int32_t](#) **Elf32_Sword**
size 4 align 4
- typedef [I4_uint64_t](#) **Elf64_Addr**
size 8 align 8
- typedef [I4_uint64_t](#) **Elf64_Off**
size 8 align 8
- typedef [I4_uint16_t](#) **Elf64_Half**

- size 2 align 2*
- typedef `l4_uint32_t` `Elf64_Word`
- size 4 align 4*
- typedef `l4_int32_t` `Elf64_Sword`
- size 4 align 4*
- typedef `l4_uint64_t` `Elf64_Xword`
- size 8 align 8*
- typedef `l4_int64_t` `Elf64_Sxword`
- size 8 align 8*

Enumerations

- enum { `EI_NIDENT` = 16 }
- enum `Elf_ETs` {
`ET_NONE` = 0 , `ET_REL` = 1 , `ET_EXEC` = 2 , `ET_DYN` = 3 ,
`ET_CORE` = 4 , `ET_LOPROC` = 0xff00 , `ET_HIPROC` = 0xffff }
- Object file type.*
- enum `Elf_EMs` {
`EM_NONE` = 0 , `EM_M32` = 1 , `EM_SPARC` = 2 , `EM_386` = 3 ,
`EM_68K` = 4 , `EM_88K` = 5 , `EM_860` = 7 , `EM_MIPS` = 8 ,
`EM_MIPS_RS4_BE` = 10 , `EM_SPARC64` = 11 , `EM_PARISC` = 15 , `EM_VPP500` = 17 ,
`EM_SPARC32PLUS` = 18 , `EM_960` = 19 , `EM_PPC` = 20 , `EM_V800` = 36 ,
`EM_FR20` = 37 , `EM_RH32` = 38 , `EM_RCE` = 39 , `EM_ARM` = 40 ,
`EM_ALPHA` = 41 , `EM_SH` = 42 , `EM_SPARCV9` = 43 , `EM_TRICORE` = 44 ,
`EM_ARC` = 45 , `EM_H8_300` = 46 , `EM_H8_300H` = 47 , `EM_H8S` = 48 ,
`EM_H8_500` = 49 , `EM_IA_64` = 50 , `EM_MIPS_X` = 51 , `EM_COLDFIRE` = 52 ,
`EM_68HC12` = 53 , `EM_X86_64` = 62 , `EM_PDSP` = 63 , `EM_FX66` = 66 ,
`EM_ST9PLUS` = 67 , `EM_ST7` = 68 , `EM_68HC16` = 69 , `EM_68HC11` = 70 ,
`EM_68HC08` = 71 , `EM_68HC05` = 72 , `EM_SVX` = 73 , `EM_ST19` = 74 ,
`EM_VAX` = 75 , `EM_CRIS` = 76 , `EM_JAVELIN` = 77 , `EM_FIREPATH` = 78 ,
`EM_ZSP` = 79 , `EM_MMIX` = 80 , `EM_HUANY` = 81 , `EM_PRISM` = 82 ,
`EM_AVR` = 83 , `EM_FR30` = 84 , `EM_D10V` = 85 , `EM_D30V` = 86 ,
`EM_V850` = 87 , `EM_M32R` = 88 , `EM_MN10300` = 89 , `EM_MN10200` = 90 ,
`EM_PJ` = 91 , `EM_OPENRISC` = 92 , `EM_ARC_A5` = 93 , `EM_XTENSA` = 94 ,
`EM_ALTERA_NIOS2` = 113 , `EM_AARCH64` = 183 , `EM_TILEPRO` = 188 , `EM_MICROBLAZE` = 189 ,
`EM_TILEGX` = 191 , `EM_RISCV` = 243 , `EM_NUM` = 244 }
- Required architecture.*
- enum `Elf_EVs` { `EV_NONE` = 0 , `EV_CURRENT` = 1 }
- Object file version.*
- enum `Elf_EIs` {
`EI_MAG0` = 0 , `EI_MAG1` = 1 , `EI_MAG2` = 2 , `EI_MAG3` = 3 ,
`EI_CLASS` = 4 , `EI_DATA` = 5 , `EI_VERSION` = 6 , `EI_OSABI` = 7 ,
`EI_ABIVERSION` = 8 , `EI_PAD` = 9 }
- Identification Indices.*
- enum `Elf_MAGs` { `ELFMAG0` = 0x7f , `ELFMAG1` = 'E' , `ELFMAG2` = 'L' , `ELFMAG3` = 'F' }
- Magic number.*
- enum `Elf_CIASSs` { `ELFCLASSNONE` = 0 , `ELFCLASS32` = 1 , `ELFCLASS64` = 2 , `ELFCLASSNUM` = 3 }
- File class or capacity.*
- enum `Elf_DATAs` { `ELFDATANONE` = 0 , `ELFDATA2LSB` = 1 , `ELFDATA2MSB` = 2 , `ELFDATANUM` = 3 }
- Data encoding.*
- enum `Elf_OSABIs` {
`ELFOSABI_NONE` = 0 , `ELFOSABI_SYSV` = 0 , `ELFOSABI_HPUX` = 1 , `ELFOSABI_NETBSD` = 2 ,
`ELFOSABI_LINUX` = 3 , `ELFOSABI_SOLARIS` = 6 , `ELFOSABI_AIX` = 7 , `ELFOSABI_IRIX` = 8 ,
`ELFOSABI_FREEBSD` = 9 , `ELFOSABI_TRU64` = 10 , `ELFOSABI_MODESTO` = 11 , `ELFOSABI_OPENBSD`
= 12 ,
`ELFOSABI_ARM` = 97 , `ELFOSABI_STANDALONE` = 255 }

Identify operating system and ABI to which the object is targeted.

- enum `Elf_SHNs` {
`SHN_UNDEF` = 0 , `SHN_LORESERVE` = 0xff00 , `SHN_LOPROC` = 0xff00 , `SHN_HIPROC` = 0xff1f ,
`SHN_ABS` = 0xffff , `SHN_COMMON` = 0xffff2 , `SHN_HIRESERVE` = 0xffff }

Special section indexes.

- enum `Elf_SHTs` {
`SHT_NULL` = 0 , `SHT_PROGBITS` = 1 , `SHT_SYMTAB` = 2 , `SHT_STRTAB` = 3 ,
`SHT_RELA` = 4 , `SHT_HASH` = 5 , `SHT_DYNAMIC` = 6 , `SHT_NOTE` = 7 ,
`SHT_NOBITS` = 8 , `SHT_REL` = 9 , `SHT_SHLIB` = 10 , `SHT_DYNSYM` = 11 ,
`SHT_INIT_ARRAY` = 14 , `SHT_FINI_ARRAY` = 15 , `SHT_PREINIT_ARRAY` = 16 , `SHT_GROUP` = 17 ,
`SHT_SYMTAB_SHNDX` = 18 , `SHT_NUM` = 19 , `SHT_LOOS` = 0x60000000 , `SHT_HIOS` = 0x6fffffff ,
`SHT_LOPROC` = 0x70000000 , `SHT_HIPROC` = 0x7fffffff , `SHT_LOUSER` = 0x80000000 , `SHT_HIUSER` =
0xffffffff }

Section type.

- enum `Elf_SHFs` {
`SHF_WRITE` = 0x1 , `SHF_ALLOC` = 0x2 , `SHF_EXECINSTR` = 0x4 , `SHF_MERGE` = 0x10 ,
`SHF_STRINGS` = 0x20 , `SHF_INFO_LINK` = 0x40 , `SHF_LINK_ORDER` = 0x80 , `SHF_OS_NONCONFORMING`
= 0x100 ,
`SHF_GROUP` = 0x200 , `SHF_TLS` = 0x400 , `SHF_MASKOS` = 0x0ff00000 , `SHF_MASKPROC` = 0xf0000000
}

Section attribute flags.

- enum `Elf_PTs` {
`PT_NULL` = 0 , `PT_LOAD` = 1 , `PT_DYNAMIC` = 2 , `PT_INTERP` = 3 ,
`PT_NOTE` = 4 , `PT_SHLIB` = 5 , `PT_PHDR` = 6 , `PT_TLS` = 7 ,
`PT_NUM` = 8 , `PT_LOOS` = 0x60000000 , `PT_HIOS` = 0x6fffffff , `PT_LOPROC` = 0x70000000 ,
`PT_HIPROC` = 0x7fffffff , `PT_GNU_EH_FRAME` = `PT_LOOS` + 0x474e550 , `PT_GNU_STACK` = `PT_LOOS`
+ 0x474e551 , `PT_GNU_RELRO` = `PT_LOOS` + 0x474e552 ,
`PT_L4_STACK` = `PT_LOOS` + 0x12 , `PT_L4_AUX` = `PT_LOOS` + 0x14 }

Segment types.

- enum `Elf_PFs` {
`PF_X` = 0x1 , `PF_W` = 0x2 , `PF_R` = 0x4 , `PF_MASKOS` = 0x0ff00000 ,
`PF_MASKPROC` = 0x7fffffff }

Segment permissions.

- enum `Elf_NT_s_core` {
`NT_PRSTATUS` = 1 , `NT_FPREGSET` = 2 , `NT_PRPSINFO` = 3 , `NT_PRXREG` = 4 ,
`NT_TASKSTRUCT` = 4 , `NT_PLATFORM` = 5 , `NT_AUXV` = 6 , `NT_GWINDOWS` = 7 ,
`NT_ASRS` = 8 , `NT_PSTATUS` = 10 , `NT_PSINFO` = 13 , `NT_PRCRED` = 14 ,
`NT_UTSNAME` = 15 , `NT_LWPSTATUS` = 16 , `NT_LWPSINFO` = 17 , `NT_PRFPXREG` = 20 }

Legal values for note segment descriptor types for core files.

- enum `Elf_NT_s_obj` { `NT_VERSION` = 1 }

Legal values for the note segment descriptor types for object files.

- enum `Elf_DT_s` {
`DT_NULL` = 0 , `DT_NEEDED` = 1 , `DT_PLTRELSZ` = 2 , `DT_PLTGOT` = 3 ,
`DT_HASH` = 4 , `DT_STRTAB` = 5 , `DT_SYMTAB` = 6 , `DT_RELA` = 7 ,
`DT_RELASZ` = 8 , `DT_RELAENT` = 9 , `DT_STRSZ` = 10 , `DT_SYMENT` = 11 ,
`DT_INIT` = 12 , `DT_FINI` = 13 , `DT_SONAME` = 14 , `DT_RPATH` = 15 ,
`DT_SYMBOLIC` = 16 , `DT_REL` = 17 , `DT_RELSZ` = 18 , `DT_RELENT` = 19 ,
`DT_PTRREL` = 20 , `DT_DEBUG` = 21 , `DT_TEXTREL` = 22 , `DT_JMPREL` = 23 ,
`DT_BIND_NOW` = 24 , `DT_INIT_ARRAY` = 25 , `DT_FINI_ARRAY` = 26 , `DT_INIT_ARRAYSZ` = 27 ,
`DT_FINI_ARRAYSZ` = 28 , `DT_RUNPATH` = 29 , `DT_FLAGS` = 30 , `DT_ENCODING` = 32 ,
`DT_PREINIT_ARRAY` = 32 , `DT_PREINIT_ARRAYSZ` = 33 , `DT_NUM` = 34 , `DT_LOOS` = 0x60000000d ,
`DT_HIOS` = 0x6ffff000 , `DT_LOPROC` = 0x70000000 , `DT_HIPROC` = 0x7fffffff }

Dynamic Array Tags.

- enum `Elf_DF_s` {
`DF_ORIGIN` = 0x00000001 , `DF_SYMBOLIC` = 0x00000002 , `DF_TEXTREL` = 0x00000004 ,
`DF_BIND_NOW` = 0x00000008 ,
`DF_STATIC_TLS` = 0x00000010 }

Values of *Elf32_Dyn.d_un.d_val*, *Elf64_Dyn.d_un.d_val* in the *DT_FLAGS* entry.

- enum *Elf_DF_1s* {
DF_1_NOW = 0x00000001 , *DF_1_GLOBAL* = 0x00000002 , *DF_1_GROUP* = 0x00000004 ,
DF_1_NODELETE = 0x00000008 ,
DF_1_LOADFLTR = 0x00000010 , *DF_1_INITFIRST* = 0x00000020 , *DF_1_NOOPEN* = 0x00000040 ,
DF_1_ORIGIN = 0x00000080 ,
DF_1_DIRECT = 0x00000100 , *DF_1_TRANS* = 0x00000200 , *DF_1_INTERPOSE* = 0x00000400 ,
DF_1_NODEFLIB = 0x00000800 ,
DF_1_NODUMP = 0x00001000 , *DF_1_CONFALT* = 0x00002000 , *DF_1_ENDFILTEE* = 0x00004000 ,
DF_1_DISPRELDNE = 0x00008000 ,
DF_1_DISPRELPND = 0x00010000 }

State flags selectable in the *Elf32_Dyn.d_un.d_val* / *Elf64_Dyn.d_un.d_val* element of the *DT_FLAGS_1* entry in the dynamic section.

- enum *Elf_DTF_1s*

Flags for the feature selection in *DT_FEATURE_1*.

- enum *Elf_DF_P1s* { *DF_P1_LAZYLOAD* = 0x00000001 , *DF_P1_GROUPPERM* = 0x00000002 }

Flags in the *DT_POSFLAG_1* entry effecting only the next *DT_** entry.

- enum *Elf_R_386_s* {
R_386_NONE = 0 , *R_386_32* = 1 , *R_386_PC32* = 2 , *R_386_GOT32* = 3 ,
R_386_PLT32 = 4 , *R_386_COPY* = 5 , *R_386_GLOB_DAT* = 6 , *R_386_JMP_SLOT* = 7 ,
R_386_RELATIVE = 8 , *R_386_GOTOFF* = 9 , *R_386_GOTPC* = 10 , *R_386_32PLT* = 11 ,
R_386_TLS_TPOFF = 14 , *R_386_TLS_IE* = 15 , *R_386_TLS_GOTIE* = 16 , *R_386_TLS_LE* = 17 ,
R_386_TLS_GD = 18 , *R_386_TLS_LDM* = 19 , *R_386_16* = 20 , *R_386_PC16* = 21 ,
R_386_8 = 22 , *R_386_PC8* = 23 , *R_386_TLS_GD_32* = 24 , *R_386_TLS_GD_PUSH* = 25 ,
R_386_TLS_GD_CALL = 26 , *R_386_TLS_GD_POP* = 27 , *R_386_TLS_LDM_32* = 28 , *R_386_TLS_LDM_PUSH*
= 29 ,
R_386_TLS_LDM_CALL = 30 , *R_386_TLS_LDM_POP* = 31 , *R_386_TLS_LDO_32* = 32 , *R_386_TLS_IE_32*
= 33 ,
R_386_TLS_LE_32 = 34 , *R_386_TLS_DTPMOD32* = 35 , *R_386_TLS_DTPOFF32* = 36 , *R_386_TLS_TPOFF32*
= 37 ,
R_386_NUM = 38 }

Relocation types (processor specific).

- enum *Elf_EF_ARM_s* { }

ARM specific declarations.

- enum *Elf_STT_ARM_s*

Additional symbol types for Thumb.

- enum *Elf_SHF_s_ARM* { *SHF_ARM_ENTRYSECT* = 0x10000000 , *SHF_ARM_COMDEF* = 0x80000000 }

ARM-specific values for *Elf32_Shdr.sh_flags* / *Elf64_Shdr.sh_flags*.

- enum *Elf_ARM_SBs* { *PF_ARM_SB* = 0x10000000 }

ARM-specific program header flags.

- enum *Elf_R_ARM_s* {
R_ARM_NONE = 0 , *R_ARM_PC24* = 1 , *R_ARM_ABS32* = 2 , *R_ARM_REL32* = 3 ,
R_ARM_PC13 = 4 , *R_ARM_ABS16* = 5 , *R_ARM_ABS12* = 6 , *R_ARM_THM_ABS5* = 7 ,
R_ARM_ABS8 = 8 , *R_ARM_SBREL32* = 9 , *R_ARM_THM_PC22* = 10 , *R_ARM_THM_PC8* = 11 ,
R_ARM_AMP_VCALL9 = 12 , *R_ARM_SWI24* = 13 , *R_ARM_THM_SWI8* = 14 , *R_ARM_XPC25* = 15 ,
R_ARM_THM_XPC22 = 16 , *R_ARM_COPY* = 20 , *R_ARM_GLOB_DAT* = 21 , *R_ARM_JUMP_SLOT* = 22 ,
R_ARM_RELATIVE = 23 , *R_ARM_GOTOFF* = 24 , *R_ARM_GOTPC* = 25 , *R_ARM_GOT32* = 26 ,
R_ARM_PLT32 = 27 , *R_ARM_ALU_PCREL_7_0* = 32 , *R_ARM_ALU_PCREL_15_8* = 33 , *R_ARM_↵*
ALU_PCREL_23_15 = 34 ,
R_ARM_LDR_SBREL_11_0 = 35 , *R_ARM_ALU_SBREL_19_12* = 36 , *R_ARM_ALU_SBREL_27_20* =
37 , *R_ARM_GNU_VTENTRY* = 100 ,
R_ARM_GNU_VTINHERIT = 101 , *R_ARM_THM_PC11* = 102 , *R_ARM_THM_PC9* = 103 , *R_ARM_↵*
RXPC25 = 249 ,
R_ARM_RSBREL32 = 250 , *R_ARM_THM_RPC22* = 251 , *R_ARM_RREL32* = 252 , *R_ARM_RABS22* =
253 ,
R_ARM_RPC24 = 254 , *R_ARM_RBASE* = 255 , *R_ARM_NUM* = 256 }

ARM relocations.

- enum [Elf_R_AARCH64_s](#) { [R_AARCH64_NONE](#) = 0 , [R_AARCH64_RELATIVE](#) = 1027 }

AARCH64 relocations.

- enum [Elf_R_X86_64_s](#) {
[R_X86_64_NONE](#) = 0 , [R_X86_64_64](#) = 1 , [R_X86_64_PC32](#) = 2 , [R_X86_64_GOT32](#) = 3 ,
[R_X86_64_PLT32](#) = 4 , [R_X86_64_COPY](#) = 5 , [R_X86_64_GLOB_DAT](#) = 6 , [R_X86_64_JUMP_SLOT](#) = 7 ,
[R_X86_64_RELATIVE](#) = 8 , [R_X86_64_GOTPCREL](#) = 9 , [R_X86_64_32](#) = 10 , [R_X86_64_32S](#) = 11 ,
[R_X86_64_16](#) = 12 , [R_X86_64_PC16](#) = 13 , [R_X86_64_8](#) = 14 , [R_X86_64_PC8](#) = 15 ,
[R_X86_64_DTPMOD64](#) = 16 , [R_X86_64_DTPOFF64](#) = 17 , [R_X86_64_TPOFF64](#) = 18 , [R_X86_64_TLSGD](#)
= 19 ,
[R_X86_64_TLSLD](#) = 20 , [R_X86_64_DTPOFF32](#) = 21 , [R_X86_64_GOTTPOFF](#) = 22 , [R_X86_64_TPOFF32](#)
= 23 ,
[R_X86_64_NUM](#) = 24 }

AMD x86-64 relocations.

- enum [Elf_STNs](#)

Symbol Table Entry.

- enum [Elf_STBs](#) {
[STB_LOCAL](#) = 0 , [STB_GLOBAL](#) = 1 , [STB_WEAK](#) = 2 , [STB_LOOS](#) = 10 ,
[STB_HIOS](#) = 12 , [STB_LOPROC](#) = 13 , [STB_HIPROC](#) = 15 }

Symbol Binding.

- enum [Elf_STTs](#) {
[STT_NOTYPE](#) = 0 , [STT_OBJECT](#) = 1 , [STT_FUNC](#) = 2 , [STT_SECTION](#) = 3 ,
[STT_FILE](#) = 4 , [STT_LOOS](#) = 10 , [STT_HIOS](#) = 12 , [STT_LOPROC](#) = 13 ,
[STT_HIPROC](#) = 15 }

Symbol Types.

- enum [Elf_ATs](#) {
[AT_NULL](#) = 0 , [AT_IGNORE](#) = 1 , [AT_EXECFD](#) = 2 , [AT_PHDR](#) = 3 ,
[AT_PHENT](#) = 4 , [AT_PHNUM](#) = 5 , [AT_PAGESZ](#) = 6 , [AT_BASE](#) = 7 ,
[AT_FLAGS](#) = 8 , [AT_ENTRY](#) = 9 , [AT_NOTELF](#) = 10 , [AT_UID](#) = 11 ,
[AT_EUID](#) = 12 , [AT_GID](#) = 13 , [AT_EGID](#) = 14 , [AT_L4_AUX](#) = 0xf0 ,
[AT_L4_ENV](#) = 0xf1 , [AT_L4_KIP](#) = 0xf2 }

Legal values for [Elf32_Auxv.atype](#) / [Elf64_Auxv.atype](#).

16.656.1 Detailed Description

ELF definition.

Date

08/18/2000

Author

Frank Mehnert fm3@os.inf.tu-dresden.de Alexander Warg aw11@os.inf.tu-dresden.de

Many structs from "Executable and Linkable Format (ELF)", Portable Formats Specification, Version 1.1 and "↵
System V Application Binary Interface - DRAFT - April 29, 1998" The Santa Cruz Operation, Inc. (see [http↵
://www.sco.com/developer/gabi/contents.html](http://www.sco.com/developer/gabi/contents.html))

Definition in file [elf.h](#).

16.657 elf.h

[Go to the documentation of this file.](#)

```

00001
00019 /*
00020  * (c) 2008-2009 Author(s)
00021  *     economic rights: Technische Universität Dresden (Germany)
00022  * License: see LICENSE.spdx (in this directory or the directories above)
00023  */
00024
00025 /* (c) 2003-2006 Technische Universitaet Dresden
00026  * License: see LICENSE.spdx (in this directory or the directories above) */
00027
00028 #pragma once
00029
00030 #include <l4/sys/l4int.h>
00031
00037
00042 typedef l4_uint32_t      Elf32_Addr;
00043 typedef l4_uint32_t      Elf32_Off;
00044 typedef l4_uint16_t      Elf32_Half;
00045 typedef l4_uint32_t      Elf32_Word;
00046 typedef l4_int32_t       Elf32_Sword;
00047 typedef l4_uint64_t      Elf64_Addr;
00048 typedef l4_uint64_t      Elf64_Off;
00049 typedef l4_uint16_t      Elf64_Half;
00050 typedef l4_uint32_t      Elf64_Word;
00051 typedef l4_int32_t       Elf64_Sword;
00052 typedef l4_uint64_t      Elf64_Xword;
00053 typedef l4_int64_t       Elf64_Sxword;
00055
00060 #if L4_MWORD_BITS == 64
00061 # define ElfW(type)      _ElfW(Elf, 64, type)
00062 #else
00063 # define ElfW(type)      _ElfW(Elf, 32, type)
00064 #endif
00065 #define _ElfW(e,w,t)      __ElfW(e, w, _##t)
00066 #define __ElfW(e,w,t)    e##w##t
00067
00068 #if defined(ARCH_x86)
00069 # define L4_ARCH_EI_DATA      ELFDATA2LSB
00070 # define L4_ARCH_E_MACHINE    EM_386
00071 # define L4_ARCH_EI_CLASS     ELFCLASS32
00072 #elif defined(ARCH_amd64)
00073 # define L4_ARCH_EI_DATA      ELFDATA2LSB
00074 # define L4_ARCH_E_MACHINE    EM_X86_64
00075 # define L4_ARCH_EI_CLASS     ELFCLASS64
00076 #elif defined(ARCH_arm)
00077 # define L4_ARCH_EI_DATA      ELFDATA2LSB
00078 # define L4_ARCH_E_MACHINE    EM_ARM
00079 # define L4_ARCH_EI_CLASS     ELFCLASS32
00080 #elif defined(ARCH_arm64)
00081 # define L4_ARCH_EI_DATA      ELFDATA2LSB
00082 # define L4_ARCH_E_MACHINE    EM_AARCH64
00083 # define L4_ARCH_EI_CLASS     ELFCLASS64
00084 #elif defined(ARCH_ppc32)
00085 # define L4_ARCH_EI_DATA      ELFDATA2MSB
00086 # define L4_ARCH_E_MACHINE    EM_PPC
00087 # define L4_ARCH_EI_CLASS     ELFCLASS32
00088 #elif defined(ARCH_sparc)
00089 # define L4_ARCH_EI_DATA      ELFDATA2MSB
00090 # define L4_ARCH_E_MACHINE    EM_SPARC
00091 # define L4_ARCH_EI_CLASS     ELFCLASS32
00092 #elif defined(ARCH_mips)
00093 # define L4_ARCH_EI_DATA      ELFDATA2LSB
00094 # define L4_ARCH_E_MACHINE    EM_MIPS
00095 # ifdef __mips64
00096 #   define L4_ARCH_EI_CLASS     ELFCLASS64
00097 # else
00098 #   define L4_ARCH_EI_CLASS     ELFCLASS32
00099 # endif
00100 #elif defined(ARCH_riscv)
00101 # define L4_ARCH_EI_DATA      ELFDATA2LSB
00102 # define L4_ARCH_E_MACHINE    EM_RISCV
00103 # if __riscv_xlen == 64
00104 #   define L4_ARCH_EI_CLASS     ELFCLASS64
00105 # else
00106 #   define L4_ARCH_EI_CLASS     ELFCLASS32
00107 # endif
00108 #else
00109 # warning elf.h: Unsupported build architecture!
00110 #endif
00111
00112
00114

```

```

00116
00117 enum
00118 {
00119     EI_NIDENT          = 16,
00120 };
00121
00125 typedef struct
00126 {
00127     unsigned char e_ident[EI_NIDENT];
00128     Elf32_Half    e_type;
00129     Elf32_Half    e_machine;
00130     Elf32_Word    e_version;
00131     Elf32_Addr    e_entry;
00132     Elf32_Off     e_phoff;
00133     Elf32_Off     e_shoff;
00134     Elf32_Word    e_flags;
00135     Elf32_Half    e_ehsize;
00136     Elf32_Half    e_phentsize;
00137     Elf32_Half    e_phnum;
00138     Elf32_Half    e_shentsize;
00139     Elf32_Half    e_shnum;
00140     Elf32_Half    e_shstrndx;
00141 } Elf32_Ehdr;
00142
00146 typedef struct
00147 {
00148     unsigned char e_ident[EI_NIDENT];
00149     Elf64_Half    e_type;
00150     Elf64_Half    e_machine;
00151     Elf64_Word    e_version;
00152     Elf64_Addr    e_entry;
00153     Elf64_Off     e_phoff;
00154     Elf64_Off     e_shoff;
00155     Elf64_Word    e_flags;
00156     Elf64_Half    e_ehsize;
00157     Elf64_Half    e_phentsize;
00158     Elf64_Half    e_phnum;
00159     Elf64_Half    e_shentsize;
00160     Elf64_Half    e_shnum;
00161     Elf64_Half    e_shstrndx;
00162 } Elf64_Ehdr;
00163
00168 enum Elf_ETs
00169 {
00170     ET_NONE          = 0,
00171     ET_REL           = 1,
00172     ET_EXEC          = 2,
00173     ET_DYN           = 3,
00174     ET_CORE          = 4,
00175     ET_LOPROC        = 0xff00,
00176     ET_HIPROC        = 0xffff,
00177 };
00178
00183 enum Elf_EMs
00184 {
00185     EM_NONE          = 0,
00186     EM_M32           = 1,
00187     EM_SPARC         = 2,
00188     EM_386           = 3,
00189     EM_68K           = 4,
00190     EM_88K           = 5,
00191     EM_860           = 7,
00192     EM_MIPS          = 8,
00193     EM_MIPS_RS4_BE   = 10,
00194     EM_SPARC64       = 11,
00195     EM_PARISC        = 15,
00196     EM_VPP500        = 17,
00197     EM_SPARC32PLUS   = 18,
00198     EM_960           = 19,
00199     EM_PPC           = 20,
00200     EM_V800          = 36,
00201     EM_FR20          = 37,
00202     EM_RH32          = 38,
00203     EM_RCE           = 39,
00204     EM_ARM           = 40,
00205     EM_ALPHA         = 41,
00206     EM_SH            = 42,
00207     EM_SPARCV9       = 43,
00208     EM_TRICORE       = 44,
00209     EM_ARC           = 45,
00210     EM_H8_300        = 46,
00211     EM_H8_300H       = 47,
00212     EM_H8S           = 48,
00213     EM_H8_500        = 49,
00214     EM_IA_64         = 50,
00215     EM_MIPS_X        = 51,
00216     EM_COLDFIRE      = 52,

```

```

00217 EM_68HC12          = 53,
00218 EM_X86_64          = 62,
00219 EM_PDSP            = 63,
00220 EM_FX66             = 66,
00221 EM_ST9PLUS         = 67,
00222 EM_ST7            = 68,
00223 EM_68HC16         = 69,
00224 EM_68HC11         = 70,
00225 EM_68HC08        = 71,
00226 EM_68HC05        = 72,
00227 EM_SVX           = 73,
00228 EM_ST19          = 74,
00229 EM_VAX            = 75,
00230 EM_CRIS            = 76,
00231 EM_JAVELIN        = 77,
00232 EM_FIREPATH       = 78,
00233 EM_ZSP            = 79,
00234 EM_MMIX           = 80,
00235 EM_HUANY           = 81,
00236 EM_PRISM           = 82,
00237 EM_AVR             = 83,
00238 EM_FR30           = 84,
00239 EM_D10V           = 85,
00240 EM_D30V           = 86,
00241 EM_V850           = 87,
00242 EM_M32R            = 88,
00243 EM_MN10300        = 89,
00244 EM_MN10200        = 90,
00245 EM_PJ              = 91,
00246 EM_OPENRISC       = 92,
00247 EM_ARC_A5         = 93,
00248 EM_XTENSA          = 94,
00249 EM_ALTERA_NIOS2    = 113,
00250 EM_AARCH64         = 183,
00251 EM_TILEPRO         = 188,
00252 EM_MICROBLAZE     = 189,
00253 EM_TILEGX         = 191,
00254 EM_RISCV          = 243,
00255 EM_NUM            = 244,
00256 };
00257
00258 #if 0
00259 #define EM_ALPHA      0x9026 /* interim value used by Linux until the
00260                               committee comes up with a final number */
00261 #define EM_S390       0xA390 /* interim value used for IBM S390 */
00262 #endif
00263
00266 enum Elf_EVs
00267 {
00268     EV_NONE          = 0,
00269     EV_CURRENT       = 1,
00270 };
00271
00274 enum Elf_EIs
00275 {
00276     EI_MAG0          = 0,
00277     EI_MAG1          = 1,
00278     EI_MAG2          = 2,
00279     EI_MAG3          = 3,
00280     EI_CLASS         = 4,
00281     EI_DATA          = 5,
00282     EI_VERSION       = 6,
00283     EI_OSABI         = 7,
00284     EI_ABIVERSION    = 8,
00285     EI_PAD           = 9,
00286 };
00287
00289 enum Elf_MAGs
00290 {
00291     ELFMAG0          = 0x7f,
00292     ELFMAG1          = 'E',
00293     ELFMAG2          = 'L',
00294     ELFMAG3          = 'F',
00295 };
00296
00298 enum Elf_CLASSs
00299 {
00300     ELFCLASSNONE     = 0,
00301     ELFCLASS32       = 1,
00302     ELFCLASS64       = 2,
00303     ELFCLASSNUM      = 3,
00304 };
00305
00307 enum Elf_DATAs
00308 {
00309     ELFDATANONE      = 0,
00310     ELFDATA2LSB     = 1,

```



```

00311     ELFDATA2MSB           = 2,
00312     ELFDATANUM           = 3,
00313 };
00314
00316 enum Elf_OSABIs
00317 {
00318     ELFOSABI_NONE           = 0,
00319     ELFOSABI_SYSV           = 0,
00320     ELFOSABI_HPUX           = 1,
00321     ELFOSABI_NETBSD         = 2,
00322     ELFOSABI_LINUX          = 3,
00323     ELFOSABI_SOLARIS        = 6,
00324     ELFOSABI_AIX            = 7,
00325     ELFOSABI_IRIX           = 8,
00326     ELFOSABI_FREEBSD        = 9,
00327     ELFOSABI_TRU64          = 10,
00328     ELFOSABI_MODESTO        = 11,
00329     ELFOSABI_OPENBSD        = 12,
00330     ELFOSABI_ARM            = 97,
00331     ELFOSABI_STANDALONE     = 255,
00332 };
00333
00335 enum Elf_SHNs
00336 {
00337     SHN_UNDEF               = 0,
00338     SHN_LORESERVE           = 0xfff0,
00339     SHN_LOPROC              = 0xfff0,
00340     SHN_HIPROC              = 0xff1f,
00341     SHN_ABS                 = 0xffff1,
00342     SHN_COMMON              = 0xffff2,
00343     SHN_HIRESERVE           = 0xffff,
00344 };
00345
00347 typedef struct
00348 {
00349     Elf32_Word               sh_name;
00350     Elf32_Word               sh_type;
00351     Elf32_Word               sh_flags;
00352     Elf32_Addr               sh_addr;
00353     Elf32_Off                sh_offset;
00354     Elf32_Word               sh_size;
00355     Elf32_Word               sh_link;
00356     Elf32_Word               sh_info;
00357     Elf32_Word               sh_addralign;
00358     Elf32_Word               sh_entsize;
00359 } Elf32_Shdr;
00360
00362 typedef struct
00363 {
00364     Elf64_Word               sh_name;
00365     Elf64_Word               sh_type;
00366     Elf64_Xword              sh_flags;
00367     Elf64_Addr               sh_addr;
00368     Elf64_Off                sh_offset;
00369     Elf64_Xword              sh_size;
00370     Elf64_Word               sh_link;
00371     Elf64_Word               sh_info;
00372     Elf64_Xword              sh_addralign;
00373     Elf64_Xword              sh_entsize;
00374 } Elf64_Shdr;
00375
00377 enum Elf_SHTs
00378 {
00379     SHT_NULL                 = 0,
00380     SHT_PROGBITS             = 1,
00381     SHT_SYMTAB               = 2,
00382     SHT_STRTAB               = 3,
00383     SHT_RELA                 = 4,
00384     SHT_HASH                  = 5,
00385     SHT_DYNAMIC               = 6,
00386     SHT_NOTE                  = 7,
00387     SHT_NOBITS               = 8,
00388     SHT_REL                  = 9,
00389     SHT_SHLIB                 = 10,
00390     SHT_DYNSYM                = 11,
00391     SHT_INIT_ARRAY            = 14,
00392     SHT_FINI_ARRAY            = 15,
00393     SHT_PREINIT_ARRAY         = 16,
00394     SHT_GROUP                  = 17,
00395     SHT_SYMTAB_SHNDX          = 18,
00396     SHT_NUM                   = 19,
00397     SHT_LOOS                  = 0x60000000,
00398     SHT_HIOS                  = 0x6fffffff,
00399     SHT_LOPROC                = 0x70000000,
00400     SHT_HIPROC                = 0x7fffffff,
00401     SHT_LOUSER                = 0x80000000,
00402     SHT_HIUSER                = 0xffffffff,

```

```

00403 };
00404
00406 enum Elf_SHFs
00407 {
00408     SHF_WRITE           = 0x1,
00409     SHF_ALLOC           = 0x2,
00410     SHF_EXECINSTR       = 0x4,
00411     SHF_MERGE           = 0x10,
00412     SHF_STRINGS         = 0x20,
00413     SHF_INFO_LINK       = 0x40,
00414     SHF_LINK_ORDER      = 0x80,
00415     SHF_OS_NONCONFORMING = 0x100,
00417     SHF_GROUP           = 0x200,
00418     SHF_TLS             = 0x400,
00419     SHF_MASKOS          = 0x0ff00000,
00420     SHF_MASKPROC        = 0xf0000000,
00421 };
00422
00423
00425 typedef struct
00426 {
00427     Elf32_Word    p_type;
00428     Elf32_Off     p_offset;
00429     Elf32_Addr    p_vaddr;
00430     Elf32_Addr    p_paddr;
00431     Elf32_Word    p_filesz;
00432     Elf32_Word    p_memsz;
00433     Elf32_Word    p_flags;
00434     Elf32_Word    p_align;
00435 } Elf32_Phdr;
00436
00438 typedef struct
00439 {
00440     Elf64_Word    p_type;
00441     Elf64_Word    p_flags;
00442     Elf64_Off     p_offset;
00443     Elf64_Addr    p_vaddr;
00444     Elf64_Addr    p_paddr;
00445     Elf64_Xword   p_filesz;
00446     Elf64_Xword   p_memsz;
00447     Elf64_Xword   p_align;
00448 } Elf64_Phdr;
00449
00451 enum Elf_PTs
00452 {
00453     PT_NULL           = 0,
00454     PT_LOAD           = 1,
00455     PT_DYNAMIC        = 2,
00456     PT_INTERP         = 3,
00457     PT_NOTE           = 4,
00458     PT_SHLIB          = 5,
00459     PT_PHDR           = 6,
00460     PT_TLS            = 7,
00461     PT_NUM            = 8,
00462     PT_LOOS           = 0x60000000,
00463     PT_HIOS           = 0x6fffffff,
00464     PT_LOPROC         = 0x70000000,
00465     PT_HIPROC         = 0x7fffffff,
00466
00467     PT_GNU_EH_FRAME   = PT_LOOS + 0x474e550,
00468     PT_GNU_STACK      = PT_LOOS + 0x474e551,
00469     PT_GNU_RELRO      = PT_LOOS + 0x474e552,
00470
00471     PT_L4_STACK       = PT_LOOS + 0x12,
00472     PT_L4_AUX          = PT_LOOS + 0x14,
00473 };
00474
00476 enum ELF_PFs
00477 {
00478     PF_X              = 0x1,
00479     PF_W              = 0x2,
00480     PF_R              = 0x4,
00481     PF_MASKOS         = 0x0ff00000,
00482     PF_MASKPROC       = 0x7fffffff,
00483 };
00484
00486 enum Elf_NTscore
00487 {
00488     NT_PRSTATUS       = 1,
00489     NT_FPREGSET       = 2,
00490     NT_PRPSINFO       = 3,
00491     NT_PRXREG         = 4,
00492     NT_TASKSTRUCT     = 4,
00493     NT_PLATFORM       = 5,
00494     NT_AUXV           = 6,
00495     NT_GWINDOWS       = 7,
00496     NT_ASRS           = 8,

```

```

00497 NT_PSTATUS           = 10,
00498 NT_PSINFO             = 13,
00499 NT_PRCREAD            = 14,
00500 NT_UTSNAME             = 15,
00501 NT_LWPSTATUS           = 16,
00502 NT_LWPSINFO           = 17,
00503 NT_PRFPXREG            = 20,
00504 };
00505
00507 enum Elf_NTs_obj
00508 {
00509     NT_VERSION           = 1,
00510 };
00511
00513 typedef struct
00514 {
00515     Elf32_Sword    d_tag;
00516     union
00517     {
00518         Elf32_Word    d_val;
00519         Elf32_Addr    d_ptr;
00520     } d_un;
00521 } Elf32_Dyn;
00522
00524 typedef struct
00525 {
00526     Elf64_Sxword    d_tag;
00527     union
00528     {
00529         Elf64_Xword    d_val;
00530         Elf64_Addr    d_ptr;
00531     } d_un;
00532 } Elf64_Dyn;
00533
00535 enum Elf_DTs
00536 {
00537     DT_NULL           = 0,
00538     DT_NEEDED         = 1,
00539     DT_PLTRELSZ       = 2,
00540     DT_PLTGOT         = 3,
00541     DT_HASH           = 4,
00542     DT_STRTAB         = 5,
00543     DT_SYMTAB         = 6,
00544     DT_RELA           = 7,
00545     DT_RELASZ         = 8,
00546     DT_RELAENT        = 9,
00547     DT_STRSZ          = 10,
00548     DT_SYMENT         = 11,
00549     DT_INIT           = 12,
00550     DT_FINI           = 13,
00551     DT_SONAME         = 14,
00552     DT_RPATH          = 15,
00553     DT_SYMBOLIC       = 16,
00554     DT_REL            = 17,
00555     DT_RELSZ          = 18,
00556     DT_RELENT         = 19,
00557     DT_PTRREL         = 20,
00558     DT_DEBUG          = 21,
00559     DT_TEXTREL        = 22,
00560     DT_JMPREL         = 23,
00561     DT_BIND_NOW       = 24,
00562     DT_INIT_ARRAY     = 25,
00563     DT_FINI_ARRAY     = 26,
00564     DT_INIT_ARRAYSZ   = 27,
00565     DT_FINI_ARRAYSZ   = 28,
00566     DT_RUNPATH        = 29,
00567     DT_FLAGS           = 30,
00568     DT_ENCODING       = 32,
00569     DT_PREINIT_ARRAY  = 32,
00570     DT_PREINIT_ARRAYSZ = 33,
00571     DT_NUM            = 34,
00572     DT_LOOS           = 0x6000000d,
00573     DT_HIOS           = 0x6ffff000,
00574     DT_LOPROC         = 0x70000000,
00575     DT_HIPROC         = 0x7fffffff,
00576 };
00577
00581 enum Elf_DFs
00582 {
00583     DF_ORIGIN         = 0x00000001,
00584     DF_SYMBOLIC       = 0x00000002,
00585     DF_TEXTREL        = 0x00000004,
00586     DF_BIND_NOW       = 0x00000008,
00587     DF_STATIC_TLS     = 0x00000010,
00588 };
00589
00594 enum Elf_DF_1s

```

```

00595 {
00596     DF_1_NOW                = 0x00000001,
00597     DF_1_GLOBAL             = 0x00000002,
00598     DF_1_GROUP              = 0x00000004,
00599     DF_1_NODELETE           = 0x00000008,
00600     DF_1_LOADFLTR           = 0x00000010,
00601     DF_1_INITFIRST          = 0x00000020,
00602     DF_1_NOOPEN             = 0x00000040,
00603     DF_1_ORIGIN             = 0x00000080,
00604     DF_1_DIRECT             = 0x00000100,
00605     DF_1_TRANS              = 0x00000200,
00606     DF_1_INTERPOSE          = 0x00000400,
00607     DF_1_NODEFLIB           = 0x00000800,
00608     DF_1_NODUMP             = 0x00001000,
00609     DF_1_CONFALT            = 0x00002000,
00610     DF_1_ENDFILTEE          = 0x00004000,
00611     DF_1_DISPRELDNE         = 0x00008000,
00612     DF_1_DISPRELPND         = 0x00010000,
00613 };
00614
00616 enum Elf_DTF_1s
00617 {
00618     DTF_1_PARINIT           = 0x00000001,
00619     DTF_1_CONFEXP          = 0x00000002,
00620 };
00621
00623 enum Elf_DF_P1s
00624 {
00625     DF_P1_LAZYLOAD          = 0x00000001,
00626     DF_P1_GROUPPERM        = 0x00000002,
00627 };
00628
00629
00631 typedef struct
00632 {
00633     Elf32_Addr              r_offset;
00634     Elf32_Word              r_info;
00635 } Elf32_Rel;
00636
00638 typedef struct
00639 {
00640     Elf32_Addr              r_offset;
00641     Elf32_Word              r_info;
00642     Elf32_Sword             r_addend;
00643 } Elf32_Rela;
00644
00646 typedef struct
00647 {
00648     Elf64_Addr              r_offset;
00649     Elf64_Xword             r_info;
00650 } Elf64_Rel;
00651
00653 typedef struct
00654 {
00655     Elf64_Addr              r_offset;
00656     Elf64_Xword             r_info;
00657     Elf64_Sxword            r_addend;
00658 } Elf64_Rela;
00659
00661 #define ELF32_R_SYM(i)      ((i)>>8)
00663 #define ELF32_R_TYPE(i)     ((unsigned char)(i))
00665 #define ELF32_R_INFO(s,t)  (((s)<<8)+(unsigned char)(t))
00666
00668 #define ELF64_R_SYM(i)      ((i)>>32)
00669
00671 #define ELF64_R_TYPE(i)     ((i)&0xffffffffL)
00672
00674 #define ELF64_R_INFO(s,t)  (((s)<<32)+(t)&0xffffffffL)
00675
00677 enum Elf_R_386_s
00678 {
00679     R_386_NONE              = 0,
00680     R_386_32                = 1,
00681     R_386_PC32              = 2,
00682     R_386_GOT32             = 3,
00683     R_386_PLT32             = 4,
00684     R_386_COPY              = 5,
00685     R_386_GLOB_DAT          = 6,
00686     R_386_JMP_SLOT          = 7,
00687     R_386_RELATIVE          = 8,
00688     R_386_GOTOFF            = 9,
00689     R_386_GOTPC             = 10,
00690     R_386_32PLT            = 11,
00691     R_386_TLS_TPOFF         = 14,
00692     R_386_TLS_IE            = 15,
00694     R_386_TLS_GOTIE         = 16,
00695     R_386_TLS_LE            = 17,
00696     R_386_TLS_GD            = 18,

```

```

00698 R_386_TLS_LDM          = 19,
00700 R_386_16                = 20,
00701 R_386_PC16            = 21,
00702 R_386_8                = 22,
00703 R_386_PC8               = 23,
00704 R_386_TLS_GD_32       = 24,
00706 R_386_TLS_GD_PUSH     = 25,
00707 R_386_TLS_GD_CALL      = 26,
00709 R_386_TLS_GD_POP     = 27,
00710 R_386_TLS_LDM_32       = 28,
00712 R_386_TLS_LDM_PUSH    = 29,
00713 R_386_TLS_LDM_CALL    = 30,
00715 R_386_TLS_LDM_POP     = 31,
00716 R_386_TLS_LDO_32      = 32,
00717 R_386_TLS_LE_32      = 33,
00719 R_386_TLS_LE_32      = 34,
00721 R_386_TLS_DTPMOD32    = 35,
00722 R_386_TLS_DTPOFF32    = 36,
00723 R_386_TLS_TPOFF32     = 37,
00724 R_386_NUM              = 38,
00725 };
00726
00728
00730 enum Elf_EF_ARM_s
00731 {
00732     EF_ARM_RELEXEC        = 0x01,
00733     EF_ARM_HASENTRY       = 0x02,
00734     EF_ARM_INTERWORK      = 0x04,
00735     EF_ARM_APCS_26        = 0x08,
00736     EF_ARM_APCS_FLOAT     = 0x10,
00737     EF_ARM_PIC            = 0x20,
00738     EF_ARM_ALIGN8         = 0x40,
00739     EF_ARM_NEW_ABI        = 0x80,
00740     EF_ARM_OLD_ABI        = 0x100,
00741
00742     /* Other constants defined in the ARM ELF spec. version B-01. */
00743     /* NB. These conflict with values defined above. */
00744     EF_ARM_SYMSARESORTED  = 0x04,
00745     EF_ARM_DYNSYMSUSESEGIDX = 0x08,
00746     EF_ARM_MAPSYMSFIRST   = 0x10,
00747     EF_ARM_EABIMASK       = 0xFF000000,
00748
00749     #define EF_ARM_EABI_VERSION(flags) ((flags) & EF_ARM_EABIMASK)
00750     EF_ARM_EABI_UNKNOWN    = 0x00000000,
00751     EF_ARM_EABI_VER1       = 0x01000000,
00752     EF_ARM_EABI_VER2       = 0x02000000,
00753 };
00754
00756 enum Elf_STT_ARM_s
00757 {
00758     STT_ARM_TFUNC          = 0xd,
00759 };
00760
00762 enum Elf_SHF_s_ARM
00763 {
00764     SHF_ARM_ENTRYSECT      = 0x10000000,
00765     SHF_ARM_COMDEF         = 0x80000000,
00766 };
00768
00770 enum Elf_ARM_SBs
00771 {
00772     PF_ARM_SB              = 0x10000000,
00773 };
00775
00777 enum Elf_R_ARM_s
00778 {
00779     R_ARM_NONE             = 0,
00780     R_ARM_PC24             = 1,
00781     R_ARM_ABS32            = 2,
00782     R_ARM_REL32            = 3,
00783     R_ARM_PC13             = 4,
00784     R_ARM_ABS16            = 5,
00785     R_ARM_ABS12            = 6,
00786     R_ARM_THM_ABS5         = 7,
00787     R_ARM_ABS8             = 8,
00788     R_ARM_SBREL32          = 9,
00789     R_ARM_THM_PC22         = 10,
00790     R_ARM_THM_PC8          = 11,
00791     R_ARM_AMP_VCALL9       = 12,
00792     R_ARM_SWI24            = 13,
00793     R_ARM_THM_SWI8         = 14,
00794     R_ARM_XPC25            = 15,
00795     R_ARM_THM_XPC22        = 16,
00796     R_ARM_COPY             = 20,
00797     R_ARM_GLOB_DAT         = 21,
00798     R_ARM_JUMP_SLOT        = 22,
00799     R_ARM_RELATIVE         = 23,

```

```

00800 R_ARM_GOTOFF          = 24,
00801 R_ARM_GOTPC          = 25,
00802 R_ARM_GOT32         = 26,
00803 R_ARM_PLT32         = 27,
00804 R_ARM_ALU_PCREL_7_0  = 32,
00805 R_ARM_ALU_PCREL_15_8 = 33,
00806 R_ARM_ALU_PCREL_23_15 = 34,
00807 R_ARM_LDR_SBREL_11_0 = 35,
00808 R_ARM_ALU_SBREL_19_12 = 36,
00809 R_ARM_ALU_SBREL_27_20 = 37,
00810 R_ARM_GNU_VTENTRY    = 100,
00811 R_ARM_GNU_VTINHERIT = 101,
00812 R_ARM_THM_PC11      = 102,
00813 R_ARM_THM_PC9        = 103,
00814 R_ARM_RXPC25        = 249,
00815 R_ARM_RSBREL32      = 250,
00816 R_ARM_THM_RPC22      = 251,
00817 R_ARM_RREL32        = 252,
00818 R_ARM_RABS22        = 253,
00819 R_ARM_RPC24          = 254,
00820 R_ARM_RBASE         = 255,
00821 R_ARM_NUM           = 256,
00822 };
00823
00825 enum Elf_R_AARCH64_s
00826 {
00827     R_AARCH64_NONE      = 0,
00828     R_AARCH64_RELATIVE  = 1027,
00829 };
00830
00832 enum Elf_R_X86_64_s
00833 {
00834     R_X86_64_NONE      = 0,
00835     R_X86_64_64        = 1,
00836     R_X86_64_PC32      = 2,
00837     R_X86_64_GOT32     = 3,
00838     R_X86_64_PLT32     = 4,
00839     R_X86_64_COPY      = 5,
00840     R_X86_64_GLOB_DAT  = 6,
00841     R_X86_64_JUMP_SLOT = 7,
00842     R_X86_64_RELATIVE  = 8,
00843     R_X86_64_GOTPCREL  = 9,
00844     R_X86_64_32        = 10,
00845     R_X86_64_32S       = 11,
00846     R_X86_64_16        = 12,
00847     R_X86_64_PC16      = 13,
00848     R_X86_64_8         = 14,
00849     R_X86_64_PC8       = 15,
00850     R_X86_64_DTPMOD64  = 16,
00851     R_X86_64_DTPOFF64  = 17,
00852     R_X86_64_TPOFF64   = 18,
00853     R_X86_64_TLSGD     = 19,
00854     R_X86_64_TLSLD     = 20,
00855     R_X86_64_DTPOFF32  = 21,
00856     R_X86_64_GOTTPOFF  = 22,
00857     R_X86_64_TPOFF32   = 23,
00858     R_X86_64_NUM       = 24,
00859 };
00860
00862 };
00863
00865 enum Elf_STNs
00866 {
00867     STN_UNDEF          = 0,
00868 };
00869
00871 typedef struct
00872 {
00873     Elf32_Word          st_name;
00874     Elf32_Addr          st_value;
00875     Elf32_Word          st_size;
00876     unsigned char       st_info;
00877     unsigned char       st_other;
00878     Elf32_Half          st_shndx;
00879 } Elf32_Sym;
00880
00882 typedef struct
00883 {
00884     Elf64_Word          st_name;
00885     unsigned char       st_info;
00886     unsigned char       st_other;
00887     Elf64_Half          st_shndx;
00888     Elf64_Addr          st_value;
00889     Elf64_Xword         st_size;
00890 } Elf64_Sym;
00891
00893 #define ELF32_ST_BIND(i)    ((i)>4)
00894
00896 #define ELF32_ST_TYPE(i)    ((i)&0xf)

```

```

00897
00899 #define ELF32_ST_INFO(b,t)  (((b) << 4) + ((t) & 0xf))
00900
00902 #define ELF64_ST_BIND(i)    ((i) >> 4)
00903
00905 #define ELF64_ST_TYPE(i)    ((i) & 0xf)
00906
00908 #define ELF64_ST_INFO(b,t)  (((b) << 4) + ((t) & 0xf))
00909
00912 enum Elf_STBs
00913 {
00914     STB_LOCAL      = 0,
00915     STB_GLOBAL     = 1,
00916     STB_WEAK       = 2,
00917     STB_LOOS       = 10,
00918     STB_HIOS       = 12,
00919     STB_LOPROC     = 13,
00920     STB_HIPROC     = 15,
00921 };
00922
00925 enum Elf_STTs
00926 {
00927     STT_NOTYPE     = 0,
00928     STT_OBJECT     = 1,
00929     STT_FUNC       = 2,
00930     STT_SECTION    = 3,
00931     STT_FILE       = 4,
00932     STT_LOOS       = 10,
00933     STT_HIOS       = 12,
00934     STT_LOPROC     = 13,
00935     STT_HIPROC     = 15,
00936 };
00937
00939 enum Elf_ATs
00940 {
00941     AT_NULL        = 0,
00942     AT_IGNORE      = 1,
00943     AT_EXECD       = 2,
00944     AT_PHDR        = 3,
00945     AT_PHEMT       = 4,
00946     AT_PHNUM       = 5,
00947     AT_PAGESZ      = 6,
00948     AT_BASE        = 7,
00949     AT_FLAGS       = 8,
00950     AT_ENTRY       = 9,
00951     AT_NOTELF      = 10,
00952     AT_UID         = 11,
00953     AT_EUID        = 12,
00954     AT_GID         = 13,
00955     AT_EGID        = 14,
00956
00957     AT_L4_AUX      = 0xf0,
00958     AT_L4_ENV      = 0xf1,
00959     AT_L4_KIP      = 0xf2,
00960 };
00961
00963 typedef struct Elf32_Auxv
00964 {
00965     Elf32_Word atype;
00966     Elf32_Word avalue;
00967 } Elf32_Auxv;
00968
00970 typedef struct Elf64_Auxv
00971 {
00972     Elf64_Word atype;
00973     Elf64_Word avalue;
00974 } Elf64_Auxv;
00975
00976 typedef struct
00977 {
00978     Elf32_Word n_namesz;
00979     Elf32_Word n_descsz;
00980     Elf32_Word n_type;
00981 } Elf32_Nhdr;
00982
00983 typedef struct
00984 {
00985     Elf64_Word n_namesz;
00986     Elf64_Word n_descsz;
00987     Elf64_Word n_type;
00988 } Elf64_Nhdr;
00989
00997 static inline int l4util_elf_check_magic(ElfW(Ehdr) const *hdr);
00998
01006 static inline int l4util_elf_check_arch(ElfW(Ehdr) const *hdr);
01007
01014 static inline ElfW(Phdr) *l4util_elf_phdr(ElfW(Ehdr) const *hdr);

```

```

01015
01016
01017 /* Implementations */
01018
01019 static inline
01020 int l4util_elf_check_magic(ElfW(Ehdr) const *hdr)
01021 {
01022     return    hdr->e_ident[EI_MAG0] == ELFMAG0
01023             && hdr->e_ident[EI_MAG1] == ELFMAG1
01024             && hdr->e_ident[EI_MAG2] == ELFMAG2
01025             && hdr->e_ident[EI_MAG3] == ELFMAG3;
01026 }
01027
01028 static inline
01029 int l4util_elf_check_arch(ElfW(Ehdr) const *hdr)
01030 {
01031     return    hdr->e_ident[EI_CLASS] == L4_ARCH_EI_CLASS
01032             && hdr->e_ident[EI_DATA]  == L4_ARCH_EI_DATA
01033             && hdr->e_machine        == L4_ARCH_E_MACHINE;
01034 }
01035
01036 static inline
01037 ElfW(Phdr) *l4util_elf_phdr(ElfW(Ehdr) const *hdr)
01038 {
01039     return (ElfW(Phdr) *) ((char *)hdr + hdr->e_phoff);
01040 }

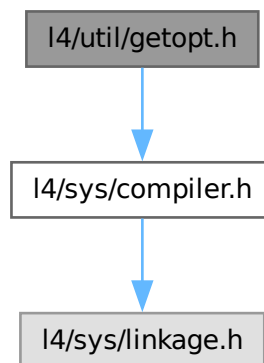
```

16.658 l4/util/getopt.h File Reference

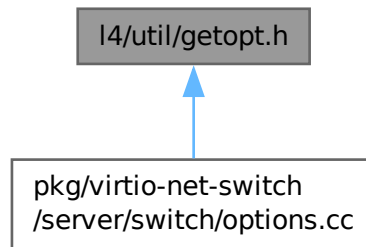
getopt

```
#include <l4/sys/compiler.h>
```

Include dependency graph for getopt.h:



This graph shows which files directly or indirectly include this file:



16.658.1 Detailed Description

getopt

Definition in file [getopt.h](#).

16.659 getopt.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011 #ifndef _GETOPT_H
00012 #define _GETOPT_H
00013
00014 #ifndef NULL
00015 #define NULL 0
00016 #endif
00017
00018 #include <l4/sys/compiler.h>
00019
00020 L4_BEGIN_DECLS
00021
00022 /* For communication from `getopt' to the caller.
00023  * When `getopt' finds an option that takes an argument,
00024  * the argument value is returned here.
00025  * Also, when `ordering' is RETURN_IN_ORDER,
00026  * each non-option ARGV-element is returned here.  */
00027 extern char *optarg;
00028
00029
00030 /* Index in ARGV of the next element to be scanned.
00031  * This is used for communication to and from the caller
00032  * and for communication between successive calls to `getopt'.
00033  *
00034  * On entry to `getopt', zero means this is the first call; initialize.
00035  *
00036  * When `getopt' returns -1, this is the index of the first of the
00037  * non-option elements that the caller should itself scan.
00038  *
00039  * Otherwise, `optind' communicates from one call to the next
00040  * how much of ARGV has been scanned so far.  */
00041
00042 extern int optind;

```

```

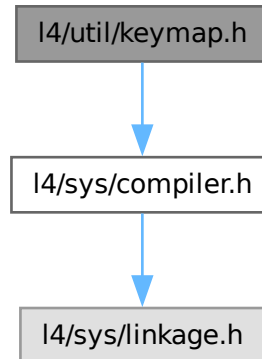
00043
00044 /* Callers store zero here to inhibit the error message `getopt' prints
00045    for unrecognized options. */
00046
00047 extern int opterr;
00048
00049 /* Set to an option character which was unrecognized. */
00050
00051 extern int getopt;
00052
00053 /* Describe the long-named options requested by the application.
00054    The LONG_OPTIONS argument to getopt_long or getopt_long_only is a vector
00055    of `struct option' terminated by an element containing a name which is
00056    zero.
00057
00058    The field `has_arg' is:
00059    no_argument      (or 0) if the option does not take an argument,
00060    required_argument (or 1) if the option requires an argument,
00061    optional_argument (or 2) if the option takes an optional argument.
00062
00063    If the field `flag' is not NULL, it points to a variable that is set
00064    to the value given in the field `val' when the option is found, but
00065    left unchanged if the option is not found.
00066
00067    To have a long-named option do something other than set an `int' to
00068    a compiled-in constant, such as set a value from `optarg', set the
00069    option's `flag' field to zero and its `val' field to a nonzero
00070    value (the equivalent single-letter option character, if there is
00071    one). For long options that have a zero `flag' field, `getopt'
00072    returns the contents of the `val' field. */
00073
00074 struct option
00075 {
00076     const char *name;
00077     /* has_arg can't be an enum because some compilers complain about
00078        type mismatches in all the code that assumes it is an int. */
00079     int has_arg;
00080     int *flag;
00081     int val;
00082 };
00083
00084 /* Names for the values of the `has_arg' field of `struct option'. */
00085
00086 #define no_argument      0
00087 #define required_argument 1
00088 #define optional_argument 2
00089
00090 L4_CV int getopt (int argc, char *const *argv, const char *shortopts);
00091
00092 L4_CV int getopt_long (int argc, char *const *argv, const char *shortopts,
00093                       const struct option *longopts, int *longind);
00094 L4_CV int getopt_long_only (int argc, char *const *argv,
00095                             const char *shortopts,
00096                             const struct option *longopts, int *longind);
00097
00098 L4_CV int _getopt_internal (int argc, char *const *argv,
00099                             const char *shortopts,
00100                             const struct option *longopts, int *longind,
00101                             int long_only);
00102
00103 L4_END_DECLS
00104
00105 #endif /* _GETOPT_H */

```

16.660 I4/util/keymap.h File Reference

Event to ASCII key mapping.

```
#include <l4/sys/compiler.h>
Include dependency graph for keymap.h:
```



16.660.1 Detailed Description

Event to ASCII key mapping.

Definition in file [keymap.h](#).

16.661 keymap.h

[Go to the documentation of this file.](#)

```
00001
00005 /*
00006  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  * License: see LICENSE.spdx (in this directory or the directories above)
00009  */
00010 #ifndef __L4UTIL__KEYMAP_H__
00011 #define __L4UTIL__KEYMAP_H__
00012
00013 #include <l4/sys/compiler.h>
00014
00015 L4_BEGIN_DECLS
00016
00017 int l4util_map_event_to_keymap(unsigned value, unsigned shift);
00018
00019 L4_END_DECLS
00020
00021
00022 #endif /* __L4UTIL__KEYMAP_H__ */
```

16.662 kip.h

```
00001 /*
00002  * Copyright (C) 2026 Kernkonzept GmbH.
00003  * Author(s): Jan Klötzke <jan.kloetzke@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
```

16.663 kip.h

```

00001 /*
00002  * Copyright (C) 2026 Kernkonzept GmbH.
00003  * Author(s): Jan Klötzke <jan.kloetzke@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once

```

16.664 kip.h

```

00001 /*
00002  * Copyright (C) 2026 Kernkonzept GmbH.
00003  * Author(s): Jan Klötzke <jan.kloetzke@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once

```

16.665 l4/sys/kip.h File Reference

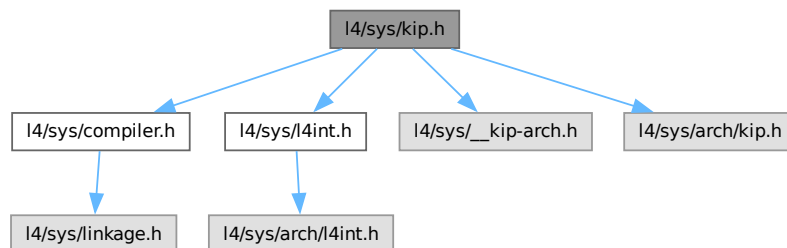
Kernel Info Page access functions.

```

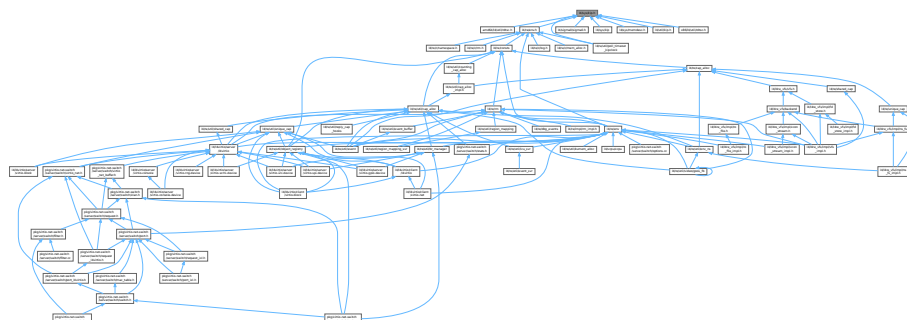
#include <l4/sys/compiler.h>
#include <l4/sys/l4int.h>
#include <l4/sys/__kip-arch.h>
#include <l4/sys/arch/kip.h>

```

Include dependency graph for kip.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [l4_kernel_info_t](#)
L4 Kernel Interface Page.

Macros

- #define **L4_KERNEL_INFO_MAGIC** (0x4BE6344CL) /* "L4μK" */
Kernel Info Page identifier ("L4μK").
- #define **l4_kip_for_each_feature**(s)
Cycle through kernel features given in the KIP.

Typedefs

- typedef struct [l4_kernel_info_t](#) [l4_kernel_info_t](#)
L4 Kernel Interface Page.

Enumerations

- enum { [L4_KIP_OFFS_READ_US](#) = 0x900 , [L4_KIP_OFFS_READ_NS](#) = 0x980 }

Functions

- [l4_kernel_info_t](#) const * [l4_kip](#) (void) [L4_NOTHROW](#)
Get Kernel Info Page.
- [l4_umword_t](#) [l4_kip_version](#) ([l4_kernel_info_t](#) const *kip) [L4_NOTHROW](#)
Get the kernel version.
- const char * [l4_kip_version_string](#) ([l4_kernel_info_t](#) const *kip) [L4_NOTHROW](#)
Get the kernel version string.
- int [l4_kernel_info_version_offset](#) ([l4_kernel_info_t](#) const *kip) [L4_NOTHROW](#)
Return offset in bytes of version_strings relative to the KIP base.
- [l4_cpu_time_t](#) [l4_kip_clock](#) ([l4_kernel_info_t](#) const *kip) [L4_NOTHROW](#)
Return clock value from the KIP.
- [l4_umword_t](#) [l4_kip_clock_lw](#) ([l4_kernel_info_t](#) const *kip) [L4_NOTHROW](#)
Return least significant machine word of clock value from the KIP.
- [l4_uint64_t](#) [l4_kip_clock_ns](#) ([l4_kernel_info_t](#) const *kip) [L4_NOTHROW](#)
Return current clock using the KIP in nanoseconds.
- int [l4_kip_kernel_has_feature](#) ([l4_kernel_info_t](#) const *kip, char const *str)
Check if kernel supports a feature.

16.665.1 Detailed Description

Kernel Info Page access functions.

Definition in file [kip.h](#).

16.665.2 Macro Definition Documentation

16.665.2.1 l4_kip_for_each_feature

```
#define l4_kip_for_each_feature(  
    s)
```

Value:

```
for (s += __builtin_strlen(s) + 1; *s; s += __builtin_strlen(s) + 1)
```

Cycle through kernel features given in the KIP.

Cycles through all KIP kernel feature strings. *s* must be a character pointer (`char const *`) initialized with [l4_kip_version_string\(\)](#).

Definition at line 272 of file [kip.h](#).

Referenced by [l4_kip_kernel_has_feature\(\)](#).

16.665.3 Function Documentation

16.665.3.1 l4_kip_kernel_has_feature()

```
int l4_kip_kernel_has_feature (  
    l4_kernel_info_t const * kip,  
    char const * str) [inline]
```

Check if kernel supports a feature.

Parameters

| | |
|------------|--|
| <i>kip</i> | Pointer to the kernel info page (KIP). |
| <i>str</i> | Feature name to check. |

Returns

1 if the kernel supports the feature, 0 if not.

Checks the feature field in the KIP for the given string.

Definition at line 286 of file [kip.h](#).

References [l4_kip_for_each_feature](#), and [l4_kip_version_string\(\)](#).

Here is the call graph for this function:



16.666 kip.h

[Go to the documentation of this file.](#)

```

00001
00006 /*
00007  * (c) 2008-2013 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00009  *      economic rights: Technische Universität Dresden (Germany)
00010  *
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013 #pragma once
00014
00015 #include <l4/sys/compiler.h>
00016 #include <l4/sys/l4int.h>
00017
00018 #include <l4/sys/__kip-arch.h>
00019
00023 struct l4_kip_platform_info
00024 {
00025     char                name[16];
00026     l4_uint32_t         is_mp;
00027     struct l4_kip_platform_info_arch arch;
00028 };
00029
00036 typedef struct l4_kernel_info_t
00037 {
00038     /* offset 0x00 */
00039     l4_uint32_t         magic;
00042     l4_uint32_t         version;
00043     l4_uint8_t          offset_version_strings;
00044     l4_uint8_t          _fill0[3];
00045     l4_uint8_t          kip_sys_calls;
00046     l4_uint8_t          node;
00047     l4_uint8_t          _fill1[2];
00048
00049     /* offset 0x10 */
00050     l4_uint64_t         sigma0_ip;
00051     l4_uint64_t         root_ip;
00052
00053     /* offset 0x20 */
00054     volatile l4_cpu_time_t _clock_val;
00055     l4_uint64_t         frequency_cpu;
00056
00057     /* offset 0x30 */
00058     l4_uint64_t         acpi_rsdp_addr;
00059     l4_uint64_t         dt_addr;
00060
00061     /* offset 0x40 */
00062     l4_uint64_t         user_ptr;
00063     l4_uint64_t         _res0[1];
00064
00065     /* offset 0x50 */
00066     l4_uint32_t         scheduler_granularity;
00067     l4_uint32_t         mem_descs;
00068     l4_uint32_t         mem_descs_num;
00069     l4_uint32_t         _res1[1];
00070
00071     /* offset 0x60 */
00072     l4_uint64_t         _res2[2];
00073
00074     /* offset 0x70 */
00075     struct l4_kip_platform_info platform_info;
00076 } l4_kernel_info_t;
00077
00085
00089 enum l4_kernel_info_consts_t
00090 {
00091     L4_KIP_VERSION_FIASCO      = 0x87004444,
00092     L4_KIP_VERSION_FIASCO_MASK = 0xff00ffff,
00093 };
00094
00095 enum
00096 {
00105     L4_KIP_OFFS_READ_US      = 0x900,
00106
00116     L4_KIP_OFFS_READ_NS      = 0x980,
00117 };
00118
00122 extern l4_kernel_info_t const *l4_global_kip;
00123
00127 #define L4_KERNEL_INFO_MAGIC (0x4BE6344CL) /* "L4µK" */
00128
00129
00135 L4_INLINE l4_kernel_info_t const *l4_kip(void) L4_NOTHROW;

```

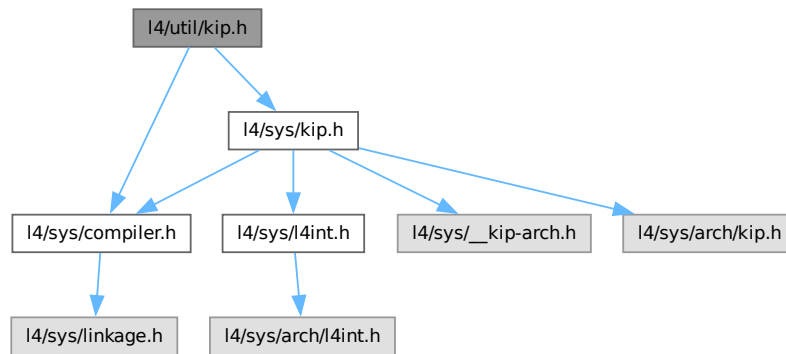
```

00136
00137
00145 L4_INLINE l4_umword_t l4_kip_version(l4_kernel_info_t const *kip) L4_NOTHROW;
00146
00154 L4_INLINE const char *l4_kip_version_string(l4_kernel_info_t const *kip) L4_NOTHROW;
00155
00164 L4_INLINE int
00165 l4_kernel_info_version_offset(l4_kernel_info_t const *kip) L4_NOTHROW;
00166
00184 L4_INLINE l4_cpu_time_t
00185 l4_kip_clock(l4_kernel_info_t const *kip) L4_NOTHROW;
00186
00199 L4_INLINE l4_umword_t
00200 l4_kip_clock_lw(l4_kernel_info_t const *kip) L4_NOTHROW
00201     L4_DEPRECATED("Use l4_kip_clock() instead");
00202
00216 L4_INLINE l4_uint64_t
00217 l4_kip_clock_ns(l4_kernel_info_t const *kip) L4_NOTHROW;
00218
00220
00221 /*****
00222  * Implementations
00223  *****/
00224
00225 L4_INLINE l4_kernel_info_t const*
00226 l4_kip(void) L4_NOTHROW
00227 { return l4_global_kip; }
00228
00229 L4_INLINE l4_umword_t
00230 l4_kip_version(l4_kernel_info_t const *kip) L4_NOTHROW
00231 { return kip->version & L4_KIP_VERSION_FIASCO_MASK; }
00232
00233 L4_INLINE const char*
00234 l4_kip_version_string(l4_kernel_info_t const *k) L4_NOTHROW
00235 { return (const char *)k + l4_kernel_info_version_offset(k); }
00236
00237 L4_INLINE int
00238 l4_kernel_info_version_offset(l4_kernel_info_t const *kip) L4_NOTHROW
00239 { return kip->offset_version_strings « 4; }
00240
00241 L4_INLINE l4_cpu_time_t
00242 l4_kip_clock(l4_kernel_info_t const *kip) L4_NOTHROW
00243 {
00244     // Use kernel-provided code to determine the current clock.
00245     typedef l4_uint64_t (*kip_time_fn_read_us)(void);
00246     kip_time_fn_read_us read_us =
00247         (kip_time_fn_read_us)((l4_uint8_t const*)kip + L4_KIP_OFFS_READ_US);
00248     return read_us();
00249 }
00250
00251 L4_INLINE l4_cpu_time_t
00252 l4_kip_clock_ns(l4_kernel_info_t const *kip) L4_NOTHROW
00253 {
00254     typedef l4_uint64_t (*kip_time_fn_read_ns)(void);
00255     kip_time_fn_read_ns read_ns =
00256         (kip_time_fn_read_ns)((l4_uint8_t const*)kip + L4_KIP_OFFS_READ_NS);
00257     return read_ns();
00258 }
00259
00260 L4_INLINE l4_umword_t
00261 l4_kip_clock_lw(l4_kernel_info_t const *kip) L4_NOTHROW
00262 {
00263     return l4_kip_clock(kip);
00264 }
00265
00272 #define l4_kip_for_each_feature(s) \
00273     for (s += __builtin_strlen(s) + 1; *s; s += __builtin_strlen(s) + 1)
00274
00285 L4_INLINE int
00286 l4_kip_kernel_has_feature(l4_kernel_info_t const *kip, char const *str)
00287 {
00288     const char *s = l4_kip_version_string(kip);
00289     if (!s)
00290         return 0;
00291     l4_kip_for_each_feature(s)
00292     {
00293         if (__builtin_strcmp(s, str) == 0)
00294             return 1;
00295     }
00296     return 0;
00297 }
00298
00299 }
00300
00301 #include <l4/sys/arch/kip.h>

```


16.667 l4/util/kip.h File Reference

```
#include <l4/sys/kip.h>
#include <l4/sys/compiler.h>
Include dependency graph for kip.h:
```



Macros

- `#define l4util_kip_for_each_feature(s)`
Cycle through kernel features given in the KIP.

Functions

- `L4_BEGIN_DECLS int l4util_kip_kernel_has_feature (l4_kernel_info_t const *k, char const *str)`
Check if kernel supports a feature.
- `unsigned long l4util_kip_kernel_abi_version (l4_kernel_info_t const *k)`
Return kernel ABI version.

16.668 kip.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011
00012 #pragma once
00013
00014 #include <l4/sys/kip.h>
00015 #include <l4/sys/compiler.h>
00016
00022
00023
00024 L4_BEGIN_DECLS
00025
00038 L4_CV int l4util_kip_kernel_has_feature(l4_kernel_info_t const *k, char const *str);
00039
00046 L4_CV unsigned long l4util_kip_kernel_abi_version(l4_kernel_info_t const *k);
00047
00048 L4_END_DECLS
00049
00058 #define l4util_kip_for_each_feature(s) l4_kip_for_each_feature(s)
00059
00061
```

16.669 kip.h

```

00001 /*
00002  * Copyright (C) 2024 Kernkonzept GmbH.
00003  * Author(s): Georg Kotheimer <georg.kotheimer@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 L4_INLINE int
00010 l4_kip_has_isa_ext(l4_kernel_info_t const *kip, L4_riscv_isa_ext ext) L4_NOTHROW
00011 {
00012     if (ext < 0 || ext >= L4_riscv_isa_ext_max)
00013         return 0;
00014     return kip->platform_info.arch.isa_ext[ext / 32] & (1 << (ext % 32));
00015 }
00016 
```

16.670 kip.h

```

00001 /*
00002  * Copyright (C) 2026 Kernkonzept GmbH.
00003  * Author(s): Jan Klötzke <jan.kloetzke@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once

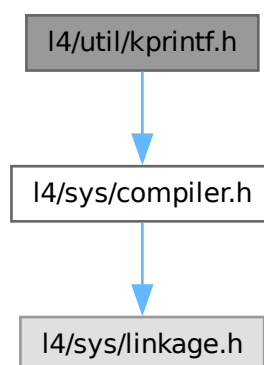
```

16.671 l4/util/kprintf.h File Reference

printf using the kernel debugger

```
#include <l4/sys/compiler.h>
```

Include dependency graph for kprintf.h:



16.671.1 Detailed Description

printf using the kernel debugger

Date

04/05/2007

Author

Adam Lackorzynski adam@os.inf.tu-dresden.de,Definition in file [kprintf.h](#).

16.672 kprintf.h

[Go to the documentation of this file.](#)

```

00001  /*****
00009  */
00010  * (c) 2007-2009 Author(s)
00011  *     economic rights: Technische Universität Dresden (Germany)
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014
00015  #ifndef __L4UTIL__INCLUDE__KPRINTF_H__
00016  #define __L4UTIL__INCLUDE__KPRINTF_H__
00017
00018  #include <l4/sys/compiler.h>
00019
00020  L4_BEGIN_DECLS
00021
00022  L4_CV int l4_kprintf(const char *fmt, ...)
00023                __attribute__((format (printf, 1, 2)));
00024
00025  L4_END_DECLS
00026
00027  #endif /* ! __L4UTIL__INCLUDE__KPRINTF_H__ */

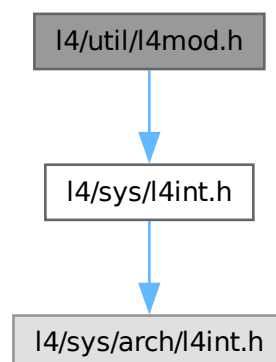
```

16.673 l4/util/l4mod.h File Reference

L4mod structures and constants.

#include <l4/sys/l4int.h>

Include dependency graph for l4mod.h:



Data Structures

- struct [l4util_l4mod_mod](#)
A single module.
- struct [l4util_l4mod_info](#)
Base module structure.

Enumerations

- enum [l4util_l4mod_mod_info_flag](#) {
[L4util_l4mod_mod_flag_unspec](#) = 0 , [L4util_l4mod_mod_flag_kernel](#) = 1 , [L4util_l4mod_mod_flag_sigma0](#) = 2 , [L4util_l4mod_mod_flag_roottask](#) = 3 ,
[L4util_l4mod_mod_flag_mask](#) = 7 << 0 }
Flags for [l4util_l4mod_mod.flags](#).

16.673.1 Detailed Description

L4mod structures and constants.

Definition in file [l4mod.h](#).

16.673.2 Enumeration Type Documentation

16.673.2.1 l4util_l4mod_mod_info_flag

```
enum l4util_l4mod_mod_info_flag
```

Flags for [l4util_l4mod_mod.flags](#).

Enumerator

| | |
|--|--------------------------------|
| L4util_l4mod_mod_flag_unspec | Flag for a generic module. |
| L4util_l4mod_mod_flag_kernel | Flag for the kernel module. |
| L4util_l4mod_mod_flag_sigma0 | Flag for the sigma0 module. |
| L4util_l4mod_mod_flag_roottask | Flag for the root task module. |
| L4util_l4mod_mod_flag_mask | Mask for specified flags. |

Definition at line 17 of file [l4mod.h](#).

16.674 l4mod.h

[Go to the documentation of this file.](#)

```

00001  /*
00002  * Copyright (C) 2021-2022, 2024 Kernkonzept GmbH.
00003  * Author(s): Adam Lackorzynski <adam@l4re.org>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00012  #pragma once
00013
00014  #include <l4/sys/l4int.h>
00015
00017  enum l4util_l4mod_mod_info_flag
00018  {
00019      L4util_l4mod_mod_flag_unspec    = 0,
00020      L4util_l4mod_mod_flag_kernel    = 1,
00021      L4util_l4mod_mod_flag_sigma0    = 2,
00022      L4util_l4mod_mod_flag_roottask   = 3,
00023      L4util_l4mod_mod_flag_mask      = 7 « 0,
00024  };
00025
00027  typedef struct
00028  {
00029      l4_uint64_t flags;
00030      l4_uint64_t mod_start;
00031      l4_uint64_t mod_end;
00032      l4_uint64_t cmdline;
00033  } l4util_l4mod_mod;
00034
00036  typedef struct
00037  {
00038      l4_uint64_t flags;
00039      l4_uint64_t cmdline;
00040      l4_uint64_t mods_addr;
00041      l4_uint32_t mods_count;
00042      l4_uint32_t _pad;
00043
00048      l4_uint64_t vbe_ctrl_info;
00049      l4_uint64_t vbe_mode_info;
00050  } l4util_l4mod_info;

```

16.675 l4/util/list_alloc.h File Reference

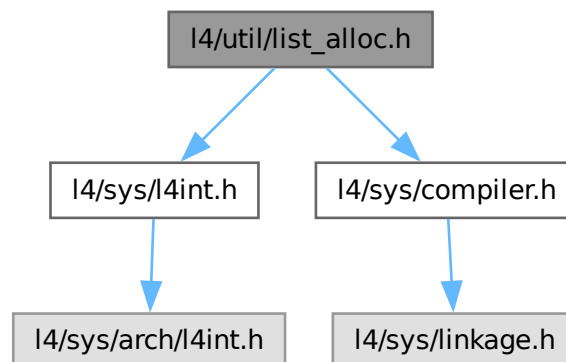
Simple list-based allocator.

```

#include <l4/sys/l4int.h>
#include <l4/sys/compiler.h>

```

Include dependency graph for list_alloc.h:



Functions

- [L4_BEGIN_DECLS](#) void [l4la_free](#) ([l4la_free_t](#) **first, void *block, [l4_size_t](#) size)
Add free memory to memory pool.
- void * [l4la_alloc](#) ([l4la_free_t](#) **first, [l4_size_t](#) size, unsigned align)
Allocate memory from pool.
- void [l4la_dump](#) ([l4la_free_t](#) **first)
Show all list members.
- void [l4la_init](#) ([l4la_free_t](#) **first)
Init memory pool.
- [l4_size_t](#) [l4la_avail](#) ([l4la_free_t](#) **first)
Show available memory in pool.

16.675.1 Detailed Description

Simple list-based allocator.

Taken from the Fiasco kernel.

Date

Alexander Warg <aw11os.inf.tu-dresden.de> Frank Mehnert fm3@os.inf.tu-dresden.de

Definition in file [list_alloc.h](#).

16.675.2 Function Documentation

16.675.2.1 l4la_alloc()

```
void * l4la_alloc (  
    l4la\_free\_t ** first,  
    l4\_size\_t size,  
    unsigned align)
```

Allocate memory from pool.

Parameters

| | |
|--------------|------------------------------------|
| <i>first</i> | list identifier |
| <i>size</i> | length of memory block to allocate |
| <i>align</i> | alignment |

References [L4_CV](#).

16.675.2.2 l4la_avail()

```
l4_size_t l4la_avail (  
    l4la_free_t ** first)
```

Show available memory in pool.

Parameters

| | |
|--------------|-----------------|
| <i>first</i> | list identifier |
|--------------|-----------------|

References [L4_CV](#), and [L4_END_DECLS](#).

16.675.2.3 l4la_dump()

```
void l4la_dump (  
    l4la_free_t ** first)
```

Show all list members.

Parameters

| | |
|--------------|-----------------|
| <i>first</i> | list identifier |
|--------------|-----------------|

References [L4_CV](#).

16.675.2.4 l4la_free()

```
L4_BEGIN_DECLS void l4la_free (  
    l4la_free_t ** first,  
    void * block,  
    l4_size_t size)
```

Add free memory to memory pool.

Parameters

| | |
|--------------|--------------------------------|
| <i>first</i> | list identifier |
| <i>block</i> | address of unused memory block |
| <i>size</i> | size of memory block |

References [L4_CV](#).

16.675.2.5 l4la_init()

```
void l4la_init (
    l4la_free_t ** first)
```

Init memory pool.

Parameters

| | |
|--------------|-----------------|
| <i>first</i> | list identifier |
|--------------|-----------------|

References [L4_CV](#).

16.676 list_alloc.h

[Go to the documentation of this file.](#)

```
00001
00007
00008 /*
00009  * (c) 2003-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00010  *          Frank Mehnert <fm3@os.inf.tu-dresden.de>
00011  *          economic rights: Technische Universität Dresden (Germany)
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014
00015 #ifndef L4UTIL_L4LA_H
00016 #define L4UTIL_L4LA_H
00017
00018 #include <l4/sys/l4int.h>
00019 #include <l4/sys/compiler.h>
00020
00021 typedef struct l4la_free_t_s
00022 {
00023     struct l4la_free_t_s *next;
00024     l4_size_t             size;
00025 } l4la_free_t;
00026
00027 #define L4LA_INITIALIZER { 0 }
00028
00029 L4_BEGIN_DECLS
00030
00035 L4_CV void      l4la_free(l4la_free_t **first, void *block, l4_size_t size);
00036
00041 L4_CV void*     l4la_alloc(l4la_free_t **first, l4_size_t size, unsigned align);
00042
00045 L4_CV void      l4la_dump(l4la_free_t **first);
00046
00049 L4_CV void      l4la_init(l4la_free_t **first);
00050
00053 L4_CV l4_size_t l4la_avail(l4la_free_t **first);
00054
00055 L4_END_DECLS
00056
00057 #endif
```

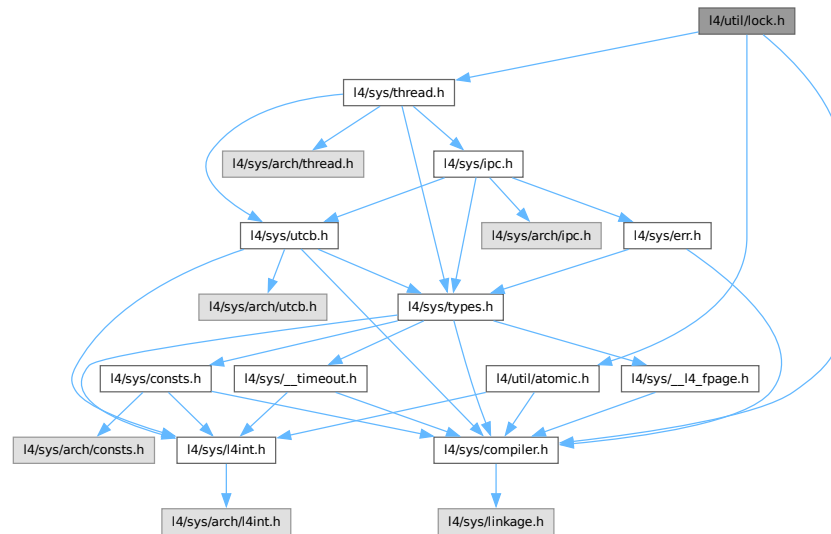
16.677 l4/util/lock.h File Reference

Simple lock implementation.

```
#include <l4/sys/thread.h>
#include <l4/sys/compiler.h>
```



```
#include <l4/util/atomic.h>
Include dependency graph for lock.h:
```



16.677.1 Detailed Description

Simple lock implementation.

Does only work if all thread have the same priority!

Date

02/1997

Author

Michael Hohmuth hohmuth@os.inf.tu-dresden.de

Definition in file [lock.h](#).

16.678 lock.h

[Go to the documentation of this file.](#)

```
00001 /*****
00009 */
00010 * (c) 2000-2009 Author(s)
00011 *     economic rights: Technische Universität Dresden (Germany)
00012 * License: see LICENSE.spdx (in this directory or the directories above)
00013 */
00014
00015 /*****
00016 #ifndef __L4UTIL_LOCK_H__
00017 #define __L4UTIL_LOCK_H__
00018
00019 #include <l4/sys/thread.h>
00020 #include <l4/sys/compiler.h>
```

```

00021 #include <l4/util/atomic.h>
00022
00023 L4_BEGIN_DECLS
00024
00025 typedef l4_uint32_t l4util_simple_lock_t;
00026
00027 L4_INLINE int l4_simple_try_lock(l4util_simple_lock_t *lock);
00028 L4_INLINE void l4_simple_unlock(l4util_simple_lock_t *lock);
00029 L4_INLINE int l4_simple_lock_locked(l4util_simple_lock_t *lock);
00030 L4_INLINE void l4_simple_lock_solid(register l4util_simple_lock_t *p);
00031 L4_INLINE void l4_simple_lock(l4util_simple_lock_t * lock);
00032
00033 L4_INLINE int
00034 l4_simple_try_lock(l4util_simple_lock_t *lock)
00035 {
00036     return l4util_xchg32(lock, 1) == 0;
00037 }
00038
00039 L4_INLINE void
00040 l4_simple_unlock(l4util_simple_lock_t *lock)
00041 {
00042     *lock = 0;
00043 }
00044
00045 L4_INLINE int
00046 l4_simple_lock_locked(l4util_simple_lock_t *lock)
00047 {
00048     return (*lock == 0) ? 0 : 1;
00049 }
00050
00051 L4_INLINE void
00052 l4_simple_lock_solid(register l4util_simple_lock_t *p)
00053 {
00054     while (l4_simple_lock_locked(p) || !l4_simple_try_lock(p))
00055         l4_thread_switch(L4_INVALID_CAP);
00056 }
00057
00058 L4_INLINE void
00059 l4_simple_lock(l4util_simple_lock_t * lock)
00060 {
00061     if (!l4_simple_try_lock(lock))
00062         l4_simple_lock_solid(lock);
00063 }
00064
00065 L4_END_DECLS
00066
00067 #endif

```

16.679 l4/util/mb_info.h File Reference

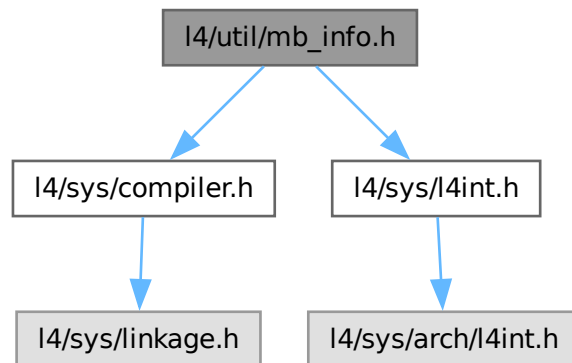
Multiboot info structure as defined by GRUB.

```

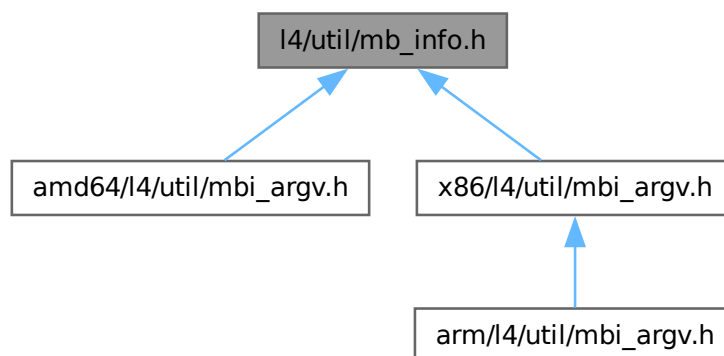
#include <l4/sys/compiler.h>
#include <l4/sys/l4int.h>

```

Include dependency graph for mb_info.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [l4util_mb_mod_t](#)
The structure type "mod_list" is used by the [multiboot_info](#) structure.
- struct [l4util_mb_addr_range_t](#)
INT-15, AX=E820 style "AddressRangeDescriptor" ...with a "size" parameter on the front which is the structure size - 4, pointing to the next one, up until the full buffer length of the memory map has been reached.
- struct [l4util_mb_drive_t](#)
Drive Info structure.
- struct [l4util_mb_apm_t](#)
APM BIOS info.
- struct [l4util_mb_vbe_ctrl_t](#)

VBE controller information.

- struct [l4util_mb_vbe_mode_t](#)

VBE mode information.

- struct [l4util_mb_info_t](#)

MultiBoot Info description.

Macros

- #define **MB_ARD_MEMORY** 1
usable memory "Type", all others are reserved.
- #define **MB_ART_MEMORY** 1
Address Range Types (ART) from "Advanced Configuration and Power Interface Specification" Rev3.0a (p.
- #define **MB_ART_RESERVED** 2
in use or reserved by system
- #define **MB_ART_ACPI** 3
ACPI Reclaim Memory (RAM that contains ACPI tables).
- #define **MB_ART_NVS** 4
ACPI NVS Memory (must not be used by the OS.
- #define **MB_ART_UNUSABLE** 5
memory in which errors have been detected
- #define [l4util_mb_for_each_mmap_entry](#)(i, mbi)
Iterate over a memory map provided in a Multiboot info.
- #define **L4UTIL_MB_MEMORY** 0x00000001
Flags to be set in the 'flags' parameter above.
- #define **L4UTIL_MB_BOOTDEV** 0x00000002
is there a boot device set?
- #define **L4UTIL_MB_CMDLINE** 0x00000004
is the command-line defined?
- #define **L4UTIL_MB_MODS** 0x00000008
are there modules to do something with?
- #define **L4UTIL_MB_AOUT_SYMS** 0x00000010
is there a symbol table loaded?
- #define **L4UTIL_MB_ELF_SHDR** 0x00000020
is there an ELF section header table?
- #define **L4UTIL_MB_MEM_MAP** 0x00000040
is there a full memory map?
- #define **L4UTIL_MB_DRIVE_INFO** 0x00000080
Is there drive info?
- #define **L4UTIL_MB_CONFIG_TABLE** 0x00000100
Is there a config table?
- #define **L4UTIL_MB_BOOT_LOADER_NAME** 0x00000200
Is there a boot loader name?
- #define **L4UTIL_MB_APM_TABLE** 0x00000400
Is there a APM table?
- #define **L4UTIL_MB_VIDEO_INFO** 0x00000800
Is there video information?
- #define **L4UTIL_MB_VALID** 0x2BADB002UL
If we are multiboot-compliant, this value is present in the eax register.

16.679.1 Detailed Description

Multiboot info structure as defined by GRUB.

Definition in file [mb_info.h](#).

16.679.2 Macro Definition Documentation

16.679.2.1 l4util_mb_for_each_mmap_entry

```
#define l4util_mb_for_each_mmap_entry(  
    i,  
    mbi)
```

Value:

```
for (i = l4util_mb_first_mmap_entry(mbi);  
     (unsigned long)i < (unsigned long)mbi->mmap_addr + mbi->mmap_length;  
     i = l4util_mb_next_mmap_entry(i))
```

Iterate over a memory map provided in a Multiboot info.

Parameters

| | |
|------------|---|
| <i>i</i> | Name of a variable of type l4util_mb_addr_range_t * that is consecutively assigned pointers to the entries of the memory map. |
| <i>mbi</i> | Pointer to the l4util_mb_info_t where the memory map can be found. |

Definition at line 332 of file [mb_info.h](#).

16.679.2.2 L4UTIL_MB_MEMORY

```
#define L4UTIL_MB_MEMORY 0x00000001
```

Flags to be set in the 'flags' parameter above.

is there basic lower/upper memory information?

Definition at line 344 of file [mb_info.h](#).

16.679.2.3 MB_ART_MEMORY

```
#define MB_ART_MEMORY 1
```

Address Range Types (ART) from "Advanced Configuration and Power Interface Specification" Rev3.0a (p.

390). Other values are undefined. available, usable RAM

Definition at line 64 of file [mb_info.h](#).

16.680 mb_info.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007  *      Frank Mehnert <fm3@os.inf.tu-dresden.de>
00008  *      economic rights: Technische Universität Dresden (Germany)
00009  * License: see LICENSE.spdx (in this directory or the directories above)
00010  */
00011
00012 #ifndef L4UTIL_MB_INFO_H
00013 #define L4UTIL_MB_INFO_H
00014
00015 /*****
00016  * Multiboot (v1)
00017  *****/
00018
00019 #ifndef __ASSEMBLY__
00020
00021 #include <l4/sys/compiler.h>
00022 #include <l4/sys/l4int.h>
00023
00024 /*
00025  * \defgroup l4util_mb_mod Multiboot v1
00026  * \ingroup l4util_api
00027  */
00028
00033 typedef struct
00034 {
00035     l4_uint32_t mod_start;
00036     l4_uint32_t mod_end;
00037     l4_uint32_t cmdline;
00038     l4_uint32_t pad;
00039 } l4util_mb_mod_t;
00040
00041
00048 typedef struct __attribute__((packed))
00049 {
00050     l4_uint32_t struct_size;
00051     l4_uint64_t addr;
00052     l4_uint64_t size;
00053     l4_uint32_t type;
00054     /* unspecified optional padding... */
00055 } l4util_mb_addr_range_t;
00056
00058 #define MB_ARD_MEMORY 1
00059
00064 #define MB_ART_MEMORY 1
00065 #define MB_ART_RESERVED 2
00066 #define MB_ART_ACPI 3
00068 #define MB_ART_NVS 4
00069 #define MB_ART_UNUSABLE 5
00070
00071
00073 typedef struct
00074 {
00075     l4_uint32_t size;
00076     l4_uint8_t drive_number;
00077     l4_uint8_t drive_mode;
00078     l4_uint16_t drive_cylinders;
00079     l4_uint8_t drive_heads;
00080     l4_uint8_t drive_sectors;
00081     l4_uint16_t drive_sectors[0];
00082 } l4util_mb_drive_t;
00083
00084 /* Drive Mode. */
00085 #define MB_DI_CHS_MODE 0
00086 #define MB_DI_LBA_MODE 1
00087
00088
00090 typedef struct
00091 {
00092     l4_uint16_t version;
00093     l4_uint16_t cseg;
00094     l4_uint32_t offset;
00095     l4_uint16_t cseg_l6;
00096     l4_uint16_t dseg_l6;
00097     l4_uint16_t flags;
00098     l4_uint16_t cseg_len;
00099     l4_uint16_t cseg_l6_len;
00100     l4_uint16_t dseg_l6_len;
00101 } __attribute__((packed)) l4util_mb_apm_t;
00102 static_assert(sizeof(l4util_mb_apm_t) == 20, "Check l4util_mb_apm_t");
00103

```

```

00104
00106 typedef struct
00107 {
00108     l4_uint8_t signature[4];
00109     l4_uint16_t version;
00110     l4_uint32_t oem_string;
00111     l4_uint32_t capabilities;
00112     l4_uint32_t video_mode;
00113     l4_uint16_t total_memory;
00114     l4_uint16_t oem_software_rev;
00115     l4_uint32_t oem_vendor_name;
00116     l4_uint32_t oem_product_name;
00117     l4_uint32_t oem_product_rev;
00118     l4_uint8_t reserved[222];
00119     l4_uint8_t oem_data[256];
00120 } __attribute__((packed)) l4util_mb_vbe_ctrl_t;
00121 static_assert(sizeof(l4util_mb_vbe_ctrl_t) == 512, "Check l4util_mb_vbe_ctrl_t");
00122
00123
00125 typedef struct
00126 {
00130     l4_uint16_t mode_attributes;
00132     l4_uint8_t win_a_attributes;
00134     l4_uint8_t win_b_attributes;
00136     l4_uint16_t win_granularity;
00138     l4_uint16_t win_size;
00140     l4_uint16_t win_a_segment;
00142     l4_uint16_t win_b_segment;
00144     l4_uint32_t win_func;
00146     l4_uint16_t bytes_per_scanline;
00148
00152     l4_uint16_t x_resolution;
00154     l4_uint16_t y_resolution;
00156     l4_uint8_t x_char_size;
00158     l4_uint8_t y_char_size;
00160     l4_uint8_t number_of_planes;
00162     l4_uint8_t bits_per_pixel;
00164     l4_uint8_t number_of_banks;
00166     l4_uint8_t memory_model;
00168     l4_uint8_t bank_size;
00170     l4_uint8_t number_of_image_pages;
00172     l4_uint8_t reserved0;
00174
00178     l4_uint8_t red_mask_size;
00180     l4_uint8_t red_field_position;
00182     l4_uint8_t green_mask_size;
00184     l4_uint8_t green_field_position;
00186     l4_uint8_t blue_mask_size;
00188     l4_uint8_t blue_field_position;
00190     l4_uint8_t reserved_mask_size;
00192     l4_uint8_t reserved_field_position;
00194     l4_uint8_t direct_color_mode_info;
00196
00200     l4_uint32_t phys_base;
00202     l4_uint32_t reserved1;
00204     l4_uint16_t reversed2;
00206
00210     l4_uint16_t linear_bytes_per_scanline;
00212     l4_uint8_t banked_number_of_image_pages;
00214     l4_uint8_t linear_number_of_image_pages;
00216     l4_uint8_t linear_red_mask_size;
00218     l4_uint8_t linear_red_field_position;
00220     l4_uint8_t linear_green_mask_size;
00222     l4_uint8_t linear_green_field_position;
00224     l4_uint8_t linear_blue_mask_size;
00226     l4_uint8_t linear_blue_field_position;
00228     l4_uint8_t linear_reserved_mask_size;
00230     l4_uint8_t linear_reserved_field_position;
00232     l4_uint32_t max_pixel_clock;
00234     l4_uint8_t reserved3[190];
00236 } __attribute__((packed)) l4util_mb_vbe_mode_t;
00237 static_assert(sizeof(l4util_mb_vbe_mode_t) == 256, "Check l4util_mb_vbe_mode_t");
00238
00239
00246
00247 typedef struct
00248 {
00249     l4_uint32_t flags;
00250     l4_uint32_t mem_lower;
00251     l4_uint32_t mem_upper;
00252     l4_uint32_t boot_device;
00253     l4_uint32_t cmdline;
00254     l4_uint32_t mods_count;
00255     l4_uint32_t mods_addr;
00256
00257     union
00258     {

```

```

00259     struct
00260     {
00262         l4_uint32_t tabsize;
00263         l4_uint32_t strsize;
00264         l4_uint32_t addr;
00265         l4_uint32_t pad;
00266     }
00267     a;
00268
00269     struct
00270     {
00272         l4_uint32_t num;
00273         l4_uint32_t size;
00274         l4_uint32_t addr;
00275         l4_uint32_t shndx;
00276     }
00277     e;
00278 }
00279 syms;
00280
00281 l4_uint32_t mmap_length;
00282 l4_uint32_t mmap_addr;
00283 l4_uint32_t drives_length;
00284 l4_uint32_t drives_addr;
00285 l4_uint32_t config_table;
00286 l4_uint32_t boot_loader_name;
00287 l4_uint32_t apm_table;
00288 l4_uint32_t vbe_ctrl_info;
00289 l4_uint32_t vbe_mode_info;
00290 l4_uint16_t vbe_mode;
00291 l4_uint16_t vbe_interface_seg;
00292 l4_uint16_t vbe_interface_off;
00293 l4_uint16_t vbe_interface_len;
00294 } l4util_mb_info_t;
00295 static_assert(sizeof(l4util_mb_info_t) == 88, "Check l4util_mb_info_t");
00296
00303 static inline l4util_mb_addr_range_t *
00304 l4util_mb_first_mmap_entry(l4util_mb_info_t *mbi)
00305 {
00306     return (l4util_mb_addr_range_t *) (l4_addr_t) mbi->mmap_addr;
00307 }
00308
00318 static inline l4util_mb_addr_range_t *
00319 l4util_mb_next_mmap_entry(l4util_mb_addr_range_t *e)
00320 {
00321     return (l4util_mb_addr_range_t *) ((l4_addr_t) e + e->struct_size
00322                                         + sizeof(e->struct_size));
00323 }
00324
00332 #define l4util_mb_for_each_mmap_entry(i, mbi)
00333     for (i = l4util_mb_first_mmap_entry(mbi);
00334          (unsigned long)i < (unsigned long)mbi->mmap_addr + mbi->mmap_length;
00335          i = l4util_mb_next_mmap_entry(i))
00336
00337 #endif /* ! __ASSEMBLY__ */
00338
00342
00344 #define L4UTIL_MB_MEMORY      0x00000001
00345
00347 #define L4UTIL_MB_BOOTDEV     0x00000002
00348
00350 #define L4UTIL_MB_CMDLINE     0x00000004
00351
00353 #define L4UTIL_MB_MODS        0x00000008
00354
00355 /* These next two are mutually exclusive */
00357 #define L4UTIL_MB_AOUT_SYMS    0x00000010
00358
00360 #define L4UTIL_MB_ELF_SHDR     0x00000020
00361
00363 #define L4UTIL_MB_MEM_MAP      0x00000040
00364
00366 #define L4UTIL_MB_DRIVE_INFO    0x00000080
00367
00369 #define L4UTIL_MB_CONFIG_TABLE 0x00000100
00370
00372 #define L4UTIL_MB_BOOT_LOADER_NAME 0x00000200
00373
00375 #define L4UTIL_MB_APM_TABLE     0x00000400
00376
00378 #define L4UTIL_MB_VIDEO_INFO    0x00000800
00379
00380
00382 #define L4UTIL_MB_VALID         0x2BADB002UL
00383 #define L4UTIL_MB_VALID_ASM     0x2BADB002
00384
00385

```



```

00386 /*****
00387  * Multiboot2
00388  *****/
00389
00390 #ifndef __ASSEMBLY__
00391
00392 typedef struct
00393 {
00394     l4_uint32_t total_size;
00395     l4_uint32_t reserved;
00396 } __attribute__((packed)) l4util_mb2_info_t;
00397
00398 typedef struct
00399 {
00400     char string[0];
00401 } __attribute__((packed)) l4util_mb2_cmdline_tag_t;
00402
00403 typedef struct
00404 {
00405     l4_uint32_t mod_start;
00406     l4_uint32_t mod_end;
00407     char string[];
00408 } __attribute__((packed)) l4util_mb2_module_tag_t;
00409
00410 typedef struct
00411 {
00412     l4_uint64_t base_addr;
00413     l4_uint64_t length;
00414     l4_uint32_t type;
00415     l4_uint32_t reserved;
00416 } __attribute__((packed)) l4util_mb2_memmap_entry_t;
00417
00418 typedef struct
00419 {
00420     l4_uint32_t entry_size;
00421     l4_uint32_t entry_version;
00422     l4util_mb2_memmap_entry_t entries[];
00423 } __attribute__((packed)) l4util_mb2_memmap_tag_t;
00424
00425 typedef struct
00426 {
00427     char data[0];
00428 } __attribute__((packed)) l4util_mb2_rsdp_tag_t;
00429
00430
00431 struct color_info_rgb_t
00432 {
00433     l4_uint8_t framebuffer_red_field_position;
00434     l4_uint8_t framebuffer_red_mask_size;
00435     l4_uint8_t framebuffer_green_field_position;
00436     l4_uint8_t framebuffer_green_mask_size;
00437     l4_uint8_t framebuffer_blue_field_position;
00438     l4_uint8_t framebuffer_blue_mask_size;
00439 } __attribute__((packed));
00440
00441 typedef struct
00442 {
00443     l4_uint64_t framebuffer_addr;
00444     l4_uint32_t framebuffer_pitch;
00445     l4_uint32_t framebuffer_width;
00446     l4_uint32_t framebuffer_height;
00447     l4_uint8_t framebuffer_bpp;
00448     l4_uint8_t framebuffer_type;
00449     l4_uint8_t reserved;
00450
00451     // color_info;
00452     union
00453     {
00454         struct color_info_rgb_t color_info_rgb;
00455     };
00456 } __attribute__((packed)) l4util_mb2_framebuffer_tag_t;
00457
00458 typedef struct
00459 {
00460     l4_uint32_t type;
00461     l4_uint32_t size;
00462
00463     union
00464     {
00465         l4util_mb2_cmdline_tag_t cmdline;
00466         l4util_mb2_module_tag_t module;
00467         l4util_mb2_memmap_tag_t memmap;
00468         l4util_mb2_framebuffer_tag_t fb;
00469         l4util_mb2_rsdp_tag_t rsdp;
00470     };
00471 } __attribute__((packed)) l4util_mb2_tag_t;
00472

```

```

00473 #endif /* ! __ASSEMBLY__ */
00474
00475
00476 #define L4UTIL_MB2_MAGIC      0xE85250D6
00477 #define L4UTIL_MB2_ARCH_I386  0x0
00478
00479 #define L4UTIL_MB2_TERMINATOR_HEADER_TAG  0
00480 #define L4UTIL_MB2_INFO_REQUEST_HEADER_TAG 1
00481 #define L4UTIL_MB2_ENTRY_ADDRESS_HEADER_TAG 3
00482 #define L4UTIL_MB2_FRAMEBUFFER_HEADER_TAG 5
00483 #define L4UTIL_MB2_RELOCATABLE_HEADER_TAG 10
00484
00485 #define L4UTIL_MB2_TAG_FLAG_REQUIRED      0
00486
00487 #define L4UTIL_MB2_TAG_ALIGN_SHIFT      3
00488 #define L4UTIL_MB2_TAG_ALIGN           8
00489
00490 #define L4UTIL_MB2_TERMINATOR_INFO_TAG      0
00491 #define L4UTIL_MB2_BOOT_CMDLINE_INFO_TAG    1
00492 #define L4UTIL_MB2_MODULE_INFO_TAG          3
00493 #define L4UTIL_MB2_MEMORY_MAP_INFO_TAG      6
00494 #define L4UTIL_MB2_FRAMEBUFFER_INFO_TAG     8
00495 #define L4UTIL_MB2_RSDP_OLD_INFO_TAG        14
00496 #define L4UTIL_MB2_RSDP_NEW_INFO_TAG        15
00497 #define L4UTIL_MB2_IMAGE_LOAD_BASE_PHYS_INFO_TAG 21
00498
00499 #define L4UTIL_MB2_RELO_PREFERRED_NONE 0
00500 #define L4UTIL_MB2_RELO_PREFERRED_MIN  1
00501 #define L4UTIL_MB2_RELO_PREFERRED_MAX  2
00502
00503 #endif

```

16.681 l4/util/parse_cmd.h File Reference

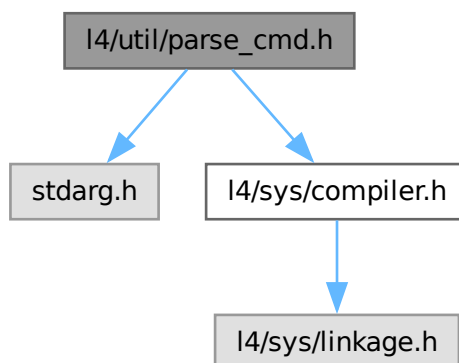
comfortable command-line parsing

```

#include <stdarg.h>
#include <l4/sys/compiler.h>

```

Include dependency graph for parse_cmd.h:



Typedefs

- typedef void(* **parse_cmd_fn_t**) (int)
Function type for `PARSE_CMD_FN`.
- typedef void(* **parse_cmd_fn_arg_t**) (int, const char *, int)
Function type for `PARSE_CMD_FN_ARG`.

Enumerations

- enum [parse_cmd_type](#)

Types for parsing.

Functions

- [L4_BEGIN_DECLS](#) int [parse_cmdline](#) (int *argc, const char ***argv, int arg0,...)

Parse the command-line for specified arguments and store the values into variables.

16.681.1 Detailed Description

comfortable command-line parsing

Date

2002

Author

Jork Loeser jork.loeser@inf.tu-dresden.de

Definition in file [parse_cmd.h](#).

16.682 parse_cmd.h

[Go to the documentation of this file.](#)

```

00001
00008
00009 /*
00010  * (c) 2003-2009 Author(s)
00011  *     economic rights: Technische Universität Dresden (Germany)
00012  * License: see LICENSE.spdx (in this directory or the directories above)
00013  */
00014
00015 #ifndef __PARSE_CMD_H
00016 #define __PARSE_CMD_H
00017
00018 #include <stdarg.h>
00019 #include <l4/sys/compiler.h>
00020
00026
00030 enum parse_cmd_type {
00031     PARSE_CMD_INT,
00032     PARSE_CMD_SWITCH,
00033     PARSE_CMD_STRING,
00034     PARSE_CMD_FN,
00035     PARSE_CMD_FN_ARG,
00036     PARSE_CMD_INC,
00037     PARSE_CMD_DEC,
00038 };
00039
00043 typedef L4_CV void (*parse_cmd_fn_t)(int);
00044
00048 typedef L4_CV void (*parse_cmd_fn_arg_t)(int, const char*, int);
00049
00050 L4_BEGIN_DECLS
00051
00138 L4_CV int parse_cmdline(int *argc, const char***argv, int arg0, ...);
00139 L4_CV int parse_cmdlinev(int *argc, const char***argv, int arg0, va_list va);
00140 L4_CV int parse_cmdline_extra(const char*argv0, const char*line, char delim,
00141                             int arg0,...);
00142
00143 L4_END_DECLS
00145
00146 #endif
00147

```

16.683 printf_helpers.h

```

00001 /*
00002  * Copyright (C) 2025 Kernkonzept GmbH.
00003  * Author(s): Frank Mehnert <frank.mehnert@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00008 #pragma once
00009
00010 #include <stddef.h>
00011 #include <stdio.h>
00012 #include <l4/sys/compiler.h>
00013
00030 L4_INLINE int l4util_human_readable_size(char *outstr, size_t outsize,
00031                                         unsigned long long bytes)
00032 {
00033     static char const *const unitstr = "BKMGT";
00034
00035     int idx = sizeof(unitstr) - 2;
00036     int order;
00037
00038     for (order = idx * 10; order > 10; order -= 10, --idx)
00039         if (bytes > (1ULL « order))
00040             break;
00041
00042     unsigned long long value = bytes » order;
00043     unsigned long long fract = (bytes - (value « order))
00044                               / ((1ULL « order) / 10 + 1);
00045
00046     return snprintf(outstr, outsize, "%llu.%11lu %ciB",
00047                    value, fract, unitstr[idx]);
00048 }

```

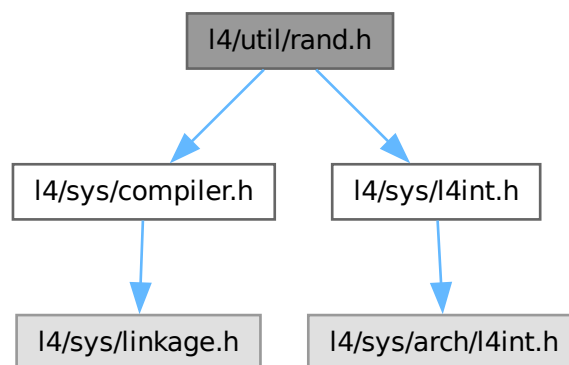
16.684 l4/util/rand.h File Reference

Simple Pseudo-Random Number Generator.

```
#include <l4/sys/compiler.h>
```

```
#include <l4/sys/l4int.h>
```

Include dependency graph for rand.h:



Functions

- [l4_uint32_t l4util_rand](#) (void)

Deliver next random number.

- void `l4util_srand` (`l4_uint32_t` seed)

Initialize random number generator.

16.684.1 Detailed Description

Simple Pseudo-Random Number Generator.

Date

1998

Author

Lars Reuther reuther@os.inf.tu-dresden.de

Definition in file [rand.h](#).

16.685 rand.h

[Go to the documentation of this file.](#)

```

00001
00008 /*
00009  * (c) 2008-2009 Author(s)
00010  *      economic rights: Technische Universität Dresden (Germany)
00011  * License: see LICENSE.spdx (in this directory or the directories above)
00012  */
00013
00014 #ifndef __L4UTIL_RAND_H
00015 #define __L4UTIL_RAND_H
00016
00017 #define L4_RAND_MAX 65535
00018
00019 #include <l4/sys/compiler.h>
00020 #include <l4/sys/l4int.h>
00021
00022 L4_BEGIN_DECLS
00023
00028
00035 L4_CV l4_uint32_t
00036 l4util_rand(void);
00037
00044 L4_CV void
00045 l4util_srand (l4_uint32_t seed);
00046
00047 L4_END_DECLS
00048
00049 #endif /* __L4UTIL_RAND_H */

```

16.686 l4/util/splitlog2.h File Reference

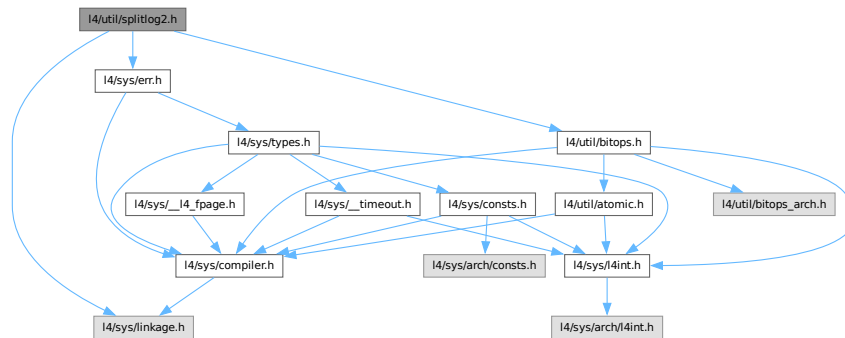
Split a range in log2 aligned and size-aligned chunks.

```

#include <l4/sys/linkage.h>
#include <l4/sys/err.h>

```

#include <l4/util/bitops.h>
 Include dependency graph for splitlog2.h:



Functions

- `L4_BEGIN_DECLS` long `l4util_splitlog2_hdl` (`l4_addr_t` start, `l4_addr_t` end, long(*handler)(`l4_addr_t` s, `l4_addr_t` e, int log2size))
Split a range into log2 base and size aligned chunks.
- `l4_addr_t` `l4util_splitlog2_size` (`l4_addr_t` start, `l4_addr_t` end)
Return log2 base and size aligned length of a range.

16.686.1 Detailed Description

Split a range in log2 aligned and size-aligned chunks.

Definition in file [splitlog2.h](#).

16.687 splitlog2.h

[Go to the documentation of this file.](#)

```

00001
00005 /*
00006  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007  *     economic rights: Technische Universität Dresden (Germany)
00008  * License: see LICENSE.spdx (in this directory or the directories above)
00009  */
00010 #ifndef __L4UTIL__INCLUDE__SPLITLOG2_H__
00011 #define __L4UTIL__INCLUDE__SPLITLOG2_H__
00012
00013 #include <l4/sys/linkage.h>
00014 #include <l4/sys/err.h>
00015 #include <l4/util/bitops.h>
00016
00017 L4_BEGIN_DECLS
00018
00031 L4_INLINE long
00032 l4util_splitlog2_hdl(l4_addr_t start, l4_addr_t end,
00033                     long (*handler)(l4_addr_t s, l4_addr_t e, int log2size));
00034
00043 L4_INLINE l4_addr_t
00044 l4util_splitlog2_size(l4_addr_t start, l4_addr_t end);
00045
00046 L4_END_DECLS
00047
00048 /* Implementation */

```

```

00049
00050 L4_INLINE long
00051 l4util_splitlog2_hdl(l4_addr_t start, l4_addr_t end,
00052                      long (*handler)(l4_addr_t s, l4_addr_t e, int log2size))
00053 {
00054     if (end < start)
00055         return -L4_EINVAL;
00056
00057     while (start <= end)
00058     {
00059         long retval;
00060         int len2 = l4util_splitlog2_size(start, end);
00061         l4_addr_t len = 1UL < len2;
00062         if ((retval = handler(start, start + len - 1, len2)))
00063             return retval;
00064         start += len;
00065     }
00066     return 0;
00067 }
00068
00069 L4_INLINE l4_addr_t
00070 l4util_splitlog2_size(l4_addr_t start, l4_addr_t end)
00071 {
00072     int start_bits = l4util_bsf(start);
00073     int len_bits = l4util_bsr(end - start + 1);
00074     if (start_bits != -1 && len_bits > start_bits)
00075         len_bits = start_bits;
00076
00077     return len_bits;
00078 }
00079
00080 #endif /* ! __L4UTIL__INCLUDE__SPLITLOG2_H__ */

```

16.688 util.h

```

00001
00002 /*
00003  * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004  * Alexander Warg <warg@os.inf.tu-dresden.de>,
00005  * Frank Mehnert <fm3@os.inf.tu-dresden.de>
00006  * economic rights: Technische Universität Dresden (Germany)
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009
00010 #ifndef __L4UTIL__UTIL_H__
00011 #define __L4UTIL__UTIL_H__
00012
00013 #include <l4/sys/types.h>
00014 #include <l4/sys/compiler.h>
00015 #include <l4/sys/ipc.h>
00016
00017
00021
00022 L4_BEGIN_DECLS
00023
00034 L4_CV l4_timeout_s l4util_micros2l4to(l4_uint64_t us) L4_NOTHROW;
00035
00041 L4_CV void l4_sleep(l4_uint32_t ms) L4_NOTHROW;
00042
00049 L4_CV void l4_usleep(l4_uint64_t us) L4_NOTHROW;
00050
00056 L4_INLINE void l4_sleep_forever(void) L4_NOTHROW L4_NORETURN;
00057
00065 L4_INLINE void
00066 l4_touch_ro(const void *addr, unsigned size) L4_NOTHROW;
00067
00075 L4_INLINE void
00076 l4_touch_rw(const void *addr, unsigned size) L4_NOTHROW;
00077
00078
00079
00080 /*
00081  * Implementations
00082  */
00083
00084 L4_INLINE void
00085 l4_sleep_forever(void) L4_NOTHROW
00086 {
00087     for (;;)
00088         l4_ipc_sleep(L4_IPC_NEVER);
00089 }
00090
00091 L4_INLINE void
00092 l4_touch_ro(const void *addr, unsigned size) L4_NOTHROW
00093 {

```

```

00094  l4_addr_t b, e;
00095
00096  b = l4_trunc_page((l4_addr_t)addr);
00097  e = l4_trunc_page((l4_addr_t)addr + size - 1);
00098
00099  for (; b <= e; b += L4_PAGESIZE)
00100      (void) (*(volatile char *)b);
00101 }
00102
00103
00104 L4_INLINE void
00105 l4_touch_rw(const void *addr, unsigned size) L4_NOTHROW
00106 {
00107     l4_addr_t b, e;
00108
00109     b = l4_trunc_page((l4_addr_t)addr);
00110     e = l4_trunc_page((l4_addr_t)addr + size - 1);
00111
00112     for (; b <= e; b += L4_PAGESIZE)
00113         *(volatile char *)b |= 0;
00114 }
00115
00116 L4_END_DECLS
00117
00118 #endif /* __L4UTIL__UTIL_H__ */

```

16.689 l4/utrace/ring_buffer File Reference

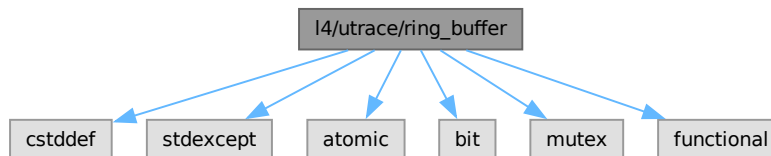
Shared-memory multiple-producer multiple-consumer head-drop ring buffer implementation.

```

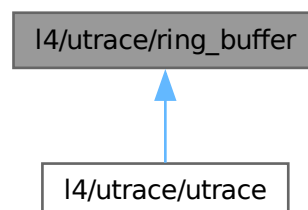
#include <cstdint>
#include <stdexcept>
#include <atomic>
#include <bit>
#include <mutex>
#include <functional>

```

Include dependency graph for ring_buffer:



This graph shows which files directly or indirectly include this file:



Data Structures

- class `utrace::Ring_buffer< SEQUENCE_TYPE >`
Ring buffer.
- class `utrace::Ring_status< SEQUENCE_TYPE >`
Ring buffer status area.
- class `utrace::Ring_slot< ITEM_TYPE, GENERATION_PTR >`
Ring buffer slot.
- class `utrace::Ring_buffer_producer< SEQUENCE_TYPE, ITEM_TYPE, GENERATION_PTR >`
Ring buffer producer.
- class `utrace::Ring_buffer_consumer_raw< SEQUENCE_TYPE, ITEM_TYPE, GENERATION_PTR >`
Polling ring buffer consumer.
- class `utrace::Ring_buffer_consumer< SEQUENCE_TYPE, ITEM_TYPE, GENERATION_PTR >`
Blocking ring buffer consumer.

16.689.1 Detailed Description

Shared-memory multiple-producer multiple-consumer head-drop ring buffer implementation.

This implements a generic ring buffer for transferring fixed-sized items in shared memory between producers and consumers without coupling. This means that the pace of producers is never affected by the pace of consumers (producers never block and never wait for the consumers).

When consumers are unable to dequeue items from the ring buffer quickly enough, they lose access to the oldest items (head-drop policy), but this condition is always detected.

When there are more producers than the total number of items actively enqueueing items into the ring buffer, the state of the ring buffer is undefined. When the sequence/generation counter of the items in the ring buffer overflow, the state of the ring buffer is undefined.

Definition in file [ring_buffer](#).

16.690 ring_buffer

[Go to the documentation of this file.](#)

```
00001 // vim:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * Copyright (C) 2026 Kernkonzept GmbH.
00004  * Author(s): Martin Decky <martin.decky@kernkonzept.com>
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008
00028
00029 #pragma once
00030
00031 #include <cstdint>
00032 #include <stdexcept>
00033 #include <atomic>
00034 #include <bit>
00035 #include <mutex>
00036 #include <functional>
00037
00038 namespace utrace {
00039
00041 #if defined(ARCH_arm64)
00042     static constexpr size_t stable_cache_alignment = 256U;
00043 #elif defined(ARCH_ppc32)
00044     static constexpr size_t stable_cache_alignment = 128U;
00045 #else
```

```

00046     static constexpr size_t stable_cache_alignment = 64U;
00047 #endif
00048
00049 #ifdef __GCC_DESTRUCTIVE_SIZE
00050 static_assert(stable_cache_alignment >= __GCC_DESTRUCTIVE_SIZE,
00051              "Stable cache alignment not sufficient");
00052 #endif // __GCC_DESTRUCTIVE_SIZE
00053
00055 static constexpr size_t string_buffer_size = 1024U;
00056
00065 template<typename SEQUENCE_TYPE>
00066 class Ring_buffer
00067 {
00068 public:
00069     using Sequence = SEQUENCE_TYPE;
00070     using Generation = std::atomic<Sequence>;
00071
00073     static constexpr Sequence const nil = 0U;
00074
00086     static void check_items(size_t const items)
00087     {
00088         if (items == 0)
00089             throw std::invalid_argument("Zero items not supported.");
00090
00091         if (items != std::bit_ceil(items))
00092             throw std::invalid_argument("Number of items must be power of two.");
00093     }
00094
00106     static void check_mask(size_t const mask)
00107     {
00108         auto items = mask + 1;
00109         if (items != std::bit_ceil(items))
00110             throw std::length_error("Invalid mask.");
00111     }
00112 };
00113
00127 template<typename SEQUENCE_TYPE>
00128 class Ring_status
00129 {
00130 public:
00131     using Sequence = SEQUENCE_TYPE;
00132     using Generation = typename Ring_buffer<Sequence>::Generation;
00133
00134     template<typename S, typename T, auto G> friend class Ring_buffer_producer;
00135     template<typename S, typename T, auto G>
00136     friend class Ring_buffer_consumer_raw;
00137
00145     unsigned version() const
00146     { return _version; }
00147
00155     size_t alignment() const
00156     { return _alignment; }
00157
00159     size_t mask() const
00160     { return _mask; }
00161
00163     size_t items() const
00164     { return _mask + 1; }
00165
00166 private:
00168     unsigned _version;
00169
00171     size_t _alignment;
00172
00174     alignas(stable_cache_alignment) size_t _mask;
00175
00177     alignas(stable_cache_alignment) Generation _tail;
00178 };
00179
00189 template<typename ITEM_TYPE>
00190 static constexpr size_t ring_slot_alignment = std::bit_ceil(sizeof(ITEM_TYPE));
00191
00206 template<typename ITEM_TYPE, auto GENERATION_PTR>
00207 class alignas(ring_slot_alignment<ITEM_TYPE>) Ring_slot
00208 {
00209 public:
00210     using This = Ring_slot;
00211     using Item = ITEM_TYPE;
00212
00213     template<typename S, typename T, auto G> friend class Ring_buffer_producer;
00214     template<typename S, typename T, auto G>
00215     friend class Ring_buffer_consumer_raw;
00216
00224     static size_t size(size_t const items)
00225     { return items * sizeof(This); }
00226
00227 private:

```

```

00230     auto generation() const noexcept
00231     { return std::atomic_ref(_data.*GENERATION_PTR); }
00232
00233
00234     Item _data;
00235 };
00236
00247 template<typename SEQUENCE_TYPE, typename ITEM_TYPE, auto GENERATION_PTR>
00248 class Ring_buffer_producer : public Ring_buffer<SEQUENCE_TYPE>
00249 {
00250 public:
00251     using This = Ring_buffer_producer;
00252     using Super = Ring_buffer<SEQUENCE_TYPE>;
00253
00254     using Item = ITEM_TYPE;
00255
00256     using Status = Ring_status<SEQUENCE_TYPE>;
00257     using Slot = Ring_slot<Item, GENERATION_PTR>;
00258
00274     Ring_buffer_producer(Status &status, Slot &slots, unsigned const version,
00275                          size_t const items) : _status(status), _slots(slots)
00276     {
00277         This::check_items(items);
00278
00279         _status._version = version;
00280         _status._alignment = stable_cache_alignment;
00281         _status._mask = items - 1;
00282         _status._tail = Super::nil;
00283
00284         for (size_t i = 0; i < items; ++i)
00285             _slots[i].generation() = Super::nil;
00286     }
00287
00289     size_t items() const
00290     { return _status.items(); }
00291
00300     void enqueue(Item const &item) const
00301     {
00302         auto next = ++_status._tail;
00303         auto index = next & _status._mask;
00304
00305         auto &generation = _slots[index].generation();
00306         auto &slot = _slots[index]._data;
00307
00308         generation.store(Super::nil, std::memory_order_release);
00309         slot = item;
00310         generation.store(next, std::memory_order_release);
00311     }
00312
00313 private:
00314     Ring_buffer_producer() = delete;
00315
00316     Status &_status;
00317     Slot &_slots;
00318 };
00319
00333 template<typename SEQUENCE_TYPE, typename ITEM_TYPE, auto GENERATION_PTR>
00334 class Ring_buffer_consumer_raw : public Ring_buffer<SEQUENCE_TYPE>
00335 {
00336 public:
00337     using This = Ring_buffer_consumer_raw;
00338     using Super = Ring_buffer<SEQUENCE_TYPE>;
00339
00340     using Sequence = SEQUENCE_TYPE;
00341     using Item = ITEM_TYPE;
00342
00343     using Status = Ring_status<Sequence>;
00344     using Slot = Ring_slot<Item, GENERATION_PTR>;
00345
00353     enum class Drop_policy
00354     {
00364         Conservative = 0,
00374         Minimal
00375     };
00376
00378     enum class State
00379     {
00380         Idle = 0,    ///< No item dequeued.
00381         Ready,      ///< An item has been dequeued.
00382         Dropped     ///< Some items have been dropped.
00383     };
00384
00394     Ring_buffer_consumer_raw(Status const &status, Slot const *slots)
00395     : _status(status), _slots(slots)
00396     { This::check_mask(_status._mask); }
00397
00399     size_t items() const

```

```

00400 { return _status.items(); }
00401
00435 State peek(Item &item, Drop_policy policy, Sequence &current,
00436             Sequence *drops = nullptr) const
00437 {
00438     if (drops)
00439         *drops = 0;
00440
00441     // First access to the ring buffer.
00442     if (current == Super::nil)
00443         current = 1;
00444
00445     auto index = current & _status._mask;
00446
00447     auto const &generation = _slots[index].generation();
00448     auto const &slot = _slots[index]._data;
00449
00450     auto seen = generation.load();
00451     if (seen < current)
00452     {
00453         // Expected item not produced yet.
00454         return State::Idle;
00455     }
00456
00457     if (seen > current)
00458         goto drop;
00459
00460     item = slot;
00461     seen = generation.load();
00462
00463     // Make sure the item has not been overwritten while we have been copying
00464     // the data.
00465     if (seen == current)
00466     {
00467         // Consume item.
00468         ++current;
00469         return State::Ready;
00470     }
00471
00472 drop:
00473     // Seen > current -> drop detected.
00474     Sequence head = _status._tail;
00475
00476     // Conservative drop policy continues at the tail.
00477     // Minimal drop policy continues at the head.
00478     if (policy == Drop_policy::Minimal)
00479         head -= _status._mask;
00480
00481     // How many items have been dropped.
00482     if (drops)
00483         *drops = head - current;
00484
00485     current = head;
00486     return State::Dropped;
00487 }
00488
00489 private:
00490     Ring_buffer_consumer_raw() = delete;
00491
00492     Status const &_status;
00493     Slot const *_slots;
00494 };
00495
00509 template<typename SEQUENCE_TYPE, typename ITEM_TYPE, auto GENERATION_PTR>
00510 class Ring_buffer_consumer
00511     : public Ring_buffer_consumer_raw<SEQUENCE_TYPE, ITEM_TYPE, GENERATION_PTR>
00512 {
00513 public:
00514     using Sequence = SEQUENCE_TYPE;
00515     using Item = ITEM_TYPE;
00516
00517     using Super = Ring_buffer_consumer_raw<Sequence, Item, GENERATION_PTR>;
00518
00519     using Drop_policy = Super::Drop_policy;
00520     using State = Super::State;
00521     using Status = Ring_status<Sequence>;
00522     using Slot = Ring_slot<Item, GENERATION_PTR>;
00523
00524     using Yield = std::function<bool (Sequence)>;
00525
00535     Ring_buffer_consumer(Status const &status, Slot const *slots)
00536         : Super(status, slots)
00537     {}
00538
00560     State peek(Item &item, Drop_policy policy, Sequence *drops = nullptr)
00561     {
00562         const std::lock_guard guard(_lock);

```

```

00563     return Super::peek(item, policy, _current, drops);
00564 }
00565
00595 size_t dequeue(Item items[], size_t capacity, size_t burst,
00596               Drop_policy policy, Yield const &yield,
00597               Sequence *drops = nullptr)
00598 {
00599     if (drops)
00600         *drops = 0;
00601
00602     size_t count = 0;
00603     size_t idle_cycles = 0;
00604
00605     for (;;)
00606     {
00607         Sequence current_drops;
00608         auto status = peek(items[count], policy, &current_drops);
00609
00610         if (status == State::Dropped)
00611         {
00612             // Some items have been dropped. We need to notify the caller
00613             // about it. Thus return with the items already dequeued.
00614
00615             if (drops)
00616                 *drops = current_drops;
00617
00618             break;
00619         }
00620
00621         if (status == State::Ready)
00622         {
00623             // Next item dequeued. Return if there is no space for more items
00624             // in the array.
00625
00626             ++count;
00627             if (count == capacity)
00628                 break;
00629
00630             continue;
00631         }
00632
00633         if (status == State::Idle)
00634         {
00635             // No next item has been produced yet. Return if we have already
00636             // dequeued enough items. Otherwise just spin, but allow the yield
00637             // function to implement passive waiting.
00638
00639             if (count >= burst)
00640                 break;
00641
00642             ++idle_cycles;
00643             if (yield(idle_cycles))
00644                 idle_cycles = 0;
00645         }
00646     }
00647
00648     return count;
00649 }
00650
00651 private:
00652     Ring_buffer_consumer() = delete;
00653     State peek(Item &, Drop_policy, Sequence &, Sequence *) const = delete;
00654
00655     std::mutex _lock;
00656
00657     Sequence _current = 0;
00660 };
00661
00662 } // namespace utrace

```

16.691 I4/utrace/utrace File Reference

[L4Re](#) tracebuffer access.

```

#include <version>
#include <bit>
#include <string>
#include <optional>

```



```

00022 #include <string>
00023 #include <optional>
00024 #include <l4/sys/ktrace_events.h>
00025 #include <l4/sys/consts.h>
00026 #include <l4/sys/capability>
00027 #include <l4/sys/debugger>
00028 #include <l4/utrace/ring_buffer>
00029
00030 #if defined(__cpp_lib_generator)
00031 #include <generator>
00032 #else
00033 #include <vector>
00034 #endif
00035
00036 namespace utrace {
00037
00044 class Tracebuffer
00045 {
00046 public:
00047     using Sequence = L4_ktrace_t__Mword;
00048     using Item = l4_tracebuffer_entry_t;
00049
00050     using Buffer = Ring_buffer_consumer<Sequence, Item, &Item::_number>;
00051     using Drop_policy = Buffer::Drop_policy;
00052     using Status = Buffer::Status;
00053     using Slot = Buffer::Slot;
00054
00055     static_assert(sizeof(Status) <= L4_PAGESIZE);
00056
00058     struct Index_desc
00059     {
00060         unsigned index;
00061         std::string name;
00062         std::string shortname;
00063     };
00064
00074     static void validate();
00075
00077     static unsigned version();
00078
00080     static std::endian endianness();
00081
00090     static std::optional<Index_desc> index(unsigned idx);
00091
00098 #if defined(__cpp_lib_generator)
00099     static std::generator<Index_desc> indexes();
00100 #else
00101     static std::vector<Index_desc> indexes();
00102 #endif
00103
00105     static Tracebuffer &instance();
00106
00108     size_t items() const;
00109
00129     size_t dequeue(Item *items, size_t capacity, size_t burst, Drop_policy policy,
00130                    Sequence *drops = nullptr);
00131
00132 private:
00133     static constexpr unsigned tracebuffer_version = 0;
00134     static constexpr size_t yield_cycles = 100;
00135
00140     class Factory
00141     {
00142     public:
00144         static Factory &instance();
00145
00147         Status &status();
00148
00150         Slot *slots();
00151
00152     private:
00163         Factory();
00164
00171         ~Factory();
00172
00186         static l4_fpage_t reserve(size_t size);
00187
00198         static Status *map_status(l4_fpage_t fpage);
00199
00211         static Slot *map_slots(l4_fpage_t fpage, size_t items);
00212
00213         l4_fpage_t _fpage_status;
00214         l4_fpage_t _fpage_slots;
00215
00216         Status *_status;
00217         Slot *_slots;
00218     };

```

```

00219
00220     Tracebuffer() = delete;
00221
00227     Tracebuffer(Factory &factory);
00228
00229     static L4::Cap<L4::Debugger> _jdb;
00230     Buffer _buffer;
00231 };
00232
00233 }; // namespace utrace

```

16.693 vbus

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2011 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00004  *     economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include <l4/vbus/vbus.h>
00011 #include <l4/vbus/vbus_pm.h>
00012 #include <l4/sys/icu>
00013
00014 #include <l4/re/dataspace>
00015 #include <l4/re/dma_space>
00016 #include <l4/re/event>
00017 #include <l4/re/inhibitor>
00018
00036
00040 namespace L4vbus {
00041
00042     class Vbus;
00043
00049     template<typename DEC>
00050     class Pm
00051     {
00052     private:
00053         DEC const *self() const { return static_cast<DEC const *>(this); }
00054         DEC *self() { return static_cast<DEC *>(this); }
00055     public:
00063         int pm_suspend() const
00064         { return l4vbus_pm_suspend(self()->bus_cap().cap(), self()->dev_handle()); }
00065
00074         int pm_resume() const
00075         { return l4vbus_pm_resume(self()->bus_cap().cap(), self()->dev_handle()); }
00076     };
00077
00078
00083     class Device : public Pm<Device>
00084     {
00085     public:
00089         Device() : _dev(L4VBUS_NULL) {}
00090
00100         Device(L4::Cap<Vbus> bus, l4vbus_device_handle_t dev)
00101         : _bus(bus), _dev(dev) {}
00102
00107         L4::Cap<Vbus> bus_cap() const { return _bus; }
00108
00116         l4vbus_device_handle_t dev_handle() const { return _dev; }
00117
00118
00148         int device_by_hid(Device *child, char const *hid,
00149                          int depth = L4VBUS_MAX_DEPTH,
00150                          l4vbus_device_t *devinfo = 0) const
00151         {
00152             child->_bus = _bus;
00153             return l4vbus_get_device_by_hid(_bus.cap(), _dev, &child->_dev, hid,
00154                                           depth, devinfo);
00155         }
00156
00171         int next_device(Device *child, int depth = L4VBUS_MAX_DEPTH,
00172                        l4vbus_device_t *devinfo = 0) const
00173         {
00174             child->_bus = _bus;
00175             return l4vbus_get_next_device(_bus.cap(), _dev, &child->_dev, depth,
00176                                         devinfo);
00177         }
00178
00189         int device(l4vbus_device_t *devinfo) const
00190         { return l4vbus_get_device(_bus.cap(), _dev, devinfo); }

```



```

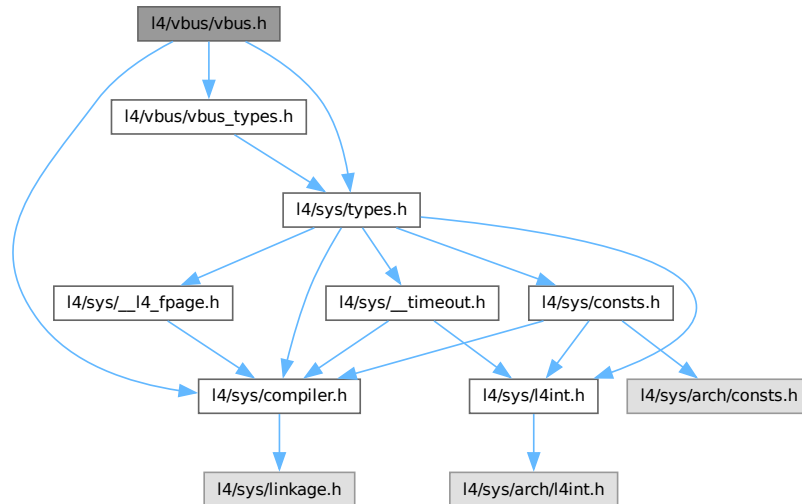
00191
00209 int get_resource(unsigned res_idx, l4vbus_resource_t *res) const
00210 {
00211     return l4vbus_get_resource(_bus.cap(), _dev, res_idx, res);
00212 }
00213
00223 int is_compatible(char const *cid) const
00224 { return l4vbus_is_compatible(_bus.cap(), _dev, cid); }
00225
00230 bool operator == (Device const &o) const
00231 {
00232     return _bus == o._bus && _dev == o._dev;
00233 }
00234
00239 bool operator != (Device const &o) const
00240 {
00241     return _bus != o._bus || _dev != o._dev;
00242 }
00243
00244 protected:
00245     L4::Cap<Vbus> _bus;
00247     l4vbus_device_handle_t _dev;
00248 };
00249
00260 class Icu : public Device
00261 {
00262 public:
00264     enum Src_types
00265     {
00273         Src_dev_handle = L4VBUS_ICU_SRC_DEV_HANDLE
00274     };
00275
00285 int vicu(L4::Cap<L4::Icu> icu) const
00286 {
00287     return l4vbus_vicu_get_cap(_bus.cap(), _dev, icu.cap());
00288 }
00289 };
00290
00298 class Vbus : public L4::Kobject_3t<Vbus, L4Re::Dataspace, L4Re::Inhibitor, L4Re::Event>
00299 {
00300 public:
00301
00311 int request_ioport(l4vbus_resource_t *res) const
00312 {
00313     return l4vbus_request_ioport(cap(), res);
00314 }
00315
00323 int release_ioport(l4vbus_resource_t *res) const
00324 {
00325     return l4vbus_release_ioport(cap(), res);
00326 }
00327
00336 Device root() const
00337 {
00338     return Device(L4::Cap<Vbus>(cap()), L4VBUS_ROOT_BUS);
00339 }
00340
00357 int assign_dma_domain(unsigned domain_id, unsigned flags,
00358                       L4::Cap<L4Re::Dma_space> dma_space) const
00359 {
00360     flags |= L4VBUS_DMAD_L4RE_DMA_SPACE;
00361     flags &= ~L4VBUS_DMAD_KERNEL_DMA_SPACE;
00362     return l4vbus_assign_dma_domain(cap(), domain_id, flags, dma_space.cap());
00363 }
00364
00382 int assign_dma_domain(unsigned domain_id, unsigned flags,
00383                       L4::Cap<L4::Task> dma_space) const
00384 {
00385     flags |= L4VBUS_DMAD_KERNEL_DMA_SPACE;
00386     flags &= ~L4VBUS_DMAD_L4RE_DMA_SPACE;
00387     return l4vbus_assign_dma_domain(cap(), domain_id, flags, dma_space.cap());
00388 }
00389
00390 typedef L4::Typeid::Raw_ipc<Vbus> Rpcs;
00391 };
00392
00393 } // namespace L4vbus

```

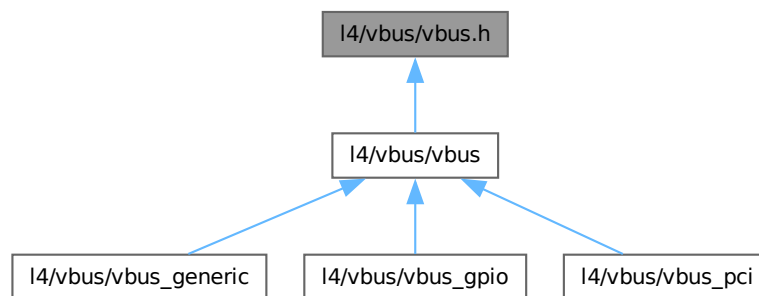
16.694 I4/vbus/vbus.h File Reference

Description of the vbus C API.

```
#include <l4/sys/compiler.h>
#include <l4/vbus/vbus_types.h>
#include <l4/sys/types.h>
Include dependency graph for vbus.h:
```



This graph shows which files directly or indirectly include this file:



Enumerations

- enum { [L4VBUS_NULL](#) = -1L , [L4VBUS_ROOT_BUS](#) = 0 }
Constants for device nodes.
- enum [l4vbus_icu_src_types](#) { [L4VBUS_ICU_SRC_DEV_HANDLE](#) = 1ULL << 63 }
Flags that can be used with the ICU on a vbus device.
- enum [L4vbus_dma_domain_assign_flags](#) { [L4VBUS_DMAD_UNBIND](#) = 0 , [L4VBUS_DMAD_BIND](#) = 1 , [L4VBUS_DMAD_L4RE_DMA_SPACE](#) = 0 , [L4VBUS_DMAD_KERNEL_DMA_SPACE](#) = 2 }
Flags for [l4vbus_assign_dma_domain\(\)](#).

Functions

- [L4_BEGIN_DECLS](#) `int l4vbus_get_device_by_hid (l4_cap_idx_t vbus, l4vbus_device_handle_t parent, l4vbus_device_handle_t *child, char const *hid, int depth, l4vbus_device_t *devinfo)`
Find a device by the hardware interface identifier (HID).
- `int l4vbus_get_next_device (l4_cap_idx_t vbus, l4vbus_device_handle_t parent, l4vbus_device_handle_t *child, int depth, l4vbus_device_t *devinfo)`
Find next child following `child`.
- `int l4vbus_get_device (l4_cap_idx_t vbus, l4vbus_device_handle_t dev, l4vbus_device_t *devinfo)`
Obtain detailed information about a Vbus device.
- `int l4vbus_get_resource (l4_cap_idx_t vbus, l4vbus_device_handle_t dev, unsigned res_idx, l4vbus_resource_t *res)`
Obtain the resource description of an individual device resource.
- `int l4vbus_is_compatible (l4_cap_idx_t vbus, l4vbus_device_handle_t dev, char const *cid)`
Check if the given device has a compatibility ID (CID) or HID that matches `cid`.
- `int l4vbus_get_hid (l4_cap_idx_t vbus, l4vbus_device_handle_t dev, char *hid, unsigned long max_len)`
Get the HID (hardware identifier) of a device.
- `int l4vbus_get_adr (l4_cap_idx_t vbus, l4vbus_device_handle_t dev, l4_uint32_t *adr)`
Get the bus-specific address of a device.
- `int l4vbus_request_ioport (l4_cap_idx_t vbus, l4vbus_resource_t const *res)`
Request an IO port resource.
- `int l4vbus_assign_dma_domain (l4_cap_idx_t vbus, unsigned domain_id, unsigned flags, l4_cap_idx_t dma_space)`
Bind or unbind a kernel [DMA space](#) or a [L4Re::Dma_space](#) to a DMA domain.
- `int l4vbus_release_ioport (l4_cap_idx_t vbus, l4vbus_resource_t const *res)`
Release a previously requested IO port resource.
- `int l4vbus_vicu_get_cap (l4_cap_idx_t vbus, l4vbus_device_handle_t icu, l4_cap_idx_t cap)`
Get capability of ICU.

16.694.1 Detailed Description

Description of the vbus C API.

Definition in file [vbus.h](#).

16.694.2 Enumeration Type Documentation

16.694.2.1 anonymous enum

`anonymous enum`

Constants for device nodes.

Enumerator

| | |
|-----------------|--------------------------|
| L4VBUS_NULL | NULL device. |
| L4VBUS_ROOT_BUS | Root device on the vbus. |

Definition at line 20 of file [vbus.h](#).

16.694.2.2 l4vbus_icu_src_types

enum [l4vbus_icu_src_types](#)

Flags that can be used with the ICU on a vbus device.

Enumerator

| | |
|---------------------------|---|
| L4VBUS_ICU_SRC_DEV_HANDLE | Flag to denote that the value should be interpreted as a device handle. This flag may be used in the <code>source</code> parameter in l4_icu_msi_info() to denote that the ICU should interpret the source ID as a device handle. |
|---------------------------|---|

Definition at line 26 of file [vbus.h](#).

16.695 vbus.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00004  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00013 #pragma once
00014
00015 #include <l4/sys/compiler.h>
00016 #include <l4/vbus/vbus_types.h>
00017 #include <l4/sys/types.h>
00018
00020 enum {
00021     L4VBUS_NULL = -1L,
00022     L4VBUS_ROOT_BUS = 0,
00023 };
00024
00026 enum l4vbus_icu_src_types {
00033     L4VBUS_ICU_SRC_DEV_HANDLE = 1ULL « 63
00034 };
00035
00053
00054 L4_BEGIN_DECLS
00055
00063 int L4_CV
00064 l4vbus_get_device_by_hid(l4_cap_idx_t vbus, l4vbus_device_handle_t parent,
00065                          l4vbus_device_handle_t *child, char const *hid,
00066                          int depth, l4vbus_device_t *devinfo);
00067
00083 int L4_CV
00084 l4vbus_get_next_device(l4_cap_idx_t vbus, l4vbus_device_handle_t parent,
00085                        l4vbus_device_handle_t *child, int depth,
00086                        l4vbus_device_t *devinfo);
00087
00101 int L4_CV
00102 l4vbus_get_device(l4_cap_idx_t vbus, l4vbus_device_handle_t dev,
00103                  l4vbus_device_t *devinfo);
00104
00113 int L4_CV
00114 l4vbus_get_resource(l4_cap_idx_t vbus, l4vbus_device_handle_t dev,
00115                    unsigned res_idx, l4vbus_resource_t *res);
00116
00117
00124 int L4_CV
00125 l4vbus_is_compatible(l4_cap_idx_t vbus, l4vbus_device_handle_t dev,
00126                     char const *cid);
00127
00138 int L4_CV
00139 l4vbus_get_hid(l4_cap_idx_t vbus, l4vbus_device_handle_t dev, char *hid,
00140               unsigned long max_len);
00141
00152 int L4_CV
00153 l4vbus_get_adr(l4_cap_idx_t vbus, l4vbus_device_handle_t dev, l4_uint32_t *adr);

```

```

00154
00168 int L4_CV
00169 l4vbus_request_ioport(l4_cap_idx_t vbus, l4vbus_resource_t const *res);
00170
00174 enum L4vbus_dma_domain_assign_flags
00175 {
00177     L4VBUS_DMAD_UNBIND = 0,
00179     L4VBUS_DMAD_BIND   = 1,
00181     L4VBUS_DMAD_L4RE_DMA_SPACE = 0,
00183     L4VBUS_DMAD_KERNEL_DMA_SPACE = 2,
00184 };
00185
00207 int L4_CV
00208 l4vbus_assign_dma_domain(l4_cap_idx_t vbus, unsigned domain_id,
00209                          unsigned flags, l4_cap_idx_t dma_space);
00210
00219 int L4_CV
00220 l4vbus_release_ioport(l4_cap_idx_t vbus, l4vbus_resource_t const *res);
00221
00231 int L4_CV
00232 l4vbus_vicu_get_cap(l4_cap_idx_t vbus, l4vbus_device_handle_t icu,
00233                    l4_cap_idx_t cap);
00234
00235 L4_END_DECLS
00236

```

16.696 vbus_generic

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2009 Alexander Warg <warg@os.inf.tu-dresden.de>,
00004  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009
00010 #pragma once
00011
00012 #include <l4/cxx/ipc_stream>
00013 #include <l4/vbus/vbus_types.h>
00014 #include <l4/vbus/vbus>
00015
00016 inline void
00017 l4vbus_device_msg(l4vbus_device_handle_t handle, l4_uint32_t op,
00018                  L4::Ipc::Iostream &s)
00019 {
00020     s « handle « op;
00021 }

```

16.697 vbus_gpio

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003  * (c) 2011 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include <l4/vbus/vbus>
00011 #include <l4/vbus/vbus_gpio.h>
00012
00018
00019 namespace L4vbus {
00020
00026 class Gpio_pin : public Device
00027 {
00028 public:
00029     Gpio_pin(Device const &dev, unsigned pin)
00030         : Device(dev), _pin(pin)
00031     {}
00032
00038     int get() const
00039     {
00040         return l4vbus_gpio_get(_bus.cap(), _dev, _pin);
00041     }

```

```

00042
00049 int set(int value) const
00050 {
00051     return l4vbus_gpio_set(_bus.cap(), _dev, _pin, value);
00052 }
00053
00064 int setup(unsigned mode, unsigned value) const
00065 {
00066     return l4vbus_gpio_setup(_bus.cap(), _dev, _pin, mode, value);
00067 }
00068
00075 int config_pull(unsigned mode) const
00076 {
00077     return l4vbus_gpio_config_pull(_bus.cap(), _dev, _pin, mode);
00078 }
00079
00089 int config_pad(unsigned func, unsigned value) const
00090 {
00091     return l4vbus_gpio_config_pad(_bus.cap(), _dev, _pin, func, value);
00092 }
00093
00102 int config_get(unsigned func, unsigned *value) const
00103 {
00104     return l4vbus_gpio_config_get(_bus.cap(), _dev, _pin, func, value);
00105 }
00106
00112 int to_irq() const
00113 {
00114     return l4vbus_gpio_to_irq(_bus.cap(), _dev, _pin);
00115 }
00116
00122 unsigned pin() const { return _pin; }
00123
00124 protected:
00125     Gpio_pin() {}
00126     unsigned _pin;
00127 };
00128
00133 class Gpio_module : public Device
00134 {
00135 public:
00136     Gpio_module(Device dev)
00137     : Device(dev)
00138     {}
00139
00146 struct Pin_slice
00147 {
00148     Pin_slice(unsigned offset, unsigned mask) : offset(offset), mask(mask) {}
00149     unsigned offset, mask;
00150 };
00151
00166 int setup(Pin_slice const &mask, unsigned mode, unsigned value) const
00167 {
00168     return l4vbus_gpio_multi_setup(_bus.cap(), _dev, mask.offset, mask.mask,
00169                                     mode, value);
00170 }
00171
00185 int config_pad(Pin_slice const &mask, unsigned func, unsigned value) const
00186 {
00187     return l4vbus_gpio_multi_config_pad(_bus.cap(), _dev, mask.offset,
00188                                         mask.mask, func, value);
00189 }
00190
00201 int get(unsigned offset, unsigned *data) const
00202 {
00203     return l4vbus_gpio_multi_get(_bus.cap(), _dev, offset, data);
00204 }
00205
00217 int set(Pin_slice const &mask, unsigned data)
00218 {
00219     return l4vbus_gpio_multi_set(_bus.cap(), _dev, mask.offset,
00220                                   mask.mask, data);
00221 }
00222
00229 Gpio_pin pin(unsigned pin) const
00230 {
00231     return Gpio_pin(*this, pin);
00232 }
00233
00234 protected:
00235     Gpio_module() {}
00236 };
00237
00238 }

```

16.698 vbus_gpio-ops.h

```

00001 /*
00002  * (c) 2011 Alexander Warg <warg@os.inf.tu-dresden.de>
00003  *      economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00008 #pragma once
00009
00010 #include <l4/vbus/vbus_interfaces.h>
00011
00012 enum L4vbus_gpio_op
00013 {
00014     L4VBUS_GPIO_OP_SETUP = L4VBUS_INTERFACE_GPIO « L4VBUS_IFACE_SHIFT,
00015     L4VBUS_GPIO_OP_CONFIG_PAD,
00016     L4VBUS_GPIO_OP_CONFIG_GET,
00017     L4VBUS_GPIO_OP_GET,
00018     L4VBUS_GPIO_OP_SET,
00019     L4VBUS_GPIO_OP_MULTI_SETUP,
00020     L4VBUS_GPIO_OP_MULTI_CONFIG_PAD,
00021     L4VBUS_GPIO_OP_MULTI_GET,
00022     L4VBUS_GPIO_OP_MULTI_SET,
00023     L4VBUS_GPIO_OP_TO_IRQ,
00024     L4VBUS_GPIO_OP_CONFIG_PULL
00025 };

```

16.699 vbus_gpio.h

```

00001 /*
00002  * (c) 2011 Alexander Warg <warg@os.inf.tu-dresden.de>
00003  *      economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include <l4/sys/compiler.h>
00010 #include <l4/sys/types.h>
00011 #include <l4/vbus/vbus_types.h>
00012
00018
00019 L4_BEGIN_DECLS
00020
00024 enum L4vbus_gpio_generic_func
00025 {
00026     L4VBUS_GPIO_SETUP_INPUT = 0x100,
00027     L4VBUS_GPIO_SETUP_OUTPUT = 0x200,
00028     L4VBUS_GPIO_SETUP_IRQ = 0x300,
00029 };
00030
00034 enum L4vbus_gpio_pull_modes
00035 {
00036     L4VBUS_GPIO_PIN_PULL_NONE = 0x100,
00037     L4VBUS_GPIO_PIN_PULL_UP = 0x200,
00038     L4VBUS_GPIO_PIN_PULL_DOWN = 0x300,
00039 };
00040
00048 int L4_CV
00049 l4vbus_gpio_setup(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00050                  unsigned pin, unsigned mode, int value);
00051
00059 int L4_CV
00060 l4vbus_gpio_config_pull(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00061                        unsigned pin, unsigned mode);
00062
00070 int L4_CV
00071 l4vbus_gpio_config_pad(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00072                       unsigned pin, unsigned func, unsigned value);
00073
00081 int L4_CV
00082 l4vbus_gpio_config_get(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00083                       unsigned pin, unsigned func, unsigned *value);
00084
00092 int L4_CV
00093 l4vbus_gpio_get(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00094                unsigned pin);
00095
00103 int L4_CV
00104 l4vbus_gpio_set(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00105                unsigned pin, int value);

```

```

00106
00115 int L4_CV
00116 l4vbus_gpio_multi_setup(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00117                          unsigned offset, unsigned mask,
00118                          unsigned mode, unsigned value);
00119
00128 int L4_CV
00129 l4vbus_gpio_multi_config_pad(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00130                             unsigned offset, unsigned mask,
00131                             unsigned func, unsigned value);
00132
00139 int L4_CV
00140 l4vbus_gpio_multi_get(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00141                      unsigned offset, unsigned *data);
00142
00151 int L4_CV
00152 l4vbus_gpio_multi_set(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00153                      unsigned offset, unsigned mask, unsigned data);
00154
00162 int L4_CV
00163 l4vbus_gpio_to_irq(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00164                   unsigned pin);
00165
00167
00168 L4_END_DECLS

```

16.700 vbus_i2c.h

```

00001 /*
00002  * (c) 2009 Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00003  *      economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include <l4/sys/compiler.h>
00010 #include <l4/sys/types.h>
00011 #include <l4/vbus/vbus_types.h>
00012
00013 L4_BEGIN_DECLS
00014
00015 int L4_CV
00016 l4vbus_i2c_write(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00017                 l4_uint16_t addr, l4_uint8_t sub_addr,
00018                 l4_uint8_t *buffer, unsigned long size);
00019
00020 int L4_CV
00021 l4vbus_i2c_read(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00022                l4_uint16_t addr, l4_uint8_t sub_addr,
00023                l4_uint8_t *buffer, unsigned long *size);
00024
00025 L4_END_DECLS

```

16.701 vbus_inhibitor.h

```

00001
00006 #pragma once
00007
00008 enum Vbus_inhibitor
00009 {
00010     L4VBUS_INHIBITOR_SUSPEND = 0,
00011     L4VBUS_INHIBITOR_SHUTDOWN = 1,
00012     L4VBUS_INHIBITOR_REBOOT = L4VBUS_INHIBITOR_SHUTDOWN,
00013     L4VBUS_INHIBITOR_WAKEUP = 2,
00014     L4VBUS_INHIBITOR_MAX
00015 };
00016

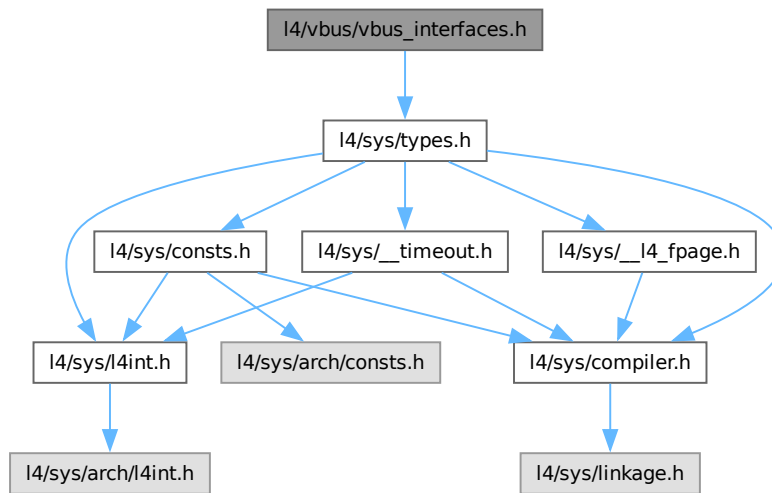
```

16.702 l4/vbus/vbus_interfaces.h File Reference

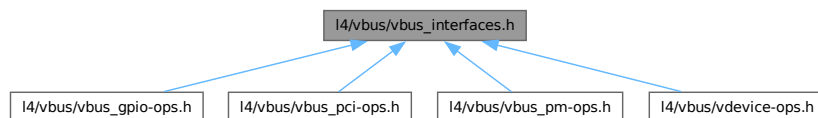
This header contains the definition of VBUS sub-interfaces and convenience functions to work with the interface IDs.


```
#include <l4/sys/types.h>
```

Include dependency graph for vbus_interfaces.h:



This graph shows which files directly or indirectly include this file:



Typedefs

- typedef enum `l4vbus_iface_type_t` `l4vbus_iface_type_t`
Different sub-interfaces a vbus device may support.

Enumerations

- enum `l4vbus_iface_type_t` {
`L4VBUS_INTERFACE_ICU` = 0 , `L4VBUS_INTERFACE_GPIO` , `L4VBUS_INTERFACE_PCI` , `L4VBUS_INTERFACE_PCIDEV` ,
`L4VBUS_INTERFACE_PM` , `L4VBUS_INTERFACE_BUS` , `L4VBUS_INTERFACE_GENERIC` = 0x20 }
Different sub-interfaces a vbus device may support.
- enum { `L4VBUS_IFACE_SHIFT` = 26 }

Functions

- unsigned **l4vbus_subinterface** (unsigned opcode)
Return the ID of the vbus sub-interface.
- unsigned **l4vbus_interface_opcode** (unsigned opcode)
Return the function opcode within the sub-interface of the vbus command.
- int **l4vbus_subinterface_supported** (l4_uint32_t dev_type, l4vbus_iface_type_t iface_type)
Check if a vbus device supports a given sub-interface.

16.702.1 Detailed Description

This header contains the definition of VBUS sub-interfaces and convenience functions to work with the interface IDs.

Definition in file [vbus_interfaces.h](#).

16.702.2 Typedef Documentation

16.702.2.1 l4vbus_iface_type_t

```
typedef enum l4vbus_iface_type_t l4vbus_iface_type_t
```

Different sub-interfaces a vbus device may support.

The IPC interface of vbus devices is divided into functional groups of sub-interfaces. Every device must implement the generic interface which provides general device information. According to the type of device, additional functionality may be supported.

The sub-interface constants are first of all used to divide the function opcode space of the interface into these functional groups (see L4VBUS_IFACE_SHIFT). They also make up a bitmask that specify the type of the device, i.e. from the point of view of the client a device is defined by the kinds of sub-interfaces it supports.

16.702.3 Enumeration Type Documentation

16.702.3.1 anonymous enum

```
anonymous enum
```

Enumerator

| | |
|--------------------|---|
| L4VBUS_IFACE_SHIFT | Sub-interface ID shift. Divides the function opcode sent via IPC into a sub-interface ID and the actual function opcode within the sub-interface. |
|--------------------|---|

Definition at line 48 of file [vbus_interfaces.h](#).

16.702.3.2 l4vbus_iface_type_t

enum [l4vbus_iface_type_t](#)

Different sub-interfaces a vbus device may support.

The IPC interface of vbus devices is divided into functional groups of sub-interfaces. Every device must implement the generic interface which provides general device information. According to the type of device, additional functionality may be supported.

The sub-interface constants are first of all used to divide the function opcode space of the interface into these functional groups (see [L4VBUS_IFACE_SHIFT](#)). They also make up a bitmask that specify the type of the device, i.e. from the point of view of the client a device is defined by the kinds of sub-interfaces it supports.

Enumerator

| | |
|--|----------------------------|
| L4VBUS_INTERFACE_ICU | Interrupt Controller. |
| L4VBUS_INTERFACE_GPIO | GPIO. |
| L4VBUS_INTERFACE_PCI | PCI. |
| L4VBUS_INTERFACE_PCIDEV | PCI Device. |
| L4VBUS_INTERFACE_PM | Power Management. |
| L4VBUS_INTERFACE_BUS | VBus. |
| L4VBUS_INTERFACE_GENERIC | No specific sub interface. |

Definition at line [29](#) of file [vbus_interfaces.h](#).

16.702.4 Function Documentation

16.702.4.1 l4vbus_subinterface_supported()

```
int l4vbus_subinterface_supported (  
    l4\_uint32\_t dev_type,  
    l4vbus\_iface\_type\_t iface_type) [inline]
```

Check if a vbus device supports a given sub-interface.

Parameters

| | |
|-------------------|--|
| <i>dev_type</i> | Device type as reported in l4vbus_device_t . |
| <i>iface_type</i> | Sub-interface type to check for. |

Returns

True if the device supports the sub-interface.

Definition at line [99](#) of file [vbus_interfaces.h](#).

References [L4_INLINE](#), and [L4VBUS_INTERFACE_GENERIC](#).

16.703 vbus_interfaces.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * (c) 2014 Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00003  *
00004  * License: see LICENSE.spdx (in this directory or the directories above)
00005  */
00011 #pragma once
00012
00013 #include <l4/sys/types.h>
00014
00029 typedef enum l4vbus_iface_type_t
00030 {
00032     L4VBUS_INTERFACE_ICU = 0,
00034     L4VBUS_INTERFACE_GPIO,
00036     L4VBUS_INTERFACE_PCI,
00038     L4VBUS_INTERFACE_PCIDEV,
00040     L4VBUS_INTERFACE_PM,
00042     L4VBUS_INTERFACE_BUS,
00044     L4VBUS_INTERFACE_GENERIC = 0x20
00045 } l4vbus_iface_type_t;
00046
00047
00048 enum {
00056     L4VBUS_IFACE_SHIFT = 26
00057 };
00058
00070 L4_INLINE unsigned l4vbus_subinterface(unsigned opcode)
00071 {
00072     return opcode » L4VBUS_IFACE_SHIFT;
00073 }
00074
00086 L4_INLINE unsigned l4vbus_interface_opcode(unsigned opcode)
00087 {
00088     return opcode & ((1 « L4VBUS_IFACE_SHIFT) - 1);
00089 }
00090
00099 L4_INLINE int l4vbus_subinterface_supported(l4_uint32_t dev_type,
00100                                             l4vbus_iface_type_t iface_type)
00101 {
00102     if (iface_type == L4VBUS_INTERFACE_GENERIC)
00103         return 1;
00104
00105     return (dev_type & (1 « iface_type)) ? 1 : 0;
00106 }

```

16.704 vbus_mcspi.h

```

00001 /*
00002  * (c) 2009 Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00003  *     economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include <l4/sys/compiler.h>
00010 #include <l4/sys/types.h>
00011 #include <l4/vbus/vbus_types.h>
00012
00013 L4_BEGIN_DECLS
00014
00015 int L4_CV
00016 l4vbus_mcspi_read(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00017                  unsigned channel, l4_umword_t *value);
00018
00019 int L4_CV
00020 l4vbus_mcspi_write(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00021                   unsigned channel, l4_umword_t value);
00022
00023 L4_END_DECLS

```

16.705 vbus_pci

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-

```

```

00002  /*
00003  * (c) 2014 Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00008 #pragma once
00009
00010 #include <l4/vbus/vbus>
00011 #include <l4/vbus/vbus_pci.h>
00012
00018
00019 namespace L4vbus {
00020
00025 class Pci_host_bridge : public Device
00026 {
00027 public:
00039 int cfg_read(l4_uint32_t bus, l4_uint32_t devfn, l4_uint32_t reg,
00040             l4_uint32_t *value, l4_uint32_t width) const
00041 {
00042     return l4vbus_pci_cfg_read(bus_cap().cap(), _dev, bus,
00043                               devfn, reg, value, width);
00044 }
00045
00046
00058 int cfg_write(l4_uint32_t bus, l4_uint32_t devfn, l4_uint32_t reg,
00059              l4_uint32_t value, l4_uint32_t width) const
00060 {
00061     return l4vbus_pci_cfg_write(bus_cap().cap(), _dev, bus,
00062                                devfn, reg, value, width);
00063 }
00064
00065
00079 int irq_enable(l4_uint32_t bus, l4_uint32_t devfn, int pin,
00080               unsigned char *trigger, unsigned char *polarity) const
00081 {
00082     return l4vbus_pci_irq_enable(bus_cap().cap(), _dev, bus,
00083                                 devfn, pin, trigger, polarity);
00084 }
00085
00086 };
00087
00088
00093 class Pci_dev : public Device
00094 {
00095 public:
00105 int cfg_read(l4_uint32_t reg, l4_uint32_t *value,
00106             l4_uint32_t width) const
00107 {
00108     return l4vbus_pcidev_cfg_read(bus_cap().cap(), _dev, reg, value, width);
00109 }
00110
00111
00121 int cfg_write(l4_uint32_t reg, l4_uint32_t value,
00122              l4_uint32_t width) const
00123 {
00124     return l4vbus_pcidev_cfg_write(bus_cap().cap(), _dev, reg, value, width);
00125 }
00126
00127
00137 int irq_enable(unsigned char *trigger, unsigned char *polarity) const
00138 {
00139     return l4vbus_pcidev_irq_enable(bus_cap().cap(), _dev, trigger, polarity);
00140 }
00141
00142 };
00143
00144 }

```

16.706 vbus_pci-ops.h

```

00001  /*
00002  * (c) 2014 Sarah Hoffmann <sarah.hoffmann@kernkonzept.com>
00003  *
00004  * License: see LICENSE.spdx (in this directory or the directories above)
00005  */
00006 #pragma once
00007
00008 #include <l4/vbus/vbus_interfaces.h>
00009
00010 enum
00011 {
00012     L4vbus_pciroot_cfg_read = L4VBUS_INTERFACE_PCI « L4VBUS_IFACE_SHIFT,

```

```

00013     L4vbus_pciroot_cfg_write,
00014     L4vbus_pciroot_cfg_irq_enable
00015 };
00016
00017 enum
00018 {
00019     L4vbus_pciroot_cfg_read = L4VBUS_INTERFACE_PCIDEV « L4VBUS_IFACE_SHIFT,
00020     L4vbus_pciroot_cfg_write,
00021     L4vbus_pciroot_cfg_irq_enable
00022 };

```

16.707 vbus_pci.h

```

00001 /*
00002  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00008 #pragma once
00009
00010 #include <14/sys/compiler.h>
00011 #include <14/vbus/vbus_types.h>
00012 #include <14/sys/types.h>
00013
00019
00020
00021 L4_BEGIN_DECLS
00022
00029 int L4_CV
00030 l4vbus_pci_cfg_read(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00031                    l4_uint32_t bus, l4_uint32_t devfn,
00032                    l4_uint32_t reg, l4_uint32_t *value, l4_uint32_t width);
00033
00040 int L4_CV
00041 l4vbus_pci_cfg_write(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00042                     l4_uint32_t bus, l4_uint32_t devfn,
00043                     l4_uint32_t reg, l4_uint32_t value, l4_uint32_t width);
00044
00051 int L4_CV
00052 l4vbus_pci_irq_enable(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00053                      l4_uint32_t bus, l4_uint32_t devfn,
00054                      int pin, unsigned char *trigger,
00055                      unsigned char *polarity);
00056
00057
00064 int L4_CV
00065 l4vbus_pciroot_cfg_read(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00066                         l4_uint32_t reg, l4_uint32_t *value, l4_uint32_t width);
00067
00074 int L4_CV
00075 l4vbus_pciroot_cfg_write(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00076                          l4_uint32_t reg, l4_uint32_t value, l4_uint32_t width);
00077
00084 int L4_CV
00085 l4vbus_pciroot_irq_enable(l4_cap_idx_t vbus, l4vbus_device_handle_t handle,
00086                           unsigned char *trigger,
00087                           unsigned char *polarity);
00088
00089
00090
00092 L4_END_DECLS

```

16.708 vbus_pm-ops.h

```

00001 /*
00002  * (c) 2013 Alexander Warg <warg@os.inf.tu-dresden.de>
00003  *      economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007
00008 #pragma once
00009
00010 #include "vbus_interfaces.h"
00011
00012 enum L4vbus_pm_op

```

```

00013 {
00014     L4VBUS_PM_OP_SUSPEND = L4VBUS_INTERFACE_PM << L4VBUS_IFACE_SHIFT,
00015     L4VBUS_PM_OP_RESUME,
00016 };
00017

```

16.709 vbus_pm.h

```

00001 /*
00002  * (c) 2013 Alexander Warg <warg@os.inf.tu-dresden.de>
00003  *     economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00007 #pragma once
00008
00009 #include <l4/sys/compiler.h>
00010 #include <l4/vbus/vbus_types.h>
00011 #include <l4/sys/types.h>
00012
00018
00019 L4_BEGIN_DECLS
00020
00027 int L4_CV
00028 l4vbus_pm_suspend(l4_cap_idx_t vbus, l4vbus_device_handle_t handle);
00029
00036 int L4_CV
00037 l4vbus_pm_resume(l4_cap_idx_t vbus, l4vbus_device_handle_t handle);
00038
00040
00041 L4_END_DECLS

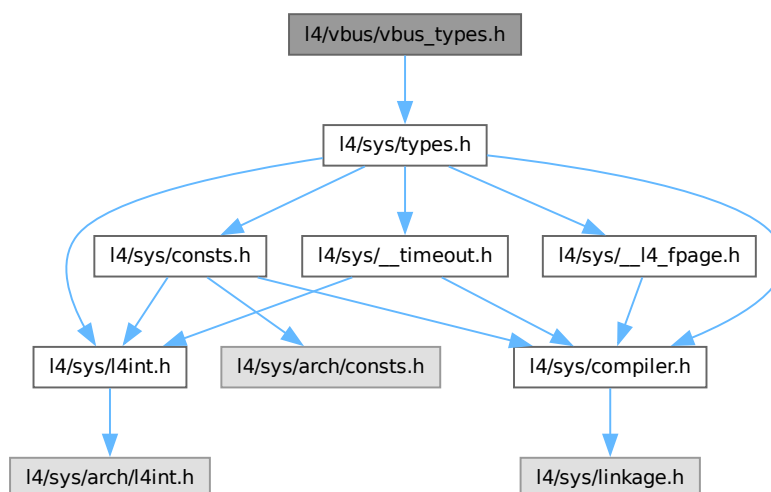
```

16.710 l4/vbus/vbus_types.h File Reference

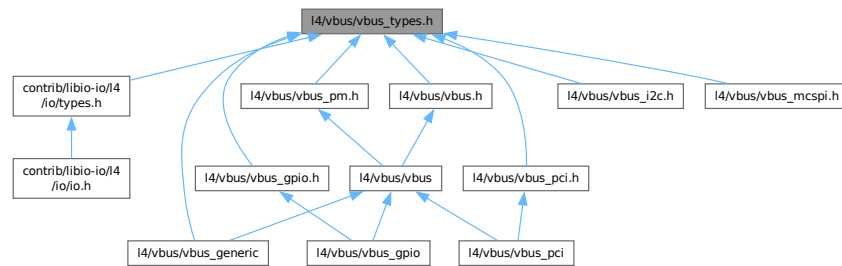
This header file contains descriptions of vbus related data types and constants.

```
#include <l4/sys/types.h>
```

Include dependency graph for vbus_types.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [l4vbus_resource_t](#)
Description of a single vbus resource.
- struct [l4vbus_device_t](#)
Detailed information about a vbus device.

Typedefs

- typedef [l4_mword_t](#) [l4vbus_device_handle_t](#)
Device handle for a device on the vbus.
- typedef [l4_addr_t](#) [l4vbus_paddr_t](#)
Address of resources on the vbus.

Enumerations

- enum [l4vbus_resource_type_t](#) {
[L4VBUS_RESOURCE_INVALID](#) = 0 , [L4VBUS_RESOURCE_IRQ](#) , [L4VBUS_RESOURCE_MEM](#) ,
[L4VBUS_RESOURCE_PORT](#) ,
[L4VBUS_RESOURCE_BUS](#) , [L4VBUS_RESOURCE_GPIO](#) , [L4VBUS_RESOURCE_DMA_DOMAIN](#) ,
[L4VBUS_RESOURCE_MAX](#) }
Description of vbus resource types.
- enum [l4vbus_resource_flags_t](#) {
[L4VBUS_RESOURCE_F_MEM_R](#) = 0x1 , [L4VBUS_RESOURCE_F_MEM_W](#) = 0x2 , [L4VBUS_RESOURCE_F_MEM_PREFE](#)
= 0x10 , [L4VBUS_RESOURCE_F_MEM_CACHEABLE](#) = 0x20 ,
[L4VBUS_RESOURCE_F_MEM_MMIO_READ](#) = 0x2000 , [L4VBUS_RESOURCE_F_MEM_MMIO_WRITE](#)
= 0x4000 }
Description of vbus resource flags.
- enum [l4vbus_device_flags_t](#) { [L4VBUS_DEVICE_F_CHILDREN](#) = 0x10 }
Flags describing device properties, see [l4vbus_device_t](#).

16.710.1 Detailed Description

This header file contains descriptions of vbus related data types and constants.

Definition in file [vbus_types.h](#).

16.710.2 Enumeration Type Documentation

16.710.2.1 l4vbus_device_flags_t

enum [l4vbus_device_flags_t](#)

Flags describing device properties, see [l4vbus_device_t](#).

Enumerator

| | |
|--------------------------|---------------------------|
| L4VBUS_DEVICE_F_CHILDREN | Device has child devices. |
|--------------------------|---------------------------|

Definition at line 92 of file [vbus_types.h](#).

16.710.2.2 l4vbus_resource_flags_t

enum [l4vbus_resource_flags_t](#)

Description of vbus resource flags.

Enumerator

| | |
|------------------------------------|---|
| L4VBUS_RESOURCE_F_MEM_R | Memory resource is readable. |
| L4VBUS_RESOURCE_F_MEM_W | Memory resource is writeable. |
| L4VBUS_RESOURCE_F_MEM_PREFETCHABLE | Memory resource is prefetchable. Clients may map it buffered or non-cached. |
| L4VBUS_RESOURCE_F_MEM_CACHEABLE | Memory resource is cacheable. This implies that the memory resource is prefetchable. If not set, clients must not map it cached. If the resource is neither cacheable nor prefetchable, clients must map it non-cached! |
| L4VBUS_RESOURCE_F_MEM_MMIO_READ | Reading needs to be performed using the MMIO space protocol. |
| L4VBUS_RESOURCE_F_MEM_MMIO_WRITE | Writing needs to be performed using the MMIO space protocol. |

Definition at line 51 of file [vbus_types.h](#).

16.710.2.3 l4vbus_resource_type_t

enum [l4vbus_resource_type_t](#)

Description of vbus resource types.

Enumerator

| | |
|----------------------------|-------------------------------|
| L4VBUS_RESOURCE_INVALID | Invalid type. |
| L4VBUS_RESOURCE_IRQ | Interrupt resource. |
| L4VBUS_RESOURCE_MEM | I/O memory resource. |
| L4VBUS_RESOURCE_PORT | I/O port resource (x86 only). |
| L4VBUS_RESOURCE_BUS | Bus resource. |
| L4VBUS_RESOURCE_GPIO | Gpio resource. |
| L4VBUS_RESOURCE_DMA_DOMAIN | DMA domain. |
| L4VBUS_RESOURCE_MAX | Maximum resource id. |

Definition at line 39 of file [vbus_types.h](#).

16.711 vbus_types.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00013 #pragma once
00014
00015 #include <l4/sys/types.h>
00016
00018 typedef l4_mword_t l4vbus_device_handle_t;
00020 typedef l4_addr_t l4vbus_paddr_t;
00021
00023 typedef struct {
00025     l4_uint16_t    type;
00027     l4_uint16_t    flags;
00029     l4vbus_paddr_t start;
00031     l4vbus_paddr_t end;
00033     l4vbus_device_handle_t provider;
00035     l4_uint32_t id;
00036 } l4vbus_resource_t;
00037
00039 enum l4vbus_resource_type_t {
00040     L4VBUS_RESOURCE_INVALID = 0,
00041     L4VBUS_RESOURCE_IRQ,
00042     L4VBUS_RESOURCE_MEM,
00043     L4VBUS_RESOURCE_PORT,
00044     L4VBUS_RESOURCE_BUS,
00045     L4VBUS_RESOURCE_GPIO,
00046     L4VBUS_RESOURCE_DMA_DOMAIN,
00047     L4VBUS_RESOURCE_MAX,
00048 };
00049
00051 enum l4vbus_resource_flags_t {
00053     L4VBUS_RESOURCE_F_MEM_R = 0x1,
00055     L4VBUS_RESOURCE_F_MEM_W = 0x2,
00060     L4VBUS_RESOURCE_F_MEM_PREFETCHABLE = 0x10,
00067     L4VBUS_RESOURCE_F_MEM_CACHEABLE = 0x20,
00069     L4VBUS_RESOURCE_F_MEM_MMIO_READ = 0x2000,
00071     L4VBUS_RESOURCE_F_MEM_MMIO_WRITE = 0x4000,
00072 };
00073
00074 enum l4vbus_consts_t {
00075     L4VBUS_DEV_NAME_LEN = 64,
00076     L4VBUS_MAX_DEPTH = 100,
00077 };
00078
00080 typedef struct {
00082     l4_uint32_t    type;
00084     char           name[L4VBUS_DEV_NAME_LEN];
00086     unsigned       num_resources;
00088     unsigned       flags;
00089 } l4vbus_device_t;
00090
00092 enum l4vbus_device_flags_t {
00093     L4VBUS_DEVICE_F_CHILDREN = 0x10,
00094 };

```

16.712 vdevice-ops.h

```

00001 /*
00002  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      Alexander Warg <warg@os.inf.tu-dresden.de>,
00004  *      Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00005  *      economic rights: Technische Universität Dresden (Germany)
00006  *
00007  * License: see LICENSE.spdx (in this directory or the directories above)
00008  */
00009 #pragma once
00010
00011 #include "vbus_interfaces.h"
00012
00013 enum L4vbus_vdevice_op
00014 {
00015     L4vbus_vdevice_hid = L4VBUS_INTERFACE_GENERIC « L4VBUS_IFACE_SHIFT,
00016     L4vbus_vdevice_adr,
00017     L4vbus_vdevice_get_by_hid,
00018     L4vbus_vdevice_get_next,
00019     L4vbus_vdevice_get_resource,
00020     L4vbus_vdevice_get_hid,
00021     L4vbus_vdevice_is_compatible,
00022     L4vbus_vdevice_get,
00023 };
00024
00025 enum {
00026     L4vbus_vbus_request_resource = L4VBUS_INTERFACE_BUS « L4VBUS_IFACE_SHIFT,
00027     L4vbus_vbus_release_resource,
00028     L4vbus_vbus_assign_dma_domain,
00029 };
00030
00031 enum
00032 {
00033     L4vbus_vicu_get_cap = L4VBUS_INTERFACE_ICU « L4VBUS_IFACE_SHIFT
00034 };
00035

```

16.713 l4/vcpu/vcpu File Reference

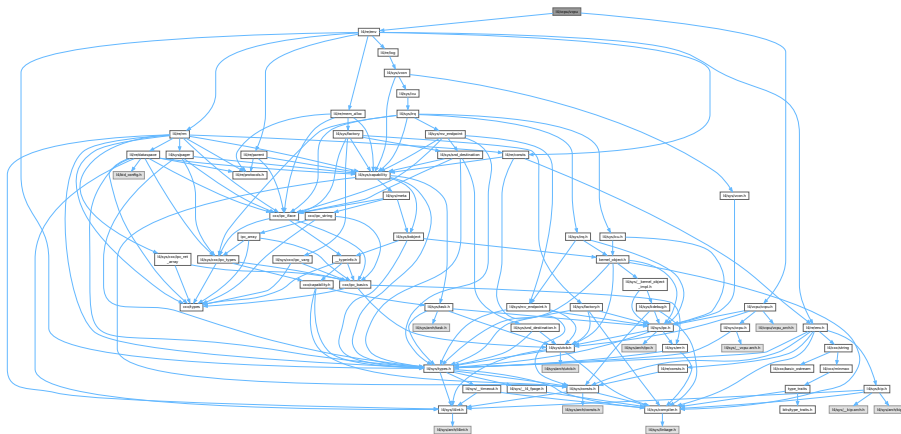
vCPU support library (C++ interface).

```

#include <l4/re/env>
#include <l4/vcpu/vcpu.h>

```

Include dependency graph for vcpu:



Data Structures

- class [L4vcpu::State](#)
C++ implementation of state word in the vCPU area.
- class [L4vcpu::Vcpu](#)
C++ implementation of the vCPU save state area.

16.713.1 Detailed Description

vCPU support library (C++ interface).

Definition in file [vcpu](#).

16.714 vcpu

[Go to the documentation of this file.](#)

```

00001 // vi:se ft=cpp:
00002 /*
00003  * (c) 2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00004  *     economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00012
00013 #pragma once
00014
00015 #include <l4/re/env>
00016 #include <l4/vcpu/vcpu.h>
00017
00018 namespace L4vcpu {
00019
00024 class State
00025 {
00026 public:
00027     State() {}
00028
00034     explicit State(unsigned v) : _s(v) {}
00035
00041     void add(unsigned bits) throw() { _s |= bits; }
00042
00048     void clear(unsigned bits) throw() { _s &= ~bits; }
00049
00055     void set(unsigned v) throw() { _s = v; }
00056
00057 private:
00058     __typeof__(((l4_vcpu_state_t *)0)->state) _s;
00059 };
00060
00065 class Vcpu : private l4_vcpu_state_t
00066 {
00067 public:
00071     void irq_disable() throw()
00072     { l4vcpu_irq_disable(this); }
00073
00078     unsigned irq_disable_save() throw()
00079     { return l4vcpu_irq_disable_save(this); }
00080
00081     l4_vcpu_state_t *s() { return this; }
00082     l4_vcpu_state_t const *s() const { return this; }
00083
00088     State *state() throw()
00089     {
00090         static_assert(sizeof(State) == sizeof(l4_vcpu_state_t::state),
00091             "size mismatch");
00092         return reinterpret_cast<State *>(&(l4_vcpu_state_t::state));
00093     }
00094
00099     State state() const throw()
00100     { return static_cast<State>(l4_vcpu_state_t::state); }
00101
00106     State *saved_state() throw()
00107     {
00108         static_assert(sizeof(State) == sizeof(l4_vcpu_state_t::saved_state),
00109             "size mismatch");
00110         return reinterpret_cast<State *>(&(l4_vcpu_state_t::saved_state));
00111     }
00116     State saved_state() const throw()
00117     { return static_cast<State>(l4_vcpu_state_t::saved_state); }
00118
00122     l4_uint16_t sticky_flags() const throw()
00123     { return l4_vcpu_state_t::sticky_flags; }
00124
00135     void irq_enable(l4_utcb_t *utcb, l4vcpu_event_hndl_t do_event_work_cb,
00136         l4vcpu_setup_ipc_t setup_ipc) throw()
00137     { l4vcpu_irq_enable(this, utcb, do_event_work_cb, setup_ipc); }

```

```

00138
00150 void irq_restore(unsigned s, l4_utcb_t *utcb,
00151                  l4vcpu_event_hndl_t do_event_work_cb,
00152                  l4vcpu_setup_ipc_t setup_ipc) throw()
00153 { l4vcpu_irq_restore(this, s, utcb, do_event_work_cb, setup_ipc); }
00154
00166 void wait_for_event(l4_utcb_t *utcb, l4vcpu_event_hndl_t do_event_work_cb,
00167                    l4vcpu_setup_ipc_t setup_ipc) throw()
00168 { l4vcpu_wait_for_event(this, utcb, do_event_work_cb, setup_ipc); }
00169
00174 void task(L4::Cap<L4::Task> const task = L4::Cap<L4::Task>::Invalid) throw()
00175 { user_task = task.cap(); }
00176
00181 int is_page_fault_entry() const
00182 { return l4vcpu_is_page_fault_entry(this); }
00183
00188 int is_irq_entry() const
00189 { return l4vcpu_is_irq_entry(this); }
00190
00195 l4_vcpu_regs_t *r() throw()
00196 { return &(l4_vcpu_state_t::r); }
00197
00202 l4_vcpu_regs_t const *r() const throw()
00203 { return &(l4_vcpu_state_t::r); }
00204
00209 l4_vcpu_ipc_regs_t *i() throw()
00210 { return &(l4_vcpu_state_t::i); }
00211
00216 l4_vcpu_ipc_regs_t const *i() const throw()
00217 { return &(l4_vcpu_state_t::i); }
00218
00225 void entry_sp(l4_umword_t sp)
00226 { l4_vcpu_state_t::entry_sp = sp; }
00227
00232 void entry_ip(l4_umword_t ip)
00233 { l4_vcpu_state_t::entry_ip = ip; }
00234
00246 L4_CV static int
00247 ext_alloc(Vcpu **vcpu,
00248           l4_addr_t *ext_state,
00249           L4::Cap<L4::Task> task = L4Re::Env::env()->task(),
00250           L4::Cap<L4Re::Rm> rm = L4Re::Env::env()->rm()) throw();
00251
00259 static inline Vcpu *cast(void *x) throw()
00260 { return reinterpret_cast<Vcpu *>(x); }
00261
00269 static inline Vcpu *cast(l4_addr_t x) throw()
00270 { return reinterpret_cast<Vcpu *>(x); }
00271
00275 void print_state(const char *prefix = "") const throw()
00276 { l4vcpu_print_state(this, prefix); }
00277 };
00278
00279
00280 }

```

16.715 l4/sys/vcpu.h File Reference

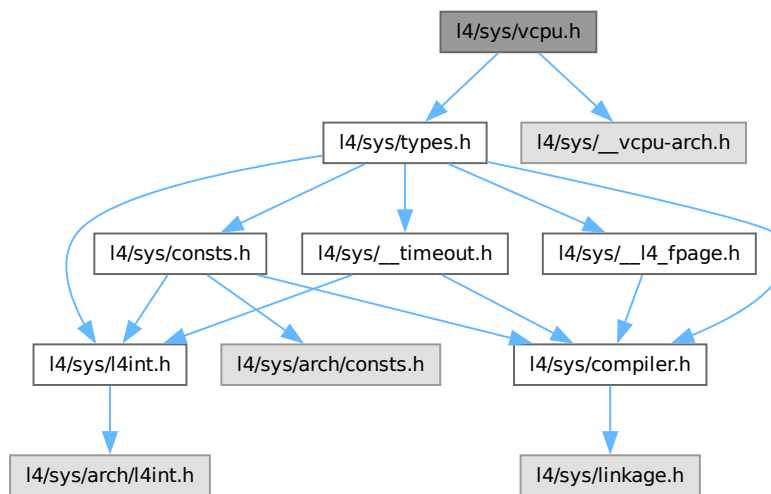
vCPU API

```

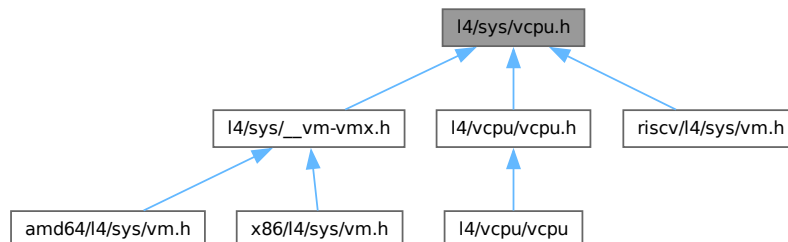
#include <l4/sys/types.h>
#include <l4/sys/__vcpu-arch.h>

```

Include dependency graph for `vcpu.h`:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct `l4_vcpu_state_t`
State of a vCPU.

Typedefs

- typedef struct `l4_vcpu_state_t` `l4_vcpu_state_t`
State of a vCPU.

Enumerations

- enum [L4_vcpu_state_flags](#) {
[L4_VCPU_F_IRQ](#) = 0x01 , [L4_VCPU_F_PAGE_FAULTS](#) = 0x02 , [L4_VCPU_F_EXCEPTIONS](#) = 0x04 ,
[L4_VCPU_F_USER_MODE](#) = 0x20 ,
[L4_VCPU_F_FPU_ENABLED](#) = 0x80 }
State flags of a vCPU.
- enum [L4_vcpu_sticky_flags](#) { [L4_VCPU_SF_IRQ_PENDING](#) = 0x01 }
Sticky flags of a vCPU.

Functions

- int [l4_vcpu_check_version](#) ([l4_vcpu_state_t](#) const *vcpu) [L4_NOTHROW](#)
Check if a vCPU state has the right version.

16.715.1 Detailed Description

vCPU API

Definition in file [vcpu.h](#).

16.715.2 Function Documentation

16.715.2.1 [l4_vcpu_check_version\(\)](#)

```
int l4_vcpu_check_version (
    l4\_vcpu\_state\_t const * vcpu) [inline]
```

Check if a vCPU state has the right version.

Parameters

| | |
|-------------|---|
| <i>vcpu</i> | A pointer to an initialized vCPU state. |
|-------------|---|

Return values

| | |
|---|--|
| 1 | If the vCPU state has a matching version ID for the current vCPU user-level structures. |
| 0 | If the vCPU state has a different (incompatible) version ID than the current vCPU user-level structures. |

Definition at line 191 of file [vcpu.h](#).

References [L4_NOTHROW](#), and [L4_VCPU_STATE_VERSION](#).

16.716 vcpu.h

[Go to the documentation of this file.](#)

```

00001  /*
00002  * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00003  *      Alexander Warg <warg@os.inf.tu-dresden.de>
00004  *      economic rights: Technische Universität Dresden (Germany)
00005  *
00006  * License: see LICENSE.spdx (in this directory or the directories above)
00007  */
00012  #pragma once
00013
00014  #include <l4/sys/types.h>
00015  #include <l4/sys/__vcpu-arch.h>
00016
00070
00075  typedef struct l4_vcpu_state_t
00076  {
00077      l4_umword_t      version;
00080      l4_umword_t      user_data[7];
00081      l4_vcpu_regs_t   r;
00082      l4_vcpu_ipc_regs_t i;
00083
00084      l4_uint16_t      state;
00085      l4_uint16_t      saved_state;
00086      l4_uint16_t      sticky_flags;
00087      l4_uint16_t      _reserved;
00088
00089      l4_cap_idx_t      user_task;
00090
00091      l4_umword_t      entry_sp;
00092      l4_umword_t      entry_ip;
00093      l4_umword_t      reserved_sp;
00094      l4_vcpu_arch_state_t arch_state;
00095  } l4_vcpu_state_t;
00096
00101  enum L4_vcpu_state_flags
00102  {
00114      L4_VCPU_F_IRQ          = 0x01,
00115
00129      L4_VCPU_F_PAGE_FAULTS = 0x02,
00130
00142      L4_VCPU_F_EXCEPTIONS  = 0x04,
00143
00152      L4_VCPU_F_USER_MODE   = 0x20,
00153
00160      L4_VCPU_F_FPU_ENABLED = 0x80,
00161  };
00162
00167  enum L4_vcpu_sticky_flags
00168  {
00171      L4_VCPU_SF_IRQ_PENDING = 0x01,
00172  };
00173
00185  L4_INLINE int
00186  l4_vcpu_check_version(l4_vcpu_state_t const *vcpu) L4_NOTHROW;
00187
00188  /* IMPLEMENTATION: -----*/
00189
00190  L4_INLINE int
00191  l4_vcpu_check_version(l4_vcpu_state_t const *vcpu) L4_NOTHROW
00192  {
00193      return vcpu->version == L4_VCPU_STATE_VERSION;
00194  }

```

16.717 l4/vcpu/vcpu.h File Reference

vCPU support library (C interface).

```

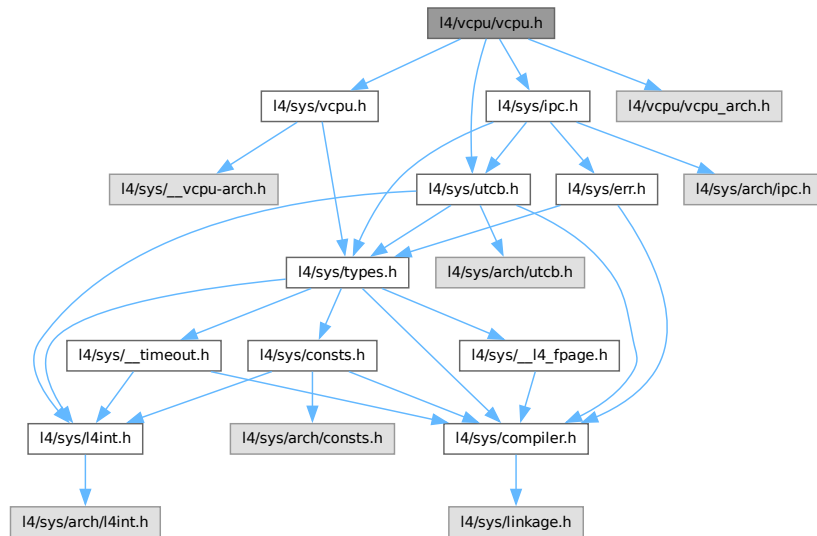
#include <l4/sys/vcpu.h>
#include <l4/sys/utcb.h>
#include <l4/sys/ipc.h>

```

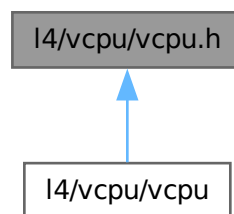


```
#include <l4/vcpu/vcpu_arch.h>
```

Include dependency graph for vcpu.h:



This graph shows which files directly or indirectly include this file:



Functions

- void [l4vcpu_irq_disable](#) ([l4_vcpu_state_t](#) *vcpu) [L4_NOTHROW](#)
Disable a vCPU for event delivery.
- unsigned [l4vcpu_irq_disable_save](#) ([l4_vcpu_state_t](#) *vcpu) [L4_NOTHROW](#)
Disable a vCPU for event delivery and return previous state.
- void [l4vcpu_irq_enable](#) ([l4_vcpu_state_t](#) *vcpu, [l4_utcb_t](#) *utcb, [l4vcpu_event_hndl_t](#) do_event_work_cb, [l4vcpu_setup_ipc_t](#) setup_ipc) [L4_NOTHROW](#)
Enable a vCPU for event delivery.
- void [l4vcpu_irq_restore](#) ([l4_vcpu_state_t](#) *vcpu, unsigned s, [l4_utcb_t](#) *utcb, [l4vcpu_event_hndl_t](#) do_event_work_cb, [l4vcpu_setup_ipc_t](#) setup_ipc) [L4_NOTHROW](#)
Restore a previously saved IRQ/event state.

- void `l4vcpu_wait_for_event` (`l4_vcpu_state_t` *vcpu, `l4_utcb_t` *utcb, `l4vcpu_event_hndl_t` do_event_work_cb, `l4vcpu_setup_ipc_t` setup_ipc) `L4_NOTHROW`
Wait for event.
- void `l4vcpu_print_state` (const `l4_vcpu_state_t` *vcpu, const char *prefix) `L4_NOTHROW`
Print the state of a vCPU.
- int `l4vcpu_is_irq_entry` (`l4_vcpu_state_t` const *vcpu) `L4_NOTHROW`
Return whether the entry reason was an IRQ/IPC message.
- int `l4vcpu_is_page_fault_entry` (`l4_vcpu_state_t` const *vcpu) `L4_NOTHROW`
Return whether the entry reason was a page fault.
- int `l4vcpu_ext_alloc` (`l4_vcpu_state_t` **vcpu, `l4_addr_t` *ext_state, `l4_cap_idx_t` task, `l4_cap_idx_t` regmgr) `L4_NOTHROW`
Allocate state area for an extended vCPU.

16.717.1 Detailed Description

vCPU support library (C interface).

Definition in file [vcpu.h](#).

16.718 vcpu.h

[Go to the documentation of this file.](#)

```

00001 /*
00002  * (c) 2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00003  *      economic rights: Technische Universität Dresden (Germany)
00004  *
00005  * License: see LICENSE.spdx (in this directory or the directories above)
00006  */
00011 #pragma once
00012
00013 #include <l4/sys/vcpu.h>
00014 #include <l4/sys/utcb.h>
00015
00016 __BEGIN_DECLS
00017
00026
00032
00033 typedef void (*l4vcpu_event_hndl_t) (l4_vcpu_state_t *vcpu);
00034 typedef void (*l4vcpu_setup_ipc_t) (l4_utcb_t *utcb);
00035
00042 L4_CV L4_INLINE
00043 void
00044 l4vcpu_irq_disable(l4_vcpu_state_t *vcpu) L4_NOTHROW;
00045
00054 L4_CV L4_INLINE
00055 unsigned
00056 l4vcpu_irq_disable_save(l4_vcpu_state_t *vcpu) L4_NOTHROW;
00057
00070 L4_CV L4_INLINE
00071 void
00072 l4vcpu_irq_enable(l4_vcpu_state_t *vcpu, l4_utcb_t *utcb,
00073                  l4vcpu_event_hndl_t do_event_work_cb,
00074                  l4vcpu_setup_ipc_t setup_ipc) L4_NOTHROW;
00075
00090 L4_CV L4_INLINE
00091 void
00092 l4vcpu_irq_restore(l4_vcpu_state_t *vcpu, unsigned s,
00093                   l4_utcb_t *utcb,
00094                   l4vcpu_event_hndl_t do_event_work_cb,
00095                   l4vcpu_setup_ipc_t setup_ipc) L4_NOTHROW;
00096
00110 L4_CV L4_INLINE
00111 void
00112 l4vcpu_wait(l4_vcpu_state_t *vcpu, l4_utcb_t *utcb,
00113            l4_timeout_t to,
00114            l4vcpu_event_hndl_t do_event_work_cb,
00115            l4vcpu_setup_ipc_t setup_ipc) L4_NOTHROW;

```

```

00116
00130 L4_CV L4_INLINE
00131 void
00132 l4vcpu_wait_for_event(l4_vcpu_state_t *vcpu, l4_utcb_t *utcb,
00133                      l4vcpu_event_hndl_t do_event_work_cb,
00134                      l4vcpu_setup_ipc_t setup_ipc) L4_NOTHROW;
00135
00136
00144 L4_CV void
00145 l4vcpu_print_state(const l4_vcpu_state_t *vcpu, const char *prefix) L4_NOTHROW;
00146
00150 L4_CV void
00151 l4vcpu_print_state_arch(const l4_vcpu_state_t *vcpu, const char *prefix) L4_NOTHROW;
00152
00153
00162 L4_CV L4_INLINE
00163 int
00164 l4vcpu_is_irq_entry(l4_vcpu_state_t const *vcpu) L4_NOTHROW;
00165
00174 L4_CV L4_INLINE
00175 int
00176 l4vcpu_is_page_fault_entry(l4_vcpu_state_t const *vcpu) L4_NOTHROW;
00177
00189 L4_CV int
00190 l4vcpu_ext_alloc(l4_vcpu_state_t **vcpu, l4_addr_t *ext_state,
00191                 l4_cap_idx_t task, l4_cap_idx_t regmgr) L4_NOTHROW;
00192
00193 /* ===== */
00194 /* Implementations */
00195
00196 #include <l4/sys/ipc.h>
00197 #include <l4/vcpu/vcpu_arch.h>
00198
00199 L4_CV L4_INLINE
00200 void
00201 l4vcpu_irq_disable(l4_vcpu_state_t *vcpu) L4_NOTHROW
00202 {
00203     vcpu->state &= ~L4_VCPU_F_IRQ;
00204     l4_barrier();
00205 }
00206
00207 L4_CV L4_INLINE
00208 unsigned
00209 l4vcpu_irq_disable_save(l4_vcpu_state_t *vcpu) L4_NOTHROW
00210 {
00211     unsigned s = vcpu->state;
00212     l4vcpu_irq_disable(vcpu);
00213     return s;
00214 }
00215
00216 L4_CV L4_INLINE
00217 void
00218 l4vcpu_wait(l4_vcpu_state_t *vcpu, l4_utcb_t *utcb,
00219             l4_timeout_t to,
00220             l4vcpu_event_hndl_t do_event_work_cb,
00221             l4vcpu_setup_ipc_t setup_ipc) L4_NOTHROW
00222 {
00223     l4vcpu_irq_disable(vcpu);
00224     setup_ipc(utcb);
00225     vcpu->i.tag = l4_ipc_wait(utcb, &vcpu->i.label, to);
00226     if (L4_LIKELY(!l4_msgtag_has_error(vcpu->i.tag)))
00227         do_event_work_cb(vcpu);
00228 }
00229
00230 L4_CV L4_INLINE
00231 void
00232 l4vcpu_irq_enable(l4_vcpu_state_t *vcpu, l4_utcb_t *utcb,
00233                  l4vcpu_event_hndl_t do_event_work_cb,
00234                  l4vcpu_setup_ipc_t setup_ipc) L4_NOTHROW
00235 {
00236     if (!(vcpu->state & L4_VCPU_F_IRQ))
00237     {
00238         setup_ipc(utcb);
00239         l4_barrier();
00240     }
00241
00242     while (1)
00243     {
00244         vcpu->state |= L4_VCPU_F_IRQ;
00245         l4_barrier();
00246
00247         if (L4_LIKELY(!(vcpu->sticky_flags & L4_VCPU_SF_IRQ_PENDING)))
00248             break;
00249
00250         l4vcpu_wait(vcpu, utcb, L4_IPC_BOTH_TIMEOUT_0,
00251                     do_event_work_cb, setup_ipc);
00252     }

```

```
00253 }
00254
00255 L4_CV L4_INLINE
00256 void
00257 l4vcpu_irq_restore(l4_vcpu_state_t *vcpu, unsigned s,
00258                  l4_utcb_t *utcb,
00259                  l4vcpu_event_hndl_t do_event_work_cb,
00260                  l4vcpu_setup_ipc_t setup_ipc) L4_NOTHROW
00261 {
00262     if (s & L4_VCPU_F_IRQ)
00263         l4vcpu_irq_enable(vcpu, utcb, do_event_work_cb, setup_ipc);
00264     else if (vcpu->state & L4_VCPU_F_IRQ)
00265         l4vcpu_irq_disable(vcpu);
00266 }
00267
00268 L4_CV L4_INLINE
00269 void
00270 l4vcpu_wait_for_event(l4_vcpu_state_t *vcpu, l4_utcb_t *utcb,
00271                     l4vcpu_event_hndl_t do_event_work_cb,
00272                     l4vcpu_setup_ipc_t setup_ipc) L4_NOTHROW
00273 {
00274     l4vcpu_wait(vcpu, utcb, L4_IPC_NEVER, do_event_work_cb, setup_ipc);
00275 }
00276
00277 __END_DECLS
```

Chapter 17

Examples

17.1 hello/server/src/main.c

This is the famous "Hello World!" program.

This is the famous "Hello World!" program.

```
/* SPDX-License-Identifier: MIT */

#include <stdio.h>
#include <unistd.h>

int main(void)
{
    for (;;)
    {
        puts("Hello World!");
        sleep(1);
    }
}
```

17.2 examples/sys/ipc/ipc_example.c

This example shows how two threads can exchange data using the [L4](#) IPC mechanism.

This example shows how two threads can exchange data using the [L4](#) IPC mechanism. One thread is sending an integer to the other thread which is returning the square of the integer. Both values are printed.

```
/* SPDX-License-Identifier: MIT */
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
 */
#include <pthread-l4.h>
#include <unistd.h>
#include <stdio.h>

#include <l4/sys/ipc.h>

static pthread_t t2;

/* Thread1 is the initiator thread, i.e. it initiates the IPC calls. In
 * other words, it takes the client role. It uses L4 IPC mechanisms to send
 * an integer value to thread2 and received a calculation result back. */
static void *thread1_fn(void *arg)
{
    l4_msgtag_t tag;
    int ipc_error;
    unsigned long value = 1;
    (void)arg;

    while (1)
```

```

{
    printf("Sending: %ld\n", value);

    /* Store the value which we want to have squared in the first message
     * register of our UTCB. */
    l4_utcb_mr()->mr[0] = value;

    /* To an L4 IPC call, i.e. send a message to thread2 and wait for a
     * reply from thread2. The '1' in the msgtag denotes that we want to
     * transfer one word of our message registers (i.e. MR0). No timeout. */
    tag = l4_ipc_call(pthread_l4_cap(t2), l4_utcb(),
                     l4_msgtag(0, 1, 0, 0), L4_IPC_NEVER);
    /* Check for IPC error, if yes, print out the IPC error code, if not,
     * print the received result. */
    ipc_error = l4_ipc_error(tag, l4_utcb());
    if (ipc_error)
        fprintf(stderr, "thread1: IPC error: %x\n", ipc_error);
    else
        printf("Received: %ld\n", l4_utcb_mr()->mr[0]);

    /* Wait some time and increment our value. */
    sleep(1);
    value++;
}
return NULL;
}

/* Thread2 is in the server role, i.e. it waits for requests from others and
 * sends back the calculation results. */
static void *thread2_fn(void *arg)
{
    l4_msgtag_t tag;
    l4_umword_t label;
    int ipc_error;
    (void)arg;

    /* Wait for requests from any thread. No timeout, i.e. wait forever. */
    tag = l4_ipc_wait(l4_utcb(), &label, L4_IPC_NEVER);
    while (1)
    {
        /* Check if we had any IPC failure, if yes, print the error code
         * and just wait again. */
        ipc_error = l4_ipc_error(tag, l4_utcb());
        if (ipc_error)
        {
            fprintf(stderr, "thread2: IPC error: %x\n", ipc_error);
            tag = l4_ipc_wait(l4_utcb(), &label, L4_IPC_NEVER);
            continue;
        }

        /* So, the IPC was ok, now take the value out of message register 0
         * of the UTCB and store the square of it back to it. */
        l4_utcb_mr()->mr[0] = l4_utcb_mr()->mr[0] * l4_utcb_mr()->mr[0];

        /* Send the reply and wait again for new messages.
         * The '1' in the msgtag indicated that we want to transfer 1 word in
         * the message registers (i.e. MR0) */
        tag = l4_ipc_reply_and_wait(l4_utcb(), l4_msgtag(0, 1, 0, 0),
                                   &label, L4_IPC_NEVER);
    }
    return NULL;
}

int main(void)
{
    // We will have two threads, one is already running the main function, the
    // other (thread2) will be created using pthread_create.

    if (pthread_create(&t2, NULL, thread2_fn, NULL))
    {
        fprintf(stderr, "Thread creation failed\n");
        return 1;
    }

    // Just run thread1 in the main thread
    thread1_fn(NULL);
    return 0;
}

```

17.3 examples/sys/ipc/ipc.cfg

Sample configuration file for the IPC example.

Sample configuration file for the IPC example.

```
# vim:se ft=lua:

local L4 = require("L4");

L4.default_loader:start({}, "rom/ex_ipc1");
```

17.4 examples/sys/start-with-exc/main.c

This example shows how to start a newly created thread with a defined set of CPU registers.

This example shows how to start a newly created thread with a defined set of CPU registers.

```
/* SPDX-License-Identifier: MIT */
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *               Alexander Warg <warg@os.inf.tu-dresden.de>,
 *               Björn Döbel <doebel@os.inf.tu-dresden.de>,
 *               Frank Mehnert <fm3@os.inf.tu-dresden.de>
 */
/*
 * Start a thread with an exception reply. This example does only work on
 * the x86-32 and ARM architectures.
 */

#include <l4/sys/thread.h>
#include <l4/sys/factory.h>
#include <l4/sys/ipc.h>
#include <l4/sys/scheduler.h>
#include <l4/sys/utcb.h>
#include <l4/util/util.h>
#include <l4/re/env.h>
#include <l4/re/c/util/cap_alloc.h>

#include <stdlib.h>
#include <stdio.h>

/* Stack for the thread to be created. 8kB are enough. */
static char thread_stack[8 « 10];

/* The thread to be created. For illustration it will print out its
 * register set.
 */
static void L4_STICKY(thread_func(l4_umword_t *d))
{
    while (1)
    {
        printf("hey, I'm a thread\n");
        printf("got register values: %ld %ld %ld %ld %ld %ld %ld\n",
            d[7], d[6], d[5], d[4], d[2], d[1], d[0]);
        l4_sleep(800);
    }
}

/* Startup trick for this example. Put all the CPU registers on the stack so
 * that the C function above can get it on the stack. */
asm(
    ".global thread      \n"
    "thread:             \n"
    #ifdef ARCH_x86
    " pusha              \n"
    " push %esp          \n"
    " call thread_func   \n"
    #endif
    #ifdef ARCH_arm
    "    push {r0-r7}      \n"
    "    mov r0, sp        \n"
    "    bl thread_func    \n"
    #endif
    #ifdef ARCH_arm64
    "    stp x0, x1, [sp, #0]! \n"
    "    stp x2, x3, [sp, #0]! \n"
    "    stp x4, x5, [sp, #0]! \n"
    "    stp x6, x7, [sp, #0]! \n"
    "    mov x0, sp        \n"
    "    bl thread_func    \n"
    #endif
);
extern void thread(void);

/* Our main function */
```

```

int main(void)
{
    /* Get a capability slot for our new thread. */
    l4_cap_idx_t t1 = l4re_util_cap_alloc();
    l4_utcb_t *u = l4_utcb();
    l4_exc_regs_t *e = l4_utcb_exc_u(u);
    l4_msgtag_t tag;
    int err;

    printf("Example showing how to start a thread with an exception.\n");
    /* We do not want to implement a pager here, take the shortcut. */
    printf("Make sure to start this program with ldr-flags=eager_map\n");

    if (l4_is_invalid_cap(t1))
        return 1;

    /* Create the thread using our default factory */
    tag = l4_factory_create_thread(l4re_env()->factory, t1);
    if (l4_error(tag))
        return 1;

    /* Setup the thread by setting the pager and task. */
    l4_thread_control_start();
    l4_thread_control_pager(l4re_env()->main_thread);
    l4_thread_control_exc_handler(l4re_env()->main_thread);
    l4_thread_control_bind((l4_utcb_t *)l4re_env()->first_free_utcb,
                          L4RE_THIS_TASK_CAP);
    tag = l4_thread_control_commit(t1);
    if (l4_error(tag))
        return 2;

    /* Start the thread by finally setting instruction and stack pointer */
    tag = l4_thread_ex_regs(t1,
                           (l4_umword_t)thread,
                           (l4_umword_t)thread_stack + sizeof(thread_stack),
                           L4_THREAD_EX_REGS_TRIGGER_EXCEPTION);

    if (l4_error(tag))
        return 3;

    l4_sched_param_t sp = l4_sched_param(1, 0);
    tag = l4_scheduler_run_thread(l4re_env()->scheduler, t1, &sp);
    if (l4_error(tag))
        return 4;

    /* Receive initial exception from just started thread */
    tag = l4_ipc_receive(t1, u, L4_IPC_NEVER);
    if ((err = l4_ipc_error(tag, u)))
    {
        printf("Umm, ipc error: %x\n", err);
        return 1;
    }
    /* We expect an exception IPC */
    if (!l4_msgtag_is_exception(tag))
    {
        printf("PF?: %lx %lx (not prepared to handle this) %ld\n",
              l4_utcb_mr_u(u)->mr[0], l4_utcb_mr_u(u)->mr[1], l4_msgtag_label(tag));
        return 1;
    }

    /* Fill out the complete register set of the new thread */
    e->sp = (l4_umword_t)(thread_stack + sizeof(thread_stack));
#ifdef ARCH_x86
    e->ip = (l4_umword_t)thread;
    e->edi = 0;
    e->esi = 1;
    e->ebp = 2;
    e->ebx = 4;
    e->edx = 5;
    e->ecx = 6;
    e->eax = 7;
#endif
#ifdef ARCH_arm
    e->pc = (l4_umword_t)thread;
    e->r[0] = 0;
    e->r[1] = 1;
    e->r[2] = 2;
    e->r[3] = 3;
    e->r[4] = 4;
    e->r[5] = 5;
    e->r[6] = 6;
    e->r[7] = 7;
#endif
#ifdef ARCH_arm64
    e->pc = (l4_umword_t)thread;
    e->r[0] = 0;
    e->r[1] = 1;

```



```

e->r[2] = 2;
e->r[3] = 3;
e->r[4] = 4;
e->r[5] = 5;
e->r[6] = 6;
e->r[7] = 7;
#endif
/* Send a complete exception */
tag = l4_msgtag(0, L4_UTCB_EXCEPTION_REGS_SIZE, 0, 0);

/* Send reply and start the thread with the defined CPU register set */
tag = l4_ipc_send(tl, u, tag, L4_IPC_NEVER);
if ((err = l4_ipc_error(tag, u)))
    printf("Error sending IPC: %x\n", err);

/* Idle around */
while (1)
    l4_sleep(10000);

return 0;
}

```

17.5 examples/sys/singlestep/main.c

This example shows how a thread can be single stepped on the x86 architecture.

This example shows how a thread can be single stepped on the x86 architecture.

```

/* SPDX-License-Identifier: MIT */
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *             Alexander Warg <warg@os.inf.tu-dresden.de>,
 *             Björn Döbel <doebel@os.inf.tu-dresden.de>
 */
/*
 * Single stepping example for the x86-32 architecture.
 */
#include <l4/sys/ipc.h>
#include <l4/sys/factory.h>
#include <l4/sys/thread.h>
#include <l4/sys/scheduler.h>
#include <l4/sys/utcb.h>
#include <l4/sys/kdebug.h>

#include <l4/util/util.h>
#include <l4/re/env.h>
#include <l4/re/c/util/cap_alloc.h>

#include <stdlib.h>
#include <stdio.h>
#include <string.h>

static char thread_stack[8 « 10];

static void thread_func(void)
{
    while (1)
    {
        unsigned long d = 0;

        /* Enable single stepping */
        asm volatile("pushf; pop %0; or $256,%0; push %0; popf\n"
                     : "=r" (d) : "r" (d));

        /* Some instructions */
        asm volatile("nop");
        asm volatile("nop");
        asm volatile("nop");
        asm volatile("mov $0x12345000, %%edx" : : : "edx"); // a non-existent cap
        asm volatile("int $0x30\n");
        asm volatile("nop");
        asm volatile("nop");
        asm volatile("nop");

        /* Disabled single stepping */
        asm volatile("pushf; pop %0; and $~256,%0; push %0; popf\n"
                     : "=r" (d) : "r" (d));

        /* You won't see those */
        asm volatile("nop");
        asm volatile("nop");
    }
}

```

```

        asm volatile("nop");
    }
}

int main(void)
{
    l4_msgtag_t tag;
    int ipc_stat = 0;
    l4_cap_idx_t th = l4re_util_cap_alloc();
    l4_exc_regs_t exc;
    l4_umword_t mr0, mr1;
    l4_utcb_t *u = l4_utcb();

    printf("Singlestep testing\n");

    if (l4_is_invalid_cap(th))
        return 1;

    l4_touch_rw(thread_stack, sizeof(thread_stack));
    l4_touch_ro(thread_func, 1);

    tag = l4_factory_create_thread(l4re_env()->factory, th);
    if (l4_error(tag))
        return 1;

    l4_thread_control_start();
    l4_thread_control_pager(l4re_env()->main_thread);
    l4_thread_control_exc_handler(l4re_env()->main_thread);
    l4_thread_control_bind((l4_utcb_t *)l4re_env()->first_free_utcb,
                          L4RE_THIS_TASK_CAP);
    l4_thread_control_alien(1);
    tag = l4_thread_control_commit(th);
    if (l4_error(tag))
        return 2;

    tag = l4_thread_ex_regs(th, (l4_umword_t)thread_func,
                          (l4_umword_t)thread_stack + sizeof(thread_stack),
                          0);

    if (l4_error(tag))
        return 3;

    l4_sched_param_t sp = l4_sched_param(1, 0);
    tag = l4_scheduler_run_thread(l4re_env()->scheduler, th, &sp);
    if (l4_error(tag))
        return 4;

    /* Pager/Exception loop */
    if (l4_msgtag_has_error(tag = l4_ipc_receive(th, u, L4_IPC_NEVER)))
    {
        printf("l4_ipc_receive failed");
        return 5;
    }
    memcpy(&exc, l4_utcb_exc(), sizeof(exc));
    mr0 = l4_utcb_mr()->mr[0];
    mr1 = l4_utcb_mr()->mr[1];

    for (;;)
    {
        if (l4_msgtag_is_exception(tag))
        {
            printf("PC = %08lx Trap = %08lx Err = %08lx, SP = %08lx SC-Nr: %lx\n",
                  l4_utcb_exc_pc(&exc), exc.trapno, exc.err,
                  exc.sp, exc.err >> 3);
            if (exc.err >> 3)
            {
                if (!(exc.err & 4))
                {
                    tag = l4_msgtag(L4_PROTO_ALLOW_SYSCALL,
                                    L4_UTCB_EXCEPTION_REGS_SIZE, 0, 0);
                    if (ipc_stat)
                        l4_kd_enter("Should not be 1");
                }
                else
                {
                    tag = l4_msgtag(L4_PROTO_NONE,
                                    L4_UTCB_EXCEPTION_REGS_SIZE, 0, 0);
                    if (!ipc_stat)
                        l4_kd_enter("Should not be 0");
                }
                ipc_stat = !ipc_stat;
            }
            l4_sleep(100);
        }
        else
            printf("Umm, non-handled request: %ld, %08lx %08lx\n",

```

```

        l4_msgtag_label(tag), mr0, mr1);

memcpy(l4_utcb_exc(), &exc, sizeof(exc));

/* Reply and wait */
if (l4_msgtag_has_error(tag = l4_ipc_call(th, u, tag, L4_IPC_NEVER)))
{
    printf("l4_ipc_call failed\n");
    return 5;
}
memcpy(&exc, l4_utcb_exc(), sizeof(exc));
mr0 = l4_utcb_mr()->mr[0];
mr1 = l4_utcb_mr()->mr[1];
}

return 0;
}

```

17.6 examples/sys/aliens/main.c

This example shows how system call tracing can be done.

This example shows how system call tracing can be done.

```

/* SPDX-License-Identifier: MIT */
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *      Alexander Warg <warg@os.inf.tu-dresden.de>,
 *      Björn Döbel <doebel@os.inf.tu-dresden.de>
 */
/*
 * Example to show syscall tracing.
 */
#if defined(ARCH_x86) || defined(ARCH_amd64)
// MEASURE only works on x86/amd64
// #define MEASURE
#endif

#include <l4/sys/ipc.h>
#include <l4/sys/thread.h>
#include <l4/sys/factory.h>
#include <l4/sys/scheduler.h>
#include <l4/sys/utcb.h>
#include <l4/util/util.h>
#include <l4/re/env.h>
#include <l4/re/c/util/cap_alloc.h>
#include <l4/re/c/util/kumem_alloc.h>
#include <l4/sys/debugger.h>

#include <stdlib.h>
#include <stdio.h>
#include <string.h>

/* Architecture specifics */
#if defined(ARCH_x86) || defined(ARCH_amd64)

static int
is_alien_after_call(l4_exc_regs_t const *exc)
{
    if defined(ARCH_x86)
        return exc->err & 4;
    else
        return exc->err == 1;
    #endif
}

static inline void
_print_exc_state(l4_exc_regs_t const *exc)
{
    printf("PC=%08lx SP=%08lx Err=%08lx Trap=%lx, %s syscall, SC-Nr: %lx\n",
        l4_utcb_exc_pc(exc), exc->sp, exc->err,
        exc->trapno, is_alien_after_call(exc) ? "after" : "before",
        exc->err >> 3);
}

#elif defined(ARCH_arm)

static int
is_alien_after_call(l4_exc_regs_t const *exc)
{
    return exc->err & 0x40; } // TODO: Should change this to (1 < 16)

static inline void

```

```

_print_exc_state(l4_exc_regs_t const *exc)
{
    printf("PC=%08lx SP=%08lx ULR=%08lx CPSR=%08lx Err=%lx/%lx, %s syscall\n",
        l4_utcb_exc_pc(exc), exc->sp, exc->ulr, exc->cpsr,
        exc->err, exc->err » 26,
        is_alien_after_call(exc) ? " after" : "before");
}

#elif defined(ARCH_arm64)

static int
is_alien_after_call(l4_exc_regs_t const *exc)
{ return exc->err & (1ul « 16); }

static inline void
_print_exc_state(l4_exc_regs_t const *exc)
{
    printf("PC=%08lx SP=%08lx PSTATE=%08lx Err=%lx/%lx, %s syscall\n",
        l4_utcb_exc_pc(exc), exc->sp, exc->pstate,
        exc->err, exc->err » 26,
        is_alien_after_call(exc) ? " after" : "before");
}

#elif defined(ARCH_mips)

static int
is_alien_after_call(l4_exc_regs_t const *exc)
{ return 0; }

static inline void
_print_exc_state(l4_exc_regs_t const *exc)
{
    printf("PC=%08lx SP=%08lx Cause=%lx, %s syscall\n",
        l4_utcb_exc_pc(exc), exc->sp, exc->cause,
        is_alien_after_call(exc) ? " after" : "before");
}

#elif defined(ARCH_riscv)

static int
is_alien_after_call(l4_exc_regs_t const *exc)
{ return exc->cause == L4_riscv_ec_l4_alien_after_syscall; }

static inline void
_print_exc_state(l4_exc_regs_t const *exc)
{
    printf("PC=%08lx SP=%08lx Cause=%lx, %s syscall\n",
        l4_utcb_exc_pc(exc), exc->sp, exc->cause,
        is_alien_after_call(exc) ? " after" : "before");
}

#else

static int
is_alien_after_call(l4_exc_regs_t const *exc)
{ return exc->err & 1; }

static inline void
_print_exc_state(l4_exc_regs_t const *exc)
{
    printf("PC=%08lx SP=%08lx, %s syscall\n",
        l4_utcb_exc_pc(exc), exc->sp,
        is_alien_after_call(exc) ? " after" : "before");
}

#endif

/* Measurement mode specifics.
 *
 * In measurement mode the code is less verbose and uses RDTSC for alien exception
 * performance measurement.
 */
#ifdef MEASURE

#include <l4/util/rdtsc.h>

static inline void
calibrate_timer(void)
{
    l4_calibrate_tsc(l4re_kip());
}

static inline void
print_timediff(l4_cpu_time_t start)
{
    e = l4_rdtsc();
    printf("time %lld\n", l4_tsc_to_ns(e - start));
}

```

```

}

static inline void
alien_sleep(void)
{
    l4_sleep(0);
}

static inline void
print_exc_state(l4_exc_regs_t const *exc)
{
    if (0)
        _print_exc_state(exc);
}

#else

static inline void
calibrate_timer(void)
{
}

static inline void
print_timediff(l4_cpu_time_t start)
{
    (void)start;
}

static inline l4_cpu_time_t
l4_rdtsc(void)
{
    return 0;
}

static inline void
alien_sleep(void)
{
    l4_sleep(1000);
}

static inline void
print_exc_state(l4_exc_regs_t const *exc)
{
    _print_exc_state(exc);
}

#endif

static char alien_thread_stack[8 « 10];
static l4_cap_idx_t alien;

static void alien_thread(void)
{
    while (1)
    {
        l4_ipc_call(0x1234 « L4_CAP_SHIFT, l4_utcb(),
                    l4_msgtag(0, 0, 0, 0), L4_IPC_NEVER);
        alien_sleep();
    }
}

int main(void)
{
    l4_msgtag_t tag;
    l4_cpu_time_t s;
    l4_utcb_t *u = l4_utcb();
    l4_exc_regs_t exc;
    l4_umword_t mr0, mr1;

    printf("Alien feature testing\n");

    l4_debugger_set_object_name(l4re_env()->main_thread, "alientest");

    /* Start alien thread */
    if (l4_is_invalid_cap(alien = l4re_util_cap_alloc()))
        return 1;

    l4_touch_rw(alien_thread_stack, sizeof(alien_thread_stack));

    tag = l4_factory_create_thread(l4re_env()->factory, alien);
    if (l4_error(tag))
        return 2;

    l4_debugger_set_object_name(alien, "alienth");

    l4_addr_t kumem;

```

```

if (l4re_util_kumem_alloc(&kumem, 0, L4RE_THIS_TASK_CAP, l4re_env()->rm))
    return 3;

l4_thread_control_start();
l4_thread_control_pager(l4re_env()->main_thread);
l4_thread_control_exc_handler(l4re_env()->main_thread);
l4_thread_control_bind((l4_utcb_t *)kumem, L4RE_THIS_TASK_CAP);
l4_thread_control_alien(1);
tag = l4_thread_control_commit(alien);
if (l4_error(tag))
    return 4;

tag = l4_thread_ex_regs(alien,
                        (l4_umword_t)alien_thread,
                        (l4_umword_t)alien_thread_stack + sizeof(alien_thread_stack),
                        0);

if (l4_error(tag))
    return 5;

l4_sched_param_t sp = l4_sched_param(1, 0);
tag = l4_scheduler_run_thread(l4re_env()->scheduler, alien, &sp);
if (l4_error(tag))
    return 6;

calibrate_timer();

/* Pager/Exception loop */
if (l4_msgtag_has_error(tag = l4_ipc_receive(alien, u, L4_IPC_NEVER)))
{
    printf("l4_ipc_receive failed");
    return 7;
}

memcpy(&exc, l4_utcb_exc(), sizeof(exc));
mr0 = l4_utcb_mr()->mr[0];
mr1 = l4_utcb_mr()->mr[1];

for (;;)
{
    s = l4_rdtsc();

    if (l4_msgtag_is_exception(tag))
    {
        print_exc_state(&exc);
        tag = l4_msgtag(is_alien_after_call(&exc)
                        ? 0 : L4_PROTO_ALLOW_SYSCALL,
                        L4_UTCB_EXCEPTION_REGS_SIZE, 0, 0);
    }
    else
        printf("Umm, non-handled request (like PF): %lx %lx\n", mr0, mr1);

    memcpy(l4_utcb_exc(), &exc, sizeof(exc));

    /* Reply and wait */
    if (l4_msgtag_has_error(tag = l4_ipc_call(alien, u, tag, L4_IPC_NEVER)))
    {
        printf("l4_ipc_call failed\n");
        return 8;
    }

    memcpy(&exc, l4_utcb_exc(), sizeof(exc));
    mr0 = l4_utcb_mr()->mr[0];
    mr1 = l4_utcb_mr()->mr[1];
    print_timediff(s);
}

return 0;
}

```

17.7 examples/sys/utcb-ipc/main.c

This example shows how to send IPC using the UTCB to store payload.

This example shows how to send IPC using the UTCB to store payload.

```

/* SPDX-License-Identifier: MIT */
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 * Alexander Warg <warg@os.inf.tu-dresden.de>,
 * Björn Döbel <doebel@os.inf.tu-dresden.de>
 */
#include <l4/sys/ipc.h>

```

```

#include <l4/sys/thread.h>
#include <l4/sys/factory.h>
#include <l4/sys/utcb.h>
#include <l4/sys/task.h>
#include <l4/sys/vcon.h>
#include <l4/re/env.h>
#include <l4/re/c/util/cap_alloc.h>
#include <l4/re/c/util/kumem_alloc.h>
#include <l4/util/thread.h>

#include <stdio.h>
#include <string.h>

static unsigned char stack2[8 « 10] __attribute__((aligned(8)));
static l4_cap_idx_t thread1_cap, thread2_cap;

static void vlogprintn(const char *s, int l)
{
    if (l > L4_VCON_WRITE_SIZE)
        l = L4_VCON_WRITE_SIZE;

    l4_vcon_send(L4_BASE_LOG_CAP, s, l);
}

static void vlogprint(const char *s)
{
    vlogprintn(s, strlen(s));
}

static void vlogputc(const char c)
{
    vlogprintn(&c, 1);
}

static void thread1(void)
{
    l4_msgregs_t *mr = l4_utcb_mr();
    l4_msgtag_t tag;
    int i, j;

    printf("Thread1 up (%p)\n", l4_utcb());

    for (i = 0; i < 10; i++)
    {
        for (j = 0; j < L4_UTCB_GENERIC_DATA_SIZE; j++)
            mr->mr[j] = 'A' + (i + j) % ('~' - 'A' + 1);
        tag = l4_msgtag(0, L4_UTCB_GENERIC_DATA_SIZE, 0, 0);
        if (l4_msgtag_has_error(l4_ipc_send(thread2_cap, l4_utcb(), tag, L4_IPC_NEVER)))
            printf("IPC-send error\n");
    }

    mr->mr[0] = 1;
    if (l4_msgtag_has_error(l4_ipc_send(thread2_cap, l4_utcb(), tag, L4_IPC_NEVER)))
        printf("IPC-send error\n");

    printf("Thread1 done\n");
}

L4UTIL_THREAD_STATIC_FUNC(thread2)
{
    l4_msgtag_t tag;
    l4_msgregs_t mr;
    unsigned i;

    // No printf() here because this would require a working pthread environment!
    vlogprint("Thread2 up\n");

    while (1)
    {
        if (l4_msgtag_has_error(tag = l4_ipc_receive(thread1_cap, l4_utcb(), L4_IPC_NEVER)))
            vlogprint("IPC receive error\n");
        memcpy(&mr, l4_utcb_mr(), sizeof(mr));
        if (mr.mr[0] == 1) // exit notification
            break;
        vlogprint("Thread2 receive: ");
        for (i = 0; i < l4_msgtag_words(tag); i++)
            vlogputc((char)mr.mr[i]);
        vlogprint("\n");
    }

    vlogprint("Thread2 done, switching to thread1\n");
    if (l4_msgtag_has_error(l4_ipc_send(thread1_cap, l4_utcb(),
                                      tag, L4_IPC_NEVER)))
        vlogprint("IPC-send error\n");

    // In theory this could hit if the above IPC send operation doesn't switch
    // to the other thread.

```

```

    __builtin_trap();
}

int main(void)
{
    l4_msgtag_t tag;

    thread1_cap = l4re_env()->main_thread;
    thread2_cap = l4re_util_cap_alloc();

    if (l4_is_invalid_cap(thread2_cap))
    {
        printf("Cannot allocate thread2 capability\n");
        return 1;
    }

    tag = l4_factory_create_thread(l4re_env()->factory, thread2_cap);
    if (l4_error(tag))
    {
        printf("Cannot create thread2\n");
        return 2;
    }

    l4_addr_t kumem;
    if (l4re_util_kumem_alloc(&kumem, 0, L4RE_THIS_TASK_CAP, l4re_env()->rm)
    {
        printf("Cannot allocate UTCB for thread2\n");
        return 3;
    }

    l4_thread_control_start();
    l4_thread_control_pager(l4re_env()->rm);
    l4_thread_control_exc_handler(l4re_env()->rm);
    l4_thread_control_bind((l4_utcb_t *)kumem, L4RE_THIS_TASK_CAP);
    tag = l4_thread_control_commit(thread2_cap);
    if (l4_error(tag))
    {
        printf("Cannot set thread2 thread parameters\n");
        return 4;
    }

    tag = l4_thread_ex_regs(thread2_cap,
                           (l4_umword_t)thread2,
                           (l4_umword_t)(stack2 + sizeof(stack2)), 0);
    if (l4_error(tag))
    {
        printf("Cannot set thread2 IP/SP\n");
        return 5;
    }

    l4_sched_param_t sp = l4_sched_param(1, 0);
    tag = l4_scheduler_run_thread(l4re_env()->scheduler, thread2_cap, &sp);
    if (l4_error(tag))
    {
        printf("Cannot start thread2\n");
        return 6;
    }

    thread1();

    if (l4_msgtag_has_error(l4_ipc_receive(thread2_cap, l4_utcb(),
                                           L4_IPC_NEVER)))
        printf("IPC-receive error\n");

    l4_task_unmap(L4RE_THIS_TASK_CAP,
                 l4_obj_fpage(thread2_cap, 0, L4_FPAGE_RWX),
                 L4_FP_ALL_SPACES);

    printf("Terminated thread2. Terminating.\n");
    return 0;
}

```

17.8 examples/sys/isr/main.c

Example of an interrupt service routine.

Example of an interrupt service routine.

```

/* SPDX-License-Identifier: MIT */
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,

```



```

*           Alexander Warg <warg@os.inf.tu-dresden.de>,
*           Björn Döbel <doebel@os.inf.tu-dresden.de>
*/
/*
* This example shall show how to connect to an interrupt, receive interrupt
* events and detach again. As the interrupt source we'll use the virtual
* key interrupt -- pin 0 of the log capability.
*/

#include <l4/re/c/util/cap_alloc.h>
#include <l4/re/c/namespace.h>
#include <l4/sys/factory.h>
#include <l4/sys/icu.h>
#include <l4/sys/irq.h>
#include <l4/sys/vcon.h>
#include <l4/sys/utcb.h>

#include <stdio.h>

int main(void)
{
    int const irqno = 0;
    l4_cap_idx_t irqcap, icucap;
    l4_vcon_attr_t attr;
    long err;

    icucap = l4re_env()->log;

    /* Get a free capability slot for the ICU capability. */
    if (l4_is_invalid_cap(icucap))
    {
        printf("Did not find the Vlog ICU.\n");
        return 1;
    }

    /* Get another free capability slot for the corresponding IRQ object. */
    if (l4_is_invalid_cap(irqcap = l4re_util_cap_alloc()))
    {
        printf("Cannot allocate capability slot.\n");
        return 1;
    }

    /* Create IRQ object. */
    if ((err = l4_error(l4_factory_create_irq(l4re_env()->factory, irqcap)))
    {
        printf("Could not create IRQ object: %ld (%s).\n", err, l4sys_errtostr(err));
        return 1;
    }

    /*
    * Bind the recently allocated IRQ object to the IRQ number irqno as provided
    * by the ICU.
    */
    if ((err = l4_error(l4_icu_bind(icucap, irqno, irqcap)))
    {
        printf("Could not bind IRQ%d to ICU: %ld (%s).\n", irqno, err, l4sys_errtostr(err));
        return 1;
    }

    if ((err = l4_error(l4_vcon_get_attr(icucap, &attr)))
    {
        printf("Could not get Vcon attributes: %ld (%s).\n", err, l4sys_errtostr(err));
        return 1;
    }

    /* Disable echo at Vcon console. */
    attr.l_flags &= ~L4_VCON_ECHO;

    if ((err = l4_error(l4_vcon_set_attr(icucap, &attr)))
    {
        printf("Could not set Vcon attributes: %ld (%s).\n", err, l4sys_errtostr(err));
        return 1;
    }

    printf("Vcon echo disabled.\n");

    /* Bind ourselves to the IRQ. Define the IPC label which is sent if an IRQ
    * IPC arrives. */
    if ((err = l4_error(l4_rcv_ep_bind_thread(irqcap, l4re_env()->main_thread, 0x1234)))
    {
        printf("Could not bind to IRQ%d: %ld (%s).\n", irqno, err, l4sys_errtostr(err));
        return 1;
    }

    printf("Attached to key IRQ %d.\nPress keys now, Shift-Q to exit.\n", irqno);

    /* IRQ receive loop. */

```

```

while (1)
{
    /* Wait for the interrupt to happen. If we received an IRQ, the label
     * return code is set to 0. If we didn't receive an IRQ, the error flag
     * in the message tag is set and l4_error() reads the IPC error code from
     * the UTCB. */
    l4_umword_t label;
    if ((err = l4_error(l4_irq_wait(irqcap, &label, L4_IPC_NEVER))))
        printf("Could not receive IRQ: %ld (%s).\n", err, l4sys_errtostr(err));
    else
    {
        char buf[128];
        int n;

        if (label != 0x1234)
        {
            printf("Unexpected label %0lx -- ignoring interrupt.\n", label);
            continue;
        }

        /* Process the interrupt -- may do a 'break' */
        printf("Got IRQ with expected label 0x%lX.\n", label);
        n = l4_vcon_read(icucap, buf, sizeof(buf));
        if (n < 0)
            printf("Could not read from Vcon interface: %d (%s).\n", n, l4sys_errtostr(n));
        else
        {
            unsigned i;
            int terminate = 0;
            for (i = 0; i < (unsigned)n && i < sizeof(buf); ++i)
            {
                int c = (unsigned char)buf[i];
                if (c >= 32 && c < 128) // Filter UTF-8 encodings.
                    printf("Got key '%c'.\n", c);
                else
                    printf("Got keycode %d.\n", c);
                if (buf[i] == 'Q')
                    terminate = 1;
            }

            if (terminate)
                break;
        }
    }

    /* We're done, detach from the interrupt. */
    if ((err = l4_error(l4_irq_detach(irqcap))))
        printf("Could not detach from IRQ: %ld (%s).\n", err, l4sys_errtostr(err));

    printf("Application terminated.\n");
    return 0;
}

```

17.9 examples/clntsrv/src/server.cc

Client/Server example using C++ infrastructure – Server implementation.

Client/Server example using C++ infrastructure – Server implementation.

```

/* SPDX-License-Identifier: MIT */
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 * Alexander Warg <warg@os.inf.tu-dresden.de>
 */
#include <stdio.h>
#include <l4/re/env>
#include <l4/re/util/cap_alloc>
#include <l4/re/util/object_registry>
#include <l4/re/util/br_manager>
#include <l4/sys/cxx/ipc_epiface>

#include "shared.h"

static L4Re::Util::Registry_server<> server;

class Calculation_server : public L4::Epiface_t<Calculation_server, Calc>
{
public:
    int op_sub(Calc::Rights, l4_uint32_t a, l4_uint32_t b, l4_uint32_t &res)
    {

```

```

    res = a - b;
    return 0;
}

int op_neg(Calc::Rights, l4_uint32_t a, l4_uint32_t &res)
{
    res = -a;
    return 0;
}
};

int
main()
{
    static Calculation_server calc;

    // Register calculation server
    if (!server.registry()->register_obj(&calc, "calc_server").is_valid())
    {
        printf("Could not register my service, is there a 'calc_server' in the caps table?\n");
        return 1;
    }

    printf("Welcome to the calculation server!\n"
           "I can do subtractions and negations.\n");

    // Wait for client requests
    server.loop();

    return 0;
}

```

17.10 examples/clntsrv/src/client.cc

Client/Server example using C++ infrastructure – Client implementation.

Client/Server example using C++ infrastructure – Client implementation.

```

/* SPDX-License-Identifier: MIT */
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *             Alexander Warg <warg@os.inf.tu-dresden.de>
 */
#include <l4/sys/err.h>
#include <l4/sys/types.h>
#include <l4/re/env>
#include <l4/re/util/cap_alloc>

#include <stdio.h>
#include "shared.h"

int
main()
{
    L4::Cap<Calc> server = L4Re::Env::env()->get_cap<Calc>("calc_server");
    if (!server.is_valid())
    {
        printf("Could not get server capability!\n");
        return 1;
    }

    l4_uint32_t val1 = 8;
    l4_uint32_t val2 = 5;

    printf("Asking for %d - %d\n", val1, val2);
    if (server->sub(val1, val2, &val1))
    {
        printf("Error talking to server\n");
        return 1;
    }
    printf("Result of subtract call: %d\n", val1);

    printf("Asking for -%d\n", val1);
    if (server->neg(val1, &val1))
    {
        printf("Error talking to server\n");
        return 1;
    }
    printf("Result of negate call: %d\n", val1);

    return 0;
}

```

17.11 examples/clntsrv/src/shared.h

Client/Server example using C++ infrastructure – Shared header file.

Client/Server example using C++ infrastructure – Shared header file.

```

/* SPDX-License-Identifier: MIT */
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
 */

#pragma once

#include <l4/sys/capability>
#include <l4/sys/cxx/ipc_iface>

struct Calc : L4::Kobject_t<Calc, L4::Kobject, 0x44>
{
    L4_INLINE_RPC(int, sub, (l4_uint32_t a, l4_uint32_t b, l4_uint32_t *res));
    L4_INLINE_RPC(int, neg, (l4_uint32_t a, l4_uint32_t *res));
    typedef L4::Typeid::Rpc<sub_t, neg_t> Rpc;
};

```

17.12 examples/clntsrv/configs/clntsrv.cfg

Sample configuration file for the client/server example.

Sample configuration file for the client/server example.

```

-- vim:set ft=lua:

-- Include L4 functionality
local L4 = require("L4");

-- Some shortcut for less typing
local ld = L4.default_loader;

-- Channel for the two programs to talk to each other.
local calc_server = ld:new_channel();

-- The server program, getting the channel in server mode.
ld:start({ caps = { calc_server = calc_server:svr() },
    log = { "server", "blue" } },
    "rom/ex_clntsrv-server");

-- The client program, getting the 'calc_server' channel to be able to talk
-- to the server. The client will be started with a green log output.
ld:start({ caps = { calc_server = calc_server },
    log = { "client", "green" } },
    "rom/ex_clntsrv-client");

```

17.13 examples/libs/l4re/c/ma+rm.c

Coarse grained memory allocation, in C.

Coarse grained memory allocation, in C.

```

/* SPDX-License-Identifier: MIT */
/*
 * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
 */

#include <l4re/c/mem_alloc.h>
#include <l4re/c/rm.h>
#include <l4re/c/util/cap_alloc.h>
#include <l4/sys/err.h>
#include <stdio.h>
#include <string.h>

static int allocate_mem(unsigned long size_in_bytes, unsigned long flags,
    void **virt_addr)
{

```

```

int r;
l4re_ds_t ds;

/* Allocate a free capability index for our data space */
ds = l4re_util_cap_alloc();
if (l4_is_invalid_cap(ds))
    return -L4_ENOMEM;

size_in_bytes = l4_trunc_page(size_in_bytes);

/* Allocate memory via a dataspace */
if ((r = l4re_ma_alloc(size_in_bytes, ds, flags)))
    return r;

/* Make the dataspace visible in our address space */
*virt_addr = 0;
if ((r = l4re_rm_attach(virt_addr, size_in_bytes,
                        L4RE_RM_F_SEARCH_ADDR | L4RE_RM_F_RWX, ds, 0,
                        flags & L4RE_MA_SUPER_PAGES
                        ? L4_SUPERPAGESHIFT : L4_PAGESHIFT)))
{
    /* Free dataspace again */
    l4re_util_cap_free_um(ds);
    return r;
}

/* Done, virtual address is in virt_addr */
return 0;
}

static int free_mem(void *virt_addr)
{
    int r;
    l4re_ds_t ds;

    /* Detach memory from our address space */
    if ((r = l4re_rm_detach_ds(virt_addr, &ds)))
        return r;

    /* Free memory at our memory allocator */
    l4re_util_cap_free_um(ds);

    /* All went ok */
    return 0;
}

int main(void)
{
    void *virt;

    /* Allocate memory: 16k Bytes (usually) */
    if (allocate_mem(4 * L4_PAGESIZE, 0, &virt))
        return 1;

    printf("Allocated memory.\n");

    /* Do something with the memory */
    memset(virt, 0x12, 4 * L4_PAGESIZE);

    printf("Touched memory.\n");

    /* Free memory */
    if (free_mem(virt))
        return 2;

    printf("Freed and done. Bye.\n");

    return 0;
}

```

17.14 examples/libs/l4re/c++/mem_alloc/ma+rm.cc

Coarse grained memory allocation, in C++.

Coarse grained memory allocation, in C++.

```

/* SPDX-License-Identifier: MIT */
/*
 * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
 */

#include <l4/re/mem_alloc>

```

```

#include <l4/re/rm>
#include <l4/re/env>
#include <l4/re/dataspace>
#include <l4/re/util/cap_alloc>
#include <l4/sys/err.h>
#include <cstdio>
#include <cstring>

static int allocate_mem(unsigned long size_in_bytes, unsigned long flags,
                        void **virt_addr)
{
    int r;
    L4::Cap<L4Re::Dataspace> d;

    /* Allocate a free capability index for our data space */
    d = L4Re::Util::cap_alloc.alloc<L4Re::Dataspace>();
    if (!d.is_valid())
        return -L4_ENOMEM;

    size_in_bytes = l4_trunc_page(size_in_bytes);

    /* Allocate memory via a dataspace */
    if ((r = L4Re::Env::env()->mem_alloc()->alloc(size_in_bytes, d, flags)))
        return r;

    /* Make the dataspace visible in our address space */
    *virt_addr = 0;
    if ((r = L4Re::Env::env()->rm()->attach(virt_addr, size_in_bytes,
                                           L4Re::Rm::F::Search_addr | L4Re::Rm::F::RW,
                                           L4::Ipc::make_cap_rw(d), 0,
                                           flags & L4Re::Mem_alloc::Super_pages
                                           ? L4_SUPERPAGESHIFT : L4_PAGESHIFT)))
        return r;

    /* Done, virtual address is in virt_addr */
    return 0;
}

static int free_mem(void *virt_addr)
{
    int r;
    L4::Cap<L4Re::Dataspace> ds;

    /* Detach memory from our address space */
    if ((r = L4Re::Env::env()->rm()->detach(virt_addr, &ds)))
        return r;

    /* Release and return capability slot to allocator */
    L4Re::Util::cap_alloc.free(ds, L4Re::Env::env()->task().cap());

    /* All went ok */
    return 0;
}

int main(void)
{
    void *virt;

    /* Allocate memory: 16k Bytes (usually) */
    if (allocate_mem(4 * L4_PAGESIZE, 0, &virt))
        return 1;

    printf("Allocated memory.\n");

    /* Do something with the memory */
    memset(virt, 0x12, 4 * L4_PAGESIZE);

    printf("Touched memory.\n");

    /* Free memory */
    if (free_mem(virt))
        return 2;

    printf("Freed and done. Bye.\n");

    return 0;
}

```

17.15 examples/libs/l4re/c++/shared_ds/ds_clnt.cc

Sharing memory between applications, client side.

Sharing memory between applications, client side.

```

/* SPDX-License-Identifier: MIT */
/*
 * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *      Alexander Warg <warg@os.inf.tu-dresden.de>
 */

#include <l4/re/util/cap_alloc> // L4::Cap
#include <l4/re/dataspace>      // L4Re::Dataspace
#include <l4/re/rm>              // L4::Rm
#include <l4/re/env>             // L4::Env
#include <l4/sys/cache.h>

#include <cstring>
#include <cstdio>
#include <unistd.h>

#include "interface.h"

int main()
{
    /*
     * Try to get server interface cap.
     */

    L4::Cap<My_interface> svr = L4Re::Env::env()->get_cap<My_interface>("shm");
    if (!svr.is_valid())
    {
        printf("Could not get the server capability\n");
        return 1;
    }

    /*
     * Alloc data space cap slot
     */
    L4::Cap<L4Re::Dataspace> ds = L4Re::Util::cap_alloc.alloc<L4Re::Dataspace>();
    if (!ds.is_valid())
    {
        printf("Could not get capability slot!\n");
        return 1;
    }

    /*
     * Alloc server notifier IRQ cap slot
     */
    L4::Cap<L4::Irq> irq = L4Re::Util::cap_alloc.alloc<L4::Irq>();
    if (!irq.is_valid())
    {
        printf("Could not get capability slot!\n");
        return 1;
    }

    /*
     * Request shared data-space cap.
     */
    if (svr->get_shared_buffer(ds, irq))
    {
        printf("Could not get shared memory dataspace!\n");
        return 1;
    }

    /*
     * Attach to arbitrary region
     */
    char *addr = 0;
    int err = L4Re::Env::env()->rm()->attach(&addr, ds->size(),
                                             L4Re::Rm::F::Search_addr | L4Re::Rm::F::RW,
                                             L4::Ipc::make_cap_rw(ds));

    if (err < 0)
    {
        printf("Error attaching data space: %s\n", l4sys_errtostr(err));
        return 1;
    }

    printf("Content: %s\n", addr);

    // wait a bit for the demo effect
    printf("Sleeping a bit...\n");
    sleep(1);

    /*
     * Fill in new stuff
     */
    memset(addr, 0, ds->size());
    char const * const msg = "Hello from client, too!";
    printf("Setting new content in shared memory\n");
    snprintf(addr, strlen(msg)+1, msg);
}

```

```

l4_cache_clean_data((unsigned long)addr,
                    (unsigned long)addr + strlen(msg) + 1);

// notify the server
irq->trigger();

/*
 * Detach region containing addr, result should be Detached_ds (other results
 * only apply if we split regions etc.).
 */
err = L4Re::Env::env()->rm()->detach(addr, 0);
if (err)
    printf("Failed to detach region\n");

/* Free objects and capabilities, just for completeness. */
L4Re::Util::cap_alloc.free(ds, L4Re::This_task);
L4Re::Util::cap_alloc.free(irq, L4Re::This_task);

return 0;
}

```

17.16 examples/libs/l4re/c++/shared_ds/ds_srv.cc

Sharing memory between applications, server/creator side.

Sharing memory between applications, server/creator side.

```

/* SPDX-License-Identifier: MIT */
/*
 * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 * Alexander Warg <warg@os.inf.tu-dresden.de>
 */

#include <l4/re/env>
#include <l4/re/error_helper>
#include <l4/re/namespace>
#include <l4/re/util/cap_alloc>
#include <l4/re/util/object_registry>
#include <l4/re/dataspace>
#include <l4/cxx/ipc_server>
#include <l4/util/util.h>

#include <l4/sys/typeinfo_svr>

#include <cstring>
#include <cstdio>
#include <unistd.h>
#include <pthread.h>
#include <pthread-l4.h>
#include <thread>

#include "interface.h"

class My_server_obj : public L4::Server_object_t<L4::Kobject>
{
private:
    L4::Cap<L4Re::Dataspace> _shm;
    L4::Cap<L4::Irq> _irq;

public:
    explicit My_server_obj(L4::Cap<L4Re::Dataspace> shm, L4::Cap<L4::Irq> irq)
        : _shm(shm), _irq(irq)
    {}

    int dispatch(l4_umword_t obj, L4::Ipc::Iostream &ios);
};

int My_server_obj::dispatch(l4_umword_t obj, L4::Ipc::Iostream &ios)
{
    // we don't care about the original object reference, however
    // we could read out the access rights from the lowest 2 bits
    (void) obj;

    l4_msgtag_t t;
    ios » t; // extract the tag

    switch (t.label())
    {
    case L4::Meta::Protocol:
        // handle the meta protocol requests, implementing the

```



```

    // runtime dynamic type system for L4 objects.
    return L4::Util::handle_meta_request<My_interface>(ios);
case 0:
    // since we have just one operation we have no opcode dispatch,
    // and just return the data-space and the notifier IRQ capabilities
    ios << _shm << _irq;
    return 0;
default:
    // every other protocol is not supported.
    return -L4_EBADPROTO;
}
}

class Shm_observer : public L4::Irq_handler_object
{
private:
    char *_shm;

public:
    explicit Shm_observer(char *shm)
        : _shm(shm)
    {}

    int dispatch(l4_umword_t obj, L4::Ipc::Iostream &ios);
};

int Shm_observer::dispatch(l4_umword_t obj, L4::Ipc::Iostream &ios)
{
    // We don't care about the original object reference, however
    // we could read out the access rights from the lowest 2 bits
    (void)obj;

    // Since we end up here in this function, we got a 'message' from the IRQ
    // that is bound to us. The 'ios' stream won't contain any valuable info.
    (void)ios;

    printf("Client sent us: %s\n", _shm);

    return 0;
}

enum
{
    DS_SIZE = 4 << 12,
};

static char *get_ds(L4::Cap<L4Re::Dataspace> *_ds)
{
    *_ds = L4Re::Util::cap_alloc.alloc<L4Re::Dataspace>();
    if (!(*_ds).is_valid())
    {
        printf("Dataspace allocation failed.\n");
        return 0;
    }

    int err = L4Re::Env::env()->mem_alloc()->alloc(DS_SIZE, *_ds, 0);
    if (err < 0)
    {
        printf("mem_alloc->alloc() failed.\n");
        L4Re::Util::cap_alloc.free(*_ds);
        return 0;
    }

    /*
     * Attach DS to local address space
     */
    char *_addr = 0;
    err = L4Re::Env::env()->rm()->attach(&_addr, (*_ds)->size(),
                                         L4Re::Rm::F::Search_addr | L4Re::Rm::F::RW,
                                         L4::Ipc::make_cap_rw(*_ds));

    if (err < 0)
    {
        printf("Error attaching data space: %s\n", l4sys_errtostr(err));
        L4Re::Util::cap_alloc.free(*_ds);
        return 0;
    }

    /*
     * Success! Write something to DS.
     */
    printf("Attached DS\n");
    static char const * msg = "[DS] Hello from server!";
    snprintf(_addr, strlen(msg) + 1, msg);

    return _addr;
}

```

```

}

static void *server_thread(void *)
{
    L4::Cap<L4::Thread> l4_thread = Pthread::L4::cap(pthread_self());
    L4Re::Util::Registry_server<> server(l4_thread, L4Re::Env::env()->factory());

    L4::Cap<L4Re::Dataspace> ds;
    char *addr;

    if (!(addr = get_ds(&ds)))
        return nullptr;

    // First the IRQ handler, because we need it in the My_server_obj object
    Shm_observer observer(addr);

    // Registering the observer as an IRQ handler, this allocates an
    // IRQ object using the factory of our server.
    L4::Cap<L4::Irq> irq = server.registry()->register_irq_obj(&observer);

    // Now the initial server object shared with the client via our parent.
    // it provides the data-space and the IRQ capabilities to a client.
    My_server_obj server_obj(ds, irq);

    // Registering the server object to the capability 'shm' in our the L4Re::Env.
    // This capability must be provided by the parent. (see the shared_ds.lua)
    server.registry()->register_obj(&server_obj, "shm");

    // Run our server loop.
    server.loop();
}

int main()
{
    pthread_attr_t pattr;

    if (pthread_attr_init(&pattr))
        L4Re::throw_error(-L4_ENOMEM, "Initialize pthread attributes");

    pthread_t thr;
    L4Re::chksys(pthread_create(&thr, &pattr, server_thread, nullptr),
        "Create server thread");
    L4Re::chksys(pthread_attr_destroy(&pattr), "Destroy pthread attributes");

    l4_sleep_forever();

    return 0;
}

```

17.17 examples/libs/l4re/c++/shared_ds/shared_ds.cfg

Sharing memory between applications, configuration file.

Sharing memory between applications, configuration file.

```

-- Include L4 functionality
local L4 = require("L4");

-- Create a channel from the client to the server
local channel = L4.default_loader:new_channel();

-- Start the server, giving the channel with full server rights.
-- The server will have a yellow log output.
L4.default_loader:start(
{
    caps = { shm = channel:svr() },
    log = { "server", "yellow" }
},
"rom/ex_l4re_ds_srv"
);

-- Start the client, giving it the channel with read only rights. The
-- log output will be green.
L4.default_loader:start(
{
    caps = { shm = channel },
    log = { "client", "green" },
    l4re_dbg = L4.Dbg.Warn
},
"rom/ex_l4re_ds_clnt"
);

```

17.18 examples/libs/l4re/streammap/server.cc

Client/Server example showing how to map a page to another task – Server implementation.

Client/Server example showing how to map a page to another task – Server implementation. Note that there's also a shared memory library that supplies this functionality in more convenient way.

```

/* SPDX-License-Identifier: MIT */
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *           Alexander Warg <warg@os.inf.tu-dresden.de>
 */
#include <stdio.h>
#include <l4re/env>
#include <l4re/util/cap_alloc>
#include <l4re/util/object_registry>
#include <l4cxx/ipc_server>

#include "shared.h"

static char page_to_map[L4_PAGESIZE] __attribute__((aligned(L4_PAGESIZE)));

static L4Re::Util::Registry_server<> server;

class Smap_server : public L4::Server_object_t<Mapper>
{
public:
    int dispatch(l4_umword_t obj, L4::Ipc::Iostream &ios);
};

int
Smap_server::dispatch(l4_umword_t, L4::Ipc::Iostream &ios)
{
    l4_msgtag_t t;
    ios » t;

    // We're only talking the Map_example protocol
    if (t.label() != Mapper::Protocol)
        return -L4_EBADPROTO;

    L4::Opcode opcode;
    ios » opcode;

    switch (opcode)
    {
    case Mapper::Do_map:
        l4_addr_t snd_base;
        ios » snd_base;
        // put something into the page to read it out at the other side
        snprintf(page_to_map, sizeof(page_to_map), "Hello from the server!");
        printf("Sending to client\n");
        // send page
        ios « L4::Ipc::Snd_fpage::mem((l4_addr_t)page_to_map, L4_PAGESHIFT,
                                     L4_FPAGE_RO, snd_base);
        return L4_EOK;
    default:
        return -L4_ENOSYS;
    }
}

int
main()
{
    static Smap_server smap;

    // Register server
    if (!server.registry()->register_obj(&smap, "smap").is_valid())
    {
        printf("Could not register my service, read-only namespace?\n");
        return 1;
    }

    printf("Welcome to the memory map example server!\n");

    // Wait for client requests
    server.loop();

    return 0;
}

```

17.19 examples/libs/l4re/streammap/client.cc

Client/Server example showing how to map a page to another task – Client implementation.

Client/Server example showing how to map a page to another task – Client implementation. Note that there's also a shared memory library that supplies this functionality in more convenient way.

```

/* SPDX-License-Identifier: MIT */
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *               Alexander Warg <warg@os.inf.tu-dresden.de>
 */
#include <l4/sys/err.h>
#include <l4/sys/types.h>
#include <l4/re/env>
#include <l4/re/util/cap_alloc>
#include <l4/cxx/ipc_stream>

#include <stdio.h>

#include "shared.h"

static int
func_smap_call(L4::Cap<void> const &server)
{
    L4::Ipc::Iostream s(l4_utcb());
    l4_addr_t addr = 0;
    int err;

    if ((err = L4Re::Env::env()->rm()->reserve_area(&addr, L4_PAGESIZE,
                                                    L4Re::Rm::F::Search_addr)))
    {
        printf("The reservation of one page within our virtual memory failed with %d\n", err);
        return 1;
    }

    s << L4::Opcode(Mapper::Do_map)
      << (l4_addr_t)addr;
    s << L4::Ipc::Rcv_fpage::mem((l4_addr_t)addr, L4_PAGESHIFT, 0);
    int r = l4_error(s.call(server.cap(), Mapper::Protocol));
    if (r)
        return r; // failure

    printf("String sent by server: %s\n", (char *)addr);

    return 0; // ok
}

int
main()
{
    L4::Cap<void> server = L4Re::Env::env()->get_cap<void>("smap");
    if (!server.is_valid())
    {
        printf("Could not get capability slot!\n");
        return 1;
    }

    printf("Asking for page from server\n");

    if (func_smap_call(server))
    {
        printf("Error talking to server\n");
        return 1;
    }
    printf("It worked!\n");

    L4Re::Util::cap_alloc.free(server, L4Re::This_task);

    return 0;
}

```

17.20 examples/libs/l4re/streammap/streammap.cfg

Sample configuration file for the client/server map example.

Sample configuration file for the client/server map example.

```
-- vim:set ft=lua:

-- Include L4 functionality
local L4 = require("L4");

-- Channel for the communication between the server and the client.
local smap_channel = L4.default_loader:new_channel();

-- The server program, using the 'smap' channel in server
-- mode. The log prefix will be 'server', colored yellow.
L4.default_loader:start({ caps = { smap = smap_channel:svr() },
                        log = { "server", "yellow" } },
                      "rom/ex_smap-server");

-- The client program.
-- It is given the 'smap' channel to be able to talk to the server.
-- The log prefix will be 'client', colored green.
L4.default_loader:start({ caps = { smap = smap_channel },
                        log = { "client", "green" } },
                      "rom/ex_smap-client");
```

17.21 examples/libs/libirq/loop.c

libirq usage example using a self-created thread.

libirq usage example using a self-created thread.

```
/* SPDX-License-Identifier: MIT */
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
 */

#include <l4/irq/irq.h>
#include <l4/util/util.h>
#include <stdio.h>
#include <pthread.h>

enum { IRQ_NO = 17 };

static void isr_handler(void)
{
    printf("Got IRQ %d\n", IRQ_NO);
}

static void *isr_thread(void *data)
{
    l4irq_t *irq;
    (void) data;

    if (!(irq = l4irq_attach(IRQ_NO)))
        return NULL;

    while (1)
    {
        if (l4irq_wait(irq))
            continue;
        isr_handler();
    }

    return NULL;
}

int main(void)
{
    pthread_t thread;

    if (pthread_create(&thread, NULL, isr_thread, NULL))
        return 1;

    l4_sleep_forever();
    return 0;
}
```

17.22 examples/libs/libirq/async_isr.c

libirq usage example using asynchronous ISR handler functionality.

libirq usage example using asynchronous ISR handler functionality.

```

/* SPDX-License-Identifier: MIT */
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
 */
/*
 * This example shall show how to use the libirq.
 */

#include <l4/irq/irq.h>
#include <l4/util/util.h>

#include <stdio.h>

enum { IRQ_NO = 17 };

static void isr_handler(void *data)
{
    (void) data;
    printf("Got IRQ %d\n", IRQ_NO);
}

int main(void)
{
    const int seconds = 5;
    l4irq_t *irqdesc;

    if (!(irqdesc = l4irq_request(IRQ_NO, isr_handler, 0, 0xff, 0)))
    {
        printf("Requesting IRQ %d failed\n", IRQ_NO);
        return 1;
    }

    printf("Attached to key IRQ %d\nPress keys now, will terminate in %d seconds\n",
           IRQ_NO, seconds);

    l4_sleep(seconds * 1000);

    if (l4irq_release(irqdesc))
    {
        printf("Failed to release IRQ\n");
        return 1;
    }

    printf("Bye\n");
    return 0;
}

```

17.23 examples/sys/migrate/thread_migrate.cc

Thread migration example.

Thread migration example.

```

/* SPDX-License-Identifier: MIT */
/*
 * (c) 2008-2009 Author(s)
 */
#include <l4/sys/scheduler>
#include <l4/re/env>
#include <l4/re/util/cap_alloc>

#include <pthread-l4.h>
#include <unistd.h>
#include <stdio.h>
#include <string.h>

enum { NR_THREADS = 12 };
static L4::Cap<L4::Thread> threads[NR_THREADS];
static l4_umword_t cpu_map, cpu_nrs;

/* Function for the threads. The content is not really relevant, so lets
 * just sleep around a bit. */

```

```

static void *thread_fn(void *)
{
    while (1)
        sleep(1);

    return 0;
}

/* Check how many CPUs we have available.
 */
static int check_cpus(void)
{
    l4_sched_cpu_set_t cs = l4_sched_cpu_set(0, 0);

    if (l4_error(L4Re::Env::env()->scheduler()->info(&cpu_nrs, &cs)) < 0)
        return 1;

    cpu_map = cs.map;

    printf("%ld maximal supported CPUs.\n", cpu_nrs);
    if (cpu_nrs >= L4_MWORD_BITS)
    {
        printf("Will only handle %ld CPUs.\n", cpu_nrs);
        cpu_nrs = L4_MWORD_BITS;
    }
    else if (cpu_nrs == 1)
        printf("Only found 1 CPU.\n");

    return cpu_nrs < 2;
}

/* Create a couple of threads and store their capabilities in an array */
static int create_threads(void)
{
    unsigned i;

    for (i = 0; i < NR_THREADS; ++i)
    {
        pthread_t t;

        if (pthread_create(&t, NULL, thread_fn, NULL))
            return 1;

        threads[i] = L4::Cap<L4::Thread>(pthread_l4_cap(t));
    }
    printf("Created %d threads.\n", NR_THREADS);
    return 0;
}

/* Helper function to get the next CPU */
static unsigned get_next_cpu(unsigned c)
{
    unsigned x = c;
    for (;;)
    {
        x = (x + 1) % cpu_nrs;
        if (L4Re::Env::env()->scheduler()->is_online(x))
            return x;
        if (x == c)
            return c;
    }
}

/* Function that shuffles the threads on the available CPUs */
static void shuffle(void)
{
    unsigned start = 0;
    while (1)
    {
        unsigned t;
        unsigned c = start;
        for (t = 0; t < NR_THREADS; ++t)
        {
            l4_sched_param_t sp = l4_sched_param(20);
            c = get_next_cpu(c);
            sp.affinity = l4_sched_cpu_set(c, 0);
            if (l4_error(L4Re::Env::env()->scheduler()->run_thread(threads[t], sp)))
                printf("Error migrating thread%02d to CPU%02d\n", t, c);
            printf("Migrated Thread%02d -> CPU%02d\n", t, c);
        }

        start++;
        if (start == cpu_nrs)
            start = 0;
        sleep(1);
    }
}

```

```

}

int main(void)
{
    if (check_cpus())
        return 1;

    if (create_threads())
        return 1;

    shuffle();

    return 0;
}

```

17.24 examples/sys/migrate/thread_migrate.cfg

Sample configuration file for the thread migration example.

Sample configuration file for the thread migration example.

```

-- vim:set ft=lua:

local L4 = require("L4");

-- The log prefix will be 'migrate', colored green.
L4.default_loader:start({ log = { "migrate", "green" } },
    "rom/ex_thread_migrate");

```

17.25 tmpfs/lib/src/fs.cc

Example file system for [L4Re::Vfs](#).

Example file system for [L4Re::Vfs](#).

```

/*
 * (c) 2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *      Alexander Warg <warg@os.inf.tu-dresden.de>
 *      economic rights: Technische Universität Dresden (Germany)
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU Lesser General Public License 2.1.
 * Please see the COPYING-LGPL-2.1 file for details.
 */

#include <l4/l4re_vfs/backend>
#include <l4/cxx/string>
#include <l4/cxx/avl_tree>

#include <sys/stat.h>
#include <sys/ioctl.h>
#include <dirent.h>

#include <cstdio>
#include <cstdlib>
#include <cstring>

namespace {

using namespace L4Re::Vfs;
using cxx::Ref_ptr;

class File_data
{
public:
    File_data() : _buf(0), _size(0) {}

    unsigned long put(unsigned long offset,
        unsigned long bufsize, void *srcbuf);
    unsigned long get(unsigned long offset,
        unsigned long bufsize, void *dstbuf);

    unsigned long size(unsigned long offset);

```



```

    unsigned long size() const { return _size; }

    ~File_data() noexcept { free(_buf); }

private:
    void *_buf;
    unsigned long _size;
};

unsigned long
File_data::put(unsigned long offset, unsigned long bufsize, void *srcbuf)
{
    if (offset + bufsize > _size)
        size(offset + bufsize);

    if (!_buf)
        return 0;

    memcpy((char *)_buf + offset, srcbuf, bufsize);
    return bufsize;
}

unsigned long
File_data::get(unsigned long offset, unsigned long bufsize, void *dstbuf)
{
    unsigned long s = bufsize;

    if (offset > _size)
        return 0;

    if (offset + bufsize > _size)
        s = _size - offset;

    memcpy(dstbuf, (char *)_buf + offset, s);
    return s;
}

unsigned long
File_data::size(unsigned long offset)
{
    if (offset != _size)
    {
        _size = offset;
        _buf = realloc(_buf, _size);
    }

    if (!_buf)
        return 0;
    return -ENOSPC;
}

class Node : public cxx::Avl_tree_node
{
public:
    Node(const char *path, mode_t mode)
        : _ref_cnt(0), _path(strdup(path))
    {
        memset(&_info, 0, sizeof(_info));
        _info.st_mode = mode;
    }

    const char *path() const { return _path; }
    struct stat64 *info() { return &_info; }

    void add_ref() noexcept { ++_ref_cnt; }
    int remove_ref() noexcept { return --_ref_cnt; }

    bool is_dir() const { return S_ISDIR(_info.st_mode); }

    virtual ~Node() { free(_path); }

private:
    int _ref_cnt;
    char *_path;
    struct stat64 _info;
};

struct Node_get_key
{
    typedef cxx::String Key_type;
    static Key_type key_of(Node const *n)
    { return n->path(); }
};

struct Path_avl_tree_compare
{

```

```

bool operator () (const char *l, const char *r) const
{ return strcmp(l, r) < 0; }
bool operator () (const cxx::String l, const cxx::String r) const
{
    int v = strncmp(l.start(), r.start(), cxx::min(l.len(), r.len()));
    return v < 0 || (v == 0 && l.len() < r.len());
}
};

class Pers_file : public Node
{
public:
    Pers_file(const char *name, mode_t mode)
        : Node(name, (mode & 0777) | S_IFREG) {}
    File_data const &data() const { return _data; }
    File_data &data() { return _data; }
private:
    File_data _data;
};

class Pers_dir : public Node
{
private:
    typedef cxx::Avl_tree<Node, Node_get_key, Path_avl_tree_compare> Tree;
    Tree _tree;

public:
    Pers_dir(const char *name, mode_t mode)
        : Node(name, (mode & 0777) | S_IFDIR) {}
    Ref_ptr<Node> find_path(cxx::String);
    bool add_node(Ref_ptr<Node> const &);

    typedef Tree::Const_iterator Const_iterator;
    Const_iterator begin() const { return _tree.begin(); }
    Const_iterator end() const { return _tree.end(); }
};

Ref_ptr<Node> Pers_dir::find_path(cxx::String path)
{
    return cxx::ref_ptr(_tree.find_node(path));
}

bool Pers_dir::add_node(Ref_ptr<Node> const &n)
{
    bool e = _tree.insert(n.ptr()).second;
    if (e)
        n->add_ref();
    return e;
}

class Tmpfs_dir : public Be_file
{
public:
    explicit Tmpfs_dir(Ref_ptr<Pers_dir> const &d) noexcept
        : _dir(d), _getdents_state(false) {}
    int get_entry(const char *, int, mode_t, Ref_ptr<File> *) noexcept override;
    ssize_t getdents(char *, size_t) noexcept override;
    int fstat(struct stat64 *buf) const noexcept override;
    int utime(const struct utimbuf *) noexcept override;
    int fchmod(mode_t) noexcept override;
    int mkdir(const char *, mode_t) noexcept override;
    int unlink(const char *) noexcept override;
    int rename(const char *, const char *) noexcept override;
    int faccessat(const char *, int, int) noexcept override;

private:
    int walk_path(cxx::String const &s,
                 Ref_ptr<Node> *ret, cxx::String *remaining = 0);

    Ref_ptr<Pers_dir> _dir;
    bool _getdents_state;
    Pers_dir::Const_iterator _getdents_iter;
};

class Tmpfs_file : public Be_file_pos
{
public:
    explicit Tmpfs_file(Ref_ptr<Pers_file> const &f) noexcept
        : Be_file_pos(), _file(f) {}

    off64_t size() const noexcept;
    int fstat(struct stat64 *buf) const noexcept override;
    int ftruncate(off64_t p) noexcept override;
    int ioctl(unsigned long, va_list) noexcept override;
    int utime(const struct utimbuf *) noexcept override;

```

```

    int fchmod(mode_t) noexcept override;

private:
    ssize_t preadv(const struct iovec *v, int iovcnt, off64_t p) noexcept
        override;
    ssize_t pwritev(const struct iovec *v, int iovcnt, off64_t p) noexcept
        override;
    Ref_ptr<Pers_file> _file;
};

ssize_t Tmpfs_file::preadv(const struct iovec *v, int iovcnt, off64_t p)
    noexcept
{
    if (iovcnt < 0)
        return -EINVAL;

    ssize_t sum = 0;
    for (int i = 0; i < iovcnt; ++i)
    {
        sum += _file->data().get(p, v[i].iov_len, v[i].iov_base);
        p += v[i].iov_len;
    }
    return sum;
}

ssize_t Tmpfs_file::pwritev(const struct iovec *v, int iovcnt, off64_t p)
    noexcept
{
    if (iovcnt < 0)
        return -EINVAL;

    ssize_t sum = 0;
    for (int i = 0; i < iovcnt; ++i)
    {
        sum += _file->data().put(p, v[i].iov_len, v[i].iov_base);
        p += v[i].iov_len;
    }
    return sum;
}

int Tmpfs_file::fstat(struct stat64 *buf) const noexcept
{
    _file->info()->st_size = _file->data().size();
    memcpy(buf, _file->info(), sizeof(*buf));
    return 0;
}

int Tmpfs_file::ftruncate(off64_t p) noexcept
{
    if (p < 0)
        return -EINVAL;

    if (_file->data().size(p) == 0)
        return 0;

    return -EIO; // most likely ENOSPC, but can't report that
}

off64_t Tmpfs_file::size() const noexcept
{
    return _file->data().size();
}

int
Tmpfs_file::ioctl(unsigned long v, va_list args) noexcept
{
    switch (v)
    {
        {
            case FIONREAD: // return amount of data still available
                int *available = va_arg(args, int *);
                *available = _file->data().size() - pos();
                return 0;
            };
        }
    return -EINVAL;
}

int
Tmpfs_file::utime(const struct utimbuf *times) noexcept
{
    _file->info()->st_atime = times->actime;
    _file->info()->st_mtime = times->modtime;
    return 0;
}

int
Tmpfs_file::fchmod(mode_t m) noexcept
{
    _file->info()->st_mode = m;
    return 0;
}

```

```

}

int
Tmpfs_dir::faccessat(const char *path, int mode, int) noexcept
{
    Ref_ptr<Node> node;
    cxx::String name = path;

    int err = walk_path(name, &node, &name);
    if (err < 0)
        return err;

    if (mode == F_OK) // existence check
        return 0;

    struct stat64 *stats = node->info();

    if ((mode & R_OK) && !(stats->st_mode & S_IRUSR))
        return -EACCES;

    if ((mode & W_OK) && !(stats->st_mode & S_IWUSR))
        return -EACCES;

    if ((mode & X_OK) && !(stats->st_mode & S_IXUSR))
        return -EACCES;

    return 0;
}

int
Tmpfs_dir::get_entry(const char *name, int flags, mode_t mode,
                    Ref_ptr<File> *file) noexcept
{
    Ref_ptr<Node> path;
    if (!*name)
    {
        *file = cxx::ref_ptr(this);
        return 0;
    }

    cxx::String n = name;

    int e = walk_path(n, &path, &n);

    if (e == -ENOTDIR)
        return e;

    if (!(flags & O_CREAT) && e < 0)
        return e;

    if ((flags & O_CREAT) && e == -ENOENT)
    {
        Ref_ptr<Node> node(new Pers_file(n.start(), mode));
        // when ENOENT is return, path is always a directory
        bool e = cxx::ref_ptr_static_cast<Pers_dir>(path)->add_node(node);
        if (!e)
            return -ENOMEM;
        path = node;
    }

    if (path->is_dir())
        *file = cxx::ref_ptr(new Tmpfs_dir(cxx::ref_ptr_static_cast<Pers_dir>(path)));
    else
        *file = cxx::ref_ptr(new Tmpfs_file(cxx::ref_ptr_static_cast<Pers_file>(path)));

    if (!*file)
        return -ENOMEM;

    return 0;
}

ssize_t
Tmpfs_dir::getdents(char *buf, size_t sz) noexcept
{
    struct dirent64 *d = (struct dirent64 *)buf;
    ssize_t ret = 0;

    if (!_getdents_state)
    {
        _getdents_iter = _dir->begin();
        _getdents_state = true;
    }
    else if (_getdents_iter == _dir->end())
    {
        _getdents_state = false;
        return 0;
    }

```

```

    }

    for (; _getdents_iter != _dir->end(); ++_getdents_iter)
    {
        unsigned l = strlen(_getdents_iter->path()) + 1;
        if (l > sizeof(d->d_name))
            l = sizeof(d->d_name);

        unsigned n = offsetof (struct dirent64, d_name) + l;
        n = (n + sizeof(long) - 1) & ~(sizeof(long) - 1);

        if (n > sz)
            break;

        d->d_ino = 1;
        d->d_off = 0;
        memcpy(d->d_name, _getdents_iter->path(), l);
        d->d_reclen = n;
        d->d_type = DT_REG;
        ret += n;
        sz -= n;
        d = (struct dirent64 *) ((unsigned long)d + n);
    }

    return ret;
}

int
Tmpfs_dir::fstat(struct stat64 *buf) const noexcept
{
    memcpy(buf, _dir->info(), sizeof(*buf));
    return 0;
}

int
Tmpfs_dir::utime(const struct utimbuf *times) noexcept
{
    _dir->info()->st_atime = times->actime;
    _dir->info()->st_mtime = times->modtime;
    return 0;
}

int
Tmpfs_dir::fchmod(mode_t m) noexcept
{
    _dir->info()->st_mode = m;
    return 0;
}

int
Tmpfs_dir::walk_path(cxx::String const &_s,
                    Ref_ptr<Node> *ret, cxx::String *remaining)
{
    Ref_ptr<Pers_dir> p = _dir;
    cxx::String s = _s;
    Ref_ptr<Node> n;

    while (1)
    {
        if (s.len() == 0)
        {
            *ret = p;
            return 0;
        }

        cxx::String::Index sep = s.find("/");

        if (sep - s.start() == 1 && *s.start() == '.')
        {
            s = s.substr(s.start() + 2);
            continue;
        }

        n = p->find_path(s.head(sep - s.start()));

        if (!n)
        {
            *ret = p;
            if (remaining)
                *remaining = s.head(sep - s.start());
            return -ENOENT;
        }

        if (sep == s.end())
        {
            *ret = n;

```

```

        return 0;
    }

    if (!n->is_dir())
        return -ENOTDIR;

    s = s.substr(sep + 1);

    p = cxx::ref_ptr_static_cast<Pers_dir>(n);
}

*ret = n;

return 0;
}

int
Tmpfs_dir::mkdir(const char *name, mode_t mode) noexcept
{
    Ref_ptr<Node> node = _dir;
    cxx::String p = cxx::String(name);
    cxx::String path, last = p;
    cxx::String::Index s = p.rfind("/");

    // trim /'s at the end
    while (p.len() && s == p.end() - 1)
    {
        p.len(p.len() - 1);
        s = p.rfind("/");
    }

    //printf("MKDIR '%s' p=%p %p\n", name, p.start(), s);

    if (s != p.end())
    {
        path = p.head(s);
        last = p.substr(s + 1, p.end() - s);

        int e = walk_path(path, &node);
        if (e < 0)
            return e;
    }

    if (!node->is_dir())
        return -ENOTDIR;

    // due to path walking we can end up with an empty name
    if (p.len() == 0 || p == cxx::String("."))
        return 0;

    Ref_ptr<Pers_dir> dnode = cxx::ref_ptr_static_cast<Pers_dir>(node);

    Ref_ptr<Pers_dir> dir(new Pers_dir(last.start(), mode));
    return dnode->add_node(dir) ? 0 : -EEXIST;
}

int
Tmpfs_dir::unlink(const char *name) noexcept
{
    cxx::Ref_ptr<Node> n;

    int e = walk_path(name, &n);
    if (e < 0)
        return -ENOENT;

    printf("Unimplemented (if file exists): %s(%s)\n", __func__, name);
    return -ENOMEM;
}

int
Tmpfs_dir::rename(const char *old, const char *newn) noexcept
{
    printf("Unimplemented: %s(%s, %s)\n", __func__, old, newn);
    return -ENOMEM;
}

class Tmpfs_fs : public Be_file_system
{
public:
    Tmpfs_fs() : Be_file_system("tmpfs") {}
    int mount(char const *source, unsigned long mountflags,
              void const *data, cxx::Ref_ptr<File> *dir) noexcept override
    {
        (void)mountflags;
        (void)source;
    }

```

```

    (void) data;
    *dir = cxx::ref_ptr(new Tmpfs_dir(cxx::ref_ptr(new Pers_dir("root", 0777))));
    if (!*dir)
        return -ENOMEM;
    return 0;
}
};

static Tmpfs_fs _tmpfs L4RE_VFS_FILE_SYSTEM_ATTRIBUTE;
}

```

17.26 examples/libs/shmc/prodcons.c

Simple shared memory example.

Simple shared memory example.

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
 *      economic rights: Technische Universität Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */

/*
 * This example uses shared memory between two threads, one producer, one
 * consumer.
 */

#include <l4/shmc/shmc.h>

#include <l4/util/util.h>

#include <stdio.h>
#include <string.h>
#include <pthread-l4.h>

#include <l4/sys/thread.h>
#include <l4/sys/debugger.h>
#include <l4/sys/kip.h>
#include <l4/re/env.h>

#define LOG(args...)      printf(NAME ": " args)
#define CHK(func)
do
{
    long r = (func);
    if (r)
    {
        printf(NAME ": Failure %ld (%s) at line %d.\n",
              r, l4sys_errtostr(r), __LINE__);
        return (void *)-1;
    }
} while (0)

static const char some_data[] = "Hi consumer!";

static inline l4_cap_idx_t self(void) { return pthread_l4_cap(pthread_self()); }

#define NAME "PRODUCER"
static void *thread_producer(void *d)
{
    (void) d;
    l4shmc_chunk_t p_one;
    l4shmc_signal_t s_one, s_done;
    l4shmc_area_t shmarea;
    l4_kernel_clock_t try_until;

    l4_debugger_set_object_name(self(), "producer");

    // attach this thread to the shm object
    CHK(l4shmc_attach("testshm", &shmarea));

    // add a chunk
    CHK(l4shmc_add_chunk(&shmarea, "one", 1024, &p_one));

    // add a signal

```

```

CHK(l4shmc_add_signal(&shmarea, "testshm_prod", &s_one));

CHK(l4shmc_attach_signal(&shmarea, "testshm_done", self(), &s_done));

// connect chunk and signal
CHK(l4shmc_connect_chunk_signal(&p_one, &s_one));

CHK(l4shmc_mark_client_initialized(&shmarea));

try_until = l4_kip_clock(l4re_kip()) + 10 * 1000000;

for (;;)
{
    l4_umword_t clients;
    l4shmc_get_initialized_clients(&shmarea, &clients);
    if (clients == 3UL)
        break;
    if (l4_kip_clock(l4re_kip()) >= try_until)
    {
        LOG("consumer not initialized within time\n");
        return (void *)-1;
    }
}

LOG("Ready.\n");

while (1)
{
    while (l4shmc_chunk_try_to_take(&p_one))
        printf("Uh, should not happen!\n"); //l4_thread_yield();

    memcpy(l4shmc_chunk_ptr(&p_one), some_data, sizeof(some_data));

    CHK(l4shmc_chunk_ready_sig(&p_one, sizeof(some_data)));

    LOG("Sent data.\n");

    CHK(l4shmc_wait_signal(&s_done));
}

l4_sleep_forever();
return NULL;
}

#undef NAME
#define NAME "CONSUMER"
static void *thread_consumer(void *d)
{
    (void)d;
    l4shmc_area_t shmarea;
    l4shmc_chunk_t p_one;
    l4shmc_signal_t s_one, s_done;
    l4_kernel_clock_t try_until;

    l4_debugger_set_object_name(self(), "consumer");

    // attach to shared memory area
    CHK(l4shmc_attach("testshm", &shmarea));

    // get chunk 'one'
    CHK(l4shmc_get_chunk(&shmarea, "one", &p_one));

    // add a signal
    CHK(l4shmc_add_signal(&shmarea, "testshm_done", &s_done));

    // attach signal to this thread
    CHK(l4shmc_attach_signal(&shmarea, "testshm_prod", self(), &s_one));

    // connect chunk and signal
    CHK(l4shmc_connect_chunk_signal(&p_one, &s_one));

    CHK(l4shmc_mark_client_initialized(&shmarea));

    try_until = l4_kip_clock(l4re_kip()) + 10 * 1000000;

    for (;;)
    {
        l4_umword_t clients;
        l4shmc_get_initialized_clients(&shmarea, &clients);
        if (clients == 3UL)
            break;
        if (l4_kip_clock(l4re_kip()) >= try_until)
        {
            LOG("producer not initialized within time\n");
            return (void *)-1;
        }
    }
}

```



```
    }

    LOG("Ready.\n");

    while (1)
    {
        CHK(l4shmc_wait_chunk(&p_one));

        LOG("Received from chunk one: '%s'.\n",
            (char *)l4shmc_chunk_ptr(&p_one));
        memset(l4shmc_chunk_ptr(&p_one), 0, l4shmc_chunk_size(&p_one));

        CHK(l4shmc_chunk_consumed(&p_one));

        CHK(l4shmc_trigger(&s_done));
    }

    return NULL;
}

int main(void)
{
    pthread_t one, two;
    long r;

    // create shared memory area
    if ((r = l4shmc_create("testshm")) < 0)
    {
        printf("Error %ld (%s) creating shared memory area\n",
            r, l4sys_errtostr(r));
        return 1;
    }

    // create two threads, one for producer, one for consumer
    pthread_create(&one, 0, thread_producer, 0);
    pthread_create(&two, 0, thread_consumer, 0);

    // now sleep, the two threads are doing the work
    l4_sleep_forever();

    return 0;
}
```


Index

%L4 Inter-Process Communication (IPC), [7](#)

__iface

L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >, [1328](#)

L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >, [1332](#)

__iface_list

L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >, [1328](#)

L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >, [1332](#)

__L4UTIL_THREAD_FUNC

thread.h, [2624](#)

__check_protocols__

L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >, [1329](#)

L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >, [1333](#)

__l4_kdebug_3_text

kdebug.h, [3290](#)

__l4_kdebug_op

kdebug.h, [3291](#)

__l4_kdebug_op_1

kdebug.h, [3292](#)

__l4_kdebug_text

kdebug.h, [3294](#)

__vcpu-arch.h

L4_vcpu_e_field_ids, [2571](#), [2574](#)

L4_VCPU_E_VTMR_CFG, [2571](#), [2574](#)

L4_VCPU_STATE_VERSION, [2568](#), [2571](#), [2574](#), [2577](#), [2579](#)

~Be_file_system

L4Re::Vfs::Be_file_system, [1898](#)

~H_list_item_t

cxx::H_list_item_t< ELEM_TYPE >, [936](#)

~Lock_guard

L4::Lock_guard, [1346](#)

~Reply_cap

L4::Reply_cap, [1387](#)

~S

L4::Factory::S, [1106](#)

~Unique_region

L4Re::Rm::Unique_region< T >, [1792](#)

Rm::Unique_region< T >, [2390](#)

a

L4Re::Video::Pixel_info, [1962](#)

access_once

cxx, [738](#)

Access_width

L4Re::Mmio_space, [1742](#)

acquire

L4Re::Inhibitor, [1711](#)

add

L4::lpc_svr::Timeout_queue, [1283](#)

L4::Thread::Modify_senders, [1471](#)

L4::Thread_group, [1474](#)

L4vcpu::State, [2066](#)

L4virtio::Svr::Driver_mem_list_t< DATA >, [2225](#)

Virtio_vlan_mangle, [2437](#)

add_block

L4virtio::Driver::Block_device, [2093](#)

add_image_info

L4::Debugger, [1064](#)

add_irq_status

L4virtio::Svr::Dev_config, [2202](#)

add_ku_mem

L4::Task, [1440](#)

add_monitor_port

Virtio_switch, [2433](#)

add_port

Virtio_switch, [2433](#)

add_timeout

L4::lpc_svr::Server_iface, [1273](#)

L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >, [1289](#)

add_trusted_dataspaces

L4virtio::Svr::Device_t< DATA >, [2220](#)

alien

L4::Thread::Attr, [1466](#)

align_to

L4::lpc::Msg, [766](#), [767](#)

alignment

utrace::Ring_status< SEQUENCE_TYPE >, [2419](#)

all

L4::Kip::Mem_desc, [1316](#)

alloc

cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >, [845](#)

cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >, [850](#)

cxx::List_alloc, [945](#)

cxx::Slab< Type, Slab_size, Max_free, Alloc >, [976](#)

cxx::Slab_static< Type, Slab_size, Max_free, Alloc >, [981](#)

L4::Reply_cap_alloc, [1390](#)

L4Re::Cap_alloc, [1640](#), [1641](#)

L4Re::Mem_alloc, [1734](#)

- L4Re::Util::_Cap_alloc, 1799
- L4Re::Util::Counting_cap_alloc< COUNTER-
TYPE, Dbg >, 1835
- alloc_buffer_demand
 - L4::lpc_svr::Br_manager_no_buffers, 1253
 - L4::lpc_svr::Server_iface, 1273
 - L4Re::Util::Br_manager, 1819
- alloc_descriptor
 - L4virtio::Driver::Virtqueue, 2125
- alloc_dynamic_entry
 - L4Re::Util::Names::Name_space, 1864
- alloc_fd
 - L4Re::Vfs::Fs, 1913
- alloc_max
 - cx::List_alloc, 945
- allocate
 - L4Re::Dataspace, 1653
 - L4Re::Util::Dataspace_svr, 1840
- AMD64 Virtual Registers (UTCB), 277
- amd64/l4/sys/___kip-arch.h, 2565
- amd64/l4/sys/___vcpu-arch.h, 2567, 2568
- amd64/l4/sys/arch/cache.h, 3135, 3136
- amd64/l4/sys/arch/consts.h, 3153, 3154
- amd64/l4/sys/arch/ipc.h, 3268
- amd64/l4/sys/arch/kip.h, 3457
- amd64/l4/sys/arch/l4int.h, 3314, 3315
- amd64/l4/sys/arch/platform_control.h, 3333
- amd64/l4/sys/arch/task.h, 3363
- amd64/l4/sys/arch/thread.h, 2611
- amd64/l4/sys/arch/utcb.h, 3384, 3385
- amd64/l4/sys/ktrace_events.h, 2580
- amd64/l4/sys/linkage.h, 2598, 2599
- amd64/l4/sys/segment.h, 2525, 2530
- amd64/l4/sys/vm.h, 2605
- amd64/l4/util/arch/l4_macros.h, 2608
- amd64/l4/util/arch/thread.h, 2611
- amd64/l4/util/bitops_arch.h, 2626
- amd64/l4/util/cpu.h, 2633, 2635
- amd64/l4/util/irq.h, 2774, 2775
- amd64/l4/util/mbi_argv.h, 2638, 2639
- amd64/l4/util/perform.h, 2534, 2535
- amd64/l4/util/port_io.h, 2546, 2548
- amd64/l4/util/rdtsc.h, 2554, 2556
- amd64/l4/util/spin.h, 2563, 2564
- Application and Server Building Blocks, 28
- Arch
 - L4::Kip::Mem_desc, 1314
- Arch_acpi_nvs
 - L4::Kip::Mem_desc, 1314
- Arch_acpi_tables
 - L4::Kip::Mem_desc, 1314
- Arch_sub_type_common
 - L4::Kip::Mem_desc, 1313
- ARM Virtual Registers (UTCB), 276
- arm/l4/sys/___kip-arch.h, 2565
- arm/l4/sys/___vcpu-arch.h, 2569, 2572
- arm/l4/sys/arch/cache.h, 3137, 3138
- arm/l4/sys/arch/consts.h, 3154, 3155
- arm/l4/sys/arch/ipc.h, 3269
- arm/l4/sys/arch/kip.h, 3458
- arm/l4/sys/arch/l4int.h, 3315, 3316
- arm/l4/sys/arch/platform_control.h, 3333
- arm/l4/sys/arch/task.h, 3363
- arm/l4/sys/arch/thread.h, 2611, 2612
- arm/l4/sys/arch/utcb.h, 3386, 3388
- arm/l4/sys/atomic.h, 3419
- arm/l4/sys/ktrace_events.h, 2583
- arm/l4/sys/linkage.h, 2599
- arm/l4/sys/mem_op.h, 2602, 2603
- arm/l4/sys/syscall_defs.h, 2604
- arm/l4/sys/vm.h, 2605, 2606
- arm/l4/util/arch/l4_macros.h, 2609
- arm/l4/util/arch/thread.h, 2612
- arm/l4/util/bitops_arch.h, 2629, 2630
- arm/l4/util/cpu.h, 2636
- arm/l4/util/irq.h, 2775, 2776
- arm/l4/util/mbi_argv.h, 2640, 2641
- ARM64 Virtual Registers (UTCB), 276
- arm64/l4/sys/___kip-arch.h, 2566
- arm64/l4/sys/___vcpu-arch.h, 2573, 2575
- arm64/l4/sys/arch/cache.h, 3139, 3140
- arm64/l4/sys/arch/consts.h, 3155, 3156
- arm64/l4/sys/arch/ipc.h, 3269
- arm64/l4/sys/arch/kip.h, 3458
- arm64/l4/sys/arch/l4int.h, 3316
- arm64/l4/sys/arch/platform_control.h, 3334
- arm64/l4/sys/arch/task.h, 3363
- arm64/l4/sys/arch/thread.h, 2612, 2613
- arm64/l4/sys/arch/utcb.h, 3389, 3390
- arm64/l4/sys/ktrace_events.h, 2586
- arm64/l4/sys/linkage.h, 2600
- arm64/l4/sys/vm.h, 2607
- arm_smccc.h
 - l4_arm_smccc_call, 3133
- assert.h
 - l4_assert, 3415
- assign_dma_domain
 - L4vbus::Vbus, 2059, 2060
- associate
 - L4Re::Dma_space, 1674
- AT_BASE
 - ELF binary format, 702
- AT_EGID
 - ELF binary format, 703
- AT_ENTRY
 - ELF binary format, 702
- AT_EUID
 - ELF binary format, 703
- AT_EXECFD
 - ELF binary format, 702
- AT_FLAGS
 - ELF binary format, 702
- AT_GID
 - ELF binary format, 703
- AT_IGNORE
 - ELF binary format, 702

- AT_L4_AUX
 - ELF binary format, [703](#)
- AT_L4_ENV
 - ELF binary format, [703](#)
- AT_L4_KIP
 - ELF binary format, [703](#)
- AT_NOTELF
 - ELF binary format, [702](#)
- AT_NULL
 - ELF binary format, [702](#)
- AT_PAGESZ
 - ELF binary format, [702](#)
- AT_PHDR
 - ELF binary format, [702](#)
- AT_PHEMT
 - ELF binary format, [702](#)
- AT_PHNUM
 - ELF binary format, [702](#)
- AT_UID
 - ELF binary format, [702](#)
- Atomic Instructions, [669](#)
 - l4util_add16, [671](#)
 - l4util_add16_res, [671](#)
 - l4util_add32, [671](#)
 - l4util_add32_res, [672](#)
 - l4util_add8, [672](#)
 - l4util_add8_res, [672](#)
 - l4util_and16, [673](#)
 - l4util_and16_res, [673](#)
 - l4util_and32, [673](#)
 - l4util_and32_res, [674](#)
 - l4util_and8, [674](#)
 - l4util_and8_res, [674](#)
 - l4util_atomic_add, [675](#)
 - l4util_atomic_inc, [675](#)
 - l4util_cmpxchg, [675](#)
 - l4util_cmpxchg16, [676](#)
 - l4util_cmpxchg32, [677](#)
 - l4util_cmpxchg8, [677](#)
 - l4util_dec16, [678](#)
 - l4util_dec16_res, [678](#)
 - l4util_dec32, [678](#)
 - l4util_dec32_res, [679](#)
 - l4util_dec8, [679](#)
 - l4util_dec8_res, [679](#)
 - l4util_inc16, [679](#)
 - l4util_inc16_res, [680](#)
 - l4util_inc32, [680](#)
 - l4util_inc32_res, [680](#)
 - l4util_inc8, [681](#)
 - l4util_inc8_res, [681](#)
 - l4util_or16, [681](#)
 - l4util_or16_res, [682](#)
 - l4util_or32, [682](#)
 - l4util_or32_res, [682](#)
 - l4util_or8, [682](#)
 - l4util_or8_res, [683](#)
 - l4util_sub16, [683](#)
 - l4util_sub16_res, [683](#)
 - l4util_sub32, [684](#)
 - l4util_sub32_res, [684](#)
 - l4util_sub8, [684](#)
 - l4util_sub8_res, [685](#)
 - l4util_xchg, [685](#)
 - l4util_xchg16, [685](#)
 - l4util_xchg32, [686](#)
 - l4util_xchg8, [686](#)
- atomic_clear_bit
 - cxx::Bitmap_base, [871](#)
 - L4Re::Util::Bitmap_base, [1808](#)
- atomic_get_and_clear
 - cxx::Bitmap_base, [871](#)
 - L4Re::Util::Bitmap_base, [1808](#)
- atomic_get_and_set
 - cxx::Bitmap_base, [872](#)
 - L4Re::Util::Bitmap_base, [1808](#)
- atomic_set_bit
 - cxx::Bitmap_base, [872](#)
 - L4Re::Util::Bitmap_base, [1809](#)
- attach
 - L4Re::Rm, [1771](#), [1773](#)
 - L4Re::Util::Event_buffer_t< PAYLOAD >, [1852](#)
 - Rm, [2374](#), [2375](#)
- Attach_flags
 - L4Re::Rm::F, [1787](#)
 - Rm::F, [2385](#)
- Attach_mask
 - L4Re::Rm::F, [1787](#)
 - Rm::F, [2385](#)
- Attr
 - L4::Thread::Attr, [1466](#)
- Attribute
 - L4Re::Dma_space, [1672](#)
- Attributes
 - L4Re::Dma_space, [1672](#)
- Attributes and additional permissions for object send items, [235](#)
 - L4_FPAGE_C_IPCGATE_SVR, [236](#)
- atype
 - Elf32_Auxv, [1001](#)
 - Elf64_Auxv, [1011](#)
- Auxiliary data, [596](#)
- avail
 - cxx::List_alloc, [946](#)
- avail_align
 - L4virtio::Virtqueue, [2324](#)
- avail_size
 - L4virtio::Virtqueue, [2324](#)
- Avl_map
 - cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >, [826](#)
- Avl_set_iter
 - cxx::Bits::Avl_set_iter< Node, Key, Node_op >, [886](#)
- ax
 - l4_vcpu_regs_t, [1603](#)

- b
 - L4Re::Video::Pixel_info, [1962](#), [1963](#)
- backtrace.h
 - l4util_backtrace, [3429](#)
- Bad_address
 - L4virtio::Svr::Bad_descriptor, [2134](#)
- Bad_descriptor
 - L4virtio::Svr::Bad_descriptor, [2134](#)
- Bad_flags
 - L4virtio::Svr::Bad_descriptor, [2134](#)
- Bad_next
 - L4virtio::Svr::Bad_descriptor, [2134](#)
- Bad_rights
 - L4virtio::Svr::Bad_descriptor, [2134](#)
- Bad_size
 - L4virtio::Svr::Bad_descriptor, [2134](#)
- Base API, [149](#)
- Base_avl_set
 - cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, [891](#)
- Basic Macros, [152](#)
 - l4_align_stack_for_direct_fncall, [156](#)
 - L4_EXPORT, [154](#)
 - L4_HIDDEN, [154](#)
 - l4_infinite_loop, [157](#)
 - L4_NOTHROW, [155](#)
- bdr
 - l4_buf_regs_t, [1575](#)
- Be_file_system
 - L4Re::Vfs::Be_file_system, [1898](#)
- begin
 - cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, [892](#)
 - cxx::Bits::Bst< Node, Get_key, Compare >, [906](#), [907](#)
- Bidirectional
 - L4Re::Dma_space, [1673](#)
- bind
 - L4::lcu, [1115](#)
 - L4::lommu, [1132](#)
 - L4::Thread::Attr, [1466](#)
- bind_notification_irq
 - L4virtio::Driver::Device, [2101](#)
- bind_rx_notification_irq
 - L4virtio::Driver::Virtio_net_device, [2115](#)
- bind_snd_destination
 - L4::Rcv_endpoint, [1379](#)
- bind_thread
 - L4::Rcv_endpoint, [1379](#)
- bind_vcpu
 - L4::lrc, [1296](#)
- bit
 - cxx::Bitmap_base, [873](#), [874](#)
 - L4Re::Util::Bitmap_base, [1809](#)
- Bit Manipulation, [687](#)
 - l4util_bsf, [688](#)
 - l4util_bsr, [689](#)
 - l4util_btc, [689](#)
 - l4util_btr, [689](#)
 - l4util_bts, [690](#)
 - l4util_clear_bit, [691](#)
 - l4util_complement_bit, [691](#)
 - l4util_find_first_set_bit, [692](#)
 - l4util_find_first_zero_bit, [692](#)
 - l4util_next_power2, [692](#)
 - l4util_set_bit, [693](#)
 - l4util_test_bit, [693](#)
- bit_index
 - cxx::Bitmap_base, [875](#)
 - L4Re::Util::Bitmap_base, [1810](#)
- Bits
 - cxx::Bitfield< T, LSB, MSB >, [855](#)
- bits_per_pixel
 - L4Re::Video::Pixel_info, [1963](#)
- Bits_type
 - cxx::Bitfield< T, LSB, MSB >, [854](#)
- Block_dev_base
 - L4virtio::Svr::Block_dev_base< Ds_data >, [2139](#)
- Bootloader
 - L4::Kip::Mem_desc, [1314](#)
- Bootstrap, the %L4 kernel bootstrapper, [91](#)
- bp
 - l4_vcpu_regs_t, [1603](#)
- Br_bytes
 - L4::lpc::Msg, [766](#)
- buf
 - L4Re::Util::Event_buffer_t< PAYLOAD >, [1852](#)
- buf_cp_in
 - L4::lpc, [758](#)
- buf_cp_out
 - L4::lpc, [758](#)
- buf_in
 - L4::lpc, [759](#)
- Buffer, [819](#)
- buffer
 - L4Re::Util::Event_t< PAYLOAD >, [1857](#)
- Buffer Registers (BRs), [273](#)
 - l4_bdr, [274](#)
 - L4_BDR_IO_SHIFT, [274](#)
 - L4_BDR_MEM_SHIFT, [274](#)
 - L4_BDR_OBJ_SHIFT, [274](#)
 - l4_buffer_desc_consts_t, [274](#)
- Bufferable
 - L4Re::Dataspace::F, [1664](#)
- Buffered
 - L4::lpc::Snd_fpage, [1223](#)
- bus_cap
 - L4vbus::Device, [2010](#)
- bx
 - l4_vcpu_regs_t, [1603](#)
- bytes_per_pixel
 - L4Re::Video::Pixel_info, [1964](#)
- c
 - L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >, [1329](#)

- L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >, 1333
- C++ Exceptions, 602
- C++ IPC Interface Definition, 193
- C_bits
 - cxx::Bitmap_base, 871
 - L4Re::Util::Bitmap_base, 1808
- Cache Consistency, 194
 - l4_cache_clean_data, 195
 - l4_cache_coherent, 196
 - l4_cache_dma_coherent, 196
 - l4_cache_flush_data, 197
 - l4_cache_inv_data, 197
- Cache_buffered
 - L4Re::Rm::F, 1788
 - Rm::F, 2386
- Cache_normal
 - L4Re::Rm::F, 1788
 - Rm::F, 2386
- Cache_uncached
 - L4Re::Rm::F, 1788
 - Rm::F, 2386
- Cacheable
 - L4Re::Dataspace::F, 1664
- Cached
 - L4::lpc::Snd_fpage, 1223
- Cacheopt
 - L4::lpc::Snd_fpage, 1223
- Caching_mask
 - L4Re::Dataspace::F, 1664
 - L4Re::Rm::F, 1788
 - Rm::F, 2386
- Caching_shift
 - L4Re::Dataspace::F, 1663
 - L4Re::Rm, 1771
 - Rm, 2374
- call
 - L4::Arm_smccc, 1028
 - L4::lpc::lostream, 1158
- Cap
 - L4::Cap< T >, 1042
 - L4::lpc::Cap< T >, 1149
- cap
 - L4::Cap_base, 1049
 - L4::Invalid_capability, 1126
 - L4::Kobject, 1323
- cap_alloc
 - L4Re Capability API, 590
- Cap_base
 - L4::Cap_base, 1047, 1048
- cap_cast
 - L4, 748, 749
- cap_dynamic_cast
 - L4, 750
- cap_equal
 - L4::Task, 1441
- Cap_mask
 - L4::lpc::Cap< T >, 1149
- cap_received
 - L4::lpc::Snd_fpage, 1226
- cap_reinterpret_cast
 - L4, 752, 753
- Cap_type
 - L4::Cap_base, 1047
- cap_valid
 - L4::Task, 1442
- Capabilities, 445
 - L4_BASE_ARM_SMCCC_CAP, 447
 - L4_BASE_CAPS_LAST, 447
 - L4_BASE_DEBUGGER_CAP, 447
 - L4_BASE_FACTORY_CAP, 447
 - L4_BASE_ICU_CAP, 447
 - L4_BASE_IOMMU_CAP, 447
 - L4_BASE_LOG_CAP, 447
 - L4_BASE_PAGER_CAP, 447
 - L4_BASE_SCHEDULER_CAP, 447
 - L4_BASE_TASK_CAP, 447
 - L4_BASE_THREAD_CAP, 447
 - l4_cap_idx_t, 446
 - l4_capability_equal, 447
 - l4_default_caps_t, 447
 - l4_is_invalid_cap, 448
 - l4_is_valid_cap, 448
- Capabilities and Naming, 23
- capability
 - L4_DISABLE_COPY, 3147
- Capability allocator, 570
 - l4re_util_cap_last, 571
- Caps
 - L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS >, 1488
- caps
 - l4re_env_t, 1985
- cast
 - L4vcpu::Vcpu, 2071
- cfg_read
 - L4vbus::Pci_dev, 2041
 - L4vbus::Pci_host_bridge, 2047
- cfg_write
 - L4vbus::Pci_dev, 2041
 - L4vbus::Pci_host_bridge, 2047
- change_mode
 - L4::Uart, 1547
 - L4::Uart_apb, 1555
- change_queue_config
 - L4virtio::Svr::Dev_config, 2202
- char_avail
 - L4::Uart, 1547
 - L4::Uart_apb, 1555
- check_castable_from
 - L4::Cap< T >, 1043
- check_convertible_from
 - L4::Cap< T >, 1043
- check_items
 - utrace::Ring_buffer< SEQUENCE_TYPE >, 2398
- check_mask

- utrace::Ring_buffer< SEQUENCE_TYPE >, 2399
- check_ports
 - Virtio_switch, 2434
- check_ready
 - L4Re::Vfs::Be_file, 1894
 - L4Re::Vfs::Generic_file, 1918
- check_size
 - L4::lpc::Msg, 767, 768
- chkcapi
 - L4Re, 779
- chkipc
 - L4Re, 780
- chksys
 - L4Re, 781–783
- Chunks, 613
 - l4shmc_add_chunk, 614
 - l4shmc_chunk_capacity, 614
 - l4shmc_chunk_ptr, 615
 - l4shmc_chunk_signal, 615
 - l4shmc_get_chunk, 616
 - l4shmc_get_chunk_to, 616
 - l4shmc_iterate_chunk, 617
- clamp
 - Small C++ Template Library, 641
- Class
 - L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >, 1329
 - L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >, 1332
- clear
 - cxx::Bits::Basic_list< POLICY >, 902
 - L4::Types::Flags< BITS_ENUM, UNDERLYING >, 1524
 - L4drivers::Register_tmpl< BITS, BLOCK >, 1631
 - L4Re::Dataspace, 1654
 - L4Re::Util::Dataspace_svr, 1841
 - L4vcpu::State, 2066
- clear_bit
 - cxx::Bitmap_base, 876
 - L4Re::Util::Bitmap_base, 1810
- Coherent
 - L4Re::Dma_space, 1673
- Color_component
 - L4Re::Video::Color_component, 1942
- Com_error
 - L4::Com_error, 1060
- Comfortable Command Line Parsing, 721
 - parse_cmdline, 721
- Compound
 - L4::lpc::Snd_fpage, 1224
- config_get
 - L4vbus::Gpio_pin, 2028
- config_pad
 - L4vbus::Gpio_module, 2020
 - L4vbus::Gpio_pin, 2028
- config_pull
 - L4vbus::Gpio_pin, 2029
- config_queue
 - L4virtio::Device, 2085
 - L4virtio::Driver::Device, 2102
- Cons, the Console Multiplexer, 85
- Conservative
 - utrace::Ring_buffer_consumer_raw< SEQUENCE_TYPE, ITEM_TYPE, GENERATION_PTR >, 2409
- Console API, 592
- console_multiport_bfm_t
 - L4virtio::Svr::Console::Features, 2171
- Console_port
 - L4virtio::Svr::Console::Control_message, 2147
- console_size_bfm_t
 - L4virtio::Svr::Console::Features, 2171
- consts.h
 - L4_CAP_SIZE, 3160
 - L4_REPLY_CAP_BIT, 3160
 - L4_SYSF_CALL, 3161
 - L4_SYSF_OPEN_WAIT, 3161
 - L4_SYSF_RECV, 3161
 - L4_SYSF_REPLY, 3161
 - L4_SYSF_REPLY_AND_WAIT, 3162
 - L4_SYSF_SEND, 3162
 - L4_SYSF_SEND_AND_WAIT, 3162
 - L4_SYSF_WAIT, 3162
- consumed
 - L4virtio::Svr::Virtqueue, 2310
- Consumer, 621, 630
 - l4shmc_chunk_consumed, 622
 - l4shmc_chunk_size, 622
 - l4shmc_chunk_try_to_take_for_reading, 622
 - l4shmc_enable_chunk, 623
 - l4shmc_enable_signal, 631
 - l4shmc_is_chunk_ready, 623
 - l4shmc_wait_any, 631
 - l4shmc_wait_any_to, 631
 - l4shmc_wait_any_try, 632
 - l4shmc_wait_chunk, 624
 - l4shmc_wait_chunk_to, 624
 - l4shmc_wait_chunk_try, 625
 - l4shmc_wait_signal, 632
 - l4shmc_wait_signal_to, 633
 - l4shmc_wait_signal_try, 633
- contains
 - L4virtio::Svr::Driver_mem_region_t< DATA >, 2232
- Continue
 - L4::lpc::Snd_fpage, 1223
- Continuous
 - L4Re::Mem_alloc, 1734
- contrib/libio-io/l4/io/io.h, 2643, 2645
- contrib/libio-io/l4/io/types.h, 3378
- control
 - L4::Thread, 1452
- Conventional
 - L4::Kip::Mem_desc, 1314
- copy
 - L4::Cap< T >, 1043
 - L4::Cap_base, 1050

- L4Re::Util::Dataspace_svr, [1841](#)
- copy_in
 - L4Re::Dataspace, [1655](#)
- copy_pkt
 - Virtio_vlan_mangle, [2437](#)
- copy_receive_cap
 - L4Re::Util::Names::Name_space, [1865](#)
- copy_to
 - L4virtio::Svr::Data_buffer, [2196](#)
- count
 - L4::Kip::Mem_desc, [1316](#), [1317](#)
- Counting_cap_alloc
 - L4Re::Util::Counting_cap_alloc< COUNTER-
TYPE, Dbg >, [1834](#)
- CPU related functions, [653](#)
 - l4util_cpu_capabilities, [654](#)
 - l4util_cpu_capabilities_nocheck, [654](#)
 - l4util_cpu_has_cpuid, [655](#)
- cpu_allow_shutdown
 - L4::Platform_control, [1364](#)
- cpu_disable
 - L4::Platform_control, [1364](#)
- cpu_enable
 - L4::Platform_control, [1365](#)
- create
 - L4::Factory, [1095](#), [1096](#)
- create_buffer
 - L4Re::Video::Goos, [1950](#)
- create_factory
 - L4::Factory, [1097](#)
- create_gate
 - L4::Factory, [1099](#)
- create_task
 - L4::Factory, [1100](#)
- create_thread_group
 - L4::Factory, [1101](#)
- create_view
 - L4Re::Video::Goos, [1950](#)
- cur_buf
 - Net_transfer, [2368](#)
- current_flags
 - L4virtio::Svr::Request_processor, [2237](#)
- cx
 - l4_vcpu_regs_t, [1603](#)
- cxx, [735](#)
 - access_once, [738](#)
 - gcd, [738](#)
 - lcm, [739](#)
 - write_now, [740](#)
- cxx::arith::Ld< V >, [821](#)
- cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE,
ALLOC >, [822](#)
 - Avl_map, [826](#)
 - emplace, [826](#)
 - insert, [826](#)
 - operator[], [827](#), [828](#)
- cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC >, [828](#)
- cxx::Avl_tree< Node, Get_key, Compare >, [832](#)
 - insert, [838](#)
 - Iterator, [838](#)
 - remove, [838](#)
- cxx::Avl_tree_node, [840](#)
- cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc
>, [842](#)
 - alloc, [845](#)
 - free, [845](#)
 - free_objects, [845](#)
 - max_free_slabs, [845](#)
 - object_size, [845](#)
 - objects_per_slab, [845](#)
 - slab_size, [845](#)
 - total_objects, [846](#)
- cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc
>::Slab_i, [846](#)
- cxx::Base_slab_static< Obj_size, Slab_size, Max_free,
Alloc >, [847](#)
 - alloc, [850](#)
 - free, [850](#)
 - free_objects, [851](#)
 - max_free_slabs, [850](#)
 - object_size, [850](#)
 - objects_per_slab, [850](#)
 - slab_size, [850](#)
 - total_objects, [851](#)
- cxx::Bitfield< T, LSB, MSB >, [852](#)
 - Bits, [855](#)
 - Bits_type, [854](#)
 - get, [855](#)
 - get_unshifted, [855](#)
 - Lsb, [855](#)
 - Msb, [855](#)
 - set, [855](#)
 - set_dirty, [856](#)
 - set_unshifted, [857](#)
 - set_unshifted_dirty, [857](#)
 - Shift_type, [854](#)
 - val, [858](#)
 - val_dirty, [858](#)
 - val_unshifted, [859](#)
- cxx::Bitfield< T, LSB, MSB >::Value< TT >, [860](#)
- cxx::Bitfield< T, LSB, MSB >::Value_base< TT >, [861](#)
- cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT >,
[862](#)
- cxx::Bitmap< BITS >, [864](#)
 - scan_zero, [867](#)
- cxx::Bitmap_base, [868](#)
 - atomic_clear_bit, [871](#)
 - atomic_get_and_clear, [871](#)
 - atomic_get_and_set, [872](#)
 - atomic_set_bit, [872](#)
 - bit, [873](#), [874](#)
 - bit_index, [875](#)
 - C_bits, [871](#)
 - clear_bit, [876](#)
 - operator[], [876](#), [877](#)
 - scan_zero, [878](#)

set_bit, 879
 W_bits, 871
 word_index, 879
 cxx::Bitmap_base::Bit, 881
 cxx::Bitmap_base::Char< BITS >, 881
 cxx::Bitmap_base::Word< BITS >, 882
 cxx::Bits, 741
 cxx::Bits::Avl_map_get_key< KEY_TYPE >, 883
 cxx::Bits::Avl_set_get_key< KEY_TYPE >, 884
 cxx::Bits::Avl_set_iter< Node, Key, Node_op >, 884
 Avl_set_iter, 886
 operator->, 887
 operator*, 887
 cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, AL-
 LOC, GET_KEY >, 888
 Base_avl_set, 891
 begin, 892
 E_exist, 891
 E_inval, 891
 E_noent, 891
 E_nomem, 891
 emplace, 892
 end, 893
 erase, 894
 find_node, 894
 insert, 894
 lower_bound_node, 895
 operator=, 896
 rbegin, 896
 remove, 896
 rend, 897
 cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, AL-
 LOC, GET_KEY >::Node, 898
 operator->, 899
 operator*, 899
 valid, 899
 cxx::Bits::Basic_list< POLICY >, 900
 clear, 902
 iter, 902
 cxx::Bits::Bst< Node, Get_key, Compare >, 903
 begin, 906, 907
 dir, 907, 908
 end, 909
 find, 910
 find_node, 910
 lower_bound_node, 911
 rbegin, 912
 remove_all, 913
 remove_tree, 914
 rend, 915
 cxx::Bits::Bst_node, 916
 cxx::Bits::Direction, 918
 Direction_e, 919
 L, 919
 N, 919
 operator!, 920
 R, 919
 cxx::Bits::Smart_ptr_list< ITEM >, 920
 pop_front, 923
 cxx::Bits::Smart_ptr_list_item< T, STORE_T >, 923
 cxx::Elide_dtor< T >, 925
 cxx::Error, 926
 cxx::H_list< T, POLICY >, 926
 erase, 930
 insert, 930
 insert_after, 931
 insert_before, 931
 iter, 932
 pop_front, 932
 remove, 933
 replace, 933
 cxx::H_list_item_t< ELEM_TYPE >, 934
 ~H_list_item_t, 936
 H_list_item_t, 936
 cxx::H_list_t< T >, 937
 cxx::List< D, Alloc >, 941
 operator[], 942
 cxx::List< D, Alloc >::Iter, 943
 cxx::List_alloc, 944
 alloc, 945
 alloc_max, 945
 avail, 946
 free, 947
 List_alloc, 945
 cxx::List_item, 948
 push_back, 949
 push_front, 949
 remove, 950
 cxx::List_item::Iter, 951
 cxx::List_item::T_iter< T, Poly >, 953
 cxx::Lt_functor< Obj >, 955
 cxx::New_allocator< _Type >, 955
 cxx::Nothrow, 957
 cxx::Pair< First, Second >, 957
 Pair, 960
 cxx::Pair_first_compare< Cmp, Typ >, 960
 operator(), 962
 Pair_first_compare, 961
 cxx::Ref_obj_list_item< T >, 962
 cxx::Ref_ptr< T, CNT >, 964
 get, 968
 ptr, 968
 Ref_ptr, 967
 release, 968
 cxx::Result< T >, 969
 Result, 969
 cxx::S_list< T, POLICY >, 970
 pop_front, 973
 cxx::Slab< Type, Slab_size, Max_free, Alloc >, 973
 alloc, 976
 free, 977
 cxx::Slab_static< Type, Slab_size, Max_free, Alloc >,
 978
 alloc, 981
 cxx::static_vector< T, IDX >, 982
 cxx::String, 983

- find, [987](#)
- from_dec, [988](#)
- from_hex, [989](#)
- starts_with, [990](#)
- String, [987](#)
- cxx::Weak_ref< T >, [991](#)
- cxx::Weak_ref_base, [993](#)
- cxx::Weak_ref_base::List, [996](#)
- d_tag
 - Elf32_Dyn, [1002](#)
 - Elf64_Dyn, [1012](#)
- data
 - L4::lpc::Varg, [1233](#)
- Data_buffer
 - L4virtio::Svr::Data_buffer, [2195](#)
- data_size
 - L4virtio::Svr::Block_request< Ds_data >, [2145](#)
- data_space
 - L4Re::Vfs::Be_file, [1894](#)
 - L4Re::Vfs::Regular_file, [1931](#)
- Dataspace interface, [526](#)
 - l4re_ds_allocate, [528](#)
 - l4re_ds_clear, [528](#)
 - l4re_ds_copy_in, [529](#)
 - L4RE_DS_F_BUFFERABLE, [527](#)
 - L4RE_DS_F_CACHEABLE, [527](#)
 - L4RE_DS_F_CACHING_MASK, [527](#)
 - L4RE_DS_F_CACHING_SHIFT, [527](#)
 - L4RE_DS_F_NORMAL, [527](#)
 - L4RE_DS_F_UNCACHEABLE, [527](#)
 - l4re_ds_flags, [529](#)
 - l4re_ds_info, [530](#)
 - l4re_ds_map_flags, [527](#)
 - l4re_ds_map_info, [530](#)
 - l4re_ds_size, [531](#)
- dbg_events
 - L4Re::Env, [1681](#)
- debug
 - L4Re::Debug_obj, [1668](#)
- Debug interface, [532](#)
 - l4re_debug_obj_debug, [532](#)
- debugger.h
 - L4_DEBUGGER_GLOBAL_ID_GET_NAME_OP, [3243](#)
 - L4_DEBUGGER_KOBJ_GET_GLOBAL_ID_OP, [3243](#)
 - L4_DEBUGGER_KOBJ_GET_NAME_OP, [3243](#)
 - L4_DEBUGGER_KOBJ_PTR_GET_GLOBAL_ID_OP, [3243](#)
 - L4_DEBUGGER_KOBJ_SET_NAME_OP, [3243](#)
 - L4_DEBUGGER_LOG_QUERY_NAME_OP, [3243](#)
 - L4_DEBUGGER_LOG_QUERY_TYPEID_OP, [3243](#)
 - L4_DEBUGGER_LOG_SWITCH_OP, [3243](#)
 - L4_DEBUGGER_OBJ_INFO_OP, [3243](#)
 - L4_DEBUGGER_TASK_ADD_IMAGE_INFO_OP, [3243](#)
- Debugging API, [593](#)
- dec
 - L4Re::Util::Counter< COUNTER >, [1830](#)
 - L4Re::Util::Counter_atomic< COUNTER >, [1832](#)
- dec_refcnt
 - L4::Kobject, [1325](#)
- declval
 - L4::Types, [774](#)
- Dedicated
 - L4::Kip::Mem_desc, [1314](#)
- delete_buffer
 - L4Re::Video::Goos, [1951](#)
- delete_obj
 - L4::Task, [1443](#)
- delete_view
 - L4Re::Video::Goos, [1952](#)
- Demand
 - L4::Kobject_typeid< T >, [1337](#)
 - L4::Type_info::Demand, [1484](#)
- demand
 - L4::Kobject_typeid< T >, [1338](#)
 - L4::Kobject_typeid< void >, [1341](#)
- Deprecated List, [95](#)
- dequeue
 - utrace::Ring_buffer_consumer< SEQUENCE_TYPE, ITEM_TYPE, GENERATION_PTR >, [2404](#)
 - utrace::Tracebuffer, [2421](#)
- desc
 - L4virtio::Driver::Virtqueue, [2125](#)
 - L4virtio::Svr::Virtqueue, [2311](#)
 - L4virtio::Svr::Virtqueue::Head_desc, [2319](#)
- desc_align
 - L4virtio::Virtqueue, [2326](#)
- desc_avail
 - L4virtio::Svr::Virtqueue, [2312](#)
- desc_size
 - L4virtio::Virtqueue, [2326](#)
- detach
 - L4::lirq, [1297](#)
 - L4Re::Rm, [1775–1777](#)
 - L4Re::Util::Event_buffer_t< PAYLOAD >, [1852](#)
 - Rm, [2377](#), [2378](#)
- Detach_again
 - L4Re::Rm, [1771](#)
 - Rm, [2374](#)
- Detach_exact
 - L4Re::Rm, [1770](#)
 - Rm, [2373](#)
- Detach_flags
 - L4Re::Rm, [1770](#)
 - Rm, [2373](#)
- Detach_free
 - L4Re::Rm::F, [1788](#)
 - Rm::F, [2385](#)
- Detach_keep
 - L4Re::Rm, [1770](#)
 - Rm, [2373](#)
- Detach_overlap
 - L4Re::Rm, [1770](#)

- Rm, [2373](#)
- Detach_result
 - L4Re::Rm, [1770](#)
 - Rm, [2373](#)
- Detached_ds
 - L4Re::Rm, [1771](#)
 - Rm, [2374](#)
- Dev_config
 - L4virtio::Svr::Dev_config, [2200](#), [2201](#)
- dev_handle
 - L4vbus::Device, [2011](#)
- Device
 - L4vbus::Device, [2010](#)
 - L4virtio::Svr::Console::Device, [2154](#), [2155](#)
- device
 - L4vbus::Device, [2011](#)
- Device_add
 - L4virtio::Svr::Console::Control_message, [2147](#)
- device_by_hid
 - L4vbus::Device, [2012](#)
- device_config
 - L4virtio::Device, [2085](#)
- device_error
 - L4virtio::Svr::Device_t< DATA >, [2220](#)
- device_notification_irq
 - L4virtio::Device, [2086](#)
- device_notify_irq
 - L4virtio::Svr::Device_t< DATA >, [2220](#)
- Device_ready
 - L4virtio::Svr::Console::Control_message, [2147](#)
- Device_remove
 - L4virtio::Svr::Console::Control_message, [2147](#)
- DF_1_CONFALT
 - ELF binary format, [704](#)
- DF_1_DIRECT
 - ELF binary format, [704](#)
- DF_1_DISPRELDNE
 - ELF binary format, [704](#)
- DF_1_DISPRELPND
 - ELF binary format, [704](#)
- DF_1_ENDFILTEE
 - ELF binary format, [704](#)
- DF_1_GLOBAL
 - ELF binary format, [704](#)
- DF_1_GROUP
 - ELF binary format, [704](#)
- DF_1_INITFIRST
 - ELF binary format, [704](#)
- DF_1_INTERPOSE
 - ELF binary format, [704](#)
- DF_1_LOADFLTR
 - ELF binary format, [704](#)
- DF_1_NODEFLIB
 - ELF binary format, [704](#)
- DF_1_NODELETE
 - ELF binary format, [704](#)
- DF_1_NODUMP
 - ELF binary format, [704](#)
- DF_1_NOOPEN
 - ELF binary format, [704](#)
- DF_1_NOW
 - ELF binary format, [704](#)
- DF_1_ORIGIN
 - ELF binary format, [704](#)
- DF_BIND_NOW
 - ELF binary format, [705](#)
- DF_ORIGIN
 - ELF binary format, [705](#)
- DF_P1_GROUPPERM
 - ELF binary format, [704](#)
- DF_P1_LAZYLOAD
 - ELF binary format, [704](#)
- DF_STATIC_TLS
 - ELF binary format, [705](#)
- DF_SYMBOLIC
 - ELF binary format, [705](#)
- DF_TEXTREL
 - ELF binary format, [705](#)
- dfl1
 - l4_vm_vmx_vcpu_infos_t, [1613](#)
- di
 - l4_vcpu_regs_t, [1603](#)
- dir
 - cxx::Bits::Bst< Node, Get_key, Compare >, [907](#), [908](#)
- Direction
 - L4Re::Dma_space, [1673](#)
- Direction_e
 - cxx::Bits::Direction, [919](#)
- disable
 - L4virtio::Virtqueue, [2327](#)
- disable_notify
 - L4virtio::Svr::Virtqueue, [2313](#)
- disassociate
 - L4Re::Dma_space, [1674](#)
- dispatch
 - L4::Basic_registry, [1033](#)
 - L4::Epiface, [1078](#)
 - L4::Epiface_t< Derived, IFACE, BASE, bool >, [1085](#)
 - L4::Irqep_t< Derived, BASE, bool >, [1310](#)
 - L4::Server_object, [1420](#), [1421](#)
- dispatch_meta_request
 - L4::Server_object_t< IFACE, BASE >, [1426](#)
- DMA space, [286](#)
- DMA Space Interface, [533](#)
 - l4re_dma_space_associate, [534](#)
 - l4re_dma_space_disassociate, [535](#)
 - l4re_dma_space_map, [535](#)
 - l4re_dma_space_t, [534](#)
 - l4re_dma_space_unmap, [536](#)
- dma_space.h
 - L4RE_DMA_SPACE_BIDIRECTIONAL, [2905](#)
 - L4RE_DMA_SPACE_COHERENT, [2905](#)
 - l4re_dma_space_direction, [2905](#)
 - L4RE_DMA_SPACE_FROM_DEVICE, [2905](#)

- L4RE_DMA_SPACE_NONE, [2905](#)
- L4RE_DMA_SPACE_PHYS_SPACE, [2905](#)
- l4re_dma_space_space_attrs, [2905](#)
- L4RE_DMA_SPACE_TO_DEVICE, [2905](#)
- done
 - L4virtio::Svr::Data_buffer, [2196](#)
 - Net_transfer, [2368](#)
- down
 - L4::Semaphore, [1409](#)
- driver_acknowledge
 - L4virtio::Driver::Device, [2102](#)
- driver_connect
 - L4virtio::Driver::Device, [2103](#)
- Driver_mem_region_t
 - L4virtio::Svr::Driver_mem_region_t< DATA >, [2232](#)
- Drop_policy
 - utrace::Ring_buffer_consumer_raw< SEQUENCE_TYPE, PLTRELSZ, ITEM_TYPE, GENERATION_PTR >, [2409](#)
- drop_requests
 - L4virtio_port, [2362](#)
 - Virtio_net_request, [2429](#)
- drv_base
 - L4virtio::Svr::Driver_mem_region_t< DATA >, [2233](#)
- ds
 - L4virtio::Svr::Dev_config, [2203](#)
 - L4virtio::Svr::Driver_mem_region_t< DATA >, [2233](#)
- Ds_map_mask
 - L4Re::Rm::F, [1788](#)
 - Rm::F, [2386](#)
- ds_offset
 - L4virtio::Svr::Driver_mem_region_t< DATA >, [2233](#)
- DT_BIND_NOW
 - ELF binary format, [706](#)
- DT_DEBUG
 - ELF binary format, [706](#)
- DT_ENCODING
 - ELF binary format, [706](#)
- DT_FINI
 - ELF binary format, [705](#)
- DT_FINI_ARRAY
 - ELF binary format, [706](#)
- DT_FINI_ARRAYSZ
 - ELF binary format, [706](#)
- DT_FLAGS
 - ELF binary format, [706](#)
- DT_HASH
 - ELF binary format, [705](#)
- DT_HIOS
 - ELF binary format, [706](#)
- DT_HIPROC
 - ELF binary format, [706](#)
- DT_INIT
 - ELF binary format, [705](#)
- DT_INIT_ARRAY
 - ELF binary format, [706](#)
- DT_INIT_ARRAYSZ
 - ELF binary format, [706](#)
- DT_JMPREL
 - ELF binary format, [706](#)
- DT_LOOS
 - ELF binary format, [706](#)
- DT_LOPROC
 - ELF binary format, [706](#)
- DT_NEEDED
 - ELF binary format, [705](#)
- DT_NULL
 - ELF binary format, [705](#)
- DT_NUM
 - ELF binary format, [706](#)
- DT_PLTGOT
 - ELF binary format, [705](#)
- DT_PLTRELSZ
 - ELF binary format, [705](#)
- DT_PREINIT_ARRAY
 - ELF binary format, [706](#)
- DT_PREINIT_ARRAYSZ
 - ELF binary format, [706](#)
- DT_PTRREL
 - ELF binary format, [706](#)
- DT_REL
 - ELF binary format, [706](#)
- DT_RELA
 - ELF binary format, [705](#)
- DT_RELAENT
 - ELF binary format, [705](#)
- DT_RELASZ
 - ELF binary format, [705](#)
- DT_RELENT
 - ELF binary format, [706](#)
- DT_RELSZ
 - ELF binary format, [706](#)
- DT_RPATH
 - ELF binary format, [705](#)
- DT_RUNPATH
 - ELF binary format, [706](#)
- DT_SONAME
 - ELF binary format, [705](#)
- DT_STRSZ
 - ELF binary format, [705](#)
- DT_STRTAB
 - ELF binary format, [705](#)
- DT_SYMBOLIC
 - ELF binary format, [705](#)
- DT_SYMENT
 - ELF binary format, [705](#)
- DT_SYMTAB
 - ELF binary format, [705](#)
- DT_TEXTREL
 - ELF binary format, [706](#)
- dump
 - L4Re::Video::Color_component, [1942](#)
 - L4Re::Video::Pixel_info, [1965](#)

- L4virtio::Virtqueue, [2328](#)
- dx
 - l4_vcpu_regs_t, [1603](#)
- E_exist
 - cxx::Bits::Base_avl_set< ITEM_TYPE, COM-PARE, ALLOC, GET_KEY >, [891](#)
- e_flags
 - Elf32_Ehdr, [1003](#)
 - Elf64_Ehdr, [1014](#)
- E_inval
 - cxx::Bits::Base_avl_set< ITEM_TYPE, COM-PARE, ALLOC, GET_KEY >, [891](#)
- e_machine
 - Elf32_Ehdr, [1003](#)
 - Elf64_Ehdr, [1014](#)
- E_noent
 - cxx::Bits::Base_avl_set< ITEM_TYPE, COM-PARE, ALLOC, GET_KEY >, [891](#)
- E_nomem
 - cxx::Bits::Base_avl_set< ITEM_TYPE, COM-PARE, ALLOC, GET_KEY >, [891](#)
- e_type
 - Elf32_Ehdr, [1004](#)
 - Elf64_Ehdr, [1014](#)
- e_version
 - Elf32_Ehdr, [1004](#)
 - Elf64_Ehdr, [1015](#)
- Eager_map
 - L4Re::Rm::F, [1787](#)
 - Rm::F, [2385](#)
- EDID parsing functionality, [454](#)
 - libedid_block_size, [455](#)
 - libedid_check_header, [455](#)
 - libedid_checksum, [455](#)
 - libedid_consts, [455](#)
 - libedid_dump, [455](#)
 - libedid_dump_standard_timings, [456](#)
 - libedid_num_ext_blocks, [456](#)
 - libedid_pnp_id, [456](#)
 - libedid_prefered_resolution, [457](#)
 - libedid_revision, [457](#)
 - libedid_version, [457](#)
- EF_ARM_ALIGN8
 - ELF binary format, [706](#)
- EI_ABIVERSION
 - ELF binary format, [707](#)
- EI_CLASS
 - ELF binary format, [707](#)
- EI_DATA
 - ELF binary format, [707](#)
- EI_MAG0
 - ELF binary format, [707](#)
- EI_MAG1
 - ELF binary format, [707](#)
- EI_MAG2
 - ELF binary format, [707](#)
- EI_MAG3
 - ELF binary format, [707](#)
- EI_NIDENT
 - ELF binary format, [702](#)
- EI_OSABI
 - ELF binary format, [707](#)
- EI_PAD
 - ELF binary format, [707](#)
- EI_VERSION
 - ELF binary format, [707](#)
- ELF binary format, [694](#)
 - AT_BASE, [702](#)
 - AT_EGID, [703](#)
 - AT_ENTRY, [702](#)
 - AT_EUID, [703](#)
 - AT_EXECFD, [702](#)
 - AT_FLAGS, [702](#)
 - AT_GID, [703](#)
 - AT_IGNORE, [702](#)
 - AT_L4_AUX, [703](#)
 - AT_L4_ENV, [703](#)
 - AT_L4_KIP, [703](#)
 - AT_NOTELF, [702](#)
 - AT_NULL, [702](#)
 - AT_PAGESZ, [702](#)
 - AT_PHDR, [702](#)
 - AT_PHENT, [702](#)
 - AT_PHNUM, [702](#)
 - AT_UID, [702](#)
 - DF_1_CONFALT, [704](#)
 - DF_1_DIRECT, [704](#)
 - DF_1_DISPRELDNE, [704](#)
 - DF_1_DISPRELPND, [704](#)
 - DF_1_ENDFILTEE, [704](#)
 - DF_1_GLOBAL, [704](#)
 - DF_1_GROUP, [704](#)
 - DF_1_INITFIRST, [704](#)
 - DF_1_INTERPOSE, [704](#)
 - DF_1_LOADFLTR, [704](#)
 - DF_1_NODEFLIB, [704](#)
 - DF_1_NODELETE, [704](#)
 - DF_1_NODUMP, [704](#)
 - DF_1_NOOPEN, [704](#)
 - DF_1_NOW, [704](#)
 - DF_1_ORIGIN, [704](#)
 - DF_BIND_NOW, [705](#)
 - DF_ORIGIN, [705](#)
 - DF_P1_GROUPPER, [704](#)
 - DF_P1_LAZYLOAD, [704](#)
 - DF_STATIC_TLS, [705](#)
 - DF_SYMBOLIC, [705](#)
 - DF_TEXTREL, [705](#)
 - DT_BIND_NOW, [706](#)
 - DT_DEBUG, [706](#)
 - DT_ENCODING, [706](#)
 - DT_FINI, [705](#)
 - DT_FINI_ARRAY, [706](#)
 - DT_FINI_ARRAYSZ, [706](#)
 - DT_FLAGS, [706](#)
 - DT_HASH, [705](#)

DT_HIOS, [706](#)
DT_HIPROC, [706](#)
DT_INIT, [705](#)
DT_INIT_ARRAY, [706](#)
DT_INIT_ARRAYSZ, [706](#)
DT_JMPREL, [706](#)
DT_LOOS, [706](#)
DT_LOPROC, [706](#)
DT_NEEDED, [705](#)
DT_NULL, [705](#)
DT_NUM, [706](#)
DT_PLTGOT, [705](#)
DT_PLTRELSZ, [705](#)
DT_PREINIT_ARRAY, [706](#)
DT_PREINIT_ARRAYSZ, [706](#)
DT_PTRREL, [706](#)
DT_REL, [706](#)
DT_RELA, [705](#)
DT_RELAENT, [705](#)
DT_RELASZ, [705](#)
DT_RELENT, [706](#)
DT_RELSZ, [706](#)
DT_RPATH, [705](#)
DT_RUNPATH, [706](#)
DT_SONAME, [705](#)
DT_STRSZ, [705](#)
DT_STRTAB, [705](#)
DT_SYMBOLIC, [705](#)
DT_SYMENT, [705](#)
DT_SYMTAB, [705](#)
DT_TEXTREL, [706](#)
EF_ARM_ALIGN8, [706](#)
EI_ABIVERSION, [707](#)
EI_CLASS, [707](#)
EI_DATA, [707](#)
EI_MAG0, [707](#)
EI_MAG1, [707](#)
EI_MAG2, [707](#)
EI_MAG3, [707](#)
EI_NIDENT, [702](#)
EI_OSABI, [707](#)
EI_PAD, [707](#)
EI_VERSION, [707](#)
ELF32_R_TYPE, [700](#)
ELF32_ST_BIND, [700](#)
ELF32_ST_TYPE, [700](#)
ELF64_R_TYPE, [700](#)
ELF64_ST_BIND, [701](#)
ELF64_ST_TYPE, [701](#)
Elf_ARM_SBs, [702](#)
Elf_ATs, [702](#)
Elf_CLASSES, [703](#)
Elf_DATAs, [703](#)
Elf_DF_1s, [703](#)
Elf_DF_P1s, [704](#)
Elf_DFs, [704](#)
Elf_DTs, [705](#)
Elf_EF_ARM_s, [706](#)
Elf_EIs, [706](#)
Elf_EMs, [707](#)
Elf_ETs, [709](#)
Elf_EVs, [709](#)
Elf_MAGs, [710](#)
Elf_NTs_core, [710](#)
Elf_NTs_obj, [711](#)
Elf_OSABIs, [711](#)
ELF_PFs, [711](#)
Elf_PTs, [712](#)
Elf_R_386_s, [712](#)
Elf_R_AARCH64_s, [713](#)
Elf_R_ARM_s, [714](#)
Elf_R_X86_64_s, [714](#)
Elf_SHF_s_ARM, [715](#)
Elf_SHFs, [716](#)
Elf_SHNs, [716](#)
Elf_SHTs, [716](#)
Elf_STBs, [717](#)
Elf_STTs, [718](#)
ELFCLASS32, [703](#)
ELFCLASS64, [703](#)
ELFCLASSNONE, [703](#)
ELFCLASSNUM, [703](#)
ELFDATA2LSB, [703](#)
ELFDATA2MSB, [703](#)
ELFDATANONE, [703](#)
ELFDATANUM, [703](#)
ELFMAG0, [710](#)
ELFMAG1, [710](#)
ELFMAG2, [710](#)
ELFMAG3, [710](#)
ELFOSABI_AIX, [711](#)
ELFOSABI_ARM, [711](#)
ELFOSABI_FREEBSD, [711](#)
ELFOSABI_HPUX, [711](#)
ELFOSABI_IRIX, [711](#)
ELFOSABI_LINUX, [711](#)
ELFOSABI_MODESTO, [711](#)
ELFOSABI_NETBSD, [711](#)
ELFOSABI_NONE, [711](#)
ELFOSABI_OPENBSD, [711](#)
ELFOSABI_SOLARIS, [711](#)
ELFOSABI_STANDALONE, [711](#)
ELFOSABI_SYSV, [711](#)
ELFOSABI_TRU64, [711](#)
EM_386, [707](#)
EM_68HC05, [708](#)
EM_68HC08, [708](#)
EM_68HC11, [708](#)
EM_68HC12, [708](#)
EM_68HC16, [708](#)
EM_68K, [707](#)
EM_860, [707](#)
EM_88K, [707](#)
EM_960, [707](#)
EM_AARCH64, [709](#)
EM_ALPHA, [708](#)

EM_ALTERA_NIOS2, 709
EM_ARC, 708
EM_ARC_A5, 709
EM_ARM, 708
EM_AVR, 708
EM_COLDFIRE, 708
EM_CRIS, 708
EM_D10V, 708
EM_D30V, 708
EM_FIREPATH, 708
EM_FR20, 708
EM_FR30, 708
EM_FX66, 708
EM_H8_300, 708
EM_H8_300H, 708
EM_H8_500, 708
EM_H8S, 708
EM_HUANY, 708
EM_IA_64, 708
EM_JAVELIN, 708
EM_M32, 707
EM_M32R, 709
EM_MICROBLAZE, 709
EM_MIPS, 707
EM_MIPS_RS4_BE, 707
EM_MIPS_X, 708
EM_MMIX, 708
EM_MN10200, 709
EM_MN10300, 709
EM_NONE, 707
EM_OPENRISC, 709
EM_PARISC, 707
EM_PDSP, 708
EM_PJ, 709
EM_PPC, 708
EM_PRISM, 708
EM_RCE, 708
EM_RH32, 708
EM_RISCV, 709
EM_SH, 708
EM_SPARC, 707
EM_SPARC32PLUS, 707
EM_SPARC64, 707
EM_SPARCV9, 708
EM_ST19, 708
EM_ST7, 708
EM_ST9PLUS, 708
EM_SVX, 708
EM_TILEGX, 709
EM_TILEPRO, 709
EM_TRICORE, 708
EM_V800, 708
EM_V850, 709
EM_VAX, 708
EM_VPP500, 707
EM_X86_64, 708
EM_XTENSA, 709
EM_ZSP, 708
ET_CORE, 709
ET_DYN, 709
ET_EXEC, 709
ET_HIPROC, 709
ET_LOPROC, 709
ET_NONE, 709
ET_REL, 709
EV_CURRENT, 710
EV_NONE, 710
NT_ASRS, 710
NT_AUXV, 710
NT_FPREGSET, 710
NT_GWINDOWS, 710
NT_LWPSINFO, 711
NT_LWPSTATUS, 711
NT_PLATFORM, 710
NT_PRCRED, 711
NT_PRFPXREG, 711
NT_PRPSINFO, 710
NT_PRSTATUS, 710
NT_PRXREG, 710
NT_PSINFO, 710
NT_PSTATUS, 710
NT_TASKSTRUCT, 710
NT_UTSNAME, 711
NT_VERSION, 711
PF_ARM_SB, 702
PF_MASKOS, 712
PF_MASKPROC, 712
PF_R, 712
PF_W, 712
PF_X, 712
PT_DYNAMIC, 712
PT_GNU_EH_FRAME, 712
PT_GNU_RELRO, 712
PT_GNU_STACK, 712
PT_HIOS, 712
PT_HIPROC, 712
PT_INTERP, 712
PT_L4_AUX, 712
PT_L4_STACK, 712
PT_LOAD, 712
PT_LOOS, 712
PT_LOPROC, 712
PT_NOTE, 712
PT_NULL, 712
PT_NUM, 712
PT_PHDR, 712
PT_SHLIB, 712
PT_TLS, 712
R_386_32, 713
R_386_COPY, 713
R_386_GLOB_DAT, 713
R_386_GOT32, 713
R_386_GOTOFF, 713
R_386_GOTPC, 713
R_386_JMP_SLOT, 713
R_386_NONE, 713

R_386_NUM, [713](#)
R_386_PC32, [713](#)
R_386_PLT32, [713](#)
R_386_RELATIVE, [713](#)
R_386_TLS_DTPMOD32, [713](#)
R_386_TLS_DTPOFF32, [713](#)
R_386_TLS_GD, [713](#)
R_386_TLS_GD_32, [713](#)
R_386_TLS_GD_CALL, [713](#)
R_386_TLS_GD_POP, [713](#)
R_386_TLS_GD_PUSH, [713](#)
R_386_TLS_GOTIE, [713](#)
R_386_TLS_IE, [713](#)
R_386_TLS_IE_32, [713](#)
R_386_TLS_LDM, [713](#)
R_386_TLS_LDM_32, [713](#)
R_386_TLS_LDM_CALL, [713](#)
R_386_TLS_LDM_POP, [713](#)
R_386_TLS_LDM_PUSH, [713](#)
R_386_TLS_LDO_32, [713](#)
R_386_TLS_LE, [713](#)
R_386_TLS_LE_32, [713](#)
R_386_TLS_TPOFF, [713](#)
R_386_TLS_TPOFF32, [713](#)
R_AARCH64_NONE, [714](#)
R_ARM_ABS12, [714](#)
R_ARM_ABS16, [714](#)
R_ARM_ABS32, [714](#)
R_ARM_ABS8, [714](#)
R_ARM_COPY, [714](#)
R_ARM_GLOB_DAT, [714](#)
R_ARM_GOT32, [714](#)
R_ARM_GOTOFF, [714](#)
R_ARM_GOTPC, [714](#)
R_ARM_JUMP_SLOT, [714](#)
R_ARM_NONE, [714](#)
R_ARM_NUM, [714](#)
R_ARM_PC24, [714](#)
R_ARM_PLT32, [714](#)
R_ARM_REL32, [714](#)
R_ARM_RELATIVE, [714](#)
R_ARM_THM_PC11, [714](#)
R_ARM_THM_PC9, [714](#)
R_X86_64_16, [715](#)
R_X86_64_32, [715](#)
R_X86_64_32S, [715](#)
R_X86_64_64, [715](#)
R_X86_64_8, [715](#)
R_X86_64_COPY, [715](#)
R_X86_64_DTPMOD64, [715](#)
R_X86_64_DTPOFF32, [715](#)
R_X86_64_DTPOFF64, [715](#)
R_X86_64_GLOB_DAT, [715](#)
R_X86_64_GOT32, [715](#)
R_X86_64_GOTPCREL, [715](#)
R_X86_64_GOTTPOFF, [715](#)
R_X86_64_JUMP_SLOT, [715](#)
R_X86_64_NONE, [715](#)
R_X86_64_PC16, [715](#)
R_X86_64_PC32, [715](#)
R_X86_64_PC8, [715](#)
R_X86_64_PLT32, [715](#)
R_X86_64_RELATIVE, [715](#)
R_X86_64_TLSD, [715](#)
R_X86_64_TLSD, [715](#)
R_X86_64_TPOFF32, [715](#)
R_X86_64_TPOFF64, [715](#)
SHF_ALLOC, [716](#)
SHF_ARM_COMDEF, [716](#)
SHF_ARM_ENTRYSECT, [715](#)
SHF_EXECINSTR, [716](#)
SHF_GROUP, [716](#)
SHF_INFO_LINK, [716](#)
SHF_LINK_ORDER, [716](#)
SHF_MASKOS, [716](#)
SHF_MASKPROC, [716](#)
SHF_MERGE, [716](#)
SHF_OS_NONCONFORMING, [716](#)
SHF_STRINGS, [716](#)
SHF_TLS, [716](#)
SHF_WRITE, [716](#)
SHN_ABS, [716](#)
SHN_COMMON, [716](#)
SHN_HIPROC, [716](#)
SHN_HIRESERVE, [716](#)
SHN_LOPROC, [716](#)
SHN_LORESERVE, [716](#)
SHN_UNDEF, [716](#)
SHT_DYNAMIC, [717](#)
SHT_DYNSYM, [717](#)
SHT_FINI_ARRAY, [717](#)
SHT_GROUP, [717](#)
SHT_HASH, [717](#)
SHT_HIOS, [717](#)
SHT_HIPROC, [717](#)
SHT_HIUSER, [717](#)
SHT_INIT_ARRAY, [717](#)
SHT_LOOS, [717](#)
SHT_LOPROC, [717](#)
SHT_LOUSER, [717](#)
SHT_NOBITS, [717](#)
SHT_NOTE, [717](#)
SHT_NULL, [717](#)
SHT_NUM, [717](#)
SHT_PREINIT_ARRAY, [717](#)
SHT_PROGBITS, [717](#)
SHT_REL, [717](#)
SHT_RELA, [717](#)
SHT_SHLIB, [717](#)
SHT_STRTAB, [717](#)
SHT_SYMTAB, [717](#)
SHT_SYMTAB_SHNDX, [717](#)
STB_GLOBAL, [718](#)
STB_HIOS, [718](#)
STB_HIPROC, [718](#)
STB_LOCAL, [718](#)

- STB_LOOS, [718](#)
- STB_LOPROC, [718](#)
- STB_WEAK, [718](#)
- STT_FILE, [718](#)
- STT_FUNC, [718](#)
- STT_HIOS, [718](#)
- STT_HIPROC, [718](#)
- STT_LOOS, [718](#)
- STT_LOPROC, [718](#)
- STT_NOTYPE, [718](#)
- STT_OBJECT, [718](#)
- STT_SECTION, [718](#)
- Elf32_Auxv, [1000](#)
 - atype, [1001](#)
- Elf32_Dyn, [1001](#)
 - d_tag, [1002](#)
- Elf32_Ehdr, [1002](#)
 - e_flags, [1003](#)
 - e_machine, [1003](#)
 - e_type, [1004](#)
 - e_version, [1004](#)
- Elf32_Phdr, [1005](#)
 - p_flags, [1006](#)
 - p_type, [1006](#)
- ELF32_R_TYPE
 - ELF binary format, [700](#)
- Elf32_Rel, [1006](#)
- Elf32_Rela, [1007](#)
- Elf32_Shdr, [1008](#)
 - sh_flags, [1009](#)
 - sh_type, [1009](#)
- ELF32_ST_BIND
 - ELF binary format, [700](#)
- ELF32_ST_TYPE
 - ELF binary format, [700](#)
- Elf32_Sym, [1010](#)
- Elf64_Auxv, [1011](#)
 - atype, [1011](#)
- Elf64_Dyn, [1012](#)
 - d_tag, [1012](#)
- Elf64_Ehdr, [1013](#)
 - e_flags, [1014](#)
 - e_machine, [1014](#)
 - e_type, [1014](#)
 - e_version, [1015](#)
- Elf64_Phdr, [1015](#)
 - p_flags, [1016](#)
 - p_type, [1016](#)
- ELF64_R_TYPE
 - ELF binary format, [700](#)
- Elf64_Rel, [1017](#)
- Elf64_Rela, [1017](#)
- Elf64_Shdr, [1018](#)
 - sh_flags, [1019](#)
 - sh_type, [1019](#)
- ELF64_ST_BIND
 - ELF binary format, [701](#)
- ELF64_ST_TYPE
 - ELF binary format, [701](#)
- Elf64_Sym, [1020](#)
- Elf_ARM_SBs
 - ELF binary format, [702](#)
- Elf_ATs
 - ELF binary format, [702](#)
- Elf_CIASSs
 - ELF binary format, [703](#)
- Elf_DATAs
 - ELF binary format, [703](#)
- Elf_DF_1s
 - ELF binary format, [703](#)
- Elf_DF_P1s
 - ELF binary format, [704](#)
- Elf_DFs
 - ELF binary format, [704](#)
- Elf_DTs
 - ELF binary format, [705](#)
- Elf_EF_ARM_s
 - ELF binary format, [706](#)
- Elf_Els
 - ELF binary format, [706](#)
- Elf_EMs
 - ELF binary format, [707](#)
- Elf_ETs
 - ELF binary format, [709](#)
- Elf_EVs
 - ELF binary format, [709](#)
- Elf_MAGs
 - ELF binary format, [710](#)
- Elf_NT_s_core
 - ELF binary format, [710](#)
- Elf_NT_s_obj
 - ELF binary format, [711](#)
- Elf_OSABIs
 - ELF binary format, [711](#)
- ELF_PFs
 - ELF binary format, [711](#)
- Elf_PT_s
 - ELF binary format, [712](#)
- Elf_R_386_s
 - ELF binary format, [712](#)
- Elf_R_AARCH64_s
 - ELF binary format, [713](#)
- Elf_R_ARM_s
 - ELF binary format, [714](#)
- Elf_R_X86_64_s
 - ELF binary format, [714](#)
- Elf_SHF_s_ARM
 - ELF binary format, [715](#)
- Elf_SHFs
 - ELF binary format, [716](#)
- Elf_SHNs
 - ELF binary format, [716](#)
- Elf_SHTs
 - ELF binary format, [716](#)
- Elf_STBs
 - ELF binary format, [717](#)

- Elf_STTs
 - ELF binary format, [718](#)
- ELFCLASS32
 - ELF binary format, [703](#)
- ELFCLASS64
 - ELF binary format, [703](#)
- ELFCLASSNONE
 - ELF binary format, [703](#)
- ELFCLASSNUM
 - ELF binary format, [703](#)
- ELFDATA2LSB
 - ELF binary format, [703](#)
- ELFDATA2MSB
 - ELF binary format, [703](#)
- ELFDATANONE
 - ELF binary format, [703](#)
- ELFDATANUM
 - ELF binary format, [703](#)
- ELFMAG0
 - ELF binary format, [710](#)
- ELFMAG1
 - ELF binary format, [710](#)
- ELFMAG2
 - ELF binary format, [710](#)
- ELFMAG3
 - ELF binary format, [710](#)
- ELFOSABI_AIX
 - ELF binary format, [711](#)
- ELFOSABI_ARM
 - ELF binary format, [711](#)
- ELFOSABI_FREEBSD
 - ELF binary format, [711](#)
- ELFOSABI_HPUX
 - ELF binary format, [711](#)
- ELFOSABI_IRIX
 - ELF binary format, [711](#)
- ELFOSABI_LINUX
 - ELF binary format, [711](#)
- ELFOSABI_MODESTO
 - ELF binary format, [711](#)
- ELFOSABI_NETBSD
 - ELF binary format, [711](#)
- ELFOSABI_NONE
 - ELF binary format, [711](#)
- ELFOSABI_OPENBSD
 - ELF binary format, [711](#)
- ELFOSABI_SOLARIS
 - ELF binary format, [711](#)
- ELFOSABI_STANDALONE
 - ELF binary format, [711](#)
- ELFOSABI_SYSV
 - ELF binary format, [711](#)
- ELFOSABI_TRU64
 - ELF binary format, [711](#)
- EM_386
 - ELF binary format, [707](#)
- EM_68HC05
 - ELF binary format, [708](#)
- EM_68HC08
 - ELF binary format, [708](#)
- EM_68HC11
 - ELF binary format, [708](#)
- EM_68HC12
 - ELF binary format, [708](#)
- EM_68HC16
 - ELF binary format, [708](#)
- EM_68K
 - ELF binary format, [707](#)
- EM_860
 - ELF binary format, [707](#)
- EM_88K
 - ELF binary format, [707](#)
- EM_960
 - ELF binary format, [707](#)
- EM_AARCH64
 - ELF binary format, [709](#)
- EM_ALPHA
 - ELF binary format, [708](#)
- EM_ALTERA_NIOS2
 - ELF binary format, [709](#)
- EM_ARC
 - ELF binary format, [708](#)
- EM_ARC_A5
 - ELF binary format, [709](#)
- EM_ARM
 - ELF binary format, [708](#)
- EM_AVR
 - ELF binary format, [708](#)
- EM_COLDFIRE
 - ELF binary format, [708](#)
- EM_CRIS
 - ELF binary format, [708](#)
- EM_D10V
 - ELF binary format, [708](#)
- EM_D30V
 - ELF binary format, [708](#)
- EM_FIREPATH
 - ELF binary format, [708](#)
- EM_FR20
 - ELF binary format, [708](#)
- EM_FR30
 - ELF binary format, [708](#)
- EM_FX66
 - ELF binary format, [708](#)
- EM_H8_300
 - ELF binary format, [708](#)
- EM_H8_300H
 - ELF binary format, [708](#)
- EM_H8_500
 - ELF binary format, [708](#)
- EM_H8S
 - ELF binary format, [708](#)
- EM_HUANY
 - ELF binary format, [708](#)
- EM_IA_64
 - ELF binary format, [708](#)

- EM_JAVELIN
 - ELF binary format, [708](#)
- EM_M32
 - ELF binary format, [707](#)
- EM_M32R
 - ELF binary format, [709](#)
- EM_MICROBLAZE
 - ELF binary format, [709](#)
- EM_MIPS
 - ELF binary format, [707](#)
- EM_MIPS_RS4_BE
 - ELF binary format, [707](#)
- EM_MIPS_X
 - ELF binary format, [708](#)
- EM_MMX
 - ELF binary format, [708](#)
- EM_MN10200
 - ELF binary format, [709](#)
- EM_MN10300
 - ELF binary format, [709](#)
- EM_NONE
 - ELF binary format, [707](#)
- EM_OPENRISC
 - ELF binary format, [709](#)
- EM_PARISC
 - ELF binary format, [707](#)
- EM_PDSP
 - ELF binary format, [708](#)
- EM_PJ
 - ELF binary format, [709](#)
- EM_PPC
 - ELF binary format, [708](#)
- EM_PRISM
 - ELF binary format, [708](#)
- EM_RCE
 - ELF binary format, [708](#)
- EM_RH32
 - ELF binary format, [708](#)
- EM_RISCV
 - ELF binary format, [709](#)
- EM_SH
 - ELF binary format, [708](#)
- EM_SPARC
 - ELF binary format, [707](#)
- EM_SPARC32PLUS
 - ELF binary format, [707](#)
- EM_SPARC64
 - ELF binary format, [707](#)
- EM_SPARCV9
 - ELF binary format, [708](#)
- EM_ST19
 - ELF binary format, [708](#)
- EM_ST7
 - ELF binary format, [708](#)
- EM_ST9PLUS
 - ELF binary format, [708](#)
- EM_SVX
 - ELF binary format, [708](#)
- EM_TILEGX
 - ELF binary format, [709](#)
- EM_TILEPRO
 - ELF binary format, [709](#)
- EM_TRICORE
 - ELF binary format, [708](#)
- EM_V800
 - ELF binary format, [708](#)
- EM_V850
 - ELF binary format, [709](#)
- EM_VAX
 - ELF binary format, [708](#)
- EM_VPP500
 - ELF binary format, [707](#)
- EM_X86_64
 - ELF binary format, [708](#)
- EM_XTENSA
 - ELF binary format, [709](#)
- EM_ZSP
 - ELF binary format, [708](#)
- emerg_write_bfm_t
 - L4virtio::Svr::Console::Features, [2171](#)
- emplace
 - cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >, [826](#)
 - cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, [892](#)
- empty
 - L4virtio::Svr::Driver_mem_region_t< DATA >, [2233](#)
- enable_notify
 - L4virtio::Svr::Virtqueue, [2313](#)
- enable_rx_irq
 - L4::Uart, [1547](#)
 - L4::Uart_apb, [1555](#)
- end
 - cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, [893](#)
 - cxx::Bits::Bst< Node, Get_key, Compare >, [909](#)
 - L4::Kip::Mem_desc, [1318](#)
- enqueue
 - utrace::Ring_buffer_producer< SEQUENCE_TYPE, ITEM_TYPE, GENERATION_PTR >, [2414](#)
- enqueue_descriptor
 - L4virtio::Driver::Virtqueue, [2125](#)
- entry_ip
 - L4vcpu::Vcpu, [2072](#)
- entry_sp
 - L4vcpu::Vcpu, [2072](#)
- Enum_bitops, [742](#)
- Enum_bitops::Enable< T >, [1021](#)
- Enum_bitops::Has_marker< T, Void< decltype(enum_bitops_enable(decltype(T >()))> >, [1025](#)
- Enum_bitops::Has_marker< typename, typename >, [1022](#)
- Enum_bitops_impl, [742](#)
- env
 - L4Re::Env, [1681](#)

- env.h
 - l4re_env_t, [2951](#)
- erase
 - cxx::Bits::Base_avl_set< ITEM_TYPE, COM-
PARE, ALLOC, GET_KEY >, [894](#)
 - cxx::H_list< T, POLICY >, [930](#)
- err_no
 - L4::Runtime_error, [1396](#)
- Error
 - L4virtio::Svr::Bad_descriptor, [2134](#)
- error
 - l4_thread_regs_t, [1595](#)
- Error codes, [208](#)
 - L4_E2BIG, [209](#)
 - L4_EACCESS, [209](#)
 - L4_EADDRNOTAVAIL, [209](#)
 - L4_EAGAIN, [209](#)
 - L4_EBADPROTO, [209](#)
 - L4_EBUSY, [209](#)
 - L4_EDROPREPLY, [210](#)
 - L4_EEXIST, [209](#)
 - L4_EFAULT, [209](#)
 - L4_EINVAL, [209](#)
 - L4_EIO, [209](#)
 - L4_EIPC_HI, [210](#)
 - L4_EIPC_LO, [210](#)
 - L4_EMSGERRRANGE, [210](#)
 - L4_EMSGMISSARG, [210](#)
 - L4_EMSGTOOLONG, [210](#)
 - L4_EMSGTOOSHORT, [209](#)
 - L4_ENAMETOOLONG, [209](#)
 - L4_ENODEV, [209](#)
 - L4_ENOENT, [209](#)
 - L4_ENOMEM, [209](#)
 - L4_ENOREPLY, [209](#)
 - L4_ENOSPC, [209](#)
 - L4_ENOSYS, [209](#)
 - L4_ENOTDIR, [209](#)
 - L4_ENXIO, [209](#)
 - L4_EOK, [209](#)
 - L4_EPERM, [209](#)
 - L4_ERANGE, [209](#)
 - L4_ERRNOMAX, [209](#)
 - l4_error_code_t, [209](#)
- Error Handling, [245](#)
 - l4_error, [247](#)
 - L4_IPC_ENOT_EXISTENT, [246](#)
 - l4_ipc_error, [248](#)
 - l4_ipc_error_code, [249](#)
 - L4_IPC_ERROR_MASK, [246](#)
 - l4_ipc_is_rcv_error, [250](#)
 - l4_ipc_is_snd_error, [250](#)
 - L4_IPC_NO_REPLY_CAP, [247](#)
 - L4_IPC_REABORTED, [247](#)
 - L4_IPC_RECANCELED, [246](#)
 - L4_IPC_REMAPFAILED, [246](#)
 - L4_IPC_REMSGCUT, [247](#)
 - L4_IPC_RERCVPFTO, [246](#)
 - L4_IPC_RESNDPFTO, [246](#)
 - L4_IPC_RETIMEOUT, [246](#)
 - L4_IPC_SEABORTED, [247](#)
 - L4_IPC_SECANCELED, [246](#)
 - L4_IPC_SEMAPFAILED, [246](#)
 - L4_IPC_SEMSGCUT, [247](#)
 - L4_IPC_SERCVPFTO, [247](#)
 - L4_IPC_SESNDPFTO, [246](#)
 - L4_IPC_SETIMEOUT, [246](#)
 - L4_IPC_SND_ERR_MASK, [246](#)
 - l4_ipc_tcr_error_t, [246](#)
- ET_CORE
 - ELF binary format, [709](#)
- ET_DYN
 - ELF binary format, [709](#)
- ET_EXEC
 - ELF binary format, [709](#)
- ET_HIPROC
 - ELF binary format, [709](#)
- ET_LOPROC
 - ELF binary format, [709](#)
- ET_NONE
 - ELF binary format, [709](#)
- ET_REL
 - ELF binary format, [709](#)
- EV_CURRENT
 - ELF binary format, [710](#)
- EV_NONE
 - ELF binary format, [710](#)
- Event API, [596](#)
- Event interface, [537](#)
 - l4re_event_get_axis_info, [538](#)
 - l4re_event_get_buffer, [538](#)
 - l4re_event_get_num_streams, [538](#)
 - l4re_event_get_stream_info, [539](#)
 - l4re_event_get_stream_info_for_id, [539](#)
- Event_buffer_t
 - L4Re::Event_buffer_t< PAYLOAD >, [1706](#)
- Events
 - L4virtio::Svr::Console::Control_message, [2147](#)
- ex_regs
 - L4::Thread, [1453](#), [1454](#)
- exc_handler
 - L4::Thread::Attr, [1467](#)
- exception
 - L4::Exception, [1089](#)
- Exception registers, [270](#)
 - l4_utcb_exc, [270](#)
 - l4_utcb_exc_is_ex_regs_exception, [271](#)
 - l4_utcb_exc_is_pf, [271](#)
 - l4_utcb_exc_pc, [272](#)
 - l4_utcb_exc_pc_set, [272](#)
- execute
 - L4Re::Ned::Cmd_control, [1755](#)
- expired
 - L4::lpc_svr::Timeout, [1281](#)
- ext_alloc
 - L4vcpu::Vcpu, [2073](#)

- Extended vCPU support, [732](#)
 - l4vcpu_ext_alloc, [733](#)
- extra_str
 - L4Re::Runtime_error, [1396](#)
- F_above
 - L4Re::Video::View, [1971](#)
- F_auto_refresh
 - L4Re::Video::Goos, [1949](#)
- F_dyn_allocated
 - L4Re::Video::View, [1971](#)
- F_dynamic_buffers
 - L4Re::Video::Goos, [1949](#)
- F_dynamic_views
 - L4Re::Video::Goos, [1949](#)
- F_flags_mask
 - L4Re::Video::View, [1971](#)
- F_fully_dynamic
 - L4Re::Video::View, [1971](#)
- F_l4re_video_goos_auto_refresh
 - Video API, [574](#)
- F_l4re_video_goos_dynamic_buffers
 - Video API, [574](#)
- F_l4re_video_goos_dynamic_views
 - Video API, [574](#)
- F_l4re_video_goos_pointer
 - Video API, [574](#)
- F_l4re_video_view_above
 - Video API, [574](#)
- F_l4re_video_view_dyn_allocated
 - Video API, [574](#)
- F_l4re_video_view_flags_mask
 - Video API, [574](#)
- F_l4re_video_view_none
 - Video API, [574](#)
- F_l4re_video_view_set_background
 - Video API, [574](#)
- F_l4re_video_view_set_buffer
 - Video API, [574](#)
- F_l4re_video_view_set_buffer_offset
 - Video API, [574](#)
- F_l4re_video_view_set_bytes_per_line
 - Video API, [574](#)
- F_l4re_video_view_set_flags
 - Video API, [574](#)
- F_l4re_video_view_set_pixel
 - Video API, [574](#)
- F_l4re_video_view_set_position
 - Video API, [574](#)
- F_none
 - L4Re::Video::View, [1970](#)
- F_pointer
 - L4Re::Video::Goos, [1949](#)
- F_set_background
 - L4Re::Video::View, [1971](#)
- F_set_buffer
 - L4Re::Video::View, [1970](#)
- F_set_buffer_offset
 - L4Re::Video::View, [1970](#)
- F_set_bytes_per_line
 - L4Re::Video::View, [1970](#)
- F_set_flags
 - L4Re::Video::View, [1971](#)
- F_set_pixel
 - L4Re::Video::View, [1971](#)
- F_set_position
 - L4Re::Video::View, [1971](#)
- faccessat
 - L4Re::Vfs::Directory, [1901](#)
- Factory, [288](#)
 - l4_factory_create, [290](#)
 - l4_factory_create_factory, [291](#)
 - l4_factory_create_gate, [292](#)
 - l4_factory_create_irq, [293](#)
 - l4_factory_create_task, [294](#)
 - l4_factory_create_thread, [295](#)
 - l4_factory_create_thread_group, [296](#)
 - l4_factory_create_vcpu_context, [297](#)
 - l4_factory_create_vm, [298](#)
- factory
 - L4Re::Env, [1682](#), [1683](#)
- fchmod
 - L4Re::Vfs::Generic_file, [1919](#)
- fdatsync
 - L4Re::Vfs::Regular_file, [1931](#)
- feature_negotiated
 - L4virtio::Driver::Device, [2105](#)
- features
 - l4_icu_info_t, [1581](#)
- Fiasco extensions, [157](#)
 - fiasco_gdt_get_entry_offset, [159](#)
 - fiasco_gdt_set, [159](#)
 - fiasco_ldt_set, [160](#)
- fiasco_amd64_segment_info
 - segment.h, [2527](#)
- fiasco_amd64_set_fs
 - segment.h, [2528](#)
- fiasco_amd64_set_segment_base
 - segment.h, [2529](#)
- fiasco_dump_kmem_stats
 - kdump.h, [3306](#)
- fiasco_gdt_get_entry_offset
 - Fiasco extensions, [159](#)
- fiasco_gdt_set
 - Fiasco extensions, [159](#)
- fiasco_ldt_set
 - Fiasco extensions, [160](#)
- fiasco_tbuf_log
 - Kernel Tracing, [171](#)
- fiasco_tbuf_log_3val
 - Kernel Tracing, [171](#)
- fiasco_tbuf_log_binary
 - Kernel Tracing, [172](#)
- fiasco_tbuf_map_slots
 - Kernel Tracing, [173](#)
- fiasco_tbuf_map_status
 - Kernel Tracing, [173](#)

- fiasco_tbuf_validate
 - Kernel Tracing, [174](#)
- finalize_request
 - L4virtio::Svr::Block_dev_base< Ds_data >, [2140](#)
- find
 - cxx::Bits::Bst< Node, Get_key, Compare >, [910](#)
 - cxx::String, [987](#)
 - L4::Basic_registry, [1034](#)
 - L4Re::Rm, [1778](#)
 - L4virtio::Svr::Driver_mem_list_t< DATA >, [2225](#)
 - Rm, [2378](#)
- find_next_used
 - L4virtio::Driver::Virtqueue, [2126](#)
- find_node
 - cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, [894](#)
 - cxx::Bits::Bst< Node, Get_key, Compare >, [910](#)
- finish
 - L4virtio::Svr::Virtqueue, [2314](#), [2315](#)
- finish_rx
 - L4virtio::Driver::Virtio_net_device, [2115](#)
- first
 - L4::Kip::Mem_desc, [1318](#)
- first_free_cap
 - L4Re::Env, [1683](#)
- first_free_reply_cap
 - L4Re::Env, [1684](#)
- first_free_utcb
 - L4Re::Env, [1684](#)
- Fixed_paddr
 - L4Re::Mem_alloc, [1734](#)
- Flags
 - L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS >, [1488](#)
 - L4::Types::Flags< BITS_ENUM, UNDERLYING >, [1523](#)
 - L4Re::Dataspace::F, [1663](#)
 - L4Re::Video::Goos, [1949](#)
 - L4Re::Video::View, [1970](#)
- flags
 - l4_exc_regs_t, [1578](#)
 - l4_msgtag_t, [1587](#)
 - L4Re::Dataspace, [1656](#)
 - l4re_env_cap_entry_t, [1983](#)
 - L4virtio::Svr::Driver_mem_region_t< DATA >, [2234](#)
- Flexpages, [175](#)
 - L4_CAP_FPAGE_D, [179](#)
 - L4_CAP_FPAGE_R, [178](#)
 - L4_cap_fpage_rights, [178](#)
 - L4_CAP_FPAGE_RO, [178](#)
 - L4_CAP_FPAGE_RS, [179](#)
 - L4_CAP_FPAGE_RSD, [179](#)
 - L4_CAP_FPAGE_RW, [179](#)
 - L4_CAP_FPAGE_RWD, [179](#)
 - L4_CAP_FPAGE_RWS, [179](#)
 - L4_CAP_FPAGE_RWSD, [179](#)
 - L4_CAP_FPAGE_S, [178](#)
 - L4_CAP_FPAGE_W, [178](#)
 - l4_fpage, [181](#)
 - L4_FPAGE_ADDR_BITS, [180](#)
 - L4_FPAGE_ADDR_SHIFT, [180](#)
 - l4_fpage_all, [182](#)
 - L4_fpage_consts, [179](#)
 - l4_fpage_contains, [182](#)
 - L4_fpage_control, [180](#)
 - L4_FPAGE_CONTROL_MASK, [180](#)
 - L4_FPAGE_CONTROL_OFFSET_SHIFT, [180](#)
 - l4_fpage_invalid, [183](#)
 - L4_FPAGE_IO, [181](#)
 - l4_fpage_ioport, [183](#)
 - l4_fpage_max_order, [184](#)
 - l4_fpage_memaddr, [185](#)
 - L4_FPAGE_MEMORY, [181](#)
 - L4_FPAGE_OBJ, [181](#)
 - l4_fpage_obj, [185](#)
 - l4_fpage_page, [186](#)
 - L4_fpage_rights, [180](#)
 - l4_fpage_rights, [186](#)
 - L4_FPAGE_RIGHTS_ALL, [180](#)
 - L4_FPAGE_RIGHTS_BITS, [180](#)
 - L4_FPAGE_RIGHTS_MASK, [180](#)
 - L4_FPAGE_RIGHTS_SHIFT, [179](#)
 - L4_FPAGE_RO, [180](#)
 - L4_FPAGE_RW, [180](#)
 - L4_FPAGE_RWX, [181](#)
 - L4_FPAGE_RX, [181](#)
 - l4_fpage_set_rights, [187](#)
 - l4_fpage_size, [188](#)
 - L4_FPAGE_SIZE_BITS, [180](#)
 - L4_FPAGE_SIZE_SHIFT, [180](#)
 - L4_FPAGE_SPECIAL, [181](#)
 - L4_fpage_type, [181](#)
 - l4_fpage_type, [188](#)
 - L4_FPAGE_TYPE_BITS, [180](#)
 - L4_FPAGE_TYPE_SHIFT, [179](#)
 - L4_FPAGE_W, [180](#)
 - L4_FPAGE_X, [180](#)
 - l4_iofpage, [189](#)
 - L4_IOPORT_MAX, [178](#)
 - l4_is_fpage_valid, [190](#)
 - l4_is_fpage_writable, [191](#)
 - l4_obj_fpage, [192](#)
 - L4_WHOLE_ADDRESS_SPACE, [177](#)
 - L4_WHOLE_IOADDRESS_SPACE, [178](#)
- flush
 - Mac_table< Size >, [2365](#)
- foreach_available_event
 - L4Re::Util::Event_buffer_consumer_t< PAYLOAD >, [1848](#)
- fpage
 - L4::Cap_base, [1051](#)
- free
 - cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >, [845](#)

- cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >, 850
 - cxx::List_alloc, 947
 - cxx::Slab< Type, Slab_size, Max_free, Alloc >, 977
 - L4::Reply_cap_alloc, 1391
 - L4Re::Cap_alloc, 1641
 - L4Re::Util::_Cap_alloc, 1799
 - L4Re::Util::Counting_cap_alloc< COUNTER-TYPE, Dbg >, 1836
- free_area
 - L4Re::Rm, 1779
 - Rm, 2379
- free_capability
 - L4Re::Util::Names::Name_space, 1865
- free_descriptor
 - L4virtio::Driver::Virtqueue, 2126
- free_dynamic_entry
 - L4Re::Util::Names::Name_space, 1866
- free_epiface
 - L4Re::Util::Names::Name_space, 1866
- free_fd
 - L4Re::Vfs::Fs, 1913
- free_marker
 - l4_thread_regs_t, 1595
- free_objects
 - cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >, 845
 - cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >, 851
- from_ci
 - L4::lpc::Cap< T >, 1150
- from_dec
 - cxx::String, 988
- From_device
 - L4Re::Dma_space, 1673
- from_hex
 - cxx::String, 989
- from_raw
 - L4::Types::Flags< BITS_ENUM, UNDERLYING >, 1524
- fstat
 - L4Re::Vfs::Be_file, 1894
 - L4Re::Vfs::Generic_file, 1919
- fsync
 - L4Re::Vfs::Regular_file, 1931
- ftruncate
 - L4Re::Vfs::Regular_file, 1932
- full
 - L4virtio::Svr::Driver_mem_list_t< DATA >, 2226
- g
 - L4Re::Video::Pixel_info, 1966
- gcd
 - cxx, 738
- generic_write
 - L4::Uart, 1548
- get
 - cxx::Bitfield< T, LSB, MSB >, 855
 - cxx::Ref_ptr< T, CNT >, 968
 - L4::lpc::Iostream, 1159, 1160
 - L4::lpc::Istream, 1168, 1169
 - L4::Reply_cap, 1388
 - L4Re::Core::Ref_ptr< T, CNT >, 1649
 - L4Re::Env, 1685
 - L4Re::Rm::Unique_region< T >, 1793
 - L4Re::Video::Color_component, 1943
 - L4vbus::Gpio_module, 2020
 - L4vbus::Gpio_pin, 2030
 - L4virtio::Ptr< T >, 2132
 - Rm::Unique_region< T >, 2390
- get_areas
 - L4Re::Rm, 1780
 - Rm, 2380
- get_attr
 - L4::Vcon, 1565
- get_avail_idx
 - L4virtio::Virtqueue, 2328
- get_axis_info
 - L4Re::Event, 1697
- get_buffer
 - L4Re::Event, 1698
- get_buffer_demand
 - L4::Epiface, 1079
 - L4::Server_object_t< IFACE, BASE >, 1426
- get_cap
 - L4Re::Env, 1685, 1686
- get_char
 - L4::Uart, 1548
 - L4::Uart_apb, 1556
- get_cmd
 - L4virtio::Svr::Dev_config, 2203
- get_epiface
 - L4Re::Util::Names::Name_space, 1867
- get_fb
 - L4Re::Util::Video::Goos_svr, 1889
- get_file
 - L4Re::Vfs::Fs, 1913
- get_info
 - L4Re::Rm, 1781
 - Rm, 2380
- get_infos
 - L4::lpc_gate, 1248
- get_lock
 - L4Re::Vfs::Regular_file, 1933
- get_num_streams
 - L4Re::Event, 1699
- get_object_name
 - L4::Debugger, 1064
- get_random
 - L4Re::Random, 1764
- get_rcv_cap
 - L4::lpc_svr::Server_iface, 1274
 - L4Re::Util::Br_manager, 1819
- get_rcv_mem
 - L4::lpc_svr::Server_iface, 1274
 - L4Re::Util::Br_manager, 1820

- get_regions
 - L4Re::Rm, [1782](#)
 - Rm, [2381](#)
- get_request
 - L4virtio::Svr::Virtio_gpio< Request_handler, Epiface >::Request_processor, [2272](#)
 - L4virtio::Svr::Virtio_i2c< Request_handler, Epiface >::Request_processor, [2283](#)
 - L4virtio::Svr::Virtio_spi< Spi_request_handler, Epiface >::Request_processor, [2305](#)
 - Virtio_net_request, [2430](#)
- get_resource
 - L4vbus::Device, [2013](#)
- get_static_buffer
 - L4Re::Video::Goos, [1953](#)
- get_status_flags
 - L4Re::Vfs::Generic_file, [1920](#)
- get_stream_info
 - L4Re::Event, [1700](#)
- get_stream_info_for_id
 - L4Re::Event, [1701](#)
- get_stream_state_for_id
 - L4Re::Event, [1702](#)
- get_tail_avail_idx
 - L4virtio::Virtqueue, [2329](#)
- get_unshifted
 - cxx::Bitfield< T, LSB, MSB >, [855](#)
- get_value
 - L4::lpc::Varg, [1233](#)
- get_writeback
 - L4virtio::Svr::Block_dev_base< Ds_data >, [2141](#)
- getitimer
 - L4Re::Itas, [1717](#)
- global_id
 - L4::Debugger, [1065](#)
- gran_offset
 - I4_sched_cpu_set_t, [1591](#)
- Grant
 - L4::lpc::Cap< T >, [1149](#)
 - L4::lpc::Snd_fpage, [1224](#)
- granularity
 - I4_sched_cpu_set_t, [1589](#)
- guest_features
 - L4virtio::Svr::Dev_config, [2204](#)
- H_list_item_t
 - cxx::H_list_item_t< ELEM_TYPE >, [936](#)
- handle_control_message
 - L4virtio::Svr::Console::Virtio_con, [2183](#)
- handle_expired_timeouts
 - L4::lpc_svr::Timeout_queue, [1283](#)
- handle_I4virtio_port_tx
 - Virtio_switch, [2434](#)
- handle_mem_cmd_write
 - L4virtio::Svr::Device_t< DATA >, [2220](#)
- has_alpha
 - L4Re::Video::Pixel_info, [1966](#)
- has_error
 - I4_msgtag_t, [1587](#)
- has_more
 - L4virtio::Svr::Request_processor, [2238](#)
- hdr
 - L4virtio::Svr::Dev_config, [2204](#)
- i
 - L4vcpu::Vcpu, [2073](#), [2074](#)
- IA32 Port I/O API, [656](#)
- I4util_in16, [657](#)
- I4util_in32, [657](#)
- I4util_in8, [657](#)
- I4util_ins16, [658](#)
- I4util_ins32, [658](#)
- I4util_ins8, [659](#)
- I4util_ioport_map, [659](#)
- I4util_out16, [660](#)
- I4util_out32, [660](#)
- I4util_out8, [661](#)
- I4util_outs16, [661](#)
- I4util_outs32, [661](#)
- I4util_outs8, [662](#)
- id
 - L4::Kobject_typeid< T >, [1338](#)
 - L4::Kobject_typeid< void >, [1342](#)
- id_received
 - L4::lpc::Snd_fpage, [1226](#)
- idle_time
 - L4::Scheduler, [1402](#)
- Ignore_sigaction
 - L4Re::Itas, [1717](#)
- In_area
 - L4Re::Rm::F, [1787](#)
 - Rm::F, [2385](#)
- inc
 - L4Re::Util::Counter< COUNTER >, [1830](#)
 - L4Re::Util::Counter_atomic< COUNTER >, [1832](#)
- include.mk - Header File Role, [38](#)
- index
 - utrace::Tracebuffer, [2421](#)
- indexes
 - utrace::Tracebuffer, [2422](#)
- indirect_bfm_t
 - L4virtio::Virtqueue::Desc::Flags, [2343](#)
- Info
 - L4::Kip::Mem_desc, [1314](#)
- info
 - L4::lcu, [1116](#)
 - L4::Scheduler, [1402](#)
 - L4Re::Dataspace, [1657](#)
 - L4Re::Mem_alloc, [1735](#)
 - L4Re::Video::Goos, [1954](#)
 - L4Re::Video::View, [1971](#)
- Info_acpi_rsdp
 - L4::Kip::Mem_desc, [1314](#)
- Info_sub_type
 - L4::Kip::Mem_desc, [1314](#)
- inhibitor.h
 - I4re_inhibitor_acquire, [2912](#)
 - I4re_inhibitor_next_lock_info, [2912](#)

- l4re_inhibitor_release, [2913](#)
- init
 - L4Re::Util::Event_t< PAYLOAD >, [1857](#)
 - L4virtio::Svr::Driver_mem_list_t< DATA >, [2227](#)
- init_infos
 - L4Re::Util::Video::Goos_svr, [1889](#)
- init_mem_info
 - L4virtio::Svr::Device_t< DATA >, [2221](#)
- init_poll
 - L4Re::Util::Event_t< PAYLOAD >, [1858](#)
- init_queue
 - L4virtio::Driver::Virtqueue, [2127](#)
- Initial Environment, [580](#)
 - l4re_env, [581](#)
 - l4re_env_get_cap, [582](#)
 - l4re_env_get_cap_e, [582](#)
 - l4re_env_get_cap_l, [583](#)
 - l4re_kip, [584](#)
- Initial Environment and Application Bootstrapping, [25](#)
- Initial Memory Allocator and Factory, [28](#)
- initial_caps
 - L4Re::Env, [1687](#)
- initialize_rings
 - L4virtio::Driver::Virtqueue, [2128](#)
- insert
 - cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >, [826](#)
 - cxx::Avl_tree< Node, Get_key, Compare >, [838](#)
 - cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, [894](#)
 - cxx::H_list< T, POLICY >, [930](#)
- insert_after
 - cxx::H_list< T, POLICY >, [931](#)
- insert_before
 - cxx::H_list< T, POLICY >, [931](#)
- Integer Types, [452](#)
- interface
 - L4::Meta, [1351](#)
- Interface Definition Language, [30](#)
- Interface for asynchronous ISR handlers with a given IRQ capability., [477](#)
 - l4irq_request_cap, [478](#)
- Interface for asynchronous ISR handlers., [475](#)
 - l4irq_release, [476](#)
 - l4irq_request, [476](#)
- Interface using direct functionality., [468](#), [473](#)
 - l4irq_attach, [469](#)
 - l4irq_attach_cap, [473](#)
 - l4irq_attach_cap_ft, [473](#)
 - l4irq_attach_ft, [469](#)
 - l4irq_attach_thread, [469](#)
 - l4irq_attach_thread_cap, [474](#)
 - l4irq_attach_thread_cap_ft, [474](#)
 - l4irq_attach_thread_ft, [470](#)
 - l4irq_detach, [470](#)
 - l4irq_unmask, [471](#)
 - l4irq_unmask_and_wait_any, [471](#)
 - l4irq_wait, [472](#)
- l4irq_wait_any, [472](#)
- Internal, [605](#)
 - L4SHMC_RINGBUF_DATA, [606](#)
 - L4SHMC_RINGBUF_DATA_SIZE, [606](#)
 - L4SHMC_RINGBUF_HEAD, [607](#)
- Internal constants, [639](#)
- Internal functions, [687](#)
- Internal Helpers, [194](#)
- internal_loop
 - L4::Server< LOOP_HOOKS >, [1416](#)
- Interrupt controller, [321](#)
 - l4_icu_bind, [324](#)
 - l4_icu_bind_u, [325](#)
 - L4_ICU_FLAG_MSI, [324](#)
 - L4_icu_flags, [324](#)
 - l4_icu_info, [326](#)
 - l4_icu_info_t, [323](#)
 - l4_icu_info_u, [327](#)
 - l4_icu_mask, [328](#)
 - l4_icu_mask_u, [329](#)
 - l4_icu_msi_info, [330](#)
 - l4_icu_msi_info_u, [331](#)
 - l4_icu_set_mode, [332](#)
 - l4_icu_set_mode_u, [333](#)
 - l4_icu_unbind, [334](#)
 - l4_icu_unbind_u, [335](#)
 - l4_icu_unmask, [336](#)
 - l4_icu_unmask_u, [337](#)
- Introduction, [3](#)
- Invalid
 - L4::Cap_base, [1047](#)
 - L4virtio::Ptr< T >, [2131](#)
- Invalid_capability
 - L4::Invalid_capability, [1125](#)
- Invalid_type
 - L4virtio::Ptr< T >, [2131](#)
- io
 - L4::lpc::Gen_fpage, [1153](#)
- io
 - L4::lpc::Rcv_fpage, [1213](#)
 - L4::lpc::Snd_fpage, [1226](#)
- IO interface, [458](#)
 - L4IO_DEVICE_ANY, [460](#)
 - L4IO_DEVICE_INVALID, [460](#)
 - L4IO_DEVICE_OTHER, [460](#)
 - L4IO_DEVICE_PCI, [460](#)
 - l4io_device_types_t, [459](#)
 - L4IO_DEVICE_USB, [460](#)
 - l4io_has_resource, [461](#)
 - l4io_iomem_flags_t, [460](#)
 - l4io_lookup_device, [461](#)
 - l4io_lookup_resource, [461](#)
 - L4IO_MEM_CACHED, [460](#)
 - L4IO_MEM_EAGER_MAP, [460](#)
 - L4IO_MEM_NONCACHED, [460](#)
 - L4IO_MEM_USE_MTRR, [460](#)
 - L4IO_MEM_USE_RESERVED_AREA, [460](#)
 - l4io_release_iomem, [462](#)

- l4io_release_ioport, [462](#)
- l4io_request_iomem, [463](#)
- l4io_request_iomem_region, [464](#)
- l4io_request_ioport, [464](#)
- l4io_request_resource_iomem, [465](#)
- L4IO_RESOURCE_ANY, [460](#)
- L4IO_RESOURCE_INVALID, [460](#)
- L4IO_RESOURCE_IRQ, [460](#)
- L4IO_RESOURCE_MEM, [460](#)
- L4IO_RESOURCE_PORT, [460](#)
- l4io_resource_t, [459](#)
- l4io_resource_types_t, [460](#)
- Io, the Io Server, [61](#)
- io.h
 - l4io_get_root_device, [2644](#)
 - l4io_iterate_devices, [2644](#)
 - l4io_request_all_ioports, [2644](#)
 - l4io_request_icu, [2645](#)
- io_page_fault
 - L4::Io_pager, [1129](#)
- ioctl
 - L4Re::Vfs::Special_file, [1940](#)
- lostream
 - L4::lpc::lostream, [1158](#)
- IPC Helpers, [466](#)
 - throw_ipc_exception, [466](#)
- IPC-Gate API, [281](#)
 - l4_ipc_gate_get_infos, [282](#)
 - l4_rcv_ep_bind_snd_destination, [283](#)
 - l4_rcv_ep_bind_thread, [284](#)
- ipc.h
 - l4_ipc_to_errno, [3273](#)
- ipc_client
 - L4_RPC_DEF, [3182](#)
- ipc_iface
 - L4_INLINE_RPC, [3191](#)
 - L4_INLINE_RPC_NF, [3191](#)
 - L4_INLINE_RPC_NF_OP, [3191](#)
 - L4_INLINE_RPC_OP, [3192](#)
 - L4_RPC, [3192](#)
 - L4_RPC_NF, [3193](#)
 - L4_RPC_NF_OP, [3193](#)
 - L4_RPC_OP, [3194](#)
- ipc_stream
 - operator<<, [2708–2710](#)
 - operator>>, [2710–2714](#)
- irq
 - L4Re::Util::Event_t< PAYLOAD >, [1858](#)
- IRQ handling library, [467](#)
- irq_disable_save
 - L4vcpu::Vcpu, [2074](#)
- irq_enable
 - L4vbus::Pci_dev, [2042](#)
 - L4vbus::Pci_host_bridge, [2048](#)
 - L4vcpu::Vcpu, [2074](#)
- irq_restore
 - L4vcpu::Vcpu, [2075](#)
- IRQs, [338](#)
- l4_irq_bind_vcpu, [340](#)
- l4_irq_bind_vcpu_u, [341](#)
- l4_irq_detach, [343](#)
- l4_irq_detach_u, [344](#)
- L4_IRQ_F_BOTH, [340](#)
- L4_IRQ_F_BOTH_EDGE, [340](#)
- L4_IRQ_F_CLEAR_WAKEUP, [340](#)
- L4_IRQ_F_EDGE, [340](#)
- L4_IRQ_F_LEVEL, [340](#)
- L4_IRQ_F_LEVEL_HIGH, [340](#)
- L4_IRQ_F_LEVEL_LOW, [340](#)
- L4_IRQ_F_MASK, [340](#)
- L4_IRQ_F_NEG, [340](#)
- L4_IRQ_F_NEG_EDGE, [340](#)
- L4_IRQ_F_NONE, [340](#)
- L4_IRQ_F_POS, [340](#)
- L4_IRQ_F_POS_EDGE, [340](#)
- L4_IRQ_F_SET_MODE, [340](#)
- L4_IRQ_F_SET_WAKEUP, [340](#)
- L4_irq_mode, [340](#)
- l4_irq_receive, [345](#)
- l4_irq_receive_u, [346](#)
- l4_irq_trigger, [347](#)
- l4_irq_trigger_u, [348](#)
- l4_irq_unmask, [349](#)
- l4_irq_unmask_u, [350](#)
- l4_irq_wait, [351](#)
- l4_irq_wait_u, [352](#)
- is_compatible
 - L4vbus::Device, [2014](#)
- is_compound
 - L4::lpc::Snd_fpage, [1227](#)
- is_irq_entry
 - L4vcpu::Vcpu, [2076](#)
- is_nil
 - L4::lpc::Varg, [1233](#)
- is_of
 - L4::lpc::Varg, [1234](#)
- is_of_int
 - L4::lpc::Varg, [1235](#)
- is_online
 - L4::Scheduler, [1403](#)
- is_page_fault_entry
 - L4vcpu::Vcpu, [2076](#)
- is_static
 - L4Re::Util::Dataspace_svr, [1842](#)
- is_valid
 - L4::Cap_base, [1052](#)
 - L4Re::Rm::Unique_region< T >, [1793](#)
 - L4virtio::Ptr< T >, [2132](#)
 - Rm::Unique_region< T >, [2390](#)
- is_virtual
 - L4::Kip::Mem_desc, [1319](#)
- is_writable
 - L4virtio::Svr::Driver_mem_region_t< DATA >, [2234](#)
- Istream
 - L4::lpc::Istream, [1167](#)

- itas
 - L4Re::Env, 1688
- Item_bytes
 - L4::lpc::Msg, 766
- Item_words
 - L4::lpc::Msg, 766
- iter
 - cxx::Bits::Basic_list< POLICY >, 902
 - cxx::H_list< T, POLICY >, 932
- Iterator
 - cxx::Avl_tree< Node, Get_key, Compare >, 838
- kdebug.h
 - __l4_kdebug_3_text, 3290
 - __l4_kdebug_op, 3291
 - __l4_kdebug_op_1, 3292
 - __l4_kdebug_text, 3294
 - l4_kd_enter, 3295
 - l4_kd_outchar, 3296
 - l4_kd_outdec, 3297
 - l4_kd_outhex12, 3297
 - l4_kd_outhex16, 3298
 - l4_kd_outhex20, 3298
 - l4_kd_outhex32, 3299
 - l4_kd_outhex64, 3299
 - l4_kd_outhex8, 3300
 - l4_kd_outnstring, 3301
 - l4_kd_outstring, 3302
 - l4_kd_outumword, 3302
 - l4_kdebug_ops_t, 3290
- kdump.h
 - fiasco_dump_kmem_stats, 3306
- Kept_ds
 - L4Re::Rm, 1771
 - Rm, 2374
- Kernel
 - L4Re::Rm::F, 1788
 - Rm::F, 2385
- Kernel ABI, 16
- Kernel Debugger, 161
 - l4_debugger_add_image_info, 162
 - l4_debugger_get_object_name, 163
 - l4_debugger_global_id, 164
 - l4_debugger_kobj_to_id, 164
 - l4_debugger_query_log_name, 165
 - l4_debugger_query_log_typeid, 166
 - l4_debugger_query_object_name, 167
 - l4_debugger_set_object_name, 168
 - l4_debugger_switch_log, 168
- Kernel Factory, 46
- Kernel Information Dump, 169
- Kernel Interface Page, 433
 - l4_kernel_info_t, 435
 - l4_kernel_info_version_offset, 436
 - l4_kip, 436
 - l4_kip_clock, 436
 - l4_kip_clock_lw, 437
 - l4_kip_clock_ns, 438
 - L4_KIP_OFFS_READ_NS, 435
 - L4_KIP_OFFS_READ_US, 435
 - l4_kip_version, 438
 - l4_kip_version_string, 439
- Kernel Interface Page API, 719
 - l4util_kip_for_each_feature, 719
 - l4util_kip_kernel_abi_version, 720
 - l4util_kip_kernel_has_feature, 720
- Kernel Objects, 278
- Kernel Tracing, 170
 - fiasco_tbuf_log, 171
 - fiasco_tbuf_log_3val, 171
 - fiasco_tbuf_log_binary, 172
 - fiasco_tbuf_map_slots, 173
 - fiasco_tbuf_map_status, 173
 - fiasco_tbuf_validate, 174
- Kernel-provided semaphore, 367
 - l4_semaphore_down, 368
 - l4_semaphore_up, 368
- kip.h
 - l4_kip_for_each_feature, 3460
 - l4_kip_kernel_has_feature, 3460
- kobj_to_id
 - L4::Debugger, 1066
- kobject_typeid
 - L4 kernel object type information, 287
- Kumem allocator utility, 571
- Kumem utilities, 591
 - kumem_alloc, 591
- kumem_alloc
 - Kumem utilities, 591
- kumem_alloc.h
 - l4re_util_kumem_alloc, 2928
- L
 - cxx::Bits::Direction, 919
- L4, 743
 - cap_cast, 748, 749
 - cap_dynamic_cast, 750
 - cap_reinterpret_cast, 752, 753
 - PROTO_ANY, 748
 - PROTO_EMPTY, 748
 - round_order, 754
 - trunc_order, 755
- L4 IPC Opcodes, 478
 - L4_ICU_OP_BIND, 480
 - L4_ICU_OP_INFO, 480
 - L4_ICU_OP_MASK, 480
 - L4_ICU_OP_MSI_INFO, 480
 - L4_ICU_OP_SET_MODE, 480
 - L4_ICU_OP_UNBIND, 480
 - L4_ICU_OP_UNMASK, 480
 - L4_icu_opcode, 479
 - L4_IPC_GATE_BIND_OP, 480
 - L4_IPC_GATE_GET_INFO_OP, 480
 - L4_ipc_gate_ops, 480
 - L4_PLATFORM_CTL_CPU_ALLOW_SHUTDOWN_OP, 481
 - L4_PLATFORM_CTL_CPU_DISABLE_OP, 481
 - L4_PLATFORM_CTL_CPU_ENABLE_OP, 481

- L4_platform_ctl_ops, [480](#)
- L4_PLATFORM_CTL_SET_TASK_ASID_OP, [481](#)
- L4_PLATFORM_CTL_SYS_SHUTDOWN_OP, [481](#)
- L4_PLATFORM_CTL_SYS_SUSPEND_OP, [481](#)
- L4_TASK_ADD_KU_MEM_OP, [481](#)
- L4_TASK_CAP_INFO_OP, [481](#)
- L4_TASK_LDT_SET_X86_OP, [481](#)
- L4_TASK_MAP_OP, [481](#)
- L4_TASK_MAP_VGICC_ARM_OP, [481](#)
- L4_task_ops, [481](#)
- L4_TASK_UNMAP_OP, [481](#)
- L4_THREAD_AMD64_GET_SEGMENT_INFO_OP, [482](#)
- L4_THREAD_AMD64_SET_SEGMENT_BASE_OP, [482](#)
- L4_THREAD_ARM_TPIDRURO_OP, [482](#)
- L4_THREAD_CONTROL_OP, [481](#)
- L4_THREAD_EX_REGS_OP, [482](#)
- L4_THREAD_MODIFY_SENDER_OP, [482](#)
- L4_THREAD_OPCODE_MASK, [482](#)
- L4_thread_ops, [481](#)
- L4_THREAD_REGISTER_DELETE_IRQ_OP, [482](#)
- L4_THREAD_REGISTER_DOORBELL_IRQ_OP, [482](#)
- L4_THREAD_STATS_OP, [482](#)
- L4_THREAD_SWITCH_OP, [482](#)
- L4_THREAD_VCPU_CONTROL_OP, [482](#)
- L4_THREAD_VCPU_RESUME_OP, [482](#)
- L4_THREAD_X86_GDT_OP, [482](#)
- L4_VCON_GET_ATTR_OP, [482](#)
- L4_vcon_ops, [482](#)
- L4_VCON_READ_OP, [482](#)
- L4_VCON_SET_ATTR_OP, [482](#)
- L4_VCON_WRITE_OP, [482](#)
- L4 kernel object type information, [286](#)
 - kobject_typeid, [287](#)
- L4 Vbus functions, [495](#)
 - l4vbus_assign_dma_domain, [497](#)
 - L4vbus_dma_domain_assign_flags, [497](#)
 - L4VBUS_DMAD_BIND, [497](#)
 - L4VBUS_DMAD_KERNEL_DMA_SPACE, [497](#)
 - L4VBUS_DMAD_L4RE_DMA_SPACE, [497](#)
 - L4VBUS_DMAD_UNBIND, [497](#)
 - l4vbus_get_adr, [498](#)
 - l4vbus_get_device, [498](#)
 - l4vbus_get_device_by_hid, [499](#)
 - l4vbus_get_hid, [500](#)
 - l4vbus_get_next_device, [501](#)
 - l4vbus_get_resource, [501](#)
 - l4vbus_is_compatible, [502](#)
 - l4vbus_release_ioport, [503](#)
 - l4vbus_request_ioport, [504](#)
 - l4vbus_vicu_get_cap, [504](#)
- L4 VIRTIO Block Device, [492](#)
 - L4virtio_block_operations, [493](#)
 - L4VIRTIO_BLOCK_S_IOERR, [493](#)
 - L4VIRTIO_BLOCK_S_OK, [493](#)
 - L4VIRTIO_BLOCK_S_UNSUPP, [493](#)
 - L4virtio_block_status, [493](#)
 - L4VIRTIO_BLOCK_T_DISCARD, [493](#)
 - L4VIRTIO_BLOCK_T_FLUSH, [493](#)
 - L4VIRTIO_BLOCK_T_GET_ID, [493](#)
 - L4VIRTIO_BLOCK_T_IN, [493](#)
 - L4VIRTIO_BLOCK_T_OUT, [493](#)
 - L4VIRTIO_BLOCK_T_WRITE_ZEROES, [493](#)
- L4 VIRTIO Input Device, [494](#)
- L4 VIRTIO Interface, [483](#)
- L4 VIRTIO Network Device, [494](#)
- L4 VIRTIO Transport Layer, [483](#)
 - L4_virtio_cmd, [486](#)
 - L4_virtio_irq_status, [486](#)
 - L4_virtio_opcodes, [486](#)
 - L4VIRTIO_CMD_CFG_CHANGED, [486](#)
 - L4VIRTIO_CMD_CFG_QUEUE, [486](#)
 - L4VIRTIO_CMD_MASK, [486](#)
 - L4VIRTIO_CMD_NONE, [486](#)
 - L4VIRTIO_CMD_NOTIFY_QUEUE, [486](#)
 - L4VIRTIO_CMD_SET_STATUS, [486](#)
 - l4virtio_config_queue, [488](#)
 - l4virtio_config_queue_t, [485](#)
 - l4virtio_config_queues, [489](#)
 - l4virtio_device_config, [489](#)
 - l4virtio_device_config_ds, [489](#)
 - L4virtio_device_ids, [487](#)
 - l4virtio_device_notification_irq, [490](#)
 - L4virtio_device_status, [487](#)
 - L4virtio_feature_bits, [488](#)
 - L4VIRTIO_FEATURE_CMD_CONFIG, [488](#)
 - L4VIRTIO_FEATURE_VERSION_1, [488](#)
 - L4VIRTIO_ID_9P, [487](#)
 - L4VIRTIO_ID_BALLOON, [487](#)
 - L4VIRTIO_ID_BLOCK, [487](#)
 - L4VIRTIO_ID_CAIF, [487](#)
 - L4VIRTIO_ID_CAN, [487](#)
 - L4VIRTIO_ID_CONSOLE, [487](#)
 - L4VIRTIO_ID_CRYPTOP, [487](#)
 - L4VIRTIO_ID_FS, [487](#)
 - L4VIRTIO_ID_GPIO, [487](#)
 - L4VIRTIO_ID_GPU, [487](#)
 - L4VIRTIO_ID_I2C, [487](#)
 - L4VIRTIO_ID_INPUT, [487](#)
 - L4VIRTIO_ID_NET, [487](#)
 - L4VIRTIO_ID_RNG, [487](#)
 - L4VIRTIO_ID_RPMSG, [487](#)
 - L4VIRTIO_ID_RPROC_SERIAL, [487](#)
 - L4VIRTIO_ID_SCMI, [487](#)
 - L4VIRTIO_ID_SCSI, [487](#)
 - L4VIRTIO_ID_SOCKET, [487](#)
 - L4VIRTIO_ID_SPI, [487](#)
 - L4VIRTIO_ID_VSOCK, [487](#)
 - L4VIRTIO_ID_WATCHDOG, [487](#)
 - L4VIRTIO_IRQ_STATUS_CONFIG, [486](#)
 - L4VIRTIO_IRQ_STATUS_VRING, [486](#)
 - L4VIRTIO_OP_CONFIG_QUEUE, [487](#)
 - L4VIRTIO_OP_DEVICE_CONFIG, [487](#)

[L4VIRTIO_OP_GET_DEVICE_IRQ](#), 487
[L4VIRTIO_OP_REGISTER_DS](#), 487
[L4VIRTIO_OP_SET_STATUS](#), 486
[l4virtio_register_ds](#), 490
[l4virtio_set_status](#), 491
[L4VIRTIO_STATUS_ACKNOWLEDGE](#), 488
[L4VIRTIO_STATUS_DEVICE_NEEDS_RESET](#), 488
[L4VIRTIO_STATUS_DRIVER](#), 488
[L4VIRTIO_STATUS_DRIVER_OK](#), 488
[L4VIRTIO_STATUS_FAILED](#), 488
[L4VIRTIO_STATUS_FEATURES_OK](#), 488
[l4/cxx/alloc.h](#), 2646, 2647
[l4/cxx/arith](#), 2647
[l4/cxx/atomic.h](#), 3419, 3420
[l4/cxx/avl_map](#), 2648, 2650
[l4/cxx/avl_set](#), 2651, 2653
[l4/cxx/avl_tree](#), 2656, 2658
[l4/cxx/basic_ostream](#), 2662, 2663
[l4/cxx/basic_vector.h](#), 2666, 2667
[l4/cxx/bitfield](#), 2667
[l4/cxx/bitmap](#), 2669
[l4/cxx/bits/bst.h](#), 2672, 2674
[l4/cxx/bits/bst_base.h](#), 2676, 2678
[l4/cxx/bits/bst_iter.h](#), 2679, 2681
[l4/cxx/bits/list_basics.h](#), 2682
[l4/cxx/bits/smart_ptr_list.h](#), 2684, 2685
[l4/cxx/bits/type_traits.h](#), 2687
[l4/cxx/dlist](#), 2690
[l4/cxx/elide_dtor](#), 2695
[l4/cxx/exceptions](#), 2696, 2698
[l4/cxx/hlist](#), 2700
[l4/cxx/iostream](#), 2702, 2703
[l4/cxx/ipc_helper](#), 2703, 2705
[l4/cxx/ipc_server](#), 3202, 3203
[l4/cxx/ipc_stream](#), 2705, 2715
[l4/cxx/ipc_timeout_queue](#), 2723
[l4/cxx/l4iostream](#), 2725, 2726
[l4/cxx/l4types.h](#), 2726, 2727
[l4/cxx/list](#), 2727
[l4/cxx/list_alloc](#), 2731
[l4/cxx/lock_guard.h](#), 2737
[l4/cxx/main_thread](#), 2738, 2739
[l4/cxx/minmax](#), 2740
[l4/cxx/numeric](#), 2741
[l4/cxx/observer](#), 2741
[l4/cxx/pair](#), 2742, 2743
[l4/cxx/ref_ptr](#), 2744
[l4/cxx/ref_ptr_list](#), 2747, 2748
[l4/cxx/result](#), 2749
[l4/cxx/slab_alloc](#), 2751
[l4/cxx/slist](#), 2754
[l4/cxx/static_container](#), 2757
[l4/cxx/static_vector](#), 2758
[l4/cxx/std_alloc](#), 2758
[l4/cxx/std_ops](#), 2759
[l4/cxx/string](#), 2759
[l4/cxx/string.h](#), 2762, 2763
[l4/cxx/thread](#), 3368, 3369
[l4/cxx/type_list](#), 2764
[l4/cxx/type_traits](#), 2764
[l4/cxx/unique_ptr](#), 2769
[l4/cxx/unique_ptr_list](#), 2770, 2772
[l4/cxx/utils](#), 2772
[l4/cxx/weak_ref](#), 2772
[l4/irq/irq.h](#), 2777, 2778
[l4/l4re_vfs/backend](#), 2785
[l4/l4re_vfs/impl/default_ops_impl.h](#), 2788
[l4/l4re_vfs/impl/fd_store.h](#), 2789
[l4/l4re_vfs/impl/fd_store_impl.h](#), 2790
[l4/l4re_vfs/impl/ns_fs.h](#), 2790
[l4/l4re_vfs/impl/ns_fs_impl.h](#), 2791
[l4/l4re_vfs/impl/ro_file.h](#), 2796
[l4/l4re_vfs/impl/ro_file_impl.h](#), 2796
[l4/l4re_vfs/impl/vcon_stream.h](#), 2798
[l4/l4re_vfs/impl/vcon_stream_impl.h](#), 2798
[l4/l4re_vfs/impl/vfs_impl.h](#), 2801
[l4/l4re_vfs/vfs.h](#), 2813
[l4/l4virtio/client/l4virtio](#), 2825
[l4/l4virtio/client/virtio-block](#), 2844
[l4/l4virtio/client/virtio-net](#), 2822
[l4/l4virtio/l4virtio](#), 2827
[l4/l4virtio/server/l4virtio](#), 2829
[l4/l4virtio/server/virtio](#), 2840
[l4/l4virtio/server/virtio-block](#), 2848
[l4/l4virtio/server/virtio-console](#), 2854
[l4/l4virtio/server/virtio-console-device](#), 2860
[l4/l4virtio/server/virtio-gpio-device](#), 2864
[l4/l4virtio/server/virtio-i2c-device](#), 2869
[l4/l4virtio/server/virtio-rng-device](#), 2874
[l4/l4virtio/server/virtio-scmi-device](#), 2876
[l4/l4virtio/server/virtio-spi-device](#), 2885
[l4/l4virtio/virtio.h](#), 2890
[l4/l4virtio/virtio_block.h](#), 2893
[l4/l4virtio/virtio_input.h](#), 2894
[l4/l4virtio/virtio_net.h](#), 2523
[l4/l4virtio/virtqueue](#), 2894
[l4/libedid/edid.h](#), 2899, 2900
[l4/re/c/dataspace.h](#), 2900, 2902
[l4/re/c/debug.h](#), 2484, 2485
[l4/re/c/dma_space.h](#), 2903, 2906
[l4/re/c/event.h](#), 2906, 2908
[l4/re/c/event_buffer.h](#), 2910
[l4/re/c/inhibitor.h](#), 2911, 2914
[l4/re/c/log.h](#), 2914, 2915
[l4/re/c/mem_alloc.h](#), 2915, 2916
[l4/re/c/namespace.h](#), 2917, 2918
[l4/re/c/parent.h](#), 2919, 2920
[l4/re/c/rm.h](#), 2921, 2922
[l4/re/c/util/cap_alloc.h](#), 2925, 2926
[l4/re/c/util/kumem_alloc.h](#), 2927, 2928
[l4/re/c/util/video/goos_fb.h](#), 2929
[l4/re/c/video/colors.h](#), 2930, 2932
[l4/re/c/video/goos.h](#), 2932, 2934
[l4/re/c/video/view.h](#), 2935, 2937
[l4/re/cap_alloc](#), 3004, 3005

l4/re/console, 2938
l4/re/consts, 3171, 3172
l4/re/consts.h, 3156, 3158
l4/re/dataspace, 2938, 2939
l4/re/dataspace-sys.h, 2941, 2942
l4/re/dbg_events, 2942
l4/re/debug, 3021, 3022
l4/re/dma_space, 2942, 2944
l4/re/elf_aux.h, 2945, 2946
l4/re/env, 2947, 2948
l4/re/env.h, 2949, 2951
l4/re/error_helper, 2953, 2955
l4/re/event, 3025
l4/re/event-sys.h, 2956
l4/re/event.h, 2909, 2910
l4/re/event_enums.h, 2956
l4/re/impl/dataspace_impl.h, 2963, 2964
l4/re/impl/mem_alloc_impl.h, 2965, 2966
l4/re/impl/namespace_impl.h, 2966, 2967
l4/re/impl/rm_impl.h, 2969, 2970
l4/re/inhibitor, 2971
l4/re/inhibitor-sys.h, 2971
l4/re/itas, 2972
l4/re/l4aux.h, 2973, 2974
l4/re/log, 2974, 2975
l4/re/log-sys.h, 2976
l4/re/mem_alloc, 2976, 2978
l4/re/mem_alloc-sys.h, 2979
l4/re/mmio_space, 2980, 2981
l4/re/namespace, 2981, 2983
l4/re/namespace-sys.h, 2984, 2985
l4/re/parent, 2985, 2986
l4/re/parent-sys.h, 2987
l4/re/protocols.h, 2987, 2988
l4/re/random, 2988, 2990
l4/re/remote_access, 2990
l4/re/rm, 2991, 2992
l4/re/rm-sys.h, 2996, 2997
l4/re/shared_cap, 3057, 3059
l4/re/unique_cap, 3065, 3067
l4/re/util/bitmap_cap_alloc, 2997, 2998
l4/re/util/br_manager, 2999
l4/re/util/cap, 3002, 3003
l4/re/util/cap_alloc, 3007, 3009
l4/re/util/cap_alloc_impl.h, 3010, 3012
l4/re/util/counting_cap_alloc, 3013, 3014
l4/re/util/dataspace_svr, 3018
l4/re/util/debug, 3022
l4/re/util/env_ns, 3024
l4/re/util/event, 3028
l4/re/util/event_buffer, 3030
l4/re/util/event_svr, 3031
l4/re/util/icu_svr, 3032
l4/re/util/item_alloc, 3035, 3036
l4/re/util/kumem_alloc, 3037, 3038
l4/re/util/meta, 3323
l4/re/util/name_space_svr, 3038
l4/re/util/object_registry, 3044
l4/re/util/poll_timeout_kipclock, 3046
l4/re/util/region_mapping, 3047, 3048
l4/re/util/region_mapping_svr, 3053
l4/re/util/reply_cap_hooks, 3056
l4/re/util/shared_cap, 3061, 3063
l4/re/util/unique_cap, 3067, 3069
l4/re/util/vcon_svr, 3069
l4/re/util/video/goos_fb, 3070
l4/re/util/video/goos_svr, 3072
l4/re/video/colors, 3074
l4/re/video/goos, 3075
l4/re/video/goos-sys.h, 3078, 3079
l4/re/video/view, 3080
l4/shmc/ringbuf.h, 3080, 3087
l4/shmc/shmc.h, 3089, 3092
l4/sigma0/sigma0.h, 3094, 3096
l4/sys/__kernel_object_impl.h, 3098
l4/sys/__ktrace-impl.h, 3098, 3100
l4/sys/__l4_fpage.h, 3101
l4/sys/__platform_control-arm.h, 3105
l4/sys/__task-arm.h, 3106
l4/sys/__timeout.h, 3106
l4/sys/__typeinfo.h, 3109, 3111
l4/sys/__vcpu-arm.h, 3121
l4/sys/__vm-svm.h, 3122
l4/sys/__vm-vmx.h, 3124
l4/sys/arm_smccc, 3130, 3132
l4/sys/arm_smccc.h, 3132, 3134
l4/sys/assert.h, 3413, 3415
l4/sys/cache.h, 3141, 3142
l4/sys/capability, 3146, 3148
l4/sys/compiler.h, 3149, 3151
l4/sys/consts.h, 3158, 3163
l4/sys/cxx/capability.h, 3167
l4/sys/cxx/consts, 3172
l4/sys/cxx/ipc_array, 3173
l4/sys/cxx/ipc_basics, 3176
l4/sys/cxx/ipc_client, 3180, 3182
l4/sys/cxx/ipc_epiface, 3183
l4/sys/cxx/ipc_iface, 3188, 3195
l4/sys/cxx/ipc_legacy, 3200
l4/sys/cxx/ipc_ret_array, 3200
l4/sys/cxx/ipc_server, 3203
l4/sys/cxx/ipc_server_loop, 3208
l4/sys/cxx/ipc_string, 3212
l4/sys/cxx/ipc_types, 3213, 3215
l4/sys/cxx/ipc_varg, 3222
l4/sys/cxx/limits, 3227
l4/sys/cxx/smart_capability_1x, 3229, 3230
l4/sys/cxx/types, 3232, 3234
l4/sys/debugger, 3238, 3240
l4/sys/debugger.h, 3241, 3243
l4/sys/err.h, 3247, 3248
l4/sys/exception, 3249, 3250
l4/sys/factory, 3250, 3252
l4/sys/factory.h, 3254, 3256
l4/sys/icu, 3260, 3261
l4/sys/icu.h, 3262, 3265

l4/sys/iommu, 3268
 l4/sys/ipc.h, 3270, 3273
 l4/sys/ipc_gate, 3279, 3281
 l4/sys/ipc_gate.h, 3281, 3283
 l4/sys/irq, 3284, 3286
 l4/sys/irq.h, 2779, 2781
 l4/sys/kdebug.h, 3288, 3303
 l4/sys/kdump.h, 3305, 3307
 l4/sys/kernel_object.h, 3307, 3308
 l4/sys/kip, 3309, 3310
 l4/sys/kip.h, 3458, 3461
 l4/sys/kobject, 3312
 l4/sys/ktrace.h, 3312, 3314
 l4/sys/l4int.h, 3317, 3318
 l4/sys/memdesc.h, 3320, 3322
 l4/sys/meta, 3324, 3325
 l4/sys/obj_info.h, 3325, 3328
 l4/sys/pager, 3329, 3331
 l4/sys/platform_control, 3331, 3333
 l4/sys/platform_control.h, 3334, 3335
 l4/sys/rcv_endpoint, 3338, 3339
 l4/sys/rcv_endpoint.h, 3340, 3341
 l4/sys/scheduler, 3342, 3343
 l4/sys/scheduler.h, 3344, 3347
 l4/sys/semaphore, 3349, 3351
 l4/sys/semaphore.h, 3351, 3353
 l4/sys/smart_capability, 3354, 3355
 l4/sys/snd_destination, 3357, 3359
 l4/sys/snd_destination.h, 3359, 3360
 l4/sys/task, 3360, 3362
 l4/sys/task.h, 3363, 3365
 l4/sys/thread, 3370, 3371
 l4/sys/thread.h, 2614, 2616
 l4/sys/thread_group, 3373
 l4/sys/thread_group.h, 3374, 3375
 l4/sys/typeinfo_svr, 3376, 3377
 l4/sys/types.h, 3378, 3381
 l4/sys/utcb.h, 3391, 3393
 l4/sys/vcon, 3400, 3402
 l4/sys/vcon.h, 3402, 3405
 l4/sys/vcpu, 3515, 3518
 l4/sys/vcpu_context, 3408, 3409
 l4/sys/vcpu_context.h, 3409
 l4/sys/vm, 3409, 3411
 l4/umalloc/umalloc.h, 3411, 3413
 l4/util/assert.h, 3416, 3417
 l4/util/atomic.h, 3421, 3424
 l4/util/backtrace.h, 3428, 3430
 l4/util/base64.h, 3430, 3432
 l4/util/bitops.h, 3432, 3434
 l4/util/elf.h, 3437, 3444
 l4/util/getopt.h, 3454, 3455
 l4/util/keymap.h, 3456, 3457
 l4/util/kip.h, 3463
 l4/util/kprintf.h, 3464, 3465
 l4/util/l4_macros.h, 2609, 2610
 l4/util/l4mod.h, 3465, 3467
 l4/util/list_alloc.h, 3467, 3470
 l4/util/lock.h, 3470, 3471
 l4/util/mb_info.h, 3472, 3476
 l4/util/parse_cmd.h, 3480, 3481
 l4/util/printf_helpers.h, 3482
 l4/util/rand.h, 3482, 3483
 l4/util/splitlog2.h, 3483, 3484
 l4/util/thread.h, 2623, 2625
 l4/util/util.h, 3485
 l4/utrace/ring_buffer, 3486, 3487
 l4/utrace/utrace, 3491, 3492
 l4/vbus/vbus, 3494
 l4/vbus/vbus.h, 3495, 3498
 l4/vbus/vbus_generic, 3499
 l4/vbus/vbus_gpio, 3499
 l4/vbus/vbus_gpio-ops.h, 3501
 l4/vbus/vbus_gpio.h, 3501
 l4/vbus/vbus_i2c.h, 3502
 l4/vbus/vbus_inhibitor.h, 3502
 l4/vbus/vbus_interfaces.h, 3502, 3506
 l4/vbus/vbus_mcspi.h, 3506
 l4/vbus/vbus_pci, 3506
 l4/vbus/vbus_pci-ops.h, 3507
 l4/vbus/vbus_pci.h, 3508
 l4/vbus/vbus_pm-ops.h, 3508
 l4/vbus/vbus_pm.h, 3509
 l4/vbus/vbus_types.h, 3509, 3512
 l4/vbus/vdevice-ops.h, 3513
 l4/vcpu/vcpu, 3513, 3514
 l4/vcpu/vcpu.h, 3518, 3520
 L4::Alloc_list, 1027
 L4::Arm_smccc, 1028
 call, 1028
 L4::Base_exception, 1030
 L4::Basic_registry, 1032
 dispatch, 1033
 find, 1034
 L4::Bounds_error, 1035
 L4::Cap< T >, 1038
 Cap, 1042
 check_castable_from, 1043
 check_convertible_from, 1043
 copy, 1043
 move, 1044
 L4::Cap_base, 1044
 cap, 1049
 Cap_base, 1047, 1048
 Cap_type, 1047
 copy, 1050
 fpage, 1051
 Invalid, 1047
 is_valid, 1052
 move, 1053
 No_init, 1047
 No_init_type, 1047
 snd_base, 1054
 validate, 1055, 1056
 L4::Com_error, 1057
 Com_error, 1060

- L4::Debugger, 1060
 - add_image_info, 1064
 - get_object_name, 1064
 - global_id, 1065
 - kobj_to_id, 1066
 - query_log_name, 1066
 - query_log_typeid, 1067
 - query_object_name, 1068
 - set_object_name, 1069
 - switch_log, 1069
- L4::Element_already_exists, 1070
- L4::Element_not_found, 1074
- L4::Epiface, 1077
 - dispatch, 1078
 - get_buffer_demand, 1079
 - obj_cap, 1079
 - server_iface, 1080
 - set_server, 1081
- L4::Epiface_t< Derived, IFACE, BASE, bool >, 1082
 - dispatch, 1085
- L4::Epiface_t0< RPC_IFACE, BASE >, 1085
 - obj_cap, 1088
- L4::Exception, 1088
 - exception, 1089
- L4::Exception_tracer, 1090
- L4::Factory, 1091
 - create, 1095, 1096
 - create_factory, 1097
 - create_gate, 1099
 - create_task, 1100
 - create_thread_group, 1101
- L4::Factory::Lstr, 1102
 - Lstr, 1103
- L4::Factory::Nil, 1104
- L4::Factory::S, 1104
 - ~S, 1106
 - operator l4_msgtag_t, 1107
 - operator<<, 1107, 1108
 - put, 1109, 1110
 - S, 1106
- L4::lcu, 1111
 - bind, 1115
 - info, 1116
 - mask, 1117
 - msi_info, 1118
 - set_mode, 1119
 - unbind, 1120
- L4::lcu::Info, 1121
- L4::Invalid_capability, 1123
 - cap, 1126
 - Invalid_capability, 1125
- L4::io_pager, 1126
 - io_page_fault, 1129
- L4::lommu, 1129
 - bind, 1132
 - unbind, 1132
- L4::IOModifier, 1134
- L4::lpc, 756
 - buf_cp_in, 758
 - buf_cp_out, 758
 - buf_in, 759
 - make_cap, 759
 - make_cap_full, 760
 - make_cap_grant, 761
 - make_cap_rw, 761
 - make_cap_rws, 762
 - msg_ptr, 763
 - read, 763
 - str_cp_in, 763
- L4::lpc::Array< ELEM_TYPE, LEN_TYPE >, 1134
- L4::lpc::Array_in_buf< ELEM_TYPE, LEN_TYPE, MAX >, 1137
- L4::lpc::Array_ref< ELEM_TYPE, LEN_TYPE >, 1140
- L4::lpc::As_value< T >, 1141
- L4::lpc::Call, 1142
- L4::lpc::Call_t< RIGHTS >, 1143
- L4::lpc::Call_zero_send_timeout, 1144
- L4::lpc::Cap< T >, 1145
 - Cap, 1149
 - Cap_mask, 1149
 - from_ci, 1150
 - Grant, 1149
 - Map, 1149
 - Map_type, 1149
 - Map_type_mask, 1148
 - Rights_mask, 1148
- L4::lpc::Gen_fpage, 1150
 - lo, 1153
 - Memory, 1153
 - Obj, 1153
 - Special, 1153
 - Type, 1152
- L4::lpc::In_out< T >, 1153
- L4::lpc::lostream, 1154
 - call, 1158
 - get, 1159, 1160
 - lostream, 1158
 - put, 1160, 1161
 - reply_and_wait, 1161, 1162
 - reset, 1163
- L4::lpc::Istream, 1164
 - get, 1168, 1169
 - Istream, 1167
 - receive, 1170
 - reset, 1171
 - skip, 1172
 - tag, 1172
 - wait, 1173, 1174
- L4::lpc::Msg, 764
 - align_to, 766, 767
 - Br_bytes, 766
 - check_size, 767, 768
 - Item_bytes, 766
 - Item_words, 766
 - Mr_bytes, 766
 - Mr_words, 766

- msg_add, [769](#)
- msg_get, [770](#)
- Word_bytes, [766](#)
- L4::lpc::Msg::Clnt_val_ops< MTYPE, DIR, CLASS >, [1175](#)
- L4::lpc::Msg::Cls_buffer, [1176](#)
- L4::lpc::Msg::Cls_data, [1178](#)
- L4::lpc::Msg::Cls_item, [1179](#)
- L4::lpc::Msg::Dir_in, [1180](#)
- L4::lpc::Msg::Dir_out, [1181](#)
- L4::lpc::Msg::Do_in_data, [1182](#)
- L4::lpc::Msg::Do_in_items, [1183](#)
- L4::lpc::Msg::Do_out_data, [1184](#)
- L4::lpc::Msg::Do_out_items, [1185](#)
- L4::lpc::Msg::Do_rcv_buffers, [1187](#)
- L4::lpc::Msg::Elem< Array< A, LEN > >, [1189](#)
- L4::lpc::Msg::Elem< Array< A, LEN > & >, [1188](#)
- L4::lpc::Msg::Elem< Array_ref< A, LEN > & >, [1190](#)
- L4::lpc::Msg::False, [1191](#)
- L4::lpc::Msg::Is_valid_rpc_type< T >, [1193](#)
- L4::lpc::Msg::Svr_arg_pack< IPC_TYPE >, [1195](#)
- L4::lpc::Msg::Svr_val_ops< MTYPE, DIR, CLASS >, [1195](#)
- L4::lpc::Msg::True, [1196](#)
- L4::lpc::Msg_ptr< T >, [1199](#)
- Msg_ptr, [1200](#)
- L4::lpc::Opt< T >, [1200](#)
- L4::lpc::Ostream, [1202](#)
- put, [1205](#), [1206](#)
- send, [1206](#)
- tag, [1207](#)
- L4::lpc::Out< T >, [1208](#)
- L4::lpc::Rcv_fpage, [1209](#)
- io, [1213](#)
- mem, [1213](#)
- obj, [1214](#)
- Rcv_fpage, [1212](#)
- rcv_task, [1215](#)
- L4::lpc::Ret_array< T >, [1216](#)
- L4::lpc::Send_only, [1217](#)
- L4::lpc::Small_buf, [1218](#)
- Small_buf, [1219](#)
- L4::lpc::Snd_fpage, [1220](#)
- Buffered, [1223](#)
- Cached, [1223](#)
- Cacheopt, [1223](#)
- cap_received, [1226](#)
- Compound, [1224](#)
- Continue, [1223](#)
- Grant, [1224](#)
- id_received, [1226](#)
- io, [1226](#)
- is_compound, [1227](#)
- Last, [1224](#)
- local_id_received, [1227](#)
- Map, [1224](#)
- Map_type, [1224](#)
- mem, [1227](#)
- More, [1224](#)
- None, [1223](#)
- obj, [1228](#)
- Single, [1224](#)
- Snd_fpage, [1224](#), [1225](#)
- Uncached, [1223](#)
- L4::lpc::Str_cp_in< T >, [1230](#)
- Str_cp_in, [1231](#)
- L4::lpc::Varg, [1231](#)
- data, [1233](#)
- get_value, [1233](#)
- is_nil, [1233](#)
- is_of, [1234](#)
- is_of_int, [1235](#)
- length, [1235](#)
- tag, [1236](#)
- type, [1236](#)
- value, [1236](#)
- L4::lpc::Varg_list< MAX >, [1237](#)
- L4::lpc::Varg_list_ref, [1240](#)
- Varg_list_ref, [1242](#)
- L4::lpc::Varg_list_ref::Iterator, [1242](#)
- L4::lpc_gate, [1244](#)
- get_infos, [1248](#)
- L4::lpc_svr, [771](#)
- L4::lpc_svr::Br_manager_no_buffers, [1249](#)
- alloc_buffer_demand, [1253](#)
- L4::lpc_svr::Compound_reply, [1254](#)
- L4::lpc_svr::Dbg_dispatch< R, Exc, Printer >, [1255](#)
- L4::lpc_svr::Default_loop_hooks, [1257](#)
- L4::lpc_svr::Default_setup_wait, [1260](#)
- L4::lpc_svr::Default_timeout, [1261](#)
- L4::lpc_svr::Direct_dispatch< R >, [1263](#)
- L4::lpc_svr::Direct_dispatch< R * >, [1265](#)
- L4::lpc_svr::Exc_dispatch< R, Exc >, [1267](#)
- L4::lpc_svr::Ignore_errors, [1269](#)
- L4::lpc_svr::Server_iface, [1270](#)
- add_timeout, [1273](#)
- alloc_buffer_demand, [1273](#)
- get_rcv_cap, [1274](#)
- get_rcv_mem, [1274](#)
- rcv_cap, [1275](#)
- realloc_rcv_cap, [1276](#)
- remove_timeout, [1277](#)
- take_reply_cap, [1277](#)
- L4::lpc_svr::Server_iface::Mem_window, [1278](#)
- L4::lpc_svr::Timeout, [1278](#)
- expired, [1281](#)
- timeout, [1281](#)
- L4::lpc_svr::Timeout_queue, [1282](#)
- add, [1283](#)
- handle_expired_timeouts, [1283](#)
- next_timeout, [1283](#)
- remove, [1284](#)
- timeout_expired, [1284](#)
- L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >, [1285](#)
- add_timeout, [1289](#)

- remove_timeout, 1289
- L4::Irq, 1290
 - bind_vcpu, 1296
 - detach, 1297
 - receive, 1298
 - unmask, 1298
 - wait, 1299
- L4::Irq_eoi, 1300
 - unmask, 1302
- L4::Irq_handler_object, 1303
- L4::Irqp_t< Derived, BASE, bool >, 1307
 - dispatch, 1310
 - obj_cap, 1310
- L4::Kip::Mem_desc, 1311
 - all, 1316
 - Arch, 1314
 - Arch_acpi_nvs, 1314
 - Arch_acpi_tables, 1314
 - Arch_sub_type_common, 1313
 - Bootloader, 1314
 - Conventional, 1314
 - count, 1316, 1317
 - Dedicated, 1314
 - end, 1318
 - first, 1318
 - Info, 1314
 - Info_acpi_rsd, 1314
 - Info_sub_type, 1314
 - is_virtual, 1319
 - Mem_desc, 1315
 - Mem_type, 1314
 - Reserved, 1314
 - Reserved_heap, 1314
 - Reserved_kernel, 1314
 - Reserved_mmio, 1314
 - set, 1320
 - Shared, 1314
 - size, 1320
 - start, 1321
 - sub_type, 1321
 - type, 1322
 - Undefined, 1314
- L4::Kobject, 1322
 - cap, 1323
 - dec_refcnt, 1325
- L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >, 1326
 - __iface, 1328
 - __iface_list, 1328
 - __check_protocols__, 1329
 - c, 1329
 - Class, 1329
- L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >, 1330
 - __iface, 1332
 - __iface_list, 1332
 - __check_protocols__, 1333
 - c, 1333
 - Class, 1332
- L4::Kobject_demand< T >, 1333
- L4::Kobject_t< Derived, Base, PROTO, S_DEMAND >, 1334
- L4::Kobject_typeid< T >, 1336
 - Demand, 1337
 - demand, 1338
 - id, 1338
 - proto_dispatch, 1338
- L4::Kobject_typeid< void >, 1340
 - demand, 1341
 - id, 1342
 - proto_dispatch, 1342
- L4::Kobject_x< Derived, ARGS >, 1343
- L4::Lock_guard, 1344
 - ~Lock_guard, 1346
 - Lock_guard, 1345
 - operator=, 1346
 - status, 1346
- L4::Meta, 1347
 - interface, 1351
 - num_interfaces, 1351
 - supports, 1352
- L4::Out_of_memory, 1353
- L4::Pager, 1357
 - page_fault, 1359
- L4::Platform_control, 1360
 - cpu_allow_shutdown, 1364
 - cpu_disable, 1364
 - cpu_enable, 1365
 - system_shutdown, 1366
 - system_suspend, 1367
- L4::Poll_timeout_counter, 1368
 - Poll_timeout_counter, 1370
 - set, 1370
 - timed_out, 1370
- L4::Poll_timeout_kipclock, 1371
 - Poll_timeout_kipclock, 1372
 - set, 1373
 - test, 1373
 - timed_out, 1374
- L4::Proto_t< P >, 1375
- L4::Rcv_endpoint, 1375
 - bind_snd_destination, 1379
 - bind_thread, 1379
- L4::Registry_iface, 1381
 - register_irq_obj, 1383
 - register_obj, 1383, 1384
 - unregister_obj, 1385
- L4::Reply_cap, 1386
 - ~Reply_cap, 1387
 - get, 1388
 - reply, 1388
 - Reply_cap, 1387
 - reset, 1388
- L4::Reply_cap_alloc, 1390
 - alloc, 1390
 - free, 1391

- L4::Reply_cap_idx, [1392](#)
- L4::Runtime_error, [1393](#)
 - err_no, [1396](#)
 - extra_str, [1396](#)
 - Runtime_error, [1395](#)
- L4::Scheduler, [1397](#)
 - idle_time, [1402](#)
 - info, [1402](#)
 - is_online, [1403](#)
 - run_thread, [1404](#)
- L4::Semaphore, [1406](#)
 - down, [1409](#)
 - up, [1410](#)
- L4::Server< LOOP_HOOKS >, [1411](#)
 - internal_loop, [1416](#)
 - loop, [1416](#)
 - loop_dbg, [1417](#)
 - Server, [1415](#)
- L4::Server_object, [1417](#)
 - dispatch, [1420](#), [1421](#)
- L4::Server_object_t< IFACE, BASE >, [1422](#)
 - dispatch_meta_request, [1426](#)
 - get_buffer_demand, [1426](#)
 - proto_dispatch, [1426](#)
- L4::Server_object_x< Derived, IFACE, BASE >, [1427](#)
- L4::Smart_cap< T, SMART >, [1431](#)
 - Smart_cap, [1435](#)
- L4::String, [1435](#)
- L4::Task, [1436](#)
 - add_ku_mem, [1440](#)
 - cap_equal, [1441](#)
 - cap_valid, [1442](#)
 - delete_obj, [1443](#)
 - map, [1444](#)
 - release_cap, [1445](#)
 - unmap, [1446](#)
 - unmap_batch, [1447](#)
- L4::Thread, [1449](#)
 - control, [1452](#)
 - ex_regs, [1453](#), [1454](#)
 - modify_senders, [1455](#)
 - register_del_irq, [1456](#)
 - register_doorbell_irq, [1457](#)
 - stats_time, [1458](#)
 - switch_to, [1459](#)
 - vcpu_control, [1460](#)
 - vcpu_control_ext, [1461](#)
 - vcpu_resume_commit, [1462](#)
 - vcpu_resume_start, [1463](#)
- L4::Thread::Attr, [1464](#)
 - alien, [1466](#)
 - Attr, [1466](#)
 - bind, [1466](#)
 - exc_handler, [1467](#)
 - pager, [1468](#)
- L4::Thread::Modify_senders, [1470](#)
 - add, [1471](#)
- L4::Thread_group, [1472](#)
 - add, [1474](#)
 - remove, [1475](#)
- L4::Triggerable, [1476](#)
 - trigger, [1479](#)
- L4::Type_info, [1480](#)
- L4::Type_info::Demand, [1481](#)
 - Demand, [1484](#)
 - no_demand, [1484](#)
- L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS >, [1485](#)
 - Caps, [1488](#)
 - Flags, [1488](#)
 - Mem, [1488](#)
 - Ports, [1488](#)
- L4::Type_info::Demand_union_t< D1, D2 >, [1488](#)
- L4::Typeid, [772](#)
- L4::Typeid::Detail::_Rpc< OPCODE, O, Default_op< R > >::Rpc< Y >, [1493](#)
- L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... >, [1493](#)
- L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... >::Rpc< Y >, [1496](#)
- L4::Typeid::Detail::_Rpc< OPCODE, O, X >, [1492](#)
- L4::Typeid::Detail::Rpc_end, [1497](#)
- L4::Typeid::P_dispatch< LIST >, [1497](#)
- L4::Typeid::Raw_ipc< CLASS >, [1498](#)
- L4::Typeid::Rpc_nocode< OPERATION >, [1500](#)
- L4::Typeid::Rpc< RPCS >, [1502](#)
- L4::Typeid::Rpc_code< OPCODE_TYPE >, [1504](#)
- L4::Typeid::Rpc_code< OPCODE_TYPE >::F< RPCS >, [1505](#)
- L4::Typeid::Rpc_sys< ARG >, [1508](#)
- L4::Types, [772](#)
 - declval, [774](#)
- L4::Types::__Add_rvalue_reference_helper< T, type-name >, [1510](#)
- L4::Types::__Add_rvalue_reference_helper< T, Void< T && > >, [1511](#)
- L4::Types::__Underlying_type_helper< T, bool >, [1513](#)
- L4::Types::__Underlying_type_helper< T, false >, [1514](#)
- L4::Types::Add_rvalue_reference< T >, [1516](#)
- L4::Types::Bool< V >, [1517](#)
- L4::Types::False, [1518](#)
- L4::Types::Flags< BITS_ENUM, UNDERLYING >, [1520](#)
 - clear, [1524](#)
 - Flags, [1523](#)
 - from_raw, [1524](#)
 - None, [1523](#)
 - None_type, [1523](#)
- L4::Types::Flags_ops_t< DT >, [1524](#)
- L4::Types::Flags_t< DT, T >, [1527](#)
- L4::Types::Int_for_size< SIZE, bool >, [1531](#)
- L4::Types::Int_for_type< T >, [1531](#)
- L4::Types::Integral_constant< T, Value >, [1532](#)
- L4::Types::Is_enum< T >, [1534](#)
- L4::Types::Same< A, B >, [1536](#)
- L4::Types::Same_template< T, Template >, [1538](#)

- L4::Types::True, [1541](#)
- L4::Types::Underlying_type< T >, [1543](#)
- L4::Uart, [1544](#)
 - change_mode, [1547](#)
 - char_avail, [1547](#)
 - enable_rx_irq, [1547](#)
 - generic_write, [1548](#)
 - get_char, [1548](#)
 - mode, [1549](#)
 - rate, [1549](#)
 - reg_shift, [1549](#)
 - shutdown, [1550](#)
 - startup, [1550](#)
 - write, [1550](#)
- L4::Uart_apb, [1551](#)
 - change_mode, [1555](#)
 - char_avail, [1555](#)
 - enable_rx_irq, [1555](#)
 - get_char, [1556](#)
 - shutdown, [1556](#)
 - startup, [1556](#)
 - write, [1557](#)
- L4::Unknown_error, [1558](#)
- L4::Vcon, [1560](#)
 - get_attr, [1565](#)
 - read, [1565](#)
 - read_with_flags, [1566](#)
 - send, [1567](#)
 - set_attr, [1568](#)
 - write, [1569](#)
- L4::Vm, [1570](#)
- l4_addr_consts_t
 - Memory related, [203](#)
- l4_align_stack_for_direct_fncall
 - Basic Macros, [156](#)
- L4_AMD64_SEGMENT_FS
 - segment.h, [2527](#)
- L4_AMD64_SEGMENT_GS
 - segment.h, [2527](#)
- l4_arm_smccc_call
 - arm_smccc.h, [3133](#)
- l4_assert
 - assert.h, [3415](#)
- L4_BASE_ARM_SMCCC_CAP
 - Capabilities, [447](#)
- L4_BASE_CAPS_LAST
 - Capabilities, [447](#)
- L4_BASE_DEBUGGER_CAP
 - Capabilities, [447](#)
- L4_BASE_FACTORY_CAP
 - Capabilities, [447](#)
- L4_BASE_ICU_CAP
 - Capabilities, [447](#)
- L4_BASE_IOMMU_CAP
 - Capabilities, [447](#)
- L4_BASE_LOG_CAP
 - Capabilities, [447](#)
- L4_BASE_PAGER_CAP
 - Capabilities, [447](#)
- L4_BASE_SCHEDULER_CAP
 - Capabilities, [447](#)
- L4_BASE_TASK_CAP
 - Capabilities, [447](#)
- L4_BASE_THREAD_CAP
 - Capabilities, [447](#)
- l4_bdr
 - Buffer Registers (BRs), [274](#)
- L4_BDR_IO_SHIFT
 - Buffer Registers (BRs), [274](#)
- L4_BDR_MEM_SHIFT
 - Buffer Registers (BRs), [274](#)
- L4_BDR_OBJ_SHIFT
 - Buffer Registers (BRs), [274](#)
- l4_buf_regs_t, [1574](#)
 - bdr, [1575](#)
- l4_buffer_desc_consts_t
 - Buffer Registers (BRs), [274](#)
- l4_busy_wait_ns
 - Timestamp Counter, [663](#)
- l4_busy_wait_us
 - Timestamp Counter, [664](#)
- l4_bytes_to_mwords
 - Memory related, [203](#)
- l4_cache_clean_data
 - Cache Consistency, [195](#)
- l4_cache_coherent
 - Cache Consistency, [196](#)
- l4_cache_dma_coherent
 - Cache Consistency, [196](#)
- l4_cache_flush_data
 - Cache Consistency, [197](#)
- l4_cache_inv_data
 - Cache Consistency, [197](#)
- l4_calibrate_tsc
 - Timestamp Counter, [665](#)
- L4_CAP_FPAGE_D
 - Flexpages, [179](#)
- L4_CAP_FPAGE_R
 - Flexpages, [178](#)
- L4_cap_fpage_rights
 - Flexpages, [178](#)
- L4_CAP_FPAGE_RO
 - Flexpages, [178](#)
- L4_CAP_FPAGE_RS
 - Flexpages, [179](#)
- L4_CAP_FPAGE_RSD
 - Flexpages, [179](#)
- L4_CAP_FPAGE_RW
 - Flexpages, [179](#)
- L4_CAP_FPAGE_RWD
 - Flexpages, [179](#)
- L4_CAP_FPAGE_RWS
 - Flexpages, [179](#)
- L4_CAP_FPAGE_RWSD
 - Flexpages, [179](#)
- L4_CAP_FPAGE_S

- Flexpages, [178](#)
- L4_CAP_FPAGE_W
 - Flexpages, [178](#)
- l4_cap_idx_t
 - Capabilities, [446](#)
- L4_CAP_SIZE
 - consts.h, [3160](#)
- l4_capability_equal
 - Capabilities, [447](#)
- l4_capability_next
 - types.h, [3381](#)
- l4_debugger_add_image_info
 - Kernel Debugger, [162](#)
- l4_debugger_get_object_name
 - Kernel Debugger, [163](#)
- l4_debugger_global_id
 - Kernel Debugger, [164](#)
- L4_DEBUGGER_GLOBAL_ID_GET_NAME_OP
 - debugger.h, [3243](#)
- L4_DEBUGGER_KOBJ_GET_GLOBAL_ID_OP
 - debugger.h, [3243](#)
- L4_DEBUGGER_KOBJ_GET_NAME_OP
 - debugger.h, [3243](#)
- L4_DEBUGGER_KOBJ_PTR_GET_GLOBAL_ID_OP
 - debugger.h, [3243](#)
- L4_DEBUGGER_KOBJ_SET_NAME_OP
 - debugger.h, [3243](#)
- l4_debugger_kobj_to_id
 - Kernel Debugger, [164](#)
- L4_DEBUGGER_LOG_QUERY_NAME_OP
 - debugger.h, [3243](#)
- L4_DEBUGGER_LOG_QUERY_TYPEID_OP
 - debugger.h, [3243](#)
- L4_DEBUGGER_LOG_SWITCH_OP
 - debugger.h, [3243](#)
- L4_DEBUGGER_OBJ_INFO_OP
 - debugger.h, [3243](#)
- l4_debugger_query_log_name
 - Kernel Debugger, [165](#)
- l4_debugger_query_log_typeid
 - Kernel Debugger, [166](#)
- l4_debugger_query_obj_infos
 - obj_info.h, [3327](#)
- l4_debugger_query_object_name
 - Kernel Debugger, [167](#)
- l4_debugger_set_object_name
 - Kernel Debugger, [168](#)
- l4_debugger_switch_log
 - Kernel Debugger, [168](#)
- L4_DEBUGGER_TASK_ADD_IMAGE_INFO_OP
 - debugger.h, [3243](#)
- l4_default_caps_t
 - Capabilities, [447](#)
- L4_DISABLE_COPY
 - capability, [3147](#)
- L4_E2BIG
 - Error codes, [209](#)
- L4_EACCESS
 - Error codes, [209](#)
- Error codes, [209](#)
- L4_EADDRNOTAVAIL
 - Error codes, [209](#)
- L4_EAGAIN
 - Error codes, [209](#)
- L4_EBADPROTO
 - Error codes, [209](#)
- L4_EBUSY
 - Error codes, [209](#)
- L4_EDROPREPLY
 - Error codes, [210](#)
- L4_EEXIST
 - Error codes, [209](#)
- L4_EFAULT
 - Error codes, [209](#)
- L4_EINVAL
 - Error codes, [209](#)
- L4_EIO
 - Error codes, [209](#)
- L4_EIPC_HI
 - Error codes, [210](#)
- L4_EIPC_LO
 - Error codes, [210](#)
- L4_EMMSGERRRANGE
 - Error codes, [210](#)
- L4_EMMSGMISSARG
 - Error codes, [210](#)
- L4_EMMSGTOOLONG
 - Error codes, [210](#)
- L4_EMMSGTOOSHORT
 - Error codes, [209](#)
- L4_ENAMETOOLONG
 - Error codes, [209](#)
- L4_ENODEV
 - Error codes, [209](#)
- L4_ENOENT
 - Error codes, [209](#)
- L4_ENOMEM
 - Error codes, [209](#)
- L4_ENOREPLY
 - Error codes, [209](#)
- L4_ENOSPC
 - Error codes, [209](#)
- L4_ENOSYS
 - Error codes, [209](#)
- L4_ENOTDIR
 - Error codes, [209](#)
- L4_ENXIO
 - Error codes, [209](#)
- L4_EOK
 - Error codes, [209](#)
- L4_EPERM
 - Error codes, [209](#)
- L4_ERANGE
 - Error codes, [209](#)
- L4_ERRNOMAX
 - Error codes, [209](#)
- l4_error

- Error Handling, [247](#)
- `l4_error_code_t`
 - Error codes, [209](#)
- `l4_exc_regs_t`, [1575](#)
 - flags, [1578](#)
 - ss, [1578](#)
- `L4_EXPORT`
 - Basic Macros, [154](#)
- `l4_factory_create`
 - Factory, [290](#)
- `l4_factory_create_factory`
 - Factory, [291](#)
- `l4_factory_create_gate`
 - Factory, [292](#)
- `l4_factory_create_irq`
 - Factory, [293](#)
- `l4_factory_create_task`
 - Factory, [294](#)
- `l4_factory_create_thread`
 - Factory, [295](#)
- `l4_factory_create_thread_group`
 - Factory, [296](#)
- `l4_factory_create_vcpu_context`
 - Factory, [297](#)
- `l4_factory_create_vm`
 - Factory, [298](#)
- `L4_FP_ALL_SPACES`
 - Task, [371](#)
- `L4_FP_DELETE_OBJ`
 - Task, [371](#)
- `L4_FP_OTHER_SPACES`
 - Task, [371](#)
- `l4_fpage`
 - Flexpages, [181](#)
- `L4_FPAGE_ADDR_BITS`
 - Flexpages, [180](#)
- `L4_FPAGE_ADDR_SHIFT`
 - Flexpages, [180](#)
- `l4_fpage_all`
 - Flexpages, [182](#)
- `L4_FPAGE_BUFFERABLE`
 - Message Items, [231](#)
- `L4_FPAGE_C_IPCGATE_SVR`
 - Attributes and additional permissions for object send items, [236](#)
- `L4_FPAGE_CACHE_OPT`
 - Message Items, [231](#)
- `l4_fpage_cacheability_opt_t`
 - Message Items, [231](#)
- `L4_FPAGE_CACHEABLE`
 - Message Items, [231](#)
- `L4_fpage_consts`
 - Flexpages, [179](#)
- `l4_fpage_contains`
 - Flexpages, [182](#)
- `L4_fpage_control`
 - Flexpages, [180](#)
- `L4_FPAGE_CONTROL_MASK`
 - Flexpages, [180](#)
- `L4_FPAGE_CONTROL_OFFSET_SHIFT`
 - Flexpages, [180](#)
- `l4_fpage_invalid`
 - Flexpages, [183](#)
- `L4_FPAGE_IO`
 - Flexpages, [181](#)
- `l4_fpage_ioport`
 - Flexpages, [183](#)
- `l4_fpage_max_order`
 - Flexpages, [184](#)
- `l4_fpage_memaddr`
 - Flexpages, [185](#)
- `L4_FPAGE_MEMORY`
 - Flexpages, [181](#)
- `L4_FPAGE_OBJ`
 - Flexpages, [181](#)
- `l4_fpage_obj`
 - Flexpages, [185](#)
- `l4_fpage_page`
 - Flexpages, [186](#)
- `L4_fpage_rights`
 - Flexpages, [180](#)
- `l4_fpage_rights`
 - Flexpages, [186](#)
- `L4_FPAGE_RIGHTS_ALL`
 - Flexpages, [180](#)
- `L4_FPAGE_RIGHTS_BITS`
 - Flexpages, [180](#)
- `L4_FPAGE_RIGHTS_MASK`
 - Flexpages, [180](#)
- `L4_FPAGE_RIGHTS_SHIFT`
 - Flexpages, [179](#)
- `L4_FPAGE_RO`
 - Flexpages, [180](#)
- `L4_FPAGE_RW`
 - Flexpages, [180](#)
- `L4_FPAGE_RWX`
 - Flexpages, [181](#)
- `L4_FPAGE_RX`
 - Flexpages, [181](#)
- `l4_fpage_set_rights`
 - Flexpages, [187](#)
- `l4_fpage_size`
 - Flexpages, [188](#)
- `L4_FPAGE_SIZE_BITS`
 - Flexpages, [180](#)
- `L4_FPAGE_SIZE_SHIFT`
 - Flexpages, [180](#)
- `L4_FPAGE_SPECIAL`
 - Flexpages, [181](#)
- `l4_fpage_t`, [1579](#)
- `L4_fpage_type`
 - Flexpages, [181](#)
- `l4_fpage_type`
 - Flexpages, [188](#)
- `L4_FPAGE_TYPE_BITS`
 - Flexpages, [180](#)

- L4_FPAGE_TYPE_SHIFT
 - Flexpages, [179](#)
- L4_FPAGE_UNCACHEABLE
 - Message Items, [231](#)
- L4_FPAGE_W
 - Flexpages, [180](#)
- L4_FPAGE_X
 - Flexpages, [180](#)
- l4_get_hz
 - Timestamp Counter, [665](#)
- L4_HIDDEN
 - Basic Macros, [154](#)
- l4_icu_bind
 - Interrupt controller, [324](#)
- l4_icu_bind_u
 - Interrupt controller, [325](#)
- L4_ICU_FLAG_MSI
 - Interrupt controller, [324](#)
- L4_icu_flags
 - Interrupt controller, [324](#)
- l4_icu_info
 - Interrupt controller, [326](#)
- l4_icu_info_t, [1579](#)
 - features, [1581](#)
 - Interrupt controller, [323](#)
- l4_icu_info_u
 - Interrupt controller, [327](#)
- l4_icu_mask
 - Interrupt controller, [328](#)
- l4_icu_mask_u
 - Interrupt controller, [329](#)
- l4_icu_msi_info
 - Interrupt controller, [330](#)
- l4_icu_msi_info_t, [1581](#)
- l4_icu_msi_info_u
 - Interrupt controller, [331](#)
- L4_ICU_OP_BIND
 - L4 IPC Opcodes, [480](#)
- L4_ICU_OP_INFO
 - L4 IPC Opcodes, [480](#)
- L4_ICU_OP_MASK
 - L4 IPC Opcodes, [480](#)
- L4_ICU_OP_MSI_INFO
 - L4 IPC Opcodes, [480](#)
- L4_ICU_OP_SET_MODE
 - L4 IPC Opcodes, [480](#)
- L4_ICU_OP_UNBIND
 - L4 IPC Opcodes, [480](#)
- L4_ICU_OP_UNMASK
 - L4 IPC Opcodes, [480](#)
- L4_icu_opcode
 - L4 IPC Opcodes, [479](#)
- l4_icu_set_mode
 - Interrupt controller, [332](#)
- l4_icu_set_mode_u
 - Interrupt controller, [333](#)
- l4_icu_unbind
 - Interrupt controller, [334](#)
- l4_icu_unbind_u
 - Interrupt controller, [335](#)
- l4_icu_unmask
 - Interrupt controller, [336](#)
- l4_icu_unmask_u
 - Interrupt controller, [337](#)
- l4_infinite_loop
 - Basic Macros, [157](#)
- L4_INLINE_RPC
 - ipc_iface, [3191](#)
- L4_INLINE_RPC_NF
 - ipc_iface, [3191](#)
- L4_INLINE_RPC_NF_OP
 - ipc_iface, [3191](#)
- L4_INLINE_RPC_OP
 - ipc_iface, [3192](#)
- L4_INVALID_ADDR
 - Memory related, [203](#)
- l4_iofpage
 - Flexpages, [189](#)
- L4_IOPORT_MAX
 - Flexpages, [178](#)
- l4_ipc
 - Object Invocation, [213](#)
- l4_ipc_call
 - Object Invocation, [214](#)
- L4_IPC_ENOT_EXISTENT
 - Error Handling, [246](#)
- l4_ipc_error
 - Error Handling, [248](#)
- l4_ipc_error_code
 - Error Handling, [249](#)
- L4_IPC_ERROR_MASK
 - Error Handling, [246](#)
- L4_IPC_GATE_BIND_OP
 - L4 IPC Opcodes, [480](#)
- L4_IPC_GATE_GET_INFO_OP
 - L4 IPC Opcodes, [480](#)
- l4_ipc_gate_get_infos
 - IPC-Gate API, [282](#)
- L4_ipc_gate_ops
 - L4 IPC Opcodes, [480](#)
- l4_ipc_is_rcv_error
 - Error Handling, [250](#)
- l4_ipc_is_snd_error
 - Error Handling, [250](#)
- L4_IPC_NO_REPLY_CAP
 - Error Handling, [247](#)
- L4_IPC_REABORTED
 - Error Handling, [247](#)
- L4_IPC_RECANCELED
 - Error Handling, [246](#)
- l4_ipc_receive
 - Object Invocation, [217](#)
- L4_IPC_REMAPFAILED
 - Error Handling, [246](#)
- L4_IPC_REMSGCUT
 - Error Handling, [247](#)

- `l4_ipc_reply`
 - Object Invocation, [219](#)
- `l4_ipc_reply_and_wait`
 - Object Invocation, [220](#)
- `L4_IPC_RERCVPFTO`
 - Error Handling, [246](#)
- `L4_IPC_RESNDPFTO`
 - Error Handling, [246](#)
- `L4_IPC_RETIMEOUT`
 - Error Handling, [246](#)
- `L4_IPC_SEABORTED`
 - Error Handling, [247](#)
- `L4_IPC_SECANCELED`
 - Error Handling, [246](#)
- `L4_IPC_SEMAPFAILED`
 - Error Handling, [246](#)
- `L4_IPC_SEMSGCUT`
 - Error Handling, [247](#)
- `l4_ipc_send`
 - Object Invocation, [221](#)
- `l4_ipc_send_and_wait`
 - Object Invocation, [223](#)
- `L4_IPC_SERCVPFTO`
 - Error Handling, [247](#)
- `L4_IPC_SESNDFPFTO`
 - Error Handling, [246](#)
- `L4_IPC_SETIMEOUT`
 - Error Handling, [246](#)
- `l4_ipc_sleep`
 - Object Invocation, [224](#)
- `l4_ipc_sleep_ms`
 - Object Invocation, [225](#)
- `l4_ipc_sleep_us`
 - Object Invocation, [226](#)
- `L4_IPC_SND_ERR_MASK`
 - Error Handling, [246](#)
- `l4_ipc_tcr_error_t`
 - Error Handling, [246](#)
- `l4_ipc_timeout`
 - Timeouts, [239](#)
- `L4_IPC_TIMEOUT_0`
 - Timeouts, [238](#)
- `l4_ipc_to_errno`
 - `ipc.h`, [3273](#)
- `l4_ipc_wait`
 - Object Invocation, [227](#)
- `l4_irq_bind_vcpu`
 - IRQs, [340](#)
- `l4_irq_bind_vcpu_u`
 - IRQs, [341](#)
- `l4_irq_detach`
 - IRQs, [343](#)
- `l4_irq_detach_u`
 - IRQs, [344](#)
- `L4_IRQ_F_BOTH`
 - IRQs, [340](#)
- `L4_IRQ_F_BOTH_EDGE`
 - IRQs, [340](#)
- `L4_IRQ_F_CLEAR_WAKEUP`
 - IRQs, [340](#)
- `L4_IRQ_F_EDGE`
 - IRQs, [340](#)
- `L4_IRQ_F_LEVEL`
 - IRQs, [340](#)
- `L4_IRQ_F_LEVEL_HIGH`
 - IRQs, [340](#)
- `L4_IRQ_F_LEVEL_LOW`
 - IRQs, [340](#)
- `L4_IRQ_F_MASK`
 - IRQs, [340](#)
- `L4_IRQ_F_NEG`
 - IRQs, [340](#)
- `L4_IRQ_F_NEG_EDGE`
 - IRQs, [340](#)
- `L4_IRQ_F_NONE`
 - IRQs, [340](#)
- `L4_IRQ_F_POS`
 - IRQs, [340](#)
- `L4_IRQ_F_POS_EDGE`
 - IRQs, [340](#)
- `L4_IRQ_F_SET_MODE`
 - IRQs, [340](#)
- `L4_IRQ_F_SET_WAKEUP`
 - IRQs, [340](#)
- `L4_irq_mode`
 - IRQs, [340](#)
- `l4_irq_receive`
 - IRQs, [345](#)
- `l4_irq_receive_u`
 - IRQs, [346](#)
- `l4_irq_trigger`
 - IRQs, [347](#)
- `l4_irq_trigger_u`
 - IRQs, [348](#)
- `l4_irq_unmask`
 - IRQs, [349](#)
- `l4_irq_unmask_u`
 - IRQs, [350](#)
- `l4_irq_wait`
 - IRQs, [351](#)
- `l4_irq_wait_u`
 - IRQs, [352](#)
- `l4_is_fpage_valid`
 - Flexpages, [190](#)
- `l4_is_fpage_writable`
 - Flexpages, [191](#)
- `l4_is_invalid_cap`
 - Capabilities, [448](#)
- `l4_is_valid_cap`
 - Capabilities, [448](#)
- `L4_ITEM_CONT`
 - Message Items, [231](#)
- `L4_ITEM_MAP`
 - Message Items, [231](#)
- `l4_kd_enter`
 - `kdebug.h`, [3295](#)

- `l4_kd_outchar`
 `kdebug.h`, [3296](#)
- `l4_kd_outdec`
 `kdebug.h`, [3297](#)
- `l4_kd_outhex12`
 `kdebug.h`, [3297](#)
- `l4_kd_outhex16`
 `kdebug.h`, [3298](#)
- `l4_kd_outhex20`
 `kdebug.h`, [3298](#)
- `l4_kd_outhex32`
 `kdebug.h`, [3299](#)
- `l4_kd_outhex64`
 `kdebug.h`, [3299](#)
- `l4_kd_outhex8`
 `kdebug.h`, [3300](#)
- `l4_kd_outnstring`
 `kdebug.h`, [3301](#)
- `l4_kd_outstring`
 `kdebug.h`, [3302](#)
- `l4_kd_outumword`
 `kdebug.h`, [3302](#)
- `l4_kdebug_ops_t`
 `kdebug.h`, [3290](#)
- `l4_kernel_info_get_mem_desc_end`
 Memory descriptors (C version), [443](#)
- `l4_kernel_info_get_mem_desc_is_virtual`
 Memory descriptors (C version), [443](#)
- `l4_kernel_info_get_mem_desc_start`
 Memory descriptors (C version), [443](#)
- `l4_kernel_info_get_mem_desc_subtype`
 Memory descriptors (C version), [443](#)
- `l4_kernel_info_get_mem_desc_type`
 Memory descriptors (C version), [444](#)
- `l4_kernel_info_get_num_mem_descs`
 Memory descriptors (C version), [444](#)
- `l4_kernel_info_mem_desc_t`, [1582](#)
 Memory descriptors (C version), [441](#)
- `l4_kernel_info_set_mem_desc`
 Memory descriptors (C version), [444](#)
- `l4_kernel_info_t`, [1583](#)
 Kernel Interface Page, [435](#)
- `l4_kernel_info_version_offset`
 Kernel Interface Page, [436](#)
- `l4_kip`
 Kernel Interface Page, [436](#)
- `l4_kip_clock`
 Kernel Interface Page, [436](#)
- `l4_kip_clock_lw`
 Kernel Interface Page, [437](#)
- `l4_kip_clock_ns`
 Kernel Interface Page, [438](#)
- `l4_kip_for_each_feature`
 `kip.h`, [3460](#)
- `l4_kip_kernel_has_feature`
 `kip.h`, [3460](#)
- `L4_KIP_OFFS_READ_NS`
 Kernel Interface Page, [435](#)
- `L4_KIP_OFFS_READ_US`
 Kernel Interface Page, [435](#)
- `l4_kip_version`
 Kernel Interface Page, [438](#)
- `l4_kip_version_string`
 Kernel Interface Page, [439](#)
- `L4_LOG2_PAGESIZE`
 Memory related, [200](#)
- `L4_LOG2_SUPERPAGESIZE`
 Memory related, [200](#)
- `l4_map_control`
 Message Items, [233](#)
- `L4_MAP_ITEM_GRANT`
 Message Items, [232](#)
- `L4_MAP_ITEM_MAP`
 Message Items, [232](#)
- `l4_map_obj_control`
 Message Items, [234](#)
- `l4_mem_archspecific_acpi_nvs`
 Memory descriptors (C version), [442](#)
- `l4_mem_archspecific_acpi_tables`
 Memory descriptors (C version), [442](#)
- `l4_mem_archspecific_sub_type_common_t`
 Memory descriptors (C version), [442](#)
- `l4_mem_info_acpi_rsdp`
 Memory descriptors (C version), [442](#)
- `l4_mem_info_sub_type_t`
 Memory descriptors (C version), [442](#)
- `L4_mem_op_widths`
 Memory Operations, [450](#)
- `l4_mem_read`
 Memory Operations, [451](#)
- `l4_mem_reserved_heap`
 Memory descriptors (C version), [442](#)
- `l4_mem_reserved_kernel`
 Memory descriptors (C version), [442](#)
- `l4_mem_reserved_mmio`
 Memory descriptors (C version), [442](#)
- `l4_mem_type_archspecific`
 Memory descriptors (C version), [443](#)
- `l4_mem_type_bootloader`
 Memory descriptors (C version), [442](#)
- `l4_mem_type_conventional`
 Memory descriptors (C version), [442](#)
- `l4_mem_type_dedicated`
 Memory descriptors (C version), [442](#)
- `l4_mem_type_info`
 Memory descriptors (C version), [442](#)
- `l4_mem_type_reserved`
 Memory descriptors (C version), [442](#)
- `l4_mem_type_shared`
 Memory descriptors (C version), [442](#)
- `l4_mem_type_t`
 Memory descriptors (C version), [442](#)
- `l4_mem_type_undefined`
 Memory descriptors (C version), [442](#)
- `L4_MEM_WIDTH_1BYTE`
 Memory Operations, [450](#)

- L4_MEM_WIDTH_2BYTE
 - Memory Operations, [450](#)
- L4_MEM_WIDTH_4BYTE
 - Memory Operations, [450](#)
- l4_mem_write
 - Memory Operations, [451](#)
- l4_msg_item_consts_t
 - Message Items, [231](#)
- l4_msg_regs_t, [1584](#)
- l4_msgtag
 - Message Tag, [254](#)
- L4_MSGTAG_ERROR
 - Message Tag, [253](#)
- L4_MSGTAG_FLAGS
 - Message Tag, [253](#)
- L4_msgtag_flags
 - Message Tag, [253](#)
- l4_msgtag_flags
 - Message Tag, [256](#)
- l4_msgtag_has_error
 - Message Tag, [257](#)
- l4_msgtag_is_exception
 - Message Tag, [258](#)
- l4_msgtag_is_io_page_fault
 - Message Tag, [259](#)
- l4_msgtag_is_page_fault
 - Message Tag, [260](#)
- l4_msgtag_is_sigma0
 - Message Tag, [261](#)
- l4_msgtag_items
 - Message Tag, [261](#)
- l4_msgtag_label
 - Message Tag, [262](#)
- L4_msgtag_protocol
 - Message Tag, [253](#)
- L4_MSGTAG_SCHEDULE
 - Message Tag, [253](#)
- l4_msgtag_t, [1585](#)
 - flags, [1587](#)
 - has_error, [1587](#)
- L4_MSGTAG_TRANSFER_FPU
 - Message Tag, [253](#)
- l4_msgtag_words
 - Message Tag, [263](#)
- L4_NOTHROW
 - Basic Macros, [155](#)
- l4_ns_to_tsc
 - Timestamp Counter, [665](#)
- l4_obj_fpage
 - Flexpages, [192](#)
- L4_PAGEMASK
 - Memory related, [200](#)
- L4_PAGESHIFT
 - Memory related, [200](#), [201](#)
- l4_platform_ctl_cpu_allow_shutdown
 - Platform Control C API, [354](#)
- L4_PLATFORM_CTL_CPU_ALLOW_SHUTDOWN_OP
 - L4 IPC Opcodes, [481](#)
- l4_platform_ctl_cpu_disable
 - Platform Control C API, [355](#)
- L4_PLATFORM_CTL_CPU_DISABLE_OP
 - L4 IPC Opcodes, [481](#)
- l4_platform_ctl_cpu_enable
 - Platform Control C API, [355](#)
- L4_PLATFORM_CTL_CPU_ENABLE_OP
 - L4 IPC Opcodes, [481](#)
- L4_platform_ctl_ops
 - L4 IPC Opcodes, [480](#)
- L4_platform_ctl_proto
 - Message Tag, [254](#)
- l4_platform_ctl_set_task_asid
 - Platform Control C API, [356](#)
- L4_PLATFORM_CTL_SET_TASK_ASID_OP
 - L4 IPC Opcodes, [481](#)
- L4_PLATFORM_CTL_SYS_SHUTDOWN_OP
 - L4 IPC Opcodes, [481](#)
- L4_PLATFORM_CTL_SYS_SUSPEND_OP
 - L4 IPC Opcodes, [481](#)
- l4_platform_ctl_system_shutdown
 - Platform Control C API, [357](#)
- l4_platform_ctl_system_suspend
 - Platform Control C API, [358](#)
- L4_PROTO_ALLOW_SYSCALL
 - Message Tag, [253](#)
- L4_PROTO_DEBUGGER
 - Message Tag, [254](#)
- L4_PROTO_DMA_SPACE
 - Message Tag, [254](#)
- L4_PROTO_EXCEPTION
 - Message Tag, [254](#)
- L4_PROTO_FACTORY
 - Message Tag, [254](#)
- L4_PROTO_IO_PAGE_FAULT
 - Message Tag, [254](#)
- L4_PROTO_IOMMU
 - Message Tag, [254](#)
- L4_PROTO_IRQ
 - Message Tag, [253](#)
- L4_PROTO_IRQ_SENDER
 - Message Tag, [254](#)
- L4_PROTO_KOBJECT
 - Message Tag, [254](#)
- L4_PROTO_LOG
 - Message Tag, [254](#)
- L4_PROTO_META
 - Message Tag, [254](#)
- L4_PROTO_NONE
 - Message Tag, [253](#)
- L4_PROTO_PAGE_FAULT
 - Message Tag, [253](#)
- L4_PROTO_PF_EXCEPTION
 - Message Tag, [253](#)
- L4_PROTO_PLATFORM_CTL
 - Message Tag, [254](#)
- L4_PROTO_SCHEDULER
 - Message Tag, [254](#)

- L4_PROTO_SEMAPHORE
 - Message Tag, [254](#)
- L4_PROTO_SIGMA0
 - Message Tag, [254](#)
- L4_PROTO_SMCCC
 - Message Tag, [254](#)
- l4_proto_t
 - types.h, [3380](#)
- L4_PROTO_TASK
 - Message Tag, [254](#)
- L4_PROTO_THREAD
 - Message Tag, [254](#)
- L4_PROTO_THREAD_GROUP
 - Message Tag, [254](#)
- L4_PROTO_VCPU_CONTEXT
 - Message Tag, [254](#)
- L4_PROTO_VM
 - Message Tag, [254](#)
- L4_RCV_EP_BIND_OP
 - rcv_endpoint.h, [3341](#)
- l4_rcv_ep_bind_snd_destination
 - IPC-Gate API, [283](#)
- l4_rcv_ep_bind_thread
 - IPC-Gate API, [284](#)
- L4_rcv_ep_ops
 - rcv_endpoint.h, [3341](#)
- L4_RCV_ITEM_FORWARD_MAPPINGS
 - Message Items, [232](#)
- L4_RCV_ITEM_LOCAL_ID
 - Message Items, [233](#)
- L4_RCV_ITEM_SINGLE_CAP
 - Message Items, [232](#)
- l4_rcv_timeout
 - Timeouts, [239](#)
- l4_rdpmc
 - Timestamp Counter, [666](#)
- l4_rdpmc_32
 - Timestamp Counter, [666](#)
- l4_rdtsc
 - Timestamp Counter, [667](#)
- l4_rdtsc_32
 - Timestamp Counter, [667](#)
- L4_REPLY_CAP_BIT
 - consts.h, [3160](#)
- l4_ret_t
 - types.h, [3380](#)
- l4_round_page
 - Memory related, [204](#)
- l4_round_size
 - Memory related, [205](#)
- L4_RPC
 - ipc_iface, [3192](#)
- L4_RPC_DEF
 - ipc_client, [3182](#)
- L4_RPC_NF
 - ipc_iface, [3193](#)
- L4_RPC_NF_OP
 - ipc_iface, [3193](#)
- L4_RPC_OP
 - ipc_iface, [3194](#)
- l4_sched_cpu_set
 - Scheduler, [361](#)
- l4_sched_cpu_set_t, [1588](#)
 - gran_offset, [1591](#)
 - granularity, [1589](#)
 - offset, [1589](#)
 - set, [1590](#)
- l4_sched_param
 - Scheduler, [362](#)
- l4_sched_param_t, [1592](#)
 - prio, [1593](#)
- L4_SCHEDULER_CLASS_FIXED_PRIO
 - Scheduler, [361](#)
- L4_SCHEDULER_CLASS_WFQ
 - Scheduler, [361](#)
- L4_scheduler_classes
 - Scheduler, [361](#)
- l4_scheduler_idle_time
 - Scheduler, [363](#)
- L4_SCHEDULER_IDLE_TIME_OP
 - Scheduler, [361](#)
- l4_scheduler_info
 - Scheduler, [364](#)
- L4_SCHEDULER_INFO_OP
 - Scheduler, [361](#)
- l4_scheduler_info_with_classes
 - Scheduler, [364](#)
- l4_scheduler_is_online
 - Scheduler, [365](#)
- L4_scheduler_ops
 - Scheduler, [361](#)
- l4_scheduler_run_thread
 - Scheduler, [366](#)
- L4_SCHEDULER_RUN_THREAD_OP
 - Scheduler, [361](#)
- l4_semaphore_down
 - Kernel-provided semaphore, [368](#)
- l4_semaphore_up
 - Kernel-provided semaphore, [368](#)
- l4_sleep
 - Utility Functions, [649](#)
- l4_snd_fpage_t, [1593](#)
- l4_snd_timeout
 - Timeouts, [239](#)
- l4_sndfpage_add
 - Object Invocation, [228](#)
- L4_SUPERPAGEMASK
 - Memory related, [201](#)
- L4_SUPERPAGESHIFT
 - Memory related, [201](#), [202](#)
- L4_SUPERPAGESIZE
 - Memory related, [202](#)
- L4_sys_segment
 - segment.h, [2527](#)
- L4_SYSF_CALL
 - consts.h, [3161](#)

- L4_SYSF_NONE
 - Object Invocation, [212](#)
- L4_SYSF_OPEN_WAIT
 - consts.h, [3161](#)
- L4_SYSF_RECV
 - consts.h, [3161](#)
- L4_SYSF_REPLY
 - consts.h, [3161](#)
- L4_SYSF_REPLY_AND_WAIT
 - consts.h, [3162](#)
- L4_SYSF_SEND
 - consts.h, [3162](#)
- L4_SYSF_SEND_AND_WAIT
 - consts.h, [3162](#)
- L4_SYSF_WAIT
 - consts.h, [3162](#)
- l4_task_add_ku_mem
 - Task, [371](#)
- L4_TASK_ADD_KU_MEM_OP
 - L4 IPC Opcodes, [481](#)
- l4_task_cap_equal
 - Task, [372](#)
- L4_TASK_CAP_INFO_OP
 - L4 IPC Opcodes, [481](#)
- l4_task_cap_valid
 - Task, [373](#)
- l4_task_delete_obj
 - Task, [374](#)
- L4_TASK_LDT_SET_X86_OP
 - L4 IPC Opcodes, [481](#)
- L4_task_ldt_x86_consts
 - segment.h, [2527](#), [2533](#)
- L4_TASK_LDT_X86_ENTRY_SIZE
 - segment.h, [2527](#), [2533](#)
- L4_TASK_LDT_X86_MAX_ENTRIES
 - segment.h, [2527](#), [2533](#)
- l4_task_map
 - Task, [375](#)
- L4_TASK_MAP_OP
 - L4 IPC Opcodes, [481](#)
- L4_TASK_MAP_VGICC_ARM_OP
 - L4 IPC Opcodes, [481](#)
- L4_task_ops
 - L4 IPC Opcodes, [481](#)
- l4_task_release_cap
 - Task, [377](#)
- l4_task_unmap
 - Task, [378](#)
- l4_task_unmap_batch
 - Task, [380](#)
- L4_TASK_UNMAP_OP
 - L4 IPC Opcodes, [481](#)
- l4_task_vgicc_map
 - Task, [381](#)
- L4_THREAD_AMD64_GET_SEGMENT_INFO_OP
 - L4 IPC Opcodes, [482](#)
- L4_THREAD_AMD64_SET_SEGMENT_BASE_OP
 - L4 IPC Opcodes, [482](#)
- l4_thread_arm_set_tpidruro
 - Thread, [386](#)
- L4_THREAD_ARM_TPIDRURO_OP
 - L4 IPC Opcodes, [482](#)
- L4_THREAD_CONTROL_ALIEN
 - Thread, [385](#)
- l4_thread_control_alien
 - Thread control, [406](#)
- l4_thread_control_bind
 - Thread control, [407](#)
- L4_THREAD_CONTROL_BIND_TASK
 - Thread, [385](#)
- l4_thread_control_commit
 - Thread control, [408](#)
- l4_thread_control_exc_handler
 - Thread control, [409](#)
- L4_thread_control_flags
 - Thread, [384](#)
- L4_THREAD_CONTROL_MR_IDX_BIND_TASK
 - Thread, [385](#)
- L4_THREAD_CONTROL_MR_IDX_BIND_UTCB
 - Thread, [385](#)
- L4_THREAD_CONTROL_MR_IDX_EXC_HANDLER
 - Thread, [385](#)
- L4_THREAD_CONTROL_MR_IDX_FLAG_VALS
 - Thread, [385](#)
- L4_THREAD_CONTROL_MR_IDX_FLAGS
 - Thread, [385](#)
- L4_THREAD_CONTROL_MR_IDX_PAGER
 - Thread, [385](#)
- L4_thread_control_mr_indices
 - Thread, [385](#)
- L4_THREAD_CONTROL_OP
 - L4 IPC Opcodes, [481](#)
- l4_thread_control_pager
 - Thread control, [409](#)
- L4_THREAD_CONTROL_SET_EXC_HANDLER
 - Thread, [385](#)
- L4_THREAD_CONTROL_SET_PAGER
 - Thread, [384](#)
- l4_thread_control_start
 - Thread control, [410](#)
- l4_thread_ex_regs
 - Thread, [387](#)
- L4_THREAD_EX_REGS_ARCH_MASK
 - Thread, [385](#)
- L4_THREAD_EX_REGS_ARM64_SET_EL_EL0
 - Thread, [386](#)
- L4_THREAD_EX_REGS_ARM64_SET_EL_EL1
 - Thread, [386](#)
- L4_THREAD_EX_REGS_ARM64_SET_EL_KEEP
 - Thread, [386](#)
- L4_THREAD_EX_REGS_ARM64_SET_EL_MASK
 - Thread, [386](#)
- L4_THREAD_EX_REGS_ARM_SET_EL_EL0
 - Thread, [386](#)
- L4_THREAD_EX_REGS_ARM_SET_EL_EL1
 - Thread, [386](#)

- L4_THREAD_EX_REGS_ARM_SET_EL_KEEP
 - Thread, [386](#)
- L4_THREAD_EX_REGS_ARM_SET_EL_MASK
 - Thread, [386](#)
- L4_THREAD_EX_REGS_CANCEL
 - Thread, [385](#)
- L4_thread_ex_regs_flags
 - Thread, [385](#)
- L4_thread_ex_regs_flags_arm
 - Thread, [385](#)
- L4_thread_ex_regs_flags_arm64
 - Thread, [386](#)
- L4_THREAD_EX_REGS_OP
 - L4 IPC Opcodes, [482](#)
- l4_thread_ex_regs_ret
 - Thread, [388](#)
- l4_thread_ex_regs_ret_u
 - Thread, [389](#)
- L4_THREAD_EX_REGS_TRIGGER_EXCEPTION
 - Thread, [385](#)
- l4_thread_ex_regs_u
 - Thread, [390](#)
- l4_thread_group_add
 - Thread groups, [417](#)
- l4_thread_group_remove
 - Thread groups, [417](#)
- l4_thread_modify_sender_add
 - Thread, [392](#)
- l4_thread_modify_sender_commit
 - Thread, [393](#)
- L4_THREAD_MODIFY_SENDER_OP
 - L4 IPC Opcodes, [482](#)
- l4_thread_modify_sender_start
 - Thread, [393](#)
- L4_THREAD_OPCODE_MASK
 - L4 IPC Opcodes, [482](#)
- L4_thread_ops
 - L4 IPC Opcodes, [481](#)
- l4_thread_register_del_irq
 - Thread, [394](#)
- L4_THREAD_REGISTER_DELETE_IRQ_OP
 - L4 IPC Opcodes, [482](#)
- l4_thread_register_doorbell_irq
 - Thread, [395](#)
- L4_THREAD_REGISTER_DOORBELL_IRQ_OP
 - L4 IPC Opcodes, [482](#)
- l4_thread_regs_t, [1594](#)
 - error, [1595](#)
 - free_marker, [1595](#)
- L4_THREAD_STATS_OP
 - L4 IPC Opcodes, [482](#)
- l4_thread_stats_time
 - Thread, [396](#)
- l4_thread_switch
 - Thread, [397](#)
- L4_THREAD_SWITCH_OP
 - L4 IPC Opcodes, [482](#)
- l4_thread_vcpu_control
 - Thread, [398](#)
- l4_thread_vcpu_control_ext
 - Thread, [399](#)
- l4_thread_vcpu_control_ext_u
 - Thread, [400](#)
- L4_THREAD_VCPU_CONTROL_OP
 - L4 IPC Opcodes, [482](#)
- l4_thread_vcpu_control_u
 - Thread, [401](#)
- l4_thread_vcpu_resume_commit
 - Thread, [402](#)
- L4_THREAD_VCPU_RESUME_OP
 - L4 IPC Opcodes, [482](#)
- l4_thread_vcpu_resume_start
 - Thread, [404](#)
- L4_THREAD_X86_GDT_OP
 - L4 IPC Opcodes, [482](#)
- l4_thread_yield
 - Thread, [404](#)
- l4_timeout
 - Timeouts, [240](#)
- l4_timeout_abs
 - Timeouts, [240](#)
- l4_timeout_get
 - Timeouts, [241](#)
- l4_timeout_is_absolute
 - Timeouts, [242](#)
- l4_timeout_rel
 - Timeouts, [243](#)
- l4_timeout_rel_get
 - Timeouts, [243](#)
- l4_timeout_s, [1595](#)
 - Timeouts, [238](#)
- l4_timeout_t, [1596](#)
 - Timeouts, [238](#)
- L4_TIMEOUT_US_MAX
 - Timeouts, [238](#)
- l4_touch_ro
 - Utility Functions, [649](#)
- l4_touch_rw
 - Utility Functions, [650](#)
- l4_trunc_page
 - Memory related, [206](#)
- l4_trunc_size
 - Memory related, [207](#)
- l4_tsc_init
 - Timestamp Counter, [667](#)
- l4_tsc_to_ns
 - Timestamp Counter, [668](#)
- l4_tsc_to_s_and_ns
 - Timestamp Counter, [668](#)
- l4_tsc_to_us
 - Timestamp Counter, [669](#)
- l4_unmap_flags_t
 - Task, [370](#)
- l4_usleep
 - Utility Functions, [650](#)
- l4_utcb_br

- Virtual Registers (UTCBS), [267](#)
- `l4_utcb_exc`
 - Exception registers, [270](#)
- `l4_utcb_exc_is_ex_regs_exception`
 - Exception registers, [271](#)
- `l4_utcb_exc_is_pf`
 - Exception registers, [271](#)
- `l4_utcb_exc_pc`
 - Exception registers, [272](#)
- `l4_utcb_exc_pc_set`
 - Exception registers, [272](#)
- `l4_utcb_mr`
 - Virtual Registers (UTCBS), [267](#)
- `l4_utcb_mr64_idx`
 - Timeouts, [244](#)
- `l4_utcb_t`
 - Virtual Registers (UTCBS), [266](#)
- `l4_utcb_tcr`
 - Virtual Registers (UTCBS), [268](#)
- `l4_vcon_attr_t`, [1597](#)
 - `set_raw`, [1598](#)
 - Virtual Console, [420](#)
- `L4_VCON_ECHO`
 - Virtual Console, [421](#)
- `l4_vcon_get_attr`
 - Virtual Console, [422](#)
- `L4_VCON_GET_ATTR_OP`
 - L4 IPC Opcodes, [482](#)
- `l4_vcon_get_attr_u`
 - Virtual Console, [422](#)
- `L4_vcon_i_flags`
 - Virtual Console, [421](#)
- `L4_VCON_ICANON`
 - Virtual Console, [421](#)
- `L4_VCON_ICRNL`
 - Virtual Console, [421](#)
- `L4_VCON_IGNCR`
 - Virtual Console, [421](#)
- `L4_VCON_INLCR`
 - Virtual Console, [421](#)
- `L4_vcon_l_flags`
 - Virtual Console, [421](#)
- `L4_vcon_o_flags`
 - Virtual Console, [421](#)
- `L4_VCON_OCRNL`
 - Virtual Console, [421](#)
- `L4_VCON_ONLCR`
 - Virtual Console, [421](#)
- `L4_VCON_ONLRET`
 - Virtual Console, [421](#)
- `L4_vcon_ops`
 - L4 IPC Opcodes, [482](#)
- `l4_vcon_read`
 - Virtual Console, [423](#)
- `L4_vcon_read_flags`
 - `vcon.h`, [3405](#)
- `L4_VCON_READ_OP`
 - L4 IPC Opcodes, [482](#)
- `L4_VCON_READ_SIZE`
 - Virtual Console, [422](#)
- `L4_VCON_READ_SIZE_MASK`
 - `vcon.h`, [3405](#)
- `L4_VCON_READ_STAT_BREAK`
 - `vcon.h`, [3405](#)
- `L4_VCON_READ_STAT_DONE`
 - `vcon.h`, [3405](#)
- `l4_vcon_read_u`
 - Virtual Console, [424](#)
- `l4_vcon_read_with_flags`
 - Virtual Console, [425](#)
- `l4_vcon_send`
 - Virtual Console, [426](#)
- `l4_vcon_send_u`
 - Virtual Console, [427](#)
- `l4_vcon_set_attr`
 - Virtual Console, [429](#)
- `L4_VCON_SET_ATTR_OP`
 - L4 IPC Opcodes, [482](#)
- `l4_vcon_set_attr_raw`
 - Virtual Console, [429](#)
- `l4_vcon_set_attr_u`
 - Virtual Console, [430](#)
- `L4_vcon_size_consts`
 - Virtual Console, [421](#)
- `l4_vcon_write`
 - Virtual Console, [431](#)
- `L4_VCON_WRITE_OP`
 - L4 IPC Opcodes, [482](#)
- `L4_VCON_WRITE_SIZE`
 - Virtual Console, [422](#)
- `l4_vcon_write_u`
 - Virtual Console, [432](#)
- `l4_vcpu_arch_state_t`, [1599](#)
- `l4_vcpu_check_version`
 - `vcpu.h`, [3517](#)
- `L4_vcpu_e_field_ids`
 - `__vcpu-arch.h`, [2571](#), [2574](#)
- `L4_VCPU_E_VTMR_CFG`
 - `__vcpu-arch.h`, [2571](#), [2574](#)
- `L4_VCPU_F_EXCEPTIONS`
 - vCPU API, [414](#)
- `L4_VCPU_F_FPU_ENABLED`
 - vCPU API, [415](#)
- `L4_VCPU_F_IRQ`
 - vCPU API, [414](#)
- `L4_VCPU_F_PAGE_FAULTS`
 - vCPU API, [414](#)
- `L4_VCPU_F_USER_MODE`
 - vCPU API, [414](#)
- `l4_vcpu_ipc_regs_t`, [1599](#)
- `L4_VCPU_OFFSET_EXT_INFOS`
 - vCPU API, [415](#), [416](#)
- `L4_VCPU_OFFSET_EXT_STATE`
 - vCPU API, [415](#), [416](#)
- `l4_vcpu_regs_t`, [1601](#)
 - `ax`, [1603](#)

- bp, [1603](#)
- bx, [1603](#)
- cx, [1603](#)
- di, [1603](#)
- dx, [1603](#)
- si, [1604](#)
- L4_VCPU_SF_IRQ_PENDING
 - vCPU API, [416](#)
- L4_vcpu_state_flags
 - vCPU API, [414](#)
- L4_vcpu_state_offset
 - vCPU API, [415](#), [416](#)
- l4_vcpu_state_t, [1604](#)
 - version, [1607](#)
- L4_VCPU_STATE_VERSION
 - __vcpu-arch.h, [2568](#), [2571](#), [2574](#), [2577](#), [2579](#)
- L4_vcpu_sticky_flags
 - vCPU API, [416](#)
- L4_virtio_cmd
 - L4 VIRTIO Transport Layer, [486](#)
- L4_virtio_irq_status
 - L4 VIRTIO Transport Layer, [486](#)
- L4_virtio_opcodes
 - L4 VIRTIO Transport Layer, [486](#)
- l4_vm_state_t, [1608](#)
- l4_vm_svm_vmcb_control_area, [1608](#)
- l4_vm_svm_vmcb_state_save_area, [1609](#)
- l4_vm_svm_vmcb_state_save_area_seg, [1610](#)
- l4_vm_svm_vmcb_t, [1611](#)
- l4_vm_tz_state, [1612](#)
- L4_VM_VMX_BASIC_REG
 - VM API for VMX, [306](#)
- L4_vm_vmx_caps_regs
 - VM API for VMX, [306](#)
- l4_vm_vmx_clear
 - VM API for VMX, [308](#)
- L4_VM_VMX_CR0_FIXED0_REG
 - VM API for VMX, [306](#)
- L4_VM_VMX_CR0_FIXED1_REG
 - VM API for VMX, [306](#)
- L4_VM_VMX_CR4_FIXED0_REG
 - VM API for VMX, [306](#)
- L4_VM_VMX_CR4_FIXED1_REG
 - VM API for VMX, [306](#)
- L4_vm_vmx_dfl1_regs
 - VM API for VMX, [306](#)
- L4_VM_VMX_ENTRY_CTLS_DFL1_REG
 - VM API for VMX, [306](#)
- L4_VM_VMX_EPT_VPID_CAP_REG
 - VM API for VMX, [306](#)
- L4_VM_VMX_EXIT_CTLS_DFL1_REG
 - VM API for VMX, [306](#)
- l4_vm_vmx_field_len
 - VM API for VMX, [308](#)
- l4_vm_vmx_field_order
 - VM API for VMX, [309](#)
- l4_vm_vmx_get_caps
 - VM API for VMX, [310](#)
- l4_vm_vmx_get_caps_default1
 - VM API for VMX, [310](#)
- l4_vm_vmx_get_cr2_index
 - VM API for VMX, [311](#)
- l4_vm_vmx_get_hw_vmcs
 - VM API for VMX, [311](#)
- L4_VM_VMX_MISC_REG
 - VM API for VMX, [306](#)
- L4_VM_VMX_NESTED_REVISION
 - VM API for VMX, [306](#)
- L4_VM_VMX_NUM_CAPS_REGS
 - VM API for VMX, [306](#)
- L4_VM_VMX_NUM_DFL1_REGS
 - VM API for VMX, [306](#)
- L4_VM_VMX_PINBASED_CTLS_DFL1_REG
 - VM API for VMX, [306](#)
- L4_VM_VMX_PROCBASED_CTLS2_REG
 - VM API for VMX, [306](#)
- L4_VM_VMX_PROCBASED_CTLS_DFL1_REG
 - VM API for VMX, [306](#)
- l4_vm_vmx_ptr_load
 - VM API for VMX, [312](#)
- l4_vm_vmx_read
 - VM API for VMX, [313](#)
- l4_vm_vmx_read_16
 - VM API for VMX, [314](#)
- l4_vm_vmx_read_32
 - VM API for VMX, [314](#)
- l4_vm_vmx_read_64
 - VM API for VMX, [315](#)
- l4_vm_vmx_read_nat
 - VM API for VMX, [315](#)
- l4_vm_vmx_set_hw_vmcs
 - VM API for VMX, [316](#)
- L4_vm_vmx_sw_fields
 - VM API for VMX, [306](#)
- L4_VM_VMX_TRUE_ENTRY_CTLS_REG
 - VM API for VMX, [306](#)
- L4_VM_VMX_TRUE_EXIT_CTLS_REG
 - VM API for VMX, [306](#)
- L4_VM_VMX_TRUE_PINBASED_CTLS_REG
 - VM API for VMX, [306](#)
- L4_VM_VMX_TRUE_PROCBASED_CTLS_REG
 - VM API for VMX, [306](#)
- l4_vm_vmx_vcpu_infos_t, [1612](#)
 - dfl1, [1613](#)
- l4_vm_vmx_vcpu_state_t, [1613](#)
 - VM API for VMX, [303](#)
- l4_vm_vmx_vcpu_vmcs_t, [1615](#)
 - VM API for VMX, [304](#)
- L4_VM_VMX_VMCS_CR2
 - VM API for VMX, [307](#)
- L4_VM_VMX_VMCS_ENUM_REG
 - VM API for VMX, [306](#)
- L4_VM_VMX_VMCS_MSR_CSTAR
 - VM API for VMX, [307](#)
- L4_VM_VMX_VMCS_MSR_KERNEL_GS_BASE
 - VM API for VMX, [307](#)

- L4_VM_VMX_VMCS_MSR_LSTAR
 - VM API for VMX, [307](#)
- L4_VM_VMX_VMCS_MSR_STAR
 - VM API for VMX, [307](#)
- L4_VM_VMX_VMCS_MSR_SYSCALL_MASK
 - VM API for VMX, [307](#)
- L4_VM_VMX_VMCS_MSR_TSC_AUX
 - VM API for VMX, [307](#)
- L4_VM_VMX_VMCS_NAT_ARG0
 - VM API for VMX, [307](#)
- L4_VM_VMX_VMCS_NAT_ARG1
 - VM API for VMX, [307](#)
- L4_VM_VMX_VMCS_NAT_ARG2
 - VM API for VMX, [307](#)
- L4_VM_VMX_VMCS_NAT_ARG3
 - VM API for VMX, [307](#)
- L4_VM_VMX_VMCS_SIZE_DIRTY_BITMAP
 - VM API for VMX, [307](#)
- L4_VM_VMX_VMCS_SIZE_VALUES
 - VM API for VMX, [307](#)
- L4_vm_vmx_vmxcs_sizes
 - VM API for VMX, [307](#)
- L4_VM_VMX_VMCS_XCR0
 - VM API for VMX, [307](#)
- l4_vm_vmx_write
 - VM API for VMX, [317](#)
- l4_vm_vmx_write_16
 - VM API for VMX, [318](#)
- l4_vm_vmx_write_32
 - VM API for VMX, [319](#)
- l4_vm_vmx_write_64
 - VM API for VMX, [319](#)
- l4_vm_vmx_write_nat
 - VM API for VMX, [320](#)
- l4_vmx_offset_table_t, [1616](#)
 - VM API for VMX, [304](#)
- L4_WHOLE_ADDRESS_SPACE
 - Flexpages, [177](#)
- L4_WHOLE_IOADDRESS_SPACE
 - Flexpages, [178](#)
- L4drivers::Mmio_register_block< MAX_BITS >, [1618](#)
- L4drivers::Register_block< MAX_BITS, BLOCK >, [1620](#)
 - operator[], [1621](#), [1622](#)
 - r, [1622](#), [1623](#)
- L4drivers::Register_block_base< MAX_BITS >, [1624](#)
- L4drivers::Register_block_impl< BASE, MAX_BITS >, [1625](#)
- L4drivers::Register_block_tmpl< BLOCK >, [1627](#)
- L4drivers::Register_tmpl< BITS, BLOCK >, [1628](#)
 - clear, [1631](#)
 - modify, [1631](#)
 - operator=, [1632](#)
 - set, [1632](#)
 - write, [1633](#)
- L4drivers::Ro_register_block< MAX_BITS, BLOCK >, [1634](#)
 - operator[], [1635](#)
 - r, [1635](#)
- L4drivers::Ro_register_tmpl< BITS, BLOCK >, [1636](#)
 - operator value_type, [1638](#)
 - read, [1638](#)
- L4IO_DEVICE_ANY
 - IO interface, [460](#)
- L4IO_DEVICE_INVALID
 - IO interface, [460](#)
- L4IO_DEVICE_OTHER
 - IO interface, [460](#)
- L4IO_DEVICE_PCI
 - IO interface, [460](#)
- l4io_device_types_t
 - IO interface, [459](#)
- L4IO_DEVICE_USB
 - IO interface, [460](#)
- l4io_get_root_device
 - io.h, [2644](#)
- l4io_has_resource
 - IO interface, [461](#)
- l4io_iomem_flags_t
 - IO interface, [460](#)
- l4io_iterate_devices
 - io.h, [2644](#)
- l4io_lookup_device
 - IO interface, [461](#)
- l4io_lookup_resource
 - IO interface, [461](#)
- L4IO_MEM_CACHED
 - IO interface, [460](#)
- L4IO_MEM_EAGER_MAP
 - IO interface, [460](#)
- L4IO_MEM_NONCACHED
 - IO interface, [460](#)
- L4IO_MEM_USE_MTRR
 - IO interface, [460](#)
- L4IO_MEM_USE_RESERVED_AREA
 - IO interface, [460](#)
- l4io_release_iomem
 - IO interface, [462](#)
- l4io_release_ioport
 - IO interface, [462](#)
- l4io_request_all_ioports
 - io.h, [2644](#)
- l4io_request_icu
 - io.h, [2645](#)
- l4io_request_iomem
 - IO interface, [463](#)
- l4io_request_iomem_region
 - IO interface, [464](#)
- l4io_request_ioport
 - IO interface, [464](#)
- l4io_request_resource_iomem
 - IO interface, [465](#)
- L4IO_RESOURCE_ANY
 - IO interface, [460](#)
- L4IO_RESOURCE_INVALID
 - IO interface, [460](#)

- L4IO_RESOURCE_IRQ
 - IO interface, [460](#)
- L4IO_RESOURCE_MEM
 - IO interface, [460](#)
- L4IO_RESOURCE_PORT
 - IO interface, [460](#)
- l4io_resource_t
 - IO interface, [459](#)
- l4io_resource_types_t
 - IO interface, [460](#)
- l4irq_attach
 - Interface using direct functionality., [469](#)
- l4irq_attach_cap
 - Interface using direct functionality., [473](#)
- l4irq_attach_cap_ft
 - Interface using direct functionality., [473](#)
- l4irq_attach_ft
 - Interface using direct functionality., [469](#)
- l4irq_attach_thread
 - Interface using direct functionality., [469](#)
- l4irq_attach_thread_cap
 - Interface using direct functionality., [474](#)
- l4irq_attach_thread_cap_ft
 - Interface using direct functionality., [474](#)
- l4irq_attach_thread_ft
 - Interface using direct functionality., [470](#)
- l4irq_detach
 - Interface using direct functionality., [470](#)
- l4irq_release
 - Interface for asynchronous ISR handlers., [476](#)
- l4irq_request
 - Interface for asynchronous ISR handlers., [476](#)
- l4irq_request_cap
 - Interface for asynchronous ISR handlers with a given IRQ capability., [478](#)
- l4irq_unmask
 - Interface using direct functionality., [471](#)
- l4irq_unmask_and_wait_any
 - Interface using direct functionality., [471](#)
- l4irq_wait
 - Interface using direct functionality., [472](#)
- l4irq_wait_any
 - Interface using direct functionality., [472](#)
- l4la_alloc
 - list_alloc.h, [3468](#)
- l4la_avail
 - list_alloc.h, [3468](#)
- l4la_dump
 - list_alloc.h, [3469](#)
- l4la_free
 - list_alloc.h, [3469](#)
- l4la_init
 - list_alloc.h, [3469](#)
- l4mod.h
 - L4util_l4mod_mod_flag_kernel, [3466](#)
 - L4util_l4mod_mod_flag_mask, [3466](#)
 - L4util_l4mod_mod_flag_roottask, [3466](#)
 - L4util_l4mod_mod_flag_sigma0, [3466](#)
 - L4util_l4mod_mod_flag_unspec, [3466](#)
 - l4util_l4mod_mod_info_flag, [3466](#)
- L4Re, [774](#)
 - chkcapp, [779](#)
 - chkipc, [780](#)
 - chksys, [781–783](#)
 - make_shared_cap, [785](#)
 - make_shared_del_cap, [786](#)
 - make_unique_cap, [787](#)
 - make_unique_del_cap, [787](#)
 - Shared_cap, [777](#)
 - shared_cap_cast, [788, 789](#)
 - shared_cap_dynamic_cast, [790, 791](#)
 - shared_cap_reinterpret_cast, [791, 792](#)
 - Shared_del_cap, [777](#)
 - shared_del_cap_cast, [793, 794](#)
 - shared_del_cap_dynamic_cast, [795](#)
 - shared_del_cap_reinterpret_cast, [796, 797](#)
 - throw_error, [798](#)
 - Unique_cap, [778](#)
 - Unique_del_cap, [778](#)
- L4Re Build System, [33](#)
- L4Re C Interface, [524](#)
- L4Re C++ Interface, [585](#)
- L4Re Capability API, [588](#)
 - cap_alloc, [590](#)
 - make_ref_cap, [589](#)
 - make_ref_del_cap, [589](#)
 - reply_cap_alloc, [590](#)
- L4Re ELF Auxiliary Information, [593](#)
 - L4RE_ELF_AUX_ELEM, [594](#)
 - L4RE_ELF_AUX_ELEM_T, [594](#)
 - L4RE_ELF_AUX_T_EX_REGS_FLAGS, [595](#)
 - L4RE_ELF_AUX_T_KIP_ADDR, [595](#)
 - L4RE_ELF_AUX_T_NONE, [595](#)
 - L4RE_ELF_AUX_T_STACK_ADDR, [595](#)
 - L4RE_ELF_AUX_T_STACK_SIZE, [595](#)
 - L4RE_ELF_AUX_T_VMA, [595](#)
- L4Re Protocol identifiers, [599](#)
 - L4RE_PROTO_DATASPACE, [600](#)
 - L4RE_PROTO_DEBUG, [600](#)
 - L4RE_PROTO_DMA_SPACE, [600](#)
 - L4RE_PROTO_EVENT, [600](#)
 - L4RE_PROTO_GOOS, [600](#)
 - L4RE_PROTO_INHIBITOR, [600](#)
 - L4RE_PROTO_ITAS, [600](#)
 - L4RE_PROTO_MEM_ALLOC, [600](#)
 - L4RE_PROTO_MMIO_SPACE, [600](#)
 - L4RE_PROTO_NAMESPACE, [600](#)
 - L4RE_PROTO_PARENT, [600](#)
 - L4RE_PROTO_REMOTE_ACCESS, [600](#)
 - L4RE_PROTO_RM, [600](#)
 - L4RE_PROTO_RSVD_1, [600](#)
 - L4re_protocols, [600](#)
- L4Re Servers, [49](#)
- L4Re Util C Interface, [526](#)
- L4Re Util C++ Interface, [587](#)
- L4Re::Cap_alloc, [1639](#)

- alloc, 1640, 1641
- free, 1641
- L4Re::Console, 1642
- L4Re::Core::Ref_ptr< T, CNT >, 1645
 - get, 1649
 - ptr, 1649
 - Ref_ptr, 1648
 - release, 1649
- L4Re::Dataspace, 1650
 - allocate, 1653
 - clear, 1654
 - copy_in, 1655
 - flags, 1656
 - info, 1657
 - map, 1658
 - map_info, 1660
 - map_region, 1661
 - size, 1662
- L4Re::Dataspace::F, 1663
 - Bufferable, 1664
 - Cacheable, 1664
 - Caching_mask, 1664
 - Caching_shift, 1663
 - Flags, 1663
 - Normal, 1664
 - R, 1664
 - Rights_mask, 1664
 - Ro, 1664
 - RW, 1664
 - RWX, 1664
 - RX, 1664
 - Uncacheable, 1664
 - W, 1664
 - X, 1664
- L4Re::Dataspace::Stats, 1664
- L4Re::Debug_obj, 1665
 - debug, 1668
- L4Re::Default_event_payload, 1669
- L4Re::Dma_space, 1670
 - associate, 1674
 - Attribute, 1672
 - Attributes, 1672
 - Bidirectional, 1673
 - Coherent, 1673
 - Direction, 1673
 - disassociate, 1674
 - From_device, 1673
 - map, 1675
 - No_sync, 1673
 - None, 1673
 - Phys_space, 1673
 - Space_attrib, 1673
 - To_device, 1673
 - unmap, 1676
- L4Re::Env, 1677
 - dbg_events, 1681
 - env, 1681
 - factory, 1682, 1683
 - first_free_cap, 1683
 - first_free_reply_cap, 1684
 - first_free_utcb, 1684
 - get, 1685
 - get_cap, 1685, 1686
 - initial_caps, 1687
 - itas, 1688
 - log, 1688
 - main_thread, 1689
 - mem_alloc, 1689
 - parent, 1690
 - rm, 1690, 1691
 - scheduler, 1691, 1692
 - task, 1692
 - utcb_area, 1692, 1693
- L4Re::Event, 1693
 - get_axis_info, 1697
 - get_buffer, 1698
 - get_num_streams, 1699
 - get_stream_info, 1700
 - get_stream_info_for_id, 1701
 - get_stream_state_for_id, 1702
- L4Re::Event_buffer_t< PAYLOAD >, 1703
 - Event_buffer_t, 1706
 - next, 1706
 - put, 1706
- L4Re::Event_buffer_t< PAYLOAD >::Event, 1707
- L4Re::Inhibitor, 1708
 - acquire, 1711
 - Name_max, 1711
 - next_lock_info, 1712
 - release, 1713
- L4Re::Itas, 1714
 - getitimer, 1717
 - Ignore_sigaction, 1717
 - raise, 1718
 - register_thread, 1719
 - setitimer, 1719
 - sigaction, 1720
 - sigaltstack, 1720
 - sigpending, 1721
 - sigprocmask, 1722
 - unregister_thread, 1723
- L4Re::Log, 1724
 - print, 1729
 - printn, 1729
- L4Re::Mem_alloc, 1730
 - alloc, 1734
 - Continuous, 1734
 - Fixed_paddr, 1734
 - info, 1735
 - Mem_alloc_flags, 1734
 - Pinned, 1734
 - Super_pages, 1734
- L4Re::Mem_alloc::Stats, 1736
 - mem_free, 1737
 - mem_limit, 1737
 - mem_used, 1737

- quota, 1738
- quota_used, 1738
- L4Re::Mmio_space, 1739
 - Access_width, 1742
 - mmio_read, 1742
 - mmio_write, 1743
 - Wd_16bit, 1742
 - Wd_32bit, 1742
 - Wd_64bit, 1742
 - Wd_8bit, 1742
- L4Re::Namespace, 1744
 - Link, 1749
 - Overwrite, 1749
 - Partly_resolved, 1748
 - query, 1749, 1750
 - Query_result_flags, 1748
 - Query_timeout, 1748
 - Register_flags, 1748
 - register_obj, 1751
 - Ro, 1749
 - Rs, 1749
 - Rw, 1749
 - Rws, 1749
 - Strong, 1749
 - To_default, 1748
 - To_non_blocking, 1748
 - Trusted, 1749
 - unlink, 1752
- L4Re::Ned::Cmd_control, 1754
 - execute, 1755
- L4Re::Parent, 1756
 - signal, 1759
- L4Re::Random, 1761
 - get_random, 1764
- L4Re::Rm, 1765
 - attach, 1771, 1773
 - Caching_shift, 1771
 - detach, 1775–1777
 - Detach_again, 1771
 - Detach_exact, 1770
 - Detach_flags, 1770
 - Detach_keep, 1770
 - Detach_overlap, 1770
 - Detach_result, 1770
 - Detached_ds, 1771
 - find, 1778
 - free_area, 1779
 - get_areas, 1780
 - get_info, 1781
 - get_regions, 1782
 - Kept_ds, 1771
 - Region_flag_shifts, 1771
 - reserve_area, 1783, 1784
 - Split_ds, 1771
- L4Re::Rm::Area, 1786
- L4Re::Rm::F, 1786
 - Attach_flags, 1787
 - Attach_mask, 1787
 - Cache_buffered, 1788
 - Cache_normal, 1788
 - Cache_uncached, 1788
 - Caching_mask, 1788
 - Detach_free, 1788
 - Ds_map_mask, 1788
 - Eager_map, 1787
 - In_area, 1787
 - Kernel, 1788
 - No_eager_map, 1787
 - Pager, 1788
 - R, 1788
 - Region_flags, 1787
 - Region_flags_mask, 1788
 - Reserved, 1788
 - Rights_mask, 1788
 - RW, 1788
 - RWX, 1788
 - RX, 1788
 - Search_addr, 1787
 - W, 1788
 - X, 1788
- L4Re::Rm::Region, 1788
- L4Re::Rm::Unique_region< T >, 1789
 - ~Unique_region, 1792
 - get, 1793
 - is_valid, 1793
 - operator=, 1793
 - release, 1794
 - reset, 1794
 - Unique_region, 1791, 1792
- L4Re::Smart_cap_auto< Unmap_flags >, 1795
- L4Re::Smart_count_cap< Unmap_flags >, 1795
- L4Re::Util, 799
 - make_shared_cap, 805
 - make_shared_del_cap, 805
 - make_unique_cap, 805
 - make_unique_del_cap, 806
 - Shared_cap, 802
 - shared_cap_cast, 806, 807
 - shared_cap_dynamic_cast, 808, 809
 - shared_cap_reinterpret_cast, 809, 810
 - Shared_del_cap, 803
 - shared_del_cap_cast, 811, 812
 - shared_del_cap_dynamic_cast, 813
 - shared_del_cap_reinterpret_cast, 814, 815
 - Unique_cap, 803
 - Unique_del_cap, 804
- L4Re::Util::_Cap_alloc, 1797
 - alloc, 1799
 - free, 1799
- L4Re::Util::Bitmap< BITS >, 1800
 - scan_zero, 1804
- L4Re::Util::Bitmap_base, 1805
 - atomic_clear_bit, 1808
 - atomic_get_and_clear, 1808
 - atomic_get_and_set, 1808
 - atomic_set_bit, 1809

- bit, [1809](#)
- bit_index, [1810](#)
- C_bits, [1808](#)
- clear_bit, [1810](#)
- operator[], [1810](#), [1811](#)
- scan_zero, [1811](#)
- set_bit, [1812](#)
- W_bits, [1808](#)
- word_index, [1812](#)
- L4Re::Util::Bitmap_base::Bit, [1813](#)
- L4Re::Util::Bitmap_base::Char< BITS >, [1813](#)
- L4Re::Util::Bitmap_base::Word< BITS >, [1814](#)
- L4Re::Util::Br_manager, [1815](#)
 - alloc_buffer_demand, [1819](#)
 - get_rcv_cap, [1819](#)
 - get_rcv_mem, [1820](#)
 - realloc_rcv_cap, [1821](#)
 - set_rcv_cap_flags, [1822](#)
- L4Re::Util::Br_manager_hooks, [1823](#)
- L4Re::Util::Br_manager_timeout_hooks, [1826](#)
- L4Re::Util::Cap_alloc_base, [1829](#)
- L4Re::Util::Counter< COUNTER >, [1830](#)
 - dec, [1830](#)
 - inc, [1830](#)
- L4Re::Util::Counter_atomic< COUNTER >, [1831](#)
 - dec, [1832](#)
 - inc, [1832](#)
- L4Re::Util::Counting_cap_alloc< COUNTERTYPE, Dbg >, [1833](#)
 - alloc, [1835](#)
 - Counting_cap_alloc, [1834](#)
 - free, [1836](#)
 - release, [1837](#)
 - setup, [1838](#)
 - take, [1838](#)
- L4Re::Util::Dataspace_svr, [1839](#)
 - allocate, [1840](#)
 - clear, [1841](#)
 - copy, [1841](#)
 - is_static, [1842](#)
 - map, [1842](#)
 - map_hook, [1843](#)
 - map_info, [1844](#)
 - page_shift, [1844](#)
 - release, [1845](#)
 - take, [1845](#)
- L4Re::Util::Event_buffer_consumer_t< PAYLOAD >, [1846](#)
 - foreach_available_event, [1848](#)
 - process, [1848](#)
- L4Re::Util::Event_buffer_t< PAYLOAD >, [1849](#)
 - attach, [1852](#)
 - buf, [1852](#)
 - detach, [1852](#)
- L4Re::Util::Event_svr< SVR >, [1853](#)
- L4Re::Util::Event_t< PAYLOAD >, [1855](#)
 - buffer, [1857](#)
 - init, [1857](#)
 - init_poll, [1858](#)
 - irq, [1858](#)
 - Mode, [1857](#)
 - Mode_irq, [1857](#)
 - Mode_polling, [1857](#)
- L4Re::Util::Item_alloc_base, [1859](#)
- L4Re::Util::Names::Name, [1860](#)
- L4Re::Util::Names::Name_space, [1863](#)
 - alloc_dynamic_entry, [1864](#)
 - copy_receive_cap, [1865](#)
 - free_capability, [1865](#)
 - free_dynamic_entry, [1866](#)
 - free_epiface, [1866](#)
 - get_epiface, [1867](#)
- L4Re::Util::Object_registry, [1868](#)
 - Object_registry, [1871](#)
 - register_irq_obj, [1872](#)
 - register_obj, [1872](#), [1873](#)
 - unregister_obj, [1874](#)
- L4Re::Util::Ref_cap< T >, [1875](#)
- L4Re::Util::Ref_del_cap< T >, [1877](#)
- L4Re::Util::Registry_server< LOOP_HOOKS >, [1878](#)
 - loop, [1882](#)
 - loop_dbg, [1883](#)
 - Registry_server, [1881](#), [1882](#)
- L4Re::Util::Smart_cap_auto< Unmap_flags >, [1884](#)
- L4Re::Util::Smart_count_cap< Unmap_flags >, [1885](#)
- L4Re::Util::Vcon_svr< SVR >, [1886](#)
- L4Re::Util::Video::Goos_svr, [1887](#)
 - get_fb, [1889](#)
 - init_infos, [1889](#)
 - refresh, [1889](#)
 - screen_info, [1890](#)
 - view_info, [1890](#)
- L4Re::Vfs, [816](#)
- L4Re::Vfs::Be_file, [1891](#)
 - check_ready, [1894](#)
 - data_space, [1894](#)
 - fstat, [1894](#)
 - unlock_all_locks, [1895](#)
- L4Re::Vfs::Be_file_system, [1896](#)
 - ~Be_file_system, [1898](#)
 - Be_file_system, [1898](#)
 - type, [1898](#)
- L4Re::Vfs::Directory, [1899](#)
 - faccessat, [1901](#)
 - link, [1901](#)
 - mkdir, [1901](#)
 - rename, [1902](#)
 - rmdir, [1902](#)
 - symlink, [1903](#)
 - unlink, [1903](#)
- L4Re::Vfs::File, [1904](#)
- L4Re::Vfs::File_system, [1908](#)
 - mount, [1909](#)
 - type, [1910](#)
- L4Re::Vfs::Fs, [1910](#)
 - alloc_fd, [1913](#)

- free_fd, 1913
- get_file, 1913
- mount, 1914
- set_fd, 1915
- L4Re::Vfs::Generic_file, 1915
 - check_ready, 1918
 - fchmod, 1919
 - fstat, 1919
 - get_status_flags, 1920
 - Ready_type, 1918
 - set_status_flags, 1921
 - unlock_all_locks, 1922
- L4Re::Vfs::Mman, 1924
- L4Re::Vfs::Ops, 1925
- L4Re::Vfs::Regular_file, 1928
 - data_space, 1931
 - fdatsync, 1931
 - fsync, 1931
 - ftruncate, 1932
 - get_lock, 1933
 - lseek, 1934
 - readv, 1935
 - set_lock, 1936
 - writew, 1937
- L4Re::Vfs::Special_file, 1938
 - ioctl, 1940
- L4Re::Video::Color_component, 1941
 - Color_component, 1942
 - dump, 1942
 - get, 1943
 - operator==, 1943
 - set, 1944
 - shift, 1944
 - size, 1945
- L4Re::Video::Goos, 1946
 - create_buffer, 1950
 - create_view, 1950
 - delete_buffer, 1951
 - delete_view, 1952
 - F_auto_refresh, 1949
 - F_dynamic_buffers, 1949
 - F_dynamic_views, 1949
 - F_pointer, 1949
 - Flags, 1949
 - get_static_buffer, 1953
 - info, 1954
 - view, 1955
- L4Re::Video::Goos::Info, 1956
- L4Re::Video::Pixel_info, 1958
 - a, 1962
 - b, 1962, 1963
 - bits_per_pixel, 1963
 - bytes_per_pixel, 1964
 - dump, 1965
 - g, 1966
 - has_alpha, 1966
 - operator==, 1966
 - padding, 1967
 - Pixel_info, 1960, 1961
 - r, 1967, 1968
- L4Re::Video::View, 1969
 - F_above, 1971
 - F_dyn_allocated, 1971
 - F_flags_mask, 1971
 - F_fully_dynamic, 1971
 - F_none, 1970
 - F_set_background, 1971
 - F_set_buffer, 1970
 - F_set_buffer_offset, 1970
 - F_set_bytes_per_line, 1970
 - F_set_flags, 1971
 - F_set_pixel, 1971
 - F_set_position, 1971
 - Flags, 1970
 - info, 1971
 - refresh, 1972
 - set_info, 1973
 - set_viewport, 1974
 - stack, 1975
 - V_flags, 1971
- L4Re::Video::View::Info, 1976
- l4re_aux_t, 1978
- l4re_debug_obj_debug
 - Debug interface, 532
- l4re_dma_space_associate
 - DMA Space Interface, 534
- L4RE_DMA_SPACE_BIDIRECTIONAL
 - dma_space.h, 2905
- L4RE_DMA_SPACE_COHERENT
 - dma_space.h, 2905
- l4re_dma_space_direction
 - dma_space.h, 2905
- l4re_dma_space_disassociate
 - DMA Space Interface, 535
- L4RE_DMA_SPACE_FROM_DEVICE
 - dma_space.h, 2905
- l4re_dma_space_map
 - DMA Space Interface, 535
- L4RE_DMA_SPACE_NONE
 - dma_space.h, 2905
- L4RE_DMA_SPACE_PHYS_SPACE
 - dma_space.h, 2905
- l4re_dma_space_space_attribs
 - dma_space.h, 2905
- l4re_dma_space_t
 - DMA Space Interface, 534
- L4RE_DMA_SPACE_TO_DEVICE
 - dma_space.h, 2905
- l4re_dma_space_unmap
 - DMA Space Interface, 536
- l4re_ds_allocate
 - Dataspace interface, 528
- l4re_ds_clear
 - Dataspace interface, 528
- l4re_ds_copy_in
 - Dataspace interface, 529

- L4RE_DS_F_BUFFERABLE
 - Dataspace interface, [527](#)
- L4RE_DS_F_CACHEABLE
 - Dataspace interface, [527](#)
- L4RE_DS_F_CACHING_MASK
 - Dataspace interface, [527](#)
- L4RE_DS_F_CACHING_SHIFT
 - Dataspace interface, [527](#)
- L4RE_DS_F_NORMAL
 - Dataspace interface, [527](#)
- L4RE_DS_F_UNCACHEABLE
 - Dataspace interface, [527](#)
- l4re_ds_flags
 - Dataspace interface, [529](#)
- l4re_ds_info
 - Dataspace interface, [530](#)
- l4re_ds_map_flags
 - Dataspace interface, [527](#)
- l4re_ds_map_info
 - Dataspace interface, [530](#)
- l4re_ds_size
 - Dataspace interface, [531](#)
- l4re_ds_stats_t, [1979](#)
- L4RE_ELF_AUX_ELEM
 - L4Re ELF Auxiliary Information, [594](#)
- L4RE_ELF_AUX_ELEM_T
 - L4Re ELF Auxiliary Information, [594](#)
- l4re_elf_aux_mword_t, [1979](#)
- l4re_elf_aux_t, [1980](#)
- L4RE_ELF_AUX_T_EX_REGS_FLAGS
 - L4Re ELF Auxiliary Information, [595](#)
- L4RE_ELF_AUX_T_KIP_ADDR
 - L4Re ELF Auxiliary Information, [595](#)
- L4RE_ELF_AUX_T_NONE
 - L4Re ELF Auxiliary Information, [595](#)
- L4RE_ELF_AUX_T_STACK_ADDR
 - L4Re ELF Auxiliary Information, [595](#)
- L4RE_ELF_AUX_T_STACK_SIZE
 - L4Re ELF Auxiliary Information, [595](#)
- L4RE_ELF_AUX_T_VMA
 - L4Re ELF Auxiliary Information, [595](#)
- l4re_elf_aux_vma_t, [1981](#)
- l4re_env
 - Initial Environment, [581](#)
- l4re_env_cap_entry_t, [1981](#)
 - flags, [1983](#)
 - l4re_env_cap_entry_t, [1982](#)
- l4re_env_get_cap
 - Initial Environment, [582](#)
- l4re_env_get_cap_e
 - Initial Environment, [582](#)
- l4re_env_get_cap_l
 - Initial Environment, [583](#)
- l4re_env_t, [1983](#)
 - caps, [1985](#)
 - env.h, [2951](#)
- l4re_event_get_axis_info
 - Event interface, [538](#)
- l4re_event_get_buffer
 - Event interface, [538](#)
- l4re_event_get_num_streams
 - Event interface, [538](#)
- l4re_event_get_stream_info
 - Event interface, [539](#)
- l4re_event_get_stream_info_for_id
 - Event interface, [539](#)
- l4re_event_t, [1986](#)
- l4re_inhibitor_acquire
 - inhibitor.h, [2912](#)
- l4re_inhibitor_next_lock_info
 - inhibitor.h, [2912](#)
- l4re_inhibitor_release
 - inhibitor.h, [2913](#)
- l4re_kip
 - Initial Environment, [584](#)
- l4re_log_print
 - Log interface, [541](#)
- l4re_log_print_srv
 - Log interface, [541](#)
- l4re_log_printn
 - Log interface, [542](#)
- l4re_log_printn_srv
 - Log interface, [542](#)
- l4re_ma_alloc
 - Memory allocator, [544](#)
- l4re_ma_alloc_align
 - Memory allocator, [545](#)
- l4re_ma_alloc_align_srv
 - Memory allocator, [547](#)
- l4re_ma_flags
 - Memory allocator, [544](#)
- l4re_ns_query_srv
 - Namespace interface, [549](#)
- l4re_ns_query_to_srv
 - Namespace interface, [550](#)
- l4re_ns_register_flags
 - Namespace interface, [549](#)
- l4re_ns_register_obj_srv
 - Namespace interface, [551](#)
- l4re_parent_signal
 - parent.h, [2920](#)
- L4RE_PROTO_DATASPACE
 - L4Re Protocol identifiers, [600](#)
- L4RE_PROTO_DEBUG
 - L4Re Protocol identifiers, [600](#)
- L4RE_PROTO_DMA_SPACE
 - L4Re Protocol identifiers, [600](#)
- L4RE_PROTO_EVENT
 - L4Re Protocol identifiers, [600](#)
- L4RE_PROTO_GOOS
 - L4Re Protocol identifiers, [600](#)
- L4RE_PROTO_INHIBITOR
 - L4Re Protocol identifiers, [600](#)
- L4RE_PROTO_ITAS
 - L4Re Protocol identifiers, [600](#)
- L4RE_PROTO_MEM_ALLOC

- L4Re Protocol identifiers, [600](#)
- L4RE_PROTO_MMIO_SPACE
 - L4Re Protocol identifiers, [600](#)
- L4RE_PROTO_NAMESPACE
 - L4Re Protocol identifiers, [600](#)
- L4RE_PROTO_PARENT
 - L4Re Protocol identifiers, [600](#)
- L4RE_PROTO_REMOTE_ACCESS
 - L4Re Protocol identifiers, [600](#)
- L4RE_PROTO_RM
 - L4Re Protocol identifiers, [600](#)
- L4RE_PROTO_RSVD_1
 - L4Re Protocol identifiers, [600](#)
- L4re_protocols
 - L4Re Protocol identifiers, [600](#)
- l4re_rm_attach
 - Region map interface, [554](#)
- l4re_rm_attach_srv
 - Region map interface, [556](#)
- l4re_rm_attach_w_info
 - Region map interface, [556](#)
- l4re_rm_attach_w_info_srv
 - Region map interface, [558](#)
- L4RE_RM_CACHING_SHIFT
 - Region map interface, [554](#)
- l4re_rm_detach
 - Region map interface, [558](#)
- l4re_rm_detach_ds
 - Region map interface, [559](#)
- l4re_rm_detach_ds_unmap
 - Region map interface, [560](#)
- l4re_rm_detach_srv
 - Region map interface, [561](#)
- l4re_rm_detach_unmap
 - Region map interface, [562](#)
- L4RE_RM_F_ATTACH_FLAGS
 - Region map interface, [554](#)
- L4RE_RM_F_CACHE_BUFFERED
 - Region map interface, [554](#)
- L4RE_RM_F_CACHE_NORMAL
 - Region map interface, [554](#)
- L4RE_RM_F_CACHE_UNCACHED
 - Region map interface, [554](#)
- L4RE_RM_F_CACHING
 - Region map interface, [554](#)
- L4RE_RM_F_DETACH_FREE
 - Region map interface, [554](#)
- L4RE_RM_F_EAGER_MAP
 - Region map interface, [554](#)
- L4RE_RM_F_IN_AREA
 - Region map interface, [554](#)
- L4RE_RM_F_KERNEL
 - Region map interface, [554](#)
- L4RE_RM_F_NO_EAGER_MAP
 - Region map interface, [554](#)
- L4RE_RM_F_PAGER
 - Region map interface, [554](#)
- L4RE_RM_F_R
 - Region map interface, [554](#)
- Region map interface, [554](#)
- L4RE_RM_F_RESERVED
 - Region map interface, [554](#)
- L4RE_RM_F_SEARCH_ADDR
 - Region map interface, [554](#)
- l4re_rm_find
 - Region map interface, [563](#)
- l4re_rm_find_srv
 - Region map interface, [564](#)
- l4re_rm_flags_values
 - Region map interface, [554](#)
- l4re_rm_free_area
 - Region map interface, [565](#)
- l4re_rm_free_area_srv
 - Region map interface, [566](#)
- l4re_rm_get_info
 - Region map interface, [566](#)
- l4re_rm_get_info_srv
 - Region map interface, [567](#)
- L4RE_RM_REGION_FLAGS
 - Region map interface, [554](#)
- l4re_rm_reserve_area
 - Region map interface, [568](#)
- l4re_rm_reserve_area_srv
 - Region map interface, [569](#)
- l4re_rm_show_lists
 - Region map interface, [570](#)
- l4re_util_cap_last
 - Capability allocator, [571](#)
- l4re_util_kumem_alloc
 - kumem_alloc.h, [2928](#)
- l4re_video_color_component_t, [1987](#)
- l4re_video_goos_create_buffer
 - Video API, [574](#)
- l4re_video_goos_create_view
 - Video API, [575](#)
- l4re_video_goos_delete_buffer
 - Video API, [575](#)
- l4re_video_goos_delete_view
 - Video API, [576](#)
- l4re_video_goos_get_static_buffer
 - Video API, [576](#)
- l4re_video_goos_get_view
 - Video API, [576](#)
- l4re_video_goos_info
 - Video API, [577](#)
- l4re_video_goos_info_flags_t
 - Video API, [574](#)
- l4re_video_goos_info_t, [1987](#)
- l4re_video_goos_refresh
 - Video API, [577](#)
- l4re_video_pixel_info_t, [1989](#)
- l4re_video_view_get_info
 - Video API, [578](#)
- l4re_video_view_info_flags_t
 - Video API, [574](#)
- l4re_video_view_info_t, [1990](#)
- l4re_video_view_refresh

- Video API, [578](#)
- `l4re_video_view_set_info`
 - Video API, [579](#)
- `l4re_video_view_set_viewport`
 - Video API, [579](#)
- `l4re_video_view_stack`
 - Video API, [580](#)
- `l4re_video_view_t`, [1992](#)
 - Video API, [573](#)
- L4SHM-based ring buffer implementation, [604](#)
- `l4shmc_add_chunk`
 - Chunks, [614](#)
- `l4shmc_add_signal`
 - Signals, [626](#)
- `l4shmc_area_overhead`
 - Shared Memory Library, [609](#)
- `l4shmc_area_size`
 - Shared Memory Library, [609](#)
- `l4shmc_area_size_free`
 - Shared Memory Library, [609](#)
- `l4shmc_attach`
 - Shared Memory Library, [609](#)
- `l4shmc_attach_signal`
 - Signals, [627](#)
- `l4shmc_check_magic`
 - Signals, [627](#)
- `l4shmc_chunk_capacity`
 - Chunks, [614](#)
- `l4shmc_chunk_consumed`
 - Consumer, [622](#)
- `l4shmc_chunk_overhead`
 - Shared Memory Library, [610](#)
- `l4shmc_chunk_ptr`
 - Chunks, [615](#)
- `l4shmc_chunk_ready`
 - Producer, [618](#)
- `l4shmc_chunk_ready_sig`
 - Producer, [619](#)
- `l4shmc_chunk_signal`
 - Chunks, [615](#)
- `l4shmc_chunk_size`
 - Consumer, [622](#)
- `l4shmc_chunk_try_to_take`
 - Producer, [619](#)
- `l4shmc_chunk_try_to_take_for_overwriting`
 - Producer, [620](#)
- `l4shmc_chunk_try_to_take_for_reading`
 - Consumer, [622](#)
- `l4shmc_chunk_try_to_take_for_writing`
 - Producer, [620](#)
- `l4shmc_connect_chunk_signal`
 - Shared Memory Library, [610](#)
- `l4shmc_create`
 - Shared Memory Library, [611](#)
- `l4shmc_enable_chunk`
 - Consumer, [623](#)
- `l4shmc_enable_signal`
 - Consumer, [631](#)
- `l4shmc_get_chunk`
 - Chunks, [616](#)
- `l4shmc_get_chunk_to`
 - Chunks, [616](#)
- `l4shmc_get_client_nr`
 - Shared Memory Library, [611](#)
- `l4shmc_get_initialized_clients`
 - Shared Memory Library, [612](#)
- `l4shmc_get_signal`
 - Signals, [628](#)
- `l4shmc_is_chunk_clear`
 - Producer, [621](#)
- `l4shmc_is_chunk_ready`
 - Consumer, [623](#)
- `l4shmc_iterate_chunk`
 - Chunks, [617](#)
- `l4shmc_mark_client_initialized`
 - Shared Memory Library, [612](#)
- `l4shmc_rb_attach_receiver`
 - `ringbuf.h`, [3081](#)
- `l4shmc_rb_attach_sender`
 - `ringbuf.h`, [3082](#)
- `l4shmc_rb_deinit_buffer`
 - `ringbuf.h`, [3082](#)
- `l4shmc_rb_init_buffer`
 - `ringbuf.h`, [3082](#)
- `l4shmc_rb_init_receiver`
 - `ringbuf.h`, [3083](#)
- `l4shmc_rb_receiver_copy_out`
 - `ringbuf.h`, [3084](#)
- `l4shmc_rb_receiver_notify_done`
 - `ringbuf.h`, [3084](#)
- `l4shmc_rb_receiver_read_next_size`
 - `ringbuf.h`, [3085](#)
- `l4shmc_rb_receiver_wait_for_data`
 - `ringbuf.h`, [3085](#)
- `l4shmc_rb_sender_alloc_packet`
 - `ringbuf.h`, [3085](#)
- `l4shmc_rb_sender_commit_packet`
 - `ringbuf.h`, [3086](#)
- `l4shmc_rb_sender_next_copy_in`
 - `ringbuf.h`, [3086](#)
- `l4shmc_rb_sender_put_data`
 - `ringbuf.h`, [3087](#)
- L4SHMC_RINGBUF_DATA
 - Internal, [606](#)
- L4SHMC_RINGBUF_DATA_SIZE
 - Internal, [606](#)
- L4SHMC_RINGBUF_HEAD
 - Internal, [607](#)
- `l4shmc_ringbuf_head_t`, [1993](#)
- `l4shmc_ringbuf_t`, [1994](#)
- `l4shmc_signal_cap`
 - Signals, [628](#)
- `l4shmc_trigger`
 - Producer, [629](#)
- `l4shmc_wait_any`
 - Consumer, [631](#)

- l4shmc_wait_any_to
 - Consumer, [631](#)
- l4shmc_wait_any_try
 - Consumer, [632](#)
- l4shmc_wait_chunk
 - Consumer, [624](#)
- l4shmc_wait_chunk_to
 - Consumer, [624](#)
- l4shmc_wait_chunk_try
 - Consumer, [625](#)
- l4shmc_wait_signal
 - Consumer, [632](#)
- l4shmc_wait_signal_to
 - Consumer, [633](#)
- l4shmc_wait_signal_try
 - Consumer, [633](#)
- l4sigma0_debug_dump
 - Sigma0 API, [636](#)
- L4SIGMA0_IPCERROR
 - Sigma0 API, [635](#)
- l4sigma0_map_anypage
 - Sigma0 API, [636](#)
- l4sigma0_map_errstr
 - Sigma0 API, [636](#)
- l4sigma0_map_iomem
 - Sigma0 API, [637](#)
- l4sigma0_map_kip
 - Sigma0 API, [637](#)
- l4sigma0_map_mem
 - Sigma0 API, [638](#)
- L4SIGMA0_NOFPAGE
 - Sigma0 API, [635](#)
- L4SIGMA0_NOTALIGNED
 - Sigma0 API, [635](#)
- L4SIGMA0_OK
 - Sigma0 API, [635](#)
- l4sigma0_return_flags_t
 - Sigma0 API, [635](#)
- L4SIGMA0_SMALLERFPAGE
 - Sigma0 API, [635](#)
- l4util_add16
 - Atomic Instructions, [671](#)
- l4util_add16_res
 - Atomic Instructions, [671](#)
- l4util_add32
 - Atomic Instructions, [671](#)
- l4util_add32_res
 - Atomic Instructions, [672](#)
- l4util_add8
 - Atomic Instructions, [672](#)
- l4util_add8_res
 - Atomic Instructions, [672](#)
- l4util_and16
 - Atomic Instructions, [673](#)
- l4util_and16_res
 - Atomic Instructions, [673](#)
- l4util_and32
 - Atomic Instructions, [673](#)
- l4util_and32_res
 - Atomic Instructions, [674](#)
- l4util_and8
 - Atomic Instructions, [674](#)
- l4util_and8_res
 - Atomic Instructions, [674](#)
- l4util_atomic_add
 - Atomic Instructions, [675](#)
- l4util_atomic_inc
 - Atomic Instructions, [675](#)
- l4util_backtrace
 - backtrace.h, [3429](#)
- l4util_bsf
 - Bit Manipulation, [688](#)
- l4util_bsr
 - Bit Manipulation, [689](#)
- l4util_btc
 - Bit Manipulation, [689](#)
- l4util_btr
 - Bit Manipulation, [689](#)
- l4util_bts
 - Bit Manipulation, [690](#)
- l4util_clear_bit
 - Bit Manipulation, [691](#)
- l4util_cmpxchg
 - Atomic Instructions, [675](#)
- l4util_cmpxchg16
 - Atomic Instructions, [676](#)
- l4util_cmpxchg32
 - Atomic Instructions, [677](#)
- l4util_cmpxchg8
 - Atomic Instructions, [677](#)
- l4util_complement_bit
 - Bit Manipulation, [691](#)
- l4util_cpu_capabilities
 - CPU related functions, [654](#)
- l4util_cpu_capabilities_nocheck
 - CPU related functions, [654](#)
- l4util_cpu_has_cpuid
 - CPU related functions, [655](#)
- l4util_dec16
 - Atomic Instructions, [678](#)
- l4util_dec16_res
 - Atomic Instructions, [678](#)
- l4util_dec32
 - Atomic Instructions, [678](#)
- l4util_dec32_res
 - Atomic Instructions, [679](#)
- l4util_dec8
 - Atomic Instructions, [679](#)
- l4util_dec8_res
 - Atomic Instructions, [679](#)
- l4util_find_first_set_bit
 - Bit Manipulation, [692](#)
- l4util_find_first_zero_bit
 - Bit Manipulation, [692](#)
- l4util_in16
 - IA32 Port I/O API, [657](#)

- l4util_in32
 - IA32 Port I/O API, [657](#)
- l4util_in8
 - IA32 Port I/O API, [657](#)
- l4util_inc16
 - Atomic Instructions, [679](#)
- l4util_inc16_res
 - Atomic Instructions, [680](#)
- l4util_inc32
 - Atomic Instructions, [680](#)
- l4util_inc32_res
 - Atomic Instructions, [680](#)
- l4util_inc8
 - Atomic Instructions, [681](#)
- l4util_inc8_res
 - Atomic Instructions, [681](#)
- l4util_ins16
 - IA32 Port I/O API, [658](#)
- l4util_ins32
 - IA32 Port I/O API, [658](#)
- l4util_ins8
 - IA32 Port I/O API, [659](#)
- l4util_ioport_map
 - IA32 Port I/O API, [659](#)
- l4util_kip_for_each_feature
 - Kernel Interface Page API, [719](#)
- l4util_kip_kernel_abi_version
 - Kernel Interface Page API, [720](#)
- l4util_kip_kernel_has_feature
 - Kernel Interface Page API, [720](#)
- l4util_l4mod_info, [1995](#)
 - vbe_ctrl_info, [1996](#)
- l4util_l4mod_mod, [1997](#)
- L4util_l4mod_mod_flag_kernel
 - l4mod.h, [3466](#)
- L4util_l4mod_mod_flag_mask
 - l4mod.h, [3466](#)
- L4util_l4mod_mod_flag_roottask
 - l4mod.h, [3466](#)
- L4util_l4mod_mod_flag_sigma0
 - l4mod.h, [3466](#)
- L4util_l4mod_mod_flag_unspec
 - l4mod.h, [3466](#)
- l4util_l4mod_mod_info_flag
 - l4mod.h, [3466](#)
- l4util_mb_addr_range_t, [1998](#)
- l4util_mb_apm_t, [1999](#)
- l4util_mb_drive_t, [1999](#)
- l4util_mb_for_each_mmap_entry
 - mb_info.h, [3475](#)
- l4util_mb_info_t, [2000](#)
- L4UTIL_MB_MEMORY
 - mb_info.h, [3475](#)
- l4util_mb_mod_t, [2002](#)
- l4util_mb_vbe_ctrl_t, [2003](#)
- l4util_mb_vbe_mode_t, [2003](#)
- l4util_micros2l4to
 - Utility Functions, [651](#)
- l4util_next_power2
 - Bit Manipulation, [692](#)
- l4util_or16
 - Atomic Instructions, [681](#)
- l4util_or16_res
 - Atomic Instructions, [682](#)
- l4util_or32
 - Atomic Instructions, [682](#)
- l4util_or32_res
 - Atomic Instructions, [682](#)
- l4util_or8
 - Atomic Instructions, [682](#)
- l4util_or8_res
 - Atomic Instructions, [683](#)
- l4util_out16
 - IA32 Port I/O API, [660](#)
- l4util_out32
 - IA32 Port I/O API, [660](#)
- l4util_out8
 - IA32 Port I/O API, [661](#)
- l4util_outs16
 - IA32 Port I/O API, [661](#)
- l4util_outs32
 - IA32 Port I/O API, [661](#)
- l4util_outs8
 - IA32 Port I/O API, [662](#)
- l4util_rand
 - Random number support, [723](#)
- l4util_set_bit
 - Bit Manipulation, [693](#)
- l4util_splitlog2_hdl
 - Utility Functions, [651](#)
- l4util_splitlog2_size
 - Utility Functions, [652](#)
- l4util_srand
 - Random number support, [723](#)
- l4util_sub16
 - Atomic Instructions, [683](#)
- l4util_sub16_res
 - Atomic Instructions, [683](#)
- l4util_sub32
 - Atomic Instructions, [684](#)
- l4util_sub32_res
 - Atomic Instructions, [684](#)
- l4util_sub8
 - Atomic Instructions, [684](#)
- l4util_sub8_res
 - Atomic Instructions, [685](#)
- l4util_test_bit
 - Bit Manipulation, [693](#)
- l4util_xchg
 - Atomic Instructions, [685](#)
- l4util_xchg16
 - Atomic Instructions, [685](#)
- l4util_xchg32
 - Atomic Instructions, [686](#)
- l4util_xchg8
 - Atomic Instructions, [686](#)

- L4vbus, [817](#)
- L4vbus GPIO functions, [505](#)
 - l4vbus_gpio_config_get, [507](#)
 - l4vbus_gpio_config_pad, [508](#)
 - l4vbus_gpio_config_pull, [508](#)
 - L4vbus_gpio_generic_func, [506](#)
 - l4vbus_gpio_get, [509](#)
 - l4vbus_gpio_multi_config_pad, [510](#)
 - l4vbus_gpio_multi_get, [511](#)
 - l4vbus_gpio_multi_set, [511](#)
 - l4vbus_gpio_multi_setup, [512](#)
 - L4VBUS_GPIO_PIN_PULL_DOWN, [507](#)
 - L4VBUS_GPIO_PIN_PULL_NONE, [507](#)
 - L4VBUS_GPIO_PIN_PULL_UP, [507](#)
 - L4vbus_gpio_pull_modes, [507](#)
 - l4vbus_gpio_set, [513](#)
 - l4vbus_gpio_setup, [514](#)
 - L4VBUS_GPIO_SETUP_INPUT, [507](#)
 - L4VBUS_GPIO_SETUP_IRQ, [507](#)
 - L4VBUS_GPIO_SETUP_OUTPUT, [507](#)
 - l4vbus_gpio_to_irq, [515](#)
- L4vbus PCI functions, [516](#)
 - l4vbus_pci_cfg_read, [516](#)
 - l4vbus_pci_cfg_write, [517](#)
 - l4vbus_pci_irq_enable, [518](#)
 - l4vbus_pciddev_cfg_read, [519](#)
 - l4vbus_pciddev_cfg_write, [520](#)
 - l4vbus_pciddev_irq_enable, [521](#)
- L4vbus power management functions, [522](#)
 - l4vbus_pm_resume, [522](#)
 - l4vbus_pm_suspend, [523](#)
- L4vbus::Device, [2007](#)
 - bus_cap, [2010](#)
 - dev_handle, [2011](#)
 - Device, [2010](#)
 - device, [2011](#)
 - device_by_hid, [2012](#)
 - get_resource, [2013](#)
 - is_compatible, [2014](#)
 - next_device, [2015](#)
 - operator!=, [2015](#)
 - operator==, [2016](#)
- L4vbus::Gpio_module, [2017](#)
 - config_pad, [2020](#)
 - get, [2020](#)
 - pin, [2021](#)
 - set, [2022](#)
 - setup, [2023](#)
- L4vbus::Gpio_module::Pin_slice, [2024](#)
- L4vbus::Gpio_pin, [2024](#)
 - config_get, [2028](#)
 - config_pad, [2028](#)
 - config_pull, [2029](#)
 - get, [2030](#)
 - pin, [2030](#)
 - set, [2030](#)
 - setup, [2031](#)
 - to_irq, [2032](#)
- L4vbus::Icu, [2033](#)
 - Src_dev_handle, [2036](#)
 - Src_types, [2036](#)
 - vicu, [2036](#)
- L4vbus::Pci_dev, [2037](#)
 - cfg_read, [2041](#)
 - cfg_write, [2041](#)
 - irq_enable, [2042](#)
- L4vbus::Pci_host_bridge, [2043](#)
 - cfg_read, [2047](#)
 - cfg_write, [2047](#)
 - irq_enable, [2048](#)
- L4vbus::Pm< DEC >, [2049](#)
 - pm_resume, [2051](#)
 - pm_suspend, [2051](#)
- L4vbus::Vbus, [2052](#)
 - assign_dma_domain, [2059](#), [2060](#)
 - release_ioport, [2061](#)
 - request_ioport, [2061](#)
 - root, [2062](#)
- l4vbus_assign_dma_domain
 - L4 Vbus functions, [497](#)
- L4VBUS_DEVICE_F_CHILDREN
 - vbus_types.h, [3511](#)
- l4vbus_device_flags_t
 - vbus_types.h, [3511](#)
- l4vbus_device_t, [2063](#)
- L4vbus_dma_domain_assign_flags
 - L4 Vbus functions, [497](#)
- L4VBUS_DMAD_BIND
 - L4 Vbus functions, [497](#)
- L4VBUS_DMAD_KERNEL_DMA_SPACE
 - L4 Vbus functions, [497](#)
- L4VBUS_DMAD_L4RE_DMA_SPACE
 - L4 Vbus functions, [497](#)
- L4VBUS_DMAD_UNBIND
 - L4 Vbus functions, [497](#)
- l4vbus_get_adr
 - L4 Vbus functions, [498](#)
- l4vbus_get_device
 - L4 Vbus functions, [498](#)
- l4vbus_get_device_by_hid
 - L4 Vbus functions, [499](#)
- l4vbus_get_hid
 - L4 Vbus functions, [500](#)
- l4vbus_get_next_device
 - L4 Vbus functions, [501](#)
- l4vbus_get_resource
 - L4 Vbus functions, [501](#)
- l4vbus_gpio_config_get
 - L4vbus GPIO functions, [507](#)
- l4vbus_gpio_config_pad
 - L4vbus GPIO functions, [508](#)
- l4vbus_gpio_config_pull
 - L4vbus GPIO functions, [508](#)
- L4vbus_gpio_generic_func
 - L4vbus GPIO functions, [506](#)
- l4vbus_gpio_get

- L4vbus GPIO functions, [509](#)
- l4vbus_gpio_multi_config_pad
 - L4vbus GPIO functions, [510](#)
- l4vbus_gpio_multi_get
 - L4vbus GPIO functions, [511](#)
- l4vbus_gpio_multi_set
 - L4vbus GPIO functions, [511](#)
- l4vbus_gpio_multi_setup
 - L4vbus GPIO functions, [512](#)
- L4VBUS_GPIO_PIN_PULL_DOWN
 - L4vbus GPIO functions, [507](#)
- L4VBUS_GPIO_PIN_PULL_NONE
 - L4vbus GPIO functions, [507](#)
- L4VBUS_GPIO_PIN_PULL_UP
 - L4vbus GPIO functions, [507](#)
- L4vbus_gpio_pull_modes
 - L4vbus GPIO functions, [507](#)
- l4vbus_gpio_set
 - L4vbus GPIO functions, [513](#)
- l4vbus_gpio_setup
 - L4vbus GPIO functions, [514](#)
- L4VBUS_GPIO_SETUP_INPUT
 - L4vbus GPIO functions, [507](#)
- L4VBUS_GPIO_SETUP_IRQ
 - L4vbus GPIO functions, [507](#)
- L4VBUS_GPIO_SETUP_OUTPUT
 - L4vbus GPIO functions, [507](#)
- l4vbus_gpio_to_irq
 - L4vbus GPIO functions, [515](#)
- L4VBUS_ICU_SRC_DEV_HANDLE
 - vbus.h, [3498](#)
- l4vbus_icu_src_types
 - vbus.h, [3497](#)
- L4VBUS_IFACE_SHIFT
 - vbus_interfaces.h, [3504](#)
- l4vbus_iface_type_t
 - vbus_interfaces.h, [3504](#)
- L4VBUS_INTERFACE_BUS
 - vbus_interfaces.h, [3505](#)
- L4VBUS_INTERFACE_GENERIC
 - vbus_interfaces.h, [3505](#)
- L4VBUS_INTERFACE_GPIO
 - vbus_interfaces.h, [3505](#)
- L4VBUS_INTERFACE_ICU
 - vbus_interfaces.h, [3505](#)
- L4VBUS_INTERFACE_PCI
 - vbus_interfaces.h, [3505](#)
- L4VBUS_INTERFACE_PCIDEV
 - vbus_interfaces.h, [3505](#)
- L4VBUS_INTERFACE_PM
 - vbus_interfaces.h, [3505](#)
- l4vbus_is_compatible
 - L4 Vbus functions, [502](#)
- L4VBUS_NULL
 - vbus.h, [3497](#)
- l4vbus_pci_cfg_read
 - L4vbus PCI functions, [516](#)
- l4vbus_pci_cfg_write
 - L4vbus PCI functions, [517](#)
- l4vbus_pci_irq_enable
 - L4vbus PCI functions, [518](#)
- l4vbus_pcidev_cfg_read
 - L4vbus PCI functions, [519](#)
- l4vbus_pcidev_cfg_write
 - L4vbus PCI functions, [520](#)
- l4vbus_pcidev_irq_enable
 - L4vbus PCI functions, [521](#)
- l4vbus_pm_resume
 - L4vbus power management functions, [522](#)
- l4vbus_pm_suspend
 - L4vbus power management functions, [523](#)
- l4vbus_release_ioport
 - L4 Vbus functions, [503](#)
- l4vbus_request_ioport
 - L4 Vbus functions, [504](#)
- L4VBUS_RESOURCE_BUS
 - vbus_types.h, [3512](#)
- L4VBUS_RESOURCE_DMA_DOMAIN
 - vbus_types.h, [3512](#)
- L4VBUS_RESOURCE_F_MEM_CACHEABLE
 - vbus_types.h, [3511](#)
- L4VBUS_RESOURCE_F_MEM_MMIO_READ
 - vbus_types.h, [3511](#)
- L4VBUS_RESOURCE_F_MEM_MMIO_WRITE
 - vbus_types.h, [3511](#)
- L4VBUS_RESOURCE_F_MEM_PREFETCHABLE
 - vbus_types.h, [3511](#)
- L4VBUS_RESOURCE_F_MEM_R
 - vbus_types.h, [3511](#)
- L4VBUS_RESOURCE_F_MEM_W
 - vbus_types.h, [3511](#)
- l4vbus_resource_flags_t
 - vbus_types.h, [3511](#)
- L4VBUS_RESOURCE_GPIO
 - vbus_types.h, [3512](#)
- L4VBUS_RESOURCE_INVALID
 - vbus_types.h, [3512](#)
- L4VBUS_RESOURCE_IRQ
 - vbus_types.h, [3512](#)
- L4VBUS_RESOURCE_MAX
 - vbus_types.h, [3512](#)
- L4VBUS_RESOURCE_MEM
 - vbus_types.h, [3512](#)
- L4VBUS_RESOURCE_PORT
 - vbus_types.h, [3512](#)
- l4vbus_resource_t, [2064](#)
- l4vbus_resource_type_t
 - vbus_types.h, [3511](#)
- L4VBUS_ROOT_BUS
 - vbus.h, [3497](#)
- l4vbus_subinterface_supported
 - vbus_interfaces.h, [3505](#)
- l4vbus_vicu_get_cap
 - L4 Vbus functions, [504](#)
- L4vcpu::State, [2065](#)
 - add, [2066](#)

- clear, 2066
- set, 2067
- State, 2066
- L4vcpu::Vcpu, 2067
 - cast, 2071
 - entry_ip, 2072
 - entry_sp, 2072
 - ext_alloc, 2073
 - i, 2073, 2074
 - irq_disable_save, 2074
 - irq_enable, 2074
 - irq_restore, 2075
 - is_irq_entry, 2076
 - is_page_fault_entry, 2076
 - r, 2077
 - saved_state, 2077
 - state, 2078
 - task, 2078
 - wait_for_event, 2079
- l4vcpu_ext_alloc
 - Extended vCPU support, 733
- l4vcpu_irq_disable
 - vCPU Support Library, 726
- l4vcpu_irq_disable_save
 - vCPU Support Library, 727
- l4vcpu_irq_enable
 - vCPU Support Library, 727
- l4vcpu_irq_restore
 - vCPU Support Library, 728
- l4vcpu_is_irq_entry
 - vCPU Support Library, 729
- l4vcpu_is_page_fault_entry
 - vCPU Support Library, 730
- l4vcpu_print_state
 - vCPU Support Library, 731
- l4vcpu_wait_for_event
 - vCPU Support Library, 731
- L4virtio, 818
- L4virtio::Device, 2080
 - config_queue, 2085
 - device_config, 2085
 - device_notification_irq, 2086
 - register_ds, 2087
 - set_status, 2088
- L4virtio::Driver::Block_device, 2090
 - add_block, 2093
 - process_request, 2093
 - process_used_queue, 2094
 - send_request, 2095
 - setup_device, 2095
 - start_request, 2097
- L4virtio::Driver::Block_device::Handle, 2098
- L4virtio::Driver::Device, 2098
 - bind_notification_irq, 2101
 - config_queue, 2102
 - driver_acknowledge, 2102
 - driver_connect, 2103
 - feature_negotiated, 2105
 - max_queue_size, 2105
 - register_ds, 2106
 - send, 2107
 - send_and_wait, 2108
 - wait, 2109
 - wait_for_next_used, 2110
- L4virtio::Driver::Virtio_net_device, 2111
 - bind_rx_notification_irq, 2115
 - finish_rx, 2115
 - rx_pkt, 2115
 - rx_queue_size, 2116
 - setup_device, 2116
 - tx, 2117
 - tx_queue_size, 2118
 - wait_rx, 2119
- L4virtio::Driver::Virtio_net_device::Packet, 2120
- L4virtio::Driver::Virtqueue, 2121
 - alloc_descriptor, 2125
 - desc, 2125
 - enqueue_descriptor, 2125
 - find_next_used, 2126
 - free_descriptor, 2126
 - init_queue, 2127
 - initialize_rings, 2128
- L4virtio::Ptr< T >, 2130
 - get, 2132
 - Invalid, 2131
 - Invalid_type, 2131
 - is_valid, 2132
- L4virtio::Svr::Bad_descriptor, 2133
 - Bad_address, 2134
 - Bad_descriptor, 2134
 - Bad_flags, 2134
 - Bad_next, 2134
 - Bad_rights, 2134
 - Bad_size, 2134
 - Error, 2134
 - message, 2135
- L4virtio::Svr::Block_dev_base< Ds_data >, 2135
 - Block_dev_base, 2139
 - finalize_request, 2140
 - get_writeback, 2141
 - set_blk_size, 2141
 - set_config_wce, 2141
 - set_discard, 2142
 - set_size_max, 2142
 - set_topology, 2142
 - set_write_zeroes, 2143
- L4virtio::Svr::Block_request< Ds_data >, 2144
 - data_size, 2145
 - next_block, 2145
- L4virtio::Svr::Console::Control_message, 2146
 - Console_port, 2147
 - Device_add, 2147
 - Device_ready, 2147
 - Device_remove, 2147
 - Events, 2147
 - Port_name, 2147

- Port_open, [2147](#)
- Port_ready, [2147](#)
- Resize, [2147](#)
- L4virtio::Svr::Console::Control_request, [2148](#)
- L4virtio::Svr::Console::Device, [2149](#)
 - Device, [2154](#), [2155](#)
 - notify_queue, [2156](#)
 - port, [2157](#)
 - port_read, [2158](#)
 - port_write, [2159](#)
 - process_device_ready, [2161](#)
 - process_port_open, [2162](#)
 - process_port_ready, [2162](#)
- L4virtio::Svr::Console::Device_port, [2163](#)
- L4virtio::Svr::Console::Features, [2167](#)
 - console_multiport_bfm_t, [2171](#)
 - console_size_bfm_t, [2171](#)
 - emerg_write_bfm_t, [2171](#)
- L4virtio::Svr::Console::Port, [2172](#)
 - Port_added, [2175](#)
 - Port_disabled, [2175](#)
 - Port_failed, [2175](#)
 - Port_num_states, [2175](#)
 - Port_open, [2175](#)
 - Port_ready, [2175](#)
 - Port_status, [2175](#)
 - state_transitions, [2176](#)
- L4virtio::Svr::Console::Port::Transition, [2176](#)
- L4virtio::Svr::Console::Virtio_con, [2177](#)
 - handle_control_message, [2183](#)
 - notify_queue, [2183](#)
 - port, [2184](#)
 - port_add, [2185](#)
 - port_name, [2186](#)
 - port_open, [2187](#)
 - port_remove, [2188](#)
 - process_device_ready, [2189](#)
 - process_port_open, [2190](#)
 - process_port_ready, [2190](#)
 - reset_device, [2191](#)
 - send_control_message, [2192](#)
 - Virtio_con, [2182](#)
- L4virtio::Svr::Data_buffer, [2194](#)
 - copy_to, [2196](#)
 - Data_buffer, [2195](#)
 - done, [2196](#)
 - set, [2197](#)
 - skip, [2197](#)
- L4virtio::Svr::Dev_config, [2198](#)
 - add_irq_status, [2202](#)
 - change_queue_config, [2202](#)
 - Dev_config, [2200](#), [2201](#)
 - ds, [2203](#)
 - get_cmd, [2203](#)
 - guest_features, [2204](#)
 - hdr, [2204](#)
 - negotiated_features, [2205](#)
 - qconfig, [2206](#)
 - reset_cmd, [2206](#)
 - reset_queue, [2207](#)
 - set_device_needs_reset, [2208](#)
 - set_device_notify_index, [2208](#)
 - set_status, [2209](#)
 - status, [2209](#)
- L4virtio::Svr::Dev_features, [2210](#)
- L4virtio::Svr::Dev_status, [2213](#)
 - running, [2216](#)
- L4virtio::Svr::Device_t< DATA >, [2216](#)
 - add_trusted_dataspaces, [2220](#)
 - device_error, [2220](#)
 - device_notify_irq, [2220](#)
 - handle_mem_cmd_write, [2220](#)
 - init_mem_info, [2221](#)
 - register_driver_irq, [2221](#)
 - reset_queue_config, [2221](#)
 - setup_queue, [2222](#)
- L4virtio::Svr::Driver_mem_list_t< DATA >, [2223](#)
 - add, [2225](#)
 - find, [2225](#)
 - full, [2226](#)
 - init, [2227](#)
 - load_desc, [2227](#), [2228](#)
 - remove, [2229](#)
- L4virtio::Svr::Driver_mem_region_t< DATA >, [2229](#)
 - contains, [2232](#)
 - Driver_mem_region_t, [2232](#)
 - drv_base, [2233](#)
 - ds, [2233](#)
 - ds_offset, [2233](#)
 - empty, [2233](#)
 - flags, [2234](#)
 - is_writable, [2234](#)
 - local, [2234](#)
 - local_base, [2235](#)
 - size, [2235](#)
- L4virtio::Svr::Request_processor, [2236](#)
 - current_flags, [2237](#)
 - has_more, [2238](#)
 - next, [2238](#)
 - start, [2240](#), [2242](#)
- L4virtio::Svr::Scmi::Base_attr_t, [2243](#)
- L4virtio::Svr::Scmi::Base_proto, [2244](#)
- L4virtio::Svr::Scmi::Perf_proto, [2246](#)
- L4virtio::Svr::Scmi::Performance_attr_t, [2249](#)
- L4virtio::Svr::Scmi::Performance_describe_level_t, [2250](#)
- L4virtio::Svr::Scmi::Performance_describe_levels_n_t, [2250](#)
- L4virtio::Svr::Scmi::Performance_domain_attr_t, [2252](#)
- L4virtio::Svr::Scmi::Proto< OBSERV >, [2254](#)
- L4virtio::Svr::Scmi::Scmi_dev, [2255](#)
- L4virtio::Svr::Scmi::Scmi_hdr_t, [2259](#)
- L4virtio::Svr::Virtio_gpio< Request_handler, Epiface >, [2261](#)
- L4virtio::Svr::Virtio_gpio< Request_handler, Epiface >::Host_irq, [2265](#)

L4virtio::Svr::Virtio_gpio< Request_handler, Epiface
 >::Irq_handler, [2269](#)
 L4virtio::Svr::Virtio_gpio< Request_handler, Epiface
 >::Request_processor, [2270](#)
 get_request, [2272](#)
 L4virtio::Svr::Virtio_i2c< Request_handler, Epiface >,
 [2273](#)
 L4virtio::Svr::Virtio_i2c< Request_handler, Epiface
 >::Host_irq, [2277](#)
 L4virtio::Svr::Virtio_i2c< Request_handler, Epiface
 >::Request_processor, [2281](#)
 get_request, [2283](#)
 L4virtio::Svr::Virtio_rng< Rnd_state, Epiface >, [2284](#)
 L4virtio::Svr::Virtio_rng< Rnd_state, Epiface >::Host_irq,
 [2288](#)
 L4virtio::Svr::Virtio_rng< Rnd_state, Epiface >::Request_processor,
 [2292](#)
 L4virtio::Svr::Virtio_spi< Spi_request_handler, Epiface
 >, [2294](#)
 L4virtio::Svr::Virtio_spi< Spi_request_handler, Epiface
 >::Host_irq, [2299](#)
 L4virtio::Svr::Virtio_spi< Spi_request_handler, Epiface
 >::Request_processor, [2303](#)
 get_request, [2305](#)
 L4virtio::Svr::Virtqueue, [2306](#)
 consumed, [2310](#)
 desc, [2311](#)
 desc_avail, [2312](#)
 disable_notify, [2313](#)
 enable_notify, [2313](#)
 finish, [2314](#), [2315](#)
 next_avail, [2316](#)
 rewind_avail, [2317](#)
 L4virtio::Svr::Virtqueue::Head_desc, [2318](#)
 desc, [2319](#)
 operator bool, [2319](#)
 valid, [2320](#)
 L4virtio::Virtqueue, [2321](#)
 avail_align, [2324](#)
 avail_size, [2324](#)
 desc_align, [2326](#)
 desc_size, [2326](#)
 disable, [2327](#)
 dump, [2328](#)
 get_avail_idx, [2328](#)
 get_tail_avail_idx, [2329](#)
 no_notify_guest, [2329](#)
 no_notify_host, [2330](#)
 num, [2330](#)
 ready, [2331](#)
 setup, [2332](#)
 setup_simple, [2334](#)
 total_size, [2334](#), [2335](#)
 used_align, [2336](#)
 used_size, [2336](#)
 L4virtio::Virtqueue::Avail, [2338](#)
 L4virtio::Virtqueue::Avail::Flags, [2339](#)
 no_irq_bfm_t, [2340](#)
 L4virtio::Virtqueue::Desc, [2340](#)
 L4virtio::Virtqueue::Desc::Flags, [2342](#)
 indirect_bfm_t, [2343](#)
 next_bfm_t, [2343](#)
 write_bfm_t, [2344](#)
 L4virtio::Virtqueue::Used, [2344](#)
 L4virtio::Virtqueue::Used::Flags, [2345](#)
 no_notify_bfm_t, [2346](#)
 L4virtio::Virtqueue::Used_elem, [2347](#)
 Used_elem, [2348](#)
 l4virtio_block_config_t, [2348](#)
 l4virtio_block_discard_t, [2349](#)
 l4virtio_block_header_t, [2350](#)
 L4virtio_block_operations
 L4 VIRTIO Block Device, [493](#)
 L4VIRTIO_BLOCK_S_IOERR
 L4 VIRTIO Block Device, [493](#)
 L4VIRTIO_BLOCK_S_OK
 L4 VIRTIO Block Device, [493](#)
 L4VIRTIO_BLOCK_S_UNSUPP
 L4 VIRTIO Block Device, [493](#)
 L4virtio_block_status
 L4 VIRTIO Block Device, [493](#)
 L4VIRTIO_BLOCK_T_DISCARD
 L4 VIRTIO Block Device, [493](#)
 L4VIRTIO_BLOCK_T_FLUSH
 L4 VIRTIO Block Device, [493](#)
 L4VIRTIO_BLOCK_T_GET_ID
 L4 VIRTIO Block Device, [493](#)
 L4VIRTIO_BLOCK_T_IN
 L4 VIRTIO Block Device, [493](#)
 L4VIRTIO_BLOCK_T_OUT
 L4 VIRTIO Block Device, [493](#)
 L4VIRTIO_BLOCK_T_WRITE_ZEROES
 L4 VIRTIO Block Device, [493](#)
 L4VIRTIO_CMD_CFG_CHANGED
 L4 VIRTIO Transport Layer, [486](#)
 L4VIRTIO_CMD_CFG_QUEUE
 L4 VIRTIO Transport Layer, [486](#)
 L4VIRTIO_CMD_MASK
 L4 VIRTIO Transport Layer, [486](#)
 L4VIRTIO_CMD_NONE
 L4 VIRTIO Transport Layer, [486](#)
 L4VIRTIO_CMD_NOTIFY_QUEUE
 L4 VIRTIO Transport Layer, [486](#)
 L4VIRTIO_CMD_SET_STATUS
 L4 VIRTIO Transport Layer, [486](#)
 l4virtio_config_hdr_t, [2351](#)
 l4virtio_config_queue
 L4 VIRTIO Transport Layer, [488](#)
 l4virtio_config_queue_t, [2352](#)
 L4 VIRTIO Transport Layer, [485](#)
 l4virtio_config_queues
 L4 VIRTIO Transport Layer, [489](#)
 l4virtio_device_config
 L4 VIRTIO Transport Layer, [489](#)
 l4virtio_device_config_ds
 L4 VIRTIO Transport Layer, [489](#)

- L4virtio_device_ids
 - L4 VIRTIO Transport Layer, [487](#)
- l4virtio_device_notification_irq
 - L4 VIRTIO Transport Layer, [490](#)
- L4virtio_device_status
 - L4 VIRTIO Transport Layer, [487](#)
- L4virtio_feature_bits
 - L4 VIRTIO Transport Layer, [488](#)
- L4VIRTIO_FEATURE_CMD_CONFIG
 - L4 VIRTIO Transport Layer, [488](#)
- L4VIRTIO_FEATURE_VERSION_1
 - L4 VIRTIO Transport Layer, [488](#)
- L4VIRTIO_ID_9P
 - L4 VIRTIO Transport Layer, [487](#)
- L4VIRTIO_ID_BALLOON
 - L4 VIRTIO Transport Layer, [487](#)
- L4VIRTIO_ID_BLOCK
 - L4 VIRTIO Transport Layer, [487](#)
- L4VIRTIO_ID_CAIF
 - L4 VIRTIO Transport Layer, [487](#)
- L4VIRTIO_ID_CAN
 - L4 VIRTIO Transport Layer, [487](#)
- L4VIRTIO_ID_CONSOLE
 - L4 VIRTIO Transport Layer, [487](#)
- L4VIRTIO_ID_CRYPTIO
 - L4 VIRTIO Transport Layer, [487](#)
- L4VIRTIO_ID_FS
 - L4 VIRTIO Transport Layer, [487](#)
- L4VIRTIO_ID_GPIO
 - L4 VIRTIO Transport Layer, [487](#)
- L4VIRTIO_ID_GPU
 - L4 VIRTIO Transport Layer, [487](#)
- L4VIRTIO_ID_I2C
 - L4 VIRTIO Transport Layer, [487](#)
- L4VIRTIO_ID_INPUT
 - L4 VIRTIO Transport Layer, [487](#)
- L4VIRTIO_ID_NET
 - L4 VIRTIO Transport Layer, [487](#)
- L4VIRTIO_ID_RNG
 - L4 VIRTIO Transport Layer, [487](#)
- L4VIRTIO_ID_RPMSG
 - L4 VIRTIO Transport Layer, [487](#)
- L4VIRTIO_ID_RPROC_SERIAL
 - L4 VIRTIO Transport Layer, [487](#)
- L4VIRTIO_ID_SCSI
 - L4 VIRTIO Transport Layer, [487](#)
- L4VIRTIO_ID_SCSI
 - L4 VIRTIO Transport Layer, [487](#)
- L4VIRTIO_ID_SOCK
 - L4 VIRTIO Transport Layer, [487](#)
- L4VIRTIO_ID_SPI
 - L4 VIRTIO Transport Layer, [487](#)
- L4VIRTIO_ID_VSOCK
 - L4 VIRTIO Transport Layer, [487](#)
- L4VIRTIO_ID_WATCHDOG
 - L4 VIRTIO Transport Layer, [487](#)
- l4virtio_input_absinfo_t, [2353](#)
- l4virtio_input_config_t, [2354](#)
- l4virtio_input_devids_t, [2354](#)
- l4virtio_input_event_t, [2355](#)
- L4VIRTIO_IRQ_STATUS_CONFIG
 - L4 VIRTIO Transport Layer, [486](#)
- L4VIRTIO_IRQ_STATUS_VRING
 - L4 VIRTIO Transport Layer, [486](#)
- l4virtio_net_config_t, [2356](#)
- l4virtio_net_header_t, [2356](#)
- L4VIRTIO_OP_CONFIG_QUEUE
 - L4 VIRTIO Transport Layer, [487](#)
- L4VIRTIO_OP_DEVICE_CONFIG
 - L4 VIRTIO Transport Layer, [487](#)
- L4VIRTIO_OP_GET_DEVICE_IRQ
 - L4 VIRTIO Transport Layer, [487](#)
- L4VIRTIO_OP_REGISTER_DS
 - L4 VIRTIO Transport Layer, [487](#)
- L4VIRTIO_OP_SET_STATUS
 - L4 VIRTIO Transport Layer, [486](#)
- L4virtio_port, [2357](#)
 - drop_requests, [2362](#)
- l4virtio_register_ds
 - L4 VIRTIO Transport Layer, [490](#)
- l4virtio_set_status
 - L4 VIRTIO Transport Layer, [491](#)
- L4VIRTIO_STATUS_ACKNOWLEDGE
 - L4 VIRTIO Transport Layer, [488](#)
- L4VIRTIO_STATUS_DEVICE_NEEDS_RESET
 - L4 VIRTIO Transport Layer, [488](#)
- L4VIRTIO_STATUS_DRIVER
 - L4 VIRTIO Transport Layer, [488](#)
- L4VIRTIO_STATUS_DRIVER_OK
 - L4 VIRTIO Transport Layer, [488](#)
- L4VIRTIO_STATUS_FAILED
 - L4 VIRTIO Transport Layer, [488](#)
- L4VIRTIO_STATUS_FEATURES_OK
 - L4 VIRTIO Transport Layer, [488](#)
- Last
 - L4::lpc::Snd_fpage, [1224](#)
- lcm
 - cxx, [739](#)
- learn
 - Mac_table< Size >, [2365](#)
- length
 - L4::lpc::Varg, [1235](#)
- Libedid_block_size
 - EDID parsing functionality, [455](#)
- libedid_check_header
 - EDID parsing functionality, [455](#)
- libedid_checksum
 - EDID parsing functionality, [455](#)
- Libedid_consts
 - EDID parsing functionality, [455](#)
- libedid_dump
 - EDID parsing functionality, [455](#)
- libedid_dump_standard_timings
 - EDID parsing functionality, [456](#)
- libedid_num_ext_blocks
 - EDID parsing functionality, [456](#)

- libedid_pnp_id
 - EDID parsing functionality, [456](#)
- libedid_prefered_resolution
 - EDID parsing functionality, [457](#)
- libedid_revision
 - EDID parsing functionality, [457](#)
- libedid_version
 - EDID parsing functionality, [457](#)
- Link
 - L4Re::Namespace, [1749](#)
- link
 - L4Re::Vfs::Directory, [1901](#)
- List_alloc
 - cxx::List_alloc, [945](#)
- list_alloc.h
 - l4la_alloc, [3468](#)
 - l4la_avail, [3468](#)
 - l4la_dump, [3469](#)
 - l4la_free, [3469](#)
 - l4la_init, [3469](#)
- load_desc
 - L4virtio::Svr::Driver_mem_list_t< DATA >, [2227](#), [2228](#)
- local
 - L4virtio::Svr::Driver_mem_region_t< DATA >, [2234](#)
- local_base
 - L4virtio::Svr::Driver_mem_region_t< DATA >, [2235](#)
- local_id_received
 - L4::lpc::Snd_fpage, [1227](#)
- Lock_guard
 - L4::Lock_guard, [1345](#)
- log
 - L4Re::Env, [1688](#)
- Log interface, [540](#)
 - l4re_log_print, [541](#)
 - l4re_log_print_srv, [541](#)
 - l4re_log_printn, [542](#)
 - l4re_log_printn_srv, [542](#)
- Logging interface, [597](#)
- lookup
 - Mac_table< Size >, [2365](#)
- loop
 - L4::Server< LOOP_HOOKS >, [1416](#)
 - L4Re::Util::Registry_server< LOOP_HOOKS >, [1882](#)
- loop_dbg
 - L4::Server< LOOP_HOOKS >, [1417](#)
 - L4Re::Util::Registry_server< LOOP_HOOKS >, [1883](#)
- Low-Level Thread Functions, [724](#)
- lower_bound_node
 - cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, [895](#)
 - cxx::Bits::Bst< Node, Get_key, Compare >, [911](#)
- Lsb
 - cxx::Bitfield< T, LSB, MSB >, [855](#)
- Iseek
 - L4Re::Vfs::Regular_file, [1934](#)
- Lstr
 - L4::Factory::Lstr, [1103](#)
- Mac_addr, [2363](#)
- Mac_table< Size >, [2363](#)
 - flush, [2365](#)
 - learn, [2365](#)
 - lookup, [2365](#)
- Mag, the GUI Multiplexer, [82](#)
- main_thread
 - L4Re::Env, [1689](#)
- make_cap
 - L4::lpc, [759](#)
- make_cap_full
 - L4::lpc, [760](#)
- make_cap_grant
 - L4::lpc, [761](#)
- make_cap_rw
 - L4::lpc, [761](#)
- make_cap_rws
 - L4::lpc, [762](#)
- make_ref_cap
 - L4Re Capability API, [589](#)
- make_ref_del_cap
 - L4Re Capability API, [589](#)
- make_shared_cap
 - L4Re, [785](#)
 - L4Re::Util, [805](#)
- make_shared_del_cap
 - L4Re, [786](#)
 - L4Re::Util, [805](#)
- make_unique_cap
 - L4Re, [787](#)
 - L4Re::Util, [805](#)
- make_unique_del_cap
 - L4Re, [787](#)
 - L4Re::Util, [806](#)
- Map
 - L4::lpc::Cap< T >, [1149](#)
 - L4::lpc::Snd_fpage, [1224](#)
- map
 - L4::Task, [1444](#)
 - L4Re::Dataspace, [1658](#)
 - L4Re::Dma_space, [1675](#)
 - L4Re::Util::Dataspace_svr, [1842](#)
- map_hook
 - L4Re::Util::Dataspace_svr, [1843](#)
- map_info
 - L4Re::Dataspace, [1660](#)
 - L4Re::Util::Dataspace_svr, [1844](#)
- map_region
 - L4Re::Dataspace, [1661](#)
- Map_type
 - L4::lpc::Cap< T >, [1149](#)
 - L4::lpc::Snd_fpage, [1224](#)
- Map_type_mask
 - L4::lpc::Cap< T >, [1148](#)

- mask
 - L4::lcu, [1117](#)
- max
 - Small C++ Template Library, [641](#), [642](#)
- max_free_slabs
 - cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >, [845](#)
 - cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >, [850](#)
- max_queue_size
 - L4virtio::Driver::Device, [2105](#)
- MB_ART_MEMORY
 - mb_info.h, [3475](#)
- mb_info.h
 - l4util_mb_for_each_mmap_entry, [3475](#)
 - L4UTIL_MB_MEMORY, [3475](#)
 - MB_ART_MEMORY, [3475](#)
- Mem
 - L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS >, [1488](#)
- mem
 - L4::lpc::Rcv_fpage, [1213](#)
 - L4::lpc::Snd_fpage, [1227](#)
- mem_alloc
 - L4Re::Env, [1689](#)
- Mem_alloc_flags
 - L4Re::Mem_alloc, [1734](#)
- Mem_desc
 - L4::Kip::Mem_desc, [1315](#)
- mem_free
 - L4Re::Mem_alloc::Stats, [1737](#)
- mem_limit
 - L4Re::Mem_alloc::Stats, [1737](#)
- Mem_type
 - L4::Kip::Mem_desc, [1314](#)
- mem_used
 - L4Re::Mem_alloc::Stats, [1737](#)
- Memory
 - L4::lpc::Gen_fpage, [1153](#)
- Memory allocator, [543](#)
 - l4re_ma_alloc, [544](#)
 - l4re_ma_alloc_align, [545](#)
 - l4re_ma_alloc_align_srv, [547](#)
 - l4re_ma_flags, [544](#)
- Memory descriptors (C version), [440](#)
 - l4_kernel_info_get_mem_desc_end, [443](#)
 - l4_kernel_info_get_mem_desc_is_virtual, [443](#)
 - l4_kernel_info_get_mem_desc_start, [443](#)
 - l4_kernel_info_get_mem_desc_subtype, [443](#)
 - l4_kernel_info_get_mem_desc_type, [444](#)
 - l4_kernel_info_get_num_mem_descs, [444](#)
 - l4_kernel_info_mem_desc_t, [441](#)
 - l4_kernel_info_set_mem_desc, [444](#)
 - l4_mem_archspecific_acpi_nvs, [442](#)
 - l4_mem_archspecific_acpi_tables, [442](#)
 - l4_mem_archspecific_sub_type_common_t, [442](#)
 - l4_mem_info_acpi_rsdp, [442](#)
 - l4_mem_info_sub_type_t, [442](#)
 - l4_mem_reserved_heap, [442](#)
 - l4_mem_reserved_kernel, [442](#)
 - l4_mem_reserved_mmio, [442](#)
 - l4_mem_type_archspecific, [443](#)
 - l4_mem_type_bootloader, [442](#)
 - l4_mem_type_conventional, [442](#)
 - l4_mem_type_dedicated, [442](#)
 - l4_mem_type_info, [442](#)
 - l4_mem_type_reserved, [442](#)
 - l4_mem_type_shared, [442](#)
 - l4_mem_type_t, [442](#)
 - l4_mem_type_undefined, [442](#)
- Memory management - Data Spaces and the Region Map, [27](#)
- Memory Operations, [449](#)
 - L4_mem_op_widths, [450](#)
 - l4_mem_read, [451](#)
 - L4_MEM_WIDTH_1BYTE, [450](#)
 - L4_MEM_WIDTH_2BYTE, [450](#)
 - L4_MEM_WIDTH_4BYTE, [450](#)
 - l4_mem_write, [451](#)
- Memory related, [198](#)
 - l4_addr_consts_t, [203](#)
 - l4_bytes_to_mwords, [203](#)
 - L4_INVALID_ADDR, [203](#)
 - L4_LOG2_PAGESIZE, [200](#)
 - L4_LOG2_SUPERPAGESIZE, [200](#)
 - L4_PAGEMASK, [200](#)
 - L4_PAGESHIFT, [200](#), [201](#)
 - l4_round_page, [204](#)
 - l4_round_size, [205](#)
 - L4_SUPERPAGEMASK, [201](#)
 - L4_SUPERPAGESHIFT, [201](#), [202](#)
 - L4_SUPERPAGESIZE, [202](#)
 - l4_trunc_page, [206](#)
 - l4_trunc_size, [207](#)
- message
 - L4virtio::Svr::Bad_descriptor, [2135](#)
- Message Items, [229](#)
 - L4_FPAGE_BUFFERABLE, [231](#)
 - L4_FPAGE_CACHE_OPT, [231](#)
 - l4_fpage_cacheability_opt_t, [231](#)
 - L4_FPAGE_CACHEABLE, [231](#)
 - L4_FPAGE_UNCACHEABLE, [231](#)
 - L4_ITEM_CONT, [231](#)
 - L4_ITEM_MAP, [231](#)
 - l4_map_control, [233](#)
 - L4_MAP_ITEM_GRANT, [232](#)
 - L4_MAP_ITEM_MAP, [232](#)
 - l4_map_obj_control, [234](#)
 - l4_msg_item_consts_t, [231](#)
 - L4_RCV_ITEM_FORWARD_MAPPINGS, [232](#)
 - L4_RCV_ITEM_LOCAL_ID, [233](#)
 - L4_RCV_ITEM_SINGLE_CAP, [232](#)
- Message Registers (MRs), [269](#)
- Message Tag, [251](#)
 - l4_msgtag, [254](#)
 - L4_MSGTAG_ERROR, [253](#)

- L4_MSGTAG_FLAGS, [253](#)
- L4_msgtag_flags, [253](#)
- l4_msgtag_flags, [256](#)
- l4_msgtag_has_error, [257](#)
- l4_msgtag_is_exception, [258](#)
- l4_msgtag_is_io_page_fault, [259](#)
- l4_msgtag_is_page_fault, [260](#)
- l4_msgtag_is_sigma0, [261](#)
- l4_msgtag_items, [261](#)
- l4_msgtag_label, [262](#)
- L4_msgtag_protocol, [253](#)
- L4_MSGTAG_SCHEDULE, [253](#)
- L4_MSGTAG_TRANSFER_FPU, [253](#)
- l4_msgtag_words, [263](#)
- L4_platform_ctl_proto, [254](#)
- L4_PROTO_ALLOW_SYSCALL, [253](#)
- L4_PROTO_DEBUGGER, [254](#)
- L4_PROTO_DMA_SPACE, [254](#)
- L4_PROTO_EXCEPTION, [254](#)
- L4_PROTO_FACTORY, [254](#)
- L4_PROTO_IO_PAGE_FAULT, [254](#)
- L4_PROTO_IOMMU, [254](#)
- L4_PROTO_IRQ, [253](#)
- L4_PROTO_IRQ_SENDER, [254](#)
- L4_PROTO_KOBJECT, [254](#)
- L4_PROTO_LOG, [254](#)
- L4_PROTO_META, [254](#)
- L4_PROTO_NONE, [253](#)
- L4_PROTO_PAGE_FAULT, [253](#)
- L4_PROTO_PF_EXCEPTION, [253](#)
- L4_PROTO_PLATFORM_CTL, [254](#)
- L4_PROTO_SCHEDULER, [254](#)
- L4_PROTO_SEMAPHORE, [254](#)
- L4_PROTO_SIGMA0, [254](#)
- L4_PROTO_SMCCC, [254](#)
- L4_PROTO_TASK, [254](#)
- L4_PROTO_THREAD, [254](#)
- L4_PROTO_THREAD_GROUP, [254](#)
- L4_PROTO_VCPU_CONTEXT, [254](#)
- L4_PROTO_VM, [254](#)
- min
 - Small C++ Template Library, [642](#), [643](#)
- Minimal
 - utrace::Ring_buffer_consumer_raw< SEQUENCE_TYPE, L4virtio::Svr::Dev_config, [2205](#)
 - ITEM_TYPE, GENERATION_PTR >, [2409](#)
- mkdir
 - L4Re::Vfs::Directory, [1901](#)
- mmio_read
 - L4Re::Mmio_space, [1742](#)
- mmio_write
 - L4Re::Mmio_space, [1743](#)
- Mode
 - L4Re::Util::Event_t< PAYLOAD >, [1857](#)
- mode
 - L4::Uart, [1549](#)
- Mode_irq
 - L4Re::Util::Event_t< PAYLOAD >, [1857](#)
- Mode_polling
 - L4Re::Util::Event_t< PAYLOAD >, [1857](#)
- modify
 - L4drivers::Register_tmpl< BITS, BLOCK >, [1631](#)
- modify_senders
 - L4::Thread, [1455](#)
- Moe, [51](#)
- More
 - L4::lpc::Snd_fpage, [1224](#)
- mount
 - L4Re::Vfs::File_system, [1909](#)
 - L4Re::Vfs::Fs, [1914](#)
- move
 - L4::Cap< T >, [1044](#)
 - L4::Cap_base, [1053](#)
- Mr_bytes
 - L4::lpc::Msg, [766](#)
- Mr_words
 - L4::lpc::Msg, [766](#)
- Msb
 - cxx::Bitfield< T, LSB, MSB >, [855](#)
- msg_add
 - L4::lpc::Msg, [769](#)
- msg_get
 - L4::lpc::Msg, [770](#)
- Msg_ptr
 - L4::lpc::Msg_ptr< T >, [1200](#)
- msg_ptr
 - L4::lpc, [763](#)
- msi_info
 - L4::lcu, [1118](#)
- N
 - cxx::Bits::Direction, [919](#)
- Name-space API, [598](#)
- Name_max
 - L4Re::Inhibitor, [1711](#)
- Namespace interface, [548](#)
 - l4re_ns_query_srv, [549](#)
 - l4re_ns_query_to_srv, [550](#)
 - l4re_ns_register_flags, [549](#)
 - l4re_ns_register_obj_srv, [551](#)
- Ned, the Init Process, [56](#)
- negotiated_features
 - L4virtio::Svr::Dev_config, [2205](#)
- Net_transfer, [2366](#)
 - cur_buf, [2368](#)
 - done, [2368](#)
- new_sched
 - vmm.lua, [2475](#)
- next
 - L4Re::Event_buffer_t< PAYLOAD >, [1706](#)
 - L4virtio::Svr::Request_processor, [2238](#)
- next_avail
 - L4virtio::Svr::Virtqueue, [2316](#)
- next_bfm_t
 - L4virtio::Virtqueue::Desc::Flags, [2343](#)
- next_block
 - L4virtio::Svr::Block_request< Ds_data >, [2145](#)
- next_device

- L4vbus::Device, [2015](#)
- next_lock_info
 - L4Re::Inhibitor, [1712](#)
- next_timeout
 - L4::lpc_svr::Timeout_queue, [1283](#)
- no_demand
 - L4::Type_info::Demand, [1484](#)
- No_eager_map
 - L4Re::Rm::F, [1787](#)
 - Rm::F, [2385](#)
- No_init
 - L4::Cap_base, [1047](#)
- No_init_type
 - L4::Cap_base, [1047](#)
- no_irq_bfm_t
 - L4virtio::Virtqueue::Avail::Flags, [2340](#)
- no_notify_bfm_t
 - L4virtio::Virtqueue::Used::Flags, [2346](#)
- no_notify_guest
 - L4virtio::Virtqueue, [2329](#)
- no_notify_host
 - L4virtio::Virtqueue, [2330](#)
- No_sync
 - L4Re::Dma_space, [1673](#)
- None
 - L4::lpc::Snd_fpage, [1223](#)
 - L4::Types::Flags< BITS_ENUM, UNDERLYING >, [1523](#)
 - L4Re::Dma_space, [1673](#)
- None_type
 - L4::Types::Flags< BITS_ENUM, UNDERLYING >, [1523](#)
- Normal
 - L4Re::Dataspace::F, [1664](#)
- notify_queue
 - L4virtio::Svr::Console::Device, [2156](#)
 - L4virtio::Svr::Console::Virtio_con, [2183](#)
 - Virtio_net, [2428](#)
- NT_ASRS
 - ELF binary format, [710](#)
- NT_AUXV
 - ELF binary format, [710](#)
- NT_FPREGSET
 - ELF binary format, [710](#)
- NT_GWINDOWS
 - ELF binary format, [710](#)
- NT_LWPSINFO
 - ELF binary format, [711](#)
- NT_LWPSTATUS
 - ELF binary format, [711](#)
- NT_PLATFORM
 - ELF binary format, [710](#)
- NT_PRURED
 - ELF binary format, [711](#)
- NT_PRFPXREG
 - ELF binary format, [711](#)
- NT_PRPSINFO
 - ELF binary format, [710](#)
- NT_PRSTATUS
 - ELF binary format, [710](#)
- NT_PRXREG
 - ELF binary format, [710](#)
- NT_PSINFO
 - ELF binary format, [710](#)
- NT_PSTATUS
 - ELF binary format, [710](#)
- NT_TASKSTRUCT
 - ELF binary format, [710](#)
- NT_UTSNAME
 - ELF binary format, [711](#)
- NT_VERSION
 - ELF binary format, [711](#)
- num
 - L4virtio::Virtqueue, [2330](#)
- num_interfaces
 - L4::Meta, [1351](#)
- Obj
 - L4::lpc::Gen_fpage, [1153](#)
- obj
 - L4::lpc::Rcv_fpage, [1214](#)
 - L4::lpc::Snd_fpage, [1228](#)
- obj_cap
 - L4::Epiface, [1079](#)
 - L4::Epiface_t0< RPC_IFACE, BASE >, [1088](#)
 - L4::lrqep_t< Derived, BASE, bool >, [1310](#)
- obj_info.h
 - l4_debugger_query_obj_infos, [3327](#)
- Object Invocation, [210](#)
- l4_ipc, [213](#)
- l4_ipc_call, [214](#)
- l4_ipc_receive, [217](#)
- l4_ipc_reply, [219](#)
- l4_ipc_reply_and_wait, [220](#)
- l4_ipc_send, [221](#)
- l4_ipc_send_and_wait, [223](#)
- l4_ipc_sleep, [224](#)
- l4_ipc_sleep_ms, [225](#)
- l4_ipc_sleep_us, [226](#)
- l4_ipc_wait, [227](#)
- l4_sndfpage_add, [228](#)
- L4_SYSF_NONE, [212](#)
- Object_registry
 - L4Re::Util::Object_registry, [1871](#)
- object_size
 - cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >, [845](#)
 - cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >, [850](#)
- objects_per_slab
 - cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >, [845](#)
 - cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >, [850](#)
- offset
 - l4_sched_cpu_set_t, [1589](#)
- op_create

- Switch_factory, [2396](#)
- operator bool
 - L4virtio::Svr::Virtqueue::Head_desc, [2319](#)
- operator l4_msgtag_t
 - L4::Factory::S, [1107](#)
- operator new
 - Small C++ Template Library, [643](#)
- operator value_type
 - L4drivers::Ro_register_tmpl< BITS, BLOCK >, [1638](#)
- operator!
 - cxx::Bits::Direction, [920](#)
- operator!=
 - L4vbus::Device, [2015](#)
- operator<<
 - ipc_stream, [2708–2710](#)
 - L4::Factory::S, [1107](#), [1108](#)
- operator>>
 - ipc_stream, [2710–2714](#)
- operator()
 - cxx::Pair_first_compare< Cmp, Typ >, [962](#)
- operator->
 - cxx::Bits::Avl_set_iter< Node, Key, Node_op >, [887](#)
 - cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node, [899](#)
- operator=
 - cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, [896](#)
 - L4::Lock_guard, [1346](#)
 - L4drivers::Register_tmpl< BITS, BLOCK >, [1632](#)
 - L4Re::Rm::Unique_region< T >, [1793](#)
 - Rm::Unique_region< T >, [2391](#)
- operator==
 - L4Re::Video::Color_component, [1943](#)
 - L4Re::Video::Pixel_info, [1966](#)
 - L4vbus::Device, [2016](#)
- operator[]
 - cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >, [827](#), [828](#)
 - cxx::Bitmap_base, [876](#), [877](#)
 - cxx::List< D, Alloc >, [942](#)
 - L4drivers::Register_block< MAX_BITS, BLOCK >, [1621](#), [1622](#)
 - L4drivers::Ro_register_block< MAX_BITS, BLOCK >, [1635](#)
 - L4Re::Util::Bitmap_base, [1810](#), [1811](#)
- operator*
 - cxx::Bits::Avl_set_iter< Node, Key, Node_op >, [887](#)
 - cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node, [899](#)
- Overview, [1](#)
- Overwrite
 - L4Re::Namespace, [1749](#)
- p_flags
 - Elf32_Phdr, [1006](#)
 - Elf64_Phdr, [1016](#)
- p_type
 - Elf32_Phdr, [1006](#)
 - Elf64_Phdr, [1016](#)
- padding
 - L4Re::Video::Pixel_info, [1967](#)
- page_fault
 - L4::Pager, [1359](#)
- page_shift
 - L4Re::Util::Dataspace_svr, [1844](#)
- Pager
 - L4Re::Rm::F, [1788](#)
 - Rm::F, [2385](#)
- pager
 - L4::Thread::Attr, [1468](#)
- Pair
 - cxx::Pair< First, Second >, [960](#)
- Pair_first_compare
 - cxx::Pair_first_compare< Cmp, Typ >, [961](#)
- parent
 - L4Re::Env, [1690](#)
- Parent API, [598](#)
- Parent interface, [552](#)
- parent.h
 - l4re_parent_signal, [2920](#)
- parse_cmdline
 - Comfortable Command Line Parsing, [721](#)
- Partly_resolved
 - L4Re::Namespace, [1748](#)
- peek
 - utrace::Ring_buffer_consumer< SEQUENCE_TYPE, ITEM_TYPE, GENERATION_PTR >, [2405](#)
 - utrace::Ring_buffer_consumer_raw< SEQUENCE_TYPE, ITEM_TYPE, GENERATION_PTR >, [2410](#)
- PF_ARM_SB
 - ELF binary format, [702](#)
- PF_MASKOS
 - ELF binary format, [712](#)
- PF_MASKPROC
 - ELF binary format, [712](#)
- PF_R
 - ELF binary format, [712](#)
- PF_W
 - ELF binary format, [712](#)
- PF_X
 - ELF binary format, [712](#)
- Phys_space
 - L4Re::Dma_space, [1673](#)
- pin
 - L4vbus::Gpio_module, [2021](#)
 - L4vbus::Gpio_pin, [2030](#)
- Pinned
 - L4Re::Mem_alloc, [1734](#)
- Pixel_info
 - L4Re::Video::Pixel_info, [1960](#), [1961](#)
- pkg/drivers-frst/include/ARCH-amd64/asm_access.h, [2441](#)
- pkg/drivers-frst/include/ARCH-arm/asm_access.h, [2442](#)

- pkg/drivers-frst/include/ARCH-arm64/asm_access.h, [2443](#)
- pkg/drivers-frst/include/ARCH-mips/asm_access.h, [2444](#)
- pkg/drivers-frst/include/ARCH-ppc32/asm_access.h, [2444](#)
- pkg/drivers-frst/include/ARCH-riscv/asm_access.h, [2444](#)
- pkg/drivers-frst/include/ARCH-sparc/asm_access.h, [2445](#)
- pkg/drivers-frst/include/ARCH-x86/asm_access.h, [2446](#)
- pkg/drivers-frst/include/asm_access_gen.h, [2446](#)
- pkg/drivers-frst/include/hw_mmio_register_block, [2447](#)
- pkg/drivers-frst/include/hw_register_block, [2448](#)
- pkg/drivers-frst/include/io_regblock.h, [2451](#)
- pkg/drivers-frst/include/io_regblock_port.h, [2453](#)
- pkg/drivers-frst/include/Makefile, [2497](#)
- pkg/drivers-frst/include/poll_timeout_counter.h, [2454](#)
- pkg/drivers-frst/uart/include/device.h, [2455](#)
- pkg/drivers-frst/uart/include/Makefile, [2497](#)
- pkg/drivers-frst/uart/include/uart_16550.h, [2456](#)
- pkg/drivers-frst/uart/include/uart_16550_dw.h, [2457](#)
- pkg/drivers-frst/uart/include/uart_apb.h, [2458](#)
- pkg/drivers-frst/uart/include/uart_base.h, [2458](#)
- pkg/drivers-frst/uart/include/uart_bcm2835.h, [2460](#)
- pkg/drivers-frst/uart/include/uart_cadence.h, [2460](#)
- pkg/drivers-frst/uart/include/uart_dcc-v6.h, [2461](#)
- pkg/drivers-frst/uart/include/uart_dm.h, [2461](#)
- pkg/drivers-frst/uart/include/uart_dummy.h, [2462](#)
- pkg/drivers-frst/uart/include/uart_geni.h, [2462](#)
- pkg/drivers-frst/uart/include/uart_imx.h, [2463](#)
- pkg/drivers-frst/uart/include/uart_leon3.h, [2464](#)
- pkg/drivers-frst/uart/include/uart_linfex.h, [2465](#)
- pkg/drivers-frst/uart/include/uart_lpuart.h, [2465](#)
- pkg/drivers-frst/uart/include/uart_mvebu.h, [2466](#)
- pkg/drivers-frst/uart/include/uart_of.h, [2466](#)
- pkg/drivers-frst/uart/include/uart_omap35x.h, [2467](#)
- pkg/drivers-frst/uart/include/uart_pl011.h, [2467](#)
- pkg/drivers-frst/uart/include/uart_s3c2410.h, [2468](#)
- pkg/drivers-frst/uart/include/uart_sa1000.h, [2469](#)
- pkg/drivers-frst/uart/include/uart_sbi.h, [2469](#)
- pkg/drivers-frst/uart/include/uart_sh.h, [2470](#)
- pkg/drivers-frst/uart/include/uart_sifive.h, [2470](#)
- pkg/drivers-frst/uart/include/uart_tegra-tcu.h, [2471](#)
- pkg/l4re-core/ned/doc/tutorial.lua, [2471](#)
- pkg/l4re-core/ned/lib/include/cmd_control, [2474](#)
- pkg/l4re-core/ned/lib/include/Makefile, [2497](#)
- pkg/uvmm/configs/vmm.lua, [2475](#), [2478](#)
- pkg/virtio-net-switch/server/switch/debug.h, [2483](#)
- pkg/virtio-net-switch/server/switch/filter.cc, [2485](#)
- pkg/virtio-net-switch/server/switch/filter.h, [2486](#)
- pkg/virtio-net-switch/server/switch/mac_addr.h, [2486](#)
- pkg/virtio-net-switch/server/switch/mac_table.h, [2487](#)
- pkg/virtio-net-switch/server/switch/main.cc, [2489](#)
- pkg/virtio-net-switch/server/switch/Makefile, [2498](#)
- pkg/virtio-net-switch/server/switch/options.cc, [2498](#)
- pkg/virtio-net-switch/server/switch/options.h, [2500](#)
- pkg/virtio-net-switch/server/switch/port.h, [2501](#)
- pkg/virtio-net-switch/server/switch/port_ixl.h, [2503](#)
- pkg/virtio-net-switch/server/switch/port_l4virtio.h, [2505](#)
- pkg/virtio-net-switch/server/switch/request.h, [2509](#)
- pkg/virtio-net-switch/server/switch/request_ixl.h, [2510](#)
- pkg/virtio-net-switch/server/switch/request_l4virtio.h, [2512](#)
- pkg/virtio-net-switch/server/switch/stats.h, [2515](#)
- pkg/virtio-net-switch/server/switch/switch.cc, [2516](#)
- pkg/virtio-net-switch/server/switch/switch.h, [2519](#)
- pkg/virtio-net-switch/server/switch/virtio_net.h, [2520](#)
- pkg/virtio-net-switch/server/switch/virtio_net_buffer.h, [2524](#)
- pkg/virtio-net-switch/server/switch/vlan.h, [2524](#)
- Platform Control C API, [353](#)
 - l4_platform_ctl_cpu_allow_shutdown, [354](#)
 - l4_platform_ctl_cpu_disable, [355](#)
 - l4_platform_ctl_cpu_enable, [355](#)
 - l4_platform_ctl_set_task_asid, [356](#)
 - l4_platform_ctl_system_shutdown, [357](#)
 - l4_platform_ctl_system_suspend, [358](#)
- pm_resume
 - L4vbus::Pm< DEC >, [2051](#)
- pm_suspend
 - L4vbus::Pm< DEC >, [2051](#)
- Poll_timeout_counter
 - L4::Poll_timeout_counter, [1370](#)
- Poll_timeout_kipclock
 - L4::Poll_timeout_kipclock, [1372](#)
- pop_front
 - cxx::Bits::Smart_ptr_list< ITEM >, [923](#)
 - cxx::H_list< T, POLICY >, [932](#)
 - cxx::S_list< T, POLICY >, [973](#)
- port
 - L4virtio::Svr::Console::Device, [2157](#)
 - L4virtio::Svr::Console::Virtio_con, [2184](#)
- port_add
 - L4virtio::Svr::Console::Virtio_con, [2185](#)
- Port_added
 - L4virtio::Svr::Console::Port, [2175](#)
- port_available
 - Virtio_switch, [2435](#)
- Port_disabled
 - L4virtio::Svr::Console::Port, [2175](#)
- Port_failed
 - L4virtio::Svr::Console::Port, [2175](#)
- Port_name
 - L4virtio::Svr::Console::Control_message, [2147](#)
- port_name
 - L4virtio::Svr::Console::Virtio_con, [2186](#)
- Port_num_states
 - L4virtio::Svr::Console::Port, [2175](#)
- Port_open
 - L4virtio::Svr::Console::Control_message, [2147](#)
 - L4virtio::Svr::Console::Port, [2175](#)
- port_open
 - L4virtio::Svr::Console::Virtio_con, [2187](#)
- port_read
 - L4virtio::Svr::Console::Device, [2158](#)

- Port_ready
 - L4virtio::Svr::Console::Control_message, [2147](#)
 - L4virtio::Svr::Console::Port, [2175](#)
- port_remove
 - L4virtio::Svr::Console::Virtio_con, [2188](#)
- Port_status
 - L4virtio::Svr::Console::Port, [2175](#)
- port_write
 - L4virtio::Svr::Console::Device, [2159](#)
- Ports
 - L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS >, [1488](#)
- print
 - L4Re::Log, [1729](#)
- println
 - L4Re::Log, [1729](#)
- prio
 - l4_sched_param_t, [1593](#)
- process
 - L4Re::Util::Event_buffer_consumer_t< PAYLOAD >, [1848](#)
- process_device_ready
 - L4virtio::Svr::Console::Device, [2161](#)
 - L4virtio::Svr::Console::Virtio_con, [2189](#)
- process_port_open
 - L4virtio::Svr::Console::Device, [2162](#)
 - L4virtio::Svr::Console::Virtio_con, [2190](#)
- process_port_ready
 - L4virtio::Svr::Console::Device, [2162](#)
 - L4virtio::Svr::Console::Virtio_con, [2190](#)
- process_request
 - L4virtio::Driver::Block_device, [2093](#)
- process_used_queue
 - L4virtio::Driver::Block_device, [2094](#)
- Producer, [618](#), [629](#)
 - l4shmc_chunk_ready, [618](#)
 - l4shmc_chunk_ready_sig, [619](#)
 - l4shmc_chunk_try_to_take, [619](#)
 - l4shmc_chunk_try_to_take_for_overwriting, [620](#)
 - l4shmc_chunk_try_to_take_for_writing, [620](#)
 - l4shmc_is_chunk_clear, [621](#)
 - l4shmc_trigger, [629](#)
- prog.mk - Application Role, [35](#)
- Program Input and Output, [28](#)
- Programming for L4Re, [7](#)
- PROTO_ANY
 - L4, [748](#)
- proto_dispatch
 - L4::Kobject_typeid< T >, [1338](#)
 - L4::Kobject_typeid< void >, [1342](#)
 - L4::Server_object_t< IFACE, BASE >, [1426](#)
- PROTO_EMPTY
 - L4, [748](#)
- PT_DYNAMIC
 - ELF binary format, [712](#)
- PT_GNU_EH_FRAME
 - ELF binary format, [712](#)
- PT_GNU_RELRO
 - ELF binary format, [712](#)
- PT_GNU_STACK
 - ELF binary format, [712](#)
- PT_HIOS
 - ELF binary format, [712](#)
- PT_HIPROC
 - ELF binary format, [712](#)
- PT_INTERP
 - ELF binary format, [712](#)
- PT_L4_AUX
 - ELF binary format, [712](#)
- PT_L4_STACK
 - ELF binary format, [712](#)
- PT_LOAD
 - ELF binary format, [712](#)
- PT_LOOS
 - ELF binary format, [712](#)
- PT_LOPROC
 - ELF binary format, [712](#)
- PT_NOTE
 - ELF binary format, [712](#)
- PT_NULL
 - ELF binary format, [712](#)
- PT_NUM
 - ELF binary format, [712](#)
- PT_PHDR
 - ELF binary format, [712](#)
- PT_SHLIB
 - ELF binary format, [712](#)
- PT_TLS
 - ELF binary format, [712](#)
- Pthread Support, [29](#)
- ptr
 - cxx::Ref_ptr< T, CNT >, [968](#)
 - L4Re::Core::Ref_ptr< T, CNT >, [1649](#)
- push_back
 - cxx::List_item, [949](#)
- push_front
 - cxx::List_item, [949](#)
- put
 - L4::Factory::S, [1109](#), [1110](#)
 - L4::lpc::lostream, [1160](#), [1161](#)
 - L4::lpc::Ostream, [1205](#), [1206](#)
 - L4Re::Event_buffer_t< PAYLOAD >, [1706](#)
- qconfig
 - L4virtio::Svr::Dev_config, [2206](#)
- query
 - L4Re::Namespace, [1749](#), [1750](#)
- query_log_name
 - L4::Debugger, [1066](#)
- query_log_typeid
 - L4::Debugger, [1067](#)
- query_object_name
 - L4::Debugger, [1068](#)
- Query_result_flags
 - L4Re::Namespace, [1748](#)
- Query_timeout
 - L4Re::Namespace, [1748](#)

- quota
 - L4Re::Mem_alloc::Stats, [1738](#)
- quota_used
 - L4Re::Mem_alloc::Stats, [1738](#)
- R
 - cxx::Bits::Direction, [919](#)
 - L4Re::Dataspace::F, [1664](#)
 - L4Re::Rm::F, [1788](#)
 - Rm::F, [2385](#)
- r
 - L4drivers::Register_block< MAX_BITS, BLOCK >, [1622](#), [1623](#)
 - L4drivers::Ro_register_block< MAX_BITS, BLOCK >, [1635](#)
 - L4Re::Video::Pixel_info, [1967](#), [1968](#)
 - L4vcpu::Vcpu, [2077](#)
- R_386_32
 - ELF binary format, [713](#)
- R_386_COPY
 - ELF binary format, [713](#)
- R_386_GLOB_DAT
 - ELF binary format, [713](#)
- R_386_GOT32
 - ELF binary format, [713](#)
- R_386_GOTOFF
 - ELF binary format, [713](#)
- R_386_GOTPC
 - ELF binary format, [713](#)
- R_386_JMP_SLOT
 - ELF binary format, [713](#)
- R_386_NONE
 - ELF binary format, [713](#)
- R_386_NUM
 - ELF binary format, [713](#)
- R_386_PC32
 - ELF binary format, [713](#)
- R_386_PLT32
 - ELF binary format, [713](#)
- R_386_RELATIVE
 - ELF binary format, [713](#)
- R_386_TLS_DTPMOD32
 - ELF binary format, [713](#)
- R_386_TLS_DTPOFF32
 - ELF binary format, [713](#)
- R_386_TLS_GD
 - ELF binary format, [713](#)
- R_386_TLS_GD_32
 - ELF binary format, [713](#)
- R_386_TLS_GD_CALL
 - ELF binary format, [713](#)
- R_386_TLS_GD_POP
 - ELF binary format, [713](#)
- R_386_TLS_GD_PUSH
 - ELF binary format, [713](#)
- R_386_TLS_GOTIE
 - ELF binary format, [713](#)
- R_386_TLS_IE
 - ELF binary format, [713](#)
- R_386_TLS_IE_32
 - ELF binary format, [713](#)
- R_386_TLS_LDM
 - ELF binary format, [713](#)
- R_386_TLS_LDM_32
 - ELF binary format, [713](#)
- R_386_TLS_LDM_CALL
 - ELF binary format, [713](#)
- R_386_TLS_LDM_POP
 - ELF binary format, [713](#)
- R_386_TLS_LDM_PUSH
 - ELF binary format, [713](#)
- R_386_TLS_LDO_32
 - ELF binary format, [713](#)
- R_386_TLS_LE
 - ELF binary format, [713](#)
- R_386_TLS_LE_32
 - ELF binary format, [713](#)
- R_386_TLS_TPOFF
 - ELF binary format, [713](#)
- R_386_TLS_TPOFF32
 - ELF binary format, [713](#)
- R_AARCH64_NONE
 - ELF binary format, [714](#)
- R_ARM_ABS12
 - ELF binary format, [714](#)
- R_ARM_ABS16
 - ELF binary format, [714](#)
- R_ARM_ABS32
 - ELF binary format, [714](#)
- R_ARM_ABS8
 - ELF binary format, [714](#)
- R_ARM_COPY
 - ELF binary format, [714](#)
- R_ARM_GLOB_DAT
 - ELF binary format, [714](#)
- R_ARM_GOT32
 - ELF binary format, [714](#)
- R_ARM_GOTOFF
 - ELF binary format, [714](#)
- R_ARM_GOTPC
 - ELF binary format, [714](#)
- R_ARM_JUMP_SLOT
 - ELF binary format, [714](#)
- R_ARM_NONE
 - ELF binary format, [714](#)
- R_ARM_NUM
 - ELF binary format, [714](#)
- R_ARM_PC24
 - ELF binary format, [714](#)
- R_ARM_PLT32
 - ELF binary format, [714](#)
- R_ARM_REL32
 - ELF binary format, [714](#)
- R_ARM_RELATIVE
 - ELF binary format, [714](#)
- R_ARM_THM_PC11
 - ELF binary format, [714](#)

- R_ARM_THM_PC9
 - ELF binary format, [714](#)
- R_X86_64_16
 - ELF binary format, [715](#)
- R_X86_64_32
 - ELF binary format, [715](#)
- R_X86_64_32S
 - ELF binary format, [715](#)
- R_X86_64_64
 - ELF binary format, [715](#)
- R_X86_64_8
 - ELF binary format, [715](#)
- R_X86_64_COPY
 - ELF binary format, [715](#)
- R_X86_64_DTPMOD64
 - ELF binary format, [715](#)
- R_X86_64_DTPOFF32
 - ELF binary format, [715](#)
- R_X86_64_DTPOFF64
 - ELF binary format, [715](#)
- R_X86_64_GLOB_DAT
 - ELF binary format, [715](#)
- R_X86_64_GOT32
 - ELF binary format, [715](#)
- R_X86_64_GOTPCREL
 - ELF binary format, [715](#)
- R_X86_64_GOTTPOFF
 - ELF binary format, [715](#)
- R_X86_64_JUMP_SLOT
 - ELF binary format, [715](#)
- R_X86_64_NONE
 - ELF binary format, [715](#)
- R_X86_64_PC16
 - ELF binary format, [715](#)
- R_X86_64_PC32
 - ELF binary format, [715](#)
- R_X86_64_PC8
 - ELF binary format, [715](#)
- R_X86_64_PLT32
 - ELF binary format, [715](#)
- R_X86_64_RELATIVE
 - ELF binary format, [715](#)
- R_X86_64_TLSGD
 - ELF binary format, [715](#)
- R_X86_64_TLSLD
 - ELF binary format, [715](#)
- R_X86_64_TPOFF32
 - ELF binary format, [715](#)
- R_X86_64_TPOFF64
 - ELF binary format, [715](#)
- raise
 - L4Re::Itas, [1718](#)
- RAM configuration, [81](#)
- Random number support, [723](#)
 - l4util_rand, [723](#)
 - l4util_srand, [723](#)
- rate
 - L4::Uart, [1549](#)
- rbegin
 - cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, [896](#)
 - cxx::Bits::Bst< Node, Get_key, Compare >, [912](#)
- rcv_cap
 - L4::lpc_svr::Server_iface, [1275](#)
- rcv_endpoint.h
 - L4_RCV_EP_BIND_OP, [3341](#)
 - L4_rcv_ep_ops, [3341](#)
- Rcv_fpage
 - L4::lpc::Rcv_fpage, [1212](#)
- rcv_task
 - L4::lpc::Rcv_fpage, [1215](#)
- read
 - L4::lpc, [763](#)
 - L4::Vcon, [1565](#)
 - L4drivers::Ro_register_tmpl< BITS, BLOCK >, [1638](#)
- read_with_flags
 - L4::Vcon, [1566](#)
- readv
 - L4Re::Vfs::Regular_file, [1935](#)
- ready
 - L4virtio::Virtqueue, [2331](#)
- Ready_type
 - L4Re::Vfs::Generic_file, [1918](#)
- realloc_rcv_cap
 - L4::lpc_svr::Server_iface, [1276](#)
 - L4Re::Util::Br_manager, [1821](#)
- Realtime API, [251](#)
- receive
 - L4::lpc::Istream, [1170](#)
 - L4::lrq, [1298](#)
- Receiver, [605](#)
- Ref_ptr
 - cxx::Ref_ptr< T, CNT >, [967](#)
 - L4Re::Core::Ref_ptr< T, CNT >, [1648](#)
- refresh
 - L4Re::Util::Video::Goos_svr, [1889](#)
 - L4Re::Video::View, [1972](#)
- reg_shift
 - L4::Uart, [1549](#)
- Region map API, [600](#)
- Region map interface, [552](#)
 - l4re_rm_attach, [554](#)
 - l4re_rm_attach_srv, [556](#)
 - l4re_rm_attach_w_info, [556](#)
 - l4re_rm_attach_w_info_srv, [558](#)
 - L4RE_RM_CACHING_SHIFT, [554](#)
 - l4re_rm_detach, [558](#)
 - l4re_rm_detach_ds, [559](#)
 - l4re_rm_detach_ds_unmap, [560](#)
 - l4re_rm_detach_srv, [561](#)
 - l4re_rm_detach_unmap, [562](#)
 - L4RE_RM_F_ATTACH_FLAGS, [554](#)
 - L4RE_RM_F_CACHE_BUFFERED, [554](#)
 - L4RE_RM_F_CACHE_NORMAL, [554](#)
 - L4RE_RM_F_CACHE_UNCACHED, [554](#)

- L4RE_RM_F_CACHING, [554](#)
- L4RE_RM_F_DETACH_FREE, [554](#)
- L4RE_RM_F_EAGER_MAP, [554](#)
- L4RE_RM_F_IN_AREA, [554](#)
- L4RE_RM_F_KERNEL, [554](#)
- L4RE_RM_F_NO_EAGER_MAP, [554](#)
- L4RE_RM_F_PAGER, [554](#)
- L4RE_RM_F_R, [554](#)
- L4RE_RM_F_RESERVED, [554](#)
- L4RE_RM_F_SEARCH_ADDR, [554](#)
- l4re_rm_find, [563](#)
- l4re_rm_find_srv, [564](#)
- l4re_rm_flags_values, [554](#)
- l4re_rm_free_area, [565](#)
- l4re_rm_free_area_srv, [566](#)
- l4re_rm_get_info, [566](#)
- l4re_rm_get_info_srv, [567](#)
- L4RE_RM_REGION_FLAGS, [554](#)
- l4re_rm_reserve_area, [568](#)
- l4re_rm_reserve_area_srv, [569](#)
- l4re_rm_show_lists, [570](#)
- Region_flag_shifts
 - L4Re::Rm, [1771](#)
 - Rm, [2374](#)
- Region_flags
 - L4Re::Rm::F, [1787](#)
 - Rm::F, [2385](#)
- Region_flags_mask
 - L4Re::Rm::F, [1788](#)
 - Rm::F, [2386](#)
- register_del_irq
 - L4::Thread, [1456](#)
- register_doorbell_irq
 - L4::Thread, [1457](#)
- register_driver_irq
 - L4virtio::Svr::Device_t< DATA >, [2221](#)
- register_ds
 - L4virtio::Device, [2087](#)
 - L4virtio::Driver::Device, [2106](#)
- Register_flags
 - L4Re::Namespace, [1748](#)
- register_irq_obj
 - L4::Registry_iface, [1383](#)
 - L4Re::Util::Object_registry, [1872](#)
- register_obj
 - L4::Registry_iface, [1383](#), [1384](#)
 - L4Re::Namespace, [1751](#)
 - L4Re::Util::Object_registry, [1872](#), [1873](#)
- register_thread
 - L4Re::Itas, [1719](#)
- Registry_server
 - L4Re::Util::Registry_server< LOOP_HOOKS >, [1881](#), [1882](#)
- release
 - cxx::Ref_ptr< T, CNT >, [968](#)
 - L4Re::Core::Ref_ptr< T, CNT >, [1649](#)
 - L4Re::Inhibitor, [1713](#)
 - L4Re::Rm::Unique_region< T >, [1794](#)
 - L4Re::Util::Counting_cap_alloc< COUNTER-TYPE, Dbg >, [1837](#)
 - L4Re::Util::Dataspace_srv, [1845](#)
 - Rm::Unique_region< T >, [2391](#)
- release_cap
 - L4::Task, [1445](#)
- release_ioport
 - L4vbus::Vbus, [2061](#)
- remove
 - cxx::Avl_tree< Node, Get_key, Compare >, [838](#)
 - cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, [896](#)
 - cxx::H_list< T, POLICY >, [933](#)
 - cxx::List_item, [950](#)
 - L4::lpc_srv::Timeout_queue, [1284](#)
 - L4::Thread_group, [1475](#)
 - L4virtio::Svr::Driver_mem_list_t< DATA >, [2229](#)
 - Virtio_vlan_mangle, [2438](#)
- remove_all
 - cxx::Bits::Bst< Node, Get_key, Compare >, [913](#)
- remove_timeout
 - L4::lpc_srv::Server_iface, [1277](#)
 - L4::lpc_srv::Timeout_queue_hooks< HOOKS, BR_MAN >, [1289](#)
- remove_tree
 - cxx::Bits::Bst< Node, Get_key, Compare >, [914](#)
- rename
 - L4Re::Vfs::Directory, [1902](#)
- rend
 - cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >, [897](#)
 - cxx::Bits::Bst< Node, Get_key, Compare >, [915](#)
- replace
 - cxx::H_list< T, POLICY >, [933](#)
- reply
 - L4::Reply_cap, [1388](#)
- reply_and_wait
 - L4::lpc::loststream, [1161](#), [1162](#)
- Reply_cap
 - L4::Reply_cap, [1387](#)
- reply_cap_alloc
 - L4Re Capability API, [590](#)
- Reply_compound
 - Server-Side IPC framework, [646](#)
- Reply_mode
 - Server-Side IPC framework, [646](#)
- Reply_separate
 - Server-Side IPC framework, [646](#)
- request_ioport
 - L4vbus::Vbus, [2061](#)
- reserve_area
 - L4Re::Rm, [1783](#), [1784](#)
 - Rm, [2381](#), [2382](#)
- Reserved
 - L4::Kip::Mem_desc, [1314](#)
 - L4Re::Rm::F, [1788](#)
 - Rm::F, [2385](#)
- Reserved_heap

- L4::Kip::Mem_desc, 1314
- Reserved_kernel
 - L4::Kip::Mem_desc, 1314
- Reserved_mmio
 - L4::Kip::Mem_desc, 1314
- reset
 - L4::lpc::loststream, 1163
 - L4::lpc::lstream, 1171
 - L4::Reply_cap, 1388
 - L4Re::Rm::Unique_region< T >, 1794
 - Rm::Unique_region< T >, 2391
- reset_cmd
 - L4virtio::Svr::Dev_config, 2206
- reset_device
 - L4virtio::Svr::Console::Virtio_con, 2191
- reset_queue
 - L4virtio::Svr::Dev_config, 2207
- reset_queue_config
 - L4virtio::Svr::Device_t< DATA >, 2221
- Resize
 - L4virtio::Svr::Console::Control_message, 2147
- Result
 - cxx::Result< T >, 969
- rewind_avail
 - L4virtio::Svr::Virtqueue, 2317
- rewrite_hdr
 - Virtio_vlan_mangle, 2439
- Rights_mask
 - L4::lpc::Cap< T >, 1148
 - L4Re::Dataspace::F, 1664
 - L4Re::Rm::F, 1788
 - Rm::F, 2385
- Ring_buffer_consumer
 - utrace::Ring_buffer_consumer< SEQUENCE_TYPE, ITEM_TYPE, GENERATION_PTR >, 2404
- Ring_buffer_consumer_raw
 - utrace::Ring_buffer_consumer_raw< SEQUENCE_TYPE, ITEM_TYPE, GENERATION_PTR >, 2410
- Ring_buffer_producer
 - utrace::Ring_buffer_producer< SEQUENCE_TYPE, ITEM_TYPE, GENERATION_PTR >, 2413
- ringbuf.h
 - l4shmc_rb_attach_receiver, 3081
 - l4shmc_rb_attach_sender, 3082
 - l4shmc_rb_deinit_buffer, 3082
 - l4shmc_rb_init_buffer, 3082
 - l4shmc_rb_init_receiver, 3083
 - l4shmc_rb_receiver_copy_out, 3084
 - l4shmc_rb_receiver_notify_done, 3084
 - l4shmc_rb_receiver_read_next_size, 3085
 - l4shmc_rb_receiver_wait_for_data, 3085
 - l4shmc_rb_sender_alloc_packet, 3085
 - l4shmc_rb_sender_commit_packet, 3086
 - l4shmc_rb_sender_next_copy_in, 3086
 - l4shmc_rb_sender_put_data, 3087
- RISC-V Virtual Registers (UTCB), 278
- riscv/l4/sys/__kip-arch.h, 2566
- riscv/l4/sys/__vcpu-arch.h, 2576, 2577
- riscv/l4/sys/arch/cache.h, 3143, 3144
- riscv/l4/sys/arch/consts.h, 3165, 3166
- riscv/l4/sys/arch/ipc.h, 3277
- riscv/l4/sys/arch/kip.h, 3464
- riscv/l4/sys/arch/l4int.h, 3318, 3319
- riscv/l4/sys/arch/platform_control.h, 3337
- riscv/l4/sys/arch/task.h, 3367
- riscv/l4/sys/arch/thread.h, 2625
- riscv/l4/sys/arch/utcb.h, 3395, 3396
- riscv/l4/sys/ktrace_events.h, 2589
- riscv/l4/sys/linkage.h, 2600
- riscv/l4/sys/vm.h, 2607
- Rm, 2369
 - attach, 2374, 2375
 - Caching_shift, 2374
 - detach, 2377, 2378
 - Detach_again, 2374
 - Detach_exact, 2373
 - Detach_flags, 2373
 - Detach_keep, 2373
 - Detach_overlap, 2373
 - Detach_result, 2373
 - Detached_ds, 2374
 - find, 2378
 - free_area, 2379
 - get_areas, 2380
 - get_info, 2380
 - get_regions, 2381
 - Kept_ds, 2374
 - Region_flag_shifts, 2374
 - reserve_area, 2381, 2382
 - Split_ds, 2374
- rm
 - L4Re::Env, 1690, 1691
 - Rm::Area, 2383
 - Rm::F, 2384
 - Attach_flags, 2385
 - Attach_mask, 2385
 - Cache_buffered, 2386
 - Cache_normal, 2386
 - Cache_uncached, 2386
 - Caching_mask, 2386
 - Detach_free, 2385
 - Ds_map_mask, 2386
 - Eager_map, 2385
 - In_area, 2385
 - Kernel, 2385
 - No_eager_map, 2385
 - Pager, 2385
 - R, 2385
 - Region_flags, 2385
 - Region_flags_mask, 2386
 - Reserved, 2385
 - Rights_mask, 2385
 - RW, 2385
 - RWX, 2385
 - RX, 2385
 - Search_addr, 2385

- W, [2385](#)
- X, [2385](#)
- Rm::Region, [2386](#)
- Rm::Unique_region< T >, [2387](#)
 - ~Unique_region, [2390](#)
 - get, [2390](#)
 - is_valid, [2390](#)
 - operator=, [2391](#)
 - release, [2391](#)
 - reset, [2391](#)
 - Unique_region, [2389](#), [2390](#)
- rmdir
 - L4Re::Vfs::Directory, [1902](#)
- Ro
 - L4Re::Dataspace::F, [1664](#)
 - L4Re::Namespace, [1749](#)
- root
 - L4vbus::Vbus, [2062](#)
- round_order
 - L4, [754](#)
- Rs
 - L4Re::Namespace, [1749](#)
- RTC driver, [82](#)
- run_thread
 - L4::Scheduler, [1404](#)
- running
 - L4virtio::Svr::Dev_status, [2216](#)
- Runtime_error
 - L4::Runtime_error, [1395](#)
- RW
 - L4Re::Dataspace::F, [1664](#)
 - L4Re::Rm::F, [1788](#)
 - Rm::F, [2385](#)
- Rw
 - L4Re::Namespace, [1749](#)
- Rws
 - L4Re::Namespace, [1749](#)
- RWX
 - L4Re::Dataspace::F, [1664](#)
 - L4Re::Rm::F, [1788](#)
 - Rm::F, [2385](#)
- RX
 - L4Re::Dataspace::F, [1664](#)
 - L4Re::Rm::F, [1788](#)
 - Rm::F, [2385](#)
- rx_pkt
 - L4virtio::Driver::Virtio_net_device, [2115](#)
- rx_queue_size
 - L4virtio::Driver::Virtio_net_device, [2116](#)
- S
 - L4::Factory::S, [1106](#)
- saved_state
 - L4vcpu::Vcpu, [2077](#)
- scan_zero
 - cxx::Bitmap< BITS >, [867](#)
 - cxx::Bitmap_base, [878](#)
 - L4Re::Util::Bitmap< BITS >, [1804](#)
 - L4Re::Util::Bitmap_base, [1811](#)
- Scheduler, [359](#)
 - l4_sched_cpu_set, [361](#)
 - l4_sched_param, [362](#)
 - L4_SCHEDULER_CLASS_FIXED_PRIO, [361](#)
 - L4_SCHEDULER_CLASS_WFQ, [361](#)
 - L4_scheduler_classes, [361](#)
 - l4_scheduler_idle_time, [363](#)
 - L4_SCHEDULER_IDLE_TIME_OP, [361](#)
 - l4_scheduler_info, [364](#)
 - L4_SCHEDULER_INFO_OP, [361](#)
 - l4_scheduler_info_with_classes, [364](#)
 - l4_scheduler_is_online, [365](#)
 - L4_scheduler_ops, [361](#)
 - l4_scheduler_run_thread, [366](#)
 - L4_SCHEDULER_RUN_THREAD_OP, [361](#)
- scheduler
 - L4Re::Env, [1691](#), [1692](#)
- screen_info
 - L4Re::Util::Video::Goos_svr, [1890](#)
- Search_addr
 - L4Re::Rm::F, [1787](#)
 - Rm::F, [2385](#)
- segment.h
 - fiasco_amd64_segment_info, [2527](#)
 - fiasco_amd64_set_fs, [2528](#)
 - fiasco_amd64_set_segment_base, [2529](#)
 - L4_AMD64_SEGMENT_FS, [2527](#)
 - L4_AMD64_SEGMENT_GS, [2527](#)
 - L4_sys_segment, [2527](#)
 - L4_task_ldt_x86_consts, [2527](#), [2533](#)
 - L4_TASK_LDT_X86_ENTRY_SIZE, [2527](#), [2533](#)
 - L4_TASK_LDT_X86_MAX_ENTRIES, [2527](#), [2533](#)
- send
 - L4::lpc::Ostream, [1206](#)
 - L4::Vcon, [1567](#)
 - L4virtio::Driver::Device, [2107](#)
- send_and_wait
 - L4virtio::Driver::Device, [2108](#)
- send_control_message
 - L4virtio::Svr::Console::Virtio_con, [2192](#)
- send_request
 - L4virtio::Driver::Block_device, [2095](#)
- Sender, [605](#)
- Server
 - L4::Server< LOOP_HOOKS >, [1415](#)
- Server-Side IPC framework, [644](#)
 - Reply_compound, [646](#)
 - Reply_mode, [646](#)
 - Reply_separate, [646](#)
- server_iface
 - L4::Epiface, [1080](#)
- set
 - cxx::Bitfield< T, LSB, MSB >, [855](#)
 - L4::Kip::Mem_desc, [1320](#)
 - L4::Poll_timeout_counter, [1370](#)
 - L4::Poll_timeout_kipclock, [1373](#)
 - l4_sched_cpu_set_t, [1590](#)
 - L4drivers::Register_tmpl< BITS, BLOCK >, [1632](#)

- L4Re::Video::Color_component, [1944](#)
- L4vbus::Gpio_module, [2022](#)
- L4vbus::Gpio_pin, [2030](#)
- L4vcpu::State, [2067](#)
- L4virtio::Svr::Data_buffer, [2197](#)
- set_attr
 - L4::Vcon, [1568](#)
- set_bit
 - cxx::Bitmap_base, [879](#)
 - L4Re::Util::Bitmap_base, [1812](#)
- set_blk_size
 - L4virtio::Svr::Block_dev_base< Ds_data >, [2141](#)
- set_config_wce
 - L4virtio::Svr::Block_dev_base< Ds_data >, [2141](#)
- set_device_needs_reset
 - L4virtio::Svr::Dev_config, [2208](#)
- set_device_notify_index
 - L4virtio::Svr::Dev_config, [2208](#)
- set_dirty
 - cxx::Bitfield< T, LSB, MSB >, [856](#)
- set_discard
 - L4virtio::Svr::Block_dev_base< Ds_data >, [2142](#)
- set_fd
 - L4Re::Vfs::Fs, [1915](#)
- set_info
 - L4Re::Video::View, [1973](#)
- set_lock
 - L4Re::Vfs::Regular_file, [1936](#)
- set_mode
 - L4::lcu, [1119](#)
- set_object_name
 - L4::Debugger, [1069](#)
- set_raw
 - l4_vcon_attr_t, [1598](#)
- set_rcv_cap_flags
 - L4Re::Util::Br_manager, [1822](#)
- set_sched
 - vmm.lua, [2475](#)
- set_server
 - L4::Epiface, [1081](#)
- set_size_max
 - L4virtio::Svr::Block_dev_base< Ds_data >, [2142](#)
- set_status
 - L4virtio::Device, [2088](#)
 - L4virtio::Svr::Dev_config, [2209](#)
- set_status_flags
 - L4Re::Vfs::Generic_file, [1921](#)
- set_topology
 - L4virtio::Svr::Block_dev_base< Ds_data >, [2142](#)
- set_unshifted
 - cxx::Bitfield< T, LSB, MSB >, [857](#)
- set_unshifted_dirty
 - cxx::Bitfield< T, LSB, MSB >, [857](#)
- set_viewport
 - L4Re::Video::View, [1974](#)
- set_write_zeroes
 - L4virtio::Svr::Block_dev_base< Ds_data >, [2143](#)
- settimer
 - L4Re::Itas, [1719](#)
- setup
 - L4Re::Util::Counting_cap_alloc< COUNTER-
TYPE, Dbg >, [1838](#)
 - L4vbus::Gpio_module, [2023](#)
 - L4vbus::Gpio_pin, [2031](#)
 - L4virtio::Virtqueue, [2332](#)
- setup_device
 - L4virtio::Driver::Block_device, [2095](#)
 - L4virtio::Driver::Virtio_net_device, [2116](#)
- setup_queue
 - L4virtio::Svr::Device_t< DATA >, [2222](#)
- setup_simple
 - L4virtio::Virtqueue, [2334](#)
- sh_flags
 - Elf32_Shdr, [1009](#)
 - Elf64_Shdr, [1019](#)
- sh_type
 - Elf32_Shdr, [1009](#)
 - Elf64_Shdr, [1019](#)
- Shared
 - L4::Kip::Mem_desc, [1314](#)
- Shared Memory Library, [607](#)
 - l4shmc_area_overhead, [609](#)
 - l4shmc_area_size, [609](#)
 - l4shmc_area_size_free, [609](#)
 - l4shmc_attach, [609](#)
 - l4shmc_chunk_overhead, [610](#)
 - l4shmc_connect_chunk_signal, [610](#)
 - l4shmc_create, [611](#)
 - l4shmc_get_client_nr, [611](#)
 - l4shmc_get_initialized_clients, [612](#)
 - l4shmc_mark_client_initialized, [612](#)
- Shared_cap
 - L4Re, [777](#)
 - L4Re::Util, [802](#)
- shared_cap_cast
 - L4Re, [788](#), [789](#)
 - L4Re::Util, [806](#), [807](#)
- shared_cap_dynamic_cast
 - L4Re, [790](#), [791](#)
 - L4Re::Util, [808](#), [809](#)
- shared_cap_reinterpret_cast
 - L4Re, [791](#), [792](#)
 - L4Re::Util, [809](#), [810](#)
- Shared_del_cap
 - L4Re, [777](#)
 - L4Re::Util, [803](#)
- shared_del_cap_cast
 - L4Re, [793](#), [794](#)
 - L4Re::Util, [811](#), [812](#)
- shared_del_cap_dynamic_cast
 - L4Re, [795](#)
 - L4Re::Util, [813](#)
- shared_del_cap_reinterpret_cast
 - L4Re, [796](#), [797](#)
 - L4Re::Util, [814](#), [815](#)
- SHF_ALLOC

- ELF binary format, [716](#)
- SHF_ARM_COMDEF
 - ELF binary format, [716](#)
- SHF_ARM_ENTRYSECT
 - ELF binary format, [715](#)
- SHF_EXECINSTR
 - ELF binary format, [716](#)
- SHF_GROUP
 - ELF binary format, [716](#)
- SHF_INFO_LINK
 - ELF binary format, [716](#)
- SHF_LINK_ORDER
 - ELF binary format, [716](#)
- SHF_MASKOS
 - ELF binary format, [716](#)
- SHF_MASKPROC
 - ELF binary format, [716](#)
- SHF_MERGE
 - ELF binary format, [716](#)
- SHF_OS_NONCONFORMING
 - ELF binary format, [716](#)
- SHF_STRINGS
 - ELF binary format, [716](#)
- SHF_TLS
 - ELF binary format, [716](#)
- SHF_WRITE
 - ELF binary format, [716](#)
- shift
 - L4Re::Video::Color_component, [1944](#)
- Shift_type
 - cxx::Bitfield< T, LSB, MSB >, [854](#)
- SHN_ABS
 - ELF binary format, [716](#)
- SHN_COMMON
 - ELF binary format, [716](#)
- SHN_HIPROC
 - ELF binary format, [716](#)
- SHN_HIRESERVE
 - ELF binary format, [716](#)
- SHN_LOPROC
 - ELF binary format, [716](#)
- SHN_LORESERVE
 - ELF binary format, [716](#)
- SHN_UNDEF
 - ELF binary format, [716](#)
- SHT_DYNAMIC
 - ELF binary format, [717](#)
- SHT_DYNSYM
 - ELF binary format, [717](#)
- SHT_FINI_ARRAY
 - ELF binary format, [717](#)
- SHT_GROUP
 - ELF binary format, [717](#)
- SHT_HASH
 - ELF binary format, [717](#)
- SHT_HIOS
 - ELF binary format, [717](#)
- SHT_HIPROC
 - ELF binary format, [717](#)
- SHT_HIUSER
 - ELF binary format, [717](#)
- SHT_INIT_ARRAY
 - ELF binary format, [717](#)
- SHT_LOOS
 - ELF binary format, [717](#)
- SHT_LOPROC
 - ELF binary format, [717](#)
- SHT_LOUSER
 - ELF binary format, [717](#)
- SHT_NOBITS
 - ELF binary format, [717](#)
- SHT_NOTE
 - ELF binary format, [717](#)
- SHT_NULL
 - ELF binary format, [717](#)
- SHT_NUM
 - ELF binary format, [717](#)
- SHT_PREINIT_ARRAY
 - ELF binary format, [717](#)
- SHT_PROGBITS
 - ELF binary format, [717](#)
- SHT_REL
 - ELF binary format, [717](#)
- SHT_RELA
 - ELF binary format, [717](#)
- SHT_SHLIB
 - ELF binary format, [717](#)
- SHT_STRTAB
 - ELF binary format, [717](#)
- SHT_SYMTAB
 - ELF binary format, [717](#)
- SHT_SYMTAB_SHNDX
 - ELF binary format, [717](#)
- shutdown
 - L4::Uart, [1550](#)
 - L4::Uart_apb, [1556](#)
- si
 - l4_vcpu_regs_t, [1604](#)
- sigaction
 - L4Re::Itas, [1720](#)
- sigaltstack
 - L4Re::Itas, [1720](#)
- Sigma0 API, [634](#)
 - l4sigma0_debug_dump, [636](#)
 - L4SIGMA0_IPCERROR, [635](#)
 - l4sigma0_map_anypage, [636](#)
 - l4sigma0_map_errstr, [636](#)
 - l4sigma0_map_iomem, [637](#)
 - l4sigma0_map_kip, [637](#)
 - l4sigma0_map_mem, [638](#)
 - L4SIGMA0_NOFPAGE, [635](#)
 - L4SIGMA0_NOTALIGNED, [635](#)
 - L4SIGMA0_OK, [635](#)
 - l4sigma0_return_flags_t, [635](#)
 - L4SIGMA0_SMALLERFPAGE, [635](#)
- Sigma0, the Root-Pager, [50](#)

- signal
 - L4Re::Parent, [1759](#)
- Signals, [626](#)
 - l4shmc_add_signal, [626](#)
 - l4shmc_attach_signal, [627](#)
 - l4shmc_check_magic, [627](#)
 - l4shmc_get_signal, [628](#)
 - l4shmc_signal_cap, [628](#)
- sigpending
 - L4Re::Itas, [1721](#)
- sigprocmask
 - L4Re::Itas, [1722](#)
- Single
 - L4::lpc::Snd_fpage, [1224](#)
- size
 - L4::Kip::Mem_desc, [1320](#)
 - L4Re::Dataspace, [1662](#)
 - L4Re::Video::Color_component, [1945](#)
 - L4virtio::Svr::Driver_mem_region_t< DATA >, [2235](#)
 - utrace::Ring_slot< ITEM_TYPE, GENERATION_PTR >, [2416](#)
- skip
 - L4::lpc::Istream, [1172](#)
 - L4virtio::Svr::Data_buffer, [2197](#)
- slab_size
 - cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >, [845](#)
 - cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >, [850](#)
- Small C++ Template Library, [640](#)
 - clamp, [641](#)
 - max, [641](#), [642](#)
 - min, [642](#), [643](#)
 - operator new, [643](#)
- Small_buf
 - L4::lpc::Small_buf, [1219](#)
- Smart_cap
 - L4::Smart_cap< T, SMART >, [1435](#)
- snd_base
 - L4::Cap_base, [1054](#)
- Snd_fpage
 - L4::lpc::Snd_fpage, [1224](#), [1225](#)
- Space_attrb
 - L4Re::Dma_space, [1673](#)
- Spaces and Mappings, [24](#)
- Special
 - L4::lpc::Gen_fpage, [1153](#)
- Split_ds
 - L4Re::Rm, [1771](#)
 - Rm, [2374](#)
- Src_dev_handle
 - L4vbus::lcu, [2036](#)
- Src_types
 - L4vbus::lcu, [2036](#)
- ss
 - l4_exc_regs_t, [1578](#)
- stack
 - L4Re::Video::View, [1975](#)
- start
 - L4::Kip::Mem_desc, [1321](#)
 - L4virtio::Svr::Request_processor, [2240](#), [2242](#)
- start_io
 - vmm.lua, [2476](#)
- start_request
 - L4virtio::Driver::Block_device, [2097](#)
- start_virtio_switch
 - vmm.lua, [2476](#)
- start_virtio_switch_tbl
 - vmm.lua, [2477](#)
- start_vm
 - vmm.lua, [2477](#)
- starts_with
 - cxx::String, [990](#)
- startup
 - L4::Uart, [1550](#)
 - L4::Uart_apb, [1556](#)
- State
 - L4vcpu::State, [2066](#)
- state
 - L4vcpu::Vcpu, [2078](#)
- state_transitions
 - L4virtio::Svr::Console::Port, [2176](#)
- stats_time
 - L4::Thread, [1458](#)
- status
 - L4::Lock_guard, [1346](#)
 - L4virtio::Svr::Dev_config, [2209](#)
- STB_GLOBAL
 - ELF binary format, [718](#)
- STB_HIOS
 - ELF binary format, [718](#)
- STB_HIPROC
 - ELF binary format, [718](#)
- STB_LOCAL
 - ELF binary format, [718](#)
- STB_LOOS
 - ELF binary format, [718](#)
- STB_LOPROC
 - ELF binary format, [718](#)
- STB_WEAK
 - ELF binary format, [718](#)
- Str_cp_in
 - L4::lpc::Str_cp_in< T >, [1231](#)
- str_cp_in
 - L4::lpc, [763](#)
- String
 - cxx::String, [987](#)
- Strong
 - L4Re::Namespace, [1749](#)
- STT_FILE
 - ELF binary format, [718](#)
- STT_FUNC
 - ELF binary format, [718](#)
- STT_HIOS
 - ELF binary format, [718](#)

- STT_HIPROC
 - ELF binary format, [718](#)
- STT_LOOS
 - ELF binary format, [718](#)
- STT_LOPROC
 - ELF binary format, [718](#)
- STT_NOTYPE
 - ELF binary format, [718](#)
- STT_OBJECT
 - ELF binary format, [718](#)
- STT_SECTION
 - ELF binary format, [718](#)
- sub_type
 - L4::Kip::Mem_desc, [1321](#)
- Super_pages
 - L4Re::Mem_alloc, [1734](#)
- supports
 - L4::Meta, [1352](#)
- Switch_factory, [2392](#)
 - op_create, [2396](#)
- switch_log
 - L4::Debugger, [1069](#)
- switch_to
 - L4::Thread, [1459](#)
- symlink
 - L4Re::Vfs::Directory, [1903](#)
- system_shutdown
 - L4::Platform_control, [1366](#)
- system_suspend
 - L4::Platform_control, [1367](#)
- tag
 - L4::lpc::Istream, [1172](#)
 - L4::lpc::Ostream, [1207](#)
 - L4::lpc::Varg, [1236](#)
- take
 - L4Re::Util::Counting_cap_alloc< COUNTER-
TYPE, Dbg >, [1838](#)
 - L4Re::Util::Dataspace_svr, [1845](#)
- take_reply_cap
 - L4::lpc_svr::Server_iface, [1277](#)
- Task, [369](#)
 - L4_FP_ALL_SPACES, [371](#)
 - L4_FP_DELETE_OBJ, [371](#)
 - L4_FP_OTHER_SPACES, [371](#)
 - I4_task_add_ku_mem, [371](#)
 - I4_task_cap_equal, [372](#)
 - I4_task_cap_valid, [373](#)
 - I4_task_delete_obj, [374](#)
 - I4_task_map, [375](#)
 - I4_task_release_cap, [377](#)
 - I4_task_unmap, [378](#)
 - I4_task_unmap_batch, [380](#)
 - I4_task_vgicc_map, [381](#)
 - I4_unmap_flags_t, [370](#)
- task
 - L4Re::Env, [1692](#)
 - L4vcpu::Vcpu, [2078](#)
- test
 - L4::Poll_timeout_kipclock, [1373](#)
 - test.mk - Test Application Role, [39](#)
 - The L4Re IPC Framework, [644](#)
 - Thread, [382](#)
 - I4_thread_arm_set_tpidruro, [386](#)
 - L4_THREAD_CONTROL_ALIEN, [385](#)
 - L4_THREAD_CONTROL_BIND_TASK, [385](#)
 - L4_thread_control_flags, [384](#)
 - L4_THREAD_CONTROL_MR_IDX_BIND_TASK, [385](#)
 - L4_THREAD_CONTROL_MR_IDX_BIND_UTCB, [385](#)
 - L4_THREAD_CONTROL_MR_IDX_EXC_HANDLER, [385](#)
 - L4_THREAD_CONTROL_MR_IDX_FLAG_VALS, [385](#)
 - L4_THREAD_CONTROL_MR_IDX_FLAGS, [385](#)
 - L4_THREAD_CONTROL_MR_IDX_PAGER, [385](#)
 - L4_thread_control_mr_indices, [385](#)
 - L4_THREAD_CONTROL_SET_EXC_HANDLER, [385](#)
 - L4_THREAD_CONTROL_SET_PAGER, [384](#)
 - I4_thread_ex_regs, [387](#)
 - L4_THREAD_EX_REGS_ARCH_MASK, [385](#)
 - L4_THREAD_EX_REGS_ARM64_SET_EL_EL0, [386](#)
 - L4_THREAD_EX_REGS_ARM64_SET_EL_EL1, [386](#)
 - L4_THREAD_EX_REGS_ARM64_SET_EL_KEEP, [386](#)
 - L4_THREAD_EX_REGS_ARM64_SET_EL_MASK, [386](#)
 - L4_THREAD_EX_REGS_ARM_SET_EL_EL0, [386](#)
 - L4_THREAD_EX_REGS_ARM_SET_EL_EL1, [386](#)
 - L4_THREAD_EX_REGS_ARM_SET_EL_KEEP, [386](#)
 - L4_THREAD_EX_REGS_ARM_SET_EL_MASK, [386](#)
 - L4_THREAD_EX_REGS_CANCEL, [385](#)
 - L4_thread_ex_regs_flags, [385](#)
 - L4_thread_ex_regs_flags_arm, [385](#)
 - L4_thread_ex_regs_flags_arm64, [386](#)
 - I4_thread_ex_regs_ret, [388](#)
 - I4_thread_ex_regs_ret_u, [389](#)
 - L4_THREAD_EX_REGS_TRIGGER_EXCEPTION, [385](#)
 - I4_thread_ex_regs_u, [390](#)
 - I4_thread_modify_sender_add, [392](#)
 - I4_thread_modify_sender_commit, [393](#)
 - I4_thread_modify_sender_start, [393](#)
 - I4_thread_register_del_irq, [394](#)
 - I4_thread_register_doorbell_irq, [395](#)
 - I4_thread_stats_time, [396](#)
 - I4_thread_switch, [397](#)
 - I4_thread_vcpu_control, [398](#)
 - I4_thread_vcpu_control_ext, [399](#)

- l4_thread_vcpu_control_ext_u, [400](#)
- l4_thread_vcpu_control_u, [401](#)
- l4_thread_vcpu_resume_commit, [402](#)
- l4_thread_vcpu_resume_start, [404](#)
- l4_thread_yield, [404](#)
- Thread control, [405](#)
 - l4_thread_control_alien, [406](#)
 - l4_thread_control_bind, [407](#)
 - l4_thread_control_commit, [408](#)
 - l4_thread_control_exc_handler, [409](#)
 - l4_thread_control_pager, [409](#)
 - l4_thread_control_start, [410](#)
- Thread Control Registers (TCRs), [275](#)
- Thread groups, [416](#)
 - l4_thread_group_add, [417](#)
 - l4_thread_group_remove, [417](#)
- thread.h
 - __L4UTIL_THREAD_FUNC, [2624](#)
- throw_error
 - L4Re, [798](#)
- throw_ipc_exception
 - IPC Helpers, [466](#)
- timed_out
 - L4::Poll_timeout_counter, [1370](#)
 - L4::Poll_timeout_kipclock, [1374](#)
- timeout
 - L4::lpc_svr::Timeout, [1281](#)
- timeout_expired
 - L4::lpc_svr::Timeout_queue, [1284](#)
- Timeouts, [236](#)
 - l4_ipc_timeout, [239](#)
 - L4_IPC_TIMEOUT_0, [238](#)
 - l4_rcv_timeout, [239](#)
 - l4_snd_timeout, [239](#)
 - l4_timeout, [240](#)
 - l4_timeout_abs, [240](#)
 - l4_timeout_get, [241](#)
 - l4_timeout_is_absolute, [242](#)
 - l4_timeout_rel, [243](#)
 - l4_timeout_rel_get, [243](#)
 - l4_timeout_s, [238](#)
 - l4_timeout_t, [238](#)
 - L4_TIMEOUT_US_MAX, [238](#)
 - l4_utcb_mr64_idx, [244](#)
- Timestamp Counter, [662](#)
 - l4_busy_wait_ns, [663](#)
 - l4_busy_wait_us, [664](#)
 - l4_calibrate_tsc, [665](#)
 - l4_get_hz, [665](#)
 - l4_ns_to_tsc, [665](#)
 - l4_rdpmc, [666](#)
 - l4_rdpmc_32, [666](#)
 - l4_rdtsc, [667](#)
 - l4_rdtsc_32, [667](#)
 - l4_tsc_init, [667](#)
 - l4_tsc_to_ns, [668](#)
 - l4_tsc_to_s_and_ns, [668](#)
 - l4_tsc_to_us, [669](#)
- To_default
 - L4Re::Namespace, [1748](#)
- To_device
 - L4Re::Dma_space, [1673](#)
- to_irq
 - L4vbus::Gpio_pin, [2032](#)
- To_non_blocking
 - L4Re::Namespace, [1748](#)
- total_objects
 - cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >, [846](#)
 - cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >, [851](#)
- total_size
 - L4virtio::Virtqueue, [2334](#), [2335](#)
- trigger
 - L4::Triggerable, [1479](#)
- trunc_order
 - L4, [755](#)
- Trusted
 - L4Re::Namespace, [1749](#)
- tx
 - L4virtio::Driver::Virtio_net_device, [2117](#)
- tx_queue_size
 - L4virtio::Driver::Virtio_net_device, [2118](#)
- Type
 - L4::lpc::Gen_fpage, [1152](#)
- type
 - L4::lpc::Varg, [1236](#)
 - L4::Kip::Mem_desc, [1322](#)
 - L4Re::Vfs::Be_file_system, [1898](#)
 - L4Re::Vfs::File_system, [1910](#)
- types.h
 - l4_capability_next, [3381](#)
 - l4_proto_t, [3380](#)
 - l4_ret_t, [3380](#)
- umalloc.h
 - umalloc_area_create, [3412](#)
 - umalloc_area_granularity, [3413](#)
- umalloc_area_create
 - umalloc.h, [3412](#)
- umalloc_area_granularity
 - umalloc.h, [3413](#)
- unbind
 - L4::lcu, [1120](#)
 - L4::lommu, [1132](#)
- Uncacheable
 - L4Re::Dataspace::F, [1664](#)
- Uncached
 - L4::lpc::Snd_fpage, [1223](#)
- Undefined
 - L4::Kip::Mem_desc, [1314](#)
- Unique_cap
 - L4Re, [778](#)
 - L4Re::Util, [803](#)
- Unique_del_cap
 - L4Re, [778](#)
 - L4Re::Util, [804](#)

- Unique_region
 - L4Re::Rm::Unique_region< T >, [1791](#), [1792](#)
 - Rm::Unique_region< T >, [2389](#), [2390](#)
- unlink
 - L4Re::Namespace, [1752](#)
 - L4Re::Vfs::Directory, [1903](#)
- unlock_all_locks
 - L4Re::Vfs::Be_file, [1895](#)
 - L4Re::Vfs::Generic_file, [1922](#)
- unmap
 - L4::Task, [1446](#)
 - L4Re::Dma_space, [1676](#)
- unmap_batch
 - L4::Task, [1447](#)
- unmask
 - L4::Irq, [1298](#)
 - L4::Irq_eoi, [1302](#)
- unregister_obj
 - L4::Registry_iface, [1385](#)
 - L4Re::Util::Object_registry, [1874](#)
- unregister_thread
 - L4Re::Itas, [1723](#)
- up
 - L4::Semaphore, [1410](#)
- used_align
 - L4virtio::Virtqueue, [2336](#)
- Used_elem
 - L4virtio::Virtqueue::Used_elem, [2348](#)
- used_size
 - L4virtio::Virtqueue, [2336](#)
- utcb_area
 - L4Re::Env, [1692](#), [1693](#)
- Utility Functions, [646](#)
 - l4_sleep, [649](#)
 - l4_touch_ro, [649](#)
 - l4_touch_rw, [650](#)
 - l4_usleep, [650](#)
 - l4util_micros2l4to, [651](#)
 - l4util_splitlog2_hdl, [651](#)
 - l4util_splitlog2_size, [652](#)
- utrace::Ring_buffer< SEQUENCE_TYPE >, [2396](#)
 - check_items, [2398](#)
 - check_mask, [2399](#)
- utrace::Ring_buffer_consumer< SEQUENCE_TYPE, ITEM_TYPE, GENERATION_PTR >, [2400](#)
 - dequeue, [2404](#)
 - peek, [2405](#)
 - Ring_buffer_consumer, [2404](#)
- utrace::Ring_buffer_consumer_raw< SEQUENCE_TYPE, ITEM_TYPE, GENERATION_PTR >, [2406](#)
 - Conservative, [2409](#)
 - Drop_policy, [2409](#)
 - Minimal, [2409](#)
 - peek, [2410](#)
 - Ring_buffer_consumer_raw, [2410](#)
- utrace::Ring_buffer_producer< SEQUENCE_TYPE, ITEM_TYPE, GENERATION_PTR >, [2411](#)
 - enqueue, [2414](#)
 - Ring_buffer_producer, [2413](#)
- utrace::Ring_slot< ITEM_TYPE, GENERATION_PTR >, [2415](#)
 - size, [2416](#)
- utrace::Ring_status< SEQUENCE_TYPE >, [2417](#)
 - alignment, [2419](#)
 - version, [2419](#)
- utrace::Tracebuffer, [2420](#)
 - dequeue, [2421](#)
 - index, [2421](#)
 - indexes, [2422](#)
 - validate, [2422](#)
- utrace::Tracebuffer::Index_desc, [2423](#)
- Uvmm, the virtual machine monitor, [72](#)
- uvmm_dtg The device tree generator for Uvmm, [89](#)
- V_flags
 - L4Re::Video::View, [1971](#)
- val
 - cxx::Bitfield< T, LSB, MSB >, [858](#)
- val_dirty
 - cxx::Bitfield< T, LSB, MSB >, [858](#)
- val_unshifted
 - cxx::Bitfield< T, LSB, MSB >, [859](#)
- valid
 - cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node, [899](#)
 - L4virtio::Svr::Virtqueue::Head_desc, [2320](#)
- validate
 - L4::Cap_base, [1055](#), [1056](#)
 - utrace::Tracebuffer, [2422](#)
- value
 - L4::lpc::Varg, [1236](#)
- Varg_list_ref
 - L4::lpc::Varg_list_ref, [1242](#)
- vbe_ctrl_info
 - l4util_l4mod_info, [1996](#)
- Vbus API, [603](#)
- vbus.h
 - L4VBUS_ICU_SRC_DEV_HANDLE, [3498](#)
 - l4vbus_icu_src_types, [3497](#)
 - L4VBUS_NULL, [3497](#)
 - L4VBUS_ROOT_BUS, [3497](#)
- vbus_interfaces.h
 - L4VBUS_IFACE_SHIFT, [3504](#)
 - l4vbus_iface_type_t, [3504](#)
 - L4VBUS_INTERFACE_BUS, [3505](#)
 - L4VBUS_INTERFACE_GENERIC, [3505](#)
 - L4VBUS_INTERFACE_GPIO, [3505](#)
 - L4VBUS_INTERFACE_ICU, [3505](#)
 - L4VBUS_INTERFACE_PCI, [3505](#)
 - L4VBUS_INTERFACE_PCIDEV, [3505](#)
 - L4VBUS_INTERFACE_PM, [3505](#)
 - l4vbus_subinterface_supported, [3505](#)
- vbus_types.h
 - L4VBUS_DEVICE_F_CHILDREN, [3511](#)
 - l4vbus_device_flags_t, [3511](#)
 - L4VBUS_RESOURCE_BUS, [3512](#)
 - L4VBUS_RESOURCE_DMA_DOMAIN, [3512](#)

- L4VBUS_RESOURCE_F_MEM_CACHEABLE, [3511](#)
- L4VBUS_RESOURCE_F_MEM_MMIO_READ, [3511](#)
- L4VBUS_RESOURCE_F_MEM_MMIO_WRITE, [3511](#)
- L4VBUS_RESOURCE_F_MEM_PREFETCHABLE, [3511](#)
- L4VBUS_RESOURCE_F_MEM_R, [3511](#)
- L4VBUS_RESOURCE_F_MEM_W, [3511](#)
- l4vbus_resource_flags_t, [3511](#)
- L4VBUS_RESOURCE_GPIO, [3512](#)
- L4VBUS_RESOURCE_INVALID, [3512](#)
- L4VBUS_RESOURCE_IRQ, [3512](#)
- L4VBUS_RESOURCE_MAX, [3512](#)
- L4VBUS_RESOURCE_MEM, [3512](#)
- L4VBUS_RESOURCE_PORT, [3512](#)
- l4vbus_resource_type_t, [3511](#)
- vcon.h
 - L4_vcon_read_flags, [3405](#)
 - L4_VCON_READ_SIZE_MASK, [3405](#)
 - L4_VCON_READ_STAT_BREAK, [3405](#)
 - L4_VCON_READ_STAT_DONE, [3405](#)
- vCPU API, [411](#)
 - L4_VCPU_F_EXCEPTIONS, [414](#)
 - L4_VCPU_F_FPU_ENABLED, [415](#)
 - L4_VCPU_F_IRQ, [414](#)
 - L4_VCPU_F_PAGE_FAULTS, [414](#)
 - L4_VCPU_F_USER_MODE, [414](#)
 - L4_VCPU_OFFSET_EXT_INFOS, [415](#), [416](#)
 - L4_VCPU_OFFSET_EXT_STATE, [415](#), [416](#)
 - L4_VCPU_SF_IRQ_PENDING, [416](#)
 - L4_vcpu_state_flags, [414](#)
 - L4_vcpu_state_offset, [415](#), [416](#)
 - L4_vcpu_sticky_flags, [416](#)
- vCPU Support Library, [725](#)
 - l4vcpu_irq_disable, [726](#)
 - l4vcpu_irq_disable_save, [727](#)
 - l4vcpu_irq_enable, [727](#)
 - l4vcpu_irq_restore, [728](#)
 - l4vcpu_is_irq_entry, [729](#)
 - l4vcpu_is_page_fault_entry, [730](#)
 - l4vcpu_print_state, [731](#)
 - l4vcpu_wait_for_event, [731](#)
- vcpu.h
 - l4_vcpu_check_version, [3517](#)
- vcpu_control
 - L4::Thread, [1460](#)
- vcpu_control_ext
 - L4::Thread, [1461](#)
- vcpu_resume_commit
 - L4::Thread, [1462](#)
- vcpu_resume_start
 - L4::Thread, [1463](#)
- version
 - l4_vcpu_state_t, [1607](#)
 - utrace::Ring_status< SEQUENCE_TYPE >, [2419](#)
- vicu
 - L4vbus::lcu, [2036](#)
- Video API, [572](#), [601](#)
 - F_l4re_video_goos_auto_refresh, [574](#)
 - F_l4re_video_goos_dynamic_buffers, [574](#)
 - F_l4re_video_goos_dynamic_views, [574](#)
 - F_l4re_video_goos_pointer, [574](#)
 - F_l4re_video_view_above, [574](#)
 - F_l4re_video_view_dyn_allocated, [574](#)
 - F_l4re_video_view_flags_mask, [574](#)
 - F_l4re_video_view_none, [574](#)
 - F_l4re_video_view_set_background, [574](#)
 - F_l4re_video_view_set_buffer, [574](#)
 - F_l4re_video_view_set_buffer_offset, [574](#)
 - F_l4re_video_view_set_bytes_per_line, [574](#)
 - F_l4re_video_view_set_flags, [574](#)
 - F_l4re_video_view_set_pixel, [574](#)
 - F_l4re_video_view_set_position, [574](#)
 - l4re_video_goos_create_buffer, [574](#)
 - l4re_video_goos_create_view, [575](#)
 - l4re_video_goos_delete_buffer, [575](#)
 - l4re_video_goos_delete_view, [576](#)
 - l4re_video_goos_get_static_buffer, [576](#)
 - l4re_video_goos_get_view, [576](#)
 - l4re_video_goos_info, [577](#)
 - l4re_video_goos_info_flags_t, [574](#)
 - l4re_video_goos_refresh, [577](#)
 - l4re_video_view_get_info, [578](#)
 - l4re_video_view_info_flags_t, [574](#)
 - l4re_video_view_refresh, [578](#)
 - l4re_video_view_set_info, [579](#)
 - l4re_video_view_set_viewport, [579](#)
 - l4re_video_view_stack, [580](#)
 - l4re_video_view_t, [573](#)
- view
 - L4Re::Video::Goos, [1955](#)
- view_info
 - L4Re::Util::Video::Goos_svr, [1890](#)
- Virtio Net P2P, a virtual network point-to-point link, [67](#)
- Virtio Net Switch, [724](#)
- Virtio Net Switch, a virtual network switch, [69](#)
- Virtio_con
 - L4virtio::Svr::Console::Virtio_con, [2182](#)
- Virtio_net, [2423](#)
 - notify_queue, [2428](#)
- Virtio_net_request, [2428](#)
 - drop_requests, [2429](#)
 - get_request, [2430](#)
- Virtio_switch, [2431](#)
 - add_monitor_port, [2433](#)
 - add_port, [2433](#)
 - check_ports, [2434](#)
 - handle_l4virtio_port_tx, [2434](#)
 - port_available, [2435](#)
 - Virtio_switch, [2433](#)
- Virtio_vlan_mangle, [2436](#)
 - add, [2437](#)
 - copy_pkt, [2437](#)
 - remove, [2438](#)

- rewrite_hdr, [2439](#)
- Virtio_vlan_mangle, [2437](#)
- Virtual Console, [418](#)
 - l4_vcon_attr_t, [420](#)
 - L4_VCON_ECHO, [421](#)
 - l4_vcon_get_attr, [422](#)
 - l4_vcon_get_attr_u, [422](#)
 - L4_vcon_i_flags, [421](#)
 - L4_VCON_ICANON, [421](#)
 - L4_VCON_ICRNL, [421](#)
 - L4_VCON_IGNCR, [421](#)
 - L4_VCON_INLCR, [421](#)
 - L4_vcon_l_flags, [421](#)
 - L4_vcon_o_flags, [421](#)
 - L4_VCON_OCRNL, [421](#)
 - L4_VCON_ONLCR, [421](#)
 - L4_VCON_ONLRET, [421](#)
 - l4_vcon_read, [423](#)
 - L4_VCON_READ_SIZE, [422](#)
 - l4_vcon_read_u, [424](#)
 - l4_vcon_read_with_flags, [425](#)
 - l4_vcon_send, [426](#)
 - l4_vcon_send_u, [427](#)
 - l4_vcon_set_attr, [429](#)
 - l4_vcon_set_attr_raw, [429](#)
 - l4_vcon_set_attr_u, [430](#)
 - L4_vcon_size_consts, [421](#)
 - l4_vcon_write, [431](#)
 - L4_VCON_WRITE_SIZE, [422](#)
 - l4_vcon_write_u, [432](#)
- Virtual Machines, [300](#)
- Virtual Registers (UTCBs), [264](#)
 - l4_utcb_br, [267](#)
 - l4_utcb_mr, [267](#)
 - l4_utcb_t, [266](#)
 - l4_utcb_tcr, [268](#)
- VM API for SVM, [300](#)
- VM API for TZ, [321](#)
- VM API for VMX, [301](#)
 - L4_VM_VMX_BASIC_REG, [306](#)
 - L4_vm_vmx_caps_regs, [306](#)
 - l4_vm_vmx_clear, [308](#)
 - L4_VM_VMX_CR0_FIXED0_REG, [306](#)
 - L4_VM_VMX_CR0_FIXED1_REG, [306](#)
 - L4_VM_VMX_CR4_FIXED0_REG, [306](#)
 - L4_VM_VMX_CR4_FIXED1_REG, [306](#)
 - L4_vm_vmx_dfl1_regs, [306](#)
 - L4_VM_VMX_ENTRY_CTL5_DFL1_REG, [306](#)
 - L4_VM_VMX_EPT_VPID_CAP_REG, [306](#)
 - L4_VM_VMX_EXIT_CTL5_DFL1_REG, [306](#)
 - l4_vm_vmx_field_len, [308](#)
 - l4_vm_vmx_field_order, [309](#)
 - l4_vm_vmx_get_caps, [310](#)
 - l4_vm_vmx_get_caps_default1, [310](#)
 - l4_vm_vmx_get_cr2_index, [311](#)
 - l4_vm_vmx_get_hw_vmcs, [311](#)
 - L4_VM_VMX_MISC_REG, [306](#)
 - L4_VM_VMX_NESTED_REVISION, [306](#)
 - L4_VM_VMX_NUM_CAPS_REGS, [306](#)
 - L4_VM_VMX_NUM_DFL1_REGS, [306](#)
 - L4_VM_VMX_PINBASED_CTL5_DFL1_REG, [306](#)
 - L4_VM_VMX_PROCBASED_CTL52_REG, [306](#)
 - L4_VM_VMX_PROCBASED_CTL5_DFL1_REG, [306](#)
 - l4_vm_vmx_ptr_load, [312](#)
 - l4_vm_vmx_read, [313](#)
 - l4_vm_vmx_read_16, [314](#)
 - l4_vm_vmx_read_32, [314](#)
 - l4_vm_vmx_read_64, [315](#)
 - l4_vm_vmx_read_nat, [315](#)
 - l4_vm_vmx_set_hw_vmcs, [316](#)
 - L4_vm_vmx_sw_fields, [306](#)
 - L4_VM_VMX_TRUE_ENTRY_CTL5_REG, [306](#)
 - L4_VM_VMX_TRUE_EXIT_CTL5_REG, [306](#)
 - L4_VM_VMX_TRUE_PINBASED_CTL5_REG, [306](#)
 - L4_VM_VMX_TRUE_PROCBASED_CTL5_REG, [306](#)
 - l4_vm_vmx_vcpu_state_t, [303](#)
 - l4_vm_vmx_vcpu_vmcs_t, [304](#)
 - L4_VM_VMX_VMCS_CR2, [307](#)
 - L4_VM_VMX_VMCS_ENUM_REG, [306](#)
 - L4_VM_VMX_VMCS_MSR_CSTAR, [307](#)
 - L4_VM_VMX_VMCS_MSR_KERNEL_GS_BASE, [307](#)
 - L4_VM_VMX_VMCS_MSR_LSTAR, [307](#)
 - L4_VM_VMX_VMCS_MSR_STAR, [307](#)
 - L4_VM_VMX_VMCS_MSR_SYSCALL_MASK, [307](#)
 - L4_VM_VMX_VMCS_MSR_TSC_AUX, [307](#)
 - L4_VM_VMX_VMCS_NAT_ARG0, [307](#)
 - L4_VM_VMX_VMCS_NAT_ARG1, [307](#)
 - L4_VM_VMX_VMCS_NAT_ARG2, [307](#)
 - L4_VM_VMX_VMCS_NAT_ARG3, [307](#)
 - L4_VM_VMX_VMCS_SIZE_DIRTY_BITMAP, [307](#)
 - L4_VM_VMX_VMCS_SIZE_VALUES, [307](#)
 - L4_vm_vmx_vmcs_sizes, [307](#)
 - L4_VM_VMX_VMCS_XCR0, [307](#)
 - l4_vm_vmx_write, [317](#)
 - l4_vm_vmx_write_16, [318](#)
 - l4_vm_vmx_write_32, [319](#)
 - l4_vm_vmx_write_64, [319](#)
 - l4_vm_vmx_write_nat, [320](#)
 - l4_vm_vmx_offset_table_t, [304](#)
- vmm.lua
 - new_sched, [2475](#)
 - set_sched, [2475](#)
 - start_io, [2476](#)
 - start_virtio_switch, [2476](#)
 - start_virtio_switch_tbl, [2477](#)
 - start_vm, [2477](#)
- W
 - L4Re::Dataspace::F, [1664](#)
 - L4Re::Rm::F, [1788](#)
 - Rm::F, [2385](#)
- W_bits

- cxx::Bitmap_base, [871](#)
 - L4Re::Util::Bitmap_base, [1808](#)
- wait
 - L4::lpc::Istream, [1173](#), [1174](#)
 - L4::Irq, [1299](#)
 - L4virtio::Driver::Device, [2109](#)
- wait_for_event
 - L4vcpu::Vcpu, [2079](#)
- wait_for_next_used
 - L4virtio::Driver::Device, [2110](#)
- wait_rx
 - L4virtio::Driver::Virtio_net_device, [2119](#)
- Wd_16bit
 - L4Re::Mmio_space, [1742](#)
- Wd_32bit
 - L4Re::Mmio_space, [1742](#)
- Wd_64bit
 - L4Re::Mmio_space, [1742](#)
- Wd_8bit
 - L4Re::Mmio_space, [1742](#)
- Word_bytes
 - L4::lpc::Msg, [766](#)
- word_index
 - cxx::Bitmap_base, [879](#)
 - L4Re::Util::Bitmap_base, [1812](#)
- write
 - L4::Uart, [1550](#)
 - L4::Uart_apb, [1557](#)
 - L4::Vcon, [1569](#)
 - L4drivers::Register_tmpl< BITS, BLOCK >, [1633](#)
- write_bfm_t
 - L4virtio::Virtqueue::Desc::Flags, [2344](#)
- write_now
 - cxx, [740](#)
- writew
 - L4Re::Vfs::Regular_file, [1937](#)
- X
 - L4Re::Dataspace::F, [1664](#)
 - L4Re::Rm::F, [1788](#)
 - Rm::F, [2385](#)
- x86 Virtual Registers (UTCB), [277](#)
- x86/l4/sys/__kip-arch.h, [2567](#)
- x86/l4/sys/__vcpu-arch.h, [2578](#), [2579](#)
- x86/l4/sys/arch/cache.h, [3145](#)
- x86/l4/sys/arch/consts.h, [3166](#)
- x86/l4/sys/arch/ipc.h, [3277](#), [3278](#)
- x86/l4/sys/arch/kip.h, [3464](#)
- x86/l4/sys/arch/l4int.h, [3319](#), [3320](#)
- x86/l4/sys/arch/platform_control.h, [3338](#)
- x86/l4/sys/arch/task.h, [3368](#)
- x86/l4/sys/arch/thread.h, [2625](#)
- x86/l4/sys/arch/utcb.h, [3397](#), [3399](#)
- x86/l4/sys/ktrace_events.h, [2595](#)
- x86/l4/sys/linkage.h, [2601](#)
- x86/l4/sys/segment.h, [2532](#), [2533](#)
- x86/l4/sys/vm.h, [2608](#)
- x86/l4/util/arch/l4_macros.h, [2610](#), [2611](#)
- x86/l4/util/arch/thread.h, [2625](#)
- x86/l4/util/bitops_arch.h, [2630](#)
- x86/l4/util/cpu.h, [2636](#), [2637](#)
- x86/l4/util/irq.h, [2783](#), [2784](#)
- x86/l4/util/mbi_argv.h, [2641](#), [2642](#)
- x86/l4/util/perform.h, [2540](#), [2541](#)
- x86/l4/util/port_io.h, [2550](#), [2552](#)
- x86/l4/util/rdtsc.h, [2558](#), [2560](#)
- x86/l4/util/spin.h, [2564](#), [2565](#)