

L4Re – L4 Runtime Environment

Generated by Doxygen 1.8.13

Contents

1	Overview	1
2	Introduction	3
2.1	Fiasco.OC	3
2.1.1	Communication	3
2.1.2	Kernel Objects	4
2.2	L4Re System Structure	4
2.3	L4 Runtime Environment (L4Re)	6
3	Tutorial	7
3.1	Customizations	8
4	Programming for L4Re	9
4.1	Capabilities and Naming	9
4.2	Initial Environment and Application Bootstrapping	10
4.2.1	Configuring an application before startup	11
4.2.2	Connecting clients and servers	11
4.3	Memory management - Data Spaces and the Region Map	12
4.3.1	User-level paging	12
4.3.1.1	Data spaces	12
4.3.1.2	Virtual Memory Handling	12
4.3.1.3	Memory Allocation	13
4.4	Program Input and Output	13
4.5	Initial Memory Allocator and Factory	13
4.6	Application and Server Building Blocks	13

4.6.1	Creating Additional Application Threads	14
4.6.2	Providing a Service	14
4.7	Pthread Support	14
4.8	Interface Definition Language	15
4.8.1	Parameter types for RPC	15
4.8.2	RPC Return Types	17
4.8.3	RPC Method Declaration	18
4.9	L4Re Build System	18
4.9.1	Building L4Re	19
4.9.2	Writing BID Make Files	20
4.9.3	prog.mk - Application Role	20
4.9.4	include.mk - Header File Role	23
4.9.5	test.mk - Test Application Role	24
5	L4Re Servers	29
5.1	Moe, the Root-Task	31
5.1.1	Memory Allocator, Generic Factory	31
5.1.2	Name-Space Provider	32
5.1.2.1	Boot FS	32
5.1.3	Log Subsystem	32
5.1.4	Scheduler Subsystem	32
5.1.5	Command-Line Options	32
5.2	Ned, the Init Process	33
5.2.1	Lua Bindings for L4Re	33
5.2.1.1	Capabilities in Lua	33
5.2.1.2	Access to L4Re::Env Capabilities	33
5.2.1.3	Constants	34
5.2.1.4	Application Startup Details	35
5.2.2	Command Line Options	35
5.3	Io, the Io Server	36
5.4	Uvmm, the virtual machine monitor	41
5.5	Mag, the GUI Multiplexer	41
5.6	Cons, the Console Multiplexer	43

6	Deprecated List	45
7	Module Index	47
7.1	Modules	47
8	Namespace Index	51
8.1	Namespace List	51
9	Hierarchical Index	53
9.1	Class Hierarchy	53
10	Data Structure Index	63
10.1	Data Structures	63
11	File Index	77
11.1	File List	77
12	Module Documentation	89
12.1	ARM Virtual Registers (UTCB)	89
12.1.1	Detailed Description	89
12.2	Atomic Instructions	90
12.2.1	Detailed Description	92
12.2.2	Function Documentation	92
12.2.2.1	l4util_add8()	92
12.2.2.2	l4util_add8_res()	92
12.2.2.3	l4util_atomic_add()	92
12.2.2.4	l4util_atomic_inc()	93
12.2.2.5	l4util_cmpxchg()	93
12.2.2.6	l4util_cmpxchg16()	94
12.2.2.7	l4util_cmpxchg32()	95
12.2.2.8	l4util_cmpxchg64()	96
12.2.2.9	l4util_cmpxchg8()	96
12.2.2.10	l4util_inc8()	97
12.2.2.11	l4util_inc8_res()	97

12.2.2.12	<code>l4util_xchg()</code>	98
12.2.2.13	<code>l4util_xchg16()</code>	98
12.2.2.14	<code>l4util_xchg32()</code>	98
12.2.2.15	<code>l4util_xchg8()</code>	99
12.3	Auxiliary data	100
12.3.1	Detailed Description	100
12.4	Base API	101
12.4.1	Detailed Description	103
12.5	Basic Macros	104
12.5.1	Detailed Description	105
12.5.2	Macro Definition Documentation	105
12.5.2.1	<code>L4_ALWAYS_INLINE</code>	105
12.5.2.2	<code>L4_HIDDEN</code>	105
12.5.2.3	<code>L4_NOTHROW</code>	106
12.6	Bit Manipulation	107
12.6.1	Detailed Description	107
12.6.2	Function Documentation	107
12.6.2.1	<code>l4util_bsf()</code>	108
12.6.2.2	<code>l4util_bsr()</code>	108
12.6.2.3	<code>l4util_btc()</code>	109
12.6.2.4	<code>l4util_btr()</code>	109
12.6.2.5	<code>l4util_bts()</code>	110
12.6.2.6	<code>l4util_clear_bit()</code>	111
12.6.2.7	<code>l4util_complement_bit()</code>	111
12.6.2.8	<code>l4util_find_first_set_bit()</code>	112
12.6.2.9	<code>l4util_find_first_zero_bit()</code>	112
12.6.2.10	<code>l4util_next_power2()</code>	113
12.6.2.11	<code>l4util_set_bit()</code>	113
12.6.2.12	<code>l4util_test_bit()</code>	114
12.7	Bitmap graphics and fonts	115

12.7.1 Detailed Description	115
12.8 Buffer Registers (BRs)	116
12.8.1 Detailed Description	116
12.8.2 Enumeration Type Documentation	116
12.8.2.1 l4_buffer_desc_consts_t	116
12.9 C++ Exceptions	118
12.9.1 Detailed Description	118
12.10 C++ IPC Interface Definition.	119
12.10.1 Detailed Description	119
12.11 CPU related functions	120
12.11.1 Detailed Description	120
12.11.2 Function Documentation	120
12.11.2.1 l4util_cpu_capabilities()	120
12.11.2.2 l4util_cpu_capabilities_nocheck()	121
12.11.2.3 l4util_cpu_has_cpuid()	122
12.12 Cache Consistency	123
12.12.1 Detailed Description	123
12.12.2 Function Documentation	123
12.12.2.1 l4_cache_clean_data()	123
12.12.2.2 l4_cache_coherent()	124
12.12.2.3 l4_cache_dma_coherent()	124
12.12.2.4 l4_cache_flush_data()	125
12.12.2.5 l4_cache_inv_data()	125
12.13 Capabilities	127
12.13.1 Detailed Description	127
12.13.2 Enumeration Type Documentation	128
12.13.2.1 l4_cap_consts_t	128
12.13.2.2 l4_default_caps_t	128
12.13.3 Function Documentation	129
12.13.3.1 l4_capability_equal()	129

12.13.3.2 <code>l4_is_invalid_cap()</code>	129
12.13.3.3 <code>l4_is_valid_cap()</code>	130
12.14 Capability allocator	131
12.14.1 Detailed Description	131
12.14.2 Function Documentation	131
12.14.2.1 <code>l4re_util_cap_last()</code>	131
12.15 Chunks	132
12.15.1 Detailed Description	132
12.15.2 Function Documentation	132
12.15.2.1 <code>l4shmc_add_chunk()</code>	132
12.15.2.2 <code>l4shmc_chunk_capacity()</code>	133
12.15.2.3 <code>l4shmc_chunk_ptr()</code>	133
12.15.2.4 <code>l4shmc_chunk_signal()</code>	134
12.15.2.5 <code>l4shmc_get_chunk()</code>	134
12.15.2.6 <code>l4shmc_get_chunk_to()</code>	135
12.15.2.7 <code>l4shmc_iterate_chunk()</code>	135
12.16 Comfortable Command Line Parsing	136
12.16.1 Detailed Description	136
12.16.2 Function Documentation	136
12.16.2.1 <code>parse_cmdline()</code>	137
12.17 Console API	139
12.17.1 Detailed Description	139
12.18 Consumer	140
12.18.1 Detailed Description	140
12.18.2 Function Documentation	140
12.18.2.1 <code>l4shmc_chunk_consumed()</code>	140
12.18.2.2 <code>l4shmc_chunk_size()</code>	141
12.18.2.3 <code>l4shmc_enable_chunk()</code>	141
12.18.2.4 <code>l4shmc_is_chunk_ready()</code>	142
12.18.2.5 <code>l4shmc_wait_chunk()</code>	142

12.18.2.6 l4shmc_wait_chunk_to()	142
12.18.2.7 l4shmc_wait_chunk_try()	143
12.19 Consumer	144
12.19.1 Detailed Description	144
12.19.2 Function Documentation	144
12.19.2.1 l4shmc_enable_signal()	144
12.19.2.2 l4shmc_wait_any()	145
12.19.2.3 l4shmc_wait_any_to()	145
12.19.2.4 l4shmc_wait_any_try()	146
12.19.2.5 l4shmc_wait_signal()	146
12.19.2.6 l4shmc_wait_signal_to()	146
12.19.2.7 l4shmc_wait_signal_try()	147
12.20 DMA API for L4Re	148
12.21 DMA Space Interface	149
12.21.1 Detailed Description	149
12.21.2 Typedef Documentation	149
12.21.2.1 l4re_dma_space_t	150
12.21.3 Function Documentation	150
12.21.3.1 l4re_dma_space_associate()	150
12.21.3.2 l4re_dma_space_disassociate()	151
12.21.3.3 l4re_dma_space_map()	151
12.21.3.4 l4re_dma_space_unmap()	152
12.22 Dataspace interface	153
12.22.1 Detailed Description	154
12.22.2 Enumeration Type Documentation	154
12.22.2.1 l4re_ds_map_flags	154
12.22.3 Function Documentation	154
12.22.3.1 l4re_ds_allocate()	154
12.22.3.2 l4re_ds_clear()	155
12.22.3.3 l4re_ds_copy_in()	155

12.22.3.4 l4re_ds_flags()	155
12.22.3.5 l4re_ds_info()	156
12.22.3.6 l4re_ds_phys()	156
12.22.3.7 l4re_ds_size()	156
12.23 Debug interface	158
12.23.1 Detailed Description	158
12.23.2 Function Documentation	158
12.23.2.1 l4re_debug_obj_debug()	158
12.24 Debugging API	159
12.24.1 Detailed Description	159
12.25 EDID parsing functionality	160
12.25.1 Detailed Description	160
12.25.2 Enumeration Type Documentation	160
12.25.2.1 Libedid_consts	160
12.25.3 Function Documentation	161
12.25.3.1 libedid_check_header()	161
12.25.3.2 libedid_checksum()	161
12.25.3.3 libedid_dump()	161
12.25.3.4 libedid_dump_standard_timings()	162
12.25.3.5 libedid_num_ext_blocks()	162
12.25.3.6 libedid_pnp_id()	162
12.25.3.7 libedid_prefered_resolution()	163
12.25.3.8 libedid_revision()	163
12.25.3.9 libedid_version()	163
12.26 ELF binary format	165
12.26.1 Detailed Description	177
12.26.2 Macro Definition Documentation	177
12.26.2.1 DF_1_CONFALT	177
12.26.2.2 DF_1_DIRECT	178
12.26.2.3 DF_1_DISPRELDNE	178

12.26.2.4 DF_1_DISPRLPND	178
12.26.2.5 DF_1_ENDFILTEE	178
12.26.2.6 DF_1_GLOBAL	178
12.26.2.7 DF_1_GROUP	179
12.26.2.8 DF_1_INTERPOSE	179
12.26.2.9 DF_1_LOADFLTR	179
12.26.2.10 DF_1_NODEFLIB	179
12.26.2.11 DF_1_NODELETE	179
12.26.2.12 DF_1_NODUMP	180
12.26.2.13 DF_1_NOOPEN	180
12.26.2.14 DF_1_NOW	180
12.26.2.15 DF_1_ORIGIN	180
12.26.2.16 DF_P1_GROUPPERM	180
12.26.2.17 DF_P1_LAZYLOAD	181
12.26.2.18 DT_HIPROC	181
12.26.2.19 DT_LOPROC	181
12.26.2.20 DT_NULL	181
12.26.2.21 EI_CLASS [1/2]	181
12.26.2.22 EI_CLASS [2/2]	182
12.26.2.23 EI_DATA [1/2]	182
12.26.2.24 EI_DATA [2/2]	182
12.26.2.25 EI_OSABI [1/2]	182
12.26.2.26 EI_OSABI [2/2]	183
12.26.2.27 EI_PAD [1/2]	183
12.26.2.28 EI_PAD [2/2]	183
12.26.2.29 EI_VERSION [1/2]	183
12.26.2.30 EI_VERSION [2/2]	184
12.26.2.31 ELFCLASSNONE [1/2]	184
12.26.2.32 ELFCLASSNONE [2/2]	184
12.26.2.33 ELFDATA2LSB [1/2]	184

12.26.2.34	ELFDATA2LSB [2/2]	185
12.26.2.35	ELFDATA2MSB [1/2]	185
12.26.2.36	ELFDATA2MSB [2/2]	185
12.26.2.37	ELFDATANONE [1/2]	185
12.26.2.38	ELFDATANONE [2/2]	186
12.26.2.39	ELFOSABI_AIX	186
12.26.2.40	ELFOSABI_FREEBSD	186
12.26.2.41	ELFOSABI_HPUX [1/2]	186
12.26.2.42	ELFOSABI_HPUX [2/2]	186
12.26.2.43	ELFOSABI_IRIX	187
12.26.2.44	ELFOSABI_LINUX	187
12.26.2.45	ELFOSABI_MODESTO	187
12.26.2.46	ELFOSABI_NETBSD	187
12.26.2.47	ELFOSABI_OPENBSD	187
12.26.2.48	ELFOSABI_SOLARIS	188
12.26.2.49	ELFOSABI_SYSV [1/2]	188
12.26.2.50	ELFOSABI_SYSV [2/2]	188
12.26.2.51	ELFOSABI_TRU64	188
12.26.2.52	EM_ARC	188
12.26.2.53	NT_VERSION	189
12.26.2.54	PT_GNU_EH_FRAME	189
12.26.2.55	PT_GNU_RELRO	189
12.26.2.56	PT_GNU_STACK	189
12.26.2.57	PT_HIOS	189
12.26.2.58	PT_HIPROC	190
12.26.2.59	PT_L4_AUX	190
12.26.2.60	PT_L4_KIP	190
12.26.2.61	PT_L4_STACK	190
12.26.2.62	PT_LOOS	190
12.26.2.63	PT_LOPROC	191

12.26.2.64 SHF_GROUP	191
12.26.2.65 SHF_MASKOS	191
12.26.2.66 SHF_TLS	191
12.26.2.67 SHT_NUM	191
12.27 Error Handling	192
12.27.1 Detailed Description	192
12.27.2 Enumeration Type Documentation	193
12.27.2.1 l4_ipc_tcr_error_t	193
12.27.3 Function Documentation	193
12.27.3.1 l4_error()	193
12.27.3.2 l4_ipc_error()	195
12.27.3.3 l4_ipc_error_code()	196
12.27.3.4 l4_ipc_is_rcv_error()	196
12.27.3.5 l4_ipc_is_snd_error()	196
12.28 Error codes	198
12.28.1 Detailed Description	198
12.28.2 Enumeration Type Documentation	198
12.28.2.1 l4_error_code_t	198
12.29 Event API	200
12.29.1 Detailed Description	200
12.30 Event interface	201
12.30.1 Detailed Description	201
12.30.2 Function Documentation	201
12.30.2.1 l4re_event_get_axis_info()	201
12.30.2.2 l4re_event_get_buffer()	202
12.30.2.3 l4re_event_get_num_streams()	202
12.30.2.4 l4re_event_get_stream_info()	203
12.30.2.5 l4re_event_get_stream_info_for_id()	203
12.31 Exception registers	205
12.31.1 Detailed Description	205

12.31.2 Function Documentation	205
12.31.2.1 l4_utcb_exc()	205
12.31.2.2 l4_utcb_exc_is_ex_regs_exception()	206
12.31.2.3 l4_utcb_exc_is_pf()	206
12.31.2.4 l4_utcb_exc_pc()	206
12.31.2.5 l4_utcb_exc_pc_set()	207
12.32 Extended vCPU support	208
12.32.1 Detailed Description	208
12.32.2 Function Documentation	208
12.32.2.1 l4vcpu_ext_alloc()	208
12.33 Factory	210
12.33.1 Detailed Description	211
12.33.2 Function Documentation	211
12.33.2.1 l4_factory_create_factory()	211
12.33.2.2 l4_factory_create_factory_u()	212
12.33.2.3 l4_factory_create_gate()	213
12.33.2.4 l4_factory_create_gate_u()	214
12.33.2.5 l4_factory_create_irq()	215
12.33.2.6 l4_factory_create_irq_u()	216
12.33.2.7 l4_factory_create_task()	217
12.33.2.8 l4_factory_create_task_u()	218
12.33.2.9 l4_factory_create_thread()	219
12.33.2.10 l4_factory_create_thread_u()	220
12.33.2.11 l4_factory_create_vm()	221
12.33.2.12 l4_factory_create_vm_u()	222
12.34 Fiasco extensions	223
12.34.1 Detailed Description	224
12.34.2 Enumeration Type Documentation	224
12.34.2.1 anonymous enum	224
12.34.3 Function Documentation	225

12.34.3.1 fiasco_gdt_get_entry_offset()	225
12.34.3.2 fiasco_gdt_set()	225
12.34.3.3 fiasco_ldt_set()	226
12.34.3.4 fiasco_tbuf_get_status()	226
12.34.3.5 fiasco_tbuf_get_status_phys()	226
12.34.3.6 fiasco_tbuf_log()	227
12.34.3.7 fiasco_tbuf_log_3val()	227
12.34.3.8 fiasco_tbuf_log_binary()	228
12.35 Fiasco real time scheduling extensions	229
12.36 Fiasco-UX Virtual devices	230
12.36.1 Detailed Description	230
12.36.2 Enumeration Type Documentation	230
12.36.2.1 l4_vhw_entry_type	230
12.37 Flex pages	232
12.37.1 Detailed Description	234
12.37.2 Enumeration Type Documentation	234
12.37.2.1 anonymous enum	234
12.37.2.2 anonymous enum	234
12.37.2.3 L4_cap_fpage_rights	235
12.37.2.4 l4_fpage_cacheability_opt_t	236
12.37.2.5 l4_fpage_consts	236
12.37.2.6 L4_fpage_rights	237
12.37.2.7 L4_obj_fpage_ctl	237
12.37.3 Function Documentation	238
12.37.3.1 l4_fpage()	238
12.37.3.2 l4_fpage_all()	238
12.37.3.3 l4_fpage_contains()	239
12.37.3.4 l4_fpage_invalid()	240
12.37.3.5 l4_fpage_ioport()	240
12.37.3.6 l4_fpage_max_order()	240

12.37.3.7 l4_fpage_memaddr()	241
12.37.3.8 l4_fpage_obj()	242
12.37.3.9 l4_fpage_page()	243
12.37.3.10 l4_fpage_rights()	243
12.37.3.11 l4_fpage_set_rights()	244
12.37.3.12 l4_fpage_size()	244
12.37.3.13 l4_fpage_type()	245
12.37.3.14 l4_iofpage()	245
12.37.3.15 l4_is_fpage_writable()	246
12.37.3.16 l4_obj_fpage()	247
12.38 Functions for rendering bitmap data in frame buffers	248
12.38.1 Detailed Description	248
12.38.2 Typedef Documentation	248
12.38.2.1 gfbxbitmap_color_pix_t	249
12.38.2.2 gfbxbitmap_color_t	249
12.38.3 Function Documentation	249
12.38.3.1 gfbxbitmap_bmap()	249
12.38.3.2 gfbxbitmap_convert_color()	250
12.38.3.3 gfbxbitmap_copy()	250
12.38.3.4 gfbxbitmap_fill()	250
12.38.3.5 gfbxbitmap_set()	251
12.39 Functions for rendering bitmap fonts to frame buffers	252
12.39.1 Detailed Description	253
12.39.2 Enumeration Type Documentation	253
12.39.2.1 anonymous enum	253
12.39.3 Function Documentation	253
12.39.3.1 gfbxbitmap_font_data()	253
12.39.3.2 gfbxbitmap_font_get()	253
12.39.3.3 gfbxbitmap_font_height()	254
12.39.3.4 gfbxbitmap_font_init()	254

12.39.3.5 gfbitmap_font_text()	254
12.39.3.6 gfbitmap_font_text_scale()	255
12.39.3.7 gfbitmap_font_width()	256
12.40 Functions to manipulate the local IDT	257
12.40.1 Detailed Description	257
12.41 IA32 Port I/O API	258
12.41.1 Detailed Description	258
12.41.2 Function Documentation	258
12.41.2.1 l4util_in16()	259
12.41.2.2 l4util_in32()	260
12.41.2.3 l4util_in8()	260
12.41.2.4 l4util_ins16()	261
12.41.2.5 l4util_ins32()	261
12.41.2.6 l4util_ins8()	262
12.41.2.7 l4util_out16()	262
12.41.2.8 l4util_out32()	262
12.41.2.9 l4util_out8()	263
12.41.2.10 l4util_outs16()	263
12.41.2.11 l4util_outs32()	264
12.41.2.12 l4util_outs8()	264
12.42 IO interface	265
12.42.1 Detailed Description	266
12.42.2 Typedef Documentation	266
12.42.2.1 l4io_resource_t	266
12.42.3 Enumeration Type Documentation	266
12.42.3.1 l4io_device_types_t	266
12.42.3.2 l4io_iomem_flags_t	266
12.42.3.3 l4io_resource_types_t	267
12.42.4 Function Documentation	267
12.42.4.1 l4io_has_resource()	267

12.42.4.2 l4io_lookup_device()	268
12.42.4.3 l4io_lookup_resource()	268
12.42.4.4 l4io_release_iomem()	269
12.42.4.5 l4io_release_ioport()	269
12.42.4.6 l4io_request_iomem()	270
12.42.4.7 l4io_request_iomem_region()	270
12.42.4.8 l4io_request_ioport()	271
12.42.4.9 l4io_request_resource_iomem()	271
12.42.4.10 l4io_search_iomem_region()	272
12.43 IPC-Gate API	273
12.43.1 Detailed Description	273
12.43.2 Function Documentation	274
12.43.2.1 l4_ipc_gate_bind_thread()	274
12.43.2.2 l4_ipc_gate_get_infos()	274
12.44 IRQ handling library	276
12.44.1 Detailed Description	276
12.45 IRQs	277
12.45.1 Detailed Description	278
12.45.2 Enumeration Type Documentation	278
12.45.2.1 L4_irq_mode	278
12.45.3 Function Documentation	279
12.45.3.1 l4_irq_attach()	279
12.45.3.2 l4_irq_attach_u()	280
12.45.3.3 l4_irq_detach()	280
12.45.3.4 l4_irq_detach_u()	281
12.45.3.5 l4_irq_mux_chain()	282
12.45.3.6 l4_irq_mux_chain_u()	283
12.45.3.7 l4_irq_receive()	284
12.45.3.8 l4_irq_receive_u()	285
12.45.3.9 l4_irq_trigger()	286

12.45.3.104_irq_trigger_u()	287
12.45.3.114_irq_unmask()	288
12.45.3.124_irq_unmask_u()	289
12.45.3.134_irq_wait()	289
12.45.3.144_irq_wait_u()	290
12.46Initial Environment	292
12.46.1 Detailed Description	292
12.46.2 Function Documentation	293
12.46.2.1 l4re_env()	293
12.46.2.2 l4re_env_get_cap()	293
12.46.2.3 l4re_env_get_cap_e()	294
12.46.2.4 l4re_env_get_cap_l()	295
12.46.2.5 l4re_kip()	296
12.47Integer Types	297
12.47.1 Detailed Description	298
12.47.2 Typedef Documentation	298
12.47.2.1 l4_int16_t	298
12.47.2.2 l4_int32_t	299
12.47.2.3 l4_int64_t	299
12.47.2.4 l4_int8_t	299
12.47.2.5 l4_uint16_t	299
12.47.2.6 l4_uint32_t	299
12.47.2.7 l4_uint64_t	299
12.47.2.8 l4_uint8_t	299
12.48Interface for asynchronous ISR handlers with a given IRQ capability.	300
12.48.1 Detailed Description	300
12.48.2 Function Documentation	300
12.48.2.1 l4irq_request_cap()	300
12.49Interface for asynchronous ISR handlers.	302
12.49.1 Detailed Description	302

12.49.2 Function Documentation	302
12.49.2.1 l4irq_release()	302
12.49.2.2 l4irq_request()	303
12.50 Interface using direct functionality.	304
12.50.1 Detailed Description	304
12.50.2 Function Documentation	304
12.50.2.1 l4irq_attach()	304
12.50.2.2 l4irq_attach_ft()	305
12.50.2.3 l4irq_attach_thread()	305
12.50.2.4 l4irq_attach_thread_ft()	306
12.50.2.5 l4irq_detach()	306
12.50.2.6 l4irq_unmask()	307
12.50.2.7 l4irq_unmask_and_wait_any()	307
12.50.2.8 l4irq_wait()	307
12.50.2.9 l4irq_wait_any()	309
12.51 Interface using direct functionality.	310
12.51.1 Detailed Description	310
12.51.2 Function Documentation	310
12.51.2.1 l4irq_attach_cap()	310
12.51.2.2 l4irq_attach_cap_ft()	311
12.51.2.3 l4irq_attach_thread_cap()	311
12.51.2.4 l4irq_attach_thread_cap_ft()	311
12.52 Internal Helpers	313
12.52.1 Detailed Description	313
12.53 Internal constants	314
12.53.1 Detailed Description	315
12.54 Internal functions	316
12.54.1 Detailed Description	316
12.55 Interrupt controller	317
12.55.1 Detailed Description	318

12.55.2 Typedef Documentation	318
12.55.2.1 l4_icu_info_t	318
12.55.3 Enumeration Type Documentation	319
12.55.3.1 L4_icu_flags	319
12.55.4 Function Documentation	319
12.55.4.1 l4_icu_bind()	319
12.55.4.2 l4_icu_bind_u()	320
12.55.4.3 l4_icu_info()	321
12.55.4.4 l4_icu_info_u()	322
12.55.4.5 l4_icu_mask()	322
12.55.4.6 l4_icu_mask_u()	323
12.55.4.7 l4_icu_msi_info()	324
12.55.4.8 l4_icu_msi_info_u()	325
12.55.4.9 l4_icu_set_mode()	325
12.55.4.10 l4_icu_set_mode_u()	326
12.55.4.11 l4_icu_unbind()	327
12.55.4.12 l4_icu_unbind_u()	328
12.55.4.13 l4_icu_unmask()	329
12.55.4.14 l4_icu_unmask_u()	329
12.56 Kernel Debugger	331
12.56.1 Detailed Description	331
12.56.2 Function Documentation	331
12.56.2.1 l4_debugger_global_id()	331
12.56.2.2 l4_debugger_kobj_to_id()	332
12.56.2.3 l4_debugger_set_object_name()	332
12.57 Kernel Interface Page	334
12.57.1 Detailed Description	335
12.57.2 Function Documentation	335
12.57.2.1 l4_kernel_info_version_offset()	335
12.57.2.2 l4_kip_clock()	336

12.57.2.3 l4_kip_clock_lw()	337
12.57.2.4 l4_kip_version()	337
12.57.2.5 l4_kip_version_string()	338
12.58 Kernel Interface Page API	339
12.58.1 Detailed Description	339
12.58.2 Macro Definition Documentation	339
12.58.2.1 l4util_kip_for_each_feature	339
12.58.3 Function Documentation	340
12.58.3.1 l4util_kip_kernel_abi_version()	340
12.58.3.2 l4util_kip_kernel_has_feature()	340
12.58.3.3 l4util_kip_kernel_is_ux()	340
12.58.3.4 l4util_memdesc_vm_high()	341
12.59 Kernel Objects	342
12.59.1 Detailed Description	343
12.60 Kumem allocator utility	344
12.60.1 Detailed Description	344
12.60.2 Function Documentation	344
12.60.2.1 l4re_util_kumem_alloc()	344
12.61 Kumem utilities	346
12.61.1 Detailed Description	346
12.61.2 Function Documentation	346
12.61.2.1 kumem_alloc()	346
12.62 L4 IPC Opcodes	348
12.62.1 Detailed Description	348
12.62.2 Enumeration Type Documentation	348
12.62.2.1 L4_icu_opcode	348
12.62.2.2 L4_ipc_gate_ops	349
12.62.2.3 L4_platform_ctl_ops	350
12.62.2.4 L4_task_ops	350
12.62.2.5 L4_thread_ops	350

12.62.2.6 L4_vcon_ops	351
12.63L4 V-BUS functions	352
12.63.1 Detailed Description	353
12.63.2 Enumeration Type Documentation	353
12.63.2.1 L4vbus_dma_domain_assign_flags	353
12.63.3 Function Documentation	353
12.63.3.1 l4vbus_assign_dma_domain()	353
12.63.3.2 l4vbus_get_device()	354
12.63.3.3 l4vbus_get_device_by_hid()	355
12.63.3.4 l4vbus_get_hid()	356
12.63.3.5 l4vbus_get_next_device()	356
12.63.3.6 l4vbus_get_resource()	357
12.63.3.7 l4vbus_is_compatible()	358
12.63.3.8 l4vbus_release_resource()	358
12.63.3.9 l4vbus_request_resource()	359
12.63.3.10 l4vbus_vicu_get_cap()	360
12.64L4 VIRTIO Block Device	361
12.64.1 Detailed Description	361
12.64.2 Enumeration Type Documentation	361
12.64.2.1 L4virtio_block_operations	361
12.64.2.2 L4virtio_block_status	362
12.65L4 VIRTIO Interface	363
12.65.1 Detailed Description	363
12.66L4 VIRTIO Transport Layer	364
12.66.1 Detailed Description	365
12.66.2 Typedef Documentation	365
12.66.2.1 l4virtio_config_queue_t	366
12.66.3 Enumeration Type Documentation	366
12.66.3.1 L4_virtio_cmd	366
12.66.3.2 L4_virtio_irq_status	366

12.66.3.3 L4_virtio_opcodes	367
12.66.3.4 L4virtio_device_ids	367
12.66.3.5 L4virtio_device_status	368
12.66.3.6 L4virtio_feature_bits	368
12.66.4 Function Documentation	368
12.66.4.1 l4virtio_config_queue()	368
12.66.4.2 l4virtio_config_queues()	369
12.66.4.3 l4virtio_device_config()	370
12.66.4.4 l4virtio_register_ds()	370
12.66.4.5 l4virtio_register_iface()	371
12.66.4.6 l4virtio_set_status()	372
12.67L4 kernel object type information	373
12.67.1 Detailed Description	373
12.67.2 Function Documentation	373
12.67.2.1 kobject_typeid()	373
12.68L4Re C Interface	375
12.68.1 Detailed Description	376
12.68.2 Function Documentation	376
12.68.2.1 l4re_inhibitor_acquire()	376
12.69L4Re C++ Interface	378
12.69.1 Detailed Description	379
12.70L4Re Capability API	380
12.70.1 Detailed Description	381
12.70.2 Function Documentation	381
12.70.2.1 make_auto_cap()	381
12.70.2.2 make_auto_del_cap()	381
12.70.2.3 make_ref_cap()	381
12.70.2.4 make_ref_del_cap()	382
12.70.3 Variable Documentation	382
12.70.3.1 cap_alloc	382

12.71 L4Re ELF Auxiliary Information	383
12.71.1 Detailed Description	384
12.71.2 Macro Definition Documentation	384
12.71.2.1 L4RE_ELF_AUX_ELEM	384
12.71.2.2 L4RE_ELF_AUX_ELEM_T	384
12.71.3 Enumeration Type Documentation	385
12.71.3.1 anonymous enum	385
12.72 L4Re Protocol identifiers	386
12.72.1 Detailed Description	386
12.73 L4Re Util C Interface	387
12.74 L4Re Util C++ Interface	388
12.74.1 Detailed Description	388
12.75 L4vbus GPIO functions	389
12.75.1 Detailed Description	390
12.75.2 Enumeration Type Documentation	390
12.75.2.1 L4vbus_gpio_generic_func	390
12.75.2.2 L4vbus_gpio_pull_modes	390
12.75.3 Function Documentation	391
12.75.3.1 l4vbus_gpio_config_get()	391
12.75.3.2 l4vbus_gpio_config_pad()	392
12.75.3.3 l4vbus_gpio_config_pull()	392
12.75.3.4 l4vbus_gpio_get()	393
12.75.3.5 l4vbus_gpio_multi_config_pad()	394
12.75.3.6 l4vbus_gpio_multi_get()	394
12.75.3.7 l4vbus_gpio_multi_set()	395
12.75.3.8 l4vbus_gpio_multi_setup()	396
12.75.3.9 l4vbus_gpio_set()	397
12.75.3.10 l4vbus_gpio_setup()	398
12.75.3.11 l4vbus_gpio_to_irq()	398
12.76 L4vbus PCI functions	400

12.76.1 Detailed Description	400
12.76.2 Function Documentation	400
12.76.2.1 l4vbus_pci_cfg_read()	400
12.76.2.2 l4vbus_pci_cfg_write()	401
12.76.2.3 l4vbus_pci_irq_enable()	402
12.76.2.4 l4vbus_pcidev_cfg_read()	403
12.76.2.5 l4vbus_pcidev_cfg_write()	404
12.76.2.6 l4vbus_pcidev_irq_enable()	404
12.77 Log interface	406
12.77.1 Detailed Description	406
12.77.2 Function Documentation	406
12.77.2.1 l4re_log_print()	406
12.77.2.2 l4re_log_print_srv()	407
12.77.2.3 l4re_log_printn()	408
12.77.2.4 l4re_log_printn_srv()	409
12.78 Logging interface	410
12.78.1 Detailed Description	410
12.79 Low-Level Thread Functions	411
12.80 Machine Restarting Function	412
12.80.1 Detailed Description	412
12.81 Memory allocator	413
12.81.1 Detailed Description	413
12.81.2 Enumeration Type Documentation	413
12.81.2.1 l4re_ma_flags	414
12.81.3 Function Documentation	414
12.81.3.1 l4re_ma_alloc()	414
12.81.3.2 l4re_ma_alloc_align()	415
12.81.3.3 l4re_ma_alloc_align_srv()	416
12.81.3.4 l4re_ma_free()	417
12.81.3.5 l4re_ma_free_srv()	418

12.82Memory descriptors (C version)	419
12.82.1 Detailed Description	420
12.82.2 Typedef Documentation	420
12.82.2.1 l4_kernel_info_mem_desc_t	420
12.82.3 Enumeration Type Documentation	420
12.82.3.1 l4_mem_info_sub_type_t	420
12.82.3.2 l4_mem_type_t	421
12.82.4 Function Documentation	421
12.82.4.1 l4_kernel_info_get_mem_desc_end()	421
12.82.4.2 l4_kernel_info_get_mem_desc_is_virtual()	421
12.82.4.3 l4_kernel_info_get_mem_desc_start()	422
12.82.4.4 l4_kernel_info_get_mem_desc_subtype()	422
12.82.4.5 l4_kernel_info_get_mem_desc_type()	422
12.82.4.6 l4_kernel_info_get_num_mem_descs()	423
12.82.4.7 l4_kernel_info_set_mem_desc()	423
12.83Memory operations.	424
12.83.1 Detailed Description	424
12.83.2 Enumeration Type Documentation	424
12.83.2.1 L4_mem_op_widths	424
12.83.3 Function Documentation	425
12.83.3.1 l4_mem_read()	425
12.83.3.2 l4_mem_write()	426
12.84Memory related	427
12.84.1 Detailed Description	428
12.84.2 Macro Definition Documentation	428
12.84.2.1 L4_LOG2_PAGESIZE	428
12.84.2.2 L4_LOG2_SUPERPAGESIZE	428
12.84.2.3 L4_PAGEMASK	429
12.84.2.4 L4_SUPERPAGEMASK	429
12.84.2.5 L4_SUPERPAGESIZE	429

12.84.3 Enumeration Type Documentation	429
12.84.3.1 l4_addr_consts_t	429
12.84.4 Function Documentation	430
12.84.4.1 l4_round_page()	430
12.84.4.2 l4_round_size()	431
12.84.4.3 l4_trunc_page()	432
12.84.4.4 l4_trunc_size()	433
12.85 Message Items	434
12.85.1 Detailed Description	434
12.85.2 Enumeration Type Documentation	434
12.85.2.1 l4_msg_item_consts_t	434
12.85.3 Function Documentation	435
12.85.3.1 l4_map_control()	435
12.85.3.2 l4_map_obj_control()	436
12.86 Message Registers (MRs)	437
12.86.1 Detailed Description	437
12.87 Message Tag	438
12.87.1 Detailed Description	439
12.87.2 Typedef Documentation	439
12.87.2.1 l4_msgtag_t	439
12.87.3 Enumeration Type Documentation	440
12.87.3.1 l4_msgtag_flags	440
12.87.3.2 l4_msgtag_protocol	440
12.87.3.3 L4_platform_ctl_proto	441
12.87.4 Function Documentation	441
12.87.4.1 l4_msgtag()	441
12.87.4.2 l4_msgtag_flags()	442
12.87.4.3 l4_msgtag_has_error()	443
12.87.4.4 l4_msgtag_is_exception()	444
12.87.4.5 l4_msgtag_is_io_page_fault()	445

12.87.4.6 l4_msgtag_is_page_fault()	446
12.87.4.7 l4_msgtag_is_preemption()	447
12.87.4.8 l4_msgtag_is_sigma0()	448
12.87.4.9 l4_msgtag_is_sys_exception()	449
12.87.4.10 l4_msgtag_items()	450
12.87.4.11 l4_msgtag_label()	451
12.87.4.12 l4_msgtag_words()	452
12.88 Name-space API	453
12.88.1 Detailed Description	453
12.89 Namespace interface	454
12.89.1 Detailed Description	454
12.89.2 Enumeration Type Documentation	454
12.89.2.1 l4re_ns_register_flags	454
12.89.3 Function Documentation	455
12.89.3.1 l4re_ns_query_srv()	455
12.89.3.2 l4re_ns_query_to_srv()	455
12.89.3.3 l4re_ns_register_obj_srv()	457
12.90 Object Invocation	459
12.90.1 Detailed Description	460
12.90.2 Enumeration Type Documentation	461
12.90.2.1 l4_syscall_flags_t	461
12.90.3 Function Documentation	461
12.90.3.1 l4_ipc()	461
12.90.3.2 l4_ipc_call()	463
12.90.3.3 l4_ipc_receive()	464
12.90.3.4 l4_ipc_reply_and_wait()	465
12.90.3.5 l4_ipc_send()	466
12.90.3.6 l4_ipc_send_and_wait()	468
12.90.3.7 l4_ipc_sleep()	469
12.90.3.8 l4_ipc_wait()	470

12.90.3.9 l4_sndfpage_add()	471
12.91 Parent API	472
12.91.1 Detailed Description	472
12.92 Platform Control C API	473
12.92.1 Detailed Description	473
12.92.2 Function Documentation	473
12.92.2.1 l4_platform_ctl_cpu_disable()	473
12.92.2.2 l4_platform_ctl_cpu_enable()	474
12.92.2.3 l4_platform_ctl_system_shutdown()	474
12.92.2.4 l4_platform_ctl_system_suspend()	475
12.93 Producer	476
12.93.1 Detailed Description	476
12.93.2 Function Documentation	476
12.93.2.1 l4shmc_chunk_ready()	476
12.93.2.2 l4shmc_chunk_ready_sig()	477
12.93.2.3 l4shmc_chunk_try_to_take()	477
12.93.2.4 l4shmc_is_chunk_clear()	478
12.94 Producer	479
12.94.1 Detailed Description	479
12.94.2 Function Documentation	479
12.94.2.1 l4shmc_trigger()	479
12.95 Random number support	480
12.95.1 Detailed Description	480
12.95.2 Function Documentation	480
12.95.2.1 l4util_rand()	480
12.95.2.2 l4util_srand()	480
12.96 Realtime API	482
12.97 Region map API	483
12.97.1 Detailed Description	483
12.98 Region map interface	484

12.98.1 Detailed Description	485
12.98.2 Enumeration Type Documentation	485
12.98.2.1 l4re_rm_flags_t	485
12.98.3 Function Documentation	485
12.98.3.1 l4re_rm_attach()	485
12.98.3.2 l4re_rm_attach_srv()	487
12.98.3.3 l4re_rm_detach()	487
12.98.3.4 l4re_rm_detach_ds()	488
12.98.3.5 l4re_rm_detach_ds_unmap()	489
12.98.3.6 l4re_rm_detach_srv()	490
12.98.3.7 l4re_rm_detach_unmap()	491
12.98.3.8 l4re_rm_find()	492
12.98.3.9 l4re_rm_find_srv()	493
12.98.3.10 l4re_rm_free_area()	493
12.98.3.11 l4re_rm_free_area_srv()	494
12.98.3.12 l4re_rm_reserve_area()	494
12.98.3.13 l4re_rm_reserve_area_srv()	495
12.98.3.14 l4re_rm_show_lists()	495
12.99 Scheduler	496
12.99.1 Detailed Description	497
12.99.2 Enumeration Type Documentation	497
12.99.2.1 L4_scheduler_ops	497
12.99.3 Function Documentation	497
12.99.3.1 l4_sched_cpu_set()	497
12.99.3.2 l4_scheduler_idle_time()	498
12.99.3.3 l4_scheduler_info()	499
12.99.3.4 l4_scheduler_is_online()	500
12.99.3.5 l4_scheduler_run_thread()	500
12.100 Server-Side IPC framework	502
12.100.1 Detailed Description	503

12.100.2	Enumeration Type Documentation	503
12.100.2.1	Reply_mode	503
12.100	Shared Memory Library	504
12.101.1	Detailed Description	505
12.101.2	Function Documentation	505
12.101.2.1	shmc_area_overhead()	505
12.101.2.2	shmc_area_size()	505
12.101.2.3	shmc_area_size_free()	506
12.101.2.4	shmc_attach()	506
12.101.2.5	shmc_attach_to()	507
12.101.2.6	shmc_chunk_overhead()	507
12.101.2.7	shmc_connect_chunk_signal()	508
12.101.2.8	shmc_create()	509
12.102	Sigma0 API	510
12.102.1	Detailed Description	511
12.102.2	Enumeration Type Documentation	511
12.102.2.1	sigma0_return_flags_t	511
12.102.3	Function Documentation	511
12.102.3.1	sigma0_debug_dump()	511
12.102.3.2	sigma0_map_anypage()	512
12.102.3.3	sigma0_map_errstr()	512
12.102.3.4	sigma0_map_iomem()	513
12.102.3.5	sigma0_map_kip()	513
12.102.3.6	sigma0_map_mem()	514
12.102.3.7	sigma0_map_tbuf()	514
12.102.3.8	sigma0_new_client()	515
12.103	Signals	516
12.103.1	Detailed Description	516
12.103.2	Function Documentation	516
12.103.2.1	shmc_add_signal()	516

12.103.2.24	shmc_attach_signal()	517
12.103.2.34	shmc_attach_signal_to()	517
12.103.2.44	shmc_check_magic()	518
12.103.2.54	shmc_get_signal_to()	518
12.103.2.64	shmc_signal_cap()	519
12.104	Small C++ Template Library	520
12.104.1	Detailed Description	521
12.104.2	Function Documentation	521
12.104.2.1	max()	521
12.104.2.2	min()	522
12.104.2.3	operator new()	522
12.105	task	524
12.105.1	Detailed Description	525
12.105.2	Enumeration Type Documentation	525
12.105.2.14	_unmap_flags_t	525
12.105.3	Function Documentation	526
12.105.3.14	_task_add_ku_mem()	526
12.105.3.24	_task_cap_equal()	526
12.105.3.34	_task_cap_has_child()	527
12.105.3.44	_task_cap_valid()	527
12.105.3.54	_task_delete_obj()	528
12.105.3.64	_task_map()	528
12.105.3.74	_task_release_cap()	529
12.105.3.84	_task_unmap()	529
12.105.3.94	_task_unmap_batch()	530
12.106	thread	532
12.106.1	Detailed Description	533
12.106.2	Enumeration Type Documentation	534
12.106.2.14	_thread_control_flags	534
12.106.2.24	_thread_control_mr_indices	534

12.106.2.3.4_thread_ex_regs_flags	536
12.106.3Function Documentation	536
12.106.3.14_thread_arm_set_tpidruro()	536
12.106.3.24_thread_ex_regs()	537
12.106.3.34_thread_ex_regs_ret()	538
12.106.3.44_thread_ex_regs_ret_u()	539
12.106.3.54_thread_ex_regs_u()	540
12.106.3.64_thread_modify_sender_add()	541
12.106.3.74_thread_modify_sender_commit()	542
12.106.3.84_thread_modify_sender_start()	542
12.106.3.94_thread_register_del_irq()	542
12.106.3.10_thread_stats_time()	543
12.106.3.11_thread_switch()	543
12.106.3.12_thread_vcpu_control()	544
12.106.3.13_thread_vcpu_control_ext()	544
12.106.3.14_thread_vcpu_control_ext_u()	545
12.106.3.15_thread_vcpu_control_u()	546
12.106.3.16_thread_vcpu_resume_commit()	547
12.106.3.17_thread_vcpu_resume_start()	548
12.106.3.18_thread_yield()	548
12.107Thread Control Registers (TCRs)	549
12.107.1Detailed Description	549
12.108Thread control	550
12.108.1Detailed Description	550
12.108.2Function Documentation	551
12.108.2.14_thread_control_alien()	551
12.108.2.24_thread_control_bind()	551
12.108.2.34_thread_control_commit()	552
12.108.2.44_thread_control_exc_handler()	552
12.108.2.54_thread_control_pager()	553

12.108.2.64_thread_control_start()	553
12.108.2.74_thread_control_ux_host_syscall()	554
12.109Timeouts	555
12.109.1Detailed Description	556
12.109.2Macro Definition Documentation	556
12.109.2.1L4_IPC_TIMEOUT_0	556
12.109.3Typedef Documentation	557
12.109.3.1l4_timeout_s	557
12.109.3.2l4_timeout_t	557
12.109.4Enumeration Type Documentation	557
12.109.4.1l4_timeout_abs_validity	557
12.109.5Function Documentation	557
12.109.5.1l4_ipc_timeout()	557
12.109.5.2l4_rcv_timeout()	558
12.109.5.3l4_snd_timeout()	558
12.109.5.4l4_timeout()	559
12.109.5.5l4_timeout_abs()	559
12.109.5.6l4_timeout_get()	560
12.109.5.7l4_timeout_is_absolute()	561
12.109.5.8l4_timeout_rel()	562
12.109.5.9l4_timeout_rel_get()	562
12.109.5.10l4_utcb_mr64_idx()	563
12.110Timestamp Counter	564
12.110.1Detailed Description	565
12.110.2Function Documentation	565
12.110.2.1l4_busy_wait_ns()	565
12.110.2.2l4_busy_wait_us()	566
12.110.2.3l4_calibrate_tsc()	567
12.110.2.4l4_get_hz()	567
12.110.2.5l4_ns_to_tsc()	567

12.110.2.64_rdpmc()	568
12.110.2.74_rdpmc_32()	569
12.110.2.84_rdtsc()	569
12.110.2.94_rdtsc_32()	570
12.110.2.10_tsc_init()	570
12.110.2.114_tsc_to_ns()	571
12.110.2.112_tsc_to_s_and_ns()	571
12.110.2.113_tsc_to_us()	572
12.11 Utility Functions	573
12.111. Detailed Description	574
12.111.2 Function Documentation	574
12.111.2.114_sleep()	574
12.111.2.24_touch_ro()	575
12.111.2.34_touch_rw()	576
12.111.2.44_usleep()	576
12.111.2.54util_micros2l4to()	577
12.111.2.64util_splitlog2_hdl()	577
12.111.2.74util_splitlog2_size()	578
12.112 VM API for SVM	580
12.112. Detailed Description	580
12.113 VM API for TZ	581
12.113. Detailed Description	581
12.114 VM API for VMX	582
12.114. Detailed Description	583
12.114.2 Enumeration Type Documentation	583
12.114.2.1 anonymous enum	583
12.114.2.2.4_vm_vmx_caps_regs	584
12.114.2.3.4_vm_vmx_dfl1_regs	585
12.114.3 Function Documentation	585
12.114.3.114_vm_vmx_clear()	585

12.114.3.24_vm_vmx_field_len()	586
12.114.3.34_vm_vmx_field_order()	586
12.114.3.44_vm_vmx_get_caps()	587
12.114.3.54_vm_vmx_get_caps_default1()	587
12.114.3.64_vm_vmx_get_cr2_index()	588
12.114.3.74_vm_vmx_ptr_load()	588
12.114.3.84_vm_vmx_read()	589
12.114.3.94_vm_vmx_read_16()	590
12.114.3.104_vm_vmx_read_32()	591
12.114.3.114_vm_vmx_read_64()	591
12.114.3.124_vm_vmx_read_nat()	592
12.114.3.134_vm_vmx_write()	593
12.114.3.144_vm_vmx_write_16()	593
12.114.3.154_vm_vmx_write_32()	594
12.114.3.164_vm_vmx_write_64()	595
12.114.3.174_vm_vmx_write_nat()	595
12.115Video API	598
12.115.1Detailed Description	599
12.115.2Typedef Documentation	599
12.115.2.14re_video_view_t	599
12.115.3Enumeration Type Documentation	600
12.115.3.14re_video_goos_info_flags_t	600
12.115.3.24re_video_view_info_flags_t	600
12.115.4Function Documentation	600
12.115.4.14re_video_goos_create_buffer()	601
12.115.4.24re_video_goos_create_view()	601
12.115.4.34re_video_goos_delete_buffer()	601
12.115.4.44re_video_goos_delete_view()	603
12.115.4.54re_video_goos_get_static_buffer()	603
12.115.4.64re_video_goos_get_view()	603

12.115.4.74	re_video_goos_info()	604
12.115.4.84	re_video_goos_refresh()	604
12.115.4.94	re_video_view_get_info()	605
12.115.4.104	re_video_view_refresh()	605
12.115.4.114	re_video_view_set_info()	606
12.115.4.124	re_video_view_set_viewport()	606
12.115.4.134	re_video_view_stack()	606
12.116	Virtual Console	608
12.116.1	Detailed Description	609
12.116.2	Enumeration Type Documentation	609
12.116.2.1	L4_vcon_i_flags	609
12.116.2.2	L4_vcon_l_flags	610
12.116.2.3	L4_vcon_o_flags	610
12.116.2.4	L4_vcon_size_consts	610
12.116.3	Function Documentation	611
12.116.3.1	L4_vcon_get_attr()	611
12.116.3.2	L4_vcon_get_attr_u()	612
12.116.3.3	L4_vcon_read()	612
12.116.3.4	L4_vcon_read_u()	613
12.116.3.5	L4_vcon_read_with_flags()	614
12.116.3.6	L4_vcon_send()	615
12.116.3.7	L4_vcon_send_u()	616
12.116.3.8	L4_vcon_set_attr()	617
12.116.3.9	L4_vcon_set_attr_u()	618
12.116.3.10	L4_vcon_write()	618
12.116.3.11	L4_vcon_write_u()	619
12.117	Virtual Machines	621
12.117.1	Detailed Description	621
12.118	Virtual Registers (UTCBS)	622
12.118.1	Detailed Description	623

12.118.2	Typedef Documentation	623
12.118.2.1	l4_utcb_t	623
12.118.3	Function Documentation	624
12.118.3.1	l4_utcb_br()	624
12.118.3.2	l4_utcb_mr()	624
12.118.3.3	l4_utcb_tcr()	625
12.119	amd64 Virtual Registers (UTCB)	626
12.119.1	Detailed Description	626
12.120	CPU API	627
12.120.1	Detailed Description	628
12.120.2	Enumeration Type Documentation	628
12.120.2.1	l4_vcpu_state_flags	628
12.120.2.2	l4_vcpu_state_offset	628
12.120.2.3	l4_vcpu_sticky_flags	628
12.121	CPU Support Library	630
12.121.1	Detailed Description	631
12.121.2	Function Documentation	631
12.121.2.1	l4_vcpu_irq_disable()	631
12.121.2.2	l4_vcpu_irq_disable_save()	632
12.121.2.3	l4_vcpu_irq_enable()	632
12.121.2.4	l4_vcpu_irq_restore()	633
12.121.2.5	l4_vcpu_is_irq_entry()	634
12.121.2.6	l4_vcpu_is_page_fault_entry()	635
12.121.2.7	l4_vcpu_print_state()	636
12.121.2.8	l4_vcpu_wait_for_event()	636
12.122	x86 Virtual Registers (UTCB)	638
12.122.1	Detailed Description	638
12.122.2	Enumeration Type Documentation	638
12.122.2.1	l4_utcb_consts_x86	638

13 Namespace Documentation	641
13.1 cxx Namespace Reference	641
13.1.1 Detailed Description	643
13.2 cxx::Bits Namespace Reference	643
13.2.1 Detailed Description	643
13.3 L4 Namespace Reference	643
13.3.1 Detailed Description	647
13.3.2 Enumeration Type Documentation	647
13.3.2.1 anonymous enum	647
13.3.3 Function Documentation	647
13.3.3.1 cap_cast() [1/2]	647
13.3.3.2 cap_cast() [2/2]	648
13.3.3.3 cap_dynamic_cast()	649
13.3.3.4 cap_reinterpret_cast() [1/2]	650
13.3.3.5 cap_reinterpret_cast() [2/2]	651
13.3.3.6 throw_ipc_exception() [1/2]	652
13.3.3.7 throw_ipc_exception() [2/2]	653
13.4 L4::lpc Namespace Reference	654
13.4.1 Detailed Description	656
13.4.2 Function Documentation	656
13.4.2.1 buf_cp_in()	656
13.4.2.2 buf_cp_out()	657
13.4.2.3 buf_in()	657
13.4.2.4 make_cap()	658
13.4.2.5 make_cap_full()	659
13.4.2.6 make_cap_rw()	660
13.4.2.7 make_cap_rws()	660
13.4.2.8 msg_ptr()	661
13.4.2.9 read()	661
13.4.2.10 str_cp_in()	662

13.5 L4::lpc::Msg Namespace Reference	662
13.5.1 Detailed Description	664
13.5.2 Enumeration Type Documentation	664
13.5.2.1 anonymous enum	664
13.5.3 Function Documentation	664
13.5.3.1 align_to() [1/2]	664
13.5.3.2 align_to() [2/2]	665
13.5.3.3 check_size() [1/2]	666
13.5.3.4 check_size() [2/2]	666
13.5.3.5 msg_add()	667
13.5.3.6 msg_get()	668
13.6 L4::lpc_svr Namespace Reference	668
13.6.1 Detailed Description	669
13.7 L4::Typeid Namespace Reference	669
13.7.1 Detailed Description	670
13.8 L4::Types Namespace Reference	670
13.8.1 Detailed Description	670
13.9 L4Re Namespace Reference	670
13.9.1 Detailed Description	671
13.9.2 Function Documentation	672
13.9.2.1 chkcap()	672
13.9.2.2 chksys() [1/3]	673
13.9.2.3 chksys() [2/3]	674
13.9.2.4 chksys() [3/3]	675
13.10 L4Re::Vfs Namespace Reference	675
13.10.1 Detailed Description	676
13.11 L4vbus Namespace Reference	676
13.11.1 Detailed Description	677
13.12 L4virtio Namespace Reference	677
13.12.1 Detailed Description	677

14 Data Structure Documentation	679
14.1 <code>cxx::Auto_ptr< T ></code> Class Template Reference	679
14.1.1 Detailed Description	680
14.1.2 Member Typedef Documentation	680
14.1.2.1 <code>Ref_type</code>	680
14.1.3 Constructor & Destructor Documentation	681
14.1.3.1 <code>Auto_ptr()</code> [1/2]	681
14.1.3.2 <code>Auto_ptr()</code> [2/2]	681
14.1.3.3 <code>~Auto_ptr()</code>	681
14.1.4 Member Function Documentation	681
14.1.4.1 <code>get()</code>	682
14.1.4.2 <code>operator Priv_type*()</code>	682
14.1.4.3 <code>operator*()</code>	682
14.1.4.4 <code>operator->()</code>	682
14.1.4.5 <code>operator=()</code>	682
14.1.4.6 <code>release()</code>	683
14.2 <code>cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC ></code> Class Template Reference	684
14.2.1 Detailed Description	686
14.2.2 Constructor & Destructor Documentation	686
14.2.2.1 <code>Avl_map()</code>	686
14.2.3 Member Function Documentation	687
14.2.3.1 <code>insert()</code>	687
14.2.3.2 <code>operator[]()</code> [1/2]	687
14.2.3.3 <code>operator[]()</code> [2/2]	688
14.3 <code>cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC ></code> Class Template Reference	688
14.3.1 Detailed Description	690
14.4 <code>cxx::Avl_tree< Node, Get_key, Compare ></code> Class Template Reference	691
14.4.1 Detailed Description	694
14.4.2 Member Function Documentation	695
14.4.2.1 <code>insert()</code>	695

14.4.2.2	remove()	695
14.5	cxx::Avl_tree_node Class Reference	696
14.5.1	Detailed Description	698
14.6	cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc > Class Template Reference	699
14.6.1	Detailed Description	700
14.6.2	Member Enumeration Documentation	701
14.6.2.1	anonymous enum	701
14.6.3	Member Function Documentation	701
14.6.3.1	free_objects()	701
14.6.3.2	total_objects()	702
14.7	cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc > Class Template Reference	702
14.7.1	Detailed Description	704
14.7.2	Member Enumeration Documentation	705
14.7.2.1	anonymous enum	705
14.7.3	Member Function Documentation	705
14.7.3.1	alloc()	705
14.7.3.2	free()	705
14.7.3.3	free_objects()	706
14.7.3.4	total_objects()	706
14.8	cxx::Bitfield< T, LSB, MSB > Class Template Reference	707
14.8.1	Detailed Description	708
14.8.2	Member Typedef Documentation	708
14.8.2.1	Bits_type	708
14.8.2.2	Ref	709
14.8.2.3	Ref_unshifted	709
14.8.2.4	Shift_type	709
14.8.2.5	Val	709
14.8.2.6	Val_unshifted	709
14.8.3	Member Enumeration Documentation	709
14.8.3.1	anonymous enum	709

14.8.3.2	Masks	710
14.8.4	Member Function Documentation	710
14.8.4.1	get()	710
14.8.4.2	get_unshifted()	711
14.8.4.3	set()	711
14.8.4.4	set_dirty()	712
14.8.4.5	set_unshifted()	713
14.8.4.6	set_unshifted_dirty()	714
14.8.4.7	val()	715
14.8.4.8	val_dirty()	716
14.8.4.9	val_unshifted()	716
14.9	cxx::Bitfield< T, LSB, MSB >::Value< TT > Class Template Reference	717
14.9.1	Detailed Description	718
14.10	cxx::Bitfield< T, LSB, MSB >::Value_base< TT > Class Template Reference	718
14.10.1	Detailed Description	719
14.11	cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT > Class Template Reference	720
14.11.1	Detailed Description	721
14.12	cxx::Bitmap< BITS > Class Template Reference	721
14.12.1	Detailed Description	723
14.12.2	Constructor & Destructor Documentation	724
14.12.2.1	Bitmap()	724
14.12.3	Member Function Documentation	724
14.12.3.1	scan_zero()	724
14.13	cxx::Bitmap_base Class Reference	725
14.13.1	Detailed Description	728
14.13.2	Member Enumeration Documentation	728
14.13.2.1	anonymous enum	728
14.13.3	Member Function Documentation	729
14.13.3.1	bit() [1/2]	729
14.13.3.2	bit() [2/2]	730

14.13.3.3 <code>bit_index()</code>	730
14.13.3.4 <code>chars()</code>	731
14.13.3.5 <code>clear_bit()</code>	731
14.13.3.6 <code>operator[]()</code> [1/2]	732
14.13.3.7 <code>operator[]()</code> [2/2]	733
14.13.3.8 <code>scan_zero()</code>	733
14.13.3.9 <code>set_bit()</code>	734
14.13.3.10 <code>word_index()</code>	735
14.13.3.11 <code>words()</code>	736
14.14 <code>cxx::Bitmap_base::Bit</code> Class Reference	736
14.14.1 Detailed Description	736
14.15 <code>cxx::Bitmap_base::Char< BITS ></code> Class Template Reference	737
14.15.1 Detailed Description	737
14.16 <code>cxx::Bitmap_base::Word< BITS ></code> Class Template Reference	737
14.16.1 Detailed Description	738
14.17 <code>cxx::Bits::Avl_map_get_key< KEY_TYPE ></code> Struct Template Reference	739
14.17.1 Detailed Description	739
14.18 <code>cxx::Bits::Avl_set_get_key< KEY_TYPE ></code> Struct Template Reference	739
14.18.1 Detailed Description	740
14.19 <code>cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY ></code> Class Template Reference	740
14.19.1 Detailed Description	743
14.19.2 Constructor & Destructor Documentation	744
14.19.2.1 <code>Base_avl_set()</code> [1/2]	744
14.19.2.2 <code>Base_avl_set()</code> [2/2]	744
14.19.3 Member Function Documentation	744
14.19.3.1 <code>begin()</code> [1/2]	745
14.19.3.2 <code>begin()</code> [2/2]	745
14.19.3.3 <code>end()</code> [1/2]	746
14.19.3.4 <code>end()</code> [2/2]	746
14.19.3.5 <code>erase()</code>	746

14.19.3.6 find_node()	747
14.19.3.7 insert()	747
14.19.3.8 lower_bound_node()	748
14.19.3.9 rbegin() [1/2]	748
14.19.3.10 begin() [2/2]	748
14.19.3.11 remove()	749
14.19.3.12 end() [1/2]	749
14.19.3.13 end() [2/2]	749
14.20 cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node Class Reference	750
14.20.1 Detailed Description	750
14.20.2 Member Function Documentation	751
14.20.2.1 operator*()	751
14.20.2.2 operator->()	751
14.20.2.3 valid()	751
14.21 cxx::Bits::Basic_list< POLICY > Class Template Reference	752
14.21.1 Detailed Description	753
14.21.2 Member Function Documentation	753
14.21.2.1 clear()	753
14.21.2.2 iter()	753
14.22 cxx::Bits::Bst< Node, Get_key, Compare > Class Template Reference	754
14.22.1 Detailed Description	757
14.22.2 Member Function Documentation	757
14.22.2.1 begin() [1/2]	758
14.22.2.2 begin() [2/2]	758
14.22.2.3 dir() [1/2]	758
14.22.2.4 dir() [2/2]	759
14.22.2.5 end() [1/2]	760
14.22.2.6 end() [2/2]	761
14.22.2.7 find()	761
14.22.2.8 find_node()	762

14.22.2.9	lower_bound_node()	762
14.22.2.10	begin() [1/2]	763
14.22.2.11	rbegin() [2/2]	764
14.22.2.12	remove_all()	764
14.22.2.13	remove_tree()	765
14.22.2.14	end() [1/2]	766
14.22.2.15	end() [2/2]	766
14.23	cxx::Bits::Bst_node Class Reference	767
14.23.1	Detailed Description	768
14.24	cxx::Bits::Direction Struct Reference	769
14.24.1	Detailed Description	770
14.24.2	Member Enumeration Documentation	770
14.24.2.1	Direction_e	770
14.24.3	Member Function Documentation	770
14.24.3.1	operator"!"()	770
14.25	cxx::H_list< T, POLICY > Class Template Reference	771
14.25.1	Detailed Description	774
14.25.2	Member Function Documentation	774
14.25.2.1	erase()	774
14.25.2.2	insert()	775
14.25.2.3	insert_after()	775
14.25.2.4	insert_before()	776
14.25.2.5	iter()	776
14.25.2.6	pop_front()	777
14.25.2.7	remove()	777
14.25.2.8	replace()	777
14.26	cxx::H_list_item_t< ELEM_TYPE > Class Template Reference	778
14.26.1	Detailed Description	779
14.26.2	Constructor & Destructor Documentation	779
14.26.2.1	H_list_item_t()	779

14.26.2.2 ~H_list_item_t()	780
14.27cxx::H_list_t< T > Struct Template Reference	780
14.27.1 Detailed Description	782
14.28cxx::List< D, Alloc > Class Template Reference	783
14.28.1 Detailed Description	784
14.28.2 Member Function Documentation	784
14.28.2.1 items()	784
14.28.2.2 operator[]() [1/2]	784
14.28.2.3 operator[]() [2/2]	784
14.28.2.4 push_back()	785
14.28.2.5 push_front()	785
14.28.2.6 remove()	785
14.28.2.7 size()	785
14.29cxx::List< D, Alloc >::Iter Class Reference	786
14.29.1 Detailed Description	786
14.30cxx::List_alloc Class Reference	787
14.30.1 Detailed Description	787
14.30.2 Constructor & Destructor Documentation	787
14.30.2.1 List_alloc()	788
14.30.3 Member Function Documentation	788
14.30.3.1 alloc()	788
14.30.3.2 alloc_max()	789
14.30.3.3 avail()	790
14.30.3.4 free()	791
14.31cxx::List_item Class Reference	791
14.31.1 Detailed Description	793
14.31.2 Member Function Documentation	793
14.31.2.1 get_next_item()	793
14.31.2.2 get_prev_item()	793
14.31.2.3 insert_next_item()	794

14.31.2.4 insert_prev_item()	794
14.31.2.5 push_back()	794
14.31.2.6 push_front()	795
14.31.2.7 remove()	795
14.31.2.8 remove_me()	796
14.32cxx::List_item::Iter Class Reference	797
14.32.1 Detailed Description	799
14.32.2 Member Function Documentation	799
14.32.2.1 remove_me()	799
14.33cxx::List_item::T_iter< T, Poly > Class Template Reference	800
14.33.1 Detailed Description	802
14.34cxx::Lt_functor< Obj > Struct Template Reference	803
14.34.1 Detailed Description	803
14.35cxx::New_allocator< _Type > Class Template Reference	803
14.35.1 Detailed Description	804
14.36cxx::Nothrow Class Reference	804
14.36.1 Detailed Description	804
14.37cxx::Pair< First, Second > Struct Template Reference	805
14.37.1 Detailed Description	805
14.37.2 Constructor & Destructor Documentation	806
14.37.2.1 Pair()	806
14.38cxx::Pair_first_compare< Cmp, Typ > Class Template Reference	806
14.38.1 Detailed Description	807
14.38.2 Constructor & Destructor Documentation	807
14.38.2.1 Pair_first_compare()	807
14.38.3 Member Function Documentation	807
14.38.3.1 operator()()	808
14.39cxx::Ref_ptr< T, CNT > Class Template Reference	808
14.39.1 Detailed Description	810
14.39.2 Constructor & Destructor Documentation	811

14.39.2.1 Ref_ptr() [1/3]	811
14.39.2.2 Ref_ptr() [2/3]	812
14.39.2.3 Ref_ptr() [3/3]	812
14.39.3 Member Function Documentation	812
14.39.3.1 get()	812
14.39.3.2 ptr()	813
14.39.3.3 release()	813
14.40cxx::S_list< T, POLICY > Class Template Reference	814
14.40.1 Detailed Description	816
14.40.2 Member Function Documentation	816
14.40.2.1 pop_front()	816
14.41cxx::Slab< Type, Slab_size, Max_free, Alloc > Class Template Reference	816
14.41.1 Detailed Description	819
14.41.2 Member Function Documentation	819
14.41.2.1 alloc()	819
14.41.2.2 free()	819
14.42cxx::Slab_static< Type, Slab_size, Max_free, Alloc > Class Template Reference	820
14.42.1 Detailed Description	822
14.42.2 Member Function Documentation	823
14.42.2.1 alloc()	823
14.43cxx::static_vector< T, IDX > Class Template Reference	823
14.43.1 Detailed Description	824
14.44cxx::String Class Reference	825
14.44.1 Detailed Description	826
14.45cxx::Weak_ref< T > Class Template Reference	826
14.45.1 Detailed Description	828
14.46cxx::Weak_ref_base Class Reference	829
14.46.1 Detailed Description	831
14.47Elf32_Dyn Struct Reference	832
14.47.1 Detailed Description	832

14.47.2 Field Documentation	832
14.47.2.1 d_val	832
14.48Elf32_Ehdr Struct Reference	833
14.48.1 Detailed Description	834
14.48.2 Field Documentation	834
14.48.2.1 e_phnum	834
14.48.2.2 e_shnum	834
14.49Elf32_Phdr Struct Reference	835
14.49.1 Detailed Description	835
14.50Elf32_Shdr Struct Reference	836
14.50.1 Detailed Description	837
14.51Elf32_Sym Struct Reference	837
14.51.1 Detailed Description	838
14.52Elf64_Dyn Struct Reference	838
14.52.1 Detailed Description	838
14.52.2 Field Documentation	839
14.52.2.1 d_val	839
14.53Elf64_Ehdr Struct Reference	839
14.53.1 Detailed Description	840
14.53.2 Field Documentation	840
14.53.2.1 e_phnum	840
14.53.2.2 e_shnum	841
14.54Elf64_Phdr Struct Reference	841
14.54.1 Detailed Description	842
14.55Elf64_Shdr Struct Reference	842
14.55.1 Detailed Description	843
14.56Elf64_Sym Struct Reference	843
14.56.1 Detailed Description	844
14.57gfxbitmap_offset Struct Reference	844
14.57.1 Detailed Description	845

14.58L4::Alloc_list Class Reference	845
14.58.1 Detailed Description	845
14.59L4::Base_exception Class Reference	846
14.59.1 Detailed Description	847
14.60L4::Basic_registry Class Reference	847
14.60.1 Detailed Description	849
14.60.2 Member Function Documentation	849
14.60.2.1 dispatch()	849
14.60.2.2 find()	850
14.61L4::Bounds_error Class Reference	850
14.61.1 Detailed Description	852
14.62L4::Cap< T > Class Template Reference	853
14.62.1 Detailed Description	855
14.62.2 Constructor & Destructor Documentation	855
14.62.2.1 Cap() [1/4]	855
14.62.2.2 Cap() [2/4]	856
14.62.2.3 Cap() [3/4]	856
14.62.2.4 Cap() [4/4]	856
14.62.3 Member Function Documentation	857
14.62.3.1 copy()	857
14.62.3.2 move()	857
14.63L4::Cap_base Class Reference	858
14.63.1 Detailed Description	859
14.63.2 Member Enumeration Documentation	860
14.63.2.1 Cap_type	860
14.63.2.2 No_init_type	860
14.63.3 Constructor & Destructor Documentation	860
14.63.3.1 Cap_base() [1/2]	860
14.63.3.2 Cap_base() [2/2]	861
14.63.4 Member Function Documentation	861

14.63.4.1 cap()	861
14.63.4.2 copy()	862
14.63.4.3 fpage()	863
14.63.4.4 is_valid()	864
14.63.4.5 move()	865
14.63.4.6 snd_base()	866
14.63.4.7 validate() [1/2]	867
14.63.4.8 validate() [2/2]	868
14.63.5 Field Documentation	868
14.63.5.1 _c	868
14.64L4::Com_error Class Reference	869
14.64.1 Detailed Description	871
14.64.2 Constructor & Destructor Documentation	871
14.64.2.1 Com_error()	871
14.65L4::Debugger Class Reference	871
14.65.1 Detailed Description	874
14.65.2 Member Function Documentation	874
14.65.2.1 get_object_name()	874
14.65.2.2 global_id()	875
14.65.2.3 kobj_to_id()	875
14.65.2.4 query_log_name()	875
14.65.2.5 query_log_typeid()	876
14.65.2.6 set_object_name()	877
14.65.2.7 switch_log()	877
14.66L4::Element_already_exists Class Reference	878
14.66.1 Detailed Description	879
14.67L4::Element_not_found Class Reference	880
14.67.1 Detailed Description	881
14.68L4::Exception_tracer Class Reference	882
14.68.1 Detailed Description	883

14.69L4::Factory Class Reference	883
14.69.1 Detailed Description	886
14.69.2 Member Function Documentation	887
14.69.2.1 create() [1/2]	887
14.69.2.2 create() [2/2]	888
14.69.2.3 create_factory()	889
14.69.2.4 create_gate()	890
14.69.2.5 create_irq()	891
14.69.2.6 create_task()	892
14.69.2.7 create_thread()	893
14.69.2.8 create_vm()	894
14.70L4::Factory::Lstr Struct Reference	895
14.70.1 Detailed Description	896
14.70.2 Constructor & Destructor Documentation	896
14.70.2.1 Lstr()	896
14.71L4::Factory::Nil Struct Reference	896
14.71.1 Detailed Description	897
14.72L4::Factory::S Class Reference	897
14.72.1 Detailed Description	898
14.72.2 Constructor & Destructor Documentation	898
14.72.2.1 S() [1/2]	898
14.72.2.2 S() [2/2]	898
14.72.3 Member Function Documentation	899
14.72.3.1 operator l4_msgtag_t()	899
14.72.3.2 operator<<() [1/6]	899
14.72.3.3 operator<<() [2/6]	900
14.72.3.4 operator<<() [3/6]	900
14.72.3.5 operator<<() [4/6]	900
14.72.3.6 operator<<() [5/6]	901
14.72.3.7 operator<<() [6/6]	901

14.73L4::lcu Class Reference	901
14.73.1 Detailed Description	904
14.73.2 Member Function Documentation	904
14.73.2.1 bind()	904
14.73.2.2 info()	905
14.73.2.3 mask()	906
14.73.2.4 msi_info()	907
14.73.2.5 set_mode()	908
14.73.2.6 unbind()	908
14.74L4::lcu::Info Class Reference	909
14.74.1 Detailed Description	910
14.75L4::Invalid_capability Class Reference	910
14.75.1 Detailed Description	912
14.75.2 Constructor & Destructor Documentation	913
14.75.2.1 Invalid_capability()	913
14.75.3 Member Function Documentation	913
14.75.3.1 cap()	913
14.76L4::io_pager Class Reference	914
14.76.1 Detailed Description	915
14.76.2 Member Function Documentation	915
14.76.2.1 io_page_fault()	915
14.77L4::lomu Class Reference	916
14.77.1 Detailed Description	917
14.77.2 Member Function Documentation	918
14.77.2.1 bind()	918
14.77.2.2 unbind()	918
14.78L4::IOModifier Class Reference	918
14.78.1 Detailed Description	919
14.79L4::lpc::Array< ELEM_TYPE, LEN_TYPE > Struct Template Reference	919
14.79.1 Detailed Description	921

14.80L4::lpc::Array_in_buf< ELEM_TYPE, LEN_TYPE, MAX > Struct Template Reference	922
14.80.1 Detailed Description	923
14.81L4::lpc::Array_ref< ELEM_TYPE, LEN_TYPE > Struct Template Reference	923
14.81.1 Detailed Description	925
14.82L4::lpc::As_value< T > Struct Template Reference	925
14.82.1 Detailed Description	926
14.83L4::lpc::Buf_item Class Reference	926
14.83.1 Detailed Description	927
14.84L4::lpc::Call Struct Reference	927
14.84.1 Detailed Description	928
14.85L4::lpc::Call_t< RIGHTS > Struct Template Reference	928
14.85.1 Detailed Description	929
14.86L4::lpc::Call_zero_send_timeout Struct Reference	930
14.86.1 Detailed Description	931
14.87L4::lpc::Cap< T > Class Template Reference	931
14.87.1 Detailed Description	933
14.87.2 Member Enumeration Documentation	933
14.87.2.1 anonymous enum	933
14.87.3 Constructor & Destructor Documentation	933
14.87.3.1 Cap()	934
14.87.4 Member Function Documentation	934
14.87.4.1 from_ci()	934
14.88L4::lpc::Gen_fpage< T > Class Template Reference	934
14.88.1 Detailed Description	936
14.88.2 Member Function Documentation	936
14.88.2.1 cap_received()	936
14.88.2.2 id_received()	937
14.88.2.3 is_compound()	937
14.88.2.4 local_id_received()	937
14.89L4::lpc::In_out< T > Struct Template Reference	938

14.89.1 Detailed Description	938
14.90L4::lpc::lostream Class Reference	938
14.90.1 Detailed Description	941
14.90.2 Constructor & Destructor Documentation	941
14.90.2.1 lostream()	941
14.90.3 Member Function Documentation	942
14.90.3.1 call()	942
14.90.3.2 reply_and_wait() [1/2]	943
14.90.3.3 reply_and_wait() [2/2]	943
14.90.3.4 reset()	944
14.91L4::lpc::Istream Class Reference	945
14.91.1 Detailed Description	948
14.91.2 Constructor & Destructor Documentation	948
14.91.2.1 Istream()	949
14.91.3 Member Function Documentation	949
14.91.3.1 get() [1/3]	949
14.91.3.2 get() [2/3]	950
14.91.3.3 get() [3/3]	950
14.91.3.4 receive()	951
14.91.3.5 reset()	952
14.91.3.6 skip()	952
14.91.3.7 tag() [1/2]	953
14.91.3.8 tag() [2/2]	954
14.91.3.9 wait() [1/2]	954
14.91.3.10wait() [2/2]	954
14.92L4::lpc::Msg::CInt_val_ops< MTYPE, DIR, CLASS > Struct Template Reference	955
14.92.1 Detailed Description	956
14.93L4::lpc::Msg::Cls_buffer Struct Reference	956
14.93.1 Detailed Description	957
14.94L4::lpc::Msg::Cls_data Struct Reference	958

14.94.1 Detailed Description	958
14.95L4::lpc::Msg::Cls_item Struct Reference	959
14.95.1 Detailed Description	959
14.96L4::lpc::Msg::Dir_in Struct Reference	960
14.96.1 Detailed Description	960
14.97L4::lpc::Msg::Dir_out Struct Reference	961
14.97.1 Detailed Description	961
14.98L4::lpc::Msg::Do_in_data Struct Reference	962
14.98.1 Detailed Description	962
14.99L4::lpc::Msg::Do_in_items Struct Reference	963
14.99.1 Detailed Description	963
14.100L4::lpc::Msg::Do_out_data Struct Reference	964
14.100.1 Detailed Description	965
14.101L4::lpc::Msg::Do_out_items Struct Reference	965
14.101.1 Detailed Description	966
14.102L4::lpc::Msg::Do_rcv_buffers Struct Reference	966
14.102.1 Detailed Description	967
14.103L4::lpc::Msg::Elem< Array< A, LEN > &> Struct Template Reference	968
14.103.1 Detailed Description	968
14.104L4::lpc::Msg::Elem< Array< A, LEN > > Struct Template Reference	969
14.104.1 Detailed Description	969
14.105L4::lpc::Msg::Elem< Array_ref< A, LEN > &> Struct Template Reference	970
14.105.1 Detailed Description	970
14.106L4::lpc::Msg::Is_valid_rpc_type< T > Struct Template Reference	971
14.106.1 Detailed Description	972
14.107L4::lpc::Msg::Svr_arg_pack< IPC_TYPE > Struct Template Reference	973
14.107.1 Detailed Description	973
14.108L4::lpc::Msg::Svr_val_ops< MTYPE, DIR, CLASS > Struct Template Reference	973
14.108.1 Detailed Description	974
14.109L4::lpc::Msg_ptr< T > Class Template Reference	974

14.109.1	Detailed Description	975
14.109.2	Constructor & Destructor Documentation	975
14.109.2.1	Msg_ptr()	975
14.110.1	ipc::Opt< T > Struct Template Reference	975
14.110.1	Detailed Description	977
14.111.1	ipc::Ostream Class Reference	977
14.111.1	Detailed Description	980
14.111.2	Member Function Documentation	980
14.111.2.1	put() [1/2]	980
14.111.2.2	put() [2/2]	981
14.111.2.3	send()	981
14.111.2.4	tag() [1/2]	982
14.111.2.5	tag() [2/2]	983
14.112.1	ipc::Out< T > Struct Template Reference	983
14.112.1	Detailed Description	983
14.113.1	ipc::Ret_array< T > Struct Template Reference	984
14.113.1	Detailed Description	984
14.114.1	ipc::Send_only Struct Reference	985
14.114.1	Detailed Description	985
14.115.1	ipc::Small_buf Class Reference	985
14.115.1	Detailed Description	986
14.115.2	Constructor & Destructor Documentation	986
14.115.2.1	Small_buf() [1/2]	986
14.115.2.2	Small_buf() [2/2]	986
14.116.1	ipc::Snd_item Class Reference	987
14.116.1	Detailed Description	987
14.117.1	ipc::Str_cp_in< T > Class Template Reference	987
14.117.1	Detailed Description	988
14.117.2	Constructor & Destructor Documentation	988
14.117.2.1	Str_cp_in()	988

14.1184::lpc::Varg Class Reference	989
14.118.1Detailed Description	990
14.118.2Member Function Documentation	990
14.118.2.1data()	990
14.118.2.2get_value()	990
14.118.2.3s_nil()	991
14.118.2.4s_of()	991
14.118.2.5s_of_int()	991
14.118.2.6length()	992
14.118.2.7tag()	992
14.118.2.8type()	993
14.118.2.9value()	993
14.1194::lpc::Varg_list< MAX > Class Template Reference	994
14.119.1Detailed Description	995
14.1204::lpc::Varg_list_ref Class Reference	995
14.120.1Detailed Description	996
14.120.2Constructor & Destructor Documentation	996
14.120.2.1Varg_list_ref()	996
14.1214::lpc_gate Class Reference	997
14.121.1Detailed Description	998
14.121.2Member Function Documentation	999
14.121.2.1bind_thread()	999
14.121.2.2get_infos()	1000
14.1224::lpc_svr::Br_manager_no_buffers Class Reference	1000
14.122.1Detailed Description	1003
14.122.2Member Function Documentation	1003
14.122.2.1alloc_buffer_demand()	1003
14.1234::lpc_svr::Compound_reply Struct Reference	1004
14.123.1Detailed Description	1004
14.1244::lpc_svr::Default_loop_hooks Struct Reference	1005

14.124. Detailed Description	1006
14.125. <code>lpc_svr::Default_setup_wait</code> Struct Reference	1006
14.125. Detailed Description	1007
14.126. <code>lpc_svr::Default_timeout</code> Struct Reference	1007
14.126. Detailed Description	1008
14.127. <code>lpc_svr::Direct_dispatch< R ></code> Struct Template Reference	1008
14.127. Detailed Description	1009
14.128. <code>lpc_svr::Direct_dispatch< R * ></code> Struct Template Reference	1010
14.128. Detailed Description	1010
14.129. <code>lpc_svr::Exc_dispatch< R, Exc ></code> Struct Template Reference	1011
14.129. Detailed Description	1012
14.130. <code>lpc_svr::Ignore_errors</code> Struct Reference	1012
14.130. Detailed Description	1013
14.131. <code>lpc_svr::Server_iface</code> Class Reference	1013
14.131. Detailed Description	1016
14.131. Member Function Documentation	1016
14.131.2.1 <code>add_timeout()</code>	1016
14.131.2.2 <code>alloc_buffer_demand()</code>	1017
14.131.2.3 <code>get_rcv_cap()</code>	1017
14.131.2.4 <code>rcv_cap()</code> [1/2]	1018
14.131.2.5 <code>rcv_cap()</code> [2/2]	1019
14.131.2.6 <code>realloc_rcv_cap()</code>	1020
14.131.2.7 <code>remove_timeout()</code>	1021
14.132. <code>lpc_svr::Timed_work< HOOKS ></code> Class Template Reference	1022
14.132. Detailed Description	1022
14.133. <code>lpc_svr::Timeout</code> Class Reference	1022
14.133. Detailed Description	1024
14.133. Member Function Documentation	1024
14.133.2.1 <code>expired()</code>	1024
14.133.2.2 <code>timeout()</code>	1025

14.134.1:ipc_srv::Timeout_queue Class Reference	1025
14.134.1.1 Detailed Description	1026
14.134.1.2 Member Function Documentation	1026
14.134.1.2.1 add()	1026
14.134.1.2.2 handle_expired_timeouts()	1027
14.134.1.2.3 next_timeout()	1027
14.134.1.2.4 remove()	1027
14.134.1.2.5 timeout_expired()	1028
14.135:ipc_srv::Timeout_queue_hooks< HOOKS, BR_MAN > Class Template Reference	1029
14.135.1 Detailed Description	1031
14.135.2 Member Function Documentation	1031
14.135.2.1 add_timeout()	1031
14.135.2.2 remove_timeout()	1032
14.136:Irq Class Reference	1032
14.136.1 Detailed Description	1035
14.136.2 Member Function Documentation	1035
14.136.2.1 attach()	1035
14.136.2.2 detach()	1036
14.136.2.3 receive()	1037
14.136.2.4 unmask()	1038
14.136.2.5 wait()	1039
14.137:Irq_eoi Class Reference	1040
14.137.1 Detailed Description	1041
14.137.2 Member Function Documentation	1041
14.137.2.1 unmask()	1041
14.138:Irq_handler_object Struct Reference	1042
14.138.1 Detailed Description	1044
14.139:Irq_mux Struct Reference	1045
14.139.1 Detailed Description	1047
14.139.2 Member Function Documentation	1047

14.139.2.1chain()	1047
14.140.1Kip::Mem_desc Class Reference	1048
14.140.1Detailed Description	1049
14.140.2Member Enumeration Documentation	1050
14.140.2.1Info_sub_type	1050
14.140.2.2Mem_type	1050
14.140.3Constructor & Destructor Documentation	1050
14.140.3.1Mem_desc()	1050
14.140.4Member Function Documentation	1051
14.140.4.1all() [1/2]	1051
14.140.4.2all() [2/2]	1052
14.140.4.3count() [1/2]	1052
14.140.4.4count() [2/2]	1053
14.140.4.5end()	1054
14.140.4.6first()	1054
14.140.4.7is_virtual()	1055
14.140.4.8set()	1055
14.140.4.9size()	1056
14.140.4.10start()	1056
14.140.4.11sub_type()	1057
14.140.4.12type()	1057
14.141.1Kobject Class Reference	1057
14.141.1Detailed Description	1058
14.141.2Member Function Documentation	1058
14.141.2.1cap()	1058
14.141.2.2dec_refcnt()	1060
14.142.1Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND > Class Template Reference	1061
14.142.1Detailed Description	1063
14.142.2Member Typedef Documentation	1064
14.142.2.1__iface	1064

14.142.2.2__lface_list	1064
14.142.2.3Class	1065
14.142.3Member Function Documentation	1065
14.142.3.1__check_protocols__()	1065
14.142.3.2()	1065
14.143::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND > Struct Template Reference	1066
14.143.1Detailed Description	1067
14.143.2Member Typedef Documentation	1068
14.143.2.1__lface	1068
14.143.2.2__lface_list	1068
14.143.2.3Class	1069
14.143.3Member Function Documentation	1069
14.143.3.1__check_protocols__()	1069
14.143.3.2()	1069
14.144::Kobject_demand< T > Struct Template Reference	1069
14.144.1Detailed Description	1070
14.145::Kobject_t< Derived, Base, PROTO, S_DEMAND > Class Template Reference	1070
14.145.1Detailed Description	1071
14.146::Kobject_typeid< T > Struct Template Reference	1072
14.146.1Detailed Description	1072
14.146.2Member Typedef Documentation	1073
14.146.2.1Demand	1073
14.146.3Member Function Documentation	1073
14.146.3.1demand()	1073
14.146.3.2d()	1074
14.146.3.3proto_dispatch()	1074
14.147::Kobject_x< Derived, ARGS > Struct Template Reference	1075
14.147.1Detailed Description	1076
14.148::Meta Class Reference	1077
14.148.1Detailed Description	1079

14.148.2Member Function Documentation	1080
14.148.2.1interface()	1080
14.148.2.2num_interfaces()	1080
14.148.2.3supports()	1080
14.1494::Out_of_memory Class Reference	1081
14.149.1Detailed Description	1084
14.1504::Pager Class Reference	1084
14.150.1Detailed Description	1086
14.150.2Member Function Documentation	1086
14.150.2.1page_fault()	1087
14.1514::Platform_control Class Reference	1087
14.151.1Detailed Description	1090
14.151.2Member Enumeration Documentation	1090
14.151.2.1Opcode	1090
14.151.3Member Function Documentation	1090
14.151.3.1cpu_disable()	1090
14.151.3.2cpu_enable()	1091
14.151.3.3system_shutdown()	1091
14.151.3.4system_suspend()	1091
14.1524::Poll_timeout_kipclock Class Reference	1092
14.152.1Detailed Description	1093
14.152.2Constructor & Destructor Documentation	1093
14.152.2.1Poll_timeout_kipclock()	1093
14.152.3Member Function Documentation	1093
14.152.3.1set()	1093
14.152.3.2test()	1094
14.152.3.3timed_out()	1095
14.1534::Proto_t< P > Struct Template Reference	1095
14.153.1Detailed Description	1096
14.1544::Registry_iface Class Reference	1097

14.154.1Detailed Description	1098
14.154.2Member Function Documentation	1098
14.154.2.1register_irq_obj() [1/2]	1098
14.154.2.2register_irq_obj() [2/2]	1099
14.154.2.3register_obj() [1/2]	1100
14.154.2.4register_obj() [2/2]	1100
14.154.2.5unregister_obj()	1101
14.154.4::Runtime_error Class Reference	1101
14.155.1Detailed Description	1103
14.155.2Constructor & Destructor Documentation	1104
14.155.2.1Runtime_error()	1104
14.155.3Member Function Documentation	1104
14.155.3.1err_no()	1104
14.155.3.2extra_str()	1104
14.156.4::Scheduler Class Reference	1105
14.156.1Detailed Description	1107
14.156.2Member Function Documentation	1107
14.156.2.1idle_time()	1107
14.156.2.2info()	1108
14.156.2.3is_online()	1109
14.156.2.4run_thread()	1109
14.157.4::Semaphore Struct Reference	1110
14.157.1Detailed Description	1113
14.157.2Member Function Documentation	1113
14.157.2.1down()	1113
14.157.2.2up()	1114
14.158.4::Server< LOOP_HOOKS > Class Template Reference	1115
14.158.1Detailed Description	1116
14.158.2Constructor & Destructor Documentation	1116
14.158.2.1Server()	1116

14.158.3	Member Function Documentation	1117
14.158.3.1	internal_loop()	1117
14.158.3.2	loop()	1118
14.159.4	Server_object Class Reference	1118
14.159.1	Detailed Description	1119
14.159.2	Member Function Documentation	1119
14.159.2.1	dispatch()	1119
14.160.4	Server_object_t< IFACE, BASE > Struct Template Reference	1120
14.160.1	Detailed Description	1122
14.160.2	Member Function Documentation	1122
14.160.2.1	dispatch_meta_request()	1122
14.160.2.2	get_buffer_demand()	1123
14.160.2.3	proto_dispatch()	1123
14.161.4	Server_object_x< Derived, IFACE, BASE > Struct Template Reference	1124
14.161.1	Detailed Description	1125
14.162.4	Smart_cap< T, SMART > Class Template Reference	1126
14.162.1	Detailed Description	1129
14.162.2	Constructor & Destructor Documentation	1129
14.162.2.1	Smart_cap()	1129
14.163.4	String Class Reference	1130
14.163.1	Detailed Description	1130
14.164.4	Task Class Reference	1130
14.164.1	Detailed Description	1133
14.164.2	Member Function Documentation	1133
14.164.2.1	add_ku_mem()	1133
14.164.2.2	cap_equal()	1134
14.164.2.3	cap_has_child()	1134
14.164.2.4	cap_valid()	1135
14.164.2.5	delete_obj()	1135
14.164.2.6	map()	1136

14.164.2.7	release_cap()	1136
14.164.2.8	unmap()	1137
14.164.2.9	unmap_batch()	1138
14.165.4	Thread Class Reference	1138
14.165.1	Detailed Description	1141
14.165.2	Member Function Documentation	1142
14.165.2.1	control()	1142
14.165.2.2	ex_regs() [1/2]	1142
14.165.2.3	ex_regs() [2/2]	1143
14.165.2.4	modify_senders()	1144
14.165.2.5	register_del_irq()	1145
14.165.2.6	stats_time()	1145
14.165.2.7	switch_to()	1146
14.165.2.8	cpu_control()	1146
14.165.2.9	cpu_control_ext()	1147
14.165.2.10	cpu_resume_commit()	1148
14.165.2.11	cpu_resume_start()	1148
14.166.4	Thread::Attr Class Reference	1149
14.166.1	Detailed Description	1150
14.166.2	Constructor & Destructor Documentation	1150
14.166.2.1	Attr()	1150
14.166.3	Member Function Documentation	1150
14.166.3.1	bind()	1150
14.166.3.2	exc_handler() [1/2]	1151
14.166.3.3	exc_handler() [2/2]	1151
14.166.3.4	pager() [1/2]	1151
14.166.3.5	pager() [2/2]	1152
14.166.3.6	ix_host_syscall()	1152
14.167.4	Thread::Modify_senders Class Reference	1152
14.167.1	Detailed Description	1153

14.167.2Member Function Documentation	1153
14.167.2.1add()	1153
14.1684::Triggerable Struct Reference	1154
14.168.1Detailed Description	1156
14.168.2Member Function Documentation	1156
14.168.2.1trigger()	1157
14.1694::Type_info Struct Reference	1158
14.169.1Detailed Description	1158
14.1704::Type_info::Demand Class Reference	1159
14.170.1Detailed Description	1160
14.170.2Constructor & Destructor Documentation	1160
14.170.2.1Demand()	1161
14.170.3Member Function Documentation	1161
14.170.3.1no_demand()	1161
14.1714::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS > Struct Template Reference	1162
14.171.1Detailed Description	1163
14.171.2Member Enumeration Documentation	1164
14.171.2.1anonymous enum	1164
14.1724::Type_info::Demand_union_t< D1, D2 > Struct Template Reference	1164
14.172.1Detailed Description	1166
14.1734::Typeid::Detail::_Rpc< OPCODE, O, X > Struct Template Reference	1167
14.173.1Detailed Description	1168
14.1744::Typeid::Detail::_Rpc< OPCODE, O, Default_op< R > >::Rpc< Y > Struct Template Reference	1168
14.174.1Detailed Description	1169
14.1754::Typeid::Detail::_Rpc< OPCODE, O, R, X... > Struct Template Reference	1169
14.175.1Detailed Description	1170
14.1764::Typeid::Detail::_Rpc< OPCODE, O, R, X... >::Rpc< Y > Struct Template Reference	1170
14.176.1Detailed Description	1171
14.1774::Typeid::Detail::Rpc_end Struct Reference	1171
14.177.1Detailed Description	1172

14.1784::Typeid::P_dispatch< LIST > Struct Template Reference	1172
14.178.1Detailed Description	1172
14.1794::Typeid::Raw_ipc< CLASS > Struct Template Reference	1173
14.179.1Detailed Description	1173
14.1804::Typeid::Rpc_nocode< OPERATION > Struct Template Reference	1173
14.180.1Detailed Description	1175
14.1814::Typeid::Rpc< RPCS > Struct Template Reference	1176
14.181.1Detailed Description	1177
14.1824::Typeid::Rpc_code< OPCODE_TYPE > Struct Template Reference	1178
14.182.1Detailed Description	1178
14.1834::Typeid::Rpc_code< OPCODE_TYPE >::F< RPCS > Struct Template Reference	1179
14.183.1Detailed Description	1180
14.1844::Typeid::Rpc_sys< ARG > Struct Template Reference	1181
14.184.1Detailed Description	1182
14.1854::Types::Bool< V > Struct Template Reference	1183
14.185.1Detailed Description	1184
14.1864::Types::False Struct Reference	1184
14.186.1Detailed Description	1186
14.1874::Types::Flags< BITS_ENUM, UNDERLYING > Class Template Reference	1186
14.187.1Detailed Description	1188
14.187.2Member Enumeration Documentation	1189
14.187.2.1None_type	1189
14.187.3Constructor & Destructor Documentation	1189
14.187.3.1Flags() [1/2]	1189
14.187.3.2Flags() [2/2]	1190
14.187.4Member Function Documentation	1190
14.187.4.1clear()	1190
14.187.4.2from_raw()	1191
14.1884::Types::Same< A, B > Struct Template Reference	1191
14.188.1Detailed Description	1192

14.189.4::Types::True Struct Reference	1193
14.189.4.1 Detailed Description	1195
14.190.4::Unknown_error Class Reference	1195
14.190.4.1 Detailed Description	1197
14.191.4::Vcon Class Reference	1198
14.191.4.1 Detailed Description	1200
14.191.4.2 Member Function Documentation	1200
14.191.4.2.1 get_attr()	1200
14.191.4.2.2 read()	1201
14.191.4.2.3 read_with_flags()	1202
14.191.4.2.4 send()	1203
14.191.4.2.5 set_attr()	1203
14.191.4.2.6 write()	1204
14.192.4::Vm Class Reference	1205
14.192.4.1 Detailed Description	1207
14.193.4::buf_regs_t Struct Reference	1208
14.193.4.1 Detailed Description	1208
14.194.4::exc_regs_t Struct Reference	1209
14.194.4.1 Detailed Description	1211
14.194.4.2 Field Documentation	1211
14.194.4.2.1 flags	1211
14.194.4.2.2 ss	1211
14.195.4::fpage_t Union Reference	1212
14.195.4.1 Detailed Description	1212
14.196.4::icu_info_t Struct Reference	1212
14.196.4.1 Detailed Description	1214
14.196.4.2 Field Documentation	1214
14.196.4.2.1 features	1214
14.197.4::icu_msi_info_t Struct Reference	1214
14.197.4.1 Detailed Description	1215

14.197.2Field Documentation	1215
14.197.2.1msi_data	1215
14.1981_kernel_info_mem_desc_t Struct Reference	1215
14.198.1Detailed Description	1216
14.1991_msg_regs_t Union Reference	1216
14.199.1Detailed Description	1216
14.2001_msgtag_t Struct Reference	1217
14.200.1Detailed Description	1218
14.200.2Member Function Documentation	1218
14.200.2.1flags()	1218
14.2011_sched_cpu_set_t Struct Reference	1219
14.201.1Detailed Description	1219
14.201.2Member Function Documentation	1219
14.201.2.1granularity()	1220
14.201.2.2offset()	1220
14.201.3Field Documentation	1220
14.201.3.1gran_offset	1221
14.2021_sched_param_t Struct Reference	1221
14.202.1Detailed Description	1222
14.2031_snd_fpage_t Struct Reference	1222
14.203.1Detailed Description	1223
14.2041_thread_regs_t Struct Reference	1223
14.204.1Detailed Description	1224
14.2051_timeout_s Struct Reference	1224
14.205.1Detailed Description	1225
14.2061_timeout_t Union Reference	1225
14.206.1Detailed Description	1226
14.2071_tracebuffer_status_t Struct Reference	1226
14.207.1Detailed Description	1227
14.207.2Field Documentation	1227

14.207.2.1cnt_jobmap_tlb_flush	1228
14.208.1_tracebuffer_status_window_t Struct Reference	1228
14.208.1Detailed Description	1229
14.209.1_vcon_attr_t Struct Reference	1229
14.209.1Detailed Description	1229
14.210.1_vcpu_ipc_regs_t Struct Reference	1230
14.210.1Detailed Description	1230
14.211.1_vcpu_regs_t Struct Reference	1231
14.211.1Detailed Description	1232
14.211.2Field Documentation	1232
14.211.2.1ax	1233
14.211.2.2bp	1233
14.211.2.3bx	1233
14.211.2.4cx	1233
14.211.2.5di	1234
14.211.2.6dx	1234
14.211.2.7si	1234
14.212.1_vcpu_state_t Struct Reference	1235
14.212.1Detailed Description	1237
14.213.1_vhw_descriptor Struct Reference	1237
14.213.1Detailed Description	1238
14.213.2Field Documentation	1239
14.213.2.1count	1239
14.213.2.2descs	1239
14.213.2.3magic	1239
14.213.2.4version	1240
14.214.1_vhw_entry Struct Reference	1240
14.214.1Detailed Description	1241
14.214.2Field Documentation	1241
14.214.2.1fd	1241

14.214.2.2	<code>rq_no</code>	1241
14.214.2.3	<code>mem_size</code>	1241
14.214.2.4	<code>mem_start</code>	1242
14.214.2.5	<code>provider_pid</code>	1242
14.214.2.6	<code>type</code>	1242
14.215	<code>vm_svm_vmcb_control_area</code> Struct Reference	1243
14.215.1	Detailed Description	1243
14.216	<code>vm_svm_vmcb_state_save_area</code> Struct Reference	1243
14.216.1	Detailed Description	1244
14.217	<code>vm_svm_vmcb_state_save_area_seg</code> Struct Reference	1245
14.217.1	Detailed Description	1245
14.218	<code>vm_svm_vmcb_t</code> Struct Reference	1245
14.218.1	Detailed Description	1246
14.219	<code>vm_tz_state</code> Struct Reference	1247
14.219.1	Detailed Description	1247
14.220	<code>Re::Cap_alloc</code> Class Reference	1247
14.220.1	Detailed Description	1248
14.220.2	Member Function Documentation	1248
14.220.2.1	<code>alloc()</code> [1/2]	1249
14.220.2.2	<code>alloc()</code> [2/2]	1249
14.220.2.3	<code>free()</code>	1250
14.220.2.4	<code>get_cap_alloc()</code>	1250
14.221	<code>Re::Console</code> Class Reference	1251
14.221.1	Detailed Description	1253
14.222	<code>Re::Dataspace</code> Class Reference	1254
14.222.1	Detailed Description	1256
14.222.2	Member Enumeration Documentation	1256
14.222.2.1	<code>Map_flags</code>	1256
14.222.3	Member Function Documentation	1257
14.222.3.1	<code>allocate()</code>	1257

14.222.3.2clear()	1257
14.222.3.3copy_in()	1258
14.222.3.4flags()	1258
14.222.3.5info()	1260
14.222.3.6map()	1260
14.222.3.7map_region()	1261
14.222.3.8phys()	1262
14.222.3.9size()	1263
14.224Re::Dataspace::Stats Struct Reference	1263
14.223.1Detailed Description	1264
14.224Re::Debug_obj Class Reference	1264
14.224.1Detailed Description	1265
14.224.2Member Function Documentation	1266
14.224.2.1debug()	1266
14.225Re::Dma_space Class Reference	1266
14.225.1Detailed Description	1267
14.225.2Member Typedef Documentation	1267
14.225.2.1Attributes	1268
14.225.3Member Enumeration Documentation	1268
14.225.3.1Attribute	1268
14.225.3.2Direction	1268
14.225.3.3Space_attrib	1269
14.225.4Member Function Documentation	1269
14.225.4.1associate()	1269
14.225.4.2disassociate()	1270
14.225.4.3map()	1270
14.225.4.4unmap()	1271
14.226Re::Env Class Reference	1271
14.226.1Detailed Description	1273
14.226.2Member Function Documentation	1274

14.226.2.1env()	1274
14.226.2.2factory() [1/2]	1275
14.226.2.3factory() [2/2]	1275
14.226.2.4first_free_cap() [1/2]	1275
14.226.2.5first_free_cap() [2/2]	1275
14.226.2.6first_free_utcb() [1/2]	1276
14.226.2.7first_free_utcb() [2/2]	1276
14.226.2.8get()	1276
14.226.2.9get_cap() [1/2]	1277
14.226.2.10get_cap() [2/2]	1278
14.226.2.11initial_caps() [1/2]	1278
14.226.2.12initial_caps() [2/2]	1279
14.226.2.13log() [1/2]	1279
14.226.2.14log() [2/2]	1279
14.226.2.15main_thread() [1/2]	1280
14.226.2.16main_thread() [2/2]	1280
14.226.2.17mem_alloc() [1/2]	1280
14.226.2.18mem_alloc() [2/2]	1281
14.226.2.19parent() [1/2]	1282
14.226.2.20parent() [2/2]	1282
14.226.2.21m() [1/2]	1282
14.226.2.22m() [2/2]	1283
14.226.2.23scheduler() [1/2]	1283
14.226.2.24scheduler() [2/2]	1283
14.226.2.25ask()	1284
14.226.2.26utcb_area() [1/2]	1284
14.226.2.27utcb_area() [2/2]	1284
14.227.1Event Class Reference	1285
14.227.2Detailed Description	1288
14.227.3Member Function Documentation	1288

14.227.2.1get_buffer()	1288
14.2284Re::Event_buffer_t< PAYLOAD > Class Template Reference	1289
14.228.1Detailed Description	1290
14.228.2Constructor & Destructor Documentation	1290
14.228.2.1Event_buffer_t()	1290
14.228.3Member Function Documentation	1291
14.228.3.1next()	1291
14.228.3.2put()	1291
14.2294Re::Event_buffer_t< PAYLOAD >::Event Struct Reference	1292
14.229.1Detailed Description	1293
14.2304Re::Inhibitor Class Reference	1293
14.230.1Detailed Description	1296
14.230.2Member Enumeration Documentation	1296
14.230.2.1anonymous enum	1296
14.230.3Member Function Documentation	1296
14.230.3.1acquire()	1296
14.230.3.2next_lock_info()	1297
14.230.3.3release()	1298
14.2314Re::Log Class Reference	1298
14.231.1Detailed Description	1301
14.231.2Member Function Documentation	1301
14.231.2.1print()	1301
14.231.2.2println()	1301
14.2324Re::Mem_alloc Class Reference	1301
14.232.1Detailed Description	1304
14.232.2Member Enumeration Documentation	1304
14.232.2.1Mem_alloc_flags	1304
14.232.3Member Function Documentation	1305
14.232.3.1alloc()	1305
14.232.3.2free()	1306

14.234Re::Mmio_space Struct Reference	1306
14.233.1Detailed Description	1309
14.233.2Member Enumeration Documentation	1309
14.233.2.1Access_width	1309
14.233.3Member Function Documentation	1309
14.233.3.1mmio_read()	1309
14.233.3.2mmio_write()	1310
14.234Re::Namespace Class Reference	1310
14.234.1Detailed Description	1313
14.234.2Member Enumeration Documentation	1313
14.234.2.1Query_result_flags	1313
14.234.2.2Register_flags	1313
14.234.3Member Function Documentation	1314
14.234.3.1query() [1/2]	1314
14.234.3.2query() [2/2]	1315
14.234.3.3register_obj()	1316
14.234.3.4unlink()	1316
14.235Re::Ned::Cmd_control Class Reference	1317
14.235.1Detailed Description	1318
14.235.2Member Function Documentation	1318
14.235.2.1execute() [1/2]	1318
14.235.2.2execute() [2/2]	1318
14.236Re::Parent Class Reference	1319
14.236.1Detailed Description	1320
14.236.2Member Function Documentation	1321
14.236.2.1signal()	1321
14.237Re::Rm Class Reference	1321
14.237.1Detailed Description	1324
14.237.2Member Enumeration Documentation	1325
14.237.2.1Attach_flags	1325

14.237.2.2	Detach_flags	1325
14.237.2.3	Detach_result	1325
14.237.2.4	Region_flags	1326
14.237.3	Member Function Documentation	1326
14.237.3.1	attach() [1/2]	1326
14.237.3.2	attach() [2/2]	1328
14.237.3.3	detach() [1/3]	1329
14.237.3.4	detach() [2/3]	1329
14.237.3.5	detach() [3/3]	1330
14.237.3.6	find()	1331
14.237.3.7	free_area()	1332
14.237.3.8	reserve_area() [1/2]	1332
14.237.3.9	reserve_area() [2/2]	1333
14.238	Re::Smart_cap_auto< Unmap_flags > Class Template Reference	1334
14.238.1	Detailed Description	1334
14.239	Re::Util::Auto_cap< T > Struct Template Reference	1335
14.239.1	Detailed Description	1335
14.240	Re::Util::Auto_del_cap< T > Struct Template Reference	1336
14.240.1	Detailed Description	1337
14.241	Re::Util::Br_manager Class Reference	1338
14.241.1	Detailed Description	1340
14.241.2	Member Function Documentation	1340
14.241.2.1	lalloc_buffer_demand()	1340
14.241.2.2	get_rcv_cap()	1340
14.241.2.3	realloc_rcv_cap()	1341
14.241.2.4	set_rcv_cap_flags()	1342
14.242	Re::Util::Br_manager_hooks Struct Reference	1342
14.242.1	Detailed Description	1343
14.243	Re::Util::Br_manager_timeout_hooks Struct Reference	1343
14.243.1	Detailed Description	1345

14.244.1	L4Re::Util::Cap_alloc_base Class Reference	1346
14.244.1	Detailed Description	1346
14.245.1	L4Re::Util::Counting_cap_alloc< COUNTERTYPE > Class Template Reference	1347
14.245.1	Detailed Description	1348
14.245.2	Constructor & Destructor Documentation	1348
14.245.2.1	Counting_cap_alloc()	1348
14.245.3	Member Function Documentation	1348
14.245.3.1	alloc() [1/2]	1348
14.245.3.2	alloc() [2/2]	1349
14.245.3.3	free()	1349
14.245.3.4	release()	1350
14.245.3.5	setup()	1351
14.245.3.6	take()	1352
14.246.1	L4Re::Util::Dataspace_svr Class Reference	1352
14.246.1	Detailed Description	1354
14.246.2	Member Function Documentation	1354
14.246.2.1	allocate()	1354
14.246.2.2	clear()	1355
14.246.2.3	copy()	1356
14.246.2.4	is_static()	1357
14.246.2.5	map()	1358
14.246.2.6	map_hook()	1358
14.246.2.7	page_shift()	1359
14.246.2.8	phys()	1360
14.246.2.9	release()	1361
14.246.2.10	take()	1362
14.247.1	L4Re::Util::Event_buffer_consumer_t< PAYLOAD > Class Template Reference	1362
14.247.1	Detailed Description	1364
14.247.2	Member Function Documentation	1365
14.247.2.1	foreach_available_event()	1365

14.247.2.2process()	1365
14.2484Re::Util::Event_buffer_t< PAYLOAD > Class Template Reference	1366
14.248.1Detailed Description	1368
14.248.2Member Function Documentation	1368
14.248.2.1attach()	1369
14.248.2.2buf()	1369
14.248.2.3detach()	1369
14.2494Re::Util::Event_t< PAYLOAD > Class Template Reference	1370
14.249.1Detailed Description	1371
14.249.2Member Enumeration Documentation	1371
14.249.2.1Mode	1371
14.249.3Member Function Documentation	1371
14.249.3.1buffer()	1371
14.249.3.2nit()	1372
14.249.3.3nit_poll()	1373
14.249.3.4irq()	1373
14.2504Re::Util::Item_alloc_base Class Reference	1374
14.250.1Detailed Description	1374
14.2514Re::Util::Names::Name Class Reference	1375
14.251.1Detailed Description	1376
14.2524Re::Util::Names::Name_space Class Reference	1376
14.252.1Detailed Description	1377
14.252.2Member Function Documentation	1378
14.252.2.1alloc_dynamic_entry()	1378
14.252.2.2copy_receive_cap()	1378
14.252.2.3free_capability()	1378
14.252.2.4free_dynamic_entry()	1379
14.252.2.5free_epiface()	1379
14.252.2.6get_epiface()	1379
14.2534Re::Util::Object_registry Class Reference	1380

14.253.1Detailed Description	1382
14.253.2Constructor & Destructor Documentation	1382
14.253.2.1Object_registry() [1/2]	1382
14.253.2.2Object_registry() [2/2]	1382
14.253.3Member Function Documentation	1383
14.253.3.1register_irq_obj() [1/2]	1383
14.253.3.2register_irq_obj() [2/2]	1384
14.253.3.3register_obj() [1/2]	1384
14.253.3.4register_obj() [2/2]	1385
14.253.3.5unregister_obj()	1385
14.254L4Re::Util::Ref_cap< T > Struct Template Reference	1386
14.254.1Detailed Description	1386
14.255L4Re::Util::Ref_del_cap< T > Struct Template Reference	1387
14.255.1Detailed Description	1387
14.256L4Re::Util::Registry_server< LOOP_HOOKS > Class Template Reference	1388
14.256.1Detailed Description	1390
14.256.2Constructor & Destructor Documentation	1391
14.256.2.1Registry_server() [1/2]	1391
14.256.2.2Registry_server() [2/2]	1391
14.256.3Member Function Documentation	1391
14.256.3.1registry() [1/2]	1392
14.256.3.2registry() [2/2]	1392
14.257L4Re::Util::Smart_cap_auto< Unmap_flags > Class Template Reference	1392
14.257.1Detailed Description	1393
14.258L4Re::Util::Smart_count_cap< Unmap_flags > Class Template Reference	1393
14.258.1Detailed Description	1393
14.259L4Re::Util::Vcon_svr< SVR > Class Template Reference	1394
14.259.1Detailed Description	1394
14.260L4Re::Util::Video::Goos_svr Class Reference	1395
14.260.1Detailed Description	1396

14.260.2Member Function Documentation	1396
14.260.2.1get_fb()	1396
14.260.2.2nit_infos()	1397
14.260.2.3refresh()	1397
14.260.2.4screen_info()	1398
14.260.2.5view_info()	1398
14.261L4Re::Vfs::Be_file Class Reference	1399
14.261.1Detailed Description	1401
14.261.2Member Function Documentation	1402
14.261.2.1data_space()	1402
14.261.2.2stat64()	1402
14.261.2.3unlock_all_locks()	1403
14.262L4Re::Vfs::Be_file_system Class Reference	1403
14.262.1Detailed Description	1404
14.262.2Constructor & Destructor Documentation	1405
14.262.2.1Be_file_system()	1405
14.262.2.2~Be_file_system()	1405
14.262.3Member Function Documentation	1405
14.262.3.1type()	1405
14.263L4Re::Vfs::Directory Class Reference	1406
14.263.1Detailed Description	1407
14.263.2Member Function Documentation	1408
14.263.2.1faccessat()	1408
14.263.2.2link()	1408
14.263.2.3mkdir()	1409
14.263.2.4rename()	1409
14.263.2.5mkdir()	1410
14.263.2.6symlink()	1410
14.263.2.7unlink()	1410
14.264L4Re::Vfs::File Class Reference	1411

14.264.1Detailed Description	1412
14.2654Re::Vfs::File_system Class Reference	1413
14.265.1Detailed Description	1414
14.265.2Member Function Documentation	1414
14.265.2.1mount()	1414
14.265.2.2type()	1415
14.2664Re::Vfs::Fs Class Reference	1415
14.266.1Detailed Description	1417
14.266.2Member Function Documentation	1417
14.266.2.1alloc_fd()	1417
14.266.2.2free_fd()	1418
14.266.2.3get_file()	1418
14.266.2.4mount()	1418
14.266.2.5set_fd()	1419
14.2674Re::Vfs::Generic_file Class Reference	1419
14.267.1Detailed Description	1421
14.267.2Member Function Documentation	1421
14.267.2.1fchmod()	1422
14.267.2.2fstat64()	1422
14.267.2.3get_status_flags()	1422
14.267.2.4set_status_flags()	1422
14.267.2.5unlock_all_locks()	1423
14.2684Re::Vfs::Mman Class Reference	1424
14.268.1Detailed Description	1425
14.2694Re::Vfs::Ops Class Reference	1425
14.269.1Detailed Description	1427
14.2704Re::Vfs::Regular_file Class Reference	1428
14.270.1Detailed Description	1429
14.270.2Member Function Documentation	1430
14.270.2.1data_space()	1430

14.270.2.2	datasync()	1430
14.270.2.3	sync()	1430
14.270.2.4	truncate64()	1430
14.270.2.5	get_lock()	1431
14.270.2.6	seek64()	1431
14.270.2.7	readv()	1432
14.270.2.8	set_lock()	1432
14.270.2.9	writev()	1432
14.271	L4Re::Vfs::Special_file Class Reference	1433
14.271.1	Detailed Description	1435
14.271.2	Member Function Documentation	1435
14.271.2.1	ioctl()	1435
14.272	L4Re::Video::Color_component Class Reference	1436
14.272.1	Detailed Description	1437
14.272.2	Constructor & Destructor Documentation	1437
14.272.2.1	Color_component()	1437
14.272.3	Member Function Documentation	1437
14.272.3.1	dump()	1437
14.272.3.2	get()	1438
14.272.3.3	operator==()	1438
14.272.3.4	set()	1438
14.272.3.5	shift()	1439
14.272.3.6	size()	1439
14.273	L4Re::Video::Goos Class Reference	1440
14.273.1	Detailed Description	1442
14.273.2	Member Enumeration Documentation	1442
14.273.2.1	Flags	1442
14.273.3	Member Function Documentation	1443
14.273.3.1	create_buffer()	1443
14.273.3.2	create_view()	1443

14.273.3.3	<code>delete_buffer()</code>	1444
14.273.3.4	<code>delete_view()</code>	1444
14.273.3.5	<code>get_static_buffer()</code>	1444
14.273.3.6	<code>info()</code>	1445
14.273.3.7	<code>view()</code>	1445
14.274	<code>Re::Video::Goos::Info</code> Struct Reference	1446
14.274.1	Detailed Description	1447
14.274.2	Member Function Documentation	1447
14.274.2.1	<code>auto_refresh()</code>	1447
14.275	<code>Re::Video::Pixel_info</code> Class Reference	1448
14.275.1	Detailed Description	1449
14.275.2	Constructor & Destructor Documentation	1449
14.275.2.1	<code>Pixel_info()</code> [1/2]	1449
14.275.2.2	<code>Pixel_info()</code> [2/2]	1450
14.275.3	Member Function Documentation	1450
14.275.3.1a	<code>()</code> [1/2]	1450
14.275.3.2a	<code>()</code> [2/2]	1450
14.275.3.3b	<code>()</code> [1/2]	1451
14.275.3.4b	<code>()</code> [2/2]	1451
14.275.3.5	<code>bits_per_pixel()</code>	1451
14.275.3.6	<code>bytes_per_pixel()</code> [1/2]	1452
14.275.3.7	<code>bytes_per_pixel()</code> [2/2]	1452
14.275.3.8	<code>dump()</code>	1452
14.275.3.9g	<code>()</code> [1/2]	1453
14.275.3.10	<code>()</code> [2/2]	1453
14.275.3.11	<code>has_alpha()</code>	1454
14.275.3.12	<code>operator==()</code>	1454
14.275.3.13	<code>()</code> [1/2]	1455
14.275.3.14	<code>()</code> [2/2]	1455
14.276	<code>Re::Video::View</code> Class Reference	1455

14.276.1Detailed Description	1457
14.276.2Member Enumeration Documentation	1457
14.276.2.1Flags	1457
14.276.2.2V_flags	1457
14.276.3Member Function Documentation	1458
14.276.3.1info()	1458
14.276.3.2refresh()	1458
14.276.3.3set_info()	1459
14.276.3.4set_viewport()	1459
14.276.3.5stack()	1460
14.277L4Re::Video::View::Info Struct Reference	1460
14.277.1Detailed Description	1462
14.278L4Re_aux_t Struct Reference	1463
14.278.1Detailed Description	1463
14.279L4Re_ds_stats_t Struct Reference	1464
14.279.1Detailed Description	1464
14.280L4Re_elf_aux_mword_t Struct Reference	1464
14.280.1Detailed Description	1465
14.281L4Re_elf_aux_t Struct Reference	1465
14.281.1Detailed Description	1466
14.282L4Re_elf_aux_vma_t Struct Reference	1466
14.282.1Detailed Description	1466
14.283L4Re_env_cap_entry_t Struct Reference	1467
14.283.1Detailed Description	1467
14.283.2Constructor & Destructor Documentation	1467
14.283.2.1L4Re_env_cap_entry_t()	1468
14.283.3Field Documentation	1468
14.283.3.1flags	1468
14.284L4Re_env_t Struct Reference	1469
14.284.1Detailed Description	1470

14.285. L4Re_event_t Struct Reference	1470
14.285.1. Detailed Description	1471
14.286. L4Re_video_color_component_t Struct Reference	1471
14.286.1. Detailed Description	1472
14.287. L4Re_video_goos_info_t Struct Reference	1472
14.287.1. Detailed Description	1473
14.288. L4Re_video_pixel_info_t Struct Reference	1473
14.288.1. Detailed Description	1474
14.289. L4Re_video_view_info_t Struct Reference	1474
14.289.1. Detailed Description	1476
14.290. L4Re_video_view_t Struct Reference	1476
14.290.1. Detailed Description	1476
14.291. L4Util_idt_desc_t Struct Reference	1477
14.291.1. Detailed Description	1477
14.292. L4Util_idt_header_t Struct Reference	1478
14.292.1. Detailed Description	1478
14.293. L4Util_mb_addr_range_t Struct Reference	1479
14.293.1. Detailed Description	1479
14.294. L4Util_mb_apm_t Struct Reference	1480
14.294.1. Detailed Description	1480
14.295. L4Util_mb_drive_t Struct Reference	1480
14.295.1. Detailed Description	1481
14.295.2. Field Documentation	1481
14.295.2.1. drive_cylinders	1481
14.295.2.2. drive_mode	1482
14.295.2.3. drive_number	1482
14.296. L4Util_mb_info_t Struct Reference	1482
14.296.1. Detailed Description	1484
14.297. L4Util_mb_mod_t Struct Reference	1484
14.297.1. Detailed Description	1484

14.297.2	Field Documentation	1485
14.297.2.1	mod_end	1485
14.297.2.2	mod_start	1485
14.298	util_mb_vbe_ctrl_t Struct Reference	1485
14.298.1	Detailed Description	1486
14.299	util_mb_vbe_mode_t Struct Reference	1486
14.299.1	Detailed Description	1488
14.300	4vbus::Device Class Reference	1489
14.300.1	Detailed Description	1491
14.300.2	Member Function Documentation	1491
14.300.2.1	bus_cap()	1491
14.300.2.2	dev_handle()	1492
14.300.2.3	device()	1493
14.300.2.4	device_by_hid()	1494
14.300.2.5	get_resource()	1495
14.300.2.6	is_compatible()	1496
14.300.2.7	next_device()	1496
14.300.2.8	operator"!="()	1497
14.300.2.9	operator=="()	1497
14.300.3	Field Documentation	1497
14.300.3.1	_bus	1498
14.301	4vbus::Gpio_module Class Reference	1498
14.301.1	Detailed Description	1501
14.301.2	Member Function Documentation	1501
14.301.2.1	config_pad()	1501
14.301.2.2	get()	1502
14.301.2.3	pin()	1502
14.301.2.4	set()	1503
14.301.2.5	setup()	1504
14.302	4vbus::Gpio_module::Pin_slice Struct Reference	1504

14.302.1 Detailed Description	1505
14.304vbus::Gpio_pin Class Reference	1505
14.303.1 Detailed Description	1508
14.303.2 Member Function Documentation	1508
14.303.2.1 config_get()	1508
14.303.2.2 config_pad()	1509
14.303.2.3 config_pull()	1509
14.303.2.4 get()	1510
14.303.2.5 pin()	1511
14.303.2.6 set()	1511
14.303.2.7 setup()	1511
14.303.2.8 o_irq()	1512
14.304vbus::Icu Class Reference	1513
14.304.1 Detailed Description	1515
14.305vbus::Pci_dev Class Reference	1515
14.305.1 Detailed Description	1518
14.305.2 Member Function Documentation	1518
14.305.2.1 cfg_read()	1518
14.305.2.2 cfg_write()	1519
14.305.2.3 irq_enable()	1519
14.306vbus::Pci_host_bridge Class Reference	1520
14.306.1 Detailed Description	1523
14.306.2 Member Function Documentation	1523
14.306.2.1 cfg_read()	1523
14.306.2.2 cfg_write()	1524
14.306.2.3 irq_enable()	1525
14.307vbus::Pm< DEC > Class Template Reference	1526
14.307.1 Detailed Description	1527
14.308vbus::Vbus Class Reference	1527
14.308.1 Detailed Description	1530

14.308.2	Member Function Documentation	1530
14.308.2.1	release_resource()	1530
14.308.2.2	request_resource()	1530
14.308.2.3	root()	1531
14.309	ivbus_device_t Struct Reference	1532
14.309.1	Detailed Description	1532
14.310	ivbus_resource_t Struct Reference	1533
14.310.1	Detailed Description	1533
14.311	l4vcpu::State Class Reference	1534
14.311.1	Detailed Description	1534
14.311.2	Constructor & Destructor Documentation	1534
14.311.2.1	State()	1534
14.311.3	Member Function Documentation	1535
14.311.3.1	add()	1535
14.311.3.2	clear()	1535
14.311.3.3	set()	1535
14.312	l4vcpu::Vcpu Class Reference	1536
14.312.1	Detailed Description	1540
14.312.2	Member Function Documentation	1540
14.312.2.1	cast() [1/2]	1540
14.312.2.2	cast() [2/2]	1540
14.312.2.3	entry_ip()	1541
14.312.2.4	entry_sp()	1541
14.312.2.5	ext_alloc()	1542
14.312.2.6	() [1/2]	1542
14.312.2.7	() [2/2]	1542
14.312.2.8	irq_disable_save()	1543
14.312.2.9	irq_enable()	1543
14.312.2.10	irq_restore()	1544
14.312.2.11	irq_entry()	1544

14.312.2.118	page_fault_entry()	1545
14.312.2.119	[1/2]	1546
14.312.2.120	[2/2]	1546
14.312.2.121	saved_state() [1/2]	1546
14.312.2.122	saved_state() [2/2]	1547
14.312.2.123	state() [1/2]	1547
14.312.2.124	state() [2/2]	1547
14.312.2.125	task()	1547
14.312.2.126	wait_for_event()	1548
14.313	virtio::Driver::Virtqueue Class Reference	1549
14.313.1	Detailed Description	1551
14.313.2	Member Function Documentation	1551
14.313.2.1	alloc_descriptor()	1551
14.313.2.2	desc()	1551
14.313.2.3	enqueue_descriptor()	1552
14.313.2.4	find_next_used()	1552
14.313.2.5	free_descriptor()	1552
14.313.2.6	init_queue() [1/2]	1553
14.313.2.7	init_queue() [2/2]	1553
14.313.2.8	initialize_rings()	1554
14.314	virtio::Ptr< T > Class Template Reference	1554
14.314.1	Detailed Description	1556
14.314.2	Member Enumeration Documentation	1556
14.314.2.1	Invalid_type	1556
14.314.3	Member Function Documentation	1556
14.314.3.1	get()	1556
14.314.3.2	is_valid()	1557
14.315	virtio::Svr::Bad_descriptor Struct Reference	1558
14.315.1	Detailed Description	1559
14.315.2	Member Enumeration Documentation	1559

14.315.2.1Error	1559
14.315.3Constructor & Destructor Documentation	1559
14.315.3.1Bad_descriptor()	1559
14.316virtio::Svr::Block_dev< Ds_data > Class Template Reference	1560
14.316.1Detailed Description	1562
14.316.2Constructor & Destructor Documentation	1562
14.316.2.1Block_dev()	1562
14.316.3Member Function Documentation	1563
14.316.3.1finalize_request()	1563
14.316.3.2process_request()	1564
14.316.3.3register_obj()	1564
14.316.3.4set_blk_size()	1565
14.316.3.5set_size_max()	1565
14.316.3.6set_topology()	1566
14.317virtio::Svr::Block_request< Ds_data > Class Template Reference	1566
14.317.1Detailed Description	1567
14.317.2Member Function Documentation	1567
14.317.2.1data_size()	1567
14.317.2.2next_block()	1568
14.318virtio::Svr::Data_buffer Struct Reference	1568
14.318.1Detailed Description	1569
14.318.2Constructor & Destructor Documentation	1570
14.318.2.1Data_buffer()	1570
14.318.3Member Function Documentation	1570
14.318.3.1copy_to()	1570
14.318.3.2done()	1571
14.318.3.3set()	1571
14.318.3.4skip()	1572
14.319virtio::Svr::Dev_config Class Reference	1572
14.319.1Detailed Description	1574

14.319.2	Constructor & Destructor Documentation	1574
14.319.2.1	Dev_config()	1574
14.319.3	Member Function Documentation	1575
14.319.3.1	change_queue_config()	1575
14.319.3.2	ds()	1576
14.319.3.3	get_cmd()	1577
14.319.3.4	hdr()	1578
14.319.3.5	qconfig()	1578
14.319.3.6	reset_cmd()	1579
14.319.3.7	reset_queue()	1580
14.319.3.8	set_failed()	1581
14.319.3.9	set_status()	1582
14.319.3.10	status()	1583
14.320.1	virtio::Svr::Dev_features Struct Reference	1583
14.320.1	Detailed Description	1584
14.320.2	Member Typedef Documentation	1585
14.320.2.1	ring_event_idx_bfm_t	1585
14.320.2.2	ring_indirect_desc_bfm_t	1585
14.320.3	Member Function Documentation	1585
14.320.3.1	ring_event_idx() [1/2]	1585
14.320.3.2	ring_event_idx() [2/2]	1585
14.320.3.3	ring_indirect_desc() [1/2]	1586
14.320.3.4	ring_indirect_desc() [2/2]	1586
14.321.1	virtio::Svr::Dev_status Struct Reference	1586
14.321.1	Detailed Description	1588
14.321.2	Member Typedef Documentation	1588
14.321.2.1	lacked_bfm_t	1588
14.321.2.2	driver_bfm_t	1588
14.321.2.3	driver_ok_bfm_t	1588
14.321.2.4	failed_bfm_t	1588

14.321.2.5feature_ok_bfm_t	1589
14.321.3Member Function Documentation	1589
14.321.3.1acked() [1/2]	1589
14.321.3.2acked() [2/2]	1589
14.321.3.3driver() [1/2]	1590
14.321.3.4driver() [2/2]	1590
14.321.3.5driver_ok() [1/2]	1590
14.321.3.6driver_ok() [2/2]	1590
14.321.3.7ailed() [1/2]	1591
14.321.3.8ailed() [2/2]	1591
14.321.3.9feature_ok() [1/2]	1591
14.321.3.10feature_ok() [2/2]	1592
14.321.3.11running()	1592
14.322virtio::Svr::Device_t< DATA > Class Template Reference	1593
14.322.1Detailed Description	1595
14.322.2Member Function Documentation	1595
14.322.2.1device_error()	1595
14.322.2.2handle_mem_cmd_write()	1595
14.322.2.3nit_mem_info()	1595
14.322.2.4reset_queue_config()	1596
14.322.2.5setup_queue()	1596
14.323virtio::Svr::Driver_mem_list_t< DATA > Class Template Reference	1597
14.323.1Detailed Description	1599
14.323.2Member Function Documentation	1599
14.323.2.1add()	1599
14.323.2.2find()	1600
14.323.2.3full()	1601
14.323.2.4nit()	1601
14.323.2.5oad_desc() [1/3]	1602
14.323.2.6oad_desc() [2/3]	1602

14.323.2.7load_desc() [3/3]	1603
14.323.2.8remove()	1603
14.324virtio::Svr::Driver_mem_region_t< DATA > Class Template Reference	1604
14.324.1Detailed Description	1605
14.324.2Constructor & Destructor Documentation	1606
14.324.2.1Driver_mem_region_t()	1606
14.324.3Member Function Documentation	1606
14.324.3.1contains()	1606
14.324.3.2drv_base()	1607
14.324.3.3ds()	1607
14.324.3.4ds_offset()	1607
14.324.3.5empty()	1608
14.324.3.6flags()	1608
14.324.3.7is_writable()	1608
14.324.3.8local()	1608
14.324.3.9local_base()	1609
14.324.3.10size()	1610
14.325virtio::Svr::Request_processor Class Reference	1610
14.325.1Detailed Description	1611
14.325.2Member Function Documentation	1611
14.325.2.1current_flags()	1611
14.325.2.2has_more()	1612
14.325.2.3next()	1612
14.325.2.4start() [1/2]	1614
14.325.2.5start() [2/2]	1615
14.326virtio::Svr::Virtqueue Class Reference	1615
14.326.1Detailed Description	1618
14.326.2Member Function Documentation	1618
14.326.2.1consumed()	1618
14.326.2.2desc()	1619

14.326.2.3	desc_avail()	1620
14.326.2.4	disable_notify()	1620
14.326.2.5	enable_notify()	1620
14.326.2.6	next_avail()	1621
14.327.1	virtio::Svr::Virtqueue::Head_desc Class Reference	1622
14.327.1	Detailed Description	1622
14.327.2	Member Function Documentation	1622
14.327.2.1	desc()	1623
14.327.2.2	operator Null_ptr_check const *()	1623
14.327.2.3	valid()	1623
14.328.1	virtio::Virtqueue Class Reference	1624
14.328.1	Detailed Description	1627
14.328.2	Member Function Documentation	1627
14.328.2.1	avail_align()	1627
14.328.2.2	avail_size()	1627
14.328.2.3	desc_align()	1628
14.328.2.4	desc_size()	1628
14.328.2.5	disable()	1629
14.328.2.6	dump()	1630
14.328.2.7	get_avail_idx()	1630
14.328.2.8	get_tail_avail_idx()	1631
14.328.2.9	no_notify_guest()	1631
14.328.2.10	no_notify_host() [1/2]	1632
14.328.2.11	no_notify_host() [2/2]	1632
14.328.2.12	um()	1633
14.328.2.13	ready()	1633
14.328.2.14	setup()	1634
14.328.2.15	setup_simple()	1634
14.328.2.16	total_size() [1/2]	1635
14.328.2.17	total_size() [2/2]	1635

14.328.2.1	used_align()	1636
14.328.2.1	used_size()	1636
14.329	virtio::Virtqueue::Avail Class Reference	1637
14.329.1	Detailed Description	1638
14.330	virtio::Virtqueue::Avail::Flags Struct Reference	1638
14.330.1	Detailed Description	1639
14.330.2	Member Typedef Documentation	1639
14.330.2.1	no_irq_bfm_t	1639
14.330.3	Member Function Documentation	1639
14.330.3.1	no_irq() [1/2]	1639
14.330.3.2	no_irq() [2/2]	1640
14.331	virtio::Virtqueue::Desc Class Reference	1640
14.331.1	Detailed Description	1642
14.332	virtio::Virtqueue::Desc::Flags Struct Reference	1642
14.332.1	Detailed Description	1643
14.332.2	Member Typedef Documentation	1643
14.332.2.1	indirect_bfm_t	1644
14.332.2.2	next_bfm_t	1644
14.332.2.3	write_bfm_t	1644
14.332.3	Member Function Documentation	1644
14.332.3.1	indirect() [1/2]	1644
14.332.3.2	indirect() [2/2]	1645
14.332.3.3	next() [1/2]	1645
14.332.3.4	next() [2/2]	1645
14.332.3.5	write() [1/2]	1645
14.332.3.6	write() [2/2]	1646
14.333	virtio::Virtqueue::Used Class Reference	1646
14.333.1	Detailed Description	1647
14.334	virtio::Virtqueue::Used::Flags Struct Reference	1647
14.334.1	Detailed Description	1648

14.334.2	Member Typedef Documentation	1648
14.334.2.1	no_notify_bfm_t	1648
14.334.3	Member Function Documentation	1648
14.334.3.1	no_notify() [1/2]	1649
14.334.3.2	no_notify() [2/2]	1649
14.335	virtio::Virtqueue::Used_elem Struct Reference	1649
14.335.1	Detailed Description	1650
14.335.2	Constructor & Destructor Documentation	1650
14.335.2.1	Used_elem()	1650
14.336	virtio_block_config_t Struct Reference	1650
14.336.1	Detailed Description	1651
14.336.2	Field Documentation	1651
14.336.2.1	blk_size	1651
14.337	virtio_block_header_t Struct Reference	1652
14.337.1	Detailed Description	1652
14.338	virtio_config_hdr_t Struct Reference	1653
14.338.1	Detailed Description	1654
14.338.2	Field Documentation	1654
14.338.2.1	magic	1654
14.338.2.2	status	1654
14.339	virtio_config_queue_t Struct Reference	1655
14.339.1	Detailed Description	1656

15 File Documentation	1657
15.1 amd64/l4/util/apic.h File Reference	1657
15.1.1 Detailed Description	1657
15.2 apic.h	1658
15.3 x86/l4/util/apic.h File Reference	1662
15.3.1 Detailed Description	1663
15.4 apic.h	1663
15.5 amd64/l4/util/idt.h File Reference	1668
15.5.1 Detailed Description	1669
15.6 idt.h	1669
15.7 x86/l4/util/idt.h File Reference	1670
15.7.1 Detailed Description	1670
15.8 idt.h	1671
15.9 amd64/l4/util/perform.h File Reference	1671
15.9.1 Detailed Description	1672
15.10perform.h	1672
15.11x86/l4/util/perform.h File Reference	1677
15.11.1 Detailed Description	1678
15.12perform.h	1678
15.13amd64/l4/util/rdtsc.h File Reference	1683
15.13.1 Detailed Description	1685
15.14rdtsc.h	1685
15.15x86/l4/util/rdtsc.h File Reference	1688
15.15.1 Detailed Description	1689
15.16rdtsc.h	1689
15.17amd64/l4/util/spin.h File Reference	1693
15.17.1 Detailed Description	1693
15.18spin.h	1693
15.19x86/l4/util/spin.h File Reference	1694
15.19.1 Detailed Description	1694

15.20spin.h	1694
15.21amd64/l4/util/util.h File Reference	1695
15.21.1 Detailed Description	1696
15.21.2 Function Documentation	1696
15.21.2.1 l4_sleep()	1696
15.21.2.2 l4util_micros2l4to()	1697
15.22util.h	1697
15.23x86/l4/util/util.h File Reference	1698
15.23.1 Detailed Description	1699
15.23.2 Function Documentation	1699
15.23.2.1 l4_sleep()	1699
15.23.2.2 l4util_micros2l4to()	1699
15.24util.h	1700
15.25amd64/l4f/l4/sys/segment.h File Reference	1700
15.25.1 Detailed Description	1701
15.25.2 Function Documentation	1701
15.25.2.1 fiasco_amd64_set_fs()	1701
15.25.2.2 fiasco_amd64_set_segment_base()	1702
15.26segment.h	1702
15.27amd64/l4/sys/segment.h File Reference	1703
15.27.1 Detailed Description	1705
15.27.2 Enumeration Type Documentation	1705
15.27.2.1 L4_task_ldt_x86_consts	1705
15.27.3 Function Documentation	1705
15.27.3.1 fiasco_amd64_segment_info()	1705
15.27.3.2 fiasco_amd64_set_fs()	1706
15.27.3.3 fiasco_amd64_set_segment_base()	1706
15.28segment.h	1707
15.29x86/l4f/l4/sys/segment.h File Reference	1708
15.29.1 Detailed Description	1709

15.30segment.h	1709
15.31x86/I4/sys/segment.h File Reference	1710
15.31.1 Detailed Description	1711
15.31.2 Enumeration Type Documentation	1711
15.31.2.1 L4_task_ldt_x86_consts	1711
15.32segment.h	1711
15.33amd64/I4f/I4/util/port_io.h File Reference	1712
15.33.1 Detailed Description	1713
15.34port_io.h	1713
15.35amd64/I4/util/port_io.h File Reference	1713
15.35.1 Detailed Description	1714
15.36port_io.h	1714
15.37x86/I4f/I4/util/port_io.h File Reference	1714
15.37.1 Detailed Description	1715
15.37.2 Function Documentation	1716
15.37.2.1 I4util_ioport_map()	1716
15.38port_io.h	1717
15.39x86/I4/util/port_io.h File Reference	1717
15.39.1 Detailed Description	1719
15.40port_io.h	1719
15.41amd64/I4f/I4/util/setjmp.h File Reference	1721
15.41.1 Detailed Description	1722
15.41.2 Function Documentation	1722
15.41.2.1 I4_thread_longjmp()	1722
15.41.2.2 I4_thread_setjmp()	1723
15.42setjmp.h	1723
15.43x86/I4f/I4/util/setjmp.h File Reference	1724
15.43.1 Detailed Description	1725
15.43.2 Function Documentation	1725
15.43.2.1 I4_thread_longjmp()	1725

15.43.2.2 l4_thread_setjmp()	1726
15.44setjmp.h	1726
15.45arm/l4/sys/linkage.h File Reference	1727
15.45.1 Detailed Description	1727
15.46linkage.h	1727
15.47amd64/l4/sys/linkage.h File Reference	1728
15.47.1 Detailed Description	1728
15.48linkage.h	1728
15.49x86/l4/sys/linkage.h File Reference	1728
15.49.1 Detailed Description	1729
15.50linkage.h	1729
15.51arm/l4/sys/mem_op.h File Reference	1729
15.51.1 Detailed Description	1730
15.52mem_op.h	1731
15.53arm/l4/sys/vm.h File Reference	1732
15.53.1 Detailed Description	1732
15.54vm.h	1732
15.55arm/l4/util/bitops_arch.h File Reference	1733
15.55.1 Detailed Description	1733
15.56bitops_arch.h	1734
15.57amd64/l4/util/bitops_arch.h File Reference	1734
15.57.1 Detailed Description	1734
15.58bitops_arch.h	1734
15.59x86/l4/util/bitops_arch.h File Reference	1737
15.59.1 Detailed Description	1738
15.60bitops_arch.h	1738
15.61arm/l4/util/cpu.h File Reference	1741
15.61.1 Detailed Description	1741
15.62cpu.h	1742
15.63amd64/l4/util/cpu.h File Reference	1742

15.63.1 Detailed Description	1743
15.64cpu.h	1743
15.65x86/l4/util/cpu.h File Reference	1744
15.65.1 Detailed Description	1745
15.66cpu.h	1745
15.67arm/l4/util/l4_macros.h File Reference	1746
15.67.1 Detailed Description	1746
15.68l4_macros.h	1746
15.69amd64/l4/util/l4_macros.h File Reference	1747
15.69.1 Detailed Description	1747
15.70l4_macros.h	1747
15.71l4/util/l4_macros.h File Reference	1747
15.71.1 Detailed Description	1748
15.72l4_macros.h	1748
15.73x86/l4/util/l4_macros.h File Reference	1748
15.73.1 Detailed Description	1748
15.74l4_macros.h	1749
15.75arm/l4/util/mbi_argv.h File Reference	1749
15.75.1 Detailed Description	1749
15.76mbi_argv.h	1750
15.77amd64/l4/util/mbi_argv.h File Reference	1750
15.77.1 Detailed Description	1750
15.78mbi_argv.h	1751
15.79x86/l4/util/mbi_argv.h File Reference	1751
15.79.1 Detailed Description	1752
15.80mbi_argv.h	1752
15.81arm/l4/util/stack_impl.h File Reference	1753
15.81.1 Detailed Description	1753
15.81.2 Function Documentation	1753
15.81.2.1 l4util_stack_get_sp()	1753

15.82	stack_impl.h	1754
15.83	amd64/l4/util/stack_impl.h File Reference	1754
15.83.1	Detailed Description	1754
15.83.2	Function Documentation	1754
15.83.2.1	l4util_stack_get_sp()	1754
15.84	stack_impl.h	1755
15.85	x86/l4/util/stack_impl.h File Reference	1755
15.85.1	Detailed Description	1755
15.85.2	Function Documentation	1756
15.85.2.1	l4util_stack_get_sp()	1756
15.86	stack_impl.h	1756
15.87	arm/l4f/l4/sys/syscall_defs.h File Reference	1756
15.87.1	Detailed Description	1757
15.88	syscall_defs.h	1757
15.89	contrib/libio-io/l4/io/io.h File Reference	1757
15.89.1	Function Documentation	1759
15.89.1.1	l4io_get_root_device()	1759
15.89.1.2	l4io_iterate_devices()	1759
15.89.1.3	l4io_request_all_ioports()	1760
15.89.1.4	l4io_request_icu()	1760
15.90	io.h	1760
15.91	l4/sys/types.h File Reference	1761
15.91.1	Detailed Description	1764
15.91.2	Function Documentation	1764
15.91.2.1	l4_capability_next()	1764
15.92	types.h	1764
15.93	l4/cxx/avl_map File Reference	1767
15.93.1	Detailed Description	1768
15.94	avl_map	1768
15.95	l4/cxx/avl_set File Reference	1770

15.95.1 Detailed Description	1771
15.96avl_set	1771
15.97I4/cxx/avl_tree File Reference	1774
15.97.1 Detailed Description	1776
15.98avl_tree	1776
15.99I4/cxx/basic_ostream File Reference	1780
15.99.1 Detailed Description	1781
15.100basic_ostream	1781
15.101I4/cxx/basic_vector.h File Reference	1784
15.101.1 Detailed Description	1784
15.102basic_vector.h	1785
15.103I4/cxx/bits/bst.h File Reference	1785
15.103.1 Detailed Description	1787
15.104bst.h	1787
15.105I4/cxx/bits/bst_base.h File Reference	1789
15.105.1 Detailed Description	1791
15.106bst_base.h	1791
15.107I4/cxx/bits/bst_iter.h File Reference	1792
15.107.1 Detailed Description	1793
15.108bst_iter.h	1794
15.109I4/cxx/exceptions File Reference	1795
15.109.1 Detailed Description	1797
15.110exceptions	1797
15.111I4/cxx/iostream File Reference	1800
15.111.1 Detailed Description	1801
15.112ostream	1801
15.113I4/cxx/ipc_helper File Reference	1801
15.113.1 Detailed Description	1802
15.114ipc_helper	1803
15.115I4/cxx/ipc_stream File Reference	1803

15.115.1	Detailed Description	1806
15.115.2	Function Documentation	1806
15.115.2.1	operator<<() [1/4]	1806
15.115.2.2	operator<<() [2/4]	1807
15.115.2.3	operator<<() [3/4]	1808
15.115.2.4	operator<<() [4/4]	1808
15.115.2.5	operator>>() [1/6]	1809
15.115.2.6	operator>>() [2/6]	1810
15.115.2.7	operator>>() [3/6]	1811
15.115.2.8	operator>>() [4/6]	1811
15.115.2.9	operator>>() [5/6]	1812
15.115.2.10	operator>>() [6/6]	1813
15.116	pc_stream	1814
15.117	/cxx/l4iostream File Reference	1824
15.117.1	Detailed Description	1825
15.118	iostream	1825
15.119	/cxx/l4types.h File Reference	1826
15.119.1	Detailed Description	1826
15.120	types.h	1827
15.121	/cxx/main_thread File Reference	1827
15.121.1	Detailed Description	1828
15.122	main_thread	1828
15.123	/cxx/pair File Reference	1829
15.123.1	Detailed Description	1830
15.124	pair	1830
15.125	/cxx/std_exc_io File Reference	1831
15.125.1	Detailed Description	1831
15.126	std_exc_io	1832
15.127	/cxx/string.h File Reference	1832
15.127.1	Detailed Description	1833

15.128	tring.h	1834
15.129	irq/irq.h File Reference	1834
15.129.1	Detailed Description	1836
15.130	q.h	1836
15.131	arm/l4/util/irq.h File Reference	1837
15.131.1	Detailed Description	1837
15.132	q.h	1838
15.133	amd64/l4/util/irq.h File Reference	1838
15.133.1	Detailed Description	1839
15.133.2	Function Documentation	1839
15.133.2.1	l4util_irq_acknowledge()	1839
15.134	q.h	1840
15.135	86/l4/util/irq.h File Reference	1841
15.135.1	Detailed Description	1841
15.135.2	Function Documentation	1842
15.135.2.1	l4util_irq_acknowledge()	1842
15.136	q.h	1842
15.137	/sys/irq.h File Reference	1843
15.137.1	Detailed Description	1845
15.138	q.h	1845
15.139	/libedid/edid.h File Reference	1848
15.140	edid.h	1849
15.141	libgfxbitmap/bitmap.h File Reference	1849
15.141.1	Detailed Description	1851
15.141.2	Macro Definition Documentation	1851
15.141.2.1	pSLIM_BMAP_START_LSB	1852
15.142	Bitmap.h	1852
15.143	libgfxbitmap/font.h File Reference	1853
15.143.1	Detailed Description	1854
15.144	font.h	1854

15.145. libgfxbitmap/support File Reference	1855
15.145.1 Detailed Description	1855
15.146 support	1856
15.147. re/c/dataspace.h File Reference	1856
15.147.1 Detailed Description	1857
15.148 dataspace.h	1858
15.149. re/c/debug.h File Reference	1859
15.149.1 Detailed Description	1859
15.150 debug.h	1859
15.151. re/c/dma_space.h File Reference	1860
15.151.1 Detailed Description	1861
15.151.2 Typedef Documentation	1861
15.151.2.1 re_dma_space_dma_addr_t	1861
15.151.3 Enumeration Type Documentation	1861
15.151.3.1 re_dma_space_direction	1861
15.151.3.2 re_dma_space_space_attris	1862
15.152 dma_space.h	1862
15.153. re/c/event.h File Reference	1863
15.153.1 Detailed Description	1864
15.154 event.h	1865
15.155. re/event.h File Reference	1865
15.155.1 Detailed Description	1866
15.156 event.h	1866
15.157. re/c/log.h File Reference	1867
15.157.1 Detailed Description	1868
15.158 bg.h	1868
15.159. re/c/mem_alloc.h File Reference	1869
15.159.1 Detailed Description	1870
15.160 mem_alloc.h	1871
15.161. re/c/namespace.h File Reference	1872

15.161. Detailed Description	1873
15.162amespace.h	1873
15.163/re/c/rm.h File Reference	1874
15.163. Detailed Description	1875
15.164m.h	1875
15.165/re/c/util/cap_alloc.h File Reference	1878
15.165. Detailed Description	1878
15.166ap_alloc.h	1879
15.167/re/c/util/kumem_alloc.h File Reference	1879
15.167. Detailed Description	1880
15.168kumem_alloc.h	1880
15.169/re/c/util/video/goos_fb.h File Reference	1881
15.169. Detailed Description	1881
15.170goos_fb.h	1882
15.171/re/c/video/colors.h File Reference	1882
15.171. Detailed Description	1884
15.172olors.h	1884
15.173/re/c/video/goos.h File Reference	1885
15.173. Detailed Description	1886
15.174goos.h	1887
15.175/re/c/video/view.h File Reference	1888
15.175. Detailed Description	1889
15.176iew.h	1890
15.177/re/cap_alloc File Reference	1891
15.177. Detailed Description	1892
15.178ap_alloc	1892
15.179/re/util/cap_alloc File Reference	1893
15.179. Detailed Description	1895
15.180ap_alloc	1895
15.181/re/consts File Reference	1897

15.181.1 Detailed Description	1898
15.182 onsts	1898
15.183/re/dataspace File Reference	1898
15.183.1 Detailed Description	1899
15.184 dataspace	1900
15.185/re/dataspace-sys.h File Reference	1901
15.185.1 Detailed Description	1901
15.186 dataspace-sys.h	1902
15.187/re/debug File Reference	1902
15.187.1 Detailed Description	1903
15.188 debug	1904
15.189/re/dma_space File Reference	1904
15.190 dma_space	1905
15.191/re/elf_aux.h File Reference	1906
15.191.1 Detailed Description	1908
15.192 elf_aux.h	1908
15.193/re/env File Reference	1909
15.193.1 Detailed Description	1910
15.194 env	1910
15.195/re/env.h File Reference	1911
15.195.1 Detailed Description	1913
15.195.2 Typedef Documentation	1913
15.195.2.1 re_env_t	1913
15.196 env.h	1913
15.197/re/error_helper File Reference	1915
15.197.1 Detailed Description	1917
15.198 error_helper	1917
15.199/re/util/event File Reference	1918
15.200 event	1919
15.201/re/impl/dataspace_impl.h File Reference	1921

15.201. Detailed Description	1922
15.202. dataspace_impl.h	1922
15.203. /re/impl/mem_alloc_impl.h File Reference	1923
15.203. Detailed Description	1924
15.204. mem_alloc_impl.h	1924
15.205. /re/impl/namespace_impl.h File Reference	1925
15.205. Detailed Description	1926
15.206. namespace_impl.h	1926
15.207. /re/impl/rm_impl.h File Reference	1927
15.207. Detailed Description	1928
15.208. rm_impl.h	1928
15.209. /re/l4aux.h File Reference	1930
15.209. Detailed Description	1930
15.210. l4aux.h	1931
15.211. /re/log File Reference	1931
15.211. Detailed Description	1933
15.212. log	1933
15.213. /re/log-sys.h File Reference	1933
15.213. Detailed Description	1934
15.214. log-sys.h	1934
15.215. /re/mem_alloc File Reference	1934
15.215. Detailed Description	1936
15.216. mem_alloc	1936
15.217. /re/mem_alloc-sys.h File Reference	1936
15.217. Detailed Description	1937
15.218. mem_alloc-sys.h	1937
15.219. /re/namespace File Reference	1938
15.219. Detailed Description	1939
15.220. namespace	1939
15.221. /re/namespace-sys.h File Reference	1940

15.221. Detailed Description	1941
15.222. namespace-sys.h	1941
15.223. re/parent File Reference	1941
15.223. Detailed Description	1943
15.224. parent	1943
15.225. re/parent-sys.h File Reference	1943
15.225. Detailed Description	1944
15.226. parent-sys.h	1944
15.227. re/rm File Reference	1944
15.227. Detailed Description	1945
15.228. m	1946
15.229. re/rm-sys.h File Reference	1949
15.229. Detailed Description	1949
15.230. m-sys.h	1950
15.231. re/util/bitmap_cap_alloc File Reference	1950
15.231. Detailed Description	1951
15.232. bitmap_cap_alloc	1952
15.233. re/util/cap File Reference	1953
15.233. Detailed Description	1954
15.234. ap	1954
15.235. re/util/cap_alloc_impl.h File Reference	1954
15.235. Detailed Description	1955
15.236. ap_alloc_impl.h	1956
15.237. re/util/counting_cap_alloc File Reference	1956
15.237. Detailed Description	1957
15.238. counting_cap_alloc	1958
15.239. re/util/item_alloc File Reference	1960
15.239. Detailed Description	1961
15.240. item_alloc	1961
15.241. re/util/kumem_alloc File Reference	1962

15.241. Detailed Description	1963
15.242. umem_alloc	1963
15.243. re/util/region_mapping File Reference	1963
15.243. Detailed Description	1964
15.244. region_mapping	1965
15.245. re/video/goos-sys.h File Reference	1969
15.245. Detailed Description	1970
15.246. goos-sys.h	1970
15.247. shmc/shmc.h File Reference	1971
15.247. Detailed Description	1973
15.248. shmc.h	1973
15.249. sigma0/sigma0.h File Reference	1975
15.249. Detailed Description	1977
15.250. sigma0.h	1978
15.251. l4/sys/__kernel_object_impl.h File Reference	1979
15.251. Detailed Description	1980
15.252. __kernel_object_impl.h	1980
15.253. l4/sys/__ktrace-impl.h File Reference	1980
15.253. Detailed Description	1982
15.254. __ktrace-impl.h	1982
15.255. l4/sys/__typeinfo.h File Reference	1983
15.255. Detailed Description	1985
15.256. __typeinfo.h	1986
15.257. l4/sys/cache.h File Reference	1996
15.257. Detailed Description	1996
15.258. cache.h	1997
15.259. arm/l4/sys/cache.h File Reference	1997
15.259. Detailed Description	1998
15.260. cache.h	1999
15.261. amd64/l4/sys/cache.h File Reference	2000

15.261. Detailed Description	2001
15.262. cache.h	2001
15.263. arm/l4/sys/cache.h File Reference	2002
15.263. Detailed Description	2002
15.264. cache.h	2002
15.265. arm/sys/capability File Reference	2003
15.265. Detailed Description	2005
15.265.2. Macro Definition Documentation	2005
15.265.2.1. L4_DISABLE_COPY	2005
15.266. capability	2005
15.267. arm/sys/compiler.h File Reference	2007
15.267. Detailed Description	2008
15.268. compiler.h	2008
15.269. arm/sys/consts.h File Reference	2010
15.269. Detailed Description	2012
15.270. consts.h	2012
15.271. arm/l4/sys/consts.h File Reference	2014
15.271. Detailed Description	2014
15.272. consts.h	2015
15.273. arm64/l4/sys/consts.h File Reference	2015
15.273. Detailed Description	2015
15.274. consts.h	2016
15.275. arm64/l4/sys/consts.h File Reference	2016
15.275. Detailed Description	2016
15.276. consts.h	2017
15.277. re/consts.h File Reference	2017
15.277. Detailed Description	2018
15.278. consts.h	2018
15.279. arm/sys/cxx/ipc_client File Reference	2018
15.279. Macro Definition Documentation	2020

15.279.1.1	l4_rpc_def	2020
15.280	ipc_client	2020
15.281	sys/cxx/ipc_iface File Reference	2021
15.281.1	Detailed Description	2023
15.281.2	Macro Definition Documentation	2023
15.281.2.1	l4_inline_rpc	2023
15.281.2.2	l4_inline_rpc_nf	2024
15.281.2.3	l4_inline_rpc_nf_op	2024
15.281.2.4	l4_inline_rpc_op	2025
15.281.2.5	l4_rpc	2026
15.281.2.6	l4_rpc_nf	2026
15.281.2.7	l4_rpc_nf_op	2027
15.281.2.8	l4_rpc_op	2027
15.282	ipc_iface	2028
15.283	sys/cxx/ipc_server File Reference	2033
15.283.1	Detailed Description	2034
15.284	ipc_server	2034
15.285	sys/cxx/ipc_types File Reference	2035
15.286	ipc_types	2037
15.287	sys/cxx/types File Reference	2044
15.288	types	2045
15.289	sys/debugger File Reference	2046
15.289.1	Detailed Description	2047
15.290	debugger	2048
15.291	sys/debugger.h File Reference	2048
15.291.1	Detailed Description	2050
15.291.2	Function Documentation	2050
15.291.2.1	l4_debugger_get_object_name()	2050
15.291.2.2	l4_debugger_query_log_name()	2051
15.291.2.3	l4_debugger_query_log_typeid()	2051

15.291.2.44_debugger_switch_log()	2052
15.292debugger.h	2052
15.293/sys/err.h File Reference	2055
15.293.1Detailed Description	2056
15.294err.h	2057
15.295/sys/factory File Reference	2057
15.295.1Detailed Description	2059
15.296factory	2059
15.297/sys/factory.h File Reference	2061
15.297.1Detailed Description	2062
15.298factory.h	2062
15.299/sys/icu File Reference	2067
15.299.1Detailed Description	2068
15.300icu	2068
15.301/sys/icu.h File Reference	2068
15.301.1Detailed Description	2071
15.302icu.h	2071
15.303/sys/ipc.h File Reference	2075
15.303.1Detailed Description	2076
15.303.2Function Documentation	2077
15.303.2.1l4_ipc_to_errno()	2077
15.304ipc.h	2077
15.305/arm/l4f/l4/sys/ipc.h File Reference	2080
15.305.1Detailed Description	2081
15.306ipc.h	2081
15.307/arm/l4f/l4/sys/ipc.h File Reference	2082
15.307.1Detailed Description	2083
15.308ipc.h	2083
15.309/sys/ipc_gate File Reference	2083
15.309.1Detailed Description	2085

15.310	pc_gate	2085
15.311	/sys/pc_gate.h File Reference	2085
15.311.1	Detailed Description	2087
15.312	pc_gate.h	2087
15.313	/sys/irq File Reference	2088
15.313.1	Detailed Description	2090
15.314	q	2090
15.315	/sys/kernel_object.h File Reference	2092
15.315.1	Detailed Description	2093
15.316	kernel_object.h	2093
15.317	/sys/kip File Reference	2094
15.317.1	Detailed Description	2095
15.318	ip	2096
15.319	/sys/ktrace.h File Reference	2097
15.319.1	Detailed Description	2099
15.320	trace.h	2099
15.321	/sys/l4int.h File Reference	2100
15.321.1	Detailed Description	2101
15.322	l4int.h	2101
15.323	arm/l4/sys/l4int.h File Reference	2102
15.323.1	Detailed Description	2102
15.324	l4int.h	2102
15.325	amd64/l4/sys/l4int.h File Reference	2102
15.325.1	Detailed Description	2103
15.326	l4int.h	2103
15.327	x86/l4/sys/l4int.h File Reference	2103
15.327.1	Detailed Description	2104
15.328	l4int.h	2104
15.329	/sys/memdesc.h File Reference	2104
15.329.1	Detailed Description	2106

15.330memdesc.h	2106
15.331/sys/meta File Reference	2108
15.331.1Detailed Description	2110
15.332meta	2110
15.333/sys/pager File Reference	2110
15.333.1Detailed Description	2112
15.334pager	2112
15.335/sys/platform_control File Reference	2112
15.335.1Detailed Description	2113
15.336platform_control	2114
15.337/sys/platform_control.h File Reference	2114
15.337.1Detailed Description	2116
15.338platform_control.h	2116
15.339/sys/scheduler File Reference	2118
15.339.1Detailed Description	2119
15.340scheduler	2119
15.341/sys/scheduler.h File Reference	2120
15.341.1Detailed Description	2121
15.342scheduler.h	2121
15.343/sys/semaphore File Reference	2124
15.343.1Detailed Description	2126
15.344semaphore	2126
15.345/sys/smart_capability File Reference	2126
15.345.1Detailed Description	2128
15.346smart_capability	2128
15.347/sys/task File Reference	2130
15.347.1Detailed Description	2132
15.348task	2132
15.349/sys/task.h File Reference	2133
15.349.1Detailed Description	2134

15.350ask.h	2134
15.351i4/sys/thread File Reference	2138
15.351.1Detailed Description	2139
15.352hread	2139
15.353i4/cxx/thread File Reference	2141
15.353.1Detailed Description	2142
15.354hread	2142
15.355i4/sys/typeinfo_svr File Reference	2143
15.355.1Detailed Description	2145
15.356ypeinfo_svr	2145
15.357i4/sys/utcb.h File Reference	2145
15.357.1Detailed Description	2147
15.358tcb.h	2147
15.359arm/i4/sys/utcb.h File Reference	2150
15.359.1Detailed Description	2151
15.360tcb.h	2151
15.361amd64/i4/sys/utcb.h File Reference	2152
15.361.1Detailed Description	2154
15.362tcb.h	2154
15.363i4/i4/sys/utcb.h File Reference	2155
15.363.1Detailed Description	2157
15.364tcb.h	2157
15.365i4/sys/vcon File Reference	2158
15.365.1Detailed Description	2160
15.366con	2160
15.367i4/sys/vcon.h File Reference	2160
15.367.1Detailed Description	2162
15.367.2Enumeration Type Documentation	2163
15.367.2.1L4_vcon_read_flags	2163
15.368con.h	2163

15.369/sys/vhw.h File Reference	2166
15.369.1 Detailed Description	2167
15.370hw.h	2168
15.371/sys/vm File Reference	2169
15.371.1 Detailed Description	2170
15.372m	2170
15.373/util/alloc.h File Reference	2170
15.373.1 Detailed Description	2171
15.374alloc.h	2171
15.375/cxx/alloc.h File Reference	2172
15.375.1 Detailed Description	2172
15.376alloc.h	2172
15.377/util/assert.h File Reference	2173
15.377.1 Detailed Description	2174
15.378assert.h	2174
15.379/sys/assert.h File Reference	2175
15.379.1 Detailed Description	2176
15.379.2 Macro Definition Documentation	2177
15.379.2.1 _assert	2177
15.380assert.h	2177
15.381/util/atomic.h File Reference	2178
15.381.1 Detailed Description	2180
15.382atomic.h	2181
15.383/arm/l4/sys/atomic.h File Reference	2185
15.383.1 Detailed Description	2185
15.384atomic.h	2186
15.385/cxx/atomic.h File Reference	2186
15.385.1 Detailed Description	2187
15.386atomic.h	2187
15.387/util/backtrace.h File Reference	2188

15.387.1 Detailed Description	2188
15.387.2 Function Documentation	2189
15.387.2.1 util_backtrace()	2189
15.388 backtrace.h	2189
15.389 util/base64.h File Reference	2189
15.389.1 Detailed Description	2190
15.390 base64.h	2191
15.391 util/bitops.h File Reference	2191
15.391.1 Detailed Description	2193
15.392 bitops.h	2193
15.393 util/elf.h File Reference	2196
15.393.1 Detailed Description	2209
15.394 elf.h	2209
15.395 util/getopt.h File Reference	2219
15.395.1 Detailed Description	2219
15.396 getopt.h	2219
15.397 util/keymap.h File Reference	2220
15.397.1 Detailed Description	2221
15.398 keymap.h	2221
15.399 util/kip.h File Reference	2221
15.400 kip.h	2222
15.401 sys/kip.h File Reference	2223
15.401.1 Detailed Description	2224
15.402 ip.h	2224
15.403 util/kprintf.h File Reference	2226
15.403.1 Detailed Description	2226
15.404 kprintf.h	2227
15.405 util/list_alloc.h File Reference	2227
15.405.1 Detailed Description	2228
15.405.2 Function Documentation	2228

15.405.2.14la_alloc()	2228
15.405.2.24la_avail()	2228
15.405.2.34la_dump()	2229
15.405.2.44la_free()	2229
15.405.2.54la_init()	2229
15.406st_alloc.h	2230
15.407util/lock.h File Reference	2230
15.407.1Detailed Description	2231
15.408ock.h	2231
15.409util/mb_info.h File Reference	2232
15.409.1Detailed Description	2234
15.409.2Macro Definition Documentation	2234
15.409.2.1LUTIL_MB_MEMORY	2234
15.409.2.2MB_ARD_MEMORY	2235
15.409.2.3MB_ART_MEMORY	2235
15.410nb_info.h	2235
15.411util/parse_cmd.h File Reference	2239
15.411.1Detailed Description	2240
15.412parse_cmd.h	2240
15.413util/rand.h File Reference	2240
15.413.1Detailed Description	2241
15.414and.h	2242
15.415util/reboot.h File Reference	2242
15.415.1Detailed Description	2243
15.416reboot.h	2243
15.417util/sll.h File Reference	2243
15.417.1Detailed Description	2244
15.418ll.h	2244
15.419util/splitlog2.h File Reference	2247
15.419.1Detailed Description	2248

15.420	splitlog2.h	2248
15.421	util/stack.h File Reference	2249
15.421.1	Detailed Description	2250
15.421.2	Function Documentation	2250
15.421.2.1	util_stack_get_sp()	2251
15.422	stack.h	2251
15.423	util/thread.h File Reference	2251
15.423.1	Detailed Description	2252
15.423.2	Macro Definition Documentation	2253
15.423.2.1	__L4UTIL_THREAD_FUNC	2253
15.424	thread.h	2253
15.425	sys/thread.h File Reference	2254
15.425.1	Detailed Description	2256
15.426	thread.h	2256
15.427	arm/l4/sys/thread.h File Reference	2264
15.427.1	Detailed Description	2264
15.428	thread.h	2264
15.429	vbus/vbus.h File Reference	2265
15.429.1	Detailed Description	2266
15.429.2	Enumeration Type Documentation	2266
15.429.2.1	anonymous enum	2266
15.430	vbus.h	2267
15.431	l4/vbus/vbus_types.h File Reference	2268
15.431.1	Detailed Description	2269
15.431.2	Enumeration Type Documentation	2269
15.431.2.1	l4vbus_resource_type_t	2269
15.432	vbus_types.h	2269

16 Example Documentation	2271
16.1 examples/clntsrv/client.cc	2271
16.2 examples/clntsrv/clntsrv.cfg	2272
16.3 examples/clntsrv/server.cc	2272
16.4 examples/libs/l4re/c++/mem_alloc/ma+rm.cc	2273
16.5 examples/libs/l4re/c++/shared_ds/ds_clnt.cc	2274
16.6 examples/libs/l4re/c++/shared_ds/ds_srv.cc	2275
16.7 examples/libs/l4re/c++/shared_ds/shared_ds.cfg	2277
16.8 examples/libs/l4re/c/ma+rm.c	2278
16.9 examples/libs/l4re/streammap/client.cc	2279
16.10examples/libs/l4re/streammap/server.cc	2280
16.11examples/libs/l4re/streammap/streammap.cfg	2281
16.12examples/libs/libirq/async_isr.c	2281
16.13examples/libs/libirq/loop.c	2282
16.14examples/libs/shmc/prodcons.c	2283
16.15examples/sys/aliens/main.c	2284
16.16examples/sys/ipc/ipc.cfg	2288
16.17examples/sys/ipc/ipc_example.c	2288
16.18examples/sys/isr/main.c	2290
16.19examples/sys/migrate/thread_migrate.cc	2291
16.20examples/sys/migrate/thread_migrate.cfg	2292
16.21 examples/sys/singlestep/main.c	2293
16.22examples/sys/start-with-exc/main.c	2295
16.23examples/sys/utcb-ipc/main.c	2297
16.24examples/sys/ux-vhw/main.c	2298
16.25hello/server/src/main.c	2299
16.26tmpfs/lib/src/fs.cc	2299
Index	2307

Chapter 1

Overview

Welcome to the documentation of the L4 Runtime Environment, 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. [Tutorial](#) helps you getting started with setting up the development environment and writing your own first L4Re application.
3. [Programming for L4Re](#) explains in detail the most important programming concepts.
4. [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 environment 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](#) environment.

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 Fiasco.OC

The Fiasco.OC microkernel is the lowest-level piece of software running in an L4-based system. The microkernel is the only program 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](#)).

Fiasco.OC kernel 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*. Fiasco system calls 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 Fiasco.OC 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 Fiasco 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.

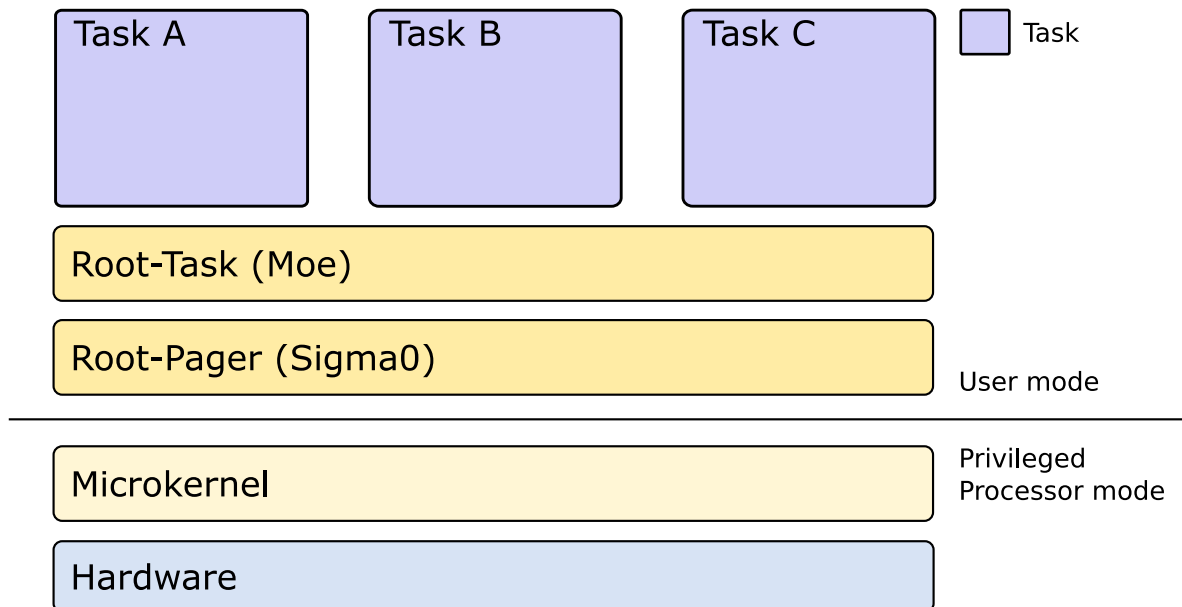


Figure 2.1 Basic Structure of an L4Re based system

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 [L4](#) 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 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 L4 Runtime Environment (L4Re)

The [L4](#) Runtime Environment ([L4Re](#)) provides a basic set of services and abstractions, which are useful to implement and run user-level applications on top of the Fiasco.OC microkernel.

[L4Re](#) consists of a set of libraries and servers. Libraries as well as server interfaces are completely object oriented. They implement prototype implementations for the classes defined by the [L4Re](#) specification.

A minimal L4Re-based application needs 3 components to be booted beforehand: the Fiasco 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

Tutorial

This tutorial assumes that the reader is familiar with the basic L4 concepts that were discussed in the [Introduction](#) section and has a working knowledge of C++.

Here you can find the first steps to boot a very simple setup. The setup consists of the following components:

- Fiasco.OC — Microkernel
- Sigma0 — Root Pager
- Moe — Root Task
- Ned — Init Process
- hello — Hello World Application

The guide assumes that you already compiled the base components and describes how to generate an ISO image, with GRUB 1 or GRUB 2 as a boot loader, that can for example be booted within QEMU.

First you need a `modules.list` file that contains an entry for the scenario.

```
modaddr 0x002000000

entry hello
  kernel fiasco -serial_esc
  roottask moe rom/hello.cfg
  module l4re
  module ned
  module hello.cfg
  module hello
```

This file describes all the binaries and scripts to put into the ISO image, and also describes the GRUB `menu.lst` contents. What you need to do is to set the `make` variable `MODULE_SEARCH_PATH` to contain the path to your Fiasco.OC build directory and the directory containing your `hello.cfg` script.

The `hello.cfg` script should look like the following. A ready to use version can be found in `I4/conf/examples`.

```
local L4 = require("L4");
L4.default_loader:start({}, "rom/hello");
```

The first line of this script ensures that the [L4](#) package is available for the script. The second line uses the default loader object defined in that package and starts the binary `rom/hello`.

Note

All modules defined in `modules.list` are available as data spaces ([L4Re::Dataspace](#)) and registered in a name space ([L4Re::Namespace](#)). This name space is in turn available as 'rom' to the init process ([Ned](#)).

Now you can go to your [L4Re](#) build directory and run the following command.

Note

The example assumes that you have created the `modules.list` and `hello.cfg` files in the `/tmp` directory. Adapt if you created them somewhere else.

```
make grubliso E=hello MODULES_LIST=/tmp/modules.list MODULE_SEARCH_PATH=/tmp:<path_to_fiasco_build_dir>
```

Or as an alternative use GRUB 2:

```
make grub2iso E=hello MODULES_LIST=/tmp/modules.list MODULE_SEARCH_PATH=/tmp:<path_to_fiasco_build_dir>
```

Now you should be able to boot the image in QEMU by running:

```
qemu-system-i386 -cdrom images/hello.iso -serial stdio
```

If you press `<ESC>` in the terminal that shows you the serial output you enter the Fiasco.OC kernel debugger... Have fun.

3.1 Customizations

A basic set of bootable entries can be found in `l4/conf/modules.list`. This file is the default for any image creation as shown above. It is recommended that local modification regarding image creation are done in `conf/Makeconf.boot`. Initially you may copy `Makeconf.boot.example` to `Makeconf.boot`. You can overwrite `MODULES_LIST` to set your own modules-list file. Set `MODULE_SEARCH_PATH` to your setup according to the examples given in the file. When configured a `make` call is reduced to:

```
make grub2iso E=hello
```

All other local configuration can be done in a `Makeconf.local` file located in the `l4` directory.

Chapter 4

Programming for L4Re

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

You should have already a basic understanding of the [L4Re](#) programming environment from the tutorial.

- [Capabilities and Naming](#)
- [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](#)

4.1 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 be accessed only 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 flex pages. 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.

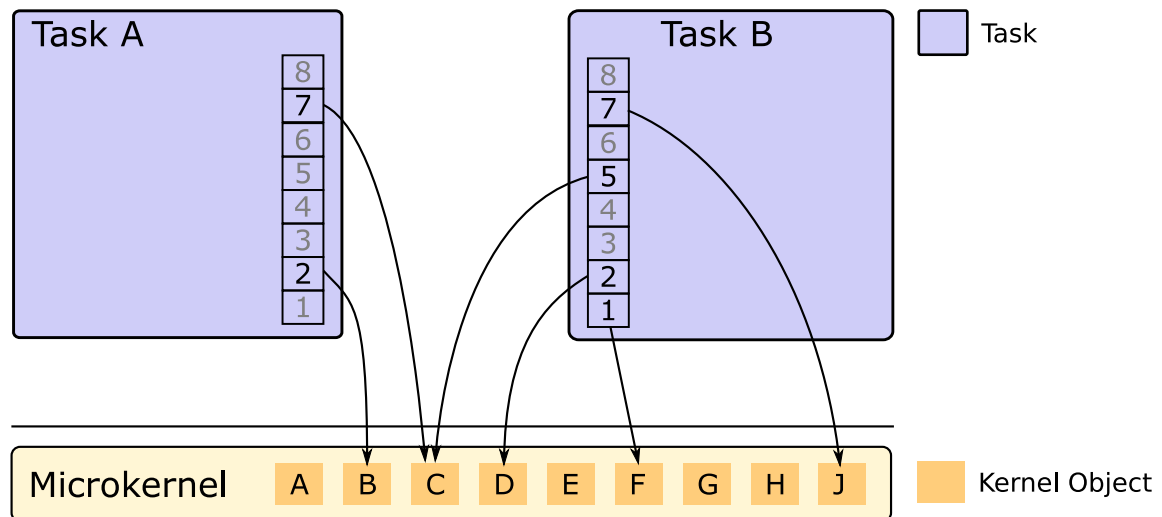


Figure 4.1 Capabilities and Local Naming in L4

So how does an application get access to a service? In general all applications are started with an initial set of objects available. 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 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.

4.2 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 (<http://www.lua.org>) as its scripting language, see [Ned Script example](#) for more details.

4.2.1 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");
```

4.2.2 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 the client and the server. 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.

4.3 Memory management - Data Spaces and the Region Map

4.3.1 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 grant the memory in a secure way, often referred to as *memory mapping*.

The described mechanism can be used to construct a memory 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.

4.3.1.1 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 the some portion are provided and further pages are inserted by the pager as a result of page faults.

4.3.1.2 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 to attach and detach data spaces to an address space as well as to reserve areas of virtual memory. Since the region-map object possesses all knowledge about virtual memory layout of a task, it also serves as an application's default pager.

4.3.1.3 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](#).

4.4 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 Fiasco kernel and connected either to the serial line or to the screen if available.

See also

[Virtual Console](#)

4.5 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 allocation [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](#)

4.6 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.

4.6.1 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.

4.6.2 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.

4.7 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...
```

4.8 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(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;
};
```

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 `Rpcs` type.

4.8.1 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.

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

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

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

Array types can be used to transmit arrays of variable length. They can either be stored in a client-provided buffer (L4::Ipc::Array), copied into a server-provided buffer (L4::Ipc::Array_in_buf) or directly read and written into the U↔TCB (L4::Ipc::Array_ref). For latter type, the start position of an array type needs to be known in advance. That implies that only one such array can be transmitted per direction and it must be the last part in the message.

Type	Direction	Client-Type	Server-Type
L4::Ipc::↔ Array<const T>	Input	L4::Ipc::↔ Array<const T>	L4::Ipc::Array_↔ ref<const T>
L4::Ipc::↔ Array<const T>	Input	L4::Ipc::↔ Array<const T>	L4::Ipc::Array_in_↔ buf<T> const &
L4::Ipc::Array<T> &	Output	L4::Ipc::Array<T> &	L4::Ipc::Array_↔ ref<T> &
L4::Ipc::Array_↔ ref<T> &	Output	L4::Ipc::Array_↔ ref<T> &	L4::Ipc::Array_↔ ref<T> &

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
L4::Ipc::Opt<T>	Input	L4::Ipc::Opt<T>	T
L4::Ipc::Opt<const T*>	Input	L4::Ipc::Opt<const T*>	T
L4::Ipc::Opt<T &>	Output	T &	L4::Ipc::Opt<T> &
L4::Ipc::Opt<T *>	Output	T *	L4::Ipc::Opt<T> &
L4::Ipc::Opt<Array_↔ _ref<T> &>	Output	Array_ref<T> &	L4::Ipc::Opt<Array_↔ _ref<T>> &

4.8.2 RPC Return Types

On the server side, the return type of an RPC handling function is always `long`. The return value is transmitted via the label field in `l4_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 `long`. Note that the client might not only receive the server return value in response but also an IPC error code.

4.8.3 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...)`.

4.9 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.

4.9.1 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.

4.9.2 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

4.9.3 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.

SRC_CC_IS_CXX11

C++ standard to use. Default is `c++0x`.

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.

4.9.4 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`.

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.

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.

4.9.5 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_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.

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_TIMEOUT

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

MOE_CFG

LUA configuration file for startup to give to moe

REQUIRED_MODULES

Additional modules needed to run the test.

QEMU_ARGS

Additional parameters to supply to QEMU.

MOE_ARGS

Additional parameters to supply to moe.

KERNEL_CONF

Features the Fiasco kernel 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.

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 2, the test is marked as skipped with the reason provided in the final line of stdout.

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

Fiasco kernel 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>, L4_LX_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.

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/l4f/l4re-core/moe/test_namespace.t
```

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

```
prove test/t/amd64_amdfam10/l4f/l4re-core/moe/test_namespace.t
prove -r test/t/amd64_amdfam10/l4f/l4re-core/
prove -rv test/t/amd64_amdfam10/l4f/l4re-core/
```

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=maketest`. 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_SLEEPTIME

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

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.

Chapter 5

L4Re Servers

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

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 Fiasco.OC you usually need to run Sigma0, nevertheless it is possible to replace Sigma0 by a different implementation.

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 of Fiasco.OC 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.
- [Uvmm, the virtual machine monitor](#)
- [Mag, the GUI Multiplexer](#)
Our default multiplexer for the graphics hardware is Mag. Mag is a Nitpicker (TODO: ref) derivate that allows secure multiplexing of the graphics and input hardware among multiple applications and multiple complete windowing environments.
- [Cons, the Console Multiplexer](#)
An interactive multiplexer for console in- and output. It buffers the output from different L4 clients and allows to switch between them to redirect input.

5.1 Moe, the Root-Task

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 micro kernel starts *Moe* as the Root-Task.

Moe provides a default implementation for the basic L4Re abstractions, such as data spaces (L4Re::Dataspace), region maps (L4Re::Rm), memory allocators (L4Re::Mem_alloc, L4::Factory), name spaces (L4Re::Namespace) and so on (see L4Re Interface).

Moe consists of the following subsystems:

- [Name-Space Provider](#) — provides instances of name spaces
- [Boot FS](#) — provides access to the files loaded during platform boot (e.g., linked into the boot image or loaded via GRUB boot loader)
- [Log Subsystem](#) — provides tagged log output for applications
- [Scheduler Subsystem](#) — provides simple scheduler objects for scheduling policy enforcement
- [Memory Allocator, Generic Factory](#) — provides allocation of physical RAM as data spaces, as well as allocation of the other L4Re objects provided by Moe

5.1.1 Memory Allocator, Generic Factory

The generic factory in Moe is responsible for all kinds of dynamic object allocation. The interface is a combination of L4::Factory and, for traditional reasons, L4Re::Mem_alloc.

The generic factory interface 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

The memory allocator in Moe is the alternative interface for allocating memory (RAM) in terms of L4Re::Dataspace-s (see also L4Re::Mem_alloc).

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)

5.1.2 Name-Space Provider

Moe provides a name spaces conforming to the [L4Re::Namespace](#) interface (see [Name-space API](#)). 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.

5.1.2.1 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 dataspace 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.

5.1.3 Log Subsystem

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

5.1.4 Scheduler Subsystem

The scheduler subsystem provides a simple scheduler proxy.

5.1.5 Command-Line Options

Moe's command-line syntax is:

```
moe [--debug=<flags>] [--init=<binary>] [--l4re-dbg=<flags>] [--ldr-flags=<flags>] [-- <init options>]
```

`--debug=<debug flags>`

This option enables debug messages from Moe itself, the `<debug flags>` values are a combination of `info`, `warn`, `boot`, `server`, `loader`, `exceptions`, and `ns` (or `all` for full verbosity).

`--init=<init process>`

This options allows to override the default init process binary, which is 'rom/ned'.

`--l4re-dbg=<debug flags>`

This option allows to set the debug options for the [L4Re](#) runtime environment of the init process. The flags are the same as for `--debug=`.

`--ldr-flags=<loader flags>`

This option allows setting some loader options for the [L4Re](#) runtime environment. The flags are `pre_alloc`, `all_segs_cow`, and `pinned_segs`.

`-- <init options>`

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

5.2 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](#), the [Root-Task](#) 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.

5.2.1 Lua Bindings for L4Re

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

5.2.1.1 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.

5.2.1.2 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.

5.2.1.3 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,
  Irq_muxer  = -19,
  Semaphore  = -20,
  Iommu      = -22,
  Ipc_gate   = 0,
}
```

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
}
```

5.2.1.4 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_tag` The string used for tagging log output of this process (defaults to the program name) (see `log_fab`).
- `log_color` The color used for the log tag (defaults to "white").
- `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.

5.2.2 Command Line Options

Ned's command line syntax is:

```
[--interactive|-i] [--cmdcap|-c CAP] [--noexit] [--execute|-e STATEMENT] <lua script> [options passed to lua s
```

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.

- Interactive Option: **interactive, i**

Start ned in interactive mode. After running the initial lua script, ned will present a prompt, where Lua statements can be typed in interactively.

Must not be used together with server mode (-c).

- Command Capability Option: **cmdcap, c**

Start ned in server mode. After running the initial Lua script, ned will accept Lua statements via IPC on the given capability (see [L4Re::Ned::Cmd_control](#)).

Must not be used together with interactive mode (-i).

- No exit Option: **noexit**

Ned does not terminate if only Return is entered at ned's prompt.

- Execute Statement Option: **execute, e**

Execute the Lua statement `STATEMENT`.

5.3 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 `Io.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 Io 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.

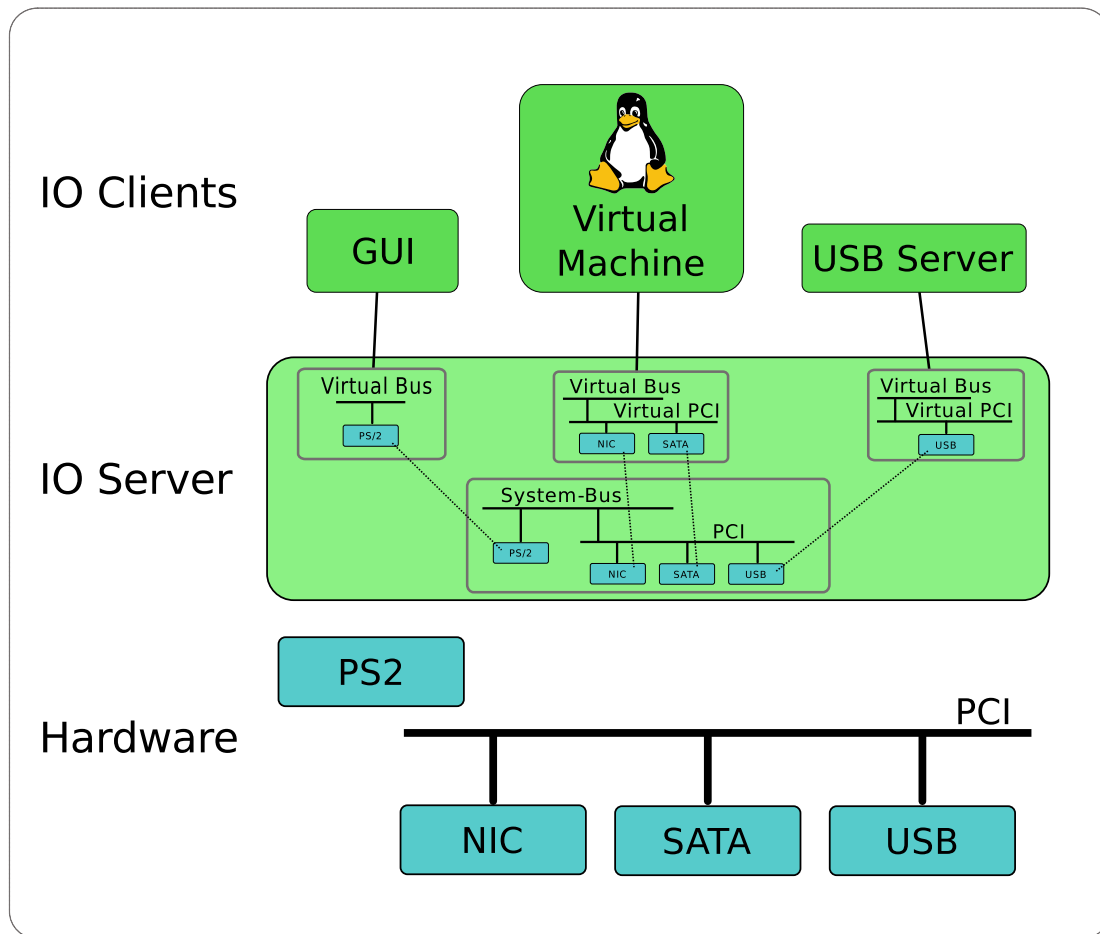


Figure 5.1 IO Service Architecture Overview

The Io 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 Io through the command line.

To allow clients access to a 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 busses correspond to the capabilities given to Io 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
    end)
```

```

-- 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("FOODEVICE"));
  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("BARDEVICE"));
  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_dddd
- 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
- `dddd` 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
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

Common Name	Description	PCI ID string
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
com_multiport_ser	Multiport serial controller	CC_0702
com_modem	Modem	CC_0703
com_gpiib	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.
```

5.4 Uvmm, the virtual machine monitor

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 '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.

5.5 Mag, the GUI Multiplexer

Mag is the default multiplexer for graphics hardware.

It is not, and does not attempt to be, a fully-fledged window manager.

Command Line Options

As Mag's only command line option it supports loading additional plugins via the application's command line. Plugins must be either a Lua file or a shared library. Shared libraries must be named `libmag-$LIBNAME.so`.

Mag Sessions

Mag provides two types of sessions which a client can create via the Factory interface.

1. 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. Mag provides the factory to create client sessions via the capability named `mag`.

2. 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. Mag provides the factory to create framebuffer sessions via the capability named `svc`.

The options described below are options the client provides to the `L4::Factory::create()` call. These options influence the appearance and behaviour of the newly created session.

Session Factory Options

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*.

Mag allows to define a text label and a color for all client session types. The label and the color are displayed when Mag enters the Xray mode.

- Label Option: **label**

`l=LABEL, label=LABEL` Set the session's text label to LABEL. The label is restricted to a length of 256 characters.

- Color Option: **col**

`col=COLOR` 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.

- `r`, `R` Red color
- `g`, `G` Green color
- `b`, `B` Blue color
- `w`, `W` White color
- `y`, `Y` Yellow color
- `v`, `V` Magenta color

Example

```
-- set label to "Linux" and use a light blue color
fb = mag_client:create(L4.Proto.Goos, "l=Linux", "col=98d9ff");
```

Mag Client Session Options

These options only apply to Mag client sessions.

- Default Background Option: **default-background**
`df1-bg, default-background` Marks this session as the default background.

Mag Client Framebuffer Session Options

These options only apply to Mag client framebuffer sessions.

- Geometry Option: **geometry**
`g=GEOMETRY, geometry=GEOMETRY` Set the session's geometry and position on the screen. GEOMETRY is provided in an X11-style format, e.g. `g=WIDTHxHEIGHT+X_OFFSET+Y_OFFSET`.
- Focus Option: **focus**
`focus` Set the focus to this session.
- Collapsed Option: **shaded**
`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.
- Fixed Option: **fixed** `fixed` The window cannot be moved on the screen.
- Barheight Option: **barheight**
`barheight=X` Set the height of the title bar in pixels.

Example

```
-- create a window of 640x480 pixels at position (100,100) on the screen.
fb = mag_fb:create(L4.Proto.Goos, "g=640x480+100+100");
```

5.6 Cons, the Console Multiplexer

cons allows to multiplex console output from different L4 clients to different L4::Vcon-capable in/output servers.

Multiplexers and Frontends

cons is able to connect multiple clients with multiple in/output servers.

Clients are handled by a *multiplexer*. Each multiplexer publishes a server capability that allows to create new client connections. The default multiplexer is normally known under the `cons` capability.

Actual in/output is handled by separate frontends. 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.

At the moment, cons only implements the L4::Vcon protocol for both, clients and frontends.

Command Line Options

cons accepts the following command line switches:

- `-a, --show-all`
Initially show output from all clients.
- `-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.
- `-k, --keep`
Keep the console buffer when a client disconnects.
- `-n, --defaultname`
Default multiplexer capability to use. Default: `cons`.
- `-B <size>, --defaultbufsize <size>`
Default buffer size per client in bytes. Default: 40960
- `-m <cap>, --mux <cap>`
Add a new multiplexer using the given capability. This is useful if output should be sent to different frontends.
- `-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. The server connected to the capability needs to understand the [L4::Vcon](#) protocol.

Chapter 6

Deprecated List

Global **L4::Factory::create_irq** (Cap< Irq >const &target_cap, l4_utcb_t *utcb=**l4_utcb()**)

Use **create()** with **Cap<Irq>** as argument instead.

Global **L4::Factory::create_thread** (Cap< Thread > const &target_cap, l4_utcb_t *utcb=**l4_utcb()**)

Use **create()** with **Cap<Thread>** as argument instead.

Global **L4::Factory::create_vm** (Cap< Vm >const &target_cap, l4_utcb_t *utcb=**l4_utcb()**)

Use **create()** with **Cap<Vm>** as argument instead.

Class **L4::lpc_svr::Timed_work< HOOKS >**

Use **L4::lpc_svr::Timeout_queue_hooks**

Global **L4::lrq::unmask** (l4_utcb_t *utcb=**l4_utcb()**)

Use **L4::lrq_eoi::unmask()**

Global **L4::Task::cap_has_child** (Cap< void > const &cap, l4_utcb_t *utcb=**l4_utcb()**)

Do not use. Undetermined future, might be removed.

Global **l4_task_cap_has_child** (l4_cap_idx_t task, l4_cap_idx_t cap) **L4_NOTHROW**)

Do not use. Undetermined future, might be removed.

Global **L4Re::Dataspace::phys** (l4_addr_t offset, l4_addr_t &phys_addr, l4_size_t &phys_size)

Use **L4Re::Dma_space** instead. This function will be removed in future releases.

Global **L4Re::Mem_alloc::free** (L4::Cap< Dataspace > mem) const)

This function is an empty stub which remains here for backward compatibility only. Use **L4::Task::unmap(mem, L4_FP_DELETE_OBJ)** or other similar means to remove a dataspace.

Global **l4re_ma_free** (l4re_ds_t const mem) **L4_NOTHROW**)

This function is deprecated. Use **l4_task_unmap()** or similar means to remove a dataspace.

Global **l4re_ma_free_srv** (l4_cap_idx_t srv, l4re_ds_t const mem) **L4_NOTHROW**)

This function is deprecated. Use **l4_task_unmap()** or similar means to remove a dataspace.

Chapter 7

Module Index

7.1 Modules

Here is a list of all modules:

Base API	101
Basic Macros	104
C++ IPC Interface Definition.	119
Internal Helpers	313
Cache Consistency	123
Capabilities	127
Error codes	198
Fiasco extensions	223
Fiasco real time scheduling extensions	229
Kernel Debugger	331
Flex pages	232
Integer Types	297
Kernel Interface Page	334
Fiasco-UX Virtual devices	230
Memory descriptors (C version)	419
Kernel Objects	342
Factory	210
IPC-Gate API	273
IRQs	277
Interrupt controller	317
L4 kernel object type information	373
Platform Control C API	473
Scheduler	496
Task	524
Thread	532
Thread control	550
vCPU API	627
Virtual Console	608
Virtual Machines	621
VM API for SVM	580
VM API for TZ	581
VM API for VMX	582
Memory operations.	424
Memory related	427

Object Invocation	459
Error Handling	192
Message Items	434
Message Tag	438
Realtime API	482
Timeouts	555
Virtual Registers (UTCBS)	622
ARM Virtual Registers (UTCB)	89
Buffer Registers (BRs)	116
Message Registers (MRs)	437
Exception registers	205
Thread Control Registers (TCRs)	549
amd64 Virtual Registers (UTCB)	626
x86 Virtual Registers (UTCB)	638
DMA API for L4Re	148
EDID parsing functionality	160
IO interface	265
IRQ handling library	276
Interface for asynchronous ISR handlers.	302
Interface for asynchronous ISR handlers with a given IRQ capability.	300
Interface using direct functionality.	304
Interface using direct functionality.	310
L4 IPC Opcodes	348
L4 V-BUS functions	352
L4vbus GPIO functions	389
L4vbus PCI functions	400
L4 VIRTIO Interface	363
L4 VIRTIO Block Device	361
L4 VIRTIO Transport Layer	364
L4Re C Interface	375
Capability allocator	131
DMA Space Interface	149
Dataspace interface	153
Debug interface	158
Event interface	201
Initial Environment	292
Kumem allocator utility	344
L4Re Util C Interface	387
Log interface	406
Memory allocator	413
Namespace interface	454
Region map interface	484
Video API	598
L4Re C++ Interface	378
Auxiliary data	100
C++ Exceptions	118
Console API	139
Debugging API	159
Event API	200
L4Re ELF Auxiliary Information	383
L4Re Protocol identifiers	386
L4Re Util C++ Interface	388
Kumem utilities	346
L4Re Capability API	380
Logging interface	410
Name-space API	453

Parent API	472
Region map API	483
Server-Side IPC framework	502
Shared Memory Library	504
Chunks	132
Consumer	140
Producer	476
Signals	516
Consumer	144
Producer	479
Sigma0 API	510
Internal constants	314
Small C++ Template Library	520
Utility Functions	573
Atomic Instructions	90
Bit Manipulation	107
Bitmap graphics and fonts	115
Functions for rendering bitmap data in frame buffers	248
Functions for rendering bitmap fonts to frame buffers	252
CPU related functions	120
Comfortable Command Line Parsing	136
ELF binary format	165
Functions to manipulate the local IDT	257
IA32 Port I/O API	258
Internal functions	316
Kernel Interface Page API	339
Low-Level Thread Functions	411
Machine Restarting Function	412
Random number support	480
Timestamp Counter	564
vCPU Support Library	630
Extended vCPU support	208

Chapter 8

Namespace Index

8.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

cxx	Our C++ library	641
cxx::Bits	Internal helpers for the cxx package	643
L4	L4 low-level kernel interface	643
L4::lpc	IPC related functionality	654
L4::lpc::Msg	IPC Message related functionality	662
L4::lpc_svr	Helper classes for L4::Server instantiation	668
L4::Typeid	Definition of interface data-type helpers	669
L4::Types	L4 basic type helpers for C++	670
L4Re	L4Re C++ Interfaces	670
L4Re::Vfs	Virtual file system for interfaces POSIX libc	675
L4vbus	C++ interface of the Vbus API	676
L4virtio	L4-VIRTIO Transport C++ API	677

Chapter 9

Hierarchical Index

9.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

L4::lpc::Array_ref< CHAR, LEN >	923
L4::lpc::Array< CHAR, LEN >	919
L4Re::Util::Auto_cap< L4::lrc >	1335
L4Re::Util::Auto_cap< L4Re::Dataspace >	1335
L4Re::Util::Auto_del_cap< L4::Triggerable >	1336
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, Bits::Avl_set_get_key< ITEM_TYPE > > .	740
cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC >	688
cxx::Bits::Base_avl_set< Pair< KEY_TYPE, DATA_TYPE >, COMPARE< KEY_TYPE >, ALLOC,	
Bits::Avl_map_get_key< KEY_TYPE > >	740
cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >	684
cxx::Bits::Base_avl_set< Pair< Region, Hdlr >, cxx::Lt_functor< Region >, Alloc, Bits::Avl_map_get_↔	
key< Region > >	740
cxx::Avl_map< Region, Hdlr, cxx::Lt_functor, Alloc >	684
cxx::Base_slab< sizeof(Type), Slab_size, Max_free, Alloc >	699
cxx::Slab< Type, Slab_size, Max_free, Alloc >	816
cxx::Base_slab_static< sizeof(Type), Slab_size, Max_free, Alloc >	702
cxx::Slab_static< Type, Slab_size, Max_free, Alloc >	820
cxx::Bits::Basic_list< Bits::Basic_list_policy< Observer, H_list_item > >	752
cxx::H_list< Observer >	771
cxx::Bits::Basic_list< Bits::Basic_list_policy< Slab_i, H_list_item > >	752
cxx::H_list< Slab_i >	771
cxx::Bits::Basic_list< Bits::Basic_list_policy< T, H_list_item > >	752
cxx::H_list< T >	771
cxx::Bits::Basic_list< Bits::Basic_list_policy< T, H_list_item_t< T > > >	752
cxx::H_list< T, Bits::Basic_list_policy< T, H_list_item_t< T > > >	771
cxx::H_list_t< T >	780
cxx::Bits::Basic_list< Bits::Basic_list_policy< T, S_list_item > >	752
cxx::S_list< T >	814
cxx::Bits::Basic_list< Bits::Basic_list_policy< Timeout, H_list_item > >	752
cxx::H_list< Timeout >	771
cxx::Bits::Basic_list< Bits::Basic_list_policy< Weak_ref_base, H_list_item_t< Weak_ref_base > > > .	752

cxx::H_list< Weak_ref_base, Bits::Basic_list_policy< Weak_ref_base, H_list_item_t< Weak_ref_↵ base > > >	771
cxx::H_list_t< Weak_ref_base >	780
L4::Types::Bool< __lface_conflict< I, I2 >::value _lface_conflict< I, LIST::type >::value >	1183
L4::Types::Bool< false >	1183
L4::Types::False	1184
L4::Types::Same< A, B >	1191
L4::Types::Bool< I1::Proto !=PROTO_EMPTY &&I1::Proto==I2::Proto >	1183
L4::Types::Bool< lface_conflict< I, L2::type >::value _Conflict< L1::type, L2::type >::value >	1183
L4::Types::Bool< true >	1183
L4::Types::True	1193
L4::lpc::Msg::ls_valid_rpc_type< A * >	971
L4::lpc::Msg::ls_valid_rpc_type< T >	971
L4::Types::Bool< Typeid::Conflict< L1::type, L2::type >::value Conflict< L1, LIST... >::value Conflict< L2, LIST... >::value >	1183
L4::Types::Bool< Typeid::lface_conflict< I::type, L::type >::value lface_conflict< I, LIST... >::value >	1183
cxx::Bits::Bst< _Node, Bits::Avl_map_get_key< KEY_TYPE >, COMPARE< KEY_TYPE > >	754
cxx::Avl_tree< _Node, Bits::Avl_map_get_key< KEY_TYPE >, COMPARE< KEY_TYPE > >	691
cxx::Bits::Bst< _Node, Bits::Avl_map_get_key< Region >, cxx::Lt_functor< Region > >	754
cxx::Avl_tree< _Node, Bits::Avl_map_get_key< Region >, cxx::Lt_functor< Region > >	691
cxx::Bits::Bst< _Node, Bits::Avl_set_get_key< ITEM_TYPE >, COMPARE >	754
cxx::Avl_tree< _Node, Bits::Avl_set_get_key< ITEM_TYPE >, COMPARE >	691
cxx::Bits::Bst< _Node, GET_KEY, COMPARE >	754
cxx::Avl_tree< _Node, GET_KEY, COMPARE >	691
cxx::Bits::Bst< Entry, Names_get_key, Lt_functor< typename Names_get_key ::Key_type > >	754
cxx::Avl_tree< Entry, Names_get_key >	691
L4::lpc::Msg::Cnt_val_ops< Detail::_Plain< T >::type, DIR, CLASS >	955
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 >	955
L4::lpc::Msg::Cnt_val_ops< Detail::_Plain< typename _Elem< A const * > ::arg_type >::type, type- name Direction< A const * >::type, typename Class< typename Detail::_Plain< A const * >::type >::type >	955
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 >	955
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 >	955
L4::lpc::Msg::Cnt_val_ops< Detail::_Plain< typename _Elem< String< A, LEN > & > ::arg_type >::type, typename Direction< String< A, LEN > & >::type, typename Class< typename Detail::↵ ::Plain< String< A, LEN > & >::type >::type >	955
L4::lpc::Msg::Cnt_val_ops< Detail::_Plain< typename _Elem< T > ::arg_type >::type, typename Direction< T >::type, typename Class< typename Detail::_Plain< T >::type >::type >	955
cxx::Auto_ptr< T >	679
cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >	699
cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >	702
cxx::Bitfield< T, LSB, MSB >	707
cxx::Bitfield< T, LSB, MSB >::Value_base< TT >	718
cxx::Bitfield< T, LSB, MSB >::Value< TT >	717
cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT >	720
cxx::Bitmap_base	725
cxx::Bitmap< BITS >	721
cxx::Bitmap_base::Bit	736
cxx::Bitmap_base::Char< BITS >	737
cxx::Bitmap_base::Word< BITS >	737

cxx::Bits::Avl_map_get_key< KEY_TYPE >	739
cxx::Bits::Avl_set_get_key< KEY_TYPE >	739
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >	740
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node	750
cxx::Bits::Basic_list< POLICY >	752
cxx::H_list< T, POLICY >	771
cxx::S_list< T, POLICY >	814
cxx::Bits::Bst< Node, Get_key, Compare >	754
cxx::Avl_tree< Node, Get_key, Compare >	691
cxx::Bits::Bst_node	767
cxx::Avl_tree_node	696
cxx::Bits::Direction	769
cxx::H_list_item_t< ELEM_TYPE >	778
L4::lpc_svr::Timeout	1022
cxx::List< D, Alloc >	783
cxx::List< D, Alloc >::Iter	786
cxx::List_alloc	787
cxx::List_item	791
cxx::List_item::Iter	797
cxx::List_item::T_iter< T, Poly >	800
cxx::List_item::T_iter< E >	800
cxx::Lt_functor< Obj >	803
cxx::New_allocator< _Type >	803
cxx::Nothrow	804
cxx::Pair< First, Second >	805
cxx::Pair_first_compare< Cmp, Typ >	806
cxx::Ref_ptr< T, CNT >	808
cxx::static_vector< T, IDX >	823
cxx::String	825
L4Re::Util::Names::Name	1375
L4virtio::Svr::Device_t< Ds_data >	1593
L4virtio::Svr::Block_dev< Ds_data >	1560
L4virtio::Svr::Driver_mem_list_t< Ds_data >	1597
L4virtio::Svr::Driver_mem_region_t< Ds_data >	1604
Elf32_Dyn	832
Elf32_Ehdr	833
Elf32_Phdr	835
Elf32_Shdr	836
Elf32_Sym	837
Elf64_Dyn	838
Elf64_Ehdr	839
Elf64_Phdr	841
Elf64_Shdr	842
Elf64_Sym	843
gfxbitmap_offset	844
cxx::H_list_item_t< Weak_ref_base >	778
cxx::Weak_ref_base	829
cxx::Weak_ref< T >	826
L4::Kobject_2t< Console, Video::Goos, Event, L4::PROTO_EMPTY >	1061
L4Re::Console	1251
L4::Kobject_2t< Debug_obj_t< BASE >, BASE, Debug_obj, L4::PROTO_EMPTY >	1061
L4::Kobject_x< lommu, Proto_t< L4_PROTO_IOMMU >, Type_info::Demand_t< 1 > >	1075
L4::lommu	916
L4::Alloc_list	845
L4::Basic_registry	847

L4Re::Util::Object_registry	1380
L4::Cap_base	858
L4::Cap< L4::Factory >	853
L4::Cap< L4::Irq >	853
L4::Cap< L4::Semaphore >	853
L4::Cap< L4::Thread >	853
L4::Cap< L4::Vcon >	853
L4::Cap< L4Re::Dataspace >	853
L4::Cap< L4Re::Namespace >	853
L4::Cap< L4Re::Rm >	853
L4::Cap< L4Re::Video::Goos >	853
L4::Cap< L4vbus::Vbus >	853
L4::Cap< void >	853
L4::Cap< T >	853
L4::Smart_cap< T, SMART >	1126
L4::Exception_tracer	882
L4::Base_exception	846
L4::Invalid_capability	910
L4::Runtime_error	1101
L4::Bounds_error	850
L4::Com_error	869
L4::Element_already_exists	878
L4::Element_not_found	880
L4::Out_of_memory	1081
L4::Unknown_error	1195
L4::Factory::Lstr	895
L4::Factory::Nil	896
L4::Factory::S	897
L4::Io_pager	914
L4::Kobject_t< Pager, Io_pager, L4_PROTO_PAGE_FAULT >	1070
L4::Pager	1084
L4::Kobject_t< Rm, L4::Pager, L4RE_PROTO_RM, L4::Type_info::Demand_t< 1 > >	1070
L4Re::Rm	1321
L4::IOModifier	918
L4::lpc::Array_in_buf< ELEM_TYPE, LEN_TYPE, MAX >	922
L4::lpc::Array_ref< ELEM_TYPE, LEN_TYPE >	923
L4::lpc::Array< ELEM_TYPE, LEN_TYPE >	919
L4::lpc::As_value< T >	925
L4::lpc::Buf_item	926
L4::lpc::Call	927
L4::lpc::Call_t< RIGHTS >	928
L4::lpc::Call_zero_send_timeout	930
L4::lpc::Cap< T >	931
L4::lpc::Gen_fpage< T >	934
L4::lpc::In_out< T >	938
L4::lpc::Istream	945
L4::lpc::Iostream	938
L4::lpc::Msg::Cnt_val_ops< MTYPE, DIR, CLASS >	955
L4::lpc::Msg::Cls_buffer	956
L4::lpc::Msg::Do_rcv_buffers	966
L4::lpc::Msg::Cls_data	958
L4::lpc::Msg::Do_in_data	962
L4::lpc::Msg::Do_out_data	964
L4::lpc::Msg::Cls_item	959
L4::lpc::Msg::Do_in_items	963

L4::lpc::Msg::Do_out_items	965
L4::lpc::Msg::Dir_in	960
L4::lpc::Msg::Do_in_data	962
L4::lpc::Msg::Do_in_items	963
L4::lpc::Msg::Do_rcv_buffers	966
L4::lpc::Msg::Dir_out	961
L4::lpc::Msg::Do_out_data	964
L4::lpc::Msg::Do_out_items	965
L4::lpc::Msg::Elem< Array< A, LEN > &>	968
L4::lpc::Msg::Elem< Array< A, LEN > >	969
L4::lpc::Msg::Elem< Array_ref< A, LEN > &>	970
L4::lpc::Msg::Svr_arg_pack< IPC_TYPE >	973
L4::lpc::Msg::Svr_val_ops< MTYPE, DIR, CLASS >	973
L4::lpc::Msg_ptr< T >	974
L4::lpc::Opt< T >	975
L4::lpc::Ostream	977
L4::lpc::lostream	938
L4::lpc::Out< T >	983
L4::lpc::Ret_array< T >	984
L4::lpc::Send_only	985
L4::lpc::Small_buf	985
L4::lpc::Snd_item	987
L4::lpc::Str_cp_in< T >	987
L4::lpc::Varg	989
L4::lpc::Varg_list_ref	995
L4::lpc::Varg_list< MAX >	994
L4::lpc_svr::Compound_reply	1004
L4::lpc_svr::Default_loop_hooks	1005
L4Re::Util::Br_manager_hooks	1342
L4::lpc_svr::Default_setup_wait	1006
L4::lpc_svr::Default_timeout	1007
L4::lpc_svr::Default_loop_hooks	1005
L4Re::Util::Br_manager_hooks	1342
L4::lpc_svr::Direct_dispatch< R >	1008
L4::lpc_svr::Exc_dispatch< R, Exc >	1011
L4::lpc_svr::Direct_dispatch< R * >	1010
L4::lpc_svr::Ignore_errors	1012
L4::lpc_svr::Default_loop_hooks	1005
L4Re::Util::Br_manager_hooks	1342
L4Re::Util::Br_manager_timeout_hooks	1343
L4::lpc_svr::Server_iface	1013
L4::lpc_svr::Br_manager_no_buffers	1000
L4::lpc_svr::Default_loop_hooks	1005
L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >	1029
L4Re::Util::Br_manager	1338
L4Re::Util::Br_manager_hooks	1342
L4::lpc_svr::Timeout_queue_hooks< Br_manager_timeout_hooks, Br_manager >	1029
L4Re::Util::Br_manager_timeout_hooks	1343
L4::lpc_svr::Timed_work< HOOKS >	1022
L4::lpc_svr::Timeout_queue	1025
L4::Irq_eoi	1040
L4::Kobject_t< lcu, Irq_eoi, L4_PROTO_IRQ, Type_info::Demand_t< 1 > >	1070
L4::lcu	901
L4::Kobject_t< Event, L4::lcu, L4RE_PROTO_EVENT >	1070
L4Re::Event	1285

L4::Kobject_3t< Vbus, L4Re::Dataspace, L4Re::Inhibitor, L4Re::Event >	1066
L4vbus::Vbus	1527
L4::Kobject_t< Scheduler, Icu, L4_PROTO_SCHEDULER, Type_info::Demand_t< 1 > >	1070
L4::Scheduler	1105
L4::Kobject_t< Vcon, Icu, L4_PROTO_LOG >	1070
L4::Vcon	1198
L4::Kobject_t< Log, L4::Vcon, L4::PROTO_EMPTY >	1070
L4Re::Log	1298
L4::Kobject_t< Triggerable, Irq_eoi, L4_PROTO_IRQ >	1070
L4::Triggerable	1154
L4::Kobject_t< Irq, Triggerable, L4_PROTO_IRQ_SENDER >	1070
L4::Irq	1032
L4::Kobject_t< Irq_mux, Triggerable, L4_PROTO_IRQ_MUX >	1070
L4::Irq_mux	1045
L4::Kobject_t< Semaphore, Triggerable, L4_PROTO_SEMAPHORE >	1070
L4::Semaphore	1110
L4::Kip::Mem_desc	1048
L4::Kobject	1057
L4::Kobject_t< Cmd_control, L4::Kobject, L4::PROTO_ANY, Type_info::Demand_t<> >	1070
L4::Kobject_t< Dataspace, L4::Kobject, L4RE_PROTO_DATASPACE, L4::Type_info::Demand_t< 1 > >	1070
L4Re::Dataspace	1254
L4::Kobject_3t< Vbus, L4Re::Dataspace, L4Re::Inhibitor, L4Re::Event >	1066
L4::Kobject_t< Debug_obj, L4::Kobject, L4RE_PROTO_DEBUG >	1070
L4Re::Debug_obj	1264
L4::Kobject_t< Debugger, Kobject, L4_PROTO_DEBUGGER >	1070
L4::Debugger	871
L4::Kobject_t< Derived, L4::Kobject, PROTO, S_DEMAND >	1070
L4::Kobject_t< Device, L4::Kobject, L4VIRTIO_PROTOCOL, L4::Type_info::Demand_t< 1 > >	1070
L4::Kobject_t< Dma_space, L4::Kobject, PROTO, L4::Type_info::Demand_t< 1 > >	1070
L4::Kobject_t< Exception, L4::Kobject, PROTO, Type_info::Demand_t<> >	1070
L4::Kobject_t< Factory, Kobject, L4_PROTO_FACTORY >	1070
L4::Factory	883
L4::Kobject_t< Mem_alloc, L4::Factory, L4::PROTO_EMPTY >	1070
L4Re::Mem_alloc	1301
L4::Kobject_t< Goos, L4::Kobject, L4RE_PROTO_GOOS >	1070
L4Re::Video::Goos	1440
L4::Kobject_t< Inhibitor, L4::Kobject, L4RE_PROTO_INHIBITOR >	1070
L4Re::Inhibitor	1293
L4::Kobject_3t< Vbus, L4Re::Dataspace, L4Re::Inhibitor, L4Re::Event >	1066
L4::Kobject_t< Io_pager, L4::Kobject, PROTO, Type_info::Demand_t<> >	1070
L4::Kobject_t< Ipc_gate, Kobject, L4_PROTO_KOBJECT, Type_info::Demand_t< 1 > >	1070
L4::Ipc_gate	997
L4::Kobject_t< Irq_eoi, L4::Kobject, PROTO, Type_info::Demand_t<> >	1070
L4::Kobject_t< Meta, Kobject, L4_PROTO_META >	1070
L4::Meta	1077
L4::Kobject_t< Mmio_space, L4::Kobject, L4RE_PROTO_MMIO_SPACE >	1070
L4Re::Mmio_space	1306
L4::Kobject_t< Namespace, L4::Kobject, L4RE_PROTO_NAMESPACE, L4::Type_info::Demand_t< 1 > >	1070
L4Re::Namespace	1310
L4::Kobject_t< Parent, L4::Kobject, L4RE_PROTO_PARENT >	1070
L4Re::Parent	1319
L4::Kobject_t< Platform_control, Kobject, L4_PROTO_PLATFORM_CTL >	1070
L4::Platform_control	1087

L4::Kobject_t< Task, Kobject, L4_PROTO_TASK, Type_info::Demand_t< 2 > >	1070
L4::Task	1130
L4::Kobject_t< Vm, Task, L4_PROTO_VM >	1070
L4::Vm	1205
L4::Kobject_t< Thread, Kobject, L4_PROTO_THREAD, Type_info::Demand_t< 1 > >	1070
L4::Thread	1138
L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >	1061
L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >	1066
L4::Kobject_demand< T >	1069
L4::Kobject_t< Derived, Base, PROTO, S_DEMAND >	1070
L4::Kobject_typeid< T >	1072
L4::Kobject_x< Derived, ARGS >	1075
L4::Poll_timeout_kipclock	1092
L4::Proto_t< P >	1095
L4::Registry_iface	1097
L4Re::Util::Object_registry	1380
L4::Server< LOOP_HOOKS >	1115
L4Re::Util::Registry_server< LOOP_HOOKS >	1388
L4::Server_object	1118
L4::Server_object_x< Derived, IFACE, BASE >	1124
L4::Server_object_t< Kobject >	1120
L4::Irq_handler_object	1042
L4::Server_object_t< IFACE, BASE >	1120
L4::Server_object_x< Derived, IFACE, BASE >	1124
L4::String	1130
L4::Thread::Attr	1149
L4::Thread::Modify_senders	1152
L4::Type_info	1158
L4::Type_info::Demand	1159
L4::Type_info::Demand_t< __l::Max< D1::Caps, D2::Caps >::Res, D1::Flags D2::Flags, __l::Max< D1::Mem, D2::Mem >::Res, __l::Max< D1::Ports, D2::Ports >::Res >	1162
L4::Type_info::Demand_union_t< D1, D2 >	1164
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 >	1162
L4::Type_info::Demand_union_t< Kobject_typeid< T1 >::Demand, Kobject_demand< T2... > >	1164
L4::Type_info::Demand_t<>	1162
L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS >	1162
L4::Typeid::Detail::_Rpc< OPCODE, O, Default_op< R > >::Rpc< Y >	1168
L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... >	1169
L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... >::Rpc< Y >	1170
L4::Typeid::Detail::Rpc_end	1171
L4::Typeid::Detail::_Rpc< L4::Opcode, 0, RPCS... >	1167
L4::Typeid::Rpc< RPCS >	1176
L4::Typeid::Detail::_Rpc< l4_umword_t, 0, ARG... >	1167
L4::Typeid::Rpc_sys< ARG >	1181
L4::Typeid::Detail::_Rpc< OPCODE_TYPE, 0, RPCS... >	1167
L4::Typeid::Rpc_code< OPCODE_TYPE >::F< RPCS >	1179
L4::Typeid::Detail::_Rpc< void, 0, OPERATION >	1167
L4::Typeid::Rpc_nocode< OPERATION >	1173
L4::Typeid::Detail::_Rpc< OPCODE, O, X >	1167
L4::Typeid::P_dispatch< LIST >	1172
L4::Typeid::Raw_ipc< CLASS >	1173

L4::Typeid::Rpc_code< OPCODE_TYPE >	1178
L4::Types::Bool< V >	1183
L4::Types::Flags< BITS_ENUM, UNDERLYING >	1186
l4_buf_regs_t	1208
l4_exc_regs_t	1209
l4_fpage_t	1212
l4_icu_info_t	1212
L4::Icu::Info	909
l4_icu_msi_info_t	1214
l4_kernel_info_mem_desc_t	1215
l4_msg_regs_t	1216
l4_msgtag_t	1217
l4_sched_cpu_set_t	1219
l4_sched_param_t	1221
l4_snd_fpage_t	1222
l4_thread_regs_t	1223
l4_timeout_s	1224
l4_timeout_t	1225
l4_tracebuffer_status_t	1226
l4_tracebuffer_status_window_t	1228
l4_vcon_attr_t	1229
l4_vcpu_ipc_regs_t	1230
l4_vcpu_regs_t	1231
l4_vcpu_state_t	1235
L4vcpu::Vcpu	1536
l4_vhw_descriptor	1237
l4_vhw_entry	1240
l4_vm_svm_vmcb_control_area	1243
l4_vm_svm_vmcb_state_save_area	1243
l4_vm_svm_vmcb_state_save_area_seg	1245
l4_vm_svm_vmcb_t	1245
l4_vm_tz_state	1247
L4Re::Cap_alloc	1247
L4Re::Dataspace::Stats	1263
L4Re::Dma_space	1266
L4Re::Env	1271
L4Re::Event_buffer_t< PAYLOAD >	1289
L4Re::Util::Event_buffer_t< PAYLOAD >	1366
L4Re::Util::Event_buffer_consumer_t< PAYLOAD >	1362
L4Re::Event_buffer_t< PAYLOAD >::Event	1292
L4Re::Ned::Cmd_control	1317
L4Re::Smart_cap_auto< Unmap_flags >	1334
L4Re::Util::Auto_cap< T >	1335
L4Re::Util::Auto_del_cap< T >	1336
L4Re::Util::Cap_alloc_base	1346
L4Re::Util::Counting_cap_alloc< COUNTERTYPE >	1347
L4Re::Util::Dataspace_svr	1352
L4Re::Util::Event_t< PAYLOAD >	1370
L4Re::Util::Item_alloc_base	1374
L4Re::Util::Names::Name_space	1376
L4Re::Util::Ref_cap< T >	1386
L4Re::Util::Ref_del_cap< T >	1387
L4Re::Util::Smart_cap_auto< Unmap_flags >	1392
L4Re::Util::Smart_count_cap< Unmap_flags >	1393
L4Re::Util::Vcon_svr< SVR >	1394
L4Re::Util::Video::Goos_svr	1395
L4Re::Vfs::Directory	1406

L4Re::Vfs::File	1411
L4Re::Vfs::Be_file	1399
L4Re::Vfs::File_system	1413
L4Re::Vfs::Be_file_system	1403
L4Re::Vfs::Fs	1415
L4Re::Vfs::Ops	1425
L4Re::Vfs::Generic_file	1419
L4Re::Vfs::File	1411
L4Re::Vfs::Mman	1424
L4Re::Vfs::Ops	1425
L4Re::Vfs::Regular_file	1428
L4Re::Vfs::File	1411
L4Re::Vfs::Special_file	1433
L4Re::Vfs::File	1411
L4Re::Video::Color_component	1436
L4Re::Video::Goos::Info	1446
L4Re::Video::Pixel_info	1448
L4Re::Video::View	1455
L4Re::Video::View::Info	1460
l4re_aux_t	1463
l4re_ds_stats_t	1464
l4re_elf_aux_mword_t	1464
l4re_elf_aux_t	1465
l4re_elf_aux_vma_t	1466
l4re_env_cap_entry_t	1467
l4re_env_t	1469
l4re_event_t	1470
l4re_video_color_component_t	1471
l4re_video_goos_info_t	1472
l4re_video_pixel_info_t	1473
l4re_video_view_info_t	1474
l4re_video_view_t	1476
l4util_idt_desc_t	1477
l4util_idt_header_t	1478
l4util_mb_addr_range_t	1479
l4util_mb_apm_t	1480
l4util_mb_drive_t	1480
l4util_mb_info_t	1482
l4util_mb_mod_t	1484
l4util_mb_vbe_ctrl_t	1485
l4util_mb_vbe_mode_t	1486
L4vbus::Gpio_module::Pin_slice	1504
L4vbus::Pm< DEC >	1526
l4vbus_device_t	1532
l4vbus_resource_t	1533
L4vcpu::State	1534
L4virtio::Ptr< T >	1554
L4virtio::Svr::Bad_descriptor	1558
L4virtio::Svr::Block_request< Ds_data >	1566
L4virtio::Svr::Data_buffer	1568
L4virtio::Svr::Dev_config	1572
L4virtio::Svr::Dev_features	1583
L4virtio::Svr::Dev_status	1586
L4virtio::Svr::Device_t< DATA >	1593
L4virtio::Svr::Driver_mem_list_t< DATA >	1597
L4virtio::Svr::Driver_mem_region_t< DATA >	1604

L4virtio::Svr::Request_processor	1610
L4virtio::Svr::Virtqueue::Head_desc	1622
L4virtio::Virtqueue	1624
L4virtio::Driver::Virtqueue	1549
L4virtio::Svr::Virtqueue	1615
L4virtio::Virtqueue::Avail	1637
L4virtio::Virtqueue::Avail::Flags	1638
L4virtio::Virtqueue::Desc	1640
L4virtio::Virtqueue::Desc::Flags	1642
L4virtio::Virtqueue::Used	1646
L4virtio::Virtqueue::Used::Flags	1647
L4virtio::Virtqueue::Used_elem	1649
l4virtio_block_config_t	1650
l4virtio_block_header_t	1652
l4virtio_config_hdr_t	1653
l4virtio_config_queue_t	1655
L4vbus::Pm< Device >	1526
L4vbus::Device	1489
L4vbus::Gpio_module	1498
L4vbus::Gpio_pin	1505
L4vbus::Icu	1513
L4vbus::Pci_dev	1515
L4vbus::Pci_host_bridge	1520
L4virtio::Ptr< void >	1554
cxx::Ref_ptr< L4Re::Vfs::File >	808
cxx::Ref_ptr< Mount_tree >	808
L4::lpc::Msg::Svr_val_ops< Array_ref< A, LEN >, Dir_in, CLASS >	973
L4::lpc::Msg::Svr_val_ops< Array_ref< A, LEN >, Dir_out, CLASS >	973
L4::lpc::Msg::Svr_val_ops< L4::lpc::Snd_fpage, Dir_in, CLASS >	973
L4::lpc::Msg::Svr_val_ops< typename _Elem< A * > ::svr_type, typename Direction< A * > ::type, type- name Class< typename Detail::_Plain< A * > ::type > ::type >	973
L4::lpc::Msg::Svr_val_ops< typename _Elem< A > ::svr_type, typename Direction< A > ::type, type- name Class< typename Detail::_Plain< A > ::type > ::type >	973
L4::lpc::Msg::Svr_val_ops< typename _Elem< A const * > ::svr_type, typename Direction< A const *> ::type, typename Class< typename Detail::_Plain< A const *> ::type > ::type >	973
L4::lpc::Msg::Svr_val_ops< typename _Elem< Array_ref< A, LEN > &> ::svr_type, typename Direction< Array_ref< A, LEN > &> ::type, typename Class< typename Detail::_Plain< Array_ref< A, LEN > &> ::type > ::type >	973
L4::lpc::Msg::Svr_val_ops< typename _Elem< Array_ref< A, LEN > > ::svr_type, typename Direction< Array_ref< A, LEN > > ::type, typename Class< typename Detail::_Plain< Array_ref< A, LEN > > ::type > ::type >	973
L4::lpc::Msg::Svr_val_ops< typename _Elem< T > ::svr_type, typename Direction< T > ::type, typename Class< typename Detail::_Plain< T > ::type > ::type >	973
cxx::Bitmap_base::Word< Bits >	737
cxx::Bitmap_base::Word< Size >	737

Chapter 10

Data Structure Index

10.1 Data Structures

Here are the data structures with brief descriptions:

cxx::Auto_ptr< T >	
Smart pointer with automatic deletion	679
cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC >	
AVL tree based associative container	684
cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC >	
AVL set for simple compareable items	688
cxx::Avl_tree< Node, Get_key, Compare >	
A generic AVL tree	691
cxx::Avl_tree_node	
Node of an AVL tree	696
cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >	
Basic slab allocator	699
cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >	
Merged slab allocator (allocators for objects of the same size are merged together)	702
cxx::Bitfield< T, LSB, MSB >	
Definition for a member (part) of a bit field	707
cxx::Bitfield< T, LSB, MSB >::Value< TT >	
Internal helper type	717
cxx::Bitfield< T, LSB, MSB >::Value_base< TT >	
Internal helper type	718
cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT >	
Internal helper type	720
cxx::Bitmap< BITS >	
A static bit map	721
cxx::Bitmap_base	
Basic bitmap abstraction	725
cxx::Bitmap_base::Bit	
A writeable bit in a bitmap	736
cxx::Bitmap_base::Char< BITS >	
Helper abstraction for a byte contained in the bitmap	737
cxx::Bitmap_base::Word< BITS >	
Helper abstraction for a word contained in the bitmap	737
cxx::Bits::Avl_map_get_key< KEY_TYPE >	
Key-getter for Avl_map	739
cxx::Bits::Avl_set_get_key< KEY_TYPE >	
Internal, key-getter for Avl_set nodes	739

cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >	
Internal: AVL set with internally managed nodes	740
cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node	
A smart pointer to a tree item	750
cxx::Bits::Basic_list< POLICY >	
Internal: Common functions for all head-based list implementations	752
cxx::Bits::Bst< Node, Get_key, Compare >	
Basic binary search tree (BST)	754
cxx::Bits::Bst_node	
Basic type of a node in a binary search tree (BST)	767
cxx::Bits::Direction	
The direction to go in a binary search tree	769
cxx::H_list< T, POLICY >	
General double-linked list of unspecified cxx::H_list_item elements	771
cxx::H_list_item_t< ELEM_TYPE >	
Basic element type for a double-linked H_list	778
cxx::H_list_t< T >	
Double-linked list of typed H_list_item_t elements	780
cxx::List< D, Alloc >	
Doubly linked list, with internal allocation	783
cxx::List< D, Alloc >::Iter	
Iterator	786
cxx::List_alloc	
Standard list-based allocator	787
cxx::List_item	
Basic list item	791
cxx::List_item::Iter	
Iterator for a list of ListItem -s	797
cxx::List_item::T_iter< T, Poly >	
Iterator for derived classes from ListItem	800
cxx::Lt_functor< Obj >	
Generic comparator class that defaults to the less-than operator	803
cxx::New_allocator< _Type >	
Standard allocator based on <code>operator new ()</code>	803
cxx::Nothrow	
Helper type to distinguish the <code>operator new</code> version that does not throw exceptions	804
cxx::Pair< First, Second >	
Pair of two values	805
cxx::Pair_first_compare< Cmp, Typ >	
Comparison functor for Pair	806
cxx::Ref_ptr< T, CNT >	
A reference-counting pointer with automatic cleanup	808
cxx::S_list< T, POLICY >	
Simple single-linked list	814
cxx::Slab< Type, Slab_size, Max_free, Alloc >	
Slab allocator for object of type <i>Type</i>	816
cxx::Slab_static< Type, Slab_size, Max_free, Alloc >	
Merged slab allocator (allocators for objects of the same size are merged together)	820
cxx::static_vector< T, IDX >	
Simple encapsulation for a dynamically allocated array	823
cxx::String	
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	825
cxx::Weak_ref< T >	
Typed weak reference to an object of type <i>T</i>	826
cxx::Weak_ref_base	
Generic (base) weak reference to some object	829

Elf32_Dyn	
ELF32 dynamic entry	832
Elf32_Ehdr	
ELF32 header	833
Elf32_Phdr	
ELF32 program header	835
Elf32_Shdr	
ELF32 section header - figure 1-9, page 1-9	836
Elf32_Sym	
ELF32 symbol table entry	837
Elf64_Dyn	
ELF64 dynamic entry	838
Elf64_Ehdr	
ELF64 header	839
Elf64_Phdr	
ELF64 program header	841
Elf64_Shdr	
ELF64 section header	842
Elf64_Sym	
ELF64 symbol table entry	843
gfxbitmap_offset	
Offsets in pmap[] and bmap[]	844
L4::Alloc_list	
A simple list-based allocator	845
L4::Base_exception	
Base class for all exceptions, thrown by the L4Re framework	846
L4::Basic_registry	
This registry returns the corresponding server object based on the label of an lpc_gate	847
L4::Bounds_error	
Access out of bounds	850
L4::Cap< T >	
C++ interface for capabilities	853
L4::Cap_base	
Base class for all kinds of capabilities	858
L4::Com_error	
Error conditions during IPC	869
L4::Debugger	
C++ debugger interface	871
L4::Element_already_exists	
Exception for duplicate element insertions	878
L4::Element_not_found	
Exception for a failed lookup (element not found)	880
L4::Exception_tracer	
Back-trace support for exceptions	882
L4::Factory	
C++ Factory interface to create kernel objects	883
L4::Factory::Lstr	
Special type to add a pascal string into the factory create stream	895
L4::Factory::Nil	
Special type to add a void argument into the factory create stream	896
L4::Factory::S	
Stream class for the create() argument stream	897
L4::lcu	
C++ lcu interface	901
L4::lcu::Info	
This class encapsulates information about an ICU	909
L4::Invalid_capability	
Indicates that an invalid object was invoked	910

L4::lo_pager	
lo_pager interface	914
L4::lommu	
Interface for IO-MMUs used for DMA remapping	916
L4::IOModifier	
Modifier class for the IO stream	918
L4::lpc::Array< ELEM_TYPE, LEN_TYPE >	
Array data type for dynamically sized arrays in RPCs	919
L4::lpc::Array_in_buf< ELEM_TYPE, LEN_TYPE, MAX >	
Server-side copy in buffer for Array	922
L4::lpc::Array_ref< ELEM_TYPE, LEN_TYPE >	
Array reference data type for arrays located in the message	923
L4::lpc::As_value< T >	
Pass the argument as plain data value	925
L4::lpc::Buf_item	
RPC warpper for a receive item	926
L4::lpc::Call	
RPC attribute for a standard RPC call	927
L4::lpc::Call_t< RIGHTS >	
RPC attribute for an RPC call with required rights	928
L4::lpc::Call_zero_send_timeout	
RPC attribute for an RPC call, with zero send timeout	930
L4::lpc::Cap< T >	
Capability type for RPC interfaces (see L4::Cap<T>)	931
L4::lpc::Gen_fpage< T >	
Generic RPC wrapper for L4 flex-pages	934
L4::lpc::In_out< T >	
Mark an argument as in-out argument	938
L4::lpc::Iostream	
Input/Output stream for IPC [un]marshalling	938
L4::lpc::Istream	
Input stream for IPC unmarshalling	945
L4::lpc::Msg::Clnt_val_ops< MTYPE, DIR, CLASS >	
Defines client-side handling of 'MTYPE' as RPC argument	955
L4::lpc::Msg::Cls_buffer	
Marker type for receive buffer values	956
L4::lpc::Msg::Cls_data	
Marker type for data values	958
L4::lpc::Msg::Cls_item	
Marker type for item values	959
L4::lpc::Msg::Dir_in	
Marker type for input values	960
L4::lpc::Msg::Dir_out	
Marker type for output values	961
L4::lpc::Msg::Do_in_data	
Marker for Input data	962
L4::lpc::Msg::Do_in_items	
Marker for Input items	963
L4::lpc::Msg::Do_out_data	
Marker for Output data	964
L4::lpc::Msg::Do_out_items	
Marker for Output items	965
L4::lpc::Msg::Do_rcv_buffers	
Marker for receive buffers	966
L4::lpc::Msg::Elem< Array< A, LEN > &>	
Array as output argument	968
L4::lpc::Msg::Elem< Array< A, LEN > >	
Array as input arguments	969

L4::lpc::Msg::Elem< Array_ref< A, LEN > &>	
Array_ref as output argument	970
L4::lpc::Msg::ls_valid_rpc_type< T >	
Type trait defining a valid RPC parameter type	971
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	973
L4::lpc::Msg::Svr_val_ops< MTYPE, DIR, CLASS >	
Defines server-side handling for MTYPE server arguments	973
L4::lpc::Msg_ptr< T >	
Pointer to an element of type T in an lpc::lstream	974
L4::lpc::Opt< T >	
Attribute for defining an optional RPC argument	975
L4::lpc::Ostream	
Output stream for IPC marshalling	977
L4::lpc::Out< T >	
Mark an argument as a output value in an RPC signature	983
L4::lpc::Ret_array< T >	
Dynamically sized output array of type T	984
L4::lpc::Send_only	
RPC attribute for a send-only RPC	985
L4::lpc::Small_buf	
A receive item for receiving a single capability	985
L4::lpc::Snd_item	
RPC wrapper for a send item	987
L4::lpc::Str_cp_in< T >	
Abstraction for extracting a zero-terminated string from an lpc::lstream	987
L4::lpc::Varg	
Variably sized RPC argument	989
L4::lpc::Varg_list< MAX >	
Self-contained list of variable-sized RPC parameters	994
L4::lpc::Varg_list_ref	
List of variable-sized RPC parameters as received by the server	995
L4::lpc_gate	
The C++ IPC gate interface	997
L4::lpc_svr::Br_manager_no_buffers	
Empty implementation of Server_iface	1000
L4::lpc_svr::Compound_reply	
Mix in for LOOP_HOOKS to always use compound reply and wait	1004
L4::lpc_svr::Default_loop_hooks	
Default LOOP_HOOKS	1005
L4::lpc_svr::Default_setup_wait	
Mix in for LOOP_HOOKS for setup_wait no op	1006
L4::lpc_svr::Default_timeout	
Mix in for LOOP_HOOKS to use a 0 send and a infinite receive timeout	1007
L4::lpc_svr::Direct_dispatch< R >	
Direct disptach helper, for forwarding dispatch calls a registry <i>R</i>	1008
L4::lpc_svr::Direct_dispatch< R * >	
Direct disptach helper, for forwarding dispatch calls via a pointer to a registry <i>R</i>	1010
L4::lpc_svr::Exc_dispatch< R, Exc >	
Dispatch helper wrapping try {} catch {} around the dispatch call	1011
L4::lpc_svr::Ignore_errors	
Mix in for LOOP_HOOKS to ignore IPC errors	1012
L4::lpc_svr::Server_iface	
Interface for server-loop related functions	1013
L4::lpc_svr::Timed_work< HOOKS >	
DEPRECATED	1022

L4::lpc_svr::Timeout	Callback interface for Timeout_queue	1022
L4::lpc_svr::Timeout_queue	Timeout queue to be used in l4re server loop	1025
L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN >	Loop hooks mixin for integrating a timeout queue into the server loop	1029
L4::Irq	C++ Irq interface	1032
L4::Irq_eoi	Interface for sending an acknowledge message to an object	1040
L4::Irq_handler_object	Server object base class for handling IRQ messages	1042
L4::Irq_mux	IRQ multiplexer for shared IRQs	1045
L4::Kip::Mem_desc	Memory descriptors stored in the kernel interface page	1048
L4::Kobject	Base class for all kinds of kernel objects and remote objects, referenced by capabilities	1057
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)	1061
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)	1066
L4::Kobject_demand< T >	Get the combined server-side resource requirements for all type T..	1069
L4::Kobject_t< Derived, Base, PROTO, S_DEMAND >	Helper class to create an L4Re interface class that is derived from a single base class	1070
L4::Kobject_typeid< T >	Meta object for handling access to type information of Kobjects	1072
L4::Kobject_x< Derived, ARGS >	Generic Kobject inheritance template	1075
L4::Meta	Meta interface that shall be implemented by each L4Re object and gives access to the dynamic type information for L4Re objects	1077
L4::Out_of_memory	Exception signalling insufficient memory	1081
L4::Pager	Pager interface including the Io_pager interface	1084
L4::Platform_control	L4 C++ interface for controlling platform-wide properties	1087
L4::Poll_timeout_kipclock	A polling timeout based on the L4Re clock	1092
L4::Proto_t< P >	Data type for defining protocol numbers	1095
L4::Registry_iface	Abstract interface for object registries	1097
L4::Runtime_error	Exception for an abstract runtime error	1101
L4::Scheduler	C++ interface of the Scheduler kernel object	1105
L4::Semaphore	Kernel-provided semaphore object	1110
L4::Server< LOOP_HOOKS >	Basic server loop for handling client requests	1115
L4::Server_object	Abstract server object to be used with L4::Server and L4::Basic_registry	1118

L4::Server_object_t< IFACE, BASE >	
Base class (template) for server implementing server objects	1120
L4::Server_object_x< Derived, IFACE, BASE >	
Helper class to implement p_dispatch based server objects	1124
L4::Smart_cap< T, SMART >	
Smart capability class	1126
L4::String	
A null-terminated string container class	1130
L4::Task	
C++ interface of the Task kernel object	1130
L4::Thread	
C++ L4 kernel thread interface	1138
L4::Thread::Attr	
Thread attributes used for control_commit()	1149
L4::Thread::Modify_senders	
Wrapper class for modifying senders	1152
L4::Triggerable	
Interface that allows an object to be triggered by some source	1154
L4::Type_info	
Dynamic Type Information for L4Re Interfaces	1158
L4::Type_info::Demand	
Data type for expressing the needed receive buffers at the server-side of an interface	1159
L4::Type_info::Demand_t< CAPS, FLAGS, MEM, PORTS >	
Template type statically describing demand of receive buffers	1162
L4::Type_info::Demand_union_t< D1, D2 >	
Template type statically describing the combination of two Demand object	1164
L4::Typeid::Detail::_Rpc< OPCODE, O, X >	
Empty list of RPCs	1167
L4::Typeid::Detail::_Rpc< OPCODE, O, Default_op< R > >::Rpc< Y >	
Find the given RPC in the list	1168
L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... >	
Non-empty list of RPCs	1169
L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... >::Rpc< Y >	
Find the given RPC in the list	1170
L4::Typeid::Detail::Rpc_end	
Internal end-of-list marker	1171
L4::Typeid::P_dispatch< LIST >	
Use for protocol based dispatch stage	1172
L4::Typeid::Raw_ipc< CLASS >	
RPCs list for passing raw incoming IPC to the server object	1173
L4::Typeid::Rpc_nocode< OPERATION >	
List of RPCs of an interface using a single operation without an opcode	1173
L4::Typeid::Rpc< RPCS >	
Standard list of RPCs of an interface	1176
L4::Typeid::Rpc_code< OPCODE_TYPE >	
List of RPCs of an interface using a special opcode type	1178
L4::Typeid::Rpc_code< OPCODE_TYPE >::F< RPCS >	1179
L4::Typeid::Rpc_sys< ARG >	
List of RPCs typically used for kernel interfaces	1181
L4::Types::Bool< V >	
Boolean meta type	1183
L4::Types::False	
False meta value	1184
L4::Types::Flags< BITS_ENUM, UNDERLYING >	
Template for defining typical Flags bitmaps	1186
L4::Types::Same< A, B >	
Compare two data types for equality	1191

L4::Types::True	
True meta value	1193
L4::Unknown_error	
Exception for an unknown condition	1195
L4::Vcon	
C++ L4 Vcon interface	1198
L4::Vm	
Virtual machine	1205
l4_buf_regs_t	
Encapsulation of the buffer-registers block in the UTCB	1208
l4_exc_regs_t	
UTCB structure for exceptions	1209
l4_fpage_t	
L4 flexpage type	1212
l4_icu_info_t	
Info structure for an ICU	1212
l4_icu_msi_info_t	
Info to use for a specific MSI	1214
l4_kernel_info_mem_desc_t	
Memory descriptor data structure	1215
l4_msg_regs_t	
Encapsulation of the message-register block in the UTCB	1216
l4_msgtag_t	
Message tag data structure	1217
l4_sched_cpu_set_t	
CPU sets	1219
l4_sched_param_t	
Scheduler parameter set	1221
l4_snd_fpage_t	
Send-flex-page types	1222
l4_thread_regs_t	
Encapsulation of the thread-control-register block of the UTCB	1223
l4_timeout_s	
Basic timeout specification	1224
l4_timeout_t	
Timeout pair	1225
l4_tracebuffer_status_t	
Trace-buffer status	1226
l4_tracebuffer_status_window_t	
Trace-buffer status window descriptor	1228
l4_vcon_attr_t	
Vcon attribute structure	1229
l4_vcpu_ipc_regs_t	
VCPU message registers	1230
l4_vcpu_regs_t	
VCPU registers	1231
l4_vcpu_state_t	
State of a vCPU	1235
l4_vhw_descriptor	
Virtual hardware devices description	1237
l4_vhw_entry	
Description of a device	1240
l4_vm_svm_vmc_b_control_area	
VMCB structure for SVM VMs	1243
l4_vm_svm_vmc_b_state_save_area	
State save area structure for SVM VMs	1243
l4_vm_svm_vmc_b_state_save_area_seg	
State save area segment selector struct	1245

l4_vm_svm_vmcb_t	Control structure for SVM VMs	1245
l4_vm_tz_state	State structure for TrustZone VMs	1247
L4Re::Cap_alloc	Capability allocator interface	1247
L4Re::Console	Console class	1251
L4Re::Dataspace	Interface for memory-like objects	1254
L4Re::Dataspace::Stats	Information about the dataspace	1263
L4Re::Debug_obj	Debug interface	1264
L4Re::Dma_space	DMA Address Space	1266
L4Re::Env	C++ interface of the initial environment that is provided to an L4 task	1271
L4Re::Event	Event class	1285
L4Re::Event_buffer_t< PAYLOAD >	Event buffer class	1289
L4Re::Event_buffer_t< PAYLOAD >::Event	Event structure used in buffer	1292
L4Re::Inhibitor	Set of inhibitor locks, which inhibit specific actions when held	1293
L4Re::Log	Log interface class	1298
L4Re::Mem_alloc	Memory allocation interface	1301
L4Re::Mmio_space	Interface for memory-like address space accessible via IPC	1306
L4Re::Namespace	Name-space interface	1310
L4Re::Ned::Cmd_control	Direct control interface for Ned	1317
L4Re::Parent	Parent interface	1319
L4Re::Rm	Region map	1321
L4Re::Smart_cap_auto< Unmap_flags >	Helper for Auto_cap and Auto_del_cap	1334
L4Re::Util::Auto_cap< T >	Automatic capability that implements automatic free and unmap of the capability selector	1335
L4Re::Util::Auto_del_cap< T >	Automatic capability that implements automatic free and unmap+delete of the capability selector	1336
L4Re::Util::Br_manager	Buffer-register (BR) manager for L4::Server	1338
L4Re::Util::Br_manager_hooks	Predefined server-loop hooks for a server loop using the Br_manager	1342
L4Re::Util::Br_manager_timeout_hooks	Predefined server-loop hooks for a server with using the Br_manager and a timeout queue	1343
L4Re::Util::Cap_alloc_base	Capability allocator	1346
L4Re::Util::Counting_cap_alloc< COUNTERTYPE >	Reference-counting cap allocator	1347
L4Re::Util::Dataspace_svr	Dataspace server class	1352

L4Re::Util::Event_buffer_consumer_t< PAYLOAD >	
An event buffer consumer	1362
L4Re::Util::Event_buffer_t< PAYLOAD >	
Event_buffer utility class	1366
L4Re::Util::Event_t< PAYLOAD >	
Convenience wrapper for getting access to an event object	1370
L4Re::Util::Item_alloc_base	
Item allocator	1374
L4Re::Util::Names::Name	
Name class	1375
L4Re::Util::Names::Name_space	
Abstract server-side implementation of the L4::Namespace interface	1376
L4Re::Util::Object_registry	
A registry that manages server objects and their attached IPC gates for a single server loop for a specific thread	1380
L4Re::Util::Ref_cap< T >	
Automatic capability that implements automatic free and unmap of the capability selector	1386
L4Re::Util::Ref_del_cap< T >	
Automatic capability that implements automatic free and unmap+delete of the capability selector	1387
L4Re::Util::Registry_server< LOOP_HOOKS >	
A server loop object which has a Object_registry included	1388
L4Re::Util::Smart_cap_auto< Unmap_flags >	
Helper for Auto_cap and Auto_del_cap	1392
L4Re::Util::Smart_count_cap< Unmap_flags >	
Helper for Ref_cap and Ref_del_cap	1393
L4Re::Util::Vcon_svr< SVR >	
Console server template class	1394
L4Re::Util::Video::Goos_svr	
Goos server class	1395
L4Re::Vfs::Be_file	
Boiler plate class for implementing an open file for L4Re::Vfs	1399
L4Re::Vfs::Be_file_system	
Boilerplate class for implementing a L4Re::Vfs::File_system	1403
L4Re::Vfs::Directory	
Interface for a POSIX file that is a directory	1406
L4Re::Vfs::File	
The basic interface for an open POSIX file	1411
L4Re::Vfs::File_system	
Basic interface for an L4Re::Vfs file system	1413
L4Re::Vfs::Fs	
POSIX File-system related functionality	1415
L4Re::Vfs::Generic_file	
The common interface for an open POSIX file	1419
L4Re::Vfs::Mman	
Interface for the POSIX memory management	1424
L4Re::Vfs::Ops	
Interface for the POSIX backends for an application	1425
L4Re::Vfs::Regular_file	
Interface for a POSIX file that provides regular file semantics	1428
L4Re::Vfs::Special_file	
Interface for a POSIX file that provides special file semantics	1433
L4Re::Video::Color_component	
A color component	1436
L4Re::Video::Goos	
A goos	1440
L4Re::Video::Goos::Info	
Information structure of a goos	1446

L4Re::Video::Pixel_info	
Pixel information	1448
L4Re::Video::View	
View	1455
L4Re::Video::View::Info	
Information structure of a view	1460
l4re_aux_t	
Auxiliary descriptor	1463
l4re_ds_stats_t	
Information about the data space	1464
l4re_elf_aux_mword_t	
Auxiliary vector element for a single unsigned data word	1464
l4re_elf_aux_t	
Generic header for each auxiliary vector element	1465
l4re_elf_aux_vma_t	
Auxiliary vector element for a reserved virtual memory area	1466
l4re_env_cap_entry_t	
Entry in the L4Re environment array for the named initial objects	1467
l4re_env_t	
Initial environment data structure	1469
l4re_event_t	
Event structure used in buffer	1470
l4re_video_color_component_t	
Color component structure	1471
l4re_video_goos_info_t	
Goos information structure	1472
l4re_video_pixel_info_t	
Pixel_info structure	1473
l4re_video_view_info_t	
View information structure	1474
l4re_video_view_t	
C representation of a goos view	1476
l4util_idt_desc_t	
IDT entry	1477
l4util_idt_header_t	
Header of an IDT table	1478
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	1479
l4util_mb_apm_t	
APM BIOS info	1480
l4util_mb_drive_t	
Drive Info structure	1480
l4util_mb_info_t	1482
l4util_mb_mod_t	1484
l4util_mb_vbe_ctrl_t	
VBE controller information	1485
l4util_mb_vbe_mode_t	
VBE mode information	1486
L4vbus::Device	
Device on a virtual bus (V-BUS)	1489
L4vbus::Gpio_module	
A Gpio_module groups multiple GPIO pins together	1498
L4vbus::Gpio_module::Pin_slice	
A slice of the pins provided by this module	1504
L4vbus::Gpio_pin	
A GPIO pin	1505

L4vbus::Icu	V-BUS Interrupt controller API (ICU)	1513
L4vbus::Pci_dev	A PCI device	1515
L4vbus::Pci_host_bridge	A Pci host bridge	1520
L4vbus::Pm< DEC >	Power-management API mixin	1526
L4vbus::Vbus	The virtual BUS	1527
l4vbus_device_t	Detailed information about a vbus device	1532
l4vbus_resource_t	Description of a single vbus resource	1533
L4vcpu::State	C++ implementation of state word in the vCPU area	1534
L4vcpu::Vcpu	C++ implementation of the vCPU save state area	1536
L4virtio::Driver::Virtqueue	Driver-side implementation of a Virtqueue	1549
L4virtio::Ptr< T >	Pointer used in virtio descriptors	1554
L4virtio::Svr::Bad_descriptor	Exception used by Queue to indicate descriptor errors	1558
L4virtio::Svr::Block_dev< Ds_data >	Base class for virtio block devices	1560
L4virtio::Svr::Block_request< Ds_data >	A request to read or write data	1566
L4virtio::Svr::Data_buffer	Abstract data buffer	1568
L4virtio::Svr::Dev_config	Abstraction for L4-Virtio device config memory	1572
L4virtio::Svr::Dev_features	Type for device feature bitmap	1583
L4virtio::Svr::Dev_status	Type of the device status register	1586
L4virtio::Svr::Device_t< DATA >	Server-side L4-VIRTIO device stub	1593
L4virtio::Svr::Driver_mem_list_t< DATA >	List of driver memory regions assigned to a single L4-VIRTIO transport instance	1597
L4virtio::Svr::Driver_mem_region_t< DATA >	Region of driver memory, that shall be managed locally	1604
L4virtio::Svr::Request_processor	Encapsulate the state for processing a VIRTIO request	1610
L4virtio::Svr::Virtqueue	Virtqueue implementation for the device	1615
L4virtio::Svr::Virtqueue::Head_desc	VIRTIO request, essentially a descriptor from the available ring	1622
L4virtio::Virtqueue	Low-level Virtqueue	1624
L4virtio::Virtqueue::Avail	Type of available ring, this is read-only for the host	1637
L4virtio::Virtqueue::Avail::Flags	Flags of the available ring	1638
L4virtio::Virtqueue::Desc	Descriptor in the descriptor table	1640
L4virtio::Virtqueue::Desc::Flags	Type for descriptor flags	1642

L4virtio::Virtqueue::Used	
Used ring	1646
L4virtio::Virtqueue::Used::Flags	
Flags for the used ring	1647
L4virtio::Virtqueue::Used_elem	
Type of an element of the used ring	1649
l4virtio_block_config_t	
Device configuration for block devices	1650
l4virtio_block_header_t	
Header structure of a request for a block device	1652
l4virtio_config_hdr_t	
L4-VIRTIO config header, provided in shared data space	1653
l4virtio_config_queue_t	
Queue configuration entry	1655

Chapter 11

File Index

11.1 File List

Here is a list of all documented files with brief descriptions:

pkg/l4re-core/ned/lib/include/ cmd_control	??
pkg/l4re-core/ned/lib/include/ Makefile	??
amd64/l4/sys/ __kip-arch.h	??
amd64/l4/sys/ __vcpu-arch.h	??
amd64/l4/sys/ cache.h	
Cache functions	2000
amd64/l4/sys/ consts.h	
Common L4 constants, amd64 version	2015
amd64/l4/sys/ ktrace_events.h	??
amd64/l4/sys/ l4int.h	
Fixed sized integer types, amd64 version	2102
amd64/l4/sys/ linkage.h	
Linkage	1728
amd64/l4/sys/ segment.h	
Segment handling	1703
amd64/l4/sys/ utcb.h	
UTCB definitions for amd64	2152
amd64/l4/sys/ vm.h	??
amd64/l4/util/ apic.h	
APIC for AMD64	1657
amd64/l4/util/ bitops_arch.h	
Amd64 bit manipulation functions	1734
amd64/l4/util/ cpu.h	
CPU related functions	1742
amd64/l4/util/ idt.h	
IDT related functions	1668
amd64/l4/util/ irq.h	
Some PIC and hardware interrupt related functions	1838
amd64/l4/util/ l4_macros.h	
Main function	1747
amd64/l4/util/ mbi_argv.h	
Command line handling	1750
amd64/l4/util/ perform.h	
Performance Monitoring using P5/P6 Measurement Counters	1671
amd64/l4/util/ port_io.h	
Port I/O functions	1713

amd64/l4/util/ rdtsc.h	
Time stamp counter related functions	1683
amd64/l4/util/ spin.h	
Spinning for amd64	1693
amd64/l4/util/ stack_impl.h	
Stack utilities for amd64	1754
amd64/l4/util/ util.h	
Utilities, amd64 version	1695
amd64/l4f/l4/sys/ ipc.h	??
amd64/l4f/l4/sys/ segment.h	
L4f specific fs/gs manipulation	1700
amd64/l4f/l4/util/ port_io.h	
Port I/O functions	1712
amd64/l4f/l4/util/ setjmp.h	
Inter-thread setjmp/longjmp	1721
arm/l4/sys/ __kip-arch.h	??
arm/l4/sys/ __vcpu-arch.h	??
arm/l4/sys/ atomic.h	
Atomic memory modifications	2185
arm/l4/sys/ cache.h	
Cache functions	1997
arm/l4/sys/ consts.h	
Common L4 constants, arm version	2014
arm/l4/sys/ ktrace_events.h	??
arm/l4/sys/ l4int.h	
Fixed sized integer types, arm version	2102
arm/l4/sys/ linkage.h	
Linkage	1727
arm/l4/sys/ mem_op.h	
Memory access functions (ARM specific)	1729
arm/l4/sys/ thread.h	
ARM-specific thread related definitions	2264
arm/l4/sys/ utcb.h	
UTCB definitions for ARM	2150
arm/l4/sys/ vm.h	
ARM virtualization interface	1732
arm/l4/util/ bitops_arch.h	
ARM specific implementation of bitops functions	1733
arm/l4/util/ cpu.h	
CPU related functions	1741
arm/l4/util/ irq.h	
ARM specific implementation of irq functions	1837
arm/l4/util/ l4_macros.h	
Main function	1746
arm/l4/util/ mbi_argv.h	
Multiboot	1749
arm/l4/util/ stack_impl.h	
Stack utilities, arm version	1753
arm/l4f/l4/sys/ ipc.h	
L4 IPC System Calls, ARM	2080
arm/l4f/l4/sys/ syscall_defs.h	
Syscall entry definitions	1756
contrib/libio-io/l4/io/ io.h	1757
contrib/libio-io/l4/io/ types.h	??
l4/cxx/ alloc.h	
Alloc list	2172
l4/cxx/ arith	??

l4/cxx/ atomic.h	
Atomic template	2186
l4/cxx/ auto_ptr	??
l4/cxx/ avl_map	
AVL map	1767
l4/cxx/ avl_set	
AVL set	1770
l4/cxx/ avl_tree	
AVL tree	1774
l4/cxx/ basic_ostream	
Basic IO stream	1780
l4/cxx/ basic_vector.h	
Basic vector	1784
l4/cxx/ bitfield	??
l4/cxx/ bitmap	??
l4/cxx/ dlist	??
l4/cxx/ exceptions	
Base exceptions	1795
l4/cxx/ hlist	??
l4/cxx/ iostream	
IO Stream	1800
l4/cxx/ iostream.h	??
l4/cxx/ ipc_helper	
IPC helper	1801
l4/cxx/ ipc_server	
IPC server loop	2033
l4/cxx/ ipc_stream	
IPC stream	1803
l4/cxx/ ipc_timeout_queue	??
l4/cxx/ l4iostream	
L4 IO stream	1824
l4/cxx/ l4iostream.h	??
l4/cxx/ l4types.h	
L4 Types	1826
l4/cxx/ list	??
l4/cxx/ list_alloc	??
l4/cxx/ main_thread	
Main thread	1827
l4/cxx/ minmax	??
l4/cxx/ observer	??
l4/cxx/ pair	
Pair implementation	1829
l4/cxx/ ref_ptr	??
l4/cxx/ slab_alloc	??
l4/cxx/ slist	??
l4/cxx/ static_container	??
l4/cxx/ static_vector	??
l4/cxx/ std_alloc	??
l4/cxx/ std_exc_io	
Base exceptions std stream operator	1831
l4/cxx/ std_ops	??
l4/cxx/ string	??
l4/cxx/ string.h	
String	1832
l4/cxx/ thread	
Thread implementation	2141
l4/cxx/ type_list	??
l4/cxx/ type_traits	??

l4/cxx/unique_ptr	??
l4/cxx/utls	??
l4/cxx/weak_ref	??
l4/cxx/bits/bst.h	
AVL tree	1785
l4/cxx/bits/bst_base.h	
AVL tree	1789
l4/cxx/bits/bst_iter.h	
AVL tree	1792
l4/cxx/bits/list_basics.h	??
l4/cxx/bits/type_traits.h	??
l4/irq/irq.h	
IRQ handling routines	1834
l4/l4re_vfs/backend	??
l4/l4re_vfs/vfs.h	??
l4/l4re_vfs/impl/default_ops_impl.h	??
l4/l4re_vfs/impl/ds_util.h	??
l4/l4re_vfs/impl/fd_store.h	??
l4/l4re_vfs/impl/fd_store_impl.h	??
l4/l4re_vfs/impl/ns_fs.h	??
l4/l4re_vfs/impl/ns_fs_impl.h	??
l4/l4re_vfs/impl/ro_file.h	??
l4/l4re_vfs/impl/ro_file_impl.h	??
l4/l4re_vfs/impl/vcon_stream.h	??
l4/l4re_vfs/impl/vcon_stream_impl.h	??
l4/l4re_vfs/impl/vfs_api.h	??
l4/l4re_vfs/impl/vfs_api_impl.h	??
l4/l4re_vfs/impl/vfs_impl.h	??
l4/l4virtio/l4virtio	??
l4/l4virtio/virtio.h	??
l4/l4virtio/virtio_block.h	??
l4/l4virtio/virtqueue	??
l4/l4virtio/server/l4virtio	??
l4/l4virtio/server/virtio	??
l4/l4virtio/server/virtio-block	??
l4/libedid/edid.h	1848
l4/libgfxbitmap/bitmap.h	
Bitmap renderer header file	1849
l4/libgfxbitmap/font.h	
Bitmap font renderer header file	1853
l4/libgfxbitmap/support	
Terminal support functionality	1855
l4/re/cap_alloc	
Abstract capability-allocator interface	1891
l4/re/console	??
l4/re/consts	
Constants	1897
l4/re/consts.h	
Constants	2017
l4/re/dataspace	
Dataspace interface	1898
l4/re/dataspace-sys.h	
Dataspace protocol definition	1901
l4/re/debug	
Debug interface	1902
l4/re/dma_space	1904
l4/re/elf_aux.h	
Auxiliary information for binaries	1906

l4/re/env	
Environment interface	1909
l4/re/env.h	
Environment interface	1911
l4/re/error_helper	
Error helper	1915
l4/re/event	??
l4/re/event-sys.h	??
l4/re/event.h	
Events	1865
l4/re/event_enums.h	??
l4/re/inhibitor	??
l4/re/inhibitor-sys.h	??
l4/re/l4aux.h	
Auxiliary definitions	1930
l4/re/log	
Log interface	1931
l4/re/log-sys.h	
Log protocol definition	1933
l4/re/mem_alloc	
Memory allocator interface	1934
l4/re/mem_alloc-sys.h	
Memory allocator protocol definitions	1936
l4/re/mmio_space	??
l4/re/namespace	
Namespace interface	1938
l4/re/namespace-sys.h	
Namespace protocol definitions	1940
l4/re/parent	
Parent interface	1941
l4/re/parent-sys.h	
Parent protocol definition	1943
l4/re/protocols.h	??
l4/re/rm	
Region mapper interface	1944
l4/re/rm-sys.h	
Region mapper protocol definitions	1949
l4/re/c/dataspace.h	
Data space C interface	1856
l4/re/c/debug.h	
Debug C interface	1859
l4/re/c/dma_space.h	
DMA space C interface	1860
l4/re/c/event.h	
Event C interface	1863
l4/re/c/event_buffer.h	??
l4/re/c/inhibitor.h	??
l4/re/c/log.h	
Log C interface	1867
l4/re/c/mem_alloc.h	
Memory allocator C interface	1869
l4/re/c/namespace.h	
Namespace functions, C interface	1872
l4/re/c/rm.h	
Region map interface, C interface	1874
l4/re/c/util/cap_alloc.h	
Capability allocator C interface	1878

l4/re/c/util/ kumem_alloc.h	
Kumem allocator utility C interface	1879
l4/re/c/util/video/ goos_fb.h	
Framebuffer utility functionality	1881
l4/re/c/video/ colors.h	1882
l4/re/c/video/ goos.h	1885
l4/re/c/video/ view.h	1888
l4/re/impl/ dataspace_impl.h	
Dataspace client stub implementation	1921
l4/re/impl/ mem_alloc_impl.h	
Memory allocator client stub implementation	1923
l4/re/impl/ namespace_impl.h	
Namespace client stub implementation	1925
l4/re/impl/ rm_impl.h	
Region map client stub implementation	1927
l4/re/util/ bitmap_cap_alloc	
Bitmap capability allocator	1950
l4/re/util/ br_manager	??
l4/re/util/ cap	
Capability utility functions	1953
l4/re/util/ cap_alloc	
Capability allocator	1893
l4/re/util/ cap_alloc_impl.h	
Capability allocator implementation	1954
l4/re/util/ counting_cap_alloc	
Reference-counting capability allocator	1956
l4/re/util/ dataspace_svr	??
l4/re/util/ debug	??
l4/re/util/ env_ns	??
l4/re/util/ event	1918
l4/re/util/ event_buffer	??
l4/re/util/ event_svr	??
l4/re/util/ icu_svr	??
l4/re/util/ item_alloc	
Item allocator	1960
l4/re/util/ kumem_alloc	
Kumem allocator helper	1962
l4/re/util/ meta	??
l4/re/util/ name_space_svr	??
l4/re/util/ object_registry	??
l4/re/util/ poll_timeout_kipclock	??
l4/re/util/ region_mapping	
Region handling	1963
l4/re/util/ region_mapping_svr	??
l4/re/util/ region_mapping_svr_2	??
l4/re/util/ vcon_svr	??
l4/re/util/video/ goos_fb	??
l4/re/util/video/ goos_svr	??
l4/re/video/ colors	??
l4/re/video/ goos	??
l4/re/video/ goos-sys.h	
Goos protocol definition	1969
l4/re/video/ view	??
l4/shmc/ shmc.h	
Shared memory library header file	1971
l4/sigma0/ sigma0.h	
Sigma0 interface	1975

l4/sys/__kernel_object_impl.h	
Low-level kernel debugger functions	1979
l4/sys/__kip-32bit.h	??
l4/sys/__kip-64bit.h	??
l4/sys/__ktrace-impl.h	
L4 kernel event tracing	1980
l4/sys/__l4_fpage.h	??
l4/sys/__timeout.h	??
l4/sys/__typeinfo.h	
Type information handling	1983
l4/sys/__vm-svm.h	??
l4/sys/__vm-vmx.h	??
l4/sys/assert.h	
Low-level assert implementation	2175
l4/sys/cache.h	
Cache-consistency functions	1996
l4/sys/capability	
L4::Cap related definitions	2003
l4/sys/compiler.h	
L4 compiler related defines	2007
l4/sys/consts.h	
Common constants	2010
l4/sys/debugger	
The debugger interface specifies common debugging related definitions	2046
l4/sys/debugger.h	
Debugger related definitions	2048
l4/sys/err.h	
Error codes	2055
l4/sys/exception	??
l4/sys/factory	
Common factory related definitions	2057
l4/sys/factory.h	
Common factory related definitions	2061
l4/sys/icu	
Interrupt controller	2067
l4/sys/icu.h	
Interrupt controller	2068
l4/sys/iommu	??
l4/sys/ipc.h	
Common IPC interface	2075
l4/sys/ipc_gate	
The C++ IPC gate interface	2083
l4/sys/ipc_gate.h	
The C IPC gate interface	2085
l4/sys/irq	
C++ Irq interface	2088
l4/sys/irq.h	
C Irq interface	1843
l4/sys/kdebug.h	??
l4/sys/kernel_object.h	
Kernel object system calls	2092
l4/sys/kip	2094
l4/sys/kip.h	
Kernel Info Page access functions	2223
l4/sys/kobject	??
l4/sys/ktrace.h	
L4 kernel event tracing	2097

l4/sys/l4int.h	
Fixed sized integer types, generic version	2100
l4/sys/memdesc.h	
Memory description functions	2104
l4/sys/meta	
Meta interface for getting dynamic type information about objects behind capabilities	2108
l4/sys/pager	
Pager and lo_pager C++ interface	2110
l4/sys/platform_control	
Platform control object	2112
l4/sys/platform_control.h	
Platform control object	2114
l4/sys/scheduler	
Scheduler object functions	2118
l4/sys/scheduler.h	
Scheduler object functions	2120
l4/sys/semaphore	
Semaphore class definition	2124
l4/sys/semaphore.h	??
l4/sys/smart_capability	
L4::Capability class	2126
l4/sys/task	
Common task related definitions	2130
l4/sys/task.h	
Common task related definitions	2133
l4/sys/thread	
Common thread related definitions	2138
l4/sys/thread.h	
Common thread related definitions	2254
l4/sys/typeinfo_svr	
Type information server template	2143
l4/sys/types.h	
Common L4 ABI Data Types	1761
l4/sys/utcb.h	
UTCB definitions	2145
l4/sys/vcon	
Virtual console interface	2158
l4/sys/vcon.h	
Virtual console interface	2160
l4/sys/vcpu.h	??
l4/sys/vhw.h	
Descriptors for virtual hardware (under UX)	2166
l4/sys/vm	
Virtualization interface	2169
l4/sys/cxx/capability.h	??
l4/sys/cxx/ipc_array	??
l4/sys/cxx/ipc_basics	??
l4/sys/cxx/ipc_client	2018
l4/sys/cxx/ipc_epiface	??
l4/sys/cxx/ipc_iface	
Interface Definition Language	2021
l4/sys/cxx/ipc_legacy	??
l4/sys/cxx/ipc_ret_array	??
l4/sys/cxx/ipc_server	??
l4/sys/cxx/ipc_server_loop	??
l4/sys/cxx/ipc_string	??
l4/sys/cxx/ipc_types	2035
l4/sys/cxx/ipc_varg	??

l4/sys/cxx/types	2044
l4/util/alloc.h	
Allocator using a bit-array	2170
l4/util/assert.h	
Some useful assert-style macros	2173
l4/util/atomic.h	
Atomic operations header and generic implementations	2178
l4/util/backtrace.h	
Backtrace	2188
l4/util/base64.h	
Base 64 encoding and decoding functions adapted from Bob Trower 08/04/01	2189
l4/util/bitops.h	
Bit manipulation functions	2191
l4/util/elf.h	
ELF definition	2196
l4/util/getopt.h	
Getopt	2219
l4/util/keymap.h	
Event to ASCII key mapping	2220
l4/util/kip.h	2221
l4/util/kprintf.h	
Printf using the kernel debugger	2226
l4/util/l4_macros.h	
Some useful generic macros, L4f version	1747
l4/util/list_alloc.h	
Simple list-based allocator	2227
l4/util/lulc.h	??
l4/util/lock.h	
Simple lock implementation	2230
l4/util/mb_info.h	
Multiboot info structure as defined by GRUB	2232
l4/util/parse_cmd.h	
Comfortable command-line parsing	2239
l4/util/rand.h	
Simple Pseudo-Random Number Generator	2240
l4/util/reboot.h	
Machine restarting functions	2242
l4/util/sll.h	
List implemenation	2243
l4/util/splitlog2.h	
Split a range in log2 aligned and size-aligned chunks	2247
l4/util/stack.h	
Some helper functions for stack manipulation	2249
l4/util/thread.h	
Low-level Thread Functions	2251
l4/util/util.h	??
l4/vbus/vbus	??
l4/vbus/vbus.h	
Description of the vbus C API	2265
l4/vbus/vbus_generic	??
l4/vbus/vbus_gpio	??
l4/vbus/vbus_gpio-ops.h	??
l4/vbus/vbus_gpio.h	??
l4/vbus/vbus_i2c.h	??
l4/vbus/vbus_inhibitor.h	??
l4/vbus/vbus_interfaces.h	??
l4/vbus/vbus_mcspi.h	??
l4/vbus/vbus_pci	??

l4/vbus/vbus_pci-ops.h	??
l4/vbus/vbus_pci.h	??
l4/vbus/vbus_pm-ops.h	??
l4/vbus/vbus_pm.h	??
l4/vbus/vbus_types.h	
This header file contains descriptions of vbus related data types and constants	2268
l4/vbus/vdevice-ops.h	??
l4/vcpu/vcpu	??
l4/vcpu/vcpu.h	??
l4/vcpu/vmx/vmcs.h	??
x86/l4/sys/__kip-arch.h	??
x86/l4/sys/__vcpu-arch.h	??
x86/l4/sys/cache.h	
Cache functions	2002
x86/l4/sys/consts.h	
Common L4 constants, x86 version	2016
x86/l4/sys/ipc-invoke.h	??
x86/l4/sys/ktrace_events.h	??
x86/l4/sys/l4int.h	
Fixed sized integer types, x86 version	2103
x86/l4/sys/linkage.h	
Linkage	1728
x86/l4/sys/segment.h	
Segment handling	1710
x86/l4/sys/syscall-invoke.h	??
x86/l4/sys/utcb.h	
UTCB definitions for X86	2155
x86/l4/sys/vm.h	??
x86/l4/util/apic.h	
APIC for X86	1662
x86/l4/util/bitops_arch.h	
X86 bit manipulation functions	1737
x86/l4/util/cpu.h	
CPU related functions	1744
x86/l4/util/idt.h	
IDT related functions	1670
x86/l4/util/irq.h	
Some PIC and hardware interrupt related functions	1841
x86/l4/util/l4_macros.h	
Main function	1748
x86/l4/util/mbi_argv.h	
Command line handling	1751
x86/l4/util/perform.h	
Performance Monitoring using P5/P6 Measurement Counters	1677
x86/l4/util/port_io.h	
X86 port I/O	1717
x86/l4/util/rdtsc.h	
Time stamp counter related functions	1688
x86/l4/util/spin.h	
Spinning for x86	1694
x86/l4/util/stack_impl.h	
Stack utilities for x86	1755
x86/l4/util/util.h	
Utilities for x86	1698
x86/l4f/l4/sys/ipc-l42-gcc3-nopic.h	??
x86/l4f/l4/sys/ipc.h	
L4 IPC System Calls, x86	2082

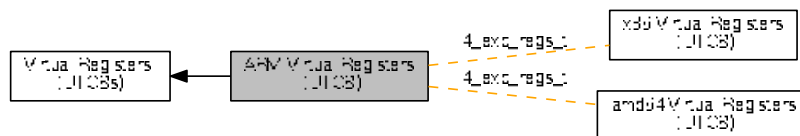
x86/l4f/l4/sys/ segment.h	
L4f specific segment manipulation	1708
x86/l4f/l4/util/ port_io.h	
Port I/O functions	1714
x86/l4f/l4/util/ setjmp.h	
Inter-thread setjmp/longjmp	1724

Chapter 12

Module Documentation

12.1 ARM Virtual Registers (UTCB)

Collaboration diagram for ARM Virtual Registers (UTCB):



Data Structures

- struct [l4_exc_regs_t](#)
UTCB structure for exceptions.

Typedefs

- typedef struct [l4_exc_regs_t](#) [l4_exc_regs_t](#)
UTCB structure for exceptions.

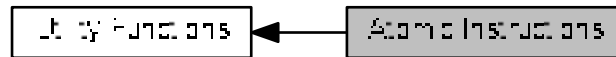
Enumerations

- enum [L4_utcb_consts_arm](#)
UTCB constants for ARM.

12.1.1 Detailed Description

12.2 Atomic Instructions

Collaboration diagram for Atomic Instructions:



Files

- file [atomic.h](#)
atomic operations header and generic implementations

Functions

- int [l4util_cmpxchg64](#) (volatile [l4_uint64_t](#) *dest, [l4_uint64_t](#) cmp_val, [l4_uint64_t](#) new_val)
Atomic compare and exchange (64 bit version)
- 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)

12.2.1 Detailed Description

12.2.2 Function Documentation

12.2.2.1 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 424 of file [atomic.h](#).

12.2.2.2 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 476 of file [atomic.h](#).

12.2.2.3 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 436 of file [atomic.h](#).

12.2.2.4 l4util_atomic_inc()

```
void l4util_atomic_inc (
    volatile long * dest ) [inline]
```

Atomic increment.

Parameters

<i>dest</i>	destination operand
-------------	---------------------

Definition at line 379 of file [atomic.h](#).

12.2.2.5 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

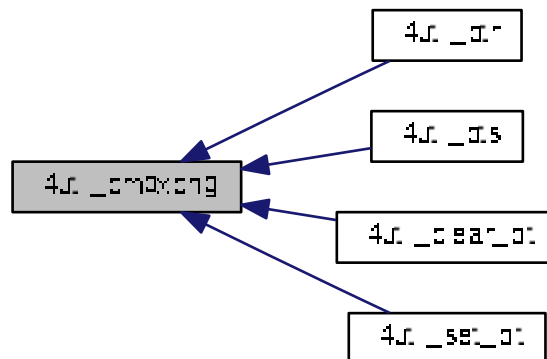
0 if comparison failed, 1 otherwise

Compare the value in *dest* with *cmp_val*, if equal set *dest* to *new_val*

Definition at line 335 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:



12.2.2.6 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 319 of file [atomic.h](#).

12.2.2.7 l4util_cmpxchg32()

```
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 311 of file [atomic.h](#).

12.2.2.8 l4util_cmpxchg64()

```
int l4util_cmpxchg64 (
    volatile l4_uint64_t * dest,
    l4_uint64_t cmp_val,
    l4_uint64_t new_val ) [inline]
```

Atomic compare and exchange (64 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, 1 otherwise

Compare the value in *dest* with *cmp_val*, if equal set *dest* to *new_val*

Definition at line 303 of file [atomic.h](#).

12.2.2.9 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 327 of file [atomic.h](#).

12.2.2.10 l4util_inc8()

```
void l4util_inc8 (
    volatile l4_uint8_t * dest ) [inline]
```

Parameters

<i>dest</i>	destination operand
-------------	---------------------

Definition at line 367 of file [atomic.h](#).

12.2.2.11 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 396 of file [atomic.h](#).

12.2.2.12 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 361 of file [atomic.h](#).

12.2.2.13 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 349 of file [atomic.h](#).

12.2.2.14 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 343 of file [atomic.h](#).

12.2.2.15 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 355 of file [atomic.h](#).

12.3 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

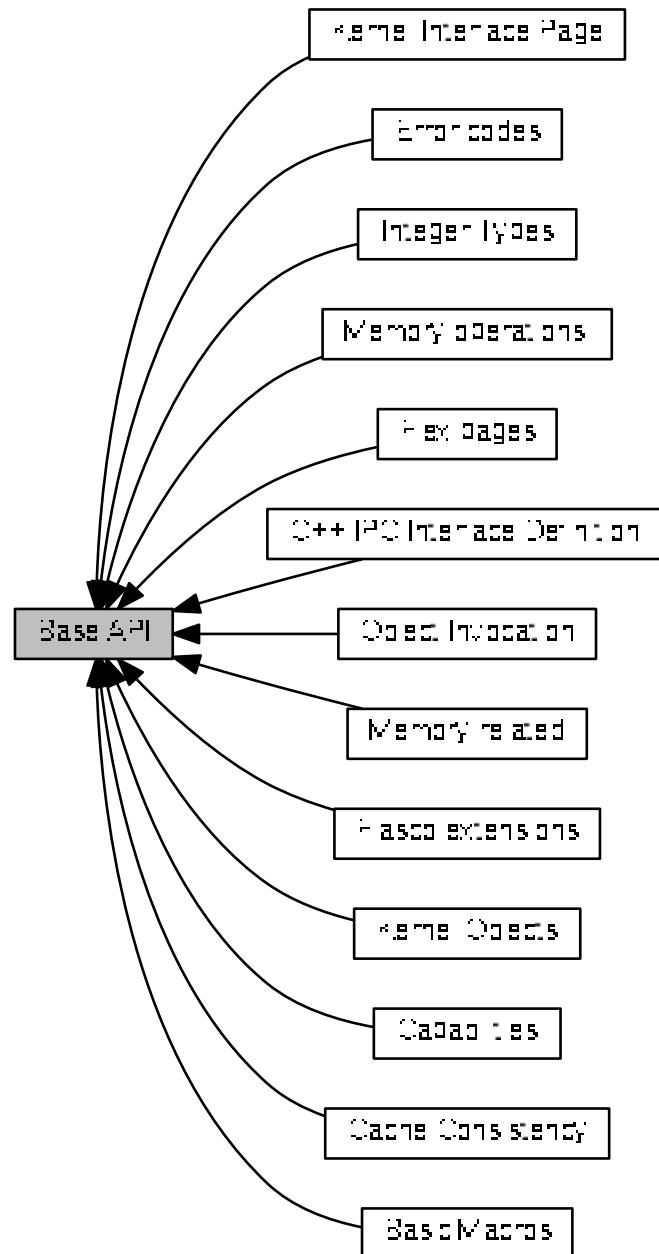
- enum [l4re_aux_ldr_flags_t](#)
Flags for program loading.

12.3.1 Detailed Description

12.4 Base API

Interfaces for all kinds of base functionality.

Collaboration diagram for Base API:



Modules

- [Basic Macros](#)

- [L4 standard macros for header files, function definitions, and public APIs etc.](#)
- [C++ IPC Interface Definition.](#)
APIs for defining IPC interfaces using C++ as language.
- [Cache Consistency](#)
Various functions for cache consistency.
- [Capabilities](#)
C interface for capabilities.
- [Error codes](#)
Common error codes.
- [Fiasco extensions](#)
Kernel debugger extensions of the Fiasco L4 implementation.
- [Flex pages](#)
Flex-page related API.
- [Integer Types](#)
- [Kernel Interface Page](#)
Kernel Interface Page.
- [Kernel Objects](#)
API of kernel objects.
- [Memory operations.](#)
Operations for memory access.
- [Memory related](#)
Memory related constants, data types and functions.
- [Object Invocation](#)
API for L4 object invocation.

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 [types.h](#)
Common L4 ABI Data Types.
- file [vhw.h](#)
Descriptors for virtual hardware (under UX).
- file [consts.h](#)
Common L4 constants, arm version.
- file [consts.h](#)
Common L4 constants, amd64 version.
- file [ipc.h](#)
L4 IPC System Calls, x86.
- file [consts.h](#)
Common L4 constants, x86 version.

12.4.1 Detailed Description

Interfaces for all kinds of base functionality.

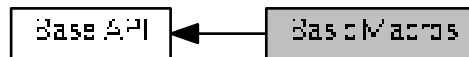
Some notes on Inter Process Communication (IPC)

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.

12.5 Basic Macros

[L4](#) standard macros for header files, function definitions, and public APIs etc.

Collaboration diagram for Basic Macros:



Macros

- `#define L4_ALWAYS_INLINE`
L4 Inline function attribute.
- `#define __END_DECLS`
End section with C types and functions.
- `#define EXTERN_C_BEGIN`
Start section with C types and functions.
- `#define EXTERN_C_END`
End section with C types and functions.
- `#define EXTERN_C`
Mark C types and functions.
- `#define L4_NOTHROW`
Mark a function declaration and definition as never throwing an exception.
- `#define L4_HIDDEN`
Attribute to mark functions, variables, and data types as being explicitly hidden from users of a library.
- `#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 __attribute__((regparm(0)))`
Define calling convention.

Functions

- void [l4_barrier](#) (void)
Memory barrier.
- void [l4_mb](#) (void)
Memory barrier.
- void [l4_wmb](#) (void)
Write memory barrier.

12.5.1 Detailed Description

[L4](#) standard macros for header files, function definitions, and public APIs etc.

Include File

```
#include <l4/sys/compiler.h>
```

12.5.2 Macro Definition Documentation

12.5.2.1 L4_ALWAYS_INLINE

```
#define L4_ALWAYS_INLINE
```

[L4](#) Inline function attribute.

Always inline a function

Definition at line [67](#) of file [compiler.h](#).

12.5.2.2 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 [212](#) of file [compiler.h](#).

12.5.2.3 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()`.

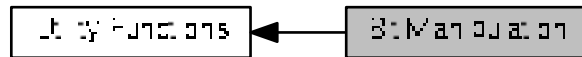
```
int foo() L4_NOTHROW;  
...  
int foo() L4_NOTHROW  
{  
    ...  
    return result;  
}
```

Definition at line 185 of file [compiler.h](#).

Referenced by [l4_msgtag_t::has_error\(\)](#), [l4_round_page\(\)](#), [l4_sleep_forever\(\)](#), [l4_trunc_page\(\)](#), [l4_trunc_size\(\)](#), [l4_vm_vmx_field_len\(\)](#), [l4vcpu_irq_disable_save\(\)](#), and [l4virtio_get_feature\(\)](#).

12.6 Bit Manipulation

Collaboration diagram for Bit Manipulation:



Files

- file [bitops.h](#)
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](#) (const unsigned long val)
Find the next power of 2 for a given number.

12.6.1 Detailed Description

12.6.2 Function Documentation

12.6.2.1 l4util_bsf()

```
int l4util_bsf (
    l4_umword_t word ) [inline]
```

Bit scan forward.

Parameters

<i>word</i>	value (machine size)
-------------	----------------------

Returns

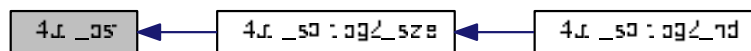
index of least significant bit set in word, -1 if no bit is set (word == 0)

"bit scan forward", find least significant bit set in word.

Definition at line 316 of file [bitops.h](#).

Referenced by [l4util_splitlog2_size\(\)](#).

Here is the caller graph for this function:



12.6.2.2 l4util_bsr()

```
int l4util_bsr (
    l4_umword_t word ) [inline]
```

Bit scan reverse.

Parameters

<i>word</i>	value (machine size)
-------------	----------------------

Returns

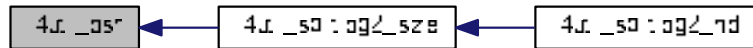
index of most significant set bit in word, -1 if no bit is set (word == 0)

"bit scan reverse", find most significant set bit in word (-> LOG2(word))

Definition at line 299 of file [bitops.h](#).

Referenced by [l4util_splitlog2_size\(\)](#).

Here is the caller graph for this function:



12.6.2.3 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](#).

12.6.2.4 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

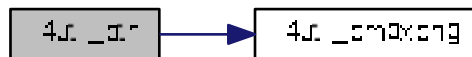
Old value of bit *b*.

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:

**12.6.2.5 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

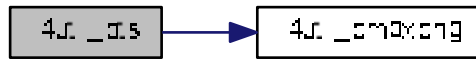
Old value of bit *b*.

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:



12.6.2.6 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:



12.6.2.7 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](#).

12.6.2.8 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 32!)

Returns

number of the first set bit, >= size if no bit is set

Definition at line 400 of file [bitops.h](#).

12.6.2.9 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 32!)

Returns

number of the first zero bit, >= size if no bit is set

Definition at line 333 of file [bitops.h](#).

12.6.2.10 l4util_next_power2()

```
int l4util_next_power2 (
    const 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](#).

12.6.2.11 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:



12.6.2.12 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

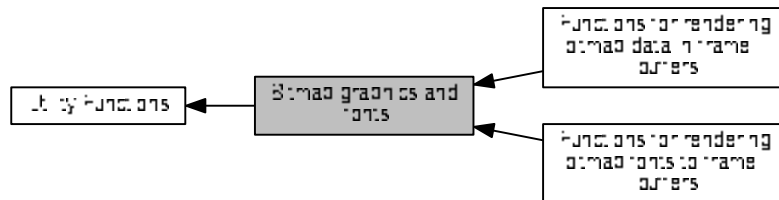
Value of bit *b*.

Definition at line 244 of file [bitops.h](#).

12.7 Bitmap graphics and fonts

This library provides some functions for bitmap handling in frame buffers.

Collaboration diagram for Bitmap graphics and fonts:



Modules

- [Functions for rendering bitmap data in frame buffers](#)
- [Functions for rendering bitmap fonts to frame buffers](#)

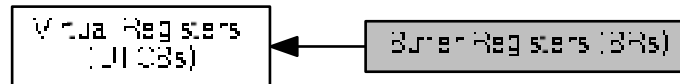
12.7.1 Detailed Description

This library provides some functions for bitmap handling in frame buffers.

Includes simple functions like filling or copying an area of the frame buffer going up to rendering text into the frame buffer using bitmap fonts.

12.8 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 }
Constants for buffer descriptors.

Functions

- void `l4_utcb_inherit_fpu` (int switch_on) `L4_NOTHROW`
Enable or disable inheritance of FPU state to receiver.

12.8.1 Detailed Description

12.8.2 Enumeration Type Documentation

12.8.2.1 `l4_buffer_desc_consts_t`

```
enum l4_buffer_desc_consts_t
```

Constants for buffer descriptors.

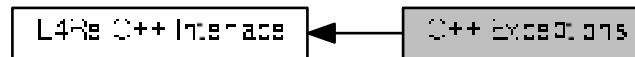
Enumerator

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 219 of file [consts.h](#).

12.9 C++ Exceptions

Collaboration diagram for C++ Exceptions:



Files

- file [exceptions](#)
Base exceptions.
- file [std_exc_io](#)
Base exceptions std stream operator.

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

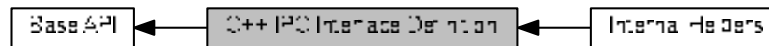
- `#define L4_CXX_EXCEPTION_BACKTRACE 20`
Number of instruction pointers in backtrace.

12.9.1 Detailed Description

12.10 C++ IPC Interface Definition.

APIs for defining IPC interfaces using C++ as language.

Collaboration diagram for C++ IPC Interface Definition.:



Modules

- [Internal Helpers](#)

Namespaces

- [L4::Typeid](#)

Definition of interface data-type helpers.

12.10.1 Detailed Description

APIs for defining IPC interfaces using C++ as language.

12.11 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.

12.11.1 Detailed Description

12.11.2 Function Documentation

12.11.2.1 l4util_cpu_capabilities()

```
unsigned int l4util_cpu_capabilities (
    void ) [inline]
```

Returns the CPU capabilities if the "cpuid" instruction is available.

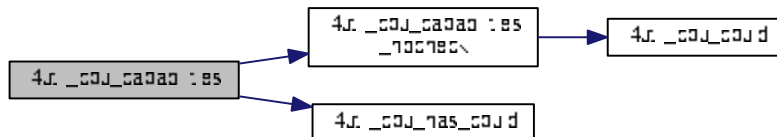
Returns

CPU capabilities if the "cpuid" instruction is available, 0 if the "cpuid" instruction is not supported.

Definition at line 97 of file [cpu.h](#).

References [EXTERN_C_END](#), [l4util_cpu_capabilities_nocheck\(\)](#), and [l4util_cpu_has_cpuid\(\)](#).

Here is the call graph for this function:

**12.11.2.2 l4util_cpu_capabilities_nocheck()**

```
unsigned int l4util_cpu_capabilities_nocheck (
    void ) [inline]
```

Returns the CPU capabilities.

Returns

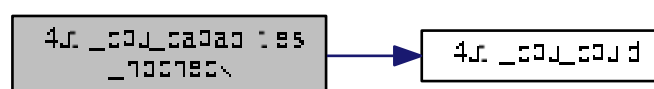
CPU capabilities.

Definition at line 86 of file [cpu.h](#).

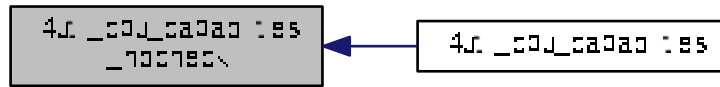
References [l4util_cpu_cpuid\(\)](#).

Referenced by [l4util_cpu_capabilities\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



12.11.2.3 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 66 of file [cpu.h](#).

Referenced by [l4util_cpu_capabilities\(\)](#).

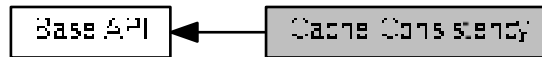
Here is the caller graph for this function:



12.12 Cache Consistency

Various functions for cache consistency.

Collaboration diagram for Cache Consistency:



Functions

- int [l4_cache_clean_data](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)
Cache clean a range in D-cache.
- int [l4_cache_flush_data](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)
Cache flush a range.
- int [l4_cache_inv_data](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)
Cache invalidate a range.
- int [l4_cache_coherent](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)
Make memory coherent between I-cache and D-cache.
- int [l4_cache_dma_coherent](#) (unsigned long start, unsigned long end) [L4_NOTHROW](#)
Make memory coherent for use with external memory.
- int [l4_cache_dma_coherent_full](#) (void) [L4_NOTHROW](#)
Make memory coherent for use with external memory.

12.12.1 Detailed Description

Various functions for cache consistency.

Include File

```
#include <l4/sys/cache.h>
```

12.12.2 Function Documentation

12.12.2.1 l4_cache_clean_data()

```
int l4_cache_clean_data (
    unsigned long start,
    unsigned long end ) [inline]
```

Cache clean a range in D-cache.

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

Examples:

[examples/libs/l4re/c++/shared_ds/ds_clnt.cc](#).

Definition at line [84](#) of file [cache.h](#).

12.12.2.2 l4_cache_coherent()

```
int l4_cache_coherent (
    unsigned long start,
    unsigned long end ) [inline]
```

Make memory coherent between I-cache and D-cache.

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 [108](#) of file [cache.h](#).

12.12.2.3 l4_cache_dma_coherent()

```
int l4_cache_dma_coherent (
    unsigned long start,
    unsigned long end ) [inline]
```

Make memory coherent for use with external memory.

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 116 of file [cache.h](#).

12.12.2.4 l4_cache_flush_data()

```
int l4_cache_flush_data (
    unsigned long start,
    unsigned long end ) [inline]
```

Cache flush a range.

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](#).

12.12.2.5 l4_cache_inv_data()

```
int l4_cache_inv_data (
    unsigned long start,
    unsigned long end ) [inline]
```

Cache invalidate a range.

Parameters

<i>start</i>	Start of range (inclusive)
<i>end</i>	End of range (exclusive)

Return values

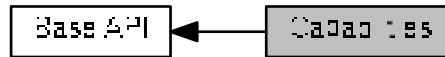
<i>0</i>	on success
<i>-EFAULT</i>	in the case of an unresolved page fault in the given area

Definition at line [100](#) of file [cache.h](#).

12.13 Capabilities

C interface for capabilities.

Collaboration diagram for Capabilities:



Typedefs

- typedef unsigned long [l4_cap_idx_t](#)
L4 Capability selector Type.

Enumerations

- enum [l4_cap_consts_t](#) { [L4_CAP_SHIFT](#), [L4_CAP_SIZE](#), [L4_CAP_MASK](#), [L4_INVALID_CAP](#) }
Constants related to capability selectors.
- 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_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 two capability selectors are equal.

12.13.1 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.

12.13.2 Enumeration Type Documentation

12.13.2.1 l4_cap_consts_t

```
enum l4_cap_consts_t
```

Constants related to capability selectors.

Enumerator

L4_CAP_SHIFT	Capability index shift.
L4_CAP_SIZE	Offset of two consecutive capability selectors.
L4_CAP_MASK	Mask to get only the relevant bits of an l4_cap_idx_t.
L4_INVALID_CAP	Invalid capability selector.

Definition at line 128 of file [consts.h](#).

12.13.2.2 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

This 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.
L4_BASE_LOG_CAP	Capability selector for the log object.
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.
L4_BASE_DEBUGGER_CAP	Capability selector for the debugger cap.
L4_BASE_CAPS_LAST	Last capability index used for base capabilities.

Definition at line 240 of file [consts.h](#).

12.13.3 Function Documentation

12.13.3.1 l4_capability_equal()

```
unsigned l4_capability_equal (
    l4_cap_idx_t c1,
    l4_cap_idx_t c2 ) [inline]
```

Test if two capability selectors are equal.

Parameters

<i>c1</i>	Capability
<i>c2</i>	Capability

Return values

0	The given capabilities are not in the same slot.
1	The given capabilities are in the same slot.

Definition at line 399 of file [types.h](#).

References [L4_CAP_SHIFT](#).

12.13.3.2 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 391 of file [types.h](#).

12.13.3.3 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

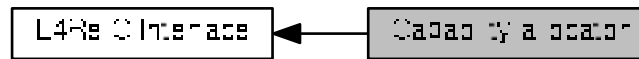
0	The capability selector is not valid.
>0	The capability selector is valid.

Definition at line 395 of file [types.h](#).

12.14 Capability allocator

Capability allocator C interface.

Collaboration diagram for Capability allocator:



Functions

- [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.

12.14.1 Detailed Description

Capability allocator C interface.

12.14.2 Function Documentation

12.14.2.1 l4re_util_cap_last()

```
long l4re_util_cap_last (
    void )
```

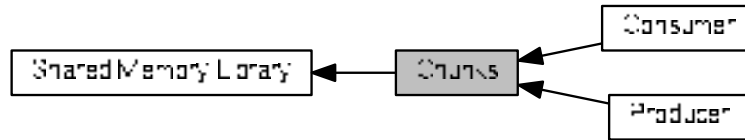
Return last capability index the allocator can return.

Returns

last/biggest capability index the allocator can return

12.15 Chunks

Collaboration diagram for Chunks:



Modules

- [Consumer](#)
- [Producer](#)

Functions

- long [l4shmc_add_chunk](#) (l4shmc_area_t *shmarea, const char *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, const char *chunk_name, l4shmc_chunk_t *chunk)
Get chunk out of shared memory area.
- long [l4shmc_get_chunk_to](#) (l4shmc_area_t *shmarea, const char *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 *shmarea, const char **chunk_name, long offs)
Iterate over names of all existing chunks.
- void * [l4shmc_chunk_ptr](#) (l4shmc_chunk_t *chunk)
Get data pointer to chunk.
- long [l4shmc_chunk_capacity](#) (l4shmc_chunk_t *chunk)
Get capacity of a chunk.
- l4shmc_signal_t * [l4shmc_chunk_signal](#) (l4shmc_chunk_t *chunk)
Get the signal of a chunk.

12.15.1 Detailed Description

12.15.2 Function Documentation

12.15.2.1 l4shmc_add_chunk()

```

long l4shmc_add_chunk (
    l4shmc_area_t * shmarea,
    const char * 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](#).

12.15.2.2 l4shmc_chunk_capacity()

```
long l4shmc_chunk_capacity (
    l4shmc_chunk_t * chunk ) [inline]
```

Get capacity of a chunk.

Parameters

<i>chunk</i>	Chunk.
--------------	--------

Return values

0	Success.
<0	Error.

12.15.2.3 l4shmc_chunk_ptr()

```
void* l4shmc_chunk_ptr (
    l4shmc_chunk_t * chunk ) [inline]
```

Get data pointer to chunk.

Parameters

<i>chunk</i>	Chunk.
--------------	--------

Return values

0	Success.
<0	Error.

Examples:

[examples/libs/shmc/prodcons.c](#).

12.15.2.4 l4shmc_chunk_signal()

```
l4shmc_signal_t* l4shmc_chunk_signal (
    l4shmc_chunk_t * chunk ) [inline]
```

Get the 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.

12.15.2.5 l4shmc_get_chunk()

```
long l4shmc_get_chunk (
    l4shmc_area_t * shmarea,
    const char * chunk_name,
    l4shmc_chunk_t * chunk ) [inline]
```

Get chunk out of shared memory area.

Parameters

	<i>shmarea</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](#).

12.15.2.6 l4shmc_get_chunk_to()

```
long l4shmc_get_chunk_to (
    l4shmc_area_t * shmarea,
    const char * 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.

12.15.2.7 l4shmc_iterate_chunk()

```
long l4shmc_iterate_chunk (
    l4shmc_area_t * shmarea,
    const char ** chunk_name,
    long offs )
```

Iterate over names of all existing chunks.

Parameters

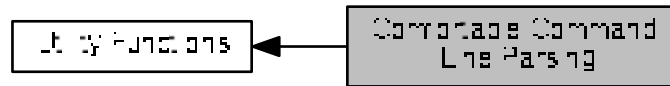
<i>shmarea</i>	Shared memory area.
<i>chunk_name</i>	Where the name of the current chunk will be stored
<i>offs</i>	0 to start iteration, return value of previous call to l4shmc_iterate_chunk() to get next chunk

Return values

0	No more chunks available.
<0	Error.
>0	Iterator value for the next call.

12.16 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

- int [parse_cmdline](#) (int *argc, const char ***argv, char arg0,...)
Parse the command-line for specified arguments and store the values into variables.

12.16.1 Detailed Description

12.16.2 Function Documentation

12.16.2.1 `parse_cmdline()`

```
int parse_cmdline (
    int * argc,
    const char *** argv,
    char 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.

12.17 Console API

[Console](#) interface.

Collaboration diagram for Console API:



Data Structures

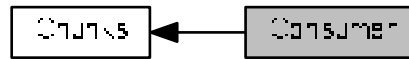
- class [L4Re::Console](#)
Console class.

12.17.1 Detailed Description

[Console](#) interface.

12.18 Consumer

Collaboration diagram for Consumer:



Functions

- 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 *chunk)
Check whether data is available.
- long [l4shmc_chunk_size](#) (l4shmc_chunk_t *chunk)
Get current size of a chunk.

12.18.1 Detailed Description

12.18.2 Function Documentation

12.18.2.1 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](#).

12.18.2.2 l4shmc_chunk_size()

```
long l4shmc_chunk_size (
    l4shmc_chunk_t * chunk ) [inline]
```

Get current size of a chunk.

Parameters

<i>chunk</i>	Chunk.
--------------	--------

Return values

0	Success.
<0	Error.

Examples:

[examples/libs/shmc/prodcons.c](#).

12.18.2.3 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.

12.18.2.4 l4shmc_is_chunk_ready()

```
long l4shmc_is_chunk_ready (
    l4shmc_chunk_t * chunk ) [inline]
```

Check whether data is available.

Parameters

<i>chunk</i>	Chunk to check.
--------------	-----------------

Return values

0	Success.
<0	Error.

12.18.2.5 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

0	Success.
<0	Error.

Examples:

[examples/libs/shmc/prodcons.c](#).

12.18.2.6 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.

12.18.2.7 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.
--------------	-----------------

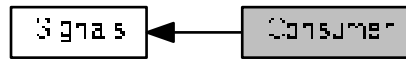
Return values

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.

12.19 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.

12.19.1 Detailed Description

12.19.2 Function Documentation

12.19.2.1 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.

12.19.2.2 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.

12.19.2.3 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.

12.19.2.4 l4shmc_wait_any_try()

```
long l4shmc_wait_any_try (
    l4shmc_signal_t ** retsignal ) [inline]
```

Check whether any waited signal has an event pending.

Parameters

<i>out</i>	<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.

12.19.2.5 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](#).

12.19.2.6 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.

12.19.2.7 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.

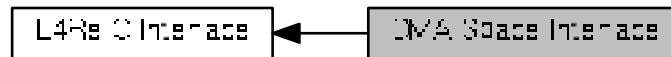
The return code indicates whether an event was pending or not. Success means an event was pending, if an receive timeout error is returned no event was pending.

12.20 DMA API for L4Re

12.21 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

- long [l4re_dma_space_map](#) ([l4re_dma_space_t](#) dma, [l4re_ds_t](#) src, [l4_addr_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.
- long [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.
- long [l4re_dma_space_associate](#) ([l4re_dma_space_t](#) dma, [l4_cap_idx_t](#) dma_task, unsigned long attr) [L4_NOTHROW](#)
Associate a DMA task for a device to this Dma_space.
- long [l4re_dma_space_disassociate](#) ([l4re_dma_space_t](#) dma)
Disassociate the DMA task from this Dma_space.

12.21.1 Detailed Description

DMA Space C interface.

12.21.2 Typedef Documentation

12.21.2.1 l4re_dma_space_t

```
typedef l4_cap_idx_t l4re_dma_space_t
```

DMA space capability type.

DMA Address Space. A `Dma_space` represents the DMA address space of a device. Whenever a device needs direct access to parts of an [L4Re::Dataspace](#), that part of the data space must be mapped to the DMA address space that is assigned to that device. After the DMA accesses to the memory are finished the memory must be unmapped from the device's DMA address space.

Mapping to a 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 59 of file [dma_space.h](#).

12.21.3 Function Documentation

12.21.3.1 l4re_dma_space_associate()

```
long l4re_dma_space_associate (
    l4re_dma_space_t dma,
    l4_cap_idx_t dma_task,
    unsigned long attr )
```

Associate a DMA task for a device to this `Dma_space`.

Parameters

<i>dma</i>	DMA space capability
------------	----------------------

Precondition

requires capability rights: {RW}

Parameters

in	<i>dma_task</i>	The DMA task used for the device that shall be associated with this DMA space. The <code>dma_task</code> might be an invalid capability when <code>#Phys_space</code> 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 <code>#Space_attr</code> .

12.21.3.2 l4re_dma_space_disassociate()

```
long l4re_dma_space_disassociate (
    l4re_dma_space_t dma )
```

Disassociate the DMA task from this Dma_space.

Parameters

<i>dma</i>	DMA space capability
------------	----------------------

Precondition

requires capability rights: {RW}

12.21.3.3 l4re_dma_space_map()

```
long l4re_dma_space_map (
    l4re_dma_space_t dma,
    l4re_ds_t src,
    l4_addr_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
------------	----------------------

Precondition

requires capability rights: {R}

Parameters

in	<i>src</i>	Source data space (that describes the memory). Caller needs write rights 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 of the region to be mapped to for DMA, after successful mapping the size returned is the size mapped for DMA as 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.

Returns

0 in the case of success, a negative error code otherwise.

12.21.3.4 l4re_dma_space_unmap()

```
long 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
------------	----------------------

Precondition

requires capability rights: {R}

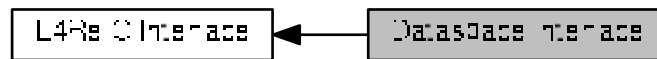
Parameters

<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.

12.22 Dataspace interface

Dataspace C interface.

Collaboration diagram for Dataspace interface:



Data Structures

- struct [l4re_ds_stats_t](#)
Information about the data space.

Typedefs

- typedef [l4_cap_idx_t](#) [l4re_ds_t](#)
Dataspace type.
- typedef [l4_cap_idx_t](#) [l4re_namespace_t](#)
Namespace type.

Enumerations

- enum [l4re_ds_map_flags](#) { ,
[L4RE_DS_MAP_NORMAL](#) = 0x00, [L4RE_DS_MAP_CACHEABLE](#) = [L4RE_DS_MAP_NORMAL](#), [L4RE_DS_MAP_BUFFERABLE](#) = 0x10, [L4RE_DS_MAP_UNCACHEABLE](#) = 0x20,
[L4RE_DS_MAP_CACHING_MASK](#) = 0x30, [L4RE_DS_MAP_CACHING_SHIFT](#) = 4 }
- Flags to specify the memory mapping type of a request.*

Functions

- long [l4re_ds_clear](#) (const [l4re_ds_t](#) ds, [l4_addr_t](#) offset, unsigned long size) [L4_NOTHROW](#)
- long [l4re_ds_allocate](#) (const [l4re_ds_t](#) ds, [l4_addr_t](#) offset, [l4_size_t](#) size) [L4_NOTHROW](#)
- int [l4re_ds_copy_in](#) (const [l4re_ds_t](#) ds, [l4_addr_t](#) dst_offs, const [l4re_ds_t](#) src, [l4_addr_t](#) src_offs, unsigned long size) [L4_NOTHROW](#)
- unsigned long [l4re_ds_size](#) (const [l4re_ds_t](#) ds) [L4_NOTHROW](#)
- long [l4re_ds_flags](#) (const [l4re_ds_t](#) ds) [L4_NOTHROW](#)
- int [l4re_ds_info](#) (const [l4re_ds_t](#) ds, [l4re_ds_stats_t](#) *stats) [L4_NOTHROW](#)
- int [l4re_ds_phys](#) (const [l4re_ds_t](#) ds, [l4_addr_t](#) offset, [l4_addr_t](#) *phys_addr, [l4_size_t](#) *phys_size) [L4_NOTHROW](#)
Return physical address.

12.22.1 Detailed Description

Dataspace C interface.

12.22.2 Enumeration Type Documentation

12.22.2.1 l4re_ds_map_flags

```
enum l4re_ds_map_flags
```

Flags to specify the memory mapping type of a request.

Enumerator

L4RE_DS_MAP_NORMAL	request normal memory mapping
L4RE_DS_MAP_CACHEABLE	request normal memory mapping
L4RE_DS_MAP_BUFFERABLE	request bufferable (write buffered) mappings
L4RE_DS_MAP_UNCACHEABLE	request uncacheable memory mappings
L4RE_DS_MAP_CACHING_MASK	mask for caching flags
L4RE_DS_MAP_CACHING_SHIFT	shift value for caching flags

Definition at line 54 of file [dataspace.h](#).

12.22.3 Function Documentation

12.22.3.1 l4re_ds_allocate()

```
long l4re_ds_allocate (
    const l4re_ds_t ds,
    l4_addr_t offset,
    l4_size_t size )
```

Returns

0 on success, <0 on errors

See also

[L4Re::Dataspace::allocate](#)

12.22.3.2 l4re_ds_clear()

```
long l4re_ds_clear (
    const l4re_ds_t ds,
    l4_addr_t offset,
    unsigned long size )
```

Returns

0 on success, <0 on errors

See also

[L4Re::Dataspace::clear](#)

12.22.3.3 l4re_ds_copy_in()

```
int l4re_ds_copy_in (
    const l4re_ds_t ds,
    l4_addr_t dst_offs,
    const l4re_ds_t src,
    l4_addr_t src_offs,
    unsigned long size )
```

Returns

0 on success, <0 on errors

See also

[L4Re::Dataspace::copy_in](#)

12.22.3.4 l4re_ds_flags()

```
long l4re_ds_flags (
    const l4re_ds_t ds )
```

See also

[L4Re::Dataspace::flags](#)

12.22.3.5 l4re_ds_info()

```
int l4re_ds_info (
    const l4re_ds_t ds,
    l4re_ds_stats_t * stats )
```

See also

[L4Re::Dataspace::info](#)

12.22.3.6 l4re_ds_phys()

```
int l4re_ds_phys (
    const l4re_ds_t ds,
    l4_addr_t offset,
    l4_addr_t * phys_addr,
    l4_size_t * phys_size )
```

Return physical address.

Parameters

<i>ds</i>	Dataspace
<i>offset</i>	Offset in bytes in dataspace

Return values

<i>phys_addr</i>	Physical address
<i>phys_size</i>	Size of physically contiguous region starting from <i>phys_addr</i> (in bytes).

Returns

0 for success, <0 on error

The function returns the physical address of an offset in a dataspace. Use multiple calls of the function to get all physical regions in case of physically non-contiguous dataspace.

See also

[L4Re::Dataspace::phys](#)

12.22.3.7 l4re_ds_size()

```
unsigned long l4re_ds_size (
    const l4re_ds_t ds )
```

Returns

Size of the dataspace in bytes.

See also

[L4Re::Dataspace::size](#)

12.23 Debug interface

Collaboration diagram for Debug interface:



Functions

- long [l4re_debug_obj_debug](#) ([l4_cap_idx_t](#) *srv*, unsigned long *function*) [L4_NOTHROW](#)
Call debug function of [L4Re](#) service.

12.23.1 Detailed Description

12.23.2 Function Documentation

12.23.2.1 l4re_debug_obj_debug()

```
long 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](#)

12.24 Debugging API

Debugging Interface.

Collaboration diagram for Debugging API:



Data Structures

- class [L4Re::Debug_obj](#)
Debug interface.

12.24.1 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.

12.25 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.

12.25.1 Detailed Description

12.25.2 Enumeration Type Documentation

12.25.2.1 Libedid_consts

enum [Libedid_consts](#)

EDID constants.

Enumerator

Libedid_block_size	Size of one EDID block in bytes.
------------------------------------	----------------------------------

Definition at line 23 of file [edid.h](#).

12.25.3 Function Documentation

12.25.3.1 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

12.25.3.2 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

12.25.3.3 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
-------------	---------------------------------

12.25.3.4 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

12.25.3.5 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

12.25.3.6 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
-------------	---------------------------------

Return values

<i>id</i>	Return the PnP id. Must point to 4 bytes.
-----------	---

12.25.3.7 libedid_prefered_resolution()

```
void libedid_prefered_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
-------------	---------------------------------

Return values

<i>w</i>	X resolution of preferred video mode in pixels.
<i>h</i>	Y resolution of preferred video mode in pixels.

12.25.3.8 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

12.25.3.9 libedid_version()

```
unsigned libedid_version (
    const unsigned char * edid )
```

Returns the EDID version number.

Parameters

<i>edid</i>	Pointer to a 128byte EDID block
-------------	---------------------------------

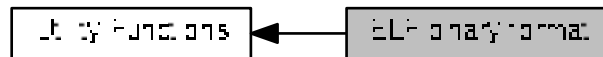
Returns

Version number

12.26 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 - figure 1-9, page 1-9.
- 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_Sym](#)
ELF32 symbol table entry.
- struct [Elf64_Sym](#)
ELF64 symbol table entry.

Macros

- #define `EI_NIDENT` 16
number of characters
- #define `EI_CLASS` 4
ELF class byte index.
- #define `EI_CLASS` 4
ELF class byte index.
- #define `ELFCLASSNONE` 0
Invalid ELF class.
- #define `ELFCLASSNONE` 0
Invalid ELF class.
- #define `ELFCLASS32` 1
32-bit objects
- #define `ELFCLASS64` 2
64-bit objects
- #define `ELFCLASSNUM` 3
Mask for 32-bit or 64-bit class.
- #define `EI_DATA` 5
Data encoding byte index.
- #define `EI_DATA` 5
Data encoding byte index.
- #define `ELFDATANONE` 0
Invalid data encoding.
- #define `ELFDATANONE` 0
Invalid data encoding.
- #define `ELFDATA2LSB` 1
2's complement, little endian
- #define `ELFDATA2LSB` 1
2's complement, little endian
- #define `ELFDATA2MSB` 2
2's complement, big endian
- #define `ELFDATA2MSB` 2
2's complement, big endian
- #define `EI_VERSION` 6
File version byte index.
- #define `EI_VERSION` 6
File version byte index.
- #define `EI_OSABI` 7
OS ABI identification.
- #define `EI_OSABI` 7
OS ABI identification.
- #define `ELFOSABI_NONE` 0
UNIX System V ABI.
- #define `ELFOSABI_SYSV` 0
Alias.
- #define `ELFOSABI_SYSV` 0
Alias.
- #define `ELFOSABI_HPUX` 1
HP-UX.
- #define `ELFOSABI_HPUX` 1

- HP-UX.*
- #define `ELFOSABI_NETBSD` 2
NetBSD.
- #define `ELFOSABI_LINUX` 3
Linux.
- #define `ELFOSABI_SOLARIS` 6
Sun Solaris.
- #define `ELFOSABI_AIX` 7
IBM AIX.
- #define `ELFOSABI_IRIX` 8
SGI Irix.
- #define `ELFOSABI_FREEBSD` 9
FreeBSD.
- #define `ELFOSABI_TRU64` 10
Compaq TRU64 UNIX.
- #define `ELFOSABI_MODESTO` 11
Novell Modesto.
- #define `ELFOSABI_OPENBSD` 12
OpenBSD.
- #define `ELFOSABI_ARM` 97
ARM.
- #define `ELFOSABI_STANDALONE` 255
Standalone (embedded) application.
- #define `ELFOSABI_STANDALONE` 255
Standalone (embedded) application.
- #define `EI_ABIVERSION` 8
ABI version.
- #define `EI_ABIVERSION` 8
ABI version.
- #define `EI_PAD` 9
Byte index of padding bytes.
- #define `EI_PAD` 9
Byte index of padding bytes.
- #define `ET_NONE` 0
no file type
- #define `ET_REL` 1
relocatable file
- #define `ET_EXEC` 2
executable file
- #define `ET_DYN` 3
shared object file
- #define `ET_CORE` 4
core file
- #define `ET_LOPROC` 0xff00
processor-specific
- #define `ET_HIPROC` 0xffff
processor-specific
- #define `EM_NONE` 0
no machine
- #define `EM_M32` 1
AT&T WE 32100.

- #define [EM_SPARC](#) 2
SPARC.
- #define [EM_386](#) 3
Intel 80386.
- #define [EM_68K](#) 4
Motorola 68000.
- #define [EM_88K](#) 5
Motorola 88000.
- #define [EM_860](#) 7
Intel 80860.
- #define [EM_MIPS](#) 8
MIPS RS3000 big-endian.
- #define [EM_MIPS_RS4_BE](#) 10
MIPS RS4000 big-endian.
- #define [EM_SPARC64](#) 11
SPARC 64-bit.
- #define [EM_PARISC](#) 15
HP PA-RISC.
- #define [EM_VPP500](#) 17
Fujitsu VPP500.
- #define [EM_SPARC32PLUS](#) 18
Sun's V8plus.
- #define [EM_960](#) 19
Intel 80960.
- #define [EM_PPC](#) 20
PowerPC.
- #define [EM_V800](#) 36
NEC V800.
- #define [EM_FR20](#) 37
Fujitsu FR20.
- #define [EM_RH32](#) 38
TRW RH-32.
- #define [EM_RCE](#) 39
Motorola RCE.
- #define [EM_ARM](#) 40
Advanced RISC Machines ARM.
- #define [EM_ALPHA](#) 41
Digital Alpha.
- #define [EM_SH](#) 42
Hitachi SuperH.
- #define [EM_SPARCV9](#) 43
SPARC v9 64-bit.
- #define [EM_TRICORE](#) 44
Siemens Tricore embedded processor.
- #define [EM_ARC](#) 45
Argonaut RISC Core, Argonaut Techn Inc.
- #define [EM_H8_300](#) 46
Hitachi H8/300.
- #define [EM_H8_300H](#) 47
Hitachi H8/300H.
- #define [EM_H8S](#) 48

- Hitachi H8/S.*
- #define [EM_H8_500](#) 49
- Hitachi H8/500.*
- #define [EM_IA_64](#) 50
- HP/Intel IA-64.*
- #define [EM_MIPS_X](#) 51
- Stanford MIPS-X.*
- #define [EM_COLDFIRE](#) 52
- Motorola Coldfire.*
- #define [EM_68HC12](#) 53
- Motorola M68HC12.*
- #define [EM_X86_64](#) 62
- Advanced Micro Devices x86-64.*
- #define [EM_PDSP](#) 63
- Sony DSP Processor.*
- #define [EM_FX66](#) 66
- Siemens FX66 microcontroller.*
- #define [EM_ST9PLUS](#) 67
- STMicroelectronics ST9+ 8/16 mc.*
- #define [EM_ST7](#) 68
- STmicroelectronics ST7 8 bit mc.*
- #define [EM_68HC16](#) 69
- Motorola MC68HC16 microcontroller.*
- #define [EM_68HC11](#) 70
- Motorola MC68HC11 microcontroller.*
- #define [EM_68HC08](#) 71
- Motorola MC68HC08 microcontroller.*
- #define [EM_68HC05](#) 72
- Motorola MC68HC05 microcontroller.*
- #define [EM_SVX](#) 73
- Silicon Graphics SVx.*
- #define [EM_ST19](#) 74
- STMicroelectronics ST19 8 bit mc.*
- #define [EM_VAX](#) 75
- Digital VAX.*
- #define [EM_CRIS](#) 76
- Axis Communications 32-bit embedded processor.*
- #define [EM_JAVELIN](#) 77
- Infineon Technologies 32-bit embedded processor.*
- #define [EM_FIREPATH](#) 78
- Element 14 64-bit DSP Processor.*
- #define [EM_ZSP](#) 79
- LSI Logic 16-bit DSP Processor.*
- #define [EM_MMIX](#) 80
- Donald Knuth's educational 64-bit processor.*
- #define [EM_HUANY](#) 81
- Harvard University machine-independent object files.*
- #define [EM_PRISM](#) 82
- SiTera Prism.*
- #define [EM_AVR](#) 83
- Atmel AVR 8-bit microcontroller.*

- #define [EM_FR30](#) 84
Fujitsu FR30.
- #define [EM_D10V](#) 85
Mitsubishi D10V.
- #define [EM_D30V](#) 86
Mitsubishi D30V.
- #define [EM_V850](#) 87
NEC v850.
- #define [EM_M32R](#) 88
Mitsubishi M32R.
- #define [EM_MN10300](#) 89
Matsushita MN10300.
- #define [EM_MN10200](#) 90
Matsushita MN10200.
- #define [EM_PJ](#) 91
picoJava
- #define [EM_OPENRISC](#) 92
OpenRISC 32-bit embedded processor.
- #define [EM_ARC_A5](#) 93
ARC Cores Tangent-A5.
- #define [EM_XTENSA](#) 94
Tensilica Xtensa Architecture.
- #define [EM_ALTERA_NIOS2](#) 113
Altera Nios II.
- #define [EM_AARCH64](#) 183
ARM AARCH64.
- #define [EM_TILEPRO](#) 188
Tilera TILEPro.
- #define [EM_MICROBLAZE](#) 189
Xilinx MicroBlaze.
- #define [EM_TILEGX](#) 191
Tilera TILE-Gx.
- #define [EV_NONE](#) 0
Invalid version.
- #define [EV_CURRENT](#) 1
Current version.
- #define [EI_MAG0](#) 0
file id
- #define [EI_MAG1](#) 1
file id
- #define [EI_MAG2](#) 2
file id
- #define [EI_MAG3](#) 3
file id
- #define [ELFMAG0](#) 0x7f
e_ident[EI_MAG0]
- #define [ELFMAG1](#) 'E'
e_ident[EI_MAG1]
- #define [ELFMAG2](#) 'L'
e_ident[EI_MAG2]
- #define [ELFMAG3](#) 'F'

- `e_ident[EI_MAG3]`
- #define `ELFCLASS32` 1
 - 32-bit object*
- #define `ELFCLASS64` 2
 - 64-bit object*
- #define `SHN_UNDEF` 0
 - undefined section header entry*
- #define `SHN_LORESERVE` 0xff00
 - lower bound of reserved indexes*
- #define `SHN_LOPROC` 0xff00
 - lower bound of proc spec entr*
- #define `SHN_HIPROC` 0xff1f
 - upper bound of proc spec entr*
- #define `SHN_ABS` 0xffff
 - absolute values for ref*
- #define `SHN_COMMON` 0xffff2
 - common symbols*
- #define `SHN_HIRESERVE` 0xffff
 - upper bound of reserved indexes*
- #define `SHT_INIT_ARRAY` 14
 - Array of constructors.*
- #define `SHT_FINI_ARRAY` 15
 - Array of destructors.*
- #define `SHT_PREINIT_ARRAY` 16
 - Array of pre-constructors.*
- #define `SHT_GROUP` 17
 - Section group.*
- #define `SHT_SYMTAB_SHNDX` 18
 - Extended section indeces.*
- #define `SHT_NUM` 19
 - Number of defined types.*
- #define `SHF_WRITE` 0x1
 - writable during execution*
- #define `SHF_ALLOC` 0x2
 - section occupies virt memory*
- #define `SHF_EXECINSTR` 0x4
 - code section*
- #define `SHF_MERGE` 0x10
 - Might be merged.*
- #define `SHF_STRINGS` 0x20
 - Contains nul-terminated strings.*
- #define `SHF_INFO_LINK` 0x40
 - 'sh_info' contains SHT index*
- #define `SHF_LINK_ORDER` 0x80
 - Preserve order after combining.*
- #define `SHF_OS_NONCONFORMING` 0x100
 - Non-standard OS specific handling required.*
- #define `SHF_GROUP` 0x200
 - Section is member of a group.*
- #define `SHF_TLS` 0x400
 - Section hold thread-local data.*

- #define SHF_MASKOS 0x0ff00000
OS-specific.
- #define SHF_MASKPROC 0xf0000000
proc spec mask
- #define PT_NULL 0
array is unused
- #define PT_LOAD 1
loadable
- #define PT_DYNAMIC 2
dynamic linking information
- #define PT_INTERP 3
path to interpreter
- #define PT_NOTE 4
auxiliary information
- #define PT_SHLIB 5
reserved
- #define PT_PHDR 6
location of the pht itself
- #define PT_TLS 7
Thread-local storage segment.
- #define PT_NUM 8
Number of defined types.
- #define PT_LOOS 0x60000000
os spec.
- #define PT_HIOS 0x6fffffff
os spec.
- #define PT_LOPROC 0x70000000
processor spec.
- #define PT_HIPROC 0x7fffffff
processor spec.
- #define PT_GNU_EH_FRAME (PT_LOOS + 0x474e550)
EH frame information.
- #define PT_GNU_STACK (PT_LOOS + 0x474e551)
Flags for stack.
- #define PT_GNU_RELRO (PT_LOOS + 0x474e552)
Read only after reloc.
- #define PT_L4_STACK (PT_LOOS + 0x12)
Address of the stack.
- #define PT_L4_KIP (PT_LOOS + 0x13)
Address of the KIP.
- #define PT_L4_AUX (PT_LOOS + 0x14)
Address of the AUX strcutures.
- #define NT_PRSTATUS 1
Contains copy of prstatus struct.
- #define NT_FPREGSET 2
Contains copy of fpregset struct.
- #define NT_PRPSINFO 3
Contains copy of prpsinfo struct.
- #define NT_PRXREG 4
Contains copy of prxregset struct.
- #define NT_TASKSTRUCT 4

- Contains copy of task structure.*
- #define **NT_PLATFORM** 5
 - String from sysinfo(SI_PLATFORM)*
- #define **NT_AUXV** 6
 - Contains copy of auxv array.*
- #define **NT_GWINDOWS** 7
 - Contains copy of gwindows struct.*
- #define **NT_ASRS** 8
 - Contains copy of asrset struct.*
- #define **NT_PSTATUS** 10
 - Contains copy of pstatus struct.*
- #define **NT_PSINFO** 13
 - Contains copy of psinfo struct.*
- #define **NT_PRCRED** 14
 - Contains copy of prcred struct.*
- #define **NT_UTSNAME** 15
 - Contains copy of utsname struct.*
- #define **NT_LWPSTATUS** 16
 - Contains copy of lwpstatus struct.*
- #define **NT_LWPSINFO** 17
 - Contains copy of lwpsinfo struct.*
- #define **NT_PRFPXREG** 20
 - Contains copy of fprxregset struct.*
- #define **NT_VERSION** 1
 - Contains a version string.*
- #define **DT_NULL** 0
 - Dynamic Array Tags, d_tag - figure 2-10, page 2-12.*
- #define **DT_NEEDED** 1
 - name of a needed library*
- #define **DT_PLTRELSZ** 2
 - total size of relocation entry*
- #define **DT_PLTGOT** 3
 - address assoc with prog link table*
- #define **DT_HASH** 4
 - address of symbol hash table*
- #define **DT_STRTAB** 5
 - address of string table*
- #define **DT_SYMTAB** 6
 - address of symbol table*
- #define **DT_RELA** 7
 - address of relocation table*
- #define **DT_RELASZ** 8
 - total size of relocation table*
- #define **DT_RELAENT** 9
 - size of DT_RELA relocation entry*
- #define **DT_STRSZ** 10
 - size of the string table*
- #define **DT_SYMENT** 11
 - size of a symbol table entry*
- #define **DT_INIT** 12
 - address of initialization function*

- `#define DT_FINI 13`
address of termination function
- `#define DT_SONAME 14`
name of the shared object
- `#define DT_RPATH 15`
search library path
- `#define DT_SYMBOLIC 16`
alter symbol resolution algorithm
- `#define DT_REL 17`
address of relocation table
- `#define DT_RELSZ 18`
total size of DT_REL relocation table
- `#define DT_RELENT 19`
size of the DT_REL relocation entry
- `#define DT_PTRREL 20`
type of relocation entry
- `#define DT_DEBUG 21`
for debugging purposes
- `#define DT_TEXTREL 22`
at least on entry changes r/o section
- `#define DT_JMPREL 23`
address of relocation entries
- `#define DT_BIND_NOW 24`
Process relocations of object.
- `#define DT_INIT_ARRAY 25`
Array with addresses of init fct.
- `#define DT_FINI_ARRAY 26`
Array with addresses of fini fct.
- `#define DT_INIT_ARRAYSZ 27`
Size in bytes of DT_INIT_ARRAY.
- `#define DT_FINI_ARRAYSZ 28`
Size in bytes of DT_FINI_ARRAY.
- `#define DT_RUNPATH 29`
Library search path.
- `#define DT_FLAGS 30`
Flags for the object being loaded.
- `#define DT_ENCODING 32`
Start of encoded range.
- `#define DT_PREINIT_ARRAY 32`
Array with addresses of preinit fct.
- `#define DT_PREINIT_ARRAYSZ 33`
size in bytes of DT_PREINIT_ARRAY
- `#define DT_NUM 34`
Number used.
- `#define DT_LOOS 0x6000000d`
Start of OS-specific.
- `#define DT_HIOS 0x6ffff000`
End of OS-specific.
- `#define DT_LOPROC 0x70000000`
processor spec.
- `#define DT_HIPROC 0x7fffffff`

- processor spec.*

 - #define `DF_ORIGIN` 0x00000001
 - Object may use DF_ORIGIN.*
 - #define `DF_SYMBOLIC` 0x00000002
 - Symbol resolutions starts here.*
 - #define `DF_TEXTREL` 0x00000004
 - Object contains text relocations.*
 - #define `DF_BIND_NOW` 0x00000008
 - No lazy binding for this object.*
 - #define `DF_STATIC_TLS` 0x00000010
 - Module uses the static TLS model.*
 - #define `DF_1_NOW` 0x00000001
 - Set RTLD_NOW for this object.*
 - #define `DF_1_GLOBAL` 0x00000002
 - Set RTLD_GLOBAL for this object.*
 - #define `DF_1_GROUP` 0x00000004
 - Set RTLD_GROUP for this object.*
 - #define `DF_1_NODELETE` 0x00000008
 - Set RTLD_NODELETE for this object.*
 - #define `DF_1_LOADFLTR` 0x00000010
 - Trigger filtee loading at runtime.*
 - #define `DF_1_INITFIRST` 0x00000020
 - Set RTLD_INITFIRST for this object.*
 - #define `DF_1_NOOPEN` 0x00000040
 - Set RTLD_NOOPEN for this object.*
 - #define `DF_1_ORIGIN` 0x00000080
 - \$ORIGIN must be handled.*
 - #define `DF_1_DIRECT` 0x00000100
 - Direct binding enabled.*
 - #define `DF_1_INTERPOSE` 0x00000400
 - Object is used to interpose.*
 - #define `DF_1_NODEFLIB` 0x00000800
 - Ignore default lib search path.*
 - #define `DF_1_NODUMP` 0x00001000
 - Object can't be dldump'ed.*
 - #define `DF_1_CONFALT` 0x00002000
 - Configuration alternative created.*
 - #define `DF_1_ENDFILTEE` 0x00004000
 - Filtee terminates filters search.*
 - #define `DF_1_DISPRELDNE` 0x00008000
 - Disp reloc applied at build time.*
 - #define `DF_1_DISPRELPND` 0x00010000
 - Disp reloc applied at run-time.*
 - #define `DF_P1_LAZYLOAD` 0x00000001
 - Lazyload following object.*
 - #define `DF_P1_GROUPEX` 0x00000002
 - Symbols from next object are not generally available.*
 - #define `R_386_NONE` 0
 - none*
 - #define `R_386_32` 1
 - S + A.*

- #define [R_386_PC32](#) 2
S + A - P
- #define [R_386_GOT32](#) 3
G + A - P.
- #define [R_386_PLT32](#) 4
L + A - P.
- #define [R_386_COPY](#) 5
none
- #define [R_386_GLOB_DAT](#) 6
S.
- #define [R_386_JMP_SLOT](#) 7
S.
- #define [R_386_RELATIVE](#) 8
B + A.
- #define [R_386_GOTOFF](#) 9
S + A - GOT.
- #define [R_386_GOTPC](#) 10
GOT + A - P.
- #define [STB_LOCAL](#) 0
not visible outside object file
- #define [STB_GLOBAL](#) 1
visible to all objects beeing combined
- #define [STB_WEAK](#) 2
resemble global symbols
- #define [STB_LOOS](#) 10
os specific
- #define [STB_HIOS](#) 12
os specific
- #define [STB_LOPROC](#) 13
proc specific
- #define [STB_HIPROC](#) 15
proc specific
- #define [STT_NOTYPE](#) 0
symbol's type not specified
- #define [STT_OBJECT](#) 1
associated with a data object
- #define [STT_FUNC](#) 2
associated with a function or other code
- #define [STT_SECTION](#) 3
associated with a section
- #define [STT_FILE](#) 4
source file name associated with object
- #define [STT_LOOS](#) 10
os specific
- #define [STT_HIOS](#) 12
os specific
- #define [STT_LOPROC](#) 13
proc specific
- #define [STT_HIPROC](#) 15
proc specific

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

12.26.1 Detailed Description

Functions and types related to ELF binaries.

12.26.2 Macro Definition Documentation

12.26.2.1 DF_1_CONFALT

```
#define DF_1_CONFALT 0x00002000
```

Configuration alternative created.

Definition at line [581](#) of file [elf.h](#).

12.26.2.2 DF_1_DIRECT

```
#define DF_1_DIRECT 0x00000100
```

Direct binding enabled.

Definition at line 576 of file [elf.h](#).

12.26.2.3 DF_1_DISPRELDNE

```
#define DF_1_DISPRELDNE 0x00008000
```

Disp reloc applied at build time.

Definition at line 583 of file [elf.h](#).

12.26.2.4 DF_1_DISPRELPND

```
#define DF_1_DISPRELPND 0x00010000
```

Disp reloc applied at run-time.

Definition at line 584 of file [elf.h](#).

12.26.2.5 DF_1_ENDFILTEE

```
#define DF_1_ENDFILTEE 0x00004000
```

Filtee terminates filters search.

Definition at line 582 of file [elf.h](#).

12.26.2.6 DF_1_GLOBAL

```
#define DF_1_GLOBAL 0x00000002
```

Set RTLD_GLOBAL for this object.

Definition at line 569 of file [elf.h](#).

12.26.2.7 DF_1_GROUP

```
#define DF_1_GROUP 0x00000004
```

Set RTLD_GROUP for this object.

Definition at line 570 of file [elf.h](#).

12.26.2.8 DF_1_INTERPOSE

```
#define DF_1_INTERPOSE 0x00000400
```

Object is used to interpose.

Definition at line 578 of file [elf.h](#).

12.26.2.9 DF_1_LOADFLTR

```
#define DF_1_LOADFLTR 0x00000010
```

Trigger filtee loading at runtime.

Definition at line 572 of file [elf.h](#).

12.26.2.10 DF_1_NODEFLIB

```
#define DF_1_NODEFLIB 0x00000800
```

Ignore default lib search path.

Definition at line 579 of file [elf.h](#).

12.26.2.11 DF_1_NODELETE

```
#define DF_1_NODELETE 0x00000008
```

Set RTLD_NODELETE for this object.

Definition at line 571 of file [elf.h](#).

12.26.2.12 DF_1_NODUMP

```
#define DF_1_NODUMP 0x00001000
```

Object can't be dldump'ed.

Definition at line 580 of file [elf.h](#).

12.26.2.13 DF_1_NOOPEN

```
#define DF_1_NOOPEN 0x00000040
```

Set RTLD_NOOPEN for this object.

Definition at line 574 of file [elf.h](#).

12.26.2.14 DF_1_NOW

```
#define DF_1_NOW 0x00000001
```

Set RTLD_NOW for this object.

Definition at line 568 of file [elf.h](#).

12.26.2.15 DF_1_ORIGIN

```
#define DF_1_ORIGIN 0x00000080
```

\$ORIGIN must be handled.

Definition at line 575 of file [elf.h](#).

12.26.2.16 DF_P1_GROUPPERM

```
#define DF_P1_GROUPPERM 0x00000002
```

Symbols from next object are not generally available.

Definition at line 592 of file [elf.h](#).

12.26.2.17 DF_P1_LAZYLOAD

```
#define DF_P1_LAZYLOAD 0x00000001
```

Lazyload following object.

Definition at line 591 of file [elf.h](#).

12.26.2.18 DT_HIPROC

```
#define DT_HIPROC 0x7fffffff
```

processor spec.

Definition at line 557 of file [elf.h](#).

12.26.2.19 DT_LOPROC

```
#define DT_LOPROC 0x70000000
```

processor spec.

Definition at line 556 of file [elf.h](#).

12.26.2.20 DT_NULL

```
#define DT_NULL 0
```

Dynamic Array Tags, d_tag - figure 2-10, page 2-12.

end of _DYNAMIC array

Definition at line 519 of file [elf.h](#).

12.26.2.21 EI_CLASS [1/2]

```
#define EI_CLASS 4
```

ELF class byte index.

file class

Definition at line 294 of file [elf.h](#).

12.26.2.22 EI_CLASS [2/2]

```
#define EI_CLASS 4
```

ELF class byte index.

file class

Definition at line 294 of file [elf.h](#).

12.26.2.23 EI_DATA [1/2]

```
#define EI_DATA 5
```

Data encoding byte index.

data encoding

Definition at line 295 of file [elf.h](#).

12.26.2.24 EI_DATA [2/2]

```
#define EI_DATA 5
```

Data encoding byte index.

data encoding

Definition at line 295 of file [elf.h](#).

12.26.2.25 EI_OSABI [1/2]

```
#define EI_OSABI 7
```

OS ABI identification.

Operating system / ABI identification.

Definition at line 297 of file [elf.h](#).

12.26.2.26 EI_OSABI [2/2]

```
#define EI_OSABI 7
```

OS ABI identification.

Operating system / ABI identification.

Definition at line 297 of file [elf.h](#).

12.26.2.27 EI_PAD [1/2]

```
#define EI_PAD 9
```

Byte index of padding bytes.

start of padding bytes

Definition at line 299 of file [elf.h](#).

12.26.2.28 EI_PAD [2/2]

```
#define EI_PAD 9
```

Byte index of padding bytes.

start of padding bytes

Definition at line 299 of file [elf.h](#).

12.26.2.29 EI_VERSION [1/2]

```
#define EI_VERSION 6
```

File version byte index.

file version

Value must be EV_CURRENT

Definition at line 296 of file [elf.h](#).

12.26.2.30 EI_VERSION [2/2]

```
#define EI_VERSION 6
```

File version byte index.

file version

Value must be EV_CURRENT

Definition at line 296 of file [elf.h](#).

12.26.2.31 ELFCLASSNONE [1/2]

```
#define ELFCLASSNONE 0
```

Invalid ELF class.

Invalid class.

Definition at line 310 of file [elf.h](#).

12.26.2.32 ELFCLASSNONE [2/2]

```
#define ELFCLASSNONE 0
```

Invalid ELF class.

Invalid class.

Definition at line 310 of file [elf.h](#).

12.26.2.33 ELFDATA2LSB [1/2]

```
#define ELFDATA2LSB 1
```

2's complement, little endian

0x01020304 => [0x04|0x03|0x02|0x01]

Definition at line 317 of file [elf.h](#).

12.26.2.34 ELFDATA2LSB [2/2]

```
#define ELFDATA2LSB 1
```

2's complement, little endian

0x01020304 => [0x04|0x03|0x02|0x01]

Definition at line 317 of file [elf.h](#).

12.26.2.35 ELFDATA2MSB [1/2]

```
#define ELFDATA2MSB 2
```

2's complement, big endian

0x01020304 => [0x01|0x02|0x03|0x04]

Definition at line 318 of file [elf.h](#).

12.26.2.36 ELFDATA2MSB [2/2]

```
#define ELFDATA2MSB 2
```

2's complement, big endian

0x01020304 => [0x01|0x02|0x03|0x04]

Definition at line 318 of file [elf.h](#).

12.26.2.37 ELFDATANONE [1/2]

```
#define ELFDATANONE 0
```

Invalid data encoding.

invalid data encoding

Definition at line 316 of file [elf.h](#).

12.26.2.38 ELFDATANONE [2/2]

```
#define ELFDATANONE 0
```

Invalid data encoding.

invalid data encoding

Definition at line 316 of file [elf.h](#).

12.26.2.39 ELFOSABI_AIX

```
#define ELFOSABI_AIX 7
```

IBM AIX.

Definition at line 181 of file [elf.h](#).

12.26.2.40 ELFOSABI_FREEBSD

```
#define ELFOSABI_FREEBSD 9
```

FreeBSD.

Definition at line 183 of file [elf.h](#).

12.26.2.41 ELFOSABI_HPUX [1/2]

```
#define ELFOSABI_HPUX 1
```

HP-UX.

HP-UX operating system.

Definition at line 323 of file [elf.h](#).

12.26.2.42 ELFOSABI_HPUX [2/2]

```
#define ELFOSABI_HPUX 1
```

HP-UX.

HP-UX operating system.

Definition at line 323 of file [elf.h](#).

12.26.2.43 ELFOSABI_IRIX

```
#define ELFOSABI_IRIX 8
```

SGI Irix.

Definition at line 182 of file [elf.h](#).

12.26.2.44 ELFOSABI_LINUX

```
#define ELFOSABI_LINUX 3
```

Linux.

Definition at line 179 of file [elf.h](#).

12.26.2.45 ELFOSABI_MODESTO

```
#define ELFOSABI_MODESTO 11
```

Novell Modesto.

Definition at line 185 of file [elf.h](#).

12.26.2.46 ELFOSABI_NETBSD

```
#define ELFOSABI_NETBSD 2
```

NetBSD.

Definition at line 178 of file [elf.h](#).

12.26.2.47 ELFOSABI_OPENBSD

```
#define ELFOSABI_OPENBSD 12
```

OpenBSD.

Definition at line 186 of file [elf.h](#).

12.26.2.48 ELFOSABI_SOLARIS

```
#define ELFOSABI_SOLARIS 6
```

Sun Solaris.

Definition at line 180 of file [elf.h](#).

12.26.2.49 ELFOSABI_SYSV [1/2]

```
#define ELFOSABI_SYSV 0
```

Alias.

UNIX System V ABI (this specification)

Definition at line 322 of file [elf.h](#).

12.26.2.50 ELFOSABI_SYSV [2/2]

```
#define ELFOSABI_SYSV 0
```

Alias.

UNIX System V ABI (this specification)

Definition at line 322 of file [elf.h](#).

12.26.2.51 ELFOSABI_TRU64

```
#define ELFOSABI_TRU64 10
```

Compaq TRU64 UNIX.

Definition at line 184 of file [elf.h](#).

12.26.2.52 EM_ARC

```
#define EM_ARC 45
```

Argonaut RISC Core, Argonaut Techn Inc.

Definition at line 230 of file [elf.h](#).

12.26.2.53 NT_VERSION

```
#define NT_VERSION 1
```

Contains a version string.

Definition at line 495 of file [elf.h](#).

12.26.2.54 PT_GNU_EH_FRAME

```
#define PT_GNU_EH_FRAME (PT_LOOS + 0x474e550)
```

EH frame information.

Definition at line 458 of file [elf.h](#).

12.26.2.55 PT_GNU_RELRO

```
#define PT_GNU_RELRO (PT_LOOS + 0x474e552)
```

Read only after reloc.

Definition at line 460 of file [elf.h](#).

12.26.2.56 PT_GNU_STACK

```
#define PT_GNU_STACK (PT_LOOS + 0x474e551)
```

Flags for stack.

Definition at line 459 of file [elf.h](#).

12.26.2.57 PT_HIOS

```
#define PT_HIOS 0x6fffffff
```

os spec.

Definition at line 454 of file [elf.h](#).

12.26.2.58 PT_HIPROC

```
#define PT_HIPROC 0x7fffffff
```

processor spec.

Definition at line 456 of file [elf.h](#).

12.26.2.59 PT_L4_AUX

```
#define PT_L4_AUX (PT_LOOS + 0x14)
```

Address of the AUX structures.

Definition at line 464 of file [elf.h](#).

12.26.2.60 PT_L4_KIP

```
#define PT_L4_KIP (PT_LOOS + 0x13)
```

Address of the KIP.

Definition at line 463 of file [elf.h](#).

12.26.2.61 PT_L4_STACK

```
#define PT_L4_STACK (PT_LOOS + 0x12)
```

Address of the stack.

Definition at line 462 of file [elf.h](#).

12.26.2.62 PT_LOOS

```
#define PT_LOOS 0x60000000
```

os spec.

Definition at line 453 of file [elf.h](#).

12.26.2.63 PT_LOPROC

```
#define PT_LOPROC 0x70000000
```

processor spec.

Definition at line 455 of file [elf.h](#).

12.26.2.64 SHF_GROUP

```
#define SHF_GROUP 0x200
```

Section is member of a group.

Definition at line 408 of file [elf.h](#).

12.26.2.65 SHF_MASKOS

```
#define SHF_MASKOS 0x0ff00000
```

OS-specific.

Definition at line 410 of file [elf.h](#).

12.26.2.66 SHF_TLS

```
#define SHF_TLS 0x400
```

Section hold thread-local data.

Definition at line 409 of file [elf.h](#).

12.26.2.67 SHT_NUM

```
#define SHT_NUM 19
```

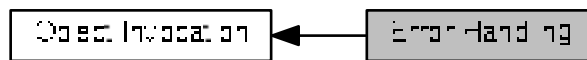
Number of defined types.

Definition at line 388 of file [elf.h](#).

12.27 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 }
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 error code for an object invocation.
- `long l4_error (l4_msgtag_t tag) L4_NOTHROW`
Return error code of a system call return message tag.
- `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.

12.27.1 Detailed Description

Error handling for L4 object invocation.

Include File

```
#include <l4/sys/ipc.h>
```

12.27.2 Enumeration Type Documentation

12.27.2.1 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	Cut receive message, due to message buffer is too small.
L4_IPC_SEMSGCUT	Cut send message. due to message buffer is too small,

Definition at line 75 of file [ipc.h](#).

12.27.3 Function Documentation

12.27.3.1 l4_error()

```
long l4_error (
    l4\_msgtag\_t tag ) [inline]
```

Return error code of a system call return message tag.

Parameters

<i>tag</i>	System call return message type
------------	---------------------------------

Returns

0 for no error, error number in case of error

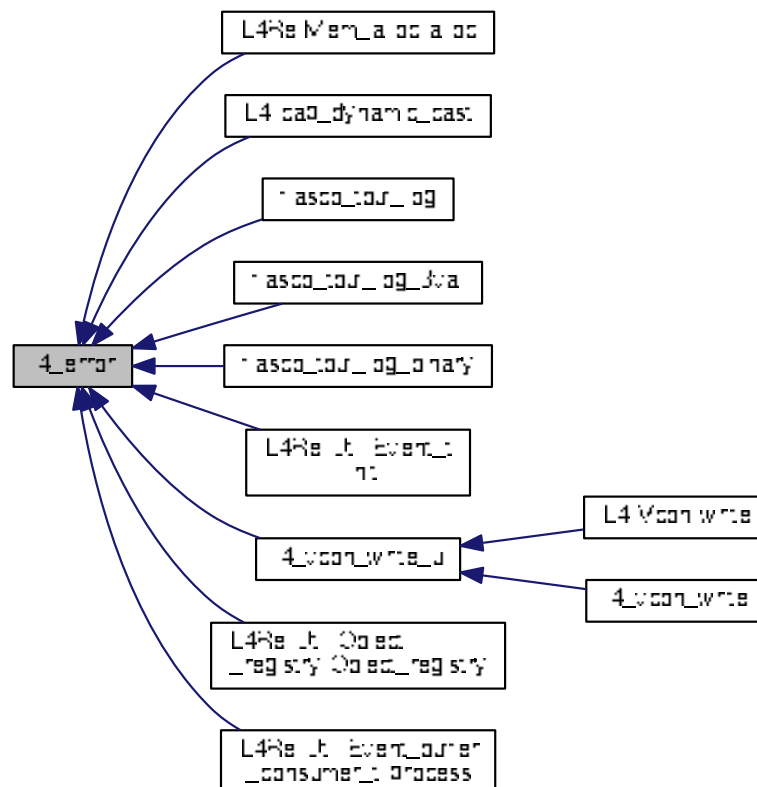
Examples:

[examples/libs/l4re/streammap/client.cc](#), [examples/sys/aliens/main.c](#), [examples/sys/isr/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 517 of file [ipc.h](#).

Referenced by [L4Re::Mem_alloc::alloc\(\)](#), [L4::cap_dynamic_cast\(\)](#), [fiasco_tbuf_log\(\)](#), [fiasco_tbuf_log_3val\(\)](#), [fiasco_tbuf_log_binary\(\)](#), [L4Re::Util::Event_t< PAYLOAD >::init\(\)](#), [l4_vcon_write_u\(\)](#), [L4Re::Util::Object_registry↵
::Object_registry\(\)](#), and [L4Re::Util::Event_buffer_consumer_t< PAYLOAD >::process\(\)](#).

Here is the caller graph for this function:



12.27.3.2 l4_ipc_error()

```
l4_umword_t l4_ipc_error (
    l4_msgtag_t tag,
    l4_utcb_t * utcb ) [inline]
```

Get the error code for an object invocation.

Parameters

<i>tag</i>	Return value of the invocation.
<i>utcb</i>	UTCB that was used for the invocation.

Returns

0 if no error condition is set, error code otherwise (see [l4_ipc_tcr_error_t](#)).

Examples:

[examples/sys/ipc/ipc_example.c](#), [examples/sys/isr/main.c](#), and [examples/sys/start-with-exc/main.c](#).

Definition at line 500 of file [ipc.h](#).

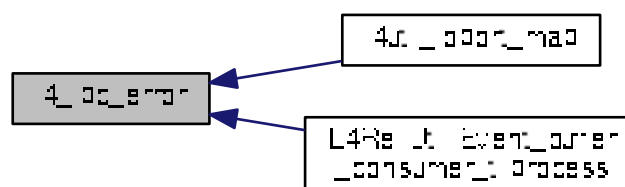
References [l4_msgtag_has_error\(\)](#).

Referenced by [l4util_ioport_map\(\)](#), and [L4Re::Util::Event_buffer_consumer_t< PAYLOAD >::process\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



12.27.3.3 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 529 of file [ipc.h](#).

12.27.3.4 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 526 of file [ipc.h](#).

12.27.3.5 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

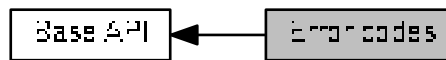
Boolean value.

Definition at line [523](#) of file [ipc.h](#).

12.28 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_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_EIPC_LO` = 2000, `L4_EIPC_HI` = 2000 + 0x1f }
L4 error codes.

12.28.1 Detailed Description

Common error codes.

Include File

```
#include <l4/sys/err.h>
```

12.28.2 Enumeration Type Documentation

12.28.2.1 `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_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_MSGTOOSHORT	Message too short.
L4_MSGTOOLONG	Message too long.
L4_MSGMISSARG	Message has invalid capability.
L4_EIPC_LO	Communication error-range low.
L4_EIPC_HI	Communication error-range high.

Definition at line 41 of file [err.h](#).

12.29 Event API

[Event](#) interface.

Collaboration diagram for Event API:



Data Structures

- class [L4Re::Event](#)
Event class.
- class [L4Re::Event_buffer_t< PAYLOAD >](#)
Event buffer class.

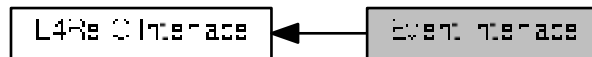
12.29.1 Detailed Description

[Event](#) interface.

12.30 Event interface

Event C interface.

Collaboration diagram for Event interface:



Functions

- long [l4re_event_get_buffer](#) (const [l4_cap_idx_t](#) server, const [l4re_ds_t](#) ds) [L4_NOTHROW](#)
Get an event signal buffer.
- long [l4re_event_get_num_streams](#) (const [l4_cap_idx_t](#) server) [L4_NOTHROW](#)
Get number of streams.
- long [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.
- long [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.
- long [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.

12.30.1 Detailed Description

Event C interface.

12.30.2 Function Documentation

12.30.2.1 l4re_event_get_axis_info()

```

long 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 <code>axis</code> 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](#)

12.30.2.2 l4re_event_get_buffer()

```
long 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](#)

12.30.2.3 l4re_event_get_num_streams()

```
long 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

12.30.2.4 l4re_event_get_stream_info()

```
long 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.

Return values

<i>info</i>	Information buffer.
-------------	---------------------

Returns

0 for success, <0 on error

See also

L4Re::Event::get_stream_info

12.30.2.5 l4re_event_get_stream_info_for_id()

```
long 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.

Return values

<i>info</i>	Information buffer.
-------------	---------------------

Returns

0 for success, <0 on error

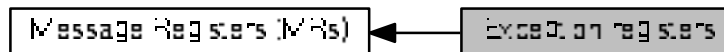
See also

L4Re::Event::get_stream_info_for_id

12.31 Exception registers

Overly definition of the MRs for exception messages.

Collaboration diagram for Exception registers:



Functions

- [l4_exc_regs_t * l4_utcb_exc](#) (void) [L4_NOTHROW](#) [L4_PURE](#)
Get the message-register block of a UTCB (for an exception IPC).
- [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\(\)](#).

12.31.1 Detailed Description

Overly definition of the MRs for exception messages.

12.31.2 Function Documentation

12.31.2.1 [l4_utcb_exc\(\)](#)

```
l4_exc_regs_t * l4_utcb_exc (
    void ) [inline]
```

Get the message-register block of a UTCB (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 [361](#) of file [utcb.h](#).

12.31.2.2 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

0	Exception was not triggered through ex_regs.
!=0	Exception was triggered through ex_regs.

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 115 of file [utcb.h](#).

References [l4_utcb_exc_typeval\(\)](#).

Here is the call graph for this function:



12.31.2.3 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 105 of file [utcb.h](#).

References [l4_exc_regs_t::trapno](#).

12.31.2.4 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 90 of file [utcb.h](#).

12.31.2.5 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.

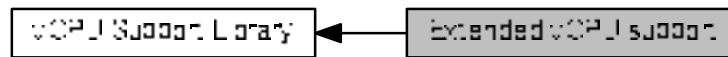
Definition at line 95 of file [utcb.h](#).

References [l4_exc_regs_t::pc](#).

12.32 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.

12.32.1 Detailed Description

extended vCPU handling functionality.

12.32.2 Function Documentation

12.32.2.1 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.

Return values

<code>vcpu</code>	Allocated vcpu-state area.
<code>ext_state</code>	Allocated extended vcpu-state area.

Parameters

<code>task</code>	Task to use for allocation.
-------------------	-----------------------------

Parameters

<i>regmgr</i>	Region manager to use for allocation.
---------------	---------------------------------------

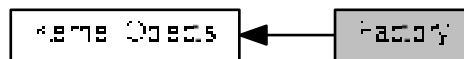
Returns

0 for success, error code otherwise

12.33 Factory

C factory interface to create kernel objects.

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 const utcb_area) L4_NOTHROW`
Create a new task.
- `l4_msgtag_t l4_factory_create_task_u (l4_cap_idx_t factory, l4_cap_idx_t target_cap, l4_fpage_t const utcb_area, l4_utcb_t *utcb) 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_thread_u (l4_cap_idx_t factory, l4_cap_idx_t target_cap, l4_utcb_t *utcb) 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_factory_u (l4_cap_idx_t factory, l4_cap_idx_t target_cap, unsigned long limit, l4_utcb_t *utcb) 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 thread_cap, l4_umword_t label) L4_NOTHROW`
Create a new IPC gate.
- `l4_msgtag_t l4_factory_create_gate_u (l4_cap_idx_t factory, l4_cap_idx_t target_cap, l4_cap_idx_t thread_cap, l4_umword_t label, l4_utcb_t *utcb) L4_NOTHROW`
Create a new IPC gate.
- `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_irq_u (l4_cap_idx_t factory, l4_cap_idx_t target_cap, l4_utcb_t *utcb) L4_NOTHROW`
Create a new IRQ.
- `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_vm_u (l4_cap_idx_t factory, l4_cap_idx_t target_cap, l4_utcb_t *utcb) L4_NOTHROW`
Create a new virtual machine.

12.33.1 Detailed Description

C factory interface to create kernel objects.

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](#).

12.33.2 Function Documentation

12.33.2.1 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

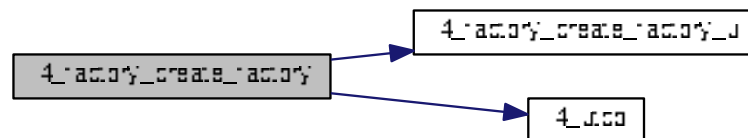
Note

The limit of the new factory is subtracted from the available amount of the factory used for creation.

Definition at line 373 of file [factory.h](#).

References [l4_factory_create_factory_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:

**12.33.2.2 l4_factory_create_factory_u()**

```

l4_msgtag_t l4_factory_create_factory_u (
    l4_cap_idx_t factory,
    l4_cap_idx_t target_cap,
    unsigned long limit,
    l4_utcb_t * utcb ) [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.
	<i>utcb</i>	The UTCB to use for the operation.

Returns

Syscall return tag

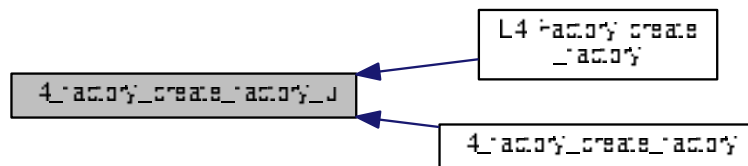
Note

The limit of the new factory is subtracted from the available amount of the factory used for creation.

Definition at line 307 of file [factory.h](#).

Referenced by [L4::Factory::create_factory\(\)](#), and [l4_factory_create_factory\(\)](#).

Here is the caller graph for this function:

**12.33.2.3 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 thread_cap,
    l4_umword_t label ) [inline]
  
```

Create a new IPC gate.

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>thread_cap</i>	Optional capability selector of the thread to bind the gate to. Use L4_INVALID_CAP to create an unbound IPC gate.
	<i>label</i>	Optional label of the gate (is used if <i>thread_cap</i> is valid).

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 <code>lpc_gate</code> object.
<i>-L4_ENOENT</i>	<code>thread_cap</code> is void or points to something that is not a thread.
<i>-L4_EPERM</i>	No write rights on <code>thread_cap</code> .

An unbound IPC gate can be bound to a thread using [l4_ipc_gate_bind_thread](#).

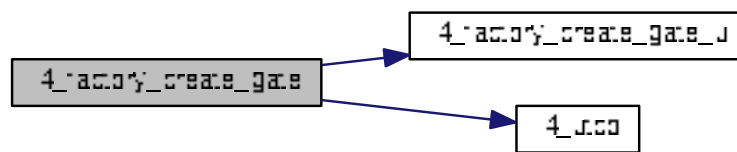
See also

[IPC-Gate API](#)

Definition at line 381 of file [factory.h](#).

References [l4_factory_create_gate_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



12.33.2.4 l4_factory_create_gate_u()

```

l4_msgtag_t l4_factory_create_gate_u (
    l4_cap_idx_t factory,
    l4_cap_idx_t target_cap,
    l4_cap_idx_t thread_cap,
    l4_umword_t label,
    l4_utcb_t * utcb ) [inline]
  
```

Create a new IPC gate.

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>thread_cap</i>	Optional capability selector of the thread to bind the gate to. Use L4_INVALID_CAP to create an unbound IPC gate.
	<i>label</i>	Optional label of the gate (is used if <i>thread_cap</i> is valid).
	<i>utcb</i>	The UTCB to use for the operation.

Returns

Syscall return tag containing one of the following return codes.

Return values

<code>L4_EOK</code>	No error occurred.
<code>-L4_ENOMEM</code>	Out-of-memory during allocation of the <code>lpc_gate</code> object.
<code>-L4_ENOENT</code>	<code>thread_cap</code> is void or points to something that is not a thread.
<code>-L4_EPERM</code>	No write rights on <code>thread_cap</code> .

An unbound IPC gate can be bound to a thread using `L4::lpc_gate::bind_thread()`.

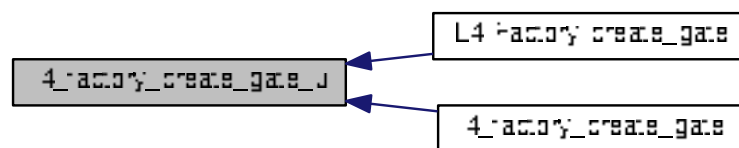
See also

[L4::lpc_gate](#)

Definition at line 318 of file [factory.h](#).

Referenced by [L4::Factory::create_gate\(\)](#), and [l4_factory_create_gate\(\)](#).

Here is the caller graph for this function:



12.33.2.5 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.

Returns

Syscall return tag

See also

[IRQs](#)

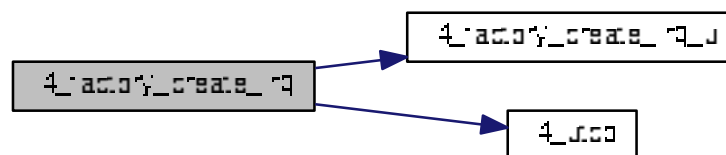
Examples:

[examples/sys/isr/main.c](#).

Definition at line 389 of file [factory.h](#).

References [l4_factory_create_irq_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



12.33.2.6 l4_factory_create_irq_u()

```
l4_msgtag_t l4_factory_create_irq_u (
    l4_cap_idx_t factory,
    l4_cap_idx_t target_cap,
    l4_utcb_t * utcb ) [inline]
```

Create a new IRQ.

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.
	<i>utcb</i>	The UTCB to use for the operation.

Returns

Syscall return tag

Deprecated Use `create()` with `Cap<Irq>` as argument instead.

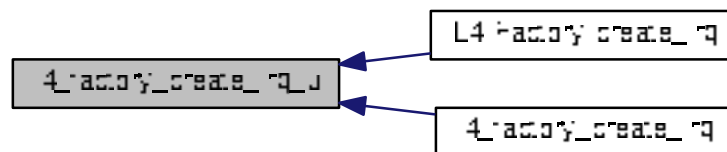
See also

[L4::Irq](#)

Definition at line 340 of file [factory.h](#).

Referenced by [L4::Factory::create_irq\(\)](#), and [l4_factory_create_irq\(\)](#).

Here is the caller graph for this function:



12.33.2.7 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 const 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.
	<i>utcb_area</i>	Flexpage that describes an area of kernel-user memory that can be used for UTCBs and vCPU state-save-areas of the new task.

Returns

Syscall return tag.

Note

The size of the UTCB area specifies indirectly the number of UTCBs available for this task. Refer to [L4::Task::add_ku_mem](#) / [l4_task_add_ku_mem\(\)](#) for adding more of this type of memory.

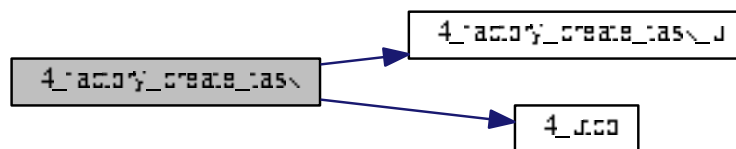
See also

[Task](#)

Definition at line 359 of file [factory.h](#).

References [l4_factory_create_task_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



12.33.2.8 l4_factory_create_task_u()

```

l4_msgtag_t l4_factory_create_task_u (
    l4_cap_idx_t factory,
    l4_cap_idx_t target_cap,
    l4_fpage_t const utcb_area,
    l4_utcb_t * utcb ) [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.
	<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.
	<i>utcb</i>	The UTCB to use for the operation.

Returns

Syscall return tag

Note

The size of the UTCB area specifies indirectly the number of UTCBs available for this task. Refer to [L4::Task::add_ku_mem](#) / [l4_task_add_ku_mem\(\)](#) for adding more of this type of memory.

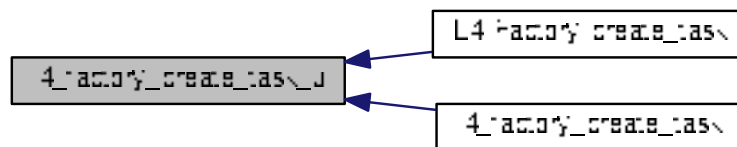
See also

[L4::Task](#)

Definition at line 289 of file [factory.h](#).

Referenced by [L4::Factory::create_task\(\)](#), and [l4_factory_create_task\(\)](#).

Here is the caller graph for this function:



12.33.2.9 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

See also

[Thread](#)

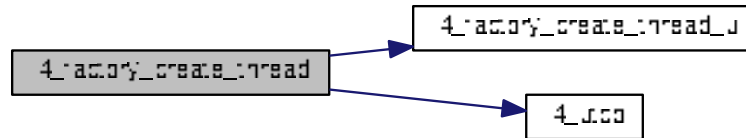
Examples:

[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 366 of file [factory.h](#).

References [l4_factory_create_thread_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



12.33.2.10 l4_factory_create_thread_u()

```

l4_msgtag_t l4_factory_create_thread_u (
    l4_cap_idx_t factory,
    l4_cap_idx_t target_cap,
    l4_utcb_t * utcb ) [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.
	<i>utcb</i>	The UTCB to use for the operation.

Returns

Syscall return tag

Deprecated Use `create()` with `Cap<Thread>` as argument instead.

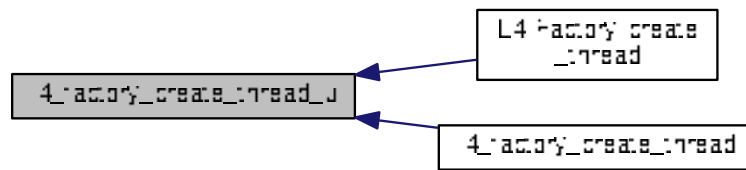
See also

[L4::Thread](#)

Definition at line 300 of file [factory.h](#).

Referenced by [L4::Factory::create_thread\(\)](#), and [l4_factory_create_thread\(\)](#).

Here is the caller graph for this function:



12.33.2.11 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

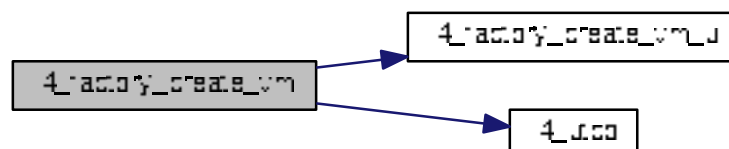
See also

[Virtual Machines](#)

Definition at line 396 of file [factory.h](#).

References [l4_factory_create_vm_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



12.33.2.12 l4_factory_create_vm_u()

```
l4_msgtag_t l4_factory_create_vm_u (
    l4_cap_idx_t factory,
    l4_cap_idx_t target_cap,
    l4_utcb_t * utcb ) [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.
	<i>utcb</i>	The UTCB to use for the operation.

Returns

Syscall return tag

Deprecated Use `create()` with `Cap<Vm>` as argument instead.

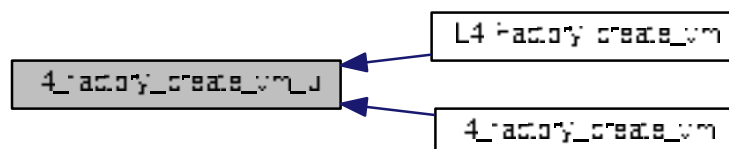
See also

[L4::Vm](#)

Definition at line 347 of file [factory.h](#).

Referenced by [L4::Factory::create_vm\(\)](#), and [l4_factory_create_vm\(\)](#).

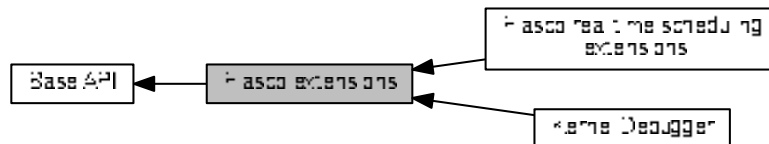
Here is the caller graph for this function:



12.34 Fiasco extensions

Kernel debugger extensions of the Fiasco [L4](#) implementation.

Collaboration diagram for Fiasco extensions:



Modules

- [Fiasco real time scheduling extensions](#)
Real time scheduling extension for the Fiasco [L4](#) implementation.
- [Kernel Debugger](#)
Kernel debugger related functionality.

Files

- file [segment.h](#)
I4f specific fs/gs manipulation
- file [segment.h](#)
I4f specific segment manipulation

Data Structures

- struct [l4_tracebuffer_status_window_t](#)
Trace-buffer status window descriptor.
- struct [l4_tracebuffer_status_t](#)
Trace-buffer status.

Enumerations

- enum {
[LOG_EVENT_CONTEXT_SWITCH](#) = 0, [LOG_EVENT_IPC_SHORTCUT](#) = 1, [LOG_EVENT_IRQ_RAISED](#) = 2, [LOG_EVENT_TIMER_IRQ](#) = 3,
[LOG_EVENT_THREAD_EX_REGS](#) = 4, [LOG_EVENT_MAX_EVENTS](#) = 16 }
Log event types.

Functions

- [l4_tracebuffer_status_t](#) * [fiasco_tbuf_get_status](#) (void)
Return trace-buffer status.
- [l4_addr_t](#) [fiasco_tbuf_get_status_phys](#) (void)
Return the physical address of the trace-buffer status struct.
- [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.
- 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.

12.34.1 Detailed Description

Kernel debugger extensions of the Fiasco [L4](#) implementation.

12.34.2 Enumeration Type Documentation

12.34.2.1 anonymous enum

anonymous enum

Log event types.

Enumerator

LOG_EVENT_CONTEXT_SWITCH	Event: context switch.
LOG_EVENT_IPC_SHORTCUT	Event: IPC shortcut.
LOG_EVENT_IRQ_RAISED	Event: IRQ occurred.
LOG_EVENT_TIMER_IRQ	Event: Timer IRQ occurred.
LOG_EVENT_THREAD_EX_REGS	Event: thread_ex_regs.
LOG_EVENT_MAX_EVENTS	Maximum number of events.

Definition at line 37 of file [ktrace.h](#).

12.34.3 Function Documentation

12.34.3.1 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 of the caller.

Definition at line 149 of file [segment.h](#).

12.34.3.2 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 3 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: 0-2).
<i>utcb</i>	UTCB of the caller.

Returns

System call error

Definition at line 52 of file [segment.h](#).

12.34.3.3 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 of the caller.

Definition at line 136 of file [segment.h](#).

References [L4_EINVAL](#), and [L4_TASK_LDT_X86_MAX_ENTRIES](#).

12.34.3.4 fiasco_tbuf_get_status()

```
l4_tracebuffer_status_t* fiasco_tbuf_get_status (
    void ) [inline]
```

Return trace-buffer status.

Returns

Pointer to trace-buffer status struct.

Definition at line 35 of file [__ktrace-impl.h](#).

12.34.3.5 fiasco_tbuf_get_status_phys()

```
l4_addr_t fiasco_tbuf_get_status_phys (
    void ) [inline]
```

Return the physical address of the trace-buffer status struct.

Returns

physical address of status struct.

Definition at line 42 of file [__ktrace-impl.h](#).

12.34.3.6 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
-------------	--------------

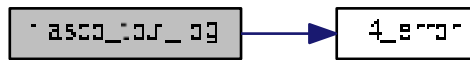
Returns

Pointer to trace-buffer entry

Definition at line 48 of file [__ktrace-impl.h](#).

References [l4_error\(\)](#).

Here is the call graph for this function:



12.34.3.7 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

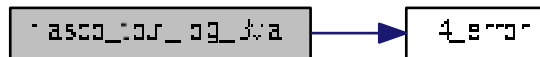
Returns

Pointer to trace-buffer entry

Definition at line 55 of file [__ktrace-impl.h](#).

References [l4_error\(\)](#).

Here is the call graph for this function:

**12.34.3.8 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 78 of file [__ktrace-impl.h](#).

References [l4_error\(\)](#).

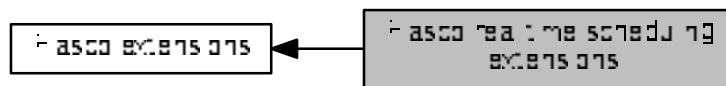
Here is the call graph for this function:



12.35 Fiasco real time scheduling extensions

Real time scheduling extension for the Fiasco [L4](#) implementation.

Collaboration diagram for Fiasco real time scheduling extensions:



Real time scheduling extension for the Fiasco [L4](#) implementation.

Enumerator

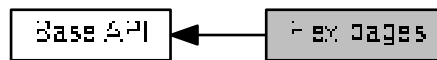
L4_TYPE_VHW_NONE	None entry.
L4_TYPE_VHW_FRAMEBUFFER	Framebuffer device.
L4_TYPE_VHW_INPUT	Input device.
L4_TYPE_VHW_NET	Network device.

Definition at line 44 of file [vhw.h](#).

12.37 Flex pages

Flex-page related API.

Collaboration diagram for Flex pages:



Data Structures

- union [l4_fpage_t](#)
L4 flexpage type.
- struct [l4_snd_fpage_t](#)
Send-flex-page types.

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 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](#) }
Flex-page 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_RW](#) | [L4_CAP_FPAGE_S](#),
[L4_CAP_FPAGE_RWSD](#) = [L4_CAP_FPAGE_RWS](#) | [L4_CAP_FPAGE_D](#), [L4_CAP_FPAGE_RWD](#) = [L4_CAP_FPAGE_RW](#) | [L4_CAP_FPAGE_D](#), [L4_CAP_FPAGE_RSD](#) = [L4_CAP_FPAGE_RS](#) | [L4_CAP_FPAGE_D](#) }
Cap-flex-page rights.
- enum [L4_fpage_type](#)
Flex-page type.
- enum [L4_fpage_control](#)

Flex-page map control flags.

- enum `L4_obj_fpage_ctl` {
`L4_FPAGE_C_REF_CNT` = 0x00, `L4_FPAGE_C_NO_REF_CNT` = 0x10, `L4_FPAGE_C_OBJ_RIGHT1` = 0x20, `L4_FPAGE_C_OBJ_RIGHT2` = 0x40,
`L4_FPAGE_C_OBJ_RIGHT3` = 0x80, `L4_FPAGE_C_OBJ_RIGHTS` = 0xe0, `L4_FPAGE_C_IPCGATE_SVR` = `L4_FPAGE_C_OBJ_RIGHT1` }

Flex-page map control for capabilities (snd_base)

- enum `l4_fpage_cacheability_opt_t` { `L4_FPAGE_CACHE_OPT` = 0x1, `L4_FPAGE_CACHEABLE` = 0x3, `L4_FPAGE_BUFFERABLE` = 0x5, `L4_FPAGE_UNCACHEABLE` = 0x1 }

Flex-page cacheability option.

- enum { `L4_WHOLE_IOADDRESS_SPACE` = 16, `L4_IOPORT_MAX` = (1L << `L4_WHOLE_IOADDRESS_SPACE`) }

Special constants for IO flex pages.

Functions

- `l4_fpage_t l4_fpage (l4_addr_t address, unsigned int size, unsigned char rights)` `L4_NOTHROW`
Create a memory flex page.
- `l4_fpage_t l4_fpage_all (void)` `L4_NOTHROW`
Get a flex page, describing all address spaces at once.
- `l4_fpage_t l4_fpage_invalid (void)` `L4_NOTHROW`
Get an invalid flex page.
- `l4_fpage_t l4_iofpage (unsigned long port, unsigned int size)` `L4_NOTHROW`
Create an IO-port flex page.
- `l4_fpage_t l4_obj_fpage (l4_cap_idx_t obj, unsigned int order, unsigned char rights)` `L4_NOTHROW`
Create a kernel-object flex page.
- `int l4_is_fpage_writable (l4_fpage_t fp)` `L4_NOTHROW`
Test if the flex page is writable.
- `unsigned l4_fpage_rights (l4_fpage_t f)` `L4_NOTHROW`
Return rights from a flex page.
- `unsigned l4_fpage_type (l4_fpage_t f)` `L4_NOTHROW`
Return type from a flex page.
- `unsigned l4_fpage_size (l4_fpage_t f)` `L4_NOTHROW`
Return size from a flex page.
- `unsigned long l4_fpage_page (l4_fpage_t f)` `L4_NOTHROW`
Return the page part from a flex page.
- `l4_addr_t l4_fpage_memaddr (l4_fpage_t f)` `L4_NOTHROW`
Return the memory address from the memory flex page.
- `l4_cap_idx_t l4_fpage_obj (l4_fpage_t f)` `L4_NOTHROW`
Return the capability index from the object flex page.
- `unsigned long l4_fpage_ioport (l4_fpage_t f)` `L4_NOTHROW`
Return the IO port number from the IO flex page.
- `l4_fpage_t l4_fpage_set_rights (l4_fpage_t src, unsigned char new_rights)` `L4_NOTHROW`
Set new right in a flex page.
- `int l4_fpage_contains (l4_fpage_t fpage, l4_addr_t addr, unsigned size)` `L4_NOTHROW`
Test whether a given range is completely within an fpage.
- `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 flex page size of a region.

12.37.1 Detailed Description

Flex-page related API.

A flex page is a page with a variable size, that can describe memory, IO-Ports (IA32 only), and sets of kernel objects.

A flex page 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 flex page also carries type and access right information for the described region. The type information selects the address space in which the flex page 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 flex page.

12.37.2 Enumeration Type Documentation

12.37.2.1 anonymous enum

anonymous enum

Constants for flexpages.

Enumerator

L4_WHOLE_ADDRESS_SPACE	Whole address space size.
------------------------	---------------------------

Definition at line 89 of file [__l4_fpage.h](#).

12.37.2.2 anonymous enum

anonymous enum

Special constants for IO flex pages.

Enumerator

L4_WHOLE_IOADDRESS_SPACE	Whole I/O address space size.
L4_IOPORT_MAX	Maximum I/O port address.

Definition at line 281 of file [__l4_fpage.h](#).

12.37.2.3 L4_cap_fpage_rights

```
enum L4_cap_fpage_rights
```

Cap-flex-page rights.

Capabilities are modified or transferred with map and unmap operations. For that capabilities are wrapped into flex-page objects. The flex-page 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

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 flex-pages. 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 flex-pages. The semantics of the 'S' right is defined by the interface. The kernel masks this right with the 'S' right of the IPC gate over which the capability is mapped. That means that the receiver capability will only have the 'S' right set if both the flex-page and the IPC gate have the 'S' bit set.
L4_CAP_FPAGE_R	Read right for capability flex-pages. This is always required, otherwise no capability is mapped.
L4_CAP_FPAGE_RO	Read right for capability flex-pages. This is always required, otherwise no capability is mapped.
L4_CAP_FPAGE_D	Delete right for capability flex-pages. This allows the receiver to delete the corresponding kernel object using unmap() 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 flex-pages. 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 flex-pages. 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 flex-pages. 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

Enumerator

L4_CAP_FPAGE_RWSD	Full rights for capability flex-pages. 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 flex-pages. 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 flex-pages. See also L4_CAP_FPAGE_R , L4_CAP_FPAGE_S , and L4_CAP_FPAGE_D

Definition at line 134 of file [__l4_fpage.h](#).

12.37.2.4 l4_fpage_cacheability_opt_t

```
enum l4_fpage_cacheability_opt_t
```

Flex-page cacheability option.

Enumerator

L4_FPAGE_CACHE_OPT	Enable the cacheability option in a send flex page.
L4_FPAGE_CACHEABLE	Cacheability option to enable caches for the mapping.
L4_FPAGE_BUFFERABLE	Cacheability option to enable buffered writes for the mapping.
L4_FPAGE_UNCACHEABLE	Cacheability option to disable caching for the mapping.

Definition at line 262 of file [__l4_fpage.h](#).

12.37.2.5 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)

Enumerator

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.

Definition at line 55 of file [__l4_fpage.h](#).

12.37.2.6 L4_fpage_rights

```
enum L4_fpage_rights
```

Flex-page rights.

Enumerator

L4_FPAGE_X	Executable flex page.
L4_FPAGE_W	Writable flex page.
L4_FPAGE_RO	Read-only flex page.
L4_FPAGE_RW	Read-write flex page.
L4_FPAGE_RX	Read-execute flex page.
L4_FPAGE_RWX	Read-write-execute flex page.

Definition at line 107 of file [__l4_fpage.h](#).

12.37.2.7 L4_obj_fpage_ctl

```
enum L4_obj_fpage_ctl
```

Flex-page map control for capabilities (snd_base)

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).

Enumerator

L4_FPAGE_C_REF_CNT	Mapping is reference-counted (default).
L4_FPAGE_C_NO_REF_CNT	Don't increase the reference counter.
L4_FPAGE_C_OBJ_RIGHT1	Object-type specific right.
L4_FPAGE_C_OBJ_RIGHT2	Object-type specific right.
L4_FPAGE_C_OBJ_RIGHT3	Object-type specific right.
L4_FPAGE_C_OBJ_RIGHTS	All Object-type specific right bits.
L4_FPAGE_C_IPCGATE_SVR Generated for L4Re by Doxygen	The receiver may invoke IPC-gate-specific functions on the capability, e.g. 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 240 of file [__l4_fpage.h](#).

12.37.3 Function Documentation

12.37.3.1 l4_fpage()

```
l4_fpage_t l4_fpage (
    l4_addr_t address,
    unsigned int size,
    unsigned char rights ) [inline]
```

Create a memory flex page.

Parameters

<i>address</i>	Flex-page start address
<i>size</i>	Flex-page size (log2), L4_WHOLE_ADDRESS_SPACE to specify the whole address space (with address 0)
<i>rights</i>	Access rights, see L4_fpage_rights

Returns

Memory flex page

Definition at line 633 of file [__l4_fpage.h](#).

Referenced by [L4Re::Rm::attach\(\)](#).

Here is the caller graph for this function:



12.37.3.2 l4_fpage_all()

```
l4_fpage_t l4_fpage_all (
    void ) [inline]
```

Get a flex page, describing all address spaces at once.

Returns

Special *all-spaces* flex page.

Definition at line 653 of file [__l4_fpage.h](#).

References [L4_WHOLE_ADDRESS_SPACE](#).

12.37.3.3 l4_fpage_contains()

```
int l4_fpage_contains (
    l4_fpage_t fpage,
    l4_addr_t addr,
    unsigned size ) [inline]
```

Test whether a given range is completely within an fpage.

Parameters

<i>fpage</i>	Flex page
<i>addr</i>	Address
<i>size</i>	Size of range in log2.

Return values

<i>==0</i>	The range is not completely in the fpage.
<i>!=0</i>	The range is within the fpage.

Definition at line 685 of file [__l4_fpage.h](#).

References [l4_fpage_memaddr\(\)](#).

Here is the call graph for this function:



12.37.3.4 l4_fpage_invalid()

```
l4_fpage_t l4_fpage_invalid (
    void ) [inline]
```

Get an invalid flex page.

Returns

Special *invalid* flex page.

Definition at line 659 of file [__l4_fpage.h](#).

12.37.3.5 l4_fpage_ioport()

```
unsigned long l4_fpage_ioport (
    l4_fpage_t f ) [inline]
```

Return the IO port number from the IO flex page.

Parameters

<i>f</i>	Flex page
----------	-----------

Returns

IO port number from the given IO flex page.

Precondition

f must be an IO flex page (`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 589 of file [__l4_fpage.h](#).

12.37.3.6 l4_fpage_max_order()

```
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 flex page size of a region.

Parameters

<i>order</i>	Order value to start with (e.g. for memory L4_LOG2_PAGESIZE would be used)
<i>addr</i>	Address to be covered by the flex page.
<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

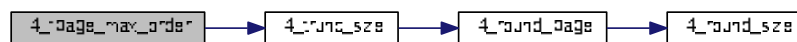
The start address of the flex-page can be determined with `l4_trunc_size(addr, returnvalue)`

Definition at line 693 of file `__l4_fpage.h`.

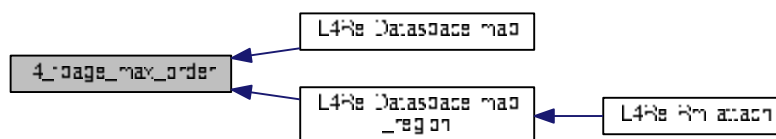
References `l4_trunc_size()`.

Referenced by `L4Re::Dataspace::map()`, and `L4Re::Dataspace::map_region()`.

Here is the call graph for this function:



Here is the caller graph for this function:



12.37.3.7 l4_fpage_memaddr()

```

l4_addr_t l4_fpage_memaddr (
    l4_fpage_t f ) [inline]
  
```

Return the memory address from the memory flex page.

Parameters

<i>f</i>	Flex page
----------	-----------

Returns

Page address from the given memory flex page.

Precondition

f must be a memory flex page (`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 595 of file [__l4_fpage.h](#).

Referenced by [l4_fpage_contains\(\)](#).

Here is the caller graph for this function:

**12.37.3.8 l4_fpage_obj()**

```
l4_cap_idx_t l4_fpage_obj (
    l4_fpage_t f ) [inline]
```

Return the capability index from the object flex page.

Parameters

<i>f</i>	Flex page
----------	-----------

Returns

Capability index from the given object flex page.

Precondition

`f` must be an object flex page (`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 601 of file [__l4_fpage.h](#).

12.37.3.9 l4_fpage_page()

```
unsigned long l4_fpage_page (
    l4_fpage_t f ) [inline]
```

Return the page part from a flex page.

Parameters

<code>f</code>	Flex page
----------------	-----------

Returns

Page part of the given flex page.

Note

The meaning of the page part depends on the flex-page type.

Definition at line 583 of file [__l4_fpage.h](#).

12.37.3.10 l4_fpage_rights()

```
unsigned l4_fpage_rights (
    l4_fpage_t f ) [inline]
```

Return rights from a flex page.

Parameters

<code>f</code>	Flex page
----------------	-----------

Returns

Size part of the given flex page.

Definition at line 565 of file [__l4_fpage.h](#).

References [L4_FPAGE_RIGHTS_MASK](#), and [L4_FPAGE_RIGHTS_SHIFT](#).

Referenced by [l4_is_fpage_writable\(\)](#).

Here is the caller graph for this function:



12.37.3.11 l4_fpage_set_rights()

```
l4_fpage_t l4_fpage_set_rights (
    l4_fpage_t src,
    unsigned char new_rights ) [inline]
```

Set new right in a flex page.

Parameters

<i>src</i>	Flex page
<i>new_rights</i>	New rights

Returns

Modified flex page with new rights.

Definition at line 624 of file [__l4_fpage.h](#).

References [l4_fpage_t::raw](#).

12.37.3.12 l4_fpage_size()

```
unsigned l4_fpage_size (
    l4_fpage_t f ) [inline]
```

Return size from a flex page.

Parameters

<i>f</i>	Flex page
----------	-----------

Returns

Size part of the given flex page.

See also

[l4_fpage_memaddr\(\)](#), [l4_fpage_obj\(\)](#), [l4_fpage_ioport\(\)](#)

Definition at line 577 of file [__l4_fpage.h](#).

12.37.3.13 l4_fpage_type()

```
unsigned l4_fpage_type (
    l4_fpage_t f ) [inline]
```

Return type from a flex page.

Parameters

<i>f</i>	Flex page
----------	-----------

Returns

Type part of the given flex page.

Definition at line 571 of file [__l4_fpage.h](#).

12.37.3.14 l4_iofpage()

```
l4_fpage_t l4_iofpage (
    unsigned long port,
    unsigned int size ) [inline]
```

Create an IO-port flex page.

Parameters

<i>port</i>	I/O-flex-page port base
<i>size</i>	I/O-flex-page size (log2), L4_WHOLE_IOADDRESS_SPACE to specify the whole I/O address space (with <code>port 0</code>)

Returns

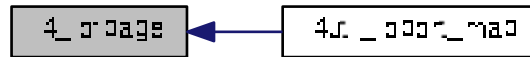
I/O flex page

Definition at line 639 of file [__l4_fpage.h](#).

References [L4_FPAGE_ADDR_SHIFT](#).

Referenced by [l4util_ioport_map\(\)](#).

Here is the caller graph for this function:



12.37.3.15 l4_is_fpage_writable()

```
int l4_is_fpage_writable (
    l4_fpage_t fp ) [inline]
```

Test if the flex page is writable.

Parameters

<i>fp</i>	Flex page.
-----------	------------

Return values

<i>!=0</i>	if flex page is writable.
<i>==0</i>	if flex page is not writable.

Definition at line 666 of file [__l4_fpage.h](#).

References [l4_fpage_rights\(\)](#), and [L4_FPAGE_W](#).

Here is the call graph for this function:



12.37.3.16 l4_obj_fpage()

```
l4_fpage_t l4_obj_fpage (
    l4_cap_idx_t obj,
    unsigned int order,
    unsigned char rights ) [inline]
```

Create a kernel-object flex page.

Parameters

<i>obj</i>	Base capability selector.
<i>order</i>	Log2 size (number of capabilities).
<i>rights</i>	Access rights

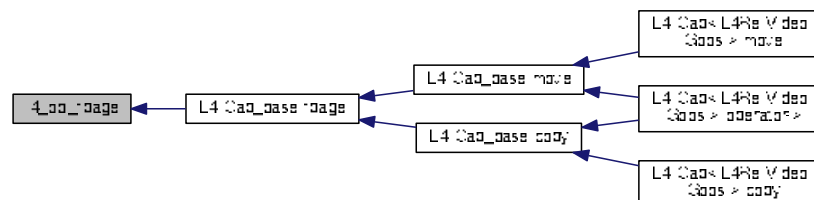
Returns

Flex page for a set of kernel objects.

Definition at line 645 of file `__l4_fpage.h`.

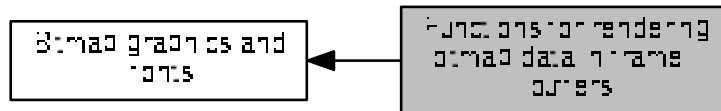
Referenced by `L4::Cap_base::fpage()`.

Here is the caller graph for this function:



12.38 Functions for rendering bitmap data in frame buffers

Collaboration diagram for Functions for rendering bitmap data in frame buffers:



Data Structures

- struct [gfxbitmap_offset](#)
offsets in pmap[] and bmap[]

Typedefs

- typedef unsigned int [gfxbitmap_color_t](#)
Standard color type.
- typedef unsigned int [gfxbitmap_color_pix_t](#)
Specific color type.

Functions

- [gfxbitmap_color_pix_t gfxbitmap_convert_color](#) ([l4re_video_view_info_t](#) *vi, [gfxbitmap_color_t](#) rgb)
Convert a color.
- void [gfxbitmap_fill](#) ([l4_uint8_t](#) *vfb, [l4re_video_view_info_t](#) *vi, int x, int y, int w, int h, [gfxbitmap_color_pix_t](#) color)
Fill a rectangular area with a color.
- void [gfxbitmap_bmap](#) ([l4_uint8_t](#) *vfb, [l4re_video_view_info_t](#) *vi, [l4_int16_t](#) x, [l4_int16_t](#) y, [l4_uint32_t](#) w, [l4_uint32_t](#) h, [l4_uint8_t](#) *bmap, [gfxbitmap_color_pix_t](#) fg, [gfxbitmap_color_pix_t](#) bg, struct [gfxbitmap_offset](#) *offset, [l4_uint8_t](#) mode)
Fill a rectangular area with a bicolor bitmap pattern.
- void [gfxbitmap_set](#) ([l4_uint8_t](#) *vfb, [l4re_video_view_info_t](#) *vi, [l4_int16_t](#) x, [l4_int16_t](#) y, [l4_uint32_t](#) w, [l4_uint32_t](#) h, [l4_uint32_t](#) xoffs, [l4_uint32_t](#) yoffs, [l4_uint8_t](#) *pmap, struct [gfxbitmap_offset](#) *offset, [l4_uint32_t](#) pwidth)
Set area from source area.
- void [gfxbitmap_copy](#) ([l4_uint8_t](#) *dest, [l4_uint8_t](#) *src, [l4re_video_view_info_t](#) *vi, int x, int y, int w, int h, int dx, int dy)
Copy a rectangular area.

12.38.1 Detailed Description

12.38.2 Typedef Documentation

12.38.2.1 `gfxbitmap_color_pix_t`

```
typedef unsigned int gfxbitmap_color_pix_t
```

Specific color type.

This color type is specific for a particular framebuffer, it can be use to write pixel on a framebuffer. Use `gfxbitmap_convert_color` to convert from `gfxbitmap_color_t` to `gfxbitmap_color_pix_t`.

Definition at line 66 of file `bitmap.h`.

12.38.2.2 `gfxbitmap_color_t`

```
typedef unsigned int gfxbitmap_color_t
```

Standard color type.

It's a RGB type with 8bits for each channel, regardless of the framebuffer used.

Definition at line 57 of file `bitmap.h`.

12.38.3 Function Documentation

12.38.3.1 `gfxbitmap_bmap()`

```
void gfxbitmap_bmap (
    l4_uint8_t * vfb,
    l4re_video_view_info_t * vi,
    l4_int16_t x,
    l4_int16_t y,
    l4_uint32_t w,
    l4_uint32_t h,
    l4_uint8_t * bmap,
    gfxbitmap_color_pix_t fg,
    gfxbitmap_color_pix_t bg,
    struct gfxbitmap_offset * offset,
    l4_uint8_t mode )
```

Fill a rectangular area with a bicolor bitmap pattern.

Parameters

<i>vfb</i>	Frame buffer.
<i>vi</i>	Frame buffer information structure.
<i>x</i>	X position of area.
<i>y</i>	Y position of area.
<i>w</i>	Width of area.
<i>h</i>	Height of area.
<i>bmap</i>	Generated for BMap pattern.
<i>fg</i>	Foreground color.
<i>bg</i>	Background color.
<i>offset</i>	Offsets.

See also

[pSLIM_BMAP_START_MSB](#) and [pSLIM_BMAP_START_LSB](#).

12.38.3.2 gfxbitmap_convert_color()

```
gfxbitmap_color_pix_t gfxbitmap_convert_color (
    l4re_video_view_info_t * vi,
    gfxbitmap_color_t rgb )
```

Convert a color.

Converts a given color in standard format to the format used in the framebuffer.

12.38.3.3 gfxbitmap_copy()

```
void gfxbitmap_copy (
    l4_uint8_t * dest,
    l4_uint8_t * src,
    l4re_video_view_info_t * vi,
    int x,
    int y,
    int w,
    int h,
    int dx,
    int dy )
```

Copy a rectangular area.

Parameters

<i>dest</i>	Destination frame buffer.
<i>src</i>	Source frame buffer.
<i>vi</i>	Frame buffer information structure.
<i>x</i>	Source X position of area.
<i>y</i>	Source Y position of area.
<i>w</i>	Width of area.
<i>h</i>	Height of area.
<i>dx</i>	Source X position of area.
<i>dy</i>	Source Y position of area.

12.38.3.4 gfxbitmap_fill()

```
void gfxbitmap_fill (
    l4_uint8_t * vfb,
```

```

    l4re_video_view_info_t * vi,
    int x,
    int y,
    int w,
    int h,
    gfxbitmap_color_pix_t color )

```

Fill a rectangular area with a color.

Parameters

<i>vfb</i>	Frame buffer.
<i>vi</i>	Frame buffer information structure.
<i>x</i>	X position of area.
<i>y</i>	Y position of area.
<i>w</i>	Width of area.
<i>h</i>	Height of area.
<i>color</i>	Color of area.

12.38.3.5 gfxbitmap_set()

```

void gfxbitmap_set (
    l4_uint8_t * vfb,
    l4re_video_view_info_t * vi,
    l4_int16_t x,
    l4_int16_t y,
    l4_uint32_t w,
    l4_uint32_t h,
    l4_uint32_t xoffs,
    l4_uint32_t yoffs,
    l4_uint8_t * pmap,
    struct gfxbitmap_offset * offset,
    l4_uint32_t pwidth )

```

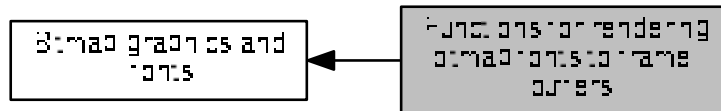
Set area from source area.

Parameters

<i>vfb</i>	Frame buffer.
<i>vi</i>	Frame buffer information structure.
<i>x</i>	X position of area.
<i>y</i>	Y position of area.
<i>w</i>	Width of area.
<i>h</i>	Height of area.
<i>pmap</i>	Source.
<i>xoffs</i>	X offset.
<i>yoffs</i>	Y offset.
<i>offset</i>	Offsets.
<i>pwidth</i>	Width of source in bytes.

12.39 Functions for rendering bitmap fonts to frame buffers

Collaboration diagram for Functions for rendering bitmap fonts to frame buffers:



Macros

- `#define GFXBITMAP_DEFAULT_FONT (void *)0`
Constant to use for the default font.

Typedefs

- `typedef void * gfxbitmap_font_t`
Font.

Enumerations

- `enum`
Constant for length field.

Functions

- `int gfxbitmap_font_init (void)`
Initialize the library.
- `gfxbitmap_font_t gfxbitmap_font_get (const char *name)`
Get a font descriptor.
- `unsigned gfxbitmap_font_width (gfxbitmap_font_t font)`
Get the font width.
- `unsigned gfxbitmap_font_height (gfxbitmap_font_t font)`
Get the font height.
- `void * gfxbitmap_font_data (gfxbitmap_font_t font, unsigned c)`
Get bitmap font data for a specific character.
- `void gfxbitmap_font_text (void *fb, l4re_video_view_info_t *vi, gfxbitmap_font_t font, const char *text, unsigned len, unsigned x, unsigned y, gfxbitmap_color_pix_t fg, gfxbitmap_color_pix_t bg)`
Render a string to a framebuffer.
- `void gfxbitmap_font_text_scale (void *fb, l4re_video_view_info_t *vi, gfxbitmap_font_t font, const char *text, unsigned len, unsigned x, unsigned y, gfxbitmap_color_pix_t fg, gfxbitmap_color_pix_t bg, int scale_x, int scale_y)`
Render a string to a framebuffer, including scaling.

12.39.1 Detailed Description

12.39.2 Enumeration Type Documentation

12.39.2.1 anonymous enum

anonymous enum

Constant for length field.

Use this if the function should call strlen on the text argument itself.

Definition at line 38 of file [font.h](#).

12.39.3 Function Documentation

12.39.3.1 gfxbitmap_font_data()

```
void* gfxbitmap_font_data (
    gfxbitmap_font_t font,
    unsigned c )
```

Get bitmap font data for a specific character.

Parameters

<i>font</i>	Font.
<i>c</i>	Character.

Returns

Pointer to bmap data, NULL on error.

12.39.3.2 gfxbitmap_font_get()

```
gfxbitmap_font_t gfxbitmap_font_get (
    const char * name )
```

Get a font descriptor.

Parameters

<i>name</i>	Name of the font.
-------------	-------------------

Returns

A (opaque) font descriptor, or NULL if font could not be found.

12.39.3.3 gfxbitmap_font_height()

```
unsigned gfxbitmap_font_height (  
    gfxbitmap_font_t font )
```

Get the font height.

Parameters

<i>font</i>	Font.
-------------	-------

Returns

Font height, 0 if font height could not be retrieved.

12.39.3.4 gfxbitmap_font_init()

```
int gfxbitmap_font_init (  
    void )
```

Initialize the library.

This function must be called before any other font function of this library.

Returns

0 on success, other on error

12.39.3.5 gfxbitmap_font_text()

```
void gfxbitmap_font_text (  
    void * fb,  
    l4re_video_view_info_t * vi,  
    gfxbitmap_font_t font,  
    const char * text,  
    unsigned len,  
    unsigned x,  
    unsigned y,  
    gfxbitmap_color_pix_t fg,  
    gfxbitmap_color_pix_t bg )
```

Render a string to a framebuffer.

Parameters

<i>fb</i>	Pointer to frame buffer.
<i>vi</i>	Frame buffer info structure.
<i>font</i>	Font.
<i>text</i>	Text string.
<i>len</i>	Length of the text string.
<i>x</i>	Horizontal position in the frame buffer.
<i>y</i>	Vertical position in the frame buffer.
<i>fg</i>	Foreground color.
<i>bg</i>	Background color.

12.39.3.6 gfxbitmap_font_text_scale()

```
void gfxbitmap_font_text_scale (
    void * fb,
    l4re_video_view_info_t * vi,
    gfxbitmap_font_t font,
    const char * text,
    unsigned len,
    unsigned x,
    unsigned y,
    gfxbitmap_color_pix_t fg,
    gfxbitmap_color_pix_t bg,
    int scale_x,
    int scale_y )
```

Render a string to a framebuffer, including scaling.

Parameters

<i>fb</i>	Pointer to frame buffer.
<i>vi</i>	Frame buffer info structure.
<i>font</i>	Font.
<i>text</i>	Text string.
<i>len</i>	Length of the text string.
<i>x</i>	Horizontal position in the frame buffer.
<i>y</i>	Vertical position in the frame buffer.
<i>fg</i>	Foreground color.
<i>bg</i>	Background color.
<i>scale_x</i>	Horizontal scale factor.
<i>scale_y</i>	Vertical scale factor.

12.39.3.7 gfxbitmap_font_width()

```
unsigned gfxbitmap_font_width (  
    gfxbitmap_font_t font )
```

Get the font width.

Parameters

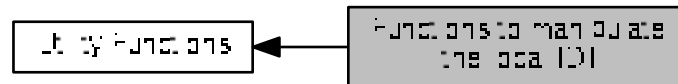
<i>font</i>	Font.
-------------	-------

Returns

Font width, 0 if font width could not be retrieved.

12.40 Functions to manipulate the local IDT

Collaboration diagram for Functions to manipulate the local IDT:



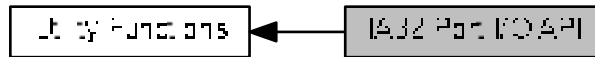
Data Structures

- struct [l4util_idt_desc_t](#)
IDT entry.
- struct [l4util_idt_header_t](#)
Header of an IDT table.

12.40.1 Detailed Description

12.41 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

12.41.1 Detailed Description

12.41.2 Function Documentation

12.41.2.1 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 180 of file [port_io.h](#).

12.41.2.2 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 188 of file [port_io.h](#).

12.41.2.3 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 172 of file [port_io.h](#).

Referenced by [l4util_irq_acknowledge\(\)](#).

Here is the caller graph for this function:



12.41.2.4 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 205 of file [port_io.h](#).

12.41.2.5 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 214 of file [port_io.h](#).

12.41.2.6 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 196 of file [port_io.h](#).

12.41.2.7 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 229 of file [port_io.h](#).

12.41.2.8 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 235 of file [port_io.h](#).

12.41.2.9 l4util_out8()

```
void l4util_out8 (  
    l4_uint8_t value,  
    l4_uint16_t port ) [inline]
```

Write byte to I/O port.

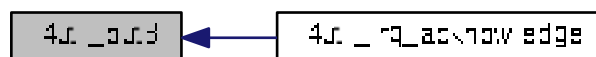
Parameters

<i>port</i>	I/O port address
<i>value</i>	value to write

Definition at line 223 of file [port_io.h](#).

Referenced by [l4util_irq_acknowledge\(\)](#).

Here is the caller graph for this function:



12.41.2.10 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 250 of file [port_io.h](#).

12.41.2.11 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 259 of file [port_io.h](#).

12.41.2.12 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 241 of file [port_io.h](#).

12.42 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_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_search_iomem_region` (`l4_addr_t` phys, `l4_addr_t` size, `l4_addr_t` *rstart, `l4_addr_t` *rsize)
Search for a 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.

12.42.1 Detailed Description

12.42.2 Typedef Documentation

12.42.2.1 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 69 of file [types.h](#).

12.42.3 Enumeration Type Documentation

12.42.3.1 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 38 of file [types.h](#).

12.42.3.2 l4io_iomem_flags_t

```
enum l4io_iomem_flags_t
```

Flags for IO memory.

Enumerator

L4IO_MEM_NONCACHED	Non-cache memory.
--------------------	-------------------

Enumerator

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 16 of file [types.h](#).

12.42.3.3 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 50 of file [types.h](#).

12.42.4 Function Documentation

12.42.4.1 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.

12.42.4.2 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
----------------	----------------

Return values

<i>dev_handle</i>	Device handle for found device, can be NULL.
<i>dev</i>	Device information, filled by the function, can be NULL.
<i>res_handle</i>	Resource handle, can be NULL.

Returns

0 on success, error code otherwise

12.42.4.3 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)
<i>reshandle</i>	Resource handle, start with handle returned by device functions.

Return values

<i>reshandle</i>	Next resource handle.
<i>res</i>	Device descriptor

Returns

0 on success, error code otherwise, esp. -L4_ENOENT if no more resources found

12.42.4.4 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

12.42.4.5 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

12.42.4.6 `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

<i>0</i>	Success.
<i>-L4_ENOENT</i>	No area in the caller's address space could be found to map the IO memory region.
<i>-L4_EPERM</i>	Operation not allowed.
<i>-L4_EINVAL</i>	Invalid value.
<i>-L4_EADDRNOTAVAIL</i>	The requested virtual address is not available.
<i>-L4_ENOMEM</i>	The requested IO memory region could not be allocated.
<i><0</i>	IPC errors.

Note

This function uses [L4Re](#) functionality to reserve a part of the virtual address space of the caller.

12.42.4.7 `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

<i>0</i>	Success.
<i>-L4_ENOENT</i>	No area could be found to map the IO memory region.
<i>-L4_EPERM</i>	Operation not allowed.
<i>-L4_EINVAL</i>	Invalid value.
<i>-L4_EADDRNOTAVAIL</i>	The requested virtual address is not available.
<i>-L4_ENOMEM</i>	The requested IO memory region could not be allocated.
<i><0</i>	IPC errors.

Note

This function uses [L4Re](#) functionality to reserve a part of the virtual address space of the caller.

12.42.4.8 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

12.42.4.9 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

<i>0</i>	An error occurred. The value of 'reshandle' is undefined.
<i>>0</i>	The virtual address of the IO memory mapping.

12.42.4.10 `l4io_search_iomem_region()`

```
long l4io_search_iomem_region (
    l4_addr_t phys,
    l4_addr_t size,
    l4_addr_t * rstart,
    l4_addr_t * rsize )
```

Search for a IO memory region.

Parameters

<i>phys</i>	Physical address to look for
<i>size</i>	Size of requested memory area

Return values

<i>rstart</i>	Start address for region
<i>rsize</i>	Size of region in bytes

Returns

0 if an IO region was found, <0 if not

12.43 IPC-Gate API

Secure communication object.

Collaboration diagram for IPC-Gate API:



Functions

- [l4_msgtag_t l4_ipc_gate_bind_thread](#) ([l4_cap_idx_t](#) gate, [l4_cap_idx_t](#) thread, [l4_umword_t](#) label)
Bind the IPC gate to a thread.
- [l4_msgtag_t l4_ipc_gate_get_infos](#) ([l4_cap_idx_t](#) gate, [l4_umword_t](#) *label)
Get information about the IPC-gate.

12.43.1 Detailed Description

Secure communication object.

IPC-Gate objects provide a means to establish secure communication channels to [L4 Threads](#) ([Thread](#)). An IPC-Gate object can be created using a [Factory](#) ([l4_factory_create_gate\(\)](#)) and get assigned a specific [L4](#) thread and a *label* as protected payload. The *label* has the size of one machine word and can only be seen by the Task running the thread that is assigned of the IPC-gate. The *label* is received as part of the IPC message. The *label* can thus be used to securely identify the IPC-gate that was used to send a message.

An IPC-gate is usually used to represent an user-level object and may be the address of the data structure for the object in the server task.

With client privileges an IPC-gate does not provide any direct API and thus an IPC-gate kernel object cannot be modified by invocations. Each invocation of an IPC-gate kernel object is translated into an IPC message to the assigned thread.

Include File

```
#include <l4/sys/ipc_gate.h>
```

For the C++ interface refer to the [L4::ipc_gate](#) documentation.

See also

[Object Invocation](#)

12.43.2 Function Documentation

12.43.2.1 l4_ipc_gate_bind_thread()

```
l4_msgtag_t l4_ipc_gate_bind_thread (
    l4_cap_idx_t gate,
    l4_cap_idx_t thread,
    l4_umword_t label ) [inline]
```

Bind the IPC gate to a thread.

Parameters

<i>gate</i>	The IPC gate object.
<i>thread</i>	The thread object that shall be bound to <i>gate</i> .
<i>label</i>	Label to assign to <i>gate</i> . The two least significant bits should usually 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>	<i>thread</i> is missing L4_CAP_FPAGE_S right.

Definition at line 133 of file [ipc_gate.h](#).

12.43.2.2 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.

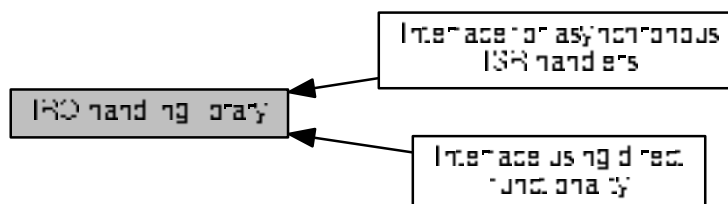
Returns

System call return tag.

Definition at line 140 of file [ipc_gate.h](#).

12.44 IRQ handling library

Collaboration diagram for IRQ handling library:



Modules

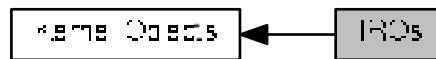
- [Interface for asynchronous ISR handlers.](#)
This interface has just two (main) functions.
- [Interface using direct functionality.](#)

12.44.1 Detailed Description

12.45 IRQs

C IRQ interface.

Collaboration diagram for IRQs:



Enumerations

- enum `L4_irq_mode` {
`L4_IRQ_F_NONE` = 0, `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` = 0x3, `L4_IRQ_F_LEVEL_LOW`
= 0x7,
`L4_IRQ_F_POS_EDGE` = 0x1, `L4_IRQ_F_NEG_EDGE` = 0x5, `L4_IRQ_F_BOTH_EDGE` = 0x9, `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_attach (l4_cap_idx_t irq, l4_umword_t label, l4_cap_idx_t thread) L4_NOTHROW`
Attach a thread to an interrupt source.
- `l4_msgtag_t l4_irq_attach_u (l4_cap_idx_t irq, l4_umword_t label, l4_cap_idx_t thread, l4_utcb_t *utcb) L4_NOTHROW`
Attach a thread to this interrupt.
- `l4_msgtag_t l4_irq_mux_chain (l4_cap_idx_t irq, l4_cap_idx_t slave) L4_NOTHROW`
Chain an IRQ to another master IRQ source.
- `l4_msgtag_t l4_irq_mux_chain_u (l4_cap_idx_t irq, l4_cap_idx_t slave, l4_utcb_t *utcb) L4_NOTHROW`
Attach an IRQ to this multiplexer.
- `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_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.
- `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 IRQ.

12.45.1 Detailed Description

C IRQ 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)

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.

12.45.2 Enumeration Type Documentation

12.45.2.1 L4_irq_mode

```
enum L4_irq_mode
```

Interrupt attributes.

Enumerator

L4_IRQ_F_NONE	Flow types. None
L4_IRQ_F_LEVEL	Level triggered.
L4_IRQ_F_EDGE	Edge triggered.
L4_IRQ_F_POS	Positive trigger.
L4_IRQ_F_NEG	Negative trigger.
L4_IRQ_F_BOTH	Both edges trigger.
L4_IRQ_F_LEVEL_HIGH	Level high trigger.
L4_IRQ_F_LEVEL_LOW	Level low trigger.
L4_IRQ_F_POS_EDGE	Positive edge trigger.
L4_IRQ_F_NEG_EDGE	Negative edge trigger.
L4_IRQ_F_BOTH_EDGE	Both edges trigger.
L4_IRQ_F_MASK	Mask.

Definition at line 67 of file [icu.h](#).

12.45.3 Function Documentation

12.45.3.1 l4_irq_attach()

```
l4_msgtag_t l4_irq_attach (
    l4_cap_idx_t irq,
    l4_umword_t label,
    l4_cap_idx_t thread ) [inline]
```

Attach a thread to an interrupt source.

Parameters

<i>irq</i>	IRQ object where <i>thread</i> is attached to.
<i>label</i>	Identifier of the IRQ.
<i>thread</i>	The thread object to attach <i>irq</i> to.

Returns

Syscall return tag

The *protected label* is stored in the kernel and sent to the attached thread with the IRQ-triggered notification. It allows the receiver thread to securely identify the IRQ.

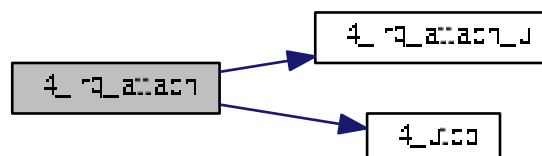
Examples:

[examples/sys/isr/main.c](#).

Definition at line 328 of file [irq.h](#).

References [l4_irq_attach_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



12.45.3.2 l4_irq_attach_u()

```
l4_msgtag_t l4_irq_attach_u (
    l4_cap_idx_t irq,
    l4_umword_t label,
    l4_cap_idx_t thread,
    l4_utcb_t * utcb ) [inline]
```

Attach a thread to this interrupt.

Parameters

<i>irq</i>	IRQ object where <i>thread</i> is attached to.
<i>label</i>	Identifier of the IRQ (<i>protected label</i> used for messages)
<i>thread</i>	Capability of the thread to attach the IRQ to.
<i>utcb</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

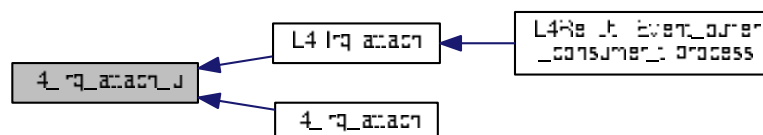
Syscall return tag

The *protected label* is stored in the kernel and sent to the attached thread with the IRQ-triggered notification. It allows the receiver thread to securely identify the IRQ.

Definition at line 258 of file [irq.h](#).

Referenced by [L4::l4::attach\(\)](#), and [l4_irq_attach\(\)](#).

Here is the caller graph for this function:



12.45.3.3 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

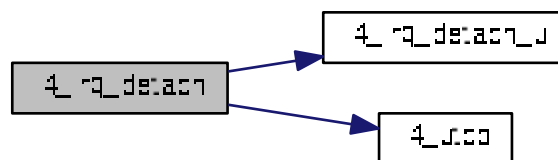
Examples:

[examples/sys/isr/main.c](#).

Definition at line 341 of file [irq.h](#).

References [l4_irq_detach_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



12.45.3.4 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, usually the UTCB of the calling thread.

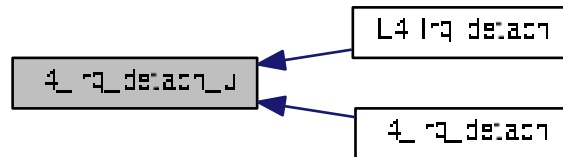
Returns

Syscall return tag

Definition at line 289 of file [irq.h](#).

Referenced by [L4::Irq::detach\(\)](#), and [l4_irq_detach\(\)](#).

Here is the caller graph for this function:



12.45.3.5 l4_irq_mux_chain()

```
l4_msgtag_t l4_irq_mux_chain (
    l4_cap_idx_t irq,
    l4_cap_idx_t slave ) [inline]
```

Chain an IRQ to another master IRQ source.

Parameters

<i>irq</i>	The master IRQ object.
<i>slave</i>	The slave that shall be attached to the master.

Returns

Syscall return tag

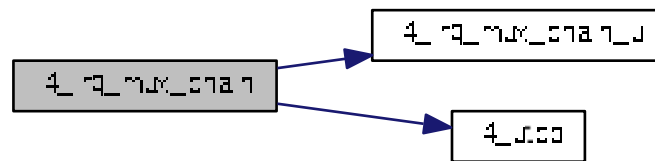
The chaining feature of IRQ objects allows to deal with shared IRQs. For chaining IRQs there must be a master IRQ object, bound to the real IRQ source. Note, the master IRQ must not have a thread attached to it.

This function allows to add a limited number of slave IRQs to this master IRQ, with the semantics that each of the slave IRQs is triggered whenever the master IRQ is triggered. The master IRQ will be masked automatically when an IRQ is delivered and shall be unmasked when all attached slave IRQs are unmasked.

Definition at line 335 of file [irq.h](#).

References [l4_irq_mux_chain_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



12.45.3.6 l4_irq_mux_chain_u()

```

l4_msgtag_t l4_irq_mux_chain_u (
    l4_cap_idx_t irq,
    l4_cap_idx_t slave,
    l4_utcb_t * utcb ) [inline]
  
```

Attach an IRQ to this multiplexer.

Parameters

<i>irq</i>	The master IRQ object.
<i>slave</i>	The slave that shall be attached to the master.
<i>utcb</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

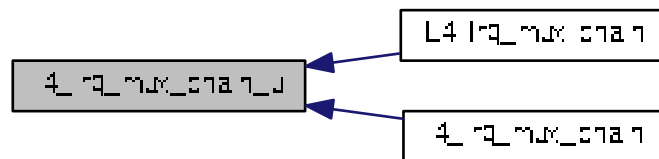
Syscall return tag

The chaining feature of IRQ objects allows to deal with shared IRQs. For chaining IRQs there must be an IRQ multiplexer (`l4_irq_mux`) bound to the real IRQ source. This function allows to add slave IRQs to this multiplexer.

Definition at line 277 of file `irq.h`.

Referenced by `L4::l4_irq_mux::chain()`, and `l4_irq_mux_chain()`.

Here is the caller graph for this function:



12.45.3.7 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

Examples:

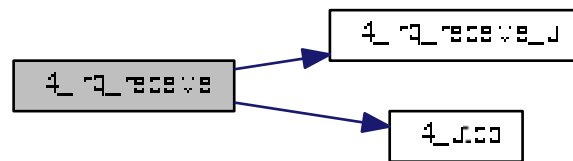
[examples/sys/isr/main.c](#).

Definition at line 353 of file [irq.h](#).

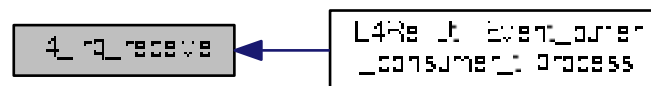
References [l4_irq_receive_u\(\)](#), and [l4_utcb\(\)](#).

Referenced by [L4Re::Util::Event_buffer_consumer_t< PAYLOAD >::process\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



12.45.3.8 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, usually the UTCB of the calling thread.

Returns

Syscall return tag

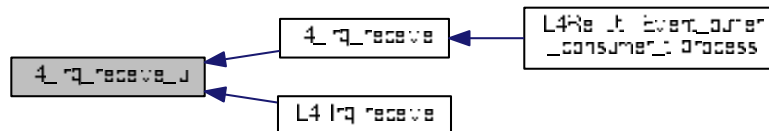
Note

If this is the function normally used for your IRQs consider using [L4::Semaphore](#) instead of [L4::Irq](#).

Definition at line 304 of file [irq.h](#).

Referenced by [l4_irq_receive\(\)](#), and [L4::Irq::receive\(\)](#).

Here is the caller graph for this function:

**12.45.3.9 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.

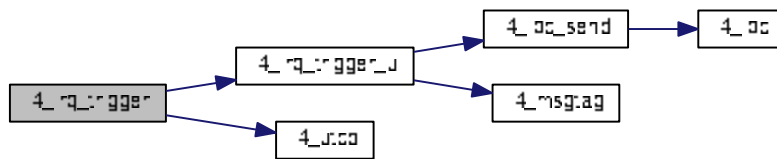
Note that this function is a send only operation, i.e. there is no return value except for a failed send operation. Especially [l4_error\(\)](#) will return an error value from the message tag which still contains the IRQ protocol used for the send operation.

Use [l4_ipc_error\(\)](#) to check for (send) errors.

Definition at line 347 of file [irq.h](#).

References [l4_irq_trigger_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



12.45.3.10 l4_irq_trigger_u()

```

l4_msgtag_t l4_irq_trigger_u (
    l4_cap_idx_t irq,
    l4_utcb_t * utcb ) [inline]
  
```

Trigger.

Parameters

<i>irq</i>	The IRQ object that shall be triggered.
<i>utcb</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

Syscall return tag for a send-only operation, use [l4_ipc_error\(\)](#) to check for errors (**do not** use [l4_error\(\)](#)).

Note

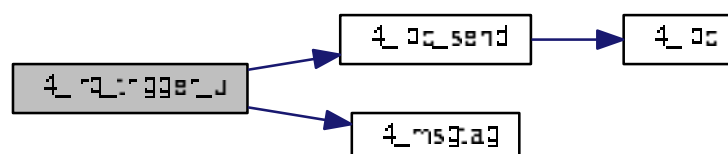
This function is a send-only operation, this means there is no return value except for a failed send operation. Use [l4_ipc_error\(\)](#) to check for errors, **do not** use [l4_error\(\)](#), because [l4_error\(\)](#) will always return an error.

Definition at line 297 of file [irq.h](#).

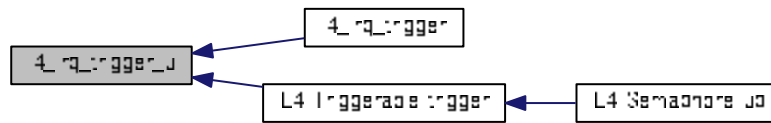
References [L4_IPC_BOTH_TIMEOUT_0](#), [l4_ipc_send\(\)](#), [l4_msgtag\(\)](#), and [L4_PROTO_IRQ](#).

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:



12.45.3.11 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

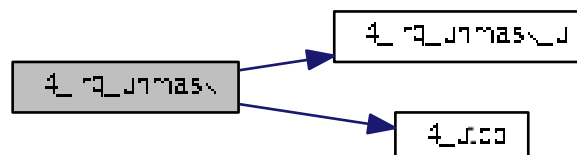
Note

`l4_irq_wait()` and `l4_irq_receive()` are doing the unmask themselves.

Definition at line 366 of file `irq.h`.

References `l4_irq_unmask_u()`, and `l4_utcb()`.

Here is the call graph for this function:



12.45.3.12 l4_irq_unmask_u()

```
l4_msgtag_t l4_irq_unmask_u (
    l4_cap_idx_t irq,
    l4_utcb_t * utcb ) [inline]
```

Unmask IRQ.

Parameters

<i>irq</i>	The IRQ object that shall be unmasked.
<i>utcb</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

Syscall return tag for a send-only operation, use [l4_ipc_error\(\)](#) to check for errors (**do not** use [l4_error\(\)](#)).

Note

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 errors, **do not** use [l4_error\(\)](#), because [l4_error\(\)](#) will always return an error.

[Irq::wait\(\)](#) and [Irq::receive\(\)](#) operations already include an [unmask\(\)](#), do not use an extra [unmask\(\)](#) in these cases.

Deprecated Use [L4::Irq_eoi::unmask\(\)](#)

Definition at line 320 of file [irq.h](#).

Referenced by [l4_irq_unmask\(\)](#).

Here is the caller graph for this function:



12.45.3.13 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.

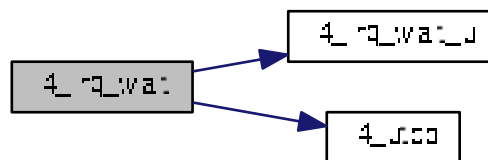
Returns

Syscall return tag

Definition at line 359 of file [irq.h](#).

References [l4_irq_wait_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



12.45.3.14 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, usually the UTCB of the calling thread.

Returns

Syscall return tag

Definition at line 311 of file [irq.h](#).

Referenced by [l4_irq_wait\(\)](#).

Here is the caller graph for this function:



12.46 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 * 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.

12.46.1 Detailed Description

C interface of the initial environment that is provided to an [L4](#) task.

Include File

```
#include <l4/re/env.h>
```

For the C++ interface refer to [L4Re::Env](#).

12.46.2 Function Documentation

12.46.2.1 `l4re_env()`

```
l4re_env_t * l4re_env (
    void ) [inline]
```

Get [L4Re](#) initial environment.

Returns

Pointer to [L4Re](#) initial environment.

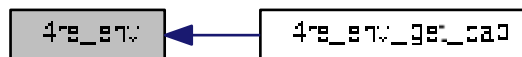
Examples:

[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 [181](#) of file [env.h](#).

Referenced by [l4re_env_get_cap\(\)](#).

Here is the caller graph for this function:



12.46.2.2 `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\(\)](#)).

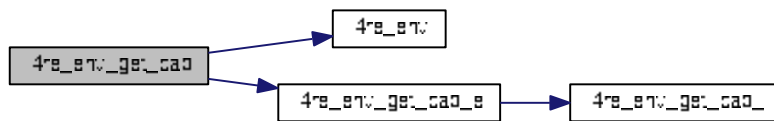
Examples:

[examples/sys/isr/main.c](#).

Definition at line 223 of file [env.h](#).

References [l4re_env\(\)](#), [l4re_env_get_cap_e\(\)](#), and [l4re_env_cap_entry_t::name](#).

Here is the call graph for this function:

**12.46.2.3 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 210 of file [env.h](#).

References [l4re_env_cap_entry_t::cap](#), [L4_INVALID_CAP](#), [l4re_env_get_cap_l\(\)](#), and [l4re_env_cap_entry_t::name](#).

Referenced by [l4re_env_get_cap\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



12.46.2.4 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

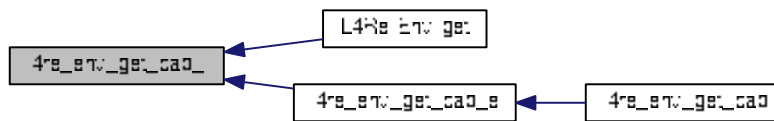
A pointer to an [l4re_env_cap_entry_t](#) if the object exists or NULL if not.

Definition at line 192 of file [env.h](#).

References [l4re_env_cap_entry_t::flags](#), 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:



12.46.2.5 l4re_kip()

```
l4_kernel_info_t * l4re_kip (
    void ) [inline]
```

Get Kernel Info Page.

Returns

Pointer to Kernel Info Page (KIP) structure.

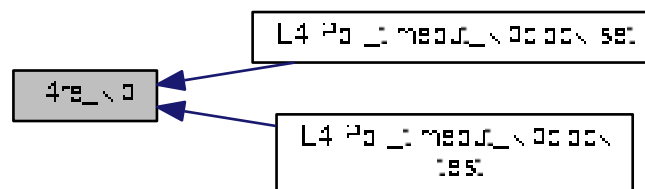
Examples:

[examples/sys/aliens/main.c](#), and [examples/sys/ux-vhw/main.c](#).

Definition at line 185 of file [env.h](#).

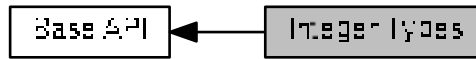
Referenced by [L4::Poll_timeout_kipclock::set\(\)](#), and [L4::Poll_timeout_kipclock::test\(\)](#).

Here is the caller graph for this function:



12.47 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, amd64 version.
- file [l4int.h](#)
Fixed sized integer types, x86 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 32`
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 int [l4_size_t](#)

Unsigned size type.

- typedef signed int [l4_ssize_t](#)

Signed size type.

12.47.1 Detailed Description

Include File

```
#include <l4/sys/l4int.h>
```

12.47.2 Typedef Documentation

12.47.2.1 l4_int16_t

```
typedef signed short int l4\_int16\_t
```

Signed 16bit value.

Definition at line 37 of file [l4int.h](#).

12.47.2.2 l4_int32_t

```
typedef signed int l4_int32_t
```

Signed 32bit value.

Definition at line 39 of file [l4int.h](#).

12.47.2.3 l4_int64_t

```
typedef signed long long l4_int64_t
```

Signed 64bit value.

Definition at line 41 of file [l4int.h](#).

12.47.2.4 l4_int8_t

```
typedef signed char l4_int8_t
```

Signed 8bit value.

Definition at line 35 of file [l4int.h](#).

12.47.2.5 l4_uint16_t

```
typedef unsigned short int l4_uint16_t
```

Unsigned 16bit value.

Definition at line 38 of file [l4int.h](#).

12.47.2.6 l4_uint32_t

```
typedef unsigned int l4_uint32_t
```

Unsigned 32bit value.

Definition at line 40 of file [l4int.h](#).

12.47.2.7 l4_uint64_t

```
typedef unsigned long long l4_uint64_t
```

Unsigned 64bit value.

Definition at line 42 of file [l4int.h](#).

12.47.2.8 l4_uint8_t

```
typedef unsigned char l4_uint8_t
```

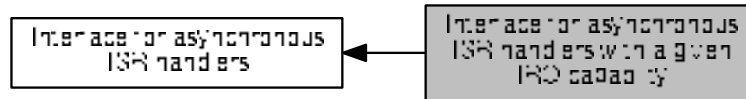
Unsigned 8bit value.

Definition at line 36 of file [l4int.h](#).

12.48 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.

12.48.1 Detailed Description

This group is just an enhanced version to [l4irq_request\(\)](#) which takes a capability object instead of a plain number.

12.48.2 Function Documentation

12.48.2.1 l4irq_request_cap()

```

l4irq_t* l4irq_request_cap (
    l4_cap_idx_t irqcap,
    void(*) (void *) isr_handler,
    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 <i>isr_handler</i>
<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](#)

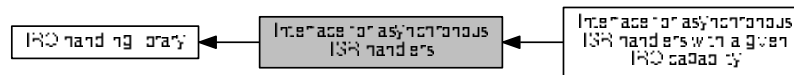
Returns

Pointer to `l4irq_t` structure, 0 on error

12.49 Interface for asynchronous ISR handlers.

This interface has just two (main) functions.

Collaboration diagram for Interface for asynchronous ISR handlers.:



Modules

- [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.

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.

12.49.1 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.

12.49.2 Function Documentation

12.49.2.1 `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.](#)

12.49.2.2 l4irq_request()

```
l4irq_t* l4irq_request (
    int irqnum,
    void(*) (void *) isr_handler,
    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

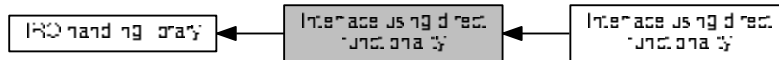
Pointer to l4irq_t structure, 0 on error

Examples:

[examples/libs/libirq/async_isr.c.](#)

12.50 Interface using direct functionality.

Collaboration diagram for Interface using direct functionality.:



Modules

- [Interface using direct functionality.](#)

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.

12.50.1 Detailed Description

12.50.2 Function Documentation

12.50.2.1 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](#).

12.50.2.2 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.

12.50.2.3 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.

12.50.2.4 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.

12.50.2.5 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

12.50.2.6 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.

12.50.2.7 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.
-------------------	--------------------------------

Return values

<i>ret_irq</i>	Received interrupt.
----------------	---------------------

Returns

0 on success, != 0 on error

12.50.2.8 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](#).

12.50.2.9 l4irq_wait_any()

```
long l4irq_wait_any (
    l4irq_t ** irq )
```

Wait for any attached IRQ.

Return values

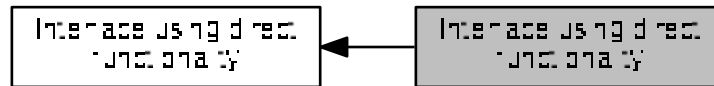
<i>irq</i>	Received interrupt.
------------	---------------------

Returns

0 on success, != 0 on error

12.51 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.

12.51.1 Detailed Description

12.51.2 Function Documentation

12.51.2.1 `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.

12.51.2.2 `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.

12.51.2.3 `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.

12.51.2.4 `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,

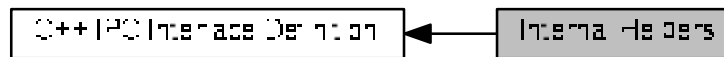
See also[L4_irq_mode](#)**Returns**

Pointer to `l4irq_t` structure, 0 on error

The pointer to the IRQ structure is used as a label in the IRQ object.

12.52 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.

12.52.1 Detailed Description

12.53 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_TBUF` 0xB0
TBUF.
- #define `SIGMA0_REQ_ID_DEBUG_DUMP` 0xC0
Debug dump.
- #define `SIGMA0_REQ_ID_NEW_CLIENT` 0xD0
New client.
- #define `SIGMA0_IS_MAGIC_REQ(d1)` ((d1 & `SIGMA0_REQ_MASK`) == `SIGMA0_REQ_MAGIC`)
Check if magic.
- #define `SIGMA0_REQ(x)` (`SIGMA0_REQ_MAGIC` + `SIGMA0_REQ_ID_## 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_TBUF` (`SIGMA0_REQ(TBUF)`)
- TBUF.*
 - #define `SIGMA0_REQ_DEBUG_DUMP` (`SIGMA0_REQ(DEBUG_DUMP)`)
- Debug dump.*
 - #define `SIGMA0_REQ_NEW_CLIENT` (`SIGMA0_REQ(NEW_CLIENT)`)
- New client.*

12.53.1 Detailed Description

Internal sigma0 definitions.

12.54 Internal functions

Collaboration diagram for Internal functions:



Functions

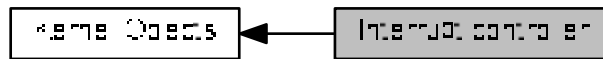
- void [base64_encode](#) (const char *infile, unsigned int in_size, char **outfile)
base-64-encode string infile
- void [base64_decode](#) (const char *infile, unsigned int in_size, char **outfile)
decode base-64-encoded string infile

12.54.1 Detailed Description

12.55 Interrupt controller

The C Icu 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 capabilities of the ICU.

- [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 capabilities of the ICU.

- [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](#)

Acknowledge 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.

12.55.1 Detailed Description

The C Icu interface.

To setup an IRQ line the following steps are required:

1. [l4_icu_set_mode\(\)](#) (optional if IRQ has a default mode)
2. [l4_irq_attach\(\)](#) to attach the IRQ capability to a thread
3. [l4_icu_bind\(\)](#)
4. [l4_icu_unmask\(\)](#) to receive the first IRQ

Include File

```
#include <l4/sys/icu.h>
```

12.55.2 Typedef Documentation

12.55.2.1 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\(\)](#).

12.55.3 Enumeration Type Documentation

12.55.3.1 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 50 of file [icu.h](#).

12.55.4 Function Documentation

12.55.4.1 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\(\)](#).

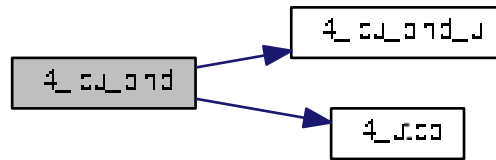
Examples:

[examples/sys/isr/main.c](#).

Definition at line 473 of file [icu.h](#).

References [l4_icu_bind_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



12.55.4.2 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 objecte to bind <i>irq</i> 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, usually the UTCB of the calling thread.

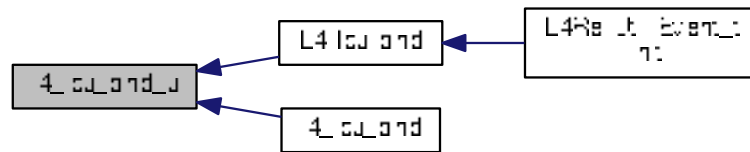
Returns

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](#).

Definition at line 373 of file [icu.h](#).

Referenced by [L4::icu::bind\(\)](#), and [l4_icu_bind\(\)](#).

Here is the caller graph for this function:



12.55.4.3 l4_icu_info()

```

l4_msgtag_t l4_icu_info (
    l4_cap_idx_t icu,
    l4_icu_info_t * info ) [inline]
  
```

Get information about the capabilities of the ICU.

Parameters

	<i>icu</i>	The ICU object from which information shall be retrieved.
out	<i>info</i>	Pointer to an info structure to be filled with information. The memory for this structure has to be allocated by the caller.

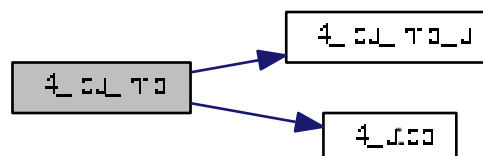
Returns

Syscall return tag

Definition at line 481 of file `icu.h`.

References `l4_icu_info_u()`, and `l4_utcb()`.

Here is the call graph for this function:



12.55.4.4 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 capabilities of the ICU.

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, usually the UTCB of the calling thread.

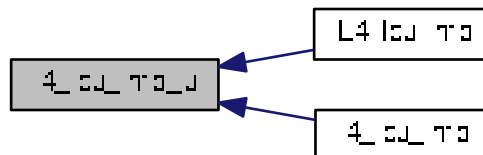
Returns

Syscall return tag

Definition at line 397 of file [icu.h](#).

Referenced by [L4::Icu::info\(\)](#), and [l4_icu_info\(\)](#).

Here is the caller graph for this function:



12.55.4.5 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 'irqnum' shall be masked.
<i>irqnum</i>	IRQ line at the ICU.
<i>label</i>	If non-NULL the function also waits for the next message.
<i>to</i>	Timeout for message to ICU, if unsure use L4_IPC_NEVER.

Returns

Syscall return tag

Definition at line 495 of file [icu.h](#).

12.55.4.6 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 'irqnum' 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.
<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, usually the UTCB of the calling thread.

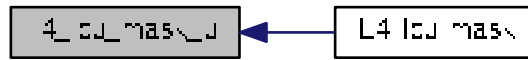
Returns

Syscall return tag

Definition at line 460 of file [icu.h](#).

Referenced by [L4::Icu::mask\(\)](#).

Here is the caller graph for this function:



12.55.4.7 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 l4_icu_msi_info_t structure receiving the address and the data value to trigger this MSI.

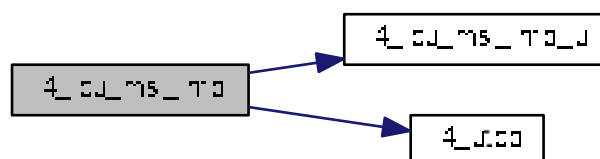
Returns

Syscall return tag

Definition at line 485 of file [icu.h](#).

References [l4_icu_msi_info_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



12.55.4.8 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, usually the UTCB of the calling thread.
	<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 411 of file [icu.h](#).

Referenced by [l4_icu_msi_info\(\)](#).

Here is the caller graph for this function:



12.55.4.9 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 .

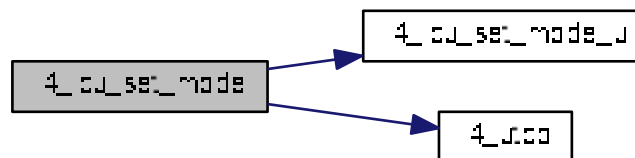
Returns

Syscall return tag

Definition at line 500 of file [icu.h](#).

References [l4_icu_set_mode_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:

12.55.4.10 `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, usually the UTCB of the calling thread.

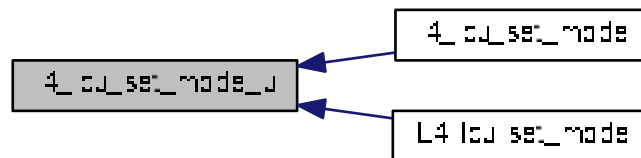
Returns

Syscall return tag

Definition at line 434 of file [icu.h](#).

Referenced by [l4_icu_set_mode\(\)](#), and [L4::Icu::set_mode\(\)](#).

Here is the caller graph for this function:

**12.55.4.11 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.

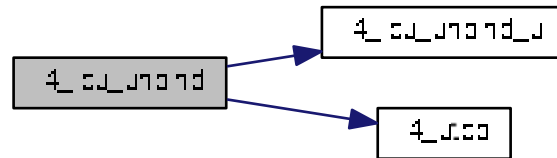
Returns

Syscall return tag

Definition at line 477 of file [icu.h](#).

References [l4_icu_unbind_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



12.55.4.12 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, usually the UTCB of the calling thread.

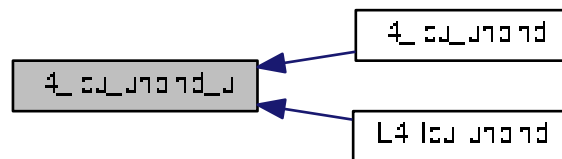
Returns

Syscall return tag

Definition at line 385 of file [icu.h](#).

Referenced by [l4_icu_unbind\(\)](#), and [L4::Icu::unbind\(\)](#).

Here is the caller graph for this function:



12.55.4.13 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 waits for the next message.
<i>to</i>	Timeout for message to ICU, if unsure use L4_IPC_NEVER.

Returns

Syscall return tag, the error values therein are undefined because `l4_icu_unmask()` is a sender-only IPC.

Definition at line 490 of file `icu.h`.

12.55.4.14 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]
```

Acknowledge the given interrupt line.

Parameters

	<i>icu</i>	The ICU object where the IRQ line shall be unmasked.
	<i>irqnum</i>	The interrupt line that shall be acknowledged.
out	<i>label</i>	If NULL this is a send-only unmask, if not NULL then 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, usually the UTCB of the calling thread.

Returns

Syscall return tag.

Note

If *label* is NULL this function is a send-only operation and there is no return value except for a failed send operation. In this case use [l4_ipc_error\(\)](#) to check for errors, **do not** use [l4_error\(\)](#), because [l4_error\(\)](#) will always return an error.

Definition at line [465](#) of file [icu.h](#).

12.56 Kernel Debugger

Kernel debugger related functionality.

Collaboration diagram for 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.
- [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.

12.56.1 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>
```

12.56.2 Function Documentation

12.56.2.1 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

$\sim 0UL$	Capability is not valid.
≥ 0	Global debugger id.

This is a debugging facility, the call might be invalid.

Definition at line 331 of file [debugger.h](#).

12.56.2.2 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

$\sim 0UL$	The capability or the kobject pointer are invalid.
≥ 0	The globally unique id.

This is a debugging facility, the call might be invalid.

Definition at line 337 of file [debugger.h](#).

12.56.2.3 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:

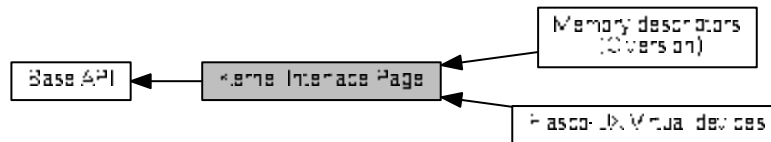
[examples/sys/aliens/main.c](#).

Definition at line [324](#) of file [debugger.h](#).

12.57 Kernel Interface Page

Kernel Interface Page.

Collaboration diagram for Kernel Interface Page:



Modules

- [Fiasco-UX Virtual devices](#)
Virtual hardware devices, provided by Fiasco-UX.
- [Memory descriptors \(C version\)](#)
C Interface for KIP memory descriptors.

Data Structures

- class [L4::Kip::Mem_desc](#)
Memory descriptors stored in the kernel interface page.

Macros

- `#define L4_KERNEL_INFO_MAGIC (0x4BE6344CL) /* "L4μK" */`
Kernel Info Page identifier ("L4μK").

Functions

- `l4_umword_t l4_kip_version (l4_kernel_info_t *kip) L4_NOTHROW`
Get the kernel version.
- `const char * l4_kip_version_string (l4_kernel_info_t *kip) L4_NOTHROW`
Get the kernel version string.
- `int l4_kernel_info_version_offset (l4_kernel_info_t *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 *kip) L4_NOTHROW`
Return clock value from the KIP.
- `l4_umword_t l4_kip_clock_lw (l4_kernel_info_t *kip) L4_NOTHROW`
Return least significant machine word of clock value from the KIP.

12.57.1 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>
```

12.57.2 Function Documentation

12.57.2.1 l4_kernel_info_version_offset()

```
int l4_kernel_info_version_offset (
    l4_kernel_info_t * 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).
------------	--

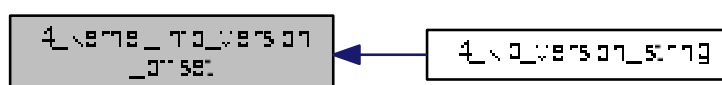
Returns

offset of version_strings relative to the KIP base address, in bytes.

Definition at line 134 of file [kip.h](#).

Referenced by [l4_kip_version_string\(\)](#).

Here is the caller graph for this function:



12.57.2.2 l4_kip_clock()

```
l4_cpu_time_t l4_kip_clock (
    l4_kernel_info_t * 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.

Definition at line 138 of file [kip.h](#).

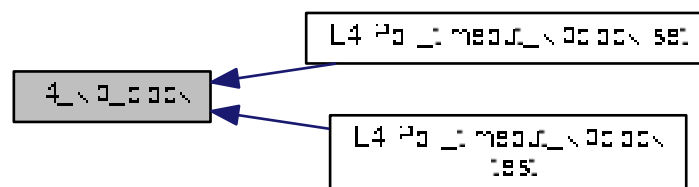
References [l4_mb\(\)](#).

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:



12.57.2.3 l4_kip_clock_lw()

```
l4_umword_t l4_kip_clock_lw (
    l4_kernel_info_t * 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.

Definition at line 160 of file [kip.h](#).

References [l4_mb\(\)](#).

Here is the call graph for this function:



12.57.2.4 l4_kip_version()

```
l4_umword_t l4_kip_version (
    l4_kernel_info_t * 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 126 of file [kip.h](#).

12.57.2.5 l4_kip_version_string()

```
const char * l4_kip_version_string (  
    l4_kernel_info_t * kip )    [inline]
```

Get the kernel version string.

Parameters

<i>kip</i>	Kernel Info Page.
------------	-------------------

Returns

Kernel version string.

Definition at line 130 of file [kip.h](#).

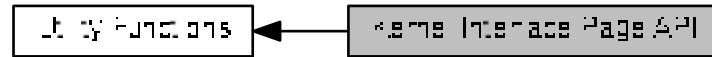
References [l4_kernel_info_version_offset\(\)](#).

Here is the call graph for this function:



12.58 Kernel Interface Page API

Collaboration diagram for Kernel Interface Page API:



Files

- file [kip.h](#)

Macros

- `#define l4util_kip_for_each_feature(s)` for (`s += strlen(s) + 1; *s; s += strlen(s) + 1`)
Cycle through kernel features given in the KIP.

Functions

- `int l4util_kip_kernel_is_ux (l4_kernel_info_t *)`
Return whether the kernel is running native or under UX.
- `int l4util_kip_kernel_has_feature (l4_kernel_info_t *, const char *str)`
Check if kernel supports a feature.
- `unsigned long l4util_kip_kernel_abi_version (l4_kernel_info_t *)`
Return kernel ABI version.
- `l4_addr_t l4util_memdesc_vm_high (l4_kernel_info_t *kinfo)`
Return end of virtual memory.

12.58.1 Detailed Description

12.58.2 Macro Definition Documentation

12.58.2.1 l4util_kip_for_each_feature

```
#define l4util_kip_for_each_feature(  
    s ) for ( s += strlen(s) + 1; *s; s += 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 *`) initialized with `l4util_kip_version_string()`.

Definition at line 74 of file [kip.h](#).

12.58.3 Function Documentation

12.58.3.1 l4util_kip_kernel_abi_version()

```
unsigned long l4util_kip_kernel_abi_version (
    l4_kernel_info_t * )
```

Return kernel ABI version.

Returns

Kernel ABI version.

12.58.3.2 l4util_kip_kernel_has_feature()

```
int l4util_kip_kernel_has_feature (
    l4_kernel_info_t * ,
    const char * str )
```

Check if kernel supports a feature.

Parameters

<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. The KIP will be mapped if not already mapped. The KIP will not be unmapped again.

12.58.3.3 l4util_kip_kernel_is_ux()

```
int l4util_kip_kernel_is_ux (
    l4_kernel_info_t * )
```

Return whether the kernel is running native or under UX.

Returns whether the kernel is running natively or under UX. The KIP will be mapped if not already mapped. The KIP will not be unmapped again.

Returns

1 when running under UX, 0 if not running under UX

Examples:

[examples/sys/ux-vhw/main.c](#).

12.58.3.4 l4util_memdesc_vm_high()

```
l4_addr_t l4util_memdesc_vm_high (
    l4_kernel_info_t * kinfo )
```

Return end of virtual memory.

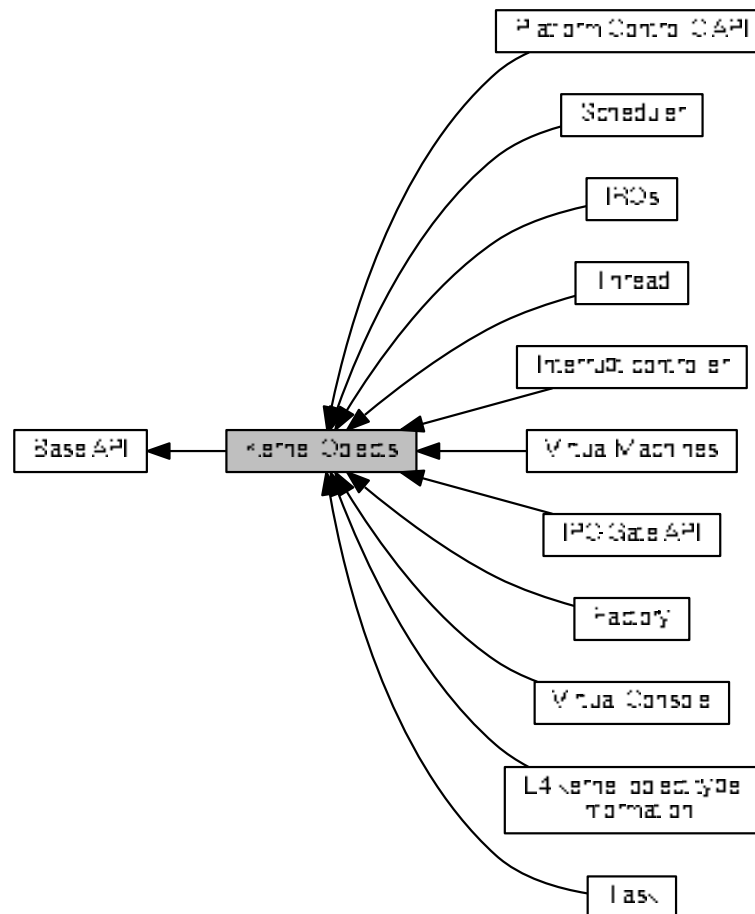
Returns

0 if memory descriptor could not be found, last address of address space otherwise

12.59 Kernel Objects

API of kernel objects.

Collaboration diagram for Kernel Objects:



Modules

- [Factory](#)
C factory interface to create kernel objects.
- [IPC-Gate API](#)
Secure communication object.
- [IRQs](#)
C IRQ interface.
- [Interrupt controller](#)
The C Icu interface.
- [L4 kernel object type information](#)
Type information for [L4](#) server objects that can be called via IPC.

- [Platform Control C API](#)

C interface for controlling platform-wide properties.

- [Scheduler](#)

C interface of the Scheduler kernel object.

- [Task](#)

C interface of the Task kernel object.

- [Thread](#)

Thread object.

- [Virtual Console](#)

Virtual console for simple character based input and output.

- [Virtual Machines](#)

Virtual Machine API.

Data Structures

- class [L4::Kobject](#)

Base class for all kinds of kernel objects and remote objects, referenced by capabilities.

12.59.1 Detailed Description

API of kernel objects.

Include File

```
#include <l4/sys/kernel_object.h>
```

12.60 Kumem allocator utility

Kumem allocator utility C interface.

Collaboration diagram for Kumem allocator utility:



Functions

- int [l4re_util_kumem_alloc](#) ([l4_addr_t](#) *mem, unsigned pages_order, [l4_cap_idx_t](#) task, [l4_cap_idx_t](#) regmgr)
[L4_NOTHROW](#)
Allocate state area.

12.60.1 Detailed Description

Kumem allocator utility C interface.

12.60.2 Function Documentation

12.60.2.1 l4re_util_kumem_alloc()

```

int l4re_util_kumem_alloc (
    l4\_addr\_t * mem,
    unsigned pages_order,
    l4\_cap\_idx\_t task,
    l4\_cap\_idx\_t regmgr )
  
```

Allocate state area.

Return values

<i>mem</i>	Pointer to memory that has been allocated.
<i>pages_order</i>	Size to allocate, in log2 pages.

Parameters

<i>task</i>	Task to use for allocation.
-------------	-----------------------------

Parameters

<i>regmgr</i>	Region manager to use for allocation.
---------------	---------------------------------------

Returns

0 for success, error code otherwise

Examples:

[examples/sys/aliens/main.c](#).

12.61 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()) throw ()`
Allocate state area.

12.61.1 Detailed Description

12.61.2 Function Documentation

12.61.2.1 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() ) throw ()

```

Allocate state area.

Return values

<i>mem</i>	Pointer to memory that has been allocated.
------------	--

Parameters

<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.

Returns

0 for success, error code otherwise

12.62 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_ENABLE_OP](#) = 3UL, [L4_PLATFORM_CTL_CPU_DISABLE_OP](#) = 4UL }
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 }
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_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.

12.62.1 Detailed Description

List of protocol specific opcodes used for communication with [L4Re](#) and Kernel objects.

12.62.2 Enumeration Type Documentation

12.62.2.1 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 93 of file [icu.h](#).

12.62.2.2 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 91 of file [ipc_gate.h](#).

12.62.2.3 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_ENABLE_OP	enable an offline CPU
L4_PLATFORM_CTL_CPU_DISABLE_OP	disable an online CPU

Definition at line [135](#) of file [platform_control.h](#).

12.62.2.4 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

Definition at line [273](#) of file [task.h](#).

12.62.2.5 L4_thread_ops

enum [L4_thread_ops](#)

Operations on thread objects.

Enumerator

L4_THREAD_CONTROL_OP	Control operation.
----------------------	--------------------

Enumerator

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_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 612 of file [thread.h](#).

12.62.2.6 L4_vcon_ops

enum [L4_vcon_ops](#)

Operations on vcon objects.

Enumerator

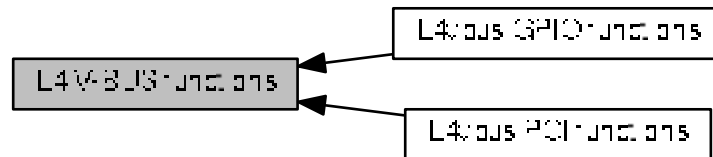
L4_VCON_WRITE_OP	Write.
L4_VCON_READ_OP	Read.
L4_VCON_SET_ATTR_OP	Get console attributes.
L4_VCON_GET_ATTR_OP	Set console attributes.

Definition at line 264 of file [vcon.h](#).

12.63 L4 V-BUS functions

C interface of the Vbus API.

Collaboration diagram for L4 V-BUS functions:



Modules

- [L4vbus GPIO functions](#)
- [L4vbus PCI functions](#)

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

- 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 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, int 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_request_resource](#) ([l4_cap_idx_t](#) vbus, [l4vbus_resource_t](#) const *res, int flags)
Request a resource of a specific type.
- 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 ([L4::Task](#)) or a [L4Re::Dma_space](#) to a DMA domain.
- int [l4vbus_release_resource](#) ([l4_cap_idx_t](#) vbus, [l4vbus_resource_t](#) const *res)
Release a previously requested 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.

12.63.1 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.

12.63.2 Enumeration Type Documentation

12.63.2.1 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 153 of file [vbus.h](#).

12.63.3 Function Documentation

12.63.3.1 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 ([L4::Task](#)) 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 <code>~0U</code> 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 <code>L4_PROTO_DMA_SPACE</code>) and the type must be reflected in the <code>flags</code> .

Return values

<code>0</code>	Operation completed successfully.
<code>-L4_ENOENT</code>	The vbus does not support a global DMA domain or no DMA domain could be found.
<code>-L4_EINVAL</code>	Invalid argument used.
<code>-L4_EBUSY</code>	DMA domain is already active, this means another DMA space is already assigned.

12.63.3.2 `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 after a successful call.

Return values

<code>0</code>	Success.
<code>-L4_ENODEV</code>	No device with the given device handle <code>dev</code> could be found.

Referenced by [L4vbus::Device::device\(\)](#).

Here is the caller graph for this function:



12.63.3.3 l4vbus_get_device_by_hid()

```

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 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 ID.

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

<i>>=</i>	0 A device with the given HID was found.
<i>-L4_ENOENT</i>	No device with the given HID could be found on the vbus.

Return values

<code>-L4_EINVAL</code>	Invalid or no HID provided.
<code>-L4_ENODEV</code>	Function called on a non-existing device.

12.63.3.4 `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

12.63.3.5 `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 0 for the system bus)
	<i>child</i>	Handle to the child device (use 0 to get the first child)
	<i>depth</i>	Depth to look for
out	<i>devinfo</i>	device information (might be NULL)

Returns

0 on success, else failure

12.63.3.6 l4vbus_get_resource()

```
int l4vbus_get_resource (
    l4_cap_idx_t vbus,
    l4vbus_device_handle_t dev,
    int 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.

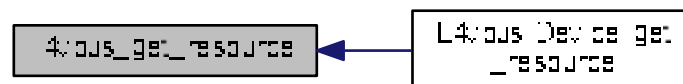
This function returns the resource descriptor of an individual device resource selected by the *res_idx* parameter.

Return values

0	Success.
-L4_ENOENT	Invalid resource index <i>res_idx</i> .

Referenced by [L4vbus::Device::get_resource\(\)](#).

Here is the caller graph for this function:



12.63.3.7 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

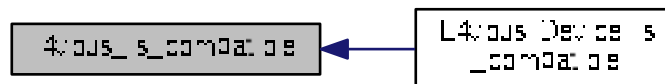
<i>vbus</i>	Capability of the system bus
<i>dev</i>	device handle for which the CID shall be tested
<i>cid</i>	the compatibility ID to test

Returns

1 when the given ID (*cid*) matches this device, 0 when the given ID does not match, <0 on error.

Referenced by [L4vbus::Device::is_compatible\(\)](#).

Here is the caller graph for this function:



12.63.3.8 l4vbus_release_resource()

```
int l4vbus_release_resource (
    l4_cap_idx_t vbus,
    l4vbus_resource_t const * res )
```

Release a previously requested resource.

Parameters

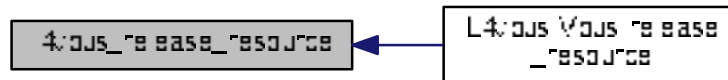
<i>vbus</i>	Capability of the system bus.
<i>res</i>	Descriptor of the resource.

Returns

0 on success, else failure

Referenced by [L4vbus::Vbus::release_resource\(\)](#).

Here is the caller graph for this function:

**12.63.3.9 l4vbus_request_resource()**

```
int l4vbus_request_resource (
    l4_cap_idx_t vbus,
    l4vbus_resource_t const * res,
    int flags )
```

Request a resource of a specific type.

Parameters

<i>vbus</i>	Capability of the system bus
<i>res</i>	Descriptor of the resource
<i>flags</i>	Optional flags

Returns

0 on success, else failure

If any resource is found that contains the requested type and addresses this resource is returned.

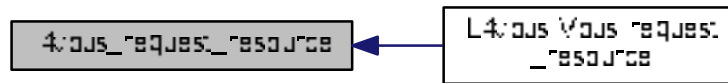
Flags are only relevant to control the memory caching. If io-memory is requested.

Returns

0 on success, else failure

Referenced by [L4vbus::Vbus::request_resource\(\)](#).

Here is the caller graph for this function:



12.63.3.10 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

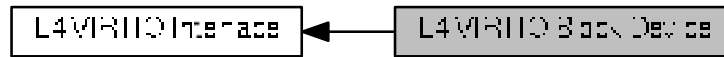
Referenced by [L4vbus::Icu::vicu\(\)](#).

Here is the caller graph for this function:



12.64 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_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_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 }
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.

12.64.1 Detailed Description

12.64.2 Enumeration Type Documentation

12.64.2.1 L4virtio_block_operations

```
enum L4virtio\_block\_operations
```

Kinds of operation over a block device.

Enumerator

L4VIRTIO_BLOCK_T_IN	Read from device.
L4VIRTIO_BLOCK_T_OUT	Write to device.
L4VIRTIO_BLOCK_T_FLUSH	Flush data to disk.
L4VIRTIO_BLOCK_T_GET_ID	Get device ID.

Definition at line 31 of file [virtio_block.h](#).

12.64.2.2 L4virtio_block_status

enum [L4virtio_block_status](#)

Status of a finished block request.

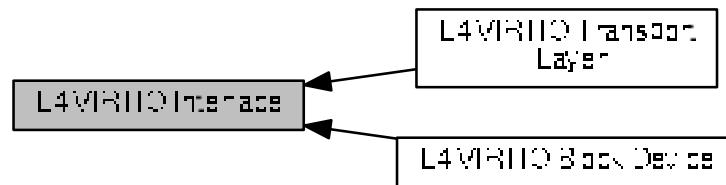
Enumerator

L4VIRTIO_BLOCK_S_OK	Request finished successfully.
L4VIRTIO_BLOCK_S_IOERR	IO error on device.
L4VIRTIO_BLOCK_S_UNSUPP	Operation is not supported.

Definition at line 42 of file [virtio_block.h](#).

12.65 L4 VIRTIO Interface

Collaboration diagram for L4 VIRTIO Interface:



Modules

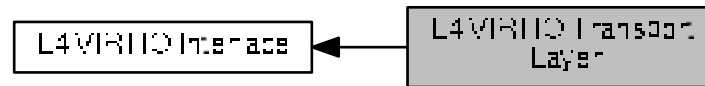
- [L4 VIRTIO Block Device](#)
- [L4 VIRTIO Transport Layer](#)
L4 specific VIRTIO Transport layer.

12.65.1 Detailed Description

12.66 L4 VIRTIO Transport Layer

L4 specific VIRTIO Transport layer.

Collaboration diagram for L4 VIRTIO Transport Layer:



Namespaces

- [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](#), [L4VIRTIO_OP_REGISTER_IFACE](#), [L4VIRTIO_OP_REGISTER_DS](#) }
L4-VIRTIO opcodes.
- 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_RPMSG](#) = 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_CRYPT](#) = 20, [L4VIRTIO_ID_SOCK](#) = 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_FAILED` = 0x80 }
Virtio device status bits.
- enum `L4virtio_feature_bits` { `L4VIRTIO_FEATURE_VERSION_1` = 32, `L4VIRTIO_FEATURE_CMD_CONFIG` = 33 }
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_MASK` = 0xff000000 }
Virtio commands for device configuration.

Functions

- `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`
Write the VIRTIO status register.
- `int l4virtio_config_queue (l4_cap_idx_t cap, unsigned queue) L4_NOTHROW`
Trigger queue configuration of the given queue.
- `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`
Register a shared data space with VIRTIO host.
- `int l4virtio_register_iface (l4_cap_idx_t cap, l4_cap_idx_t guest_irq, l4_cap_idx_t host_irq, l4_cap_idx_t config_ds) L4_NOTHROW`
Register client to the given L4-VIRTIO host.

12.66.1 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.

12.66.2 Typedef Documentation

12.66.2.1 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.

12.66.3 Enumeration Type Documentation

12.66.3.1 L4_virtio_cmd

```
enum L4_virtio_cmd
```

Virtio commands for device configuration.

Enumerator

<code>L4VIRTIO_CMD_NONE</code>	No command pending.
<code>L4VIRTIO_CMD_SET_STATUS</code>	Set the status register.
<code>L4VIRTIO_CMD_CFG_QUEUE</code>	Configure a queue.
<code>L4VIRTIO_CMD_MASK</code>	Mask to get command bits.

Definition at line 116 of file `virtio.h`.

12.66.3.2 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 107 of file [virtio.h](#).

12.66.3.3 L4_virtio_opcodes

enum [L4_virtio_opcodes](#)

L4-VIRTIO opcodes.

Enumerator

L4VIRTIO_OP_SET_STATUS	Set status register in device config.
L4VIRTIO_OP_CONFIG_QUEUE	Set queue config in device config.
L4VIRTIO_OP_REGISTER_IFACE	Register a transport driver to the device.
L4VIRTIO_OP_REGISTER_DS	Register a data space as transport memory.

Definition at line 55 of file [virtio.h](#).

12.66.3.4 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_RPMSG	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_SOCKET	Inofficial socket device.

Definition at line 64 of file [virtio.h](#).

12.66.3.5 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_FAILED	Fatal error in driver or device.

Definition at line 85 of file [virtio.h](#).

12.66.3.6 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 95 of file [virtio.h](#).

12.66.4 Function Documentation

12.66.4.1 l4virtio_config_queue()

```
int l4virtio_config_queue (
    l4_cap_idx_t cap,
    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

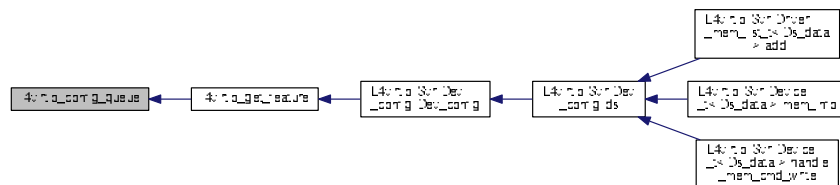
<i>cap</i>	Capability to the VIRTIO host.
<i>queue</i>	Queue index for the queue to be configured.

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.

Referenced by [l4virtio_get_feature\(\)](#).

Here is the caller graph for this function:



12.66.4.2 l4virtio_config_queues()

```
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 237 of file virtio.h.

12.66.4.3 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 248 of file [virtio.h](#).

12.66.4.4 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 )
```

Register a shared data space with VIRTIO host.

Parameters

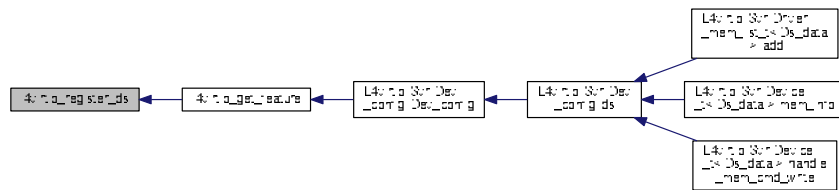
<i>cap</i>	Capability to the VIRTIO host
<i>ds_cap</i>	Data-space capability to register. The lower 8 bits determine the rights mask with which the guest's rights are masked during the registration of the dataspace at the VIRTIO host.
<i>base</i>	VIRTIO guest physical start address of shared memory region
<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

Returns

0 on success, < 0 on error

Referenced by [l4virtio_get_feature\(\)](#).

Here is the caller graph for this function:



12.66.4.5 l4virtio_register_iface()

```

int l4virtio_register_iface (
    l4_cap_idx_t cap,
    l4_cap_idx_t guest_irq,
    l4_cap_idx_t host_irq,
    l4_cap_idx_t config_ds )

```

Register client to the given L4-VIRTIO host.

Parameters

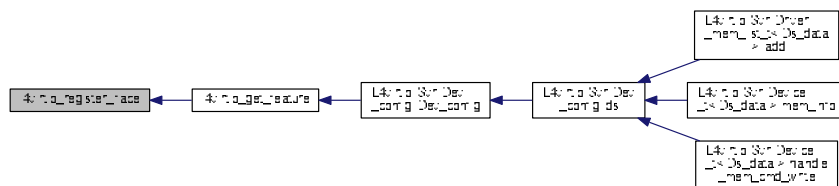
<i>cap</i>	Capability to the L4-VIRTIO host
<i>guest_irq</i>	IRQ capability for valid IRQ object for host-to-guest notifications
<i>host_irq</i>	Capability selector for receiving the guest-to-host notifications IRQ capability.
<i>config_ds</i>	Capability for receiving the data-space capability for the shared L4-VIRTIO config data space.

Return values

0	on success.
-L4_EINVAL	The host did not receive the <i>guest_irq</i> cap.

Referenced by [l4virtio_get_feature\(\)](#).

Here is the caller graph for this function:



12.66.4.6 l4virtio_set_status()

```
int l4virtio_set_status (
    l4_cap_idx_t cap,
    unsigned status )
```

Write the VIRTIO status register.

Note

All other registers are accessed via shared memory.

Parameters

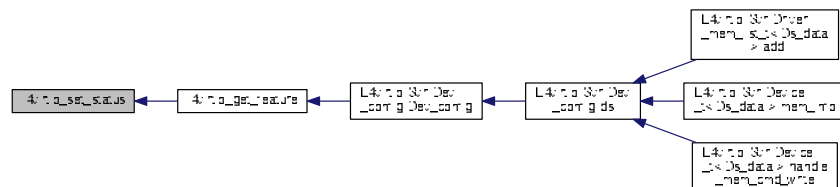
<i>cap</i>	Capability to the VIRTIO host
<i>status</i>	Status word to write to the VIRTIO status.

Returns

0 on success, <0 on error.

Referenced by [l4virtio_get_feature\(\)](#).

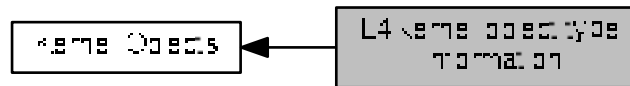
Here is the caller graph for this function:



12.67 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](#) ()
Get the [L4::Type_info](#) for the [L4Re](#) interface given in T.

12.67.1 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.

12.67.2 Function Documentation

12.67.2.1 kobject_typeid()

```
template<typename T >
Type\_info const* L4::kobject\_typeid ( ) [inline]
```

Get the [L4::Type_info](#) for the [L4Re](#) interface given in T.

Template Parameters

<i>T</i>	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 686 of file [__typeinfo.h](#).

References [L4::Kobject_typeid< T >::id\(\)](#), and [L4::PROTO_ANY](#).

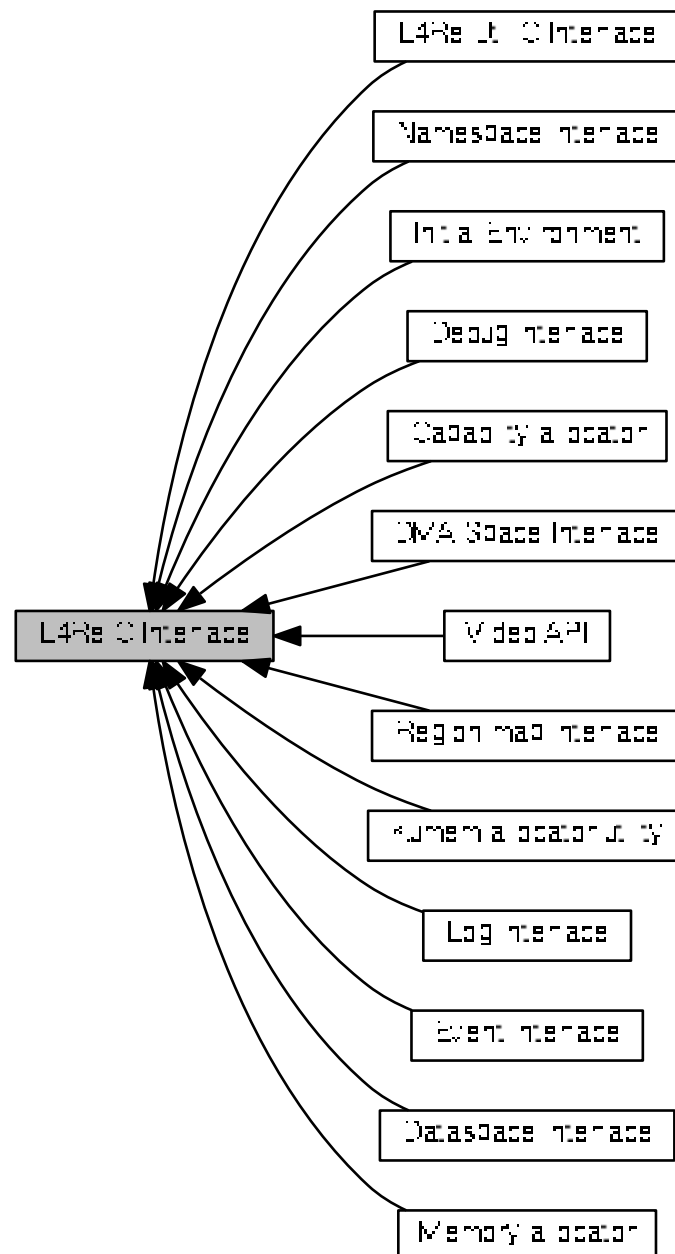
Here is the call graph for this function:



12.68 L4Re C Interface

Documentation for the [L4Re C Interface](#).

Collaboration diagram for L4Re C Interface:



Modules

- [Capability allocator](#)

- [Capability allocator C interface.](#)
- [DMA Space Interface](#)
DMA Space C interface.
- [Dataspace interface](#)
Dataspace C interface.
- [Debug interface](#)
- [Event interface](#)
Event C interface.
- [Initial Environment](#)
C interface of the initial environment that is provided to an [L4](#) task.
- [Kumem allocator utility](#)
Kumem allocator utility C interface.
- [L4Re Util C Interface](#)
Documentation of the [L4](#) Runtime Environment utility functionality in C.
- [Log interface](#)
Log C interface.
- [Memory allocator](#)
Memory allocator C interface.
- [Namespace interface](#)
Namespace C interface.
- [Region map interface](#)
Region map C interface.
- [Video API](#)

Functions

- `long l4re_inhibitor_acquire (l4_cap_idx_t cap, l4_umword_t id, char const *reason)`
Inhibitor C interface.

12.68.1 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.

12.68.2 Function Documentation

12.68.2.1 l4re_inhibitor_acquire()

```
long l4re_inhibitor_acquire (
    l4_cap_idx_t cap,
    l4_umword_t id,
    char const * reason )
```

Inhibitor C interface.

Acquire an inhibitor lock.

Parameters

<i>cap</i>	Capability for the Inhibitor object (
------------	---------------------------------------

See also

[L4Re::Inhibitor](#))

Parameters

<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

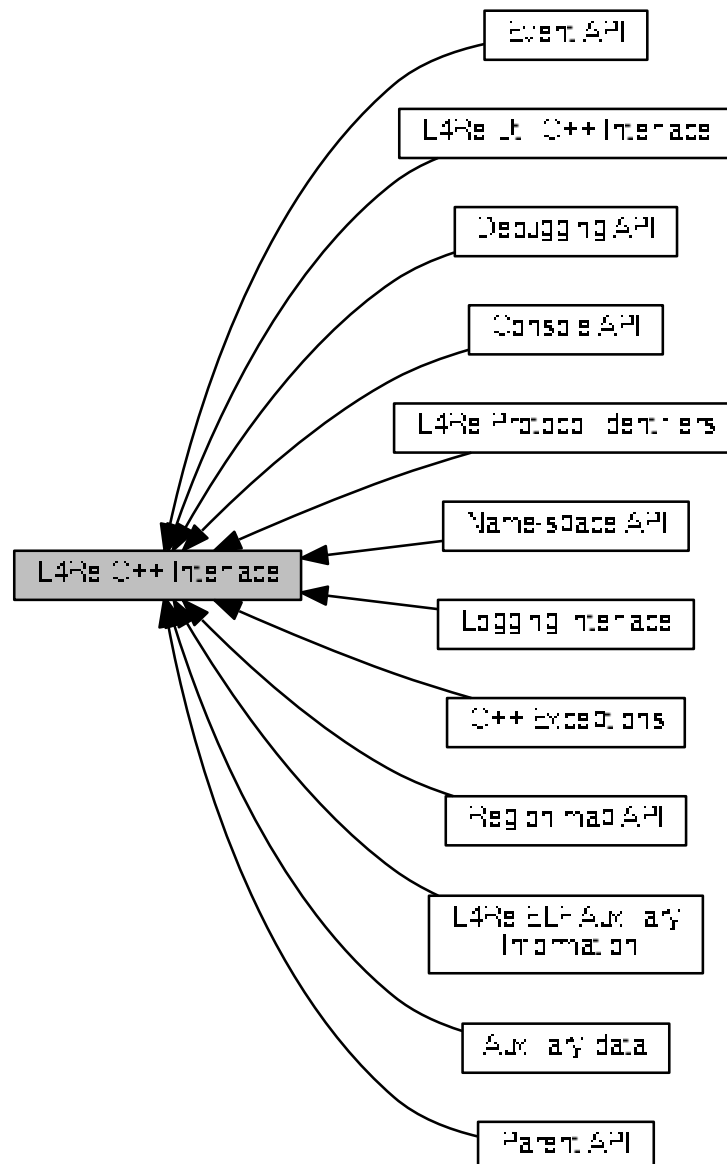
See also

[L4Re::Inhibitor::acquire\(\)](#)

12.69 L4Re C++ Interface

Documentation of the [L4](#) Runtime Environment C++ API.

Collaboration diagram for L4Re C++ Interface:



Modules

- [Auxiliary data](#)
- [C++ Exceptions](#)
- [Console API](#)

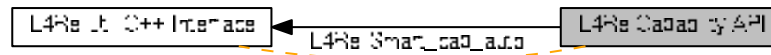
- [Console interface.](#)
- [Debugging API](#)
Debugging Interface.
- [Event API](#)
Event interface.
- [L4Re ELF Auxiliary Information](#)
API for embedding auxiliary information into binary programs.
- [L4Re Protocol identifiers](#)
L4Re Protocol Constants (C version)
- [L4Re Util C++ Interface](#)
Documentation of the [L4 Runtime Environment](#) utility functionality in C++.
- [Logging interface](#)
Interface for log output.
- [Name-space API](#)
API for name spaces that store capabilities.
- [Parent API](#)
Parent interface.
- [Region map API](#)
Virtual address-space management.

12.69.1 Detailed Description

Documentation of the [L4 Runtime Environment](#) C++ API.

12.70 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 [Auto_cap](#) and [Auto_del_cap](#).
- class [L4Re::Util::Smart_cap_auto< Unmap_flags >](#)
Helper for [Auto_cap](#) and [Auto_del_cap](#).
- class [L4Re::Util::Smart_count_cap< Unmap_flags >](#)
Helper for [Ref_cap](#) and [Ref_del_cap](#).
- struct [L4Re::Util::Auto_cap< T >](#)
Automatic capability that implements automatic free and unmap of the capability selector.
- struct [L4Re::Util::Auto_del_cap< T >](#)
Automatic capability that implements automatic free and unmap+delete of the capability selector.
- 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 >
[Auto_cap< T >::Cap L4Re::Util::make_auto_cap \(\)](#)
Allocate a capability slot and wrap it in an [Auto_cap](#).
- template<typename T >
[Auto_del_cap< T >::Cap L4Re::Util::make_auto_del_cap \(\)](#)
Allocate a capability slot and wrap it in an [Auto_del_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<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](#).
- virtual [L4Re::Cap_alloc::~Cap_alloc \(\)=0](#)
Destructor.

Variables

- [_Cap_alloc](#) & [L4Re::Util::cap_alloc](#)
Capability allocator.

12.70.1 Detailed Description

12.70.2 Function Documentation

12.70.2.1 `make_auto_cap()`

```
template<typename T >
Auto\_cap<T>::Cap L4Re::Util::make_auto_cap ( )
```

Allocate a capability slot and wrap it in an [Auto_cap](#).

Template Parameters

<i>T</i>	Type of capability the slot is used for.
----------	--

Definition at line 284 of file [cap_alloc](#).

12.70.2.2 `make_auto_del_cap()`

```
template<typename T >
Auto\_del\_cap<T>::Cap L4Re::Util::make_auto_del_cap ( )
```

Allocate a capability slot and wrap it in an [Auto_del_cap](#).

Template Parameters

<i>T</i>	Type of capability the slot is used for.
----------	--

Definition at line 294 of file [cap_alloc](#).

12.70.2.3 `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 304 of file [cap_alloc](#).

12.70.2.4 `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 313 of file [cap_alloc](#).

12.70.3 Variable Documentation

12.70.3.1 `cap_alloc`

```
_Cap_alloc& L4Re::Util::cap_alloc
```

Capability allocator.

This is the instance of the capability allocator that is used by usual applications. The actual implementation of the allocator depends on the configuration of the system.

Per default we use [Counting_cap_alloc](#), a reference-counting capability allocator, that keeps a reference counter for each managed capability selector.

Note

This capability allocator is not thread-safe.

Examples:

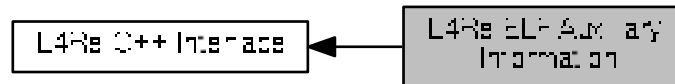
[examples/libs/l4re/c++/mem_alloc/ma+rm.cc](#), [examples/libs/l4re/c++/shared_ds/ds_clnt.cc](#), [examples/libs/l4re/c++/shared_ds/ds_srv.cc](#), and [examples/libs/l4re/streammap/client.cc](#).

Referenced by [L4virtio::Svr::Block_dev< Ds_data >::Block_dev\(\)](#), [L4virtio::Svr::Dev_config::Dev_config\(\)](#), [L4Re::Vfs::Fs::mount\(\)](#), and [L4Re::Util::Object_registry::unregister_obj\(\)](#).

12.71 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(".rol4re_elf_aux"), aligned(sizeof(l4re_elf_aux_mword_t))))`
Define an auxiliary vector element.
- `#define L4RE_ELF_AUX_ELEM_T(type, id, tag, val...) static L4RE_ELF_AUX_ELEM type id = {tag, sizeof(type), 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 }`

12.71.1 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.

12.71.2 Macro Definition Documentation

12.71.2.1 L4RE_ELF_AUX_ELEM

```
#define L4RE_ELF_AUX_ELEM const __attribute__((used, section(".ro14re_elf_aux"), aligned(sizeof(l4re_elf_aux_vma_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 52 of file `elf_aux.h`.

12.71.2.2 L4RE_ELF_AUX_ELEM_T

```
#define L4RE_ELF_AUX_ELEM_T(
    type,
    id,
    tag,
    val... ) 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., <code>l4re_elf_aux_vma_t</code>)
<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., <code>L4RE_ELF_AUX_T_VMA</code>
<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 67 of file [elf_aux.h](#).

12.71.3 Enumeration Type Documentation

12.71.3.1 anonymous enum

anonymous enum

Enumerator

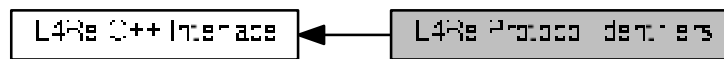
L4RE_ELF_AUX_T_NONE	Tag for an invalid element in the auxiliary vector.
L4RE_ELF_AUX_T_VMA	Tag for descriptor for a reserved virtual memory area.
L4RE_ELF_AUX_T_STACK_SIZE	Tag for descriptor that defines the stack size for the first application thread.
L4RE_ELF_AUX_T_STACK_ADDR	Tag for descriptor that 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 binaries address space.

Definition at line 70 of file [elf_aux.h](#).

12.72 L4Re Protocol identifiers

[L4Re](#) Protocol Constants (C version)

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::Rm_::Opcodes](#)
Region-map communication-protocol opcodes.
- enum [L4Re::Video::Goos_::Opcodes](#)
Frame buffer communication-protocol opcodes.

12.72.1 Detailed Description

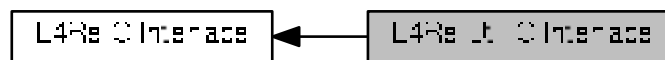
[L4Re](#) Protocol Constants (C version)

Fix [L4Re](#) Protocol Constants.

12.73 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.

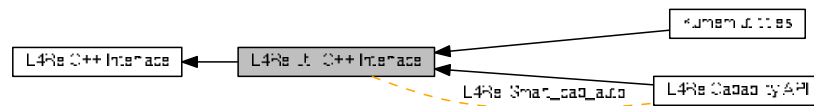
The interface functions closely align with the C++ functions and add no further functionalities.

For new programs it is advised to use the C++ interface.

12.74 L4Re Util C++ Interface

Documentation of the [L4](#) Runtime Environment utility functionality in C++.

Collaboration diagram for L4Re Util C++ Interface:



Modules

- [Kumem utilities](#)
- [L4Re Capability API](#)

Data Structures

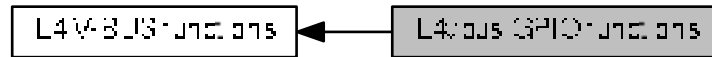
- class [L4Re::Smart_cap_auto< Unmap_flags >](#)
Helper for Auto_cap and Auto_del_cap.
- class [L4Re::Util::Cap_alloc_base](#)
Capability allocator.
- class [L4Re::Util::Br_manager](#)
Buffer-register (BR) manager for [L4::Server](#).
- class [L4Re::Util::Counting_cap_alloc< COUNTERTYPE >](#)
Reference-counting 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.

12.74.1 Detailed Description

Documentation of the [L4](#) Runtime Environment utility functionality in C++.

12.75 L4vbus GPIO functions

Collaboration diagram for L4vbus GPIO functions:



Data Structures

- class `L4vbus::Gpio_pin`
A GPIO pin.
- class `L4vbus::Gpio_module`
A *Gpio_module* groups multiple GPIO pins together.

Enumerations

- enum `L4vbus_gpio_generic_func` { `L4VBUS_GPIO_SETUP_INPUT` = 0x100, `L4VBUS_GPIO_SETUP_OUTPUT` = 0x200, `L4VBUS_GPIO_SETUP_IRQ` = 0x300 }
 - 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 functions.
- 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.

12.75.1 Detailed Description

12.75.2 Enumeration Type Documentation

12.75.2.1 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 26 of file [vbus_gpio.h](#).

12.75.2.2 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 36 of file [vbus_gpio.h](#).

12.75.3 Function Documentation

12.75.3.1 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

Referenced by [L4vbus::Gpio_pin::config_get\(\)](#).

Here is the caller graph for this function:



12.75.3.2 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

Referenced by [L4vbus::Gpio_pin::config_pad\(\)](#).

Here is the caller graph for this function:



12.75.3.3 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

Referenced by [L4vbus::Gpio_pin::config_pull\(\)](#).

Here is the caller graph for this function:

**12.75.3.4 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.

Referenced by [L4vbus::Gpio_pin::get\(\)](#).

Here is the caller graph for this function:



12.75.3.5 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

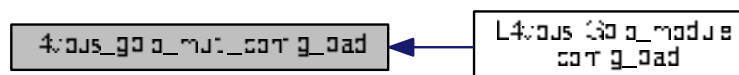
<i>vbus</i>	V-BUS capability
<i>handle</i>	Device handle for the GPIO chip
<i>offset</i>	Pin corresponding to the LSB in <i>mask</i> . Note: allowed may be hardware specific.
<i>mask</i>	Mask of GPIO pins to configure. A bit set to 1 configures this pin. A maximum of 32 pins can be configured at once. The real number depends on the hardware and the driver implementation.
<i>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

Referenced by [L4vbus::Gpio_module::config_pad\(\)](#).

Here is the caller graph for this function:



12.75.3.6 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

Referenced by [L4vbus::Gpio_module::get\(\)](#).

Here is the caller graph for this function:



12.75.3.7 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

Referenced by [L4vbus::Gpio_module::set\(\)](#).

Here is the caller graph for this function:

**12.75.3.8 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

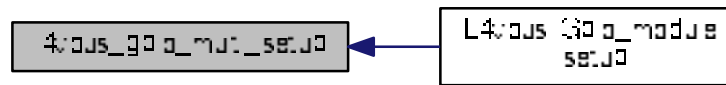
<i>vbus</i>	V-BUS capability
<i>handle</i>	Device handle for the GPIO chip
<i>offset</i>	Pin corresponding to the LSB in <i>mask</i> . Note: allowed may be hardware specific.
<i>mask</i>	Mask of GPIO pins to configure. A bit set to 1 configures this pin. A maximum of 32 pins can be configured at once. The real number depends on the hardware and the driver implementation.
<i>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

Referenced by [L4vbus::Gpio_module::setup\(\)](#).

Here is the caller graph for this function:



12.75.3.9 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

Referenced by [L4vbus::Gpio_pin::set\(\)](#).

Here is the caller graph for this function:



12.75.3.10 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

Referenced by [L4vbus::Gpio_pin::setup\(\)](#).

Here is the caller graph for this function:



12.75.3.11 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

Referenced by [L4vbus::Gpio_pin::to_irq\(\)](#).

Here is the caller graph for this function:



12.76 L4vbus PCI functions

Collaboration diagram for L4vbus PCI functions:



Functions

- `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_pcidv_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_pcidv_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_pcidv_irq_enable (l4_cap_idx_t vbus, l4vbus_device_handle_t handle, unsigned char *trigger, unsigned char *polarity)`
Enable the device's PCI interrupt.

12.76.1 Detailed Description

12.76.2 Function Documentation

12.76.2.1 l4vbus_pci_cfg_read()

```

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

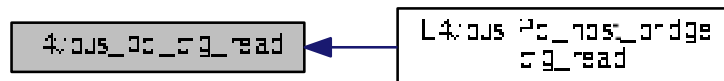
	<i>vbus</i>	Capability of the system bus
	<i>handle</i>	Device handle of the PCI root bridge
	<i>bus</i>	Bus number
	<i>devfn</i>	Device id (upper 16bit) and function (lower 16bit)
	<i>reg</i>	Register in configuration space to read
out	<i>value</i>	Value that has been read
	<i>width</i>	Width to read in bits (e.g. 8, 16, 32)

Returns

0 on success, else failure

Referenced by [L4vbus::Pci_host_bridge::cfg_read\(\)](#).

Here is the caller graph for this function:



12.76.2.2 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

Referenced by [L4vbus::Pci_host_bridge::cfg_write\(\)](#).

Here is the caller graph for this function:

**12.76.2.3 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

Referenced by [L4vbus::Pci_host_bridge::irq_enable\(\)](#).

Here is the caller graph for this function:



12.76.2.4 l4vbus_pci_dev_cfg_read()

```

int l4vbus_pci_dev_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

Referenced by [L4vbus::Pci_dev::cfg_read\(\)](#).

Here is the caller graph for this function:



12.76.2.5 l4vbus_pcidev_cfg_write()

```
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.

Parameters

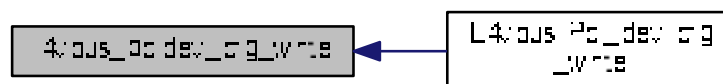
<i>vbus</i>	Capability of the system bus
<i>handle</i>	Device handle of the PCI device
<i>reg</i>	Register in configuration space to write
<i>value</i>	Value to write
<i>width</i>	Width to write in bits (e.g. 8, 16, 32)

Returns

0 on success, else failure

Referenced by [L4vbus::Pci_dev::cfg_write\(\)](#).

Here is the caller graph for this function:



12.76.2.6 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

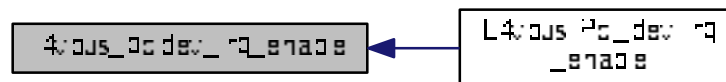
	<i>vbus</i>	Capability of the system bus
	<i>handle</i>	Device handle of the PCI device
out	<i>trigger</i>	False if interrupt is level-triggered
out	<i>polarity</i>	True if interrupt is of low polarity

Returns

On success: Interrupt line to be used, else failure

Referenced by [L4vbus::Pci_dev::irq_enable\(\)](#).

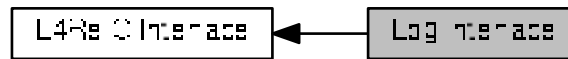
Here is the caller graph for this function:



12.77 Log interface

Log C interface.

Collaboration diagram for Log interface:



Functions

- 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.

12.77.1 Detailed Description

Log C interface.

12.77.2 Function Documentation

12.77.2.1 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.
---------------	---------------------------------

Returns

0 for success, <0 on error

See also

[L4Re::Log::print](#)

Definition at line 99 of file [log.h](#).

References [l4re_log_print_srv\(\)](#).

Here is the call graph for this function:

**12.77.2.2 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.

Returns

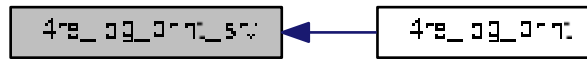
0 for success, <0 on error

See also

[L4Re::Log::print](#)

Referenced by [l4re_log_print\(\)](#).

Here is the caller graph for this function:



12.77.2.3 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.

Returns

0 for success, <0 on error

See also

[L4Re::Log::println](#)

Definition at line 105 of file [log.h](#).

References [l4re_log_printn_srv\(\)](#).

Here is the call graph for this function:



12.77.2.4 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.

Returns

0 for success, <0 on error

See also

[L4Re::Log::println](#)

Referenced by [l4re_log_printn\(\)](#).

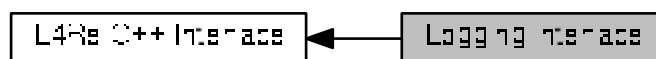
Here is the caller graph for this function:



12.78 Logging interface

Interface for log output.

Collaboration diagram for Logging interface:



Data Structures

- class [L4Re::Log](#)
Log interface class.

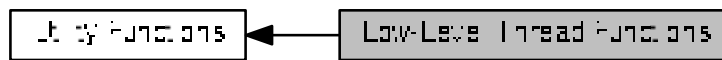
12.78.1 Detailed Description

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.

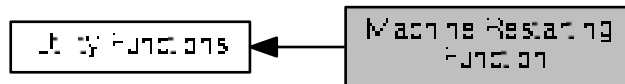
12.79 Low-Level Thread Functions

Collaboration diagram for Low-Level Thread Functions:



12.80 Machine Restarting Function

Collaboration diagram for Machine Restarting Function:



Functions

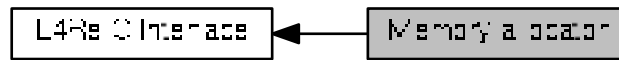
- void [l4util_reboot](#) (void)
Machine reboot.

12.80.1 Detailed Description

12.81 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

- long [l4re_ma_alloc](#) (long size, [l4re_ds_t](#) const mem, unsigned long flags) [L4_NOTHROW](#)
Allocate memory.
- long [l4re_ma_alloc_align](#) (long size, [l4re_ds_t](#) const mem, unsigned long flags, unsigned long align) [L4_NOTHROW](#)
Allocate memory.
- long [l4re_ma_free](#) ([l4re_ds_t](#) const mem) [L4_NOTHROW](#))
Free memory.
- long [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.
- long [l4re_ma_free_srv](#) ([l4_cap_idx_t](#) srv, [l4re_ds_t](#) const mem) [L4_NOTHROW](#))
Free memory.

12.81.1 Detailed Description

Memory allocator C interface.

12.81.2 Enumeration Type Documentation

12.81.2.1 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 42 of file [mem_alloc.h](#).

12.81.3 Function Documentation

12.81.3.1 l4re_ma_alloc()

```
long l4re_ma_alloc (
    long size,
    l4re_ds_t const mem,
    unsigned long flags ) [inline]
```

Allocate memory.

Parameters

<i>size</i>	Size to be requested in bytes (granularity is (super)pages and the size is rounded up to this granularity).
<i>mem</i>	Capability slot to put the requested dataspace in
<i>flags</i>	Flags, see l4re_ma_flags

Returns

0 on success, <0 on 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 167 of file [mem_alloc.h](#).

References [l4re_ma_alloc_align_srv\(\)](#).

Here is the call graph for this function:



12.81.3.2 l4re_ma_alloc_align()

```

long 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 to be requested in bytes (granularity is (super)pages and the size is rounded up to this granularity).
<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](#) 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 175 of file [mem_alloc.h](#).

References [l4re_ma_alloc_align_srv\(\)](#).

Here is the call graph for this function:

**12.81.3.3 l4re_ma_alloc_align_srv()**

```

long 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

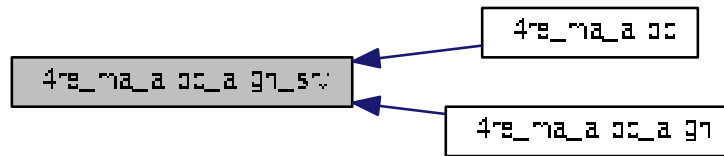
See also

[L4Re::Mem_alloc::alloc](#)

The memory allocator returns a dataspace.

Referenced by [l4re_ma_alloc\(\)](#), and [l4re_ma_alloc_align\(\)](#).

Here is the caller graph for this function:



12.81.3.4 l4re_ma_free()

```
long l4re_ma_free (
    l4re_ds_t const mem ) [inline]
```

Free memory.

Parameters

<i>mem</i>	Dataspace to free.
------------	--------------------

Returns

0 on success, <0 on error

See also

[L4Re::Mem_alloc::free](#)

Note

This function is using the [L4Re::Env::env\(\)](#)->[mem_alloc\(\)](#) service.

Deprecated This function is deprecated. Use [l4_task_unmap\(\)](#) or similar means to remove a dataspace.

Definition at line 183 of file [mem_alloc.h](#).

References [l4re_ma_free_srv\(\)](#).

Here is the call graph for this function:



12.81.3.5 l4re_ma_free_srv()

```

long l4re_ma_free_srv (
    l4_cap_idx_t srv,
    l4re_ds_t const mem )
  
```

Free memory.

Parameters

<i>srv</i>	Memory allocator service.
<i>mem</i>	Dataspace to free.

Returns

0 on success, <0 on error

See also

[L4Re::Mem_alloc::free](#)

Deprecated This function is deprecated. Use [l4_task_unmap\(\)](#) or similar means to remove a dataspace.

Referenced by [l4re_ma_free\(\)](#).

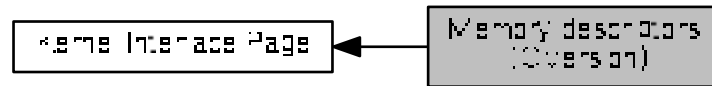
Here is the caller graph for this function:



12.82 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 }
Memory sub types for `l4_mem_type_info` 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.

12.82.1 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).

12.82.2 Typedef Documentation

12.82.2.1 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.

12.82.3 Enumeration Type Documentation

12.82.3.1 l4_mem_info_sub_type_t

```
enum l4_mem_info_sub_type_t
```

Memory sub types for l4_mem_type_info descriptors.

Enumerator

<code>l4_mem_info_acpi_rsdp</code>	Physical address of the ACPI root pointer.
------------------------------------	--

Definition at line 61 of file [memdesc.h](#).

12.82.3.2 l4_mem_type_t

enum [l4_mem_type_t](#)

Type of a memory descriptor.

Enumerator

l4_mem_type_undefined	Undefined, unused descriptor.
l4_mem_type_conventional	Conventional memory.
l4_mem_type_reserved	Reserved memory for kernel etc.
l4_mem_type_dedicated	Dedicated memory (some device memory)
l4_mem_type_shared	Shared memory (not implemented)
l4_mem_type_info	Info from the boot loader.
l4_mem_type_bootloader	Memory owned by the boot loader.
l4_mem_type_archspecific	Architecture specific memory (e.g., ACPI memory)

Definition at line 44 of file [memdesc.h](#).

12.82.4 Function Documentation

12.82.4.1 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 217 of file [memdesc.h](#).

12.82.4.2 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 238 of file [memdesc.h](#).

12.82.4.3 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 210 of file [memdesc.h](#).

12.82.4.4 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 231 of file [memdesc.h](#).

12.82.4.5 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 224 of file [memdesc.h](#).

12.82.4.6 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 188 of file [memdesc.h](#).

12.82.4.7 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 195 of file [memdesc.h](#).

12.83 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 memory from kernel privilege level.
- void `l4_mem_write` (unsigned long virtaddress, unsigned width, unsigned long value)
Write memory from kernel privilege level.

12.83.1 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>
```

12.83.2 Enumeration Type Documentation

12.83.2.1 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 51 of file [mem_op.h](#).

12.83.3 Function Documentation

12.83.3.1 l4_mem_read()

```
unsigned long l4_mem_read (  
    unsigned long virtaddress,  
    unsigned width ) [inline]
```

Read 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 an given invalid address or invalid width value the function does nothing.

Definition at line 141 of file [mem_op.h](#).

References [l4_mem_arm_op_call\(\)](#).

Here is the call graph for this function:



12.83.3.2 l4_mem_write()

```
void l4_mem_write (
    unsigned long virtaddress,
    unsigned width,
    unsigned long value ) [inline]
```

Write memory from kernel privilege level.

Parameters

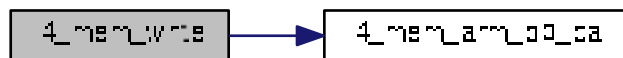
<i>virtaddress</i>	Virtual address in the calling task.
<i>width</i>	Width of access in bytes in log2 (i.e. allowed values: 0, 1, 2)
<i>value</i>	Value to write.

Upon an given invalid address or invalid width value the function does nothing.

Definition at line [147](#) of file [mem_op.h](#).

References [l4_mem_arm_op_call\(\)](#).

Here is the call graph for this function:



12.84 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 22`
Size of a large page log2-based.

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 flex page 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 \(l4_umword_t value, unsigned char bits\) L4_NOTHROW](#)
Round value up to the next alignment with bits size.

12.84.1 Detailed Description

Memory related constants, data types and functions.

12.84.2 Macro Definition Documentation

12.84.2.1 L4_LOG2_PAGESIZE

```
#define L4_LOG2_PAGESIZE
```

Number of bits used for page offset.

Size of page in log2.

Definition at line 295 of file [consts.h](#).

Referenced by [L4Re::Rm::attach\(\)](#), [L4Re::Dataspace::map\(\)](#), [L4Re::Dataspace::map_region\(\)](#), and [L4Re::Util::↵
Dataspace_srv::page_shift\(\)](#).

12.84.2.2 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 321 of file [consts.h](#).

12.84.2.3 L4_PAGEMASK

```
#define L4_PAGEMASK
```

Mask for the page number.

Note

The most significant bits are set.

Definition at line 286 of file [consts.h](#).

Referenced by [l4_round_page\(\)](#), [l4_sleep_forever\(\)](#), and [l4_trunc_page\(\)](#).

12.84.2.4 L4_SUPERPAGEMASK

```
#define L4_SUPERPAGEMASK
```

Mask for the number of a large page.

Note

The most significant bits are set.

Definition at line 313 of file [consts.h](#).

12.84.2.5 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 304 of file [consts.h](#).

12.84.3 Enumeration Type Documentation

12.84.3.1 l4_addr_consts_t

```
enum l4_addr_consts_t
```

Address related constants.

Enumerator

L4_INVALID_ADDR	Invalid address.
-----------------	------------------

Definition at line 377 of file [consts.h](#).

12.84.4 Function Documentation

12.84.4.1 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 359 of file [consts.h](#).

References [L4_NOTHROW](#), [L4_PAGEMASK](#), [L4_PAGESIZE](#), and [l4_round_size\(\)](#).

Referenced by [L4Re::Rm::attach\(\)](#), [l4_trunc_size\(\)](#), [L4Re::Dataspace::map\(\)](#), and [L4Re::Dataspace::map_region\(\)](#).

Here is the call graph for this function:



[illegible]

```
l4_addr_t l4_round_size (
    l4_umword_t value,
    unsigned char bits ) [inline]
```

Parameters

<i>value</i>	The value to round up to the next size-alignment.
<i>bits</i>	The size of the alignment (log2).

Referenced by `cxx::List_alloc::alloc_max()`, `L4Re::Util::Dataspace_svr::is_static()`, `l4_round_page()`, `L4Re::Dataspace::map_region()`, and `L4virtio::Virtqueue::setup_simple()`.

[illegible]

12.84.4.3 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 334 of file [consts.h](#).

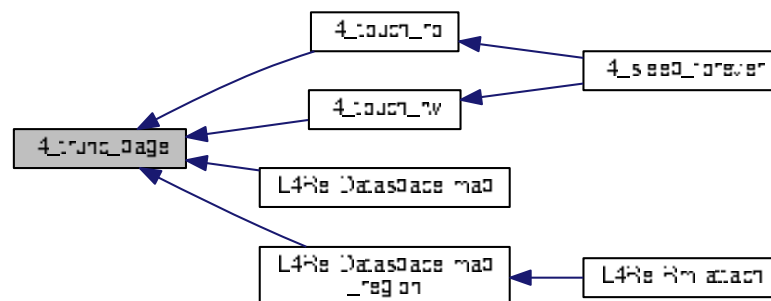
References [L4_NOTHROW](#), [L4_PAGEMASK](#), and [l4_trunc_size\(\)](#).

Referenced by [l4_touch_ro\(\)](#), [l4_touch_rw\(\)](#), [L4Re::Dataspace::map\(\)](#), and [L4Re::Dataspace::map_region\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



12.84.4.4 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 flex page with size *bits*.

Parameters

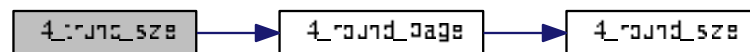
<i>address</i>	The address to round.
<i>bits</i>	The size of the flex page (log2).

Definition at line 345 of file consts.h.

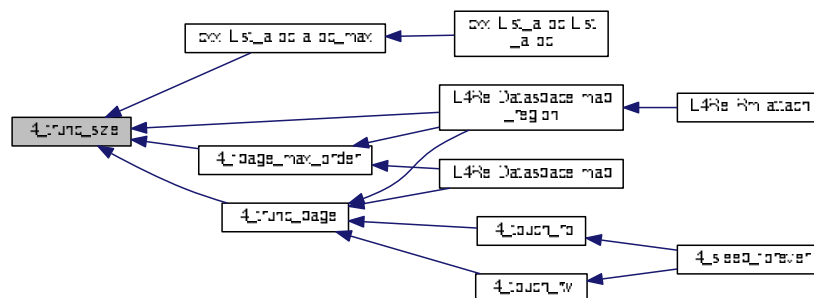
References [L4_NOTHROW](#), and [l4_round_page\(\)](#).

Referenced by `cxx::List_alloc::alloc_max()`, `l4_fpage_max_order()`, `l4_trunc_page()`, and `L4Re::Dataspace::map←_region()`.

Here is the call graph for this function:



Here is the caller graph for this function:



12.85 Message Items

Message item related functions.

Collaboration diagram for Message Items:



Enumerations

- 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_SINGLE_CAP = L4_ITEM_MAP | 2, L4_RCV_ITEM_LOCAL_ID = 4 }`

Constants for message items.

Functions

- `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 for the memory space.
- `l4_umword_t l4_map_obj_control (l4_umword_t spot, unsigned grant) L4_NOTHROW`
Create the first word for a map item for the object space.

12.85.1 Detailed Description

Message item related functions.

Message items are typed items that can be transferred via IPC operations. Message items are also used to specify receive windows for typed items to be received. Message items are placed in the message registers (MRs) of the UTCB of the sending thread. Receive items are placed in the buffer registers (BRs) of the UTCB of the receiving thread.

Message items are usually two-word data structures. The first word denotes the type of the message item (for example a memory flex-page, io flex-page or object flex-page) and the second word contains information depending on the type. There is actually one exception that is a small (one word) receive buffer item for a single capability.

12.85.2 Enumeration Type Documentation

12.85.2.1 `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	Flag as <i>grant</i> instead of <i>map</i> operation.
L4_MAP_ITEM_MAP	Flag as usual <i>map</i> operation.
L4_RCV_ITEM_SINGLE_CAP	Mark the receive buffer to be a small receive item that describes a buffer for a single capability.
L4_RCV_ITEM_LOCAL_ID	The receiver requests to receive a local ID instead of a mapping whenever possible.

Definition at line 187 of file [consts.h](#).

12.85.3 Function Documentation

12.85.3.1 l4_map_control()

```
l4_umword_t l4_map_control (
    l4_umword_t spot,
    unsigned char cache,
    unsigned grant ) [inline]
```

Create the first word for a map item for the memory space.

Parameters

<i>spot</i>	Hot spot address, used to determine what is actually mapped when send and receive flex page have differing sizes.
<i>cache</i>	Cacheability hints for memory flex pages. See Cacheability options
<i>grant</i>	Indicates if it is a map or a grant item.

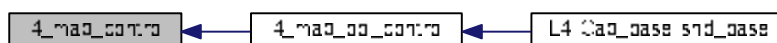
Returns

The value to be used as first word in a map item for memory.

Definition at line 672 of file [__l4_fpage.h](#).

Referenced by [l4_map_obj_control\(\)](#).

Here is the caller graph for this function:



12.85.3.2 l4_map_obj_control()

```
l4_umword_t l4_map_obj_control (
    l4_umword_t spot,
    unsigned grant ) [inline]
```

Create the first word for a map item for the object space.

Parameters

<i>spot</i>	Hot spot address, used to determine what is actually mapped when send and receive flex pages have different size.
<i>grant</i>	Indicates if it is a map item or a grant item.

Returns

The value to be used as first word in a map item for kernel objects or IO-ports.

Definition at line 679 of file [__l4_fpage.h](#).

References [l4_map_control\(\)](#).

Referenced by [L4::Cap_base::snd_base\(\)](#).

Here is the call graph for this function:

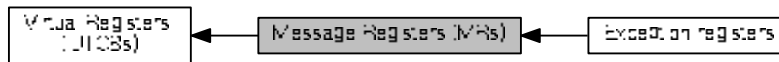


Here is the caller graph for this function:



12.86 Message Registers (MRs)

Collaboration diagram for Message Registers (MRs):



Modules

- [Exception registers](#)
Overly definition of the MRs for exception messages.

Data Structures

- [union l4_msg_regs_t](#)
Encapsulation of the message-register block in the UTCB.

Typedefs

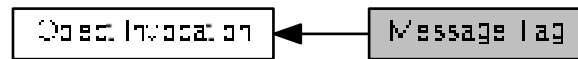
- `typedef union l4_msg_regs_t l4_msg_regs_t`
Encapsulation of the message-register block in the UTCB.

12.86.1 Detailed Description

12.87 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.

Typedefs

- typedef struct [l4_msgtag_t](#) [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_TO_IRQ](#) = -1L,
[L4_PROTO_PAGE_FAULT](#) = -2L, [L4_PROTO_PREEMPTION](#) = -3L, [L4_PROTO_SYS_EXCEPTION](#) = -4L,
[L4_PROTO_EXCEPTION](#) = -5L,
[L4_PROTO_SIGMA0](#) = -6L, [L4_PROTO_IO_PAGE_FAULT](#) = -8L, [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_TO_IRQ_MUX](#) = -19L,
[L4_PROTO_SEMAPHORE](#) = -20L, [L4_PROTO_META](#) = -21L, [L4_PROTO_IOMMU](#) = -22L, [L4_PROTO_DEBUGGER](#) = -23L }
Message tag for IPC operations.
- enum [l4_msgtag_flags](#) {
[L4_MSGTAG_ERROR](#), [L4_MSGTAG_TRANSFER_FPU](#), [L4_MSGTAG_SCHEDULE](#), [L4_MSGTAG_PRIVILEGE](#),
[L4_MSGTAG_FLAGS](#) }
Flags for message tags.

Functions

- `l4_msgtag_t l4_msgtag` (long label, unsigned words, unsigned items, unsigned flags) `L4_NOTHROW`
Create a message tag from the specified values.
- 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_preemption` (`l4_msgtag_t t`) `L4_NOTHROW`
Test for preemption protocol.
- unsigned `l4_msgtag_is_sys_exception` (`l4_msgtag_t t`) `L4_NOTHROW`
Test for system-exception 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.

12.87.1 Detailed Description

API related to the message tag data type.

Include File

```
#include <l4/sys/types.h>
```

12.87.2 Typedef Documentation

12.87.2.1 `l4_msgtag_t`

```
typedef struct l4_msgtag_t l4_msgtag_t
```

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.

12.87.3 Enumeration Type Documentation

12.87.3.1 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_PROPAGATE	Enable IPC propagation. This flag enables IPC propagation, which means an IPC reply-connection from the current caller will be propagated to the new IPC receiver. This makes it possible to propagate an IPC call to a third thread, which may then directly answer to the caller.
L4_MSGTAG_FLAGS	Mask for all flags.

Definition at line 94 of file [types.h](#).

12.87.3.2 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_PREEMPTION	Preemption message.
L4_PROTO_SYS_EXCEPTION	System exception.
L4_PROTO_EXCEPTION	Exception.
L4_PROTO_SIGMA0	Sigma0 protocol.

Enumerator

L4_PROTO_IO_PAGE_FAULT	I/O page fault message.
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_IRQ_MUX	Protocol for IRQ mux (IRQ -> n x IRQ)
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 ddebugger.

Definition at line 49 of file [types.h](#).

12.87.3.3 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 147 of file [platform_control.h](#).

12.87.4 Function Documentation

12.87.4.1 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>t</i>	The tag
----------	---------

Returns

Flags

Definition at line 430 of file [types.h](#).

References [l4_msgtag_t::raw](#).

12.87.4.3 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:

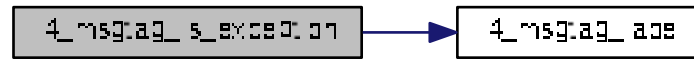
[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 435 of file [types.h](#).

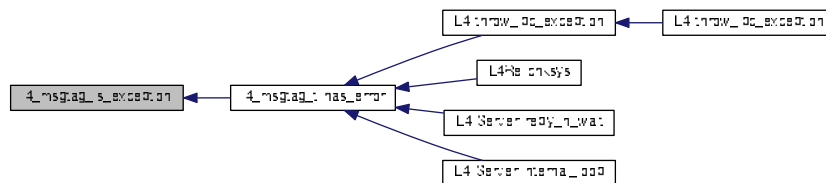
References [L4_MSGTAG_ERROR](#), and [l4_msgtag_t::raw](#).

Referenced by [l4_msgtag_t::has_error\(\)](#), [l4_ipc_error\(\)](#), [l4_vcon_write_u\(\)](#), and [l4vcpu_irq_disable_save\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



12.87.4.5 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

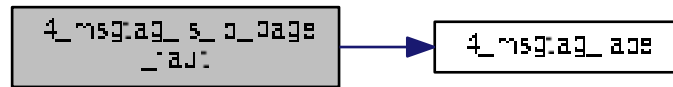
Boolean value

Definition at line 455 of file [types.h](#).

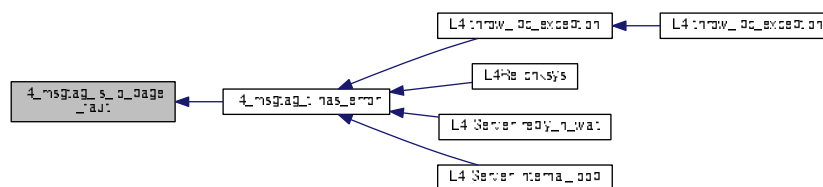
References [l4_msgtag_label\(\)](#), and [L4_PROTO_IO_PAGE_FAULT](#).

Referenced by [l4_msgtag_t::has_error\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



12.87.4.6 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

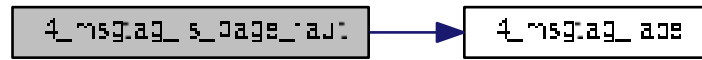
Boolean value

Definition at line 440 of file [types.h](#).

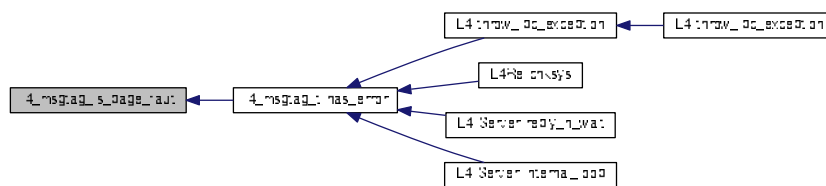
References [l4_msgtag_label\(\)](#), and [L4_PROTO_PAGE_FAULT](#).

Referenced by [l4_msgtag_t::has_error\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



12.87.4.7 l4_msgtag_is_preemption()

```

unsigned l4_msgtag_is_preemption (
    l4_msgtag_t t ) [inline]
  
```

Test for preemption protocol.

Parameters

<i>t</i>	The tag
----------	---------

Returns

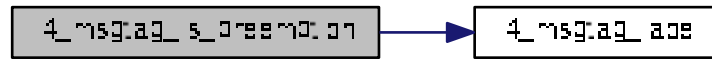
Boolean value

Definition at line 443 of file [types.h](#).

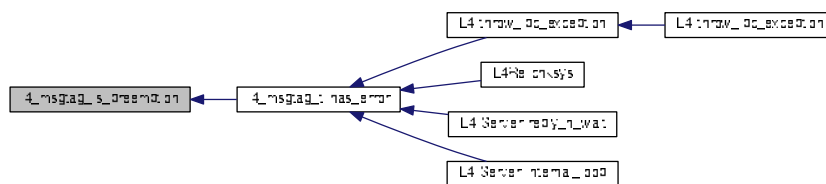
References [l4_msgtag_label\(\)](#), and [L4_PROTO_PREEMPTION](#).

Referenced by [l4_msgtag_t::has_error\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



12.87.4.8 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 452 of file [types.h](#).

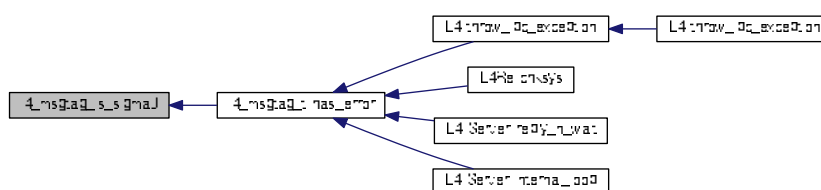
References [l4_msgtag_label\(\)](#), and [L4_PROTO_SIGMA0](#).

Referenced by [l4_msgtag_t::has_error\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



12.87.4.9 l4_msgtag_is_sys_exception()

```
unsigned l4_msgtag_is_sys_exception (
    l4_msgtag_t t ) [inline]
```

Test for system-exception protocol.

Parameters

<i>t</i>	The tag
----------	---------

Returns

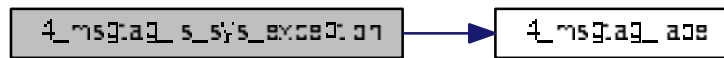
Boolean value

Definition at line 446 of file [types.h](#).

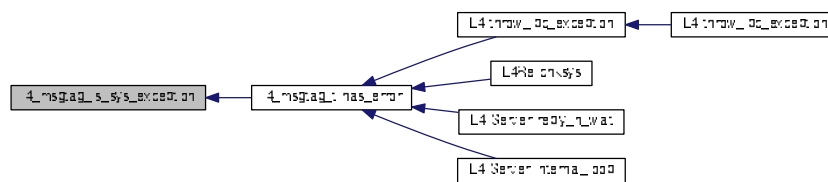
References [l4_msgtag_label\(\)](#), and [L4_PROTO_SYS_EXCEPTION](#).

Referenced by [l4_msgtag_t::has_error\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



12.87.4.10 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

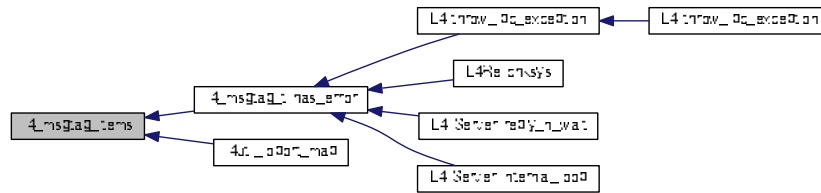
Number of items.

Definition at line 426 of file [types.h](#).

References [l4_msgtag_t::raw](#).

Referenced by [l4_msgtag_t::has_error\(\)](#), and [l4util_ioport_map\(\)](#).

Here is the caller graph for this function:



12.87.4.11 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:

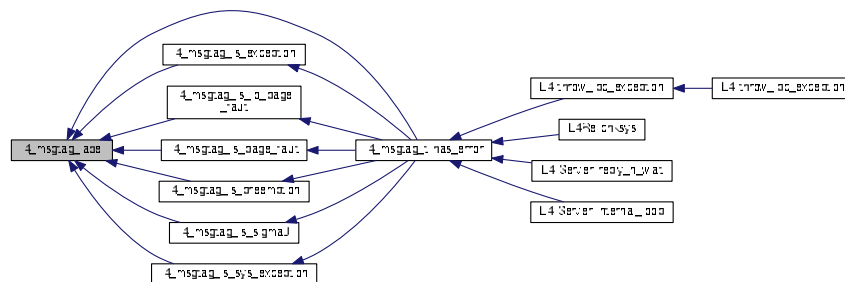
[examples/sys/singlestep/main.c](#), and [examples/sys/start-with-exc/main.c](#).

Definition at line 418 of file [types.h](#).

References [l4_msgtag_t::raw](#).

Referenced by [l4_msgtag_t::has_error\(\)](#), [l4_msgtag_is_exception\(\)](#), [l4_msgtag_is_io_page_fault\(\)](#), [l4_msgtag_is_io_page_fault\(\)](#), [l4_msgtag_is_preemption\(\)](#), [l4_msgtag_is_sigma0\(\)](#), and [l4_msgtag_is_sys_exception\(\)](#).

Here is the caller graph for this function:



12.87.4.12 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:

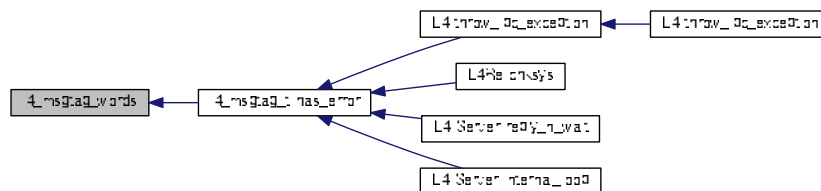
[examples/sys/utcb-ipc/main.c](#).

Definition at line 422 of file [types.h](#).

References [l4_msgtag_t::raw](#).

Referenced by [l4_msgtag_t::has_error\(\)](#).

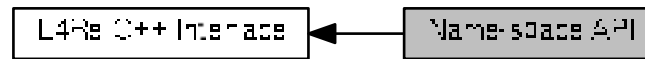
Here is the caller graph for this function:



12.88 Name-space API

API for name spaces that store capabilities.

Collaboration diagram for Name-space API:



Data Structures

- class [L4Re::Namespace](#)
Name-space interface.

12.88.1 Detailed Description

API for name spaces that store capabilities.

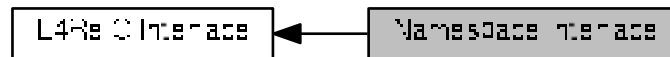
This is a basic abstraction for managing a mapping from human-readable names to capabilities. In particular, a name can also be mapped to a capability that refers to another name space object. By this means name spaces can be constructed hierarchically.

Name spaces play a central role in [L4Re](#), because the implementation of the name space objects determine the policy which capabilities (which objects) are accessible to a client of a name space.

12.89 Namespace interface

Namespace C interface.

Collaboration diagram for Namespace interface:



Enumerations

- enum [l4re_ns_register_flags](#)
Namespace register flags.

Functions

- long [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.
- long [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.
- long [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.

12.89.1 Detailed Description

Namespace C interface.

12.89.2 Enumeration Type Documentation

12.89.2.1 l4re_ns_register_flags

```
enum l4re_ns_register_flags
```

Namespace register flags.

See also

[L4Re::Namespace::Register_flags](#)

Definition at line 39 of file [namespace.h](#).

12.89.3 Function Documentation

12.89.3.1 l4re_ns_query_srv()

```
long 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 105 of file [namespace.h](#).

References [EXTERN_C_END](#), and [l4re_ns_query_to_srv\(\)](#).

Here is the call graph for this function:



12.89.3.2 l4re_ns_query_to_srv()

```
long 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

0	Name could be fully resolved.
>0	Name could only be partly resolved. The number of remaining characters is returned.
-L4_ENOENT	Entry could not be found.
-L4_EAGAIN	Entry exists but no object is yet attached. Try again later.
<0	IPC errors, see l4_error_code_t .

Referenced by [l4re_ns_query_srv\(\)](#).

Here is the caller graph for this function:



12.89.3.3 l4re_ns_register_obj_srv()

```

long 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

0	Object was successfully registered with <i>name</i> .
-L4_EEXIST	Name already registered.
-L4_EPERM	Caller doesn't have necessary permissions.
-L4_ENOMEM	Server has insufficient resources.
-L4_EINVAL	Invalid parameter.
<0	IPC errors, see l4_error_code_t .

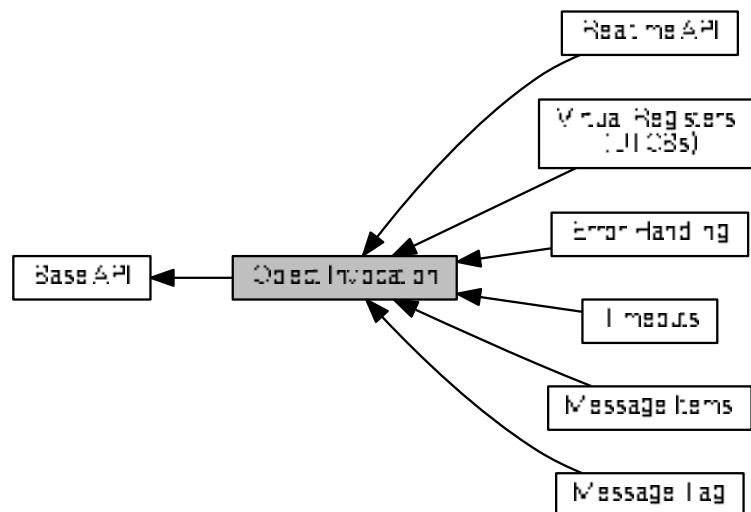
Precondition

requires capability rights: {RW}

12.90 Object Invocation

API for [L4](#) object invocation.

Collaboration diagram for Object Invocation:



Modules

- [Error Handling](#)
Error handling for [L4](#) object invocation.
- [Message Items](#)
Message item related functions.
- [Message Tag](#)
API related to the message tag data type.
- [Realtime API](#)
- [Timeouts](#)
All kinds of timeouts and time related functions.
- [Virtual Registers \(UTCBs\)](#)
[L4](#) Virtual Registers (UTCB).

Files

- file [utcb.h](#)
UTCB definitions.

Enumerations

- enum `l4_syscall_flags_t` {
`L4_SYSF_NONE`, `L4_SYSF_SEND`, `L4_SYSF_RECV`, `L4_SYSF_OPEN_WAIT`,
`L4_SYSF_REPLY`, `L4_SYSF_CALL`, `L4_SYSF_WAIT`, `L4_SYSF_SEND_AND_WAIT`,
`L4_SYSF_REPLY_AND_WAIT` }

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_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_ALWAYS_INLINE 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.
- `int l4_sndfpage_add (l4_fpage_t const snd_fpage, unsigned long snd_base, l4_msgtag_t *tag) L4_NOTHROW`
Add a flex-page to be sent to the UTCB.

12.90.1 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](#)

12.90.2 Enumeration Type Documentation

12.90.2.1 l4_syscall_flags_t

enum `l4_syscall_flags_t`

Capability selector flags.

These flags determine the concrete operation when a kernel object is invoked.

Enumerator

<code>L4_SYSF_NONE</code>	Default flags (call to a kernel object). Using this value as flags in the capability selector for an invocation indicates a call (send and wait for a reply).
<code>L4_SYSF_SEND</code>	Send-phase flag. Setting this flag in a capability selector induces a send phase, this means a message is send to the object denoted by the capability. For receive phase see L4_SYSF_RECV .
<code>L4_SYSF_RECV</code>	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 .
<code>L4_SYSF_OPEN_WAIT</code>	Open-wait flag. This flag indicates that the receive operation (see L4_SYSF_RECV) shall be an <i>open wait</i> . <i>Open wait</i> means that the invoking thread shall wait for a message from any possible sender and <i>not</i> from the sender denoted by the capability.
<code>L4_SYSF_REPLY</code>	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.
<code>L4_SYSF_CALL</code>	Call flags (combines send and receive). Combines L4_SYSF_SEND and L4_SYSF_RECV .
<code>L4_SYSF_WAIT</code>	Wait flags (combines receive and open wait). Combines L4_SYSF_RECV and L4_SYSF_OPEN_WAIT .
<code>L4_SYSF_SEND_AND_WAIT</code>	Send-and-wait flags. Combines L4_SYSF_SEND and L4_SYSF_WAIT .
<code>L4_SYSF_REPLY_AND_WAIT</code>	Reply-and-wait flags. Combines L4_SYSF_SEND , L4_SYSF_REPLY , and L4_SYSF_WAIT .

Definition at line 39 of file [consts.h](#).

12.90.3 Function Documentation

12.90.3.1 l4_ipc()

```
L4_ALWAYS_INLINE l4_msgtag_t l4_ipc (
    l4_cap_idx_t dest,
    l4_utcb_t * utcb,
    l4_umword_t flags,
```


12.90.3.2 `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.
<i>utcb</i>	UTCB of the caller.
<i>tag</i>	Message tag to describe the message to be sent.
<i>timeout</i>	Timeout pair for send an 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.

Examples:

[examples/sys/aliens/main.c](#), [examples/sys/ipc/ipc_example.c](#), and [examples/sys/singlestep/main.c](#).

Definition at line [445](#) of file [ipc.h](#).

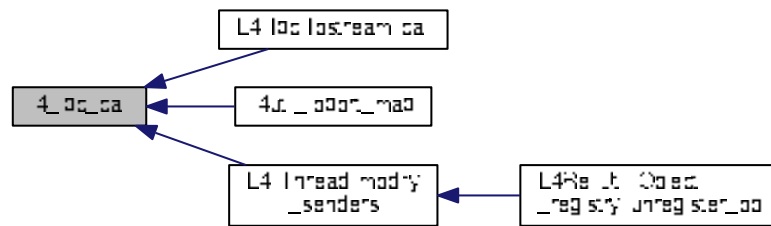
References [l4_ipc\(\)](#), and [L4_SYSF_CALL](#).

Referenced by [L4::ipc::loststream::call\(\)](#), [l4util_ioport_map\(\)](#), and [L4::Thread::modify_senders\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



12.90.3.3 l4_ipc_receive()

```

l4_msgtag_t l4_ipc_receive (
    l4_cap_idx_t object,
    l4_utcb_t * utcb,
    l4_timeout_t timeout ) [inline]
  
```

Wait for a message from a specific source.

Parameters

<i>object</i>	Object to receive a message from.
<i>timeout</i>	Timeout pair (see l4_timeout_t , only the receive part matters).
<i>utcb</i>	UTCB of the caller.

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.

Note

This operation is usually used to receive messages from a specific IRQ or thread. However, it is not common to use this operation for normal applications.

Examples:

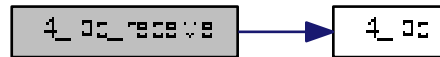
[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line [487](#) of file [ipc.h](#).

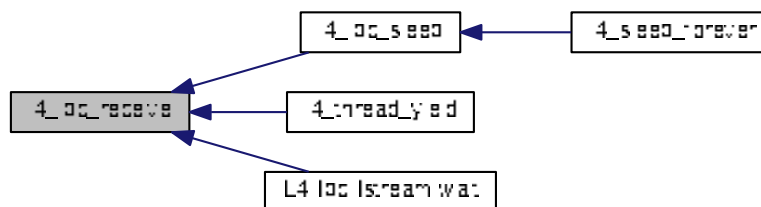
References [l4_ipc\(\)](#), [L4_SYSF_RECV](#), and [l4_msgtag_t::raw](#).

Referenced by [l4_ipc_sleep\(\)](#), [l4_thread_yield\(\)](#), and [L4::lpc::lstream::wait\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



12.90.3.4 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>tag</i>	Describes the message to be sent as reply.
	<i>utcb</i>	UTCB of the caller.
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.

Examples:

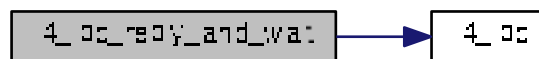
[examples/sys/ipc/ipc_example.c](#).

Definition at line 453 of file [ipc.h](#).

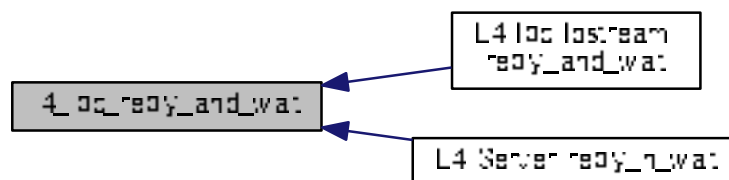
References [L4_INVALID_CAP](#), [l4_ipc\(\)](#), and [L4_SYSF_REPLY_AND_WAIT](#).

Referenced by [L4::lpc::lostream::reply_and_wait\(\)](#), and [L4::Server< LOOP_HOOKS >::reply_n_wait\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**12.90.3.5 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.
<i>utcb</i>	UTCB of the caller.
<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

result tag

A message is sent to the destination object. There is no receive phase included. The invoker continues working after sending the message.

Attention

This is a special-purpose message transfer, objects usually support only invocation via [l4_ipc_call\(\)](#).

Examples:

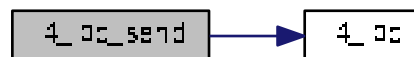
[examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 470 of file [ipc.h](#).

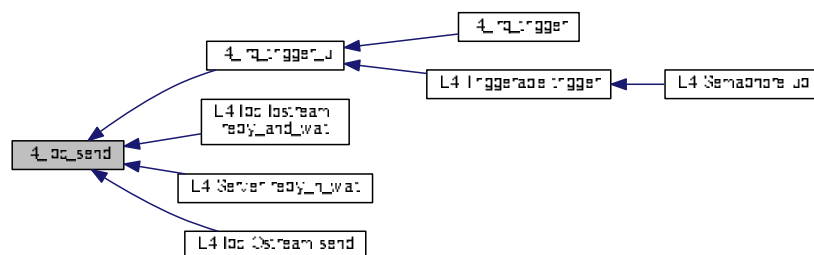
References [l4_ipc\(\)](#), and [L4_SYSF_SEND](#).

Referenced by [l4_irq_trigger_u\(\)](#), [L4::lpc::lstream::reply_and_wait\(\)](#), [L4::Server< LOOP_HOOKS >::reply_n_wait\(\)](#), and [L4::lpc::Ostream::send\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



12.90.3.6 l4_ipc_send_and_wait()

```
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 ) [inline]
```

Send a message and do an open wait.

Parameters

	<i>dest</i>	Object to send a message to.
	<i>utcb</i>	UTCB of the caller.
	<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

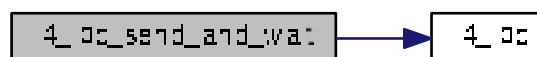
This is a special-purpose operation and shall not be used in general applications.

Definition at line 461 of file [ipc.h](#).

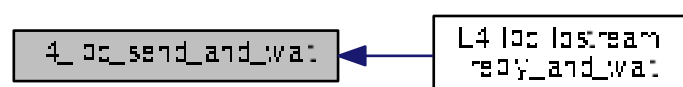
References [l4_ipc\(\)](#), and [L4_SYSF_SEND_AND_WAIT](#).

Referenced by [L4::ipc::lostream::reply_and_wait\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



12.90.3.7 `l4_ipc_sleep()`

```
l4_msgtag_t l4_ipc_sleep (
    l4_timeout_t timeout ) [inline]
```

Sleep for an amount of time.

Parameters

<code>timeout</code>	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\(\)](#)).

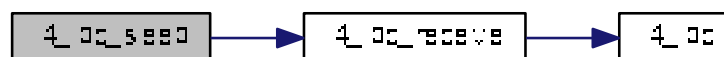
The invoking thread waits until the timeout is expired or the wait was aborted by another thread by [l4_thread_ex_regs\(\)](#).

Definition at line 496 of file [ipc.h](#).

References [L4_INVALID_CAP](#), and [l4_ipc_receive\(\)](#).

Referenced by [l4_sleep_forever\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



12.90.3.8 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 of the caller.
-------------	---------------------

Return values

<i>label</i>	Label assigned to the source object (IPC gate or IRQ).
--------------	--

Parameters

<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\(\)](#).

Examples:

[examples/sys/ipc/ipc_example.c](#).

Definition at line [478](#) of file [ipc.h](#).

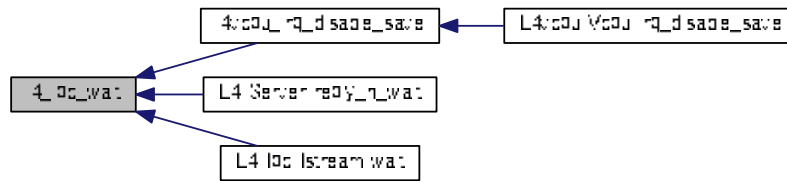
References [L4_INVALID_CAP](#), [l4_ipc\(\)](#), [L4_SYSF_WAIT](#), and [l4_msgtag_t::raw](#).

Referenced by [l4vcpu_irq_disable_save\(\)](#), [L4::Server< LOOP_HOOKS >::reply_n_wait\(\)](#), and [L4::lpc::lstream< >::wait\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



12.90.3.9 l4_sndpage_add()

```
int l4_sndpage_add (
    l4_fpage_t const snd_fpage,
    unsigned long snd_base,
    l4_msgtag_t * tag ) [inline]
```

Add a flex-page to be sent to the UTCB.

Parameters

<i>snd_fpage</i>	Flex-page.
<i>snd_base</i>	Send base.
<i>tag</i>	Tag to be modified.

Return values

<i>tag</i>	Modified tag, the number of items will be increased, all other values in the tag will be retained.
------------	--

Returns

0 on success, negative error code otherwise

Definition at line 556 of file [ipc.h](#).

12.91 Parent API

[Parent](#) interface.

Collaboration diagram for Parent API:



Data Structures

- class [L4Re::Parent](#)
[Parent](#) interface.

12.91.1 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.

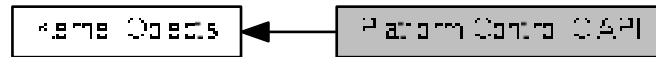
See also

[L4Re::Parent](#) for information about the concrete interface.

12.92 Platform Control C API

C interface for controlling platform-wide properties.

Collaboration diagram for Platform Control C API:



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_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.

12.92.1 Detailed Description

C interface for controlling platform-wide properties.

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](#)

12.92.2 Function Documentation

12.92.2.1 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

Definition at line 232 of file [platform_control.h](#).

12.92.2.2 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

Definition at line 225 of file [platform_control.h](#).

12.92.2.3 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>pfc</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 194 of file [platform_control.h](#).

12.92.2.4 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.

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.

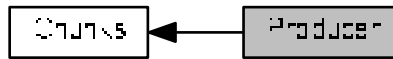
Returns

Syscall return tag

Definition at line 187 of file [platform_control.h](#).

12.93 Producer

Collaboration diagram for Producer:



Functions

- long [l4shmc_chunk_try_to_take](#) (l4shmc_chunk_t *chunk)
Try to mark chunk busy.
- 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 *chunk)
Check whether chunk is free.

12.93.1 Detailed Description

12.93.2 Function Documentation

12.93.2.1 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.

12.93.2.2 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](#).

12.93.2.3 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](#).

12.93.2.4 l4shmc_is_chunk_clear()

```
long l4shmc_is_chunk_clear (
    l4shmc_chunk_t * chunk )    [inline]
```

Check whether chunk is free.

Parameters

<i>chunk</i>	Chunk to check.
--------------	-----------------

Return values

0	Success.
<0	Error.

12.94 Producer

Collaboration diagram for Producer:



Functions

- long [l4shmc_trigger](#) (l4shmc_signal_t *signal)
Trigger a signal.

12.94.1 Detailed Description

12.94.2 Function Documentation

12.94.2.1 l4shmc_trigger()

```
long l4shmc_trigger (
    l4shmc_signal_t * signal ) [inline]
```

Trigger a signal.

Parameters

<i>signal</i>	Signal to trigger.
---------------	--------------------

Return values

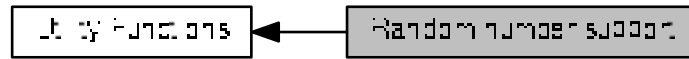
0	Success.
<0	Error.

Examples:

[examples/libs/shmc/prodcons.c](#).

12.95 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.

12.95.1 Detailed Description

12.95.2 Function Documentation

12.95.2.1 l4util_rand()

```
l4_uint32_t l4util_rand (
    void )
```

Deliver next random number.

Returns

A new random number

12.95.2.2 l4util_srand()

```
void l4util_srand (
    l4_uint32_t seed )
```

Initialize random number generator.

Parameters

<i>seed</i>	Value to initialize
-------------	---------------------

12.96 Realtime API

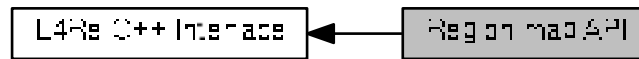
Collaboration diagram for Realtime API:



12.97 Region map API

Virtual address-space management.

Collaboration diagram for Region map API:



Data Structures

- class [L4Re::Rm](#)
Region map.

12.97.1 Detailed Description

Virtual address-space management.

The central purpose of the region-map API is to provide means to manage the virtual memory address space of an [L4](#) task. 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, that allows to attach a data-space object to a region of the virtual address space.

There are two basic concepts provided by a region-map abstraction:

- Regions provide a means to create a view to a data space (or parts of a data space).
- Areas provide a means to reserve areas in a virtual memory address space for special purposes. A reserved area is skipped when searching for an available range of virtual memory, or may be explicitly used to search only within that area.

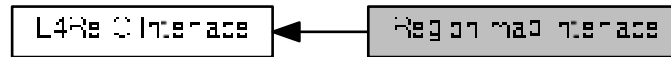
See also

[L4Re::Dataspace](#), [L4Re::Rm](#)

12.98 Region map interface

Region map C interface.

Collaboration diagram for Region map interface:



Enumerations

- enum `l4re_rm_flags_t` {
`L4RE_RM_READ_ONLY` = 0x01, `L4RE_RM_NO_ALIAS` = 0x02, `L4RE_RM_PAGER` = 0x04, `L4RE_RM_RESERVED` = 0x08,
`L4RE_RM_REGION_FLAGS` = 0x0f, `L4RE_RM_OVERMAP` = 0x10, `L4RE_RM_SEARCH_ADDR` = 0x20,
`L4RE_RM_IN_AREA` = 0x40,
`L4RE_RM_EAGER_MAP` = 0x80, `L4RE_RM_ATTACH_FLAGS` = 0xf0 }

Flags for region operations.

Functions

- int `l4re_rm_reserve_area` (`l4_addr_t` *start, unsigned long size, unsigned flags, unsigned char align) `L4_NOTHROW`
- int `l4re_rm_free_area` (`l4_addr_t` addr) `L4_NOTHROW`
- int `l4re_rm_attach` (void **start, unsigned long size, unsigned long flags, `l4re_ds_t` const mem, `l4_addr_t` offs, unsigned char align) `L4_NOTHROW`
- int `l4re_rm_detach` (void *addr) `L4_NOTHROW`
Detach and unmap in current task.
- int `l4re_rm_detach_ds` (void *addr, `l4re_ds_t` *ds) `L4_NOTHROW`
Detach, unmap and return affected dataspace in 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, `l4_addr_t` *offset, unsigned *flags, `l4re_ds_t` *m) `L4_NOTHROW`
- 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, unsigned 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, unsigned long flags, `l4re_ds_t` const mem, `l4_addr_t` offs, unsigned char align) `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, `l4_addr_t` *offset, unsigned *flags, `l4re_ds_t` *m) `L4_NOTHROW`
- void `l4re_rm_show_lists_srv` (`l4_cap_idx_t` rm) `L4_NOTHROW`
Dump region map internal data structures.

12.98.1 Detailed Description

Region map C interface.

12.98.2 Enumeration Type Documentation

12.98.2.1 l4re_rm_flags_t

```
enum l4re_rm_flags_t
```

Flags for region operations.

Enumerator

L4RE_RM_READ_ONLY	Region is read-only.
L4RE_RM_NO_ALIAS	The region contains exclusive memory that is not mapped anywhere else.
L4RE_RM_PAGER	Region has a pager.
L4RE_RM_RESERVED	Region is reserved (blocked)
L4RE_RM_REGION_FLAGS	Mask of all region flags.
L4RE_RM_OVERMAP	Unmap memory already mapped in the region.
L4RE_RM_SEARCH_ADDR	Search for a suitable address range.
L4RE_RM_IN_AREA	Search only in area, or map into area.
L4RE_RM_EAGER_MAP	Eagerly map the attached data space in.
L4RE_RM_ATTACH_FLAGS	Mask of all attach flags.

Definition at line 40 of file [rm.h](#).

12.98.3 Function Documentation

12.98.3.1 l4re_rm_attach()

```
int l4re_rm_attach (
    void ** start,
    unsigned long size,
    unsigned long flags,
    l4re_ds_t const mem,
    l4_addr_t offs,
    unsigned char align ) [inline]
```

Parameters

<i>in, out</i>	<i>start</i>	Virtual start address where the region manager shall attach the data space. If L4Re::Rm::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::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)
	<i>flags</i>	Flags, see L4Re::Rm::Attach_flags and L4Re::Rm::Region_flags . 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::Search_addr flag is used.

Return values

<i>0</i>	Success
<i>-L4_ENOENT</i>	No area could be found (see L4Re::Rm::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](#)).

Returns

0 on success, <0 on error

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 243 of file [rm.h](#).

References [l4re_rm_attach_srv\(\)](#).

Here is the call graph for this function:



12.98.3.2 l4re_rm_attach_srv()

```
int l4re_rm_attach_srv (
    l4_cap_idx_t rm,
    void ** start,
    unsigned long size,
    unsigned long flags,
    l4re_ds_t const mem,
    l4_addr_t offs,
    unsigned char align )
```

See also

[L4Re::Rm::attach](#)

Referenced by [l4re_rm_attach\(\)](#).

Here is the caller graph for this function:



12.98.3.3 l4re_rm_detach()

```
int l4re_rm_detach (
    void * addr ) [inline]
```

Detach and unmap in current task.

Parameters

<i>addr</i>	Address of the region to detach.
-------------	----------------------------------

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 253 of file `rm.h`.

References [l4re_rm_detach_srv\(\)](#).

Here is the call graph for this function:

**12.98.3.4 l4re_rm_detach_ds()**

```
int l4re_rm_detach_ds (
    void * addr,
    l4re_ds_t * ds ) [inline]
```

Detach, unmap and return affected dataspace in current task.

Parameters

<i>addr</i>	Address of the region to detach.
-------------	----------------------------------

Return values

<i>ds</i>	Returns dataspace that is affected.
-----------	-------------------------------------

Returns

0 on success, <0 on error

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 266 of file [rm.h](#).

References [l4re_rm_detach_srv\(\)](#).

Here is the call graph for this function:

**12.98.3.5 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.
-------------	----------------------------------

Return values

<i>ds</i>	Returns dataspace that is affected.
-----------	-------------------------------------

Parameters

<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 273 of file [rm.h](#).

References [l4re_rm_detach_srv\(\)](#).

Here is the call graph for this function:

**12.98.3.6 l4re_rm_detach_srv()**

```

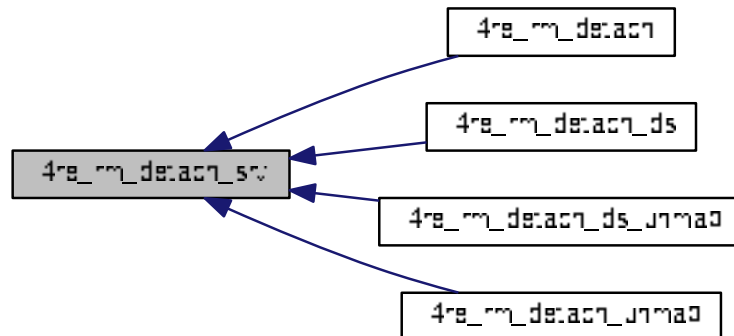
int l4re_rm_detach_srv (
    l4_cap_idx_t rm,
    l4_addr_t addr,
    l4re_ds_t * ds,
    l4_cap_idx_t task )
  
```

See also

[L4Re::Rm::detach](#)

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:



12.98.3.7 `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 <code>L4_INVALID_CAP</code> 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 260 of file `rm.h`.

References [l4re_rm_detach_srv\(\)](#).

Here is the call graph for this function:



12.98.3.8 l4re_rm_find()

```

int l4re_rm_find (
    l4_addr_t * addr,
    unsigned long * size,
    l4_addr_t * offset,
    unsigned * flags,
    l4re_ds_t * m ) [inline]
  
```

Returns

0 on success, <0 on error

See also

[L4Re::Rm::find](#)

Definition at line 280 of file `rm.h`.

References [l4re_rm_find_srv\(\)](#).

Here is the call graph for this function:



12.98.3.9 l4re_rm_find_srv()

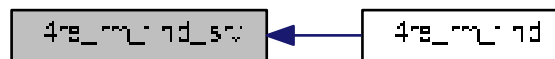
```
int l4re_rm_find_srv (
    l4_cap_idx_t rm,
    l4_addr_t * addr,
    unsigned long * size,
    l4_addr_t * offset,
    unsigned * flags,
    l4re_ds_t * m )
```

See also

[L4Re::Rm::find](#)

Referenced by [l4re_rm_find\(\)](#).

Here is the caller graph for this function:



12.98.3.10 l4re_rm_free_area()

```
int l4re_rm_free_area (
    l4_addr_t addr ) [inline]
```

Returns

0 on success, <0 on error

See also

[L4Re::Rm::free_area](#)

This function is using the `L4::Env::env()->rm()` service.

Definition at line 237 of file `rm.h`.

References [l4re_rm_free_area_srv\(\)](#).

Here is the call graph for this function:



12.98.3.11 `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](#)

Referenced by [l4re_rm_free_area\(\)](#).

Here is the caller graph for this function:

12.98.3.12 `l4re_rm_reserve_area()`

```
int l4re_rm_reserve_area (
    l4_addr_t * start,
    unsigned long size,
    unsigned flags,
    unsigned char align ) [inline]
```

Returns

0 on success, <0 on error

See also

[L4Re::Rm::reserve_area](#)

This function is using the `L4::Env::env()->rm()` service.

Definition at line 229 of file [rm.h](#).

References [l4re_rm_reserve_area_srv\(\)](#).

Here is the call graph for this function:



12.98.3.13 l4re_rm_reserve_area_srv()

```
int l4re_rm_reserve_area_srv (
    l4_cap_idx_t rm,
    l4_addr_t * start,
    unsigned long size,
    unsigned flags,
    unsigned char align )
```

See also

[L4Re::Rm::reserve_area](#)

Referenced by [l4re_rm_reserve_area\(\)](#).

Here is the caller graph for this function:



12.98.3.14 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 287 of file [rm.h](#).

References [l4re_rm_show_lists_srv\(\)](#).

Here is the call graph for this function:



12.99 Scheduler

C interface of the Scheduler kernel object.

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_ops](#) { [L4_SCHEDULER_INFO_OP](#) = 0UL, [L4_SCHEDULER_RUN_THREAD_OP](#) = 1↔UL, [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_sched_param_t](#) [l4_sched_param](#) (unsigned prio, [l4_cpu_time_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 μs) 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.

12.99.1 Detailed Description

C interface of the Scheduler kernel object.

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.

Include File

```
#include <l4/sys/scheduler.h>
```

12.99.2 Enumeration Type Documentation

12.99.2.1 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 200 of file [scheduler.h](#).

12.99.3 Function Documentation

12.99.3.1 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.
<i>granularity</i>	Granularity in log2 notation.
<i>map</i>	Bitmap of CPUs, defaults to 1 in C++.

Returns

CPU set.

Examples:

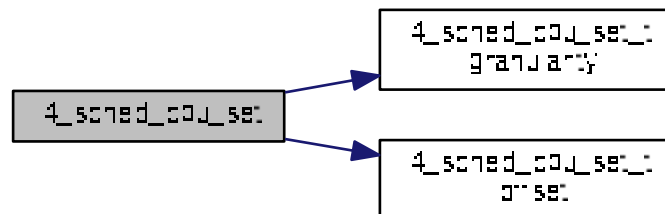
[examples/sys/migrate/thread_migrate.cc](#).

Definition at line 210 of file [scheduler.h](#).

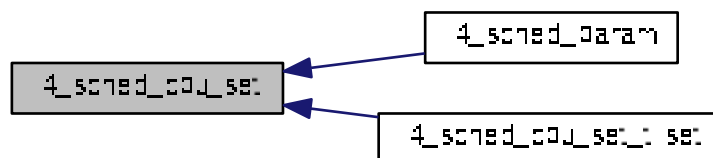
References [l4_sched_cpu_set_t::gran_offset](#), [l4_sched_cpu_set_t::granularity\(\)](#), [l4_sched_cpu_set_t::map](#), and [l4_sched_cpu_set_t::offset\(\)](#).

Referenced by [l4_sched_param\(\)](#), and [l4_sched_cpu_set_t::set\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



12.99.3.2 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 <code>cpu</code> 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 322 of file [scheduler.h](#).

12.99.3.3 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.
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.

Return values

0	Success.
-L4_EINVAL	The given CPU offset is larger than the maximum number of CPUs.

Definition at line 308 of file [scheduler.h](#).

12.99.3.4 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 329 of file [scheduler.h](#).

12.99.3.5 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

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 physical CPUs.
- Two threads with a single identical CPU selected in the CPU set must be scheduled to the same physical 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 315 of file [scheduler.h](#).

12.100 Server-Side IPC framework

Server-Side framework for implementing object-oriented servers.

Namespaces

- [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 a 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::Timed_work< HOOKS >](#)
DEPRECATED.
- 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::Irq_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.

12.100.1 Detailed Description

Server-Side framework for implementing object-oriented servers.

12.100.2 Enumeration Type Documentation

12.100.2.1 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

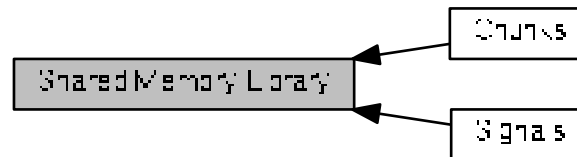
<code>Reply_compound</code>	Server shall use a compound reply and wait (fast).
<code>Reply_separate</code>	Server shall call reply and wait separately.

Definition at line 50 of file [ipc_server_loop](#).

12.101 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:



Modules

- [Chunks](#)
- [Signals](#)

Functions

- long [l4shmc_create](#) (const char *shmc_name, [l4_umword_t](#) shm_size)
Create a shared memory area.
- long [l4shmc_attach](#) (const char *shmc_name, [l4shmc_area_t](#) *shmarea)
Attach to a shared memory area.
- long [l4shmc_attach_to](#) (const char *shmc_name, [l4_umword_t](#) timeout_ms, [l4shmc_area_t](#) *shmarea)
Attach to a shared memory area, with limited waiting.
- 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](#) *shmarea)
Get size of shared memory area.
- long [l4shmc_area_size_free](#) ([l4shmc_area_t](#) *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.

12.101.1 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).

12.101.2 Function Documentation

12.101.2.1 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

12.101.2.2 l4shmc_area_size()

```
long l4shmc_area_size (  
    l4shmc_area_t * shmarea ) [inline]
```

Get size of shared memory area.

Parameters

<i>shmarea</i>	Shared memory area.
----------------	---------------------

Return values

>0	Size of the shared memory area.
<0	Error.

12.101.2.3 l4shmc_area_size_free()

```
long l4shmc_area_size_free (
    l4shmc_area_t * 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.
----------------	---------------------

Return values

0	Free capacity in the area.
<0	Error.

12.101.2.4 l4shmc_attach()

```
long l4shmc_attach (
    const char * shmc_name,
    l4shmc_area_t * shmarea ) [inline]
```

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.

Return values

0	Success.
-----	----------

Return values

<0	Error.
------	--------

Examples:

[examples/libs/shmc/prodcons.c](#).

12.101.2.5 `l4shmc_attach_to()`

```
long l4shmc_attach_to (
    const char * shmc_name,
    l4_umword_t timeout_ms,
    l4shmc_area_t * shmarea )
```

Attach to a shared memory area, with limited waiting.

Parameters

	<i>shmc_name</i>	Name of the shared memory area.
	<i>timeout_ms</i>	Timeout to wait for shm area in milliseconds.
out	<i>shmarea</i>	Pointer to shared memory area descriptor to be filled with information for the shared memory area.

Return values

0	Success.
<0	Error.

12.101.2.6 `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

12.101.2.7 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.

Returns

0 on success, <0 on error

Examples:

[examples/libs/shmc/prodcons.c](#).

12.101.2.8 `l4shmc_create()`

```
long l4shmc_create (
    const char * shmc_name,
    l4_umword_t shm_size )
```

Create a shared memory area.

Parameters

<i>shmc_name</i>	Name of the shared memory area.
<i>shm_size</i>	Size of the whole shared memory area.

Return values

0	Success.
<0	Error.

Examples:

[examples/libs/shmc/prodcons.c](#).

12.102 Sigma0 API

Sigma0 API bindings.

Collaboration diagram for Sigma0 API:



Modules

- [Internal constants](#)
Internal sigma0 definitions.

Files

- file [sigma0.h](#)
Sigma0 interface.

Enumerations

- enum [l4sigma0_return_flags_t](#) {
L4SIGMA0_OK, L4SIGMA0_NOTALIGNED, L4SIGMA0_IPCERROR, L4SIGMA0_NOFPAGE ,
L4SIGMA0_SMALLERFPAGE }
Return flags of libsigma0 functions.

Functions

- [l4_kernel_info_t](#) * [l4sigma0_map_kip](#) ([l4_cap_idx_t](#) sigma0, void *addr, unsigned log2_size)
Map the kernel info page from pager 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.
- int [l4sigma0_map_tbuf](#) ([l4_cap_idx_t](#) sigma0, [l4_addr_t](#) virt)
Request Fiasco trace buffer.
- void [l4sigma0_debug_dump](#) ([l4_cap_idx_t](#) sigma0)
Request sigma0 to dump internal debug information.
- int [l4sigma0_new_client](#) ([l4_cap_idx_t](#) sigma0, [l4_cap_idx_t](#) gate)
Create a new IPC gate for a new Sigma0 client.
- char const * [l4sigma0_map_errstr](#) (int err)
Get a user readable error messages for the return codes.

12.102.1 Detailed Description

Sigma0 API bindings.

Convenience bindings for the Sigma0 protocol.

12.102.2 Enumeration Type Documentation

12.102.2.1 l4sigma0_return_flags_t

```
enum l4sigma0_return_flags_t
```

Return flags of libsigma0 functions.

Enumerator

L4SIGMA0_OK	Ok.
L4SIGMA0_NOTALIGNED	Phys, virt or size not aligned.
L4SIGMA0_IPCERROR	IPC error.
L4SIGMA0_NOFPAGE	No fpage received.
L4SIGMA0_SMALLERFPAGE	Superpage requested but smaller flexpage received.

Definition at line 81 of file [sigma0.h](#).

12.102.3 Function Documentation

12.102.3.1 l4sigma0_debug_dump()

```
void l4sigma0_debug_dump (  
    l4_cap_idx_t sigma0 )
```

Request sigma0 to dump internal debug information.

The debug information, such as internal memory maps, as well as statistics about the internal allocators is dumped to the kernel debugger.

Parameters

<i>sigma0</i>	the sigma0 thread id.
---------------	-----------------------

12.102.3.2 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.

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\(\)](#).

Parameters

<i>sigma0</i>	usually the thread id of sigma0.
<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.

Return values

<i>base</i>	physical address of the page received (i.e., the send base of the received mapping if any).
-------------	---

Parameters

<i>sz</i>	Size to map by the server, in 2^{sz} bytes.
-----------	--

Returns

0 on success, !=0 else (see [l4sigma0_map_errstr\(\)](#)).

12.102.3.3 l4sigma0_map_errstr()

```
char const * l4sigma0_map_errstr (
    int err ) [inline]
```

Get a 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 213 of file [sigma0.h](#).

References [EXTERN_C_END](#).

12.102.3.4 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 to pager implementing the Sigma0 protocol.
<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	phys, virt, or size 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.

12.102.3.5 l4sigma0_map_kip()

```
l4_kernel_info_t* l4sigma0_map_kip (
    l4_cap_idx_t sigma0,
    void * addr,
    unsigned log2_size )
```

Map the kernel info page from pager 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.

12.102.3.6 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>	ID of service talking the sigma0 protocol.
<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.

Returns

0 on success, !=0 else (see [l4sigma0_map_errstr\(\)](#)).

12.102.3.7 l4sigma0_map_tbuf()

```
int l4sigma0_map_tbuf (
    l4_cap_idx_t sigma0,
    l4_addr_t virt )
```

Request Fiasco trace buffer.

This is a Fiasco specific feature. Where you can request the kernel internal trace buffer for user-level evaluation. This is for special debugging tools, such as Ferret.

Parameters

<i>sigma0</i>	as usual the sigma0 thread id.
<i>virt</i>	the virtual address where the trace buffer should be mapped,

Returns

0 on success, !=0 else (see [l4sigma0_map_errstr\(\)](#)).

12.102.3.8 l4sigma0_new_client()

```
int l4sigma0_new_client (
    l4_cap_idx_t sigma0,
    l4_cap_idx_t gate )
```

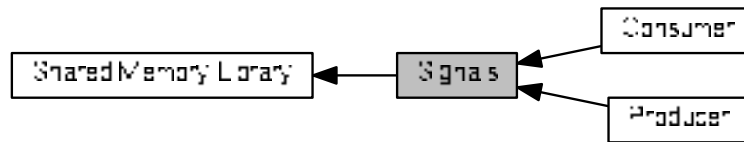
Create a new IPC gate for a new Sigma0 client.

Parameters

<i>sigma0</i>	Capability selector for sigma0 gate.
<i>gate</i>	Capability selector to use for the new gate.

12.103 Signals

Collaboration diagram for Signals:



Modules

- [Consumer](#)
- [Producer](#)

Functions

- long [l4shmc_add_signal](#) (l4shmc_area_t *shmarea, const char *signal_name, l4shmc_signal_t *signal)
Add a signal for the shared memory area.
- long [l4shmc_attach_signal](#) (l4shmc_area_t *shmarea, const char *signal_name, [l4_cap_idx_t](#) thread, l4shmc_signal_t *signal)
Attach to signal.
- long [l4shmc_attach_signal_to](#) (l4shmc_area_t *shmarea, const char *signal_name, [l4_cap_idx_t](#) thread, [l4_umword_t](#) timeout_ms, l4shmc_signal_t *signal)
Attach to signal, with timeout.
- long [l4shmc_get_signal_to](#) (l4shmc_area_t *shmarea, const char *signal_name, [l4_umword_t](#) timeout_ms, l4shmc_signal_t *signal)
Get signal object from the shared memory area.
- [l4_cap_idx_t](#) [l4shmc_signal_cap](#) (l4shmc_signal_t *signal)
Get the signal capability of a signal.
- long [l4shmc_check_magic](#) (l4shmc_chunk_t *chunk)
Check magic value of a chunk.

12.103.1 Detailed Description

12.103.2 Function Documentation

12.103.2.1 l4shmc_add_signal()

```

long l4shmc_add_signal (
    l4shmc_area_t * shmarea,
    const char * signal_name,
    l4shmc_signal_t * signal )

```

Add a signal for the shared memory area.

Parameters

	<i>shmarea</i>	The shared memory area to put the chunk in.
	<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](#).

12.103.2.2 l4shmc_attach_signal()

```
long l4shmc_attach_signal (
    l4shmc_area_t * shmarea,
    const char * signal_name,
    l4_cap_idx_t thread,
    l4shmc_signal_t * signal ) [inline]
```

Attach to signal.

Parameters

	<i>shmarea</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.

12.103.2.3 l4shmc_attach_signal_to()

```
long l4shmc_attach_signal_to (
    l4shmc_area_t * shmarea,
    const char * signal_name,
    l4_cap_idx_t thread,
```

```

    l4_umword_t timeout_ms,
    l4shmc_signal_t * signal )

```

Attach to signal, with timeout.

Parameters

	<i>shmarea</i>	Shared memory area.
	<i>signal_name</i>	Name of the signal.
	<i>thread</i>	Thread capability index to attach the signal to.
	<i>timeout_ms</i>	Timeout in milliseconds to wait for the chunk to appear in the shared memory area.
out	<i>signal</i>	Signal data structure to fill.

Return values

0	Success.
<0	Error.

Examples:

[examples/libs/shmc/prodcons.c](#).

12.103.2.4 l4shmc_check_magic()

```

long l4shmc_check_magic (
    l4shmc_chunk_t * 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.

12.103.2.5 l4shmc_get_signal_to()

```

long l4shmc_get_signal_to (
    l4shmc_area_t * shmarea,
    const char * signal_name,

```



```
l4_umword_t timeout_ms,  
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.
	<i>timeout_ms</i>	Timeout in milliseconds to wait for signal of a chunk to appear in the shared memory area.
out	<i>signal</i>	Signal data structure to fill.

Return values

0	Success.
<0	Error.

12.103.2.6 l4shmc_signal_cap()

```
l4_cap_idx_t l4shmc_signal_cap (  
    l4shmc_signal_t * signal ) [inline]
```

Get the signal capability of a signal.

Parameters

<i>signal</i>	Signal.
---------------	---------

Returns

Capability of the signal object.

12.104 Small C++ Template Library

Namespaces

- [cxx](#)

Our C++ library.

Data Structures

- class [L4::Alloc_list](#)
A simple list-based allocator.
- class [cxx::Auto_ptr< T >](#)
Smart pointer with automatic deletion.
- class [cxx::Bitmap_base](#)
Basic bitmap abstraction.
- class [cxx::Bitmap< BITS >](#)
A static bit map.
- 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 T1 >`
`T1 cxx::min (T1 a, T1 b)`
Get the minimum of a and b.
- `template<typename T1 >`
`T1 cxx::max (T1 a, T1 b)`
Get the maximum of a and b.
- `void * operator new (size_t, void *mem, cxx::Nothrow const &) throw ()`
Simple placement new operator.
- `void * operator new (size_t, cxx::Nothrow const &) throw ()`
New operator that does not throw exceptions.

12.104.1 Detailed Description

12.104.2 Function Documentation

12.104.2.1 max()

```
template<typename T1 >
T1 cxx::max (
    T1 a,
    T1 b ) [inline]
```

Get the maximum of *a* and *b*.

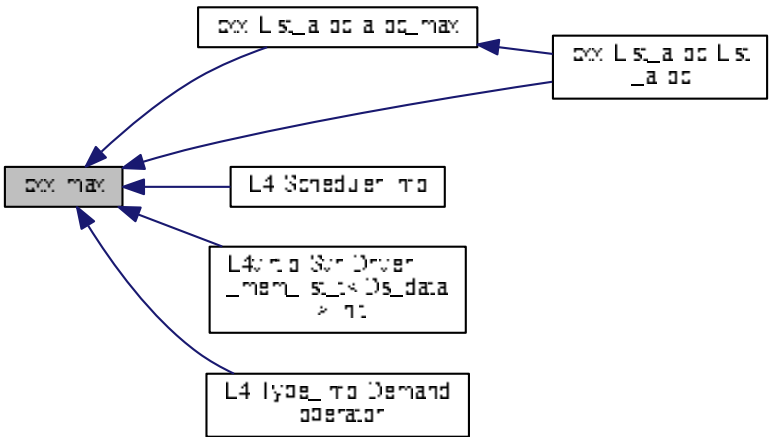
Parameters

<i>a</i>	the first value.
<i>b</i>	the second value.

Definition at line 45 of file [minmax](#).

Referenced by [cxx::List_alloc::alloc_max\(\)](#), [L4::Scheduler::info\(\)](#), [L4virtio::Svr::Driver_mem_list_t< Ds_data >::init\(\)](#), [cxx::List_alloc::List_alloc\(\)](#), and [L4::Type_info::Demand::operator|\(\)](#).

Here is the caller graph for this function:



12.104.2.2 min()

```
template<typename T1 >
T1 cxx::min (
    T1 a,
    T1 b ) [inline]
```

Get the minimum of *a* and *b*.

Parameters

<i>a</i>	the first value.
<i>b</i>	the second value.

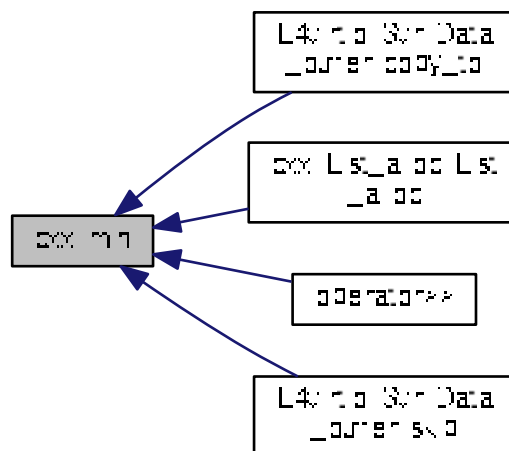
Examples:

[tmpfs/lib/src/fs.cc](#).

Definition at line 35 of file [minmax](#).

Referenced by [L4virtio::Svr::Data_buffer::copy_to\(\)](#), [cxx::List_alloc::List_alloc\(\)](#), [operator>>\(\)](#), and [L4virtio::Svr::Data_buffer::skip\(\)](#).

Here is the caller graph for this function:



12.104.2.3 operator new()

```
void* operator new (
    size_t ,
    void * mem,
    cxx::Nothrow const & ) throw ) [inline]
```

Simple placement new operator.

Parameters

<i>mem</i>	the address of the memory block to place the new object.
------------	--

Returns

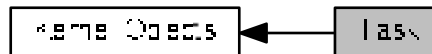
the address given by *mem*.

Definition at line 39 of file [std_alloc](#).

12.105 Task

C interface of the Task kernel object.

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_map (l4_cap_idx_t dst_task, l4_cap_idx_t src_task, l4_fpage_t snd_fpage, l4_addr_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, unsigned long 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 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_has_child (l4_cap_idx_t task, l4_cap_idx_t cap) L4_NOTHROW`
Test whether a capability has child mappings (in another task).
- `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 rights.
- `l4_msgtag_t l4_task_add_ku_mem (l4_cap_idx_t task, l4_fpage_t ku_mem) L4_NOTHROW`
Add kernel-user memory.

12.105.1 Detailed Description

C interface of the Task kernel object.

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>
```

12.105.2 Enumeration Type Documentation

12.105.2.1 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	Flag to tell the unmap operation to unmap all child mappings including the mapping in the invoked task. See also L4::Task::unmap() l4_task_unmap()
L4_FP_DELETE_OBJ	Flag that indicates that the unmap operation on a capability shall try to delete the corresponding objects immediately. See also L4::Task::unmap() l4_task_unmap()
L4_FP_OTHER_SPACES	Counterpart to L4_FP_ALL_SPACES , unmap only child mappings. See also L4::Task::unmap() l4_task_unmap()

Definition at line 157 of file [consts.h](#).

12.105.3 Function Documentation

12.105.3.1 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>ku_mem</i>	Flexpage describing the virtual area the memory goes to.

Returns

Syscall return tag

Definition at line 436 of file [task.h](#).

12.105.3.2 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 rights.

Parameters

<i>task</i>	Capability selector of the destination task to do the lookup in
<i>cap_a</i>	Capability selector to compare
<i>cap_b</i>	Capability selector to compare

Returns

label contains 1 if equal, 0 if not equal

Definition at line 429 of file [task.h](#).

12.105.3.3 l4_task_cap_has_child()

```
l4_msgtag_t l4_task_cap_has_child (
    l4_cap_idx_t task,
    l4_cap_idx_t cap ) [inline]
```

Test whether a capability has child mappings (in another task).

Parameters

<i>task</i>	Capability selector of the destination task to do the lookup in
<i>cap</i>	Capability selector to look up in the destination task

Returns

label contains 1 if it has at least one child, 0 if not or invalid

Deprecated Do not use. Undetermined future, might be removed.

Definition at line 423 of file [task.h](#).

12.105.3.4 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>	Capability to check for presence..

Return values

<i>tag.label()</i>	> 0 Capability is present (refers to an object).
<i>tag.label()</i>	== 0 No capability present (void object).

A capability is considered present when it refers to an existing kernel object.

Definition at line 417 of file [task.h](#).

12.105.3.5 `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 selector of object to delete

Returns

Syscall return tag

The object will be deleted if the *obj* has sufficient rights. No error will be reported if the rights are insufficient, however, the capability is removed in all cases.

This operation calls `l4_task_unmap()` with `L4_FP_DELETE_OBJ`.

Definition at line 396 of file `task.h`.

12.105.3.6 `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_addr_t snd_base ) [inline]
```

Map resources available in the source task to a destination task.

Parameters

<i>dst_task</i>	Capability selector of destination task
<i>src_task</i>	Capability selector of 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.

Returns

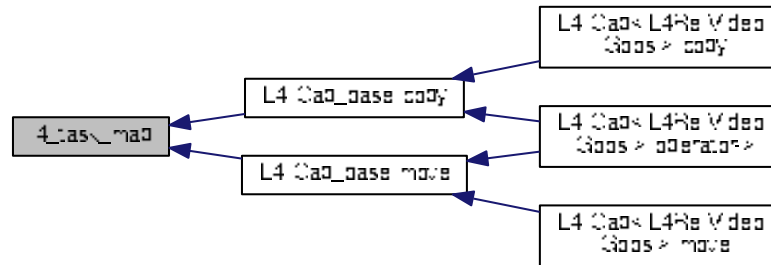
Syscall return tag

This method allows for asynchronous rights delegation from one task to another. It can be used to share memory as well as to delegate access to objects.

Definition at line 366 of file `task.h`.

Referenced by [L4::Cap_base::copy\(\)](#), and [L4::Cap_base::move\(\)](#).

Here is the caller graph for this function:



12.105.3.7 l4_task_release_cap()

```
l4_msgtag_t l4_task_release_cap (
    l4_cap_idx_t task,
    l4_cap_idx_t cap ) [inline]
```

Release capability.

Parameters

<i>task</i>	Capability selector of destination task
<i>cap</i>	Capability selector to release

Returns

Syscall return tag

This operation unmaps the capability from the specified task.

Definition at line 411 of file [task.h](#).

12.105.3.8 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 the address space or object space of the destination task
<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 and from all the tasks that got the rights delegated from that task (i.e., this operation does a recursive rights revocation).

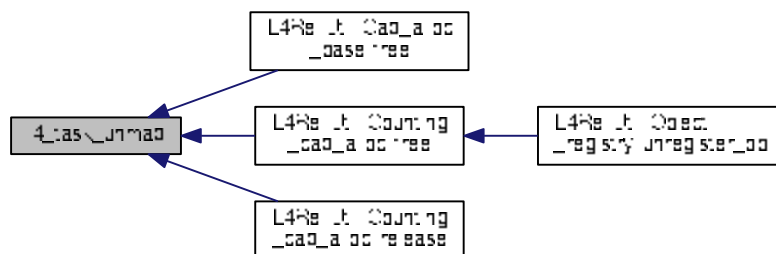
Note

Calling this function on the object space can cause a root capability of an object to be destructed, which destroys the object itself.

Definition at line 373 of file [task.h](#).

Referenced by [L4Re::Util::Cap_alloc_base::free\(\)](#), [L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::free\(\)](#), and [L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::release\(\)](#).

Here is the caller graph for this function:



12.105.3.9 l4_task_unmap_batch()

```

l4_msgtag_t l4_task_unmap_batch (
    l4_cap_idx_t task,
    l4_fpage_t const * fpages,
    unsigned num_fpages,
    unsigned long map_mask ) [inline]
  
```

Revoke rights from a task.

Parameters

<i>task</i>	Capability selector of destination task
<i>fpages</i>	An array of flexpages that describes an area in the address space or object space of the destination task each
<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

This method allows to revoke rights from the destination task and from all the tasks that got the rights delegated from that task (i.e., this operation does a recursive rights revocation).

Precondition

The caller needs to take care that num_fpages is not bigger than L4_UTCB_GENERIC_DATA_SIZE - 2.

Note

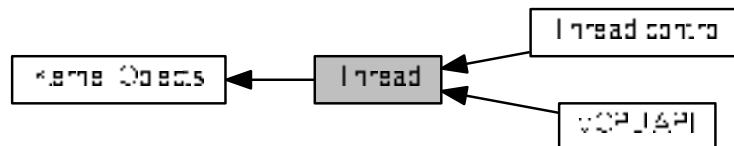
Calling this function on the object space can cause a root capability of an object to be destructed, which destroys the object itself.

Definition at line [380](#) of file [task.h](#).

12.106 Thread

Thread object.

Collaboration diagram for Thread:



Modules

- [Thread control](#)
API for Thread Control method.
- [vCPU API](#)
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_UX_NATIVE](#) = 0x0800000,
[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_EXCEPTION](#) = 0x20000UL }
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_umword_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 or disable 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 or disable 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 or disable 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 or disable 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_arm_set_tpidruro (l4_cap_idx_t thread, l4_addr_t tpidruro) L4_NOTHROW`
Set the TPIDRURO thread specific register.

12.106.1 Detailed Description

Thread object.

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.

Include File

```
#include <l4/sys/thread.h>
```

For the C++ interface refer to [L4::Thread](#).

12.106.2 Enumeration Type Documentation

12.106.2.1 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_UX_NATIVE	Fiasco-UX only: pass-through of host system calls is set.
L4_THREAD_CONTROL_SET_EXC_HANDLER	The exception handler of the thread will be given.

Definition at line [640](#) of file [thread.h](#).

12.106.2.2 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 flex-page for bind.

Definition at line 663 of file [thread.h](#).

12.106.2.3 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.

Definition at line 678 of file [thread.h](#).

12.106.3 Function Documentation

12.106.3.1 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

Definition at line 59 of file [thread.h](#).

12.106.3.2 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 ~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 .

Returns

System call return tag

This method allows to manipulate 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`).

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.

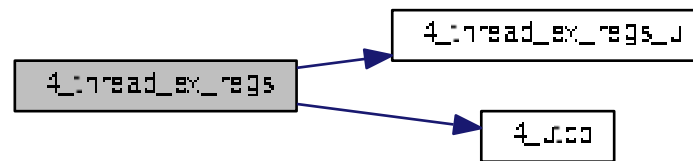
Examples:

[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 829 of file [thread.h](#).

References [l4_thread_ex_regs_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



12.106.3.3 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 ~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.

Returns

System call return tag

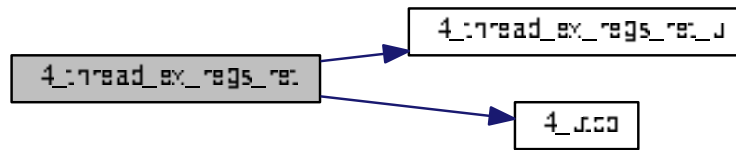
This method allows to manipulate and start a thread. The basic functionality is to set the instruction pointer and the stack pointer of a thread. Additionally, this method allows also to cancel ongoing IPC operations and to force the thread to raise an artificial exception (see *flags*).

Returned values are valid only if function returns successfully.

Definition at line 836 of file [thread.h](#).

References [l4_thread_ex_regs_ret_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



12.106.3.4 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.
in, out	<i>ip</i>	New instruction pointer, use <code>~0UL</code> to leave the instruction pointer unchanged, return previous instruction pointer.
in, out	<i>sp</i>	New stack pointer, use <code>~0UL</code> 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 use for this operation.

Returns

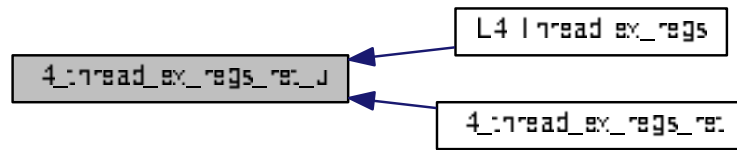
System call return tag. [out] parameters are only valid if the function returns successfully. Use [l4_error\(\)](#) to check.

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`).

Definition at line 702 of file [thread.h](#).

Referenced by [L4::Thread::ex_regs\(\)](#), and [l4_thread_ex_regs_ret\(\)](#).

Here is the caller graph for this function:



12.106.3.5 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 use for this operation.

Returns

System call return tag

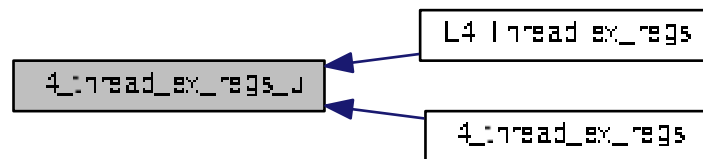
This method allows to manipulate a thread. The basic functionality is to set the instruction pointer and the stack pointer of a thread. Additionally, this method allows also to cancel ongoing IPC operations and to force the thread to raise an artificial exception (see `flags`).

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 691 of file [thread.h](#).

Referenced by [L4::Thread::ex_regs\(\)](#), and [l4_thread_ex_regs\(\)](#).

Here is the caller graph for this function:



12.106.3.6 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) { label = (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 1009 of file [thread.h](#).

12.106.3.7 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.

See also

[l4_thread_modify_sender_start](#)

[l4_thread_modify_sender_add](#)

Definition at line 1020 of file [thread.h](#).

12.106.3.8 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\(\)](#).

See also

[l4_thread_modify_sender_add](#)

[l4_thread_modify_sender_commit](#)

Definition at line 1003 of file [thread.h](#).

12.106.3.9 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.

An example of a deletion event is the removal of an IPC gate that is bound to this thread.

Definition at line 930 of file [thread.h](#).

12.106.3.10 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 898 of file [thread.h](#).

12.106.3.11 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 889 of file [thread.h](#).

12.106.3.12 `l4_thread_vcpu_control()`

```
l4_msgtag_t l4_thread_vcpu_control (
    l4_cap_idx_t thread,
    l4_addr_t vcpu_state ) [inline]
```

Enable or disable the vCPU feature for the thread.

Parameters

<i>thread</i>	Capability selector of the thread for which the vCPU feature shall be enabled or disabled.
<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

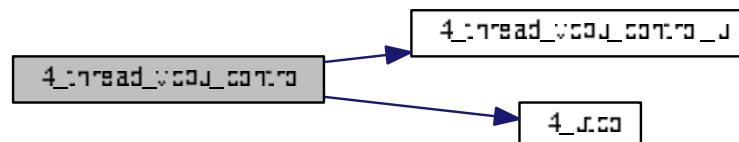
Syscall return tag.

This function enables the vCPU feature of the `thread` if `vcpu_state` is set to a valid kernel-user-memory address, or disables the vCPU feature if `vcpu_state` is 0. (Disable: optional, currently unsupported.)

Definition at line 947 of file [thread.h](#).

References [l4_thread_vcpu_control_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:

12.106.3.13 `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 or disable 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 or disabled.
<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

Systemcall result message 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` if `ext_vcpu_state` is set to a valid kernel-user-memory address, or disables the vCPU feature if `ext_vcpu_state` is 0.

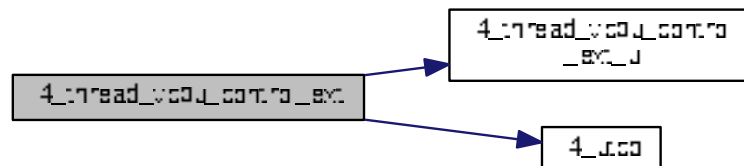
Note

The extended vCPU mode includes the normal vCPU mode.

Definition at line 962 of file [thread.h](#).

References [l4_thread_vcpu_control_ext_u\(\)](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



12.106.3.14 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 or disable 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 or disabled.
<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 use for this operation.

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 `this thread` if `ext_vcpu_state` is set to a valid kernel-user-memory address, or disables the vCPU feature if `ext_vcpu_state` is 0.

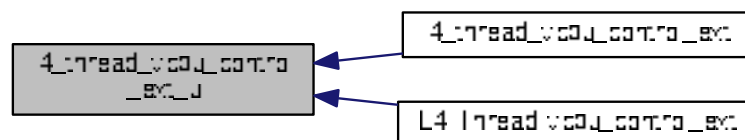
Note

The extended vCPU mode includes the normal vCPU mode.

Definition at line 952 of file [thread.h](#).

Referenced by [l4_thread_vcpu_control_ext\(\)](#), and [L4::Thread::vcpu_control_ext\(\)](#).

Here is the caller graph for this function:



12.106.3.15 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 or disable the vCPU feature for the thread.

Parameters

<i>thread</i>	Capability selector of the thread for which the vCPU feature shall be enabled or disabled.
<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()).
<i>utcb</i>	UTCB to use for this operation.

Returns

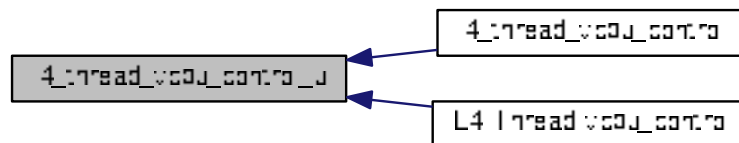
Syscall return tag.

This function enables the vCPU feature of *this* thread if *vcpu_state* is set to a valid kernel-user-memory address, or disables the vCPU feature if *vcpu_state* is 0. (Disable: optional, currently unsupported.)

Definition at line 937 of file [thread.h](#).

Referenced by [l4_thread_vcpu_control\(\)](#), and [L4::Thread::vcpu_control\(\)](#).

Here is the caller graph for this function:



12.106.3.16 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 l4_thread_vcpu_resume_start()

Returns

System call result message tag. In extended vCPU mode and when the virtual interrupts are cleared, the return code 1 flags an incoming IPC message, whereas 0 indicates a VM exit. An error is returned upon:

- Insufficient rights on the given task capability (-L4_EPERM).
- Given task capability is invalid (-L4_ENOENT).
- A supplied mapping failed.

To resume into another address space the capability to the target task must be set in the vCPU-state, with all lower bits in the task capability cleared (see [L4_CAP_MASK](#)). The kernel adds the [L4_SYSF_SEND](#) flag to this field to indicate that the capability has been referenced in the kernel. Consecutive resumes will not reference the task capability again until all bits are cleared again. To release a task use the different task capability or use an invalid capability with the [L4_SYSF_REPLY](#) flag set.

See also

[l4_vcpu_state_t](#)

Definition at line 910 of file [thread.h](#).

12.106.3.17 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-flex-pages using [l4_sndfpage_add\(\)](#).

Definition at line 904 of file [thread.h](#).

12.106.3.18 l4_thread_yield()

```
l4_msgtag_t l4_thread_yield (
    void ) [inline]
```

Yield current time slice.

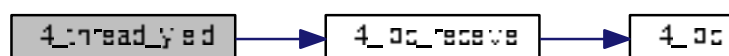
Returns

system call return tag

Definition at line 778 of file [thread.h](#).

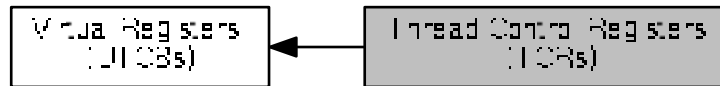
References [L4_INVALID_CAP](#), and [l4_ipc_receive\(\)](#).

Here is the call graph for this function:



12.107 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

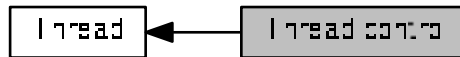
- typedef struct [l4_thread_regs_t](#) [l4_thread_regs_t](#)
Encapsulation of the thread-control-register block of the UTCB.

12.107.1 Detailed Description

12.108 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.
- void [l4_thread_control_ux_host_syscall](#) (int on) [L4_NOTHROW](#)
Enable pass through of native host (Linux) system calls.
- [l4_msgtag_t](#) [l4_thread_control_commit](#) ([l4_cap_idx_t](#) thread) [L4_NOTHROW](#)
Commit the thread control parameters.

12.108.1 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_utcb_t *u = l4_utcb();
l4_thread_control_start(u);
l4_thread_control_pager(u, pager_cap);
l4_thread_control_bind (u, thread_utcb, task);
l4_thread_control_commit(u, thread_cap);
  
```


12.108.2 Function Documentation

12.108.2.1 `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.
-----------	--

Alien mode means the thread is not allowed to invoke [L4](#) kernel objects directly and it is also not allowed to allocate FPU state. All those operations result in an exception IPC that gets sent through the pager capability. The responsible pager can then selectively allow an object invocation or allocate FPU state for the thread.

This feature can be used to attach a debugger to a thread and trace all object invocations.

Examples:

[examples/sys/aliens/main.c](#), and [examples/sys/singlestep/main.c](#).

Definition at line [868](#) of file [thread.h](#).

12.108.2.2 `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 address of the UTCB in the target task.
<i>task</i>	The target task of the thread.

Binding a thread to a task has the effect that the thread afterwards executes code within that task and has access to the resources visible within that task.

Note

There should not be more than one thread use a UTCB 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 862 of file [thread.h](#).

12.108.2.3 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

system call return tag

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 880 of file [thread.h](#).

12.108.2.4 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 855 of file [thread.h](#).

12.108.2.5 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 849 of file [thread.h](#).

12.108.2.6 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\(\)](#)
- [l4_thread_control_ux_host_syscall\(\)](#) (Fiasco-UX only)

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, 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 843 of file [thread.h](#).

12.108.2.7 l4_thread_control_ux_host_syscall()

```
void l4_thread_control_ux_host_syscall (  
    int on ) [inline]
```

Enable pass through of native host (Linux) system calls.

Parameters

<i>on</i>	Boolean value defining the state of the feature.
-----------	--

Precondition

Running on Fiasco-UX

This enables the thread to do host system calls. This feature is only available in Fiasco-UX and ignored in other environments.

Definition at line [874](#) of file [thread.h](#).

12.109 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, init
- #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

Typedefs

- typedef struct [l4_timeout_s](#) [l4_timeout_s](#)
Basic timeout specification.
- typedef union [l4_timeout_t](#) [l4_timeout_t](#)
Timeout pair.

Enumerations

- enum [l4_timeout_abs_validity](#)
Intervals of validity for absolute timeouts
Times are actually 2^x values (e.g.

Functions

- [l4_timeout_s l4_timeout_rel](#) (unsigned man, unsigned exp) [L4_NOTHROW](#)
Get relative timeout consisting of mantissa and exponent.
- [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_timeout_t l4_timeout](#) ([l4_timeout_s](#) snd, [l4_timeout_s](#) rcv) [L4_NOTHROW](#)
Combine send and receive timeout in a timeout.
- void [l4_snd_timeout](#) ([l4_timeout_s](#) snd, [l4_timeout_t](#) *to) [L4_NOTHROW](#)
Set send timeout in given to timeout.
- void [l4_rcv_timeout](#) ([l4_timeout_s](#) rcv, [l4_timeout_t](#) *to) [L4_NOTHROW](#)
Set receive timeout in given to timeout.
- [l4_kernel_clock_t l4_timeout_rel_get](#) ([l4_timeout_s](#) to) [L4_NOTHROW](#)
Get clock value of out timeout.
- unsigned [l4_timeout_is_absolute](#) ([l4_timeout_s](#) to) [L4_NOTHROW](#)
Return whether the given timeout is absolute or not.
- [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.

12.109.1 Detailed Description

All kinds of timeouts and time related functions.

12.109.2 Macro Definition Documentation

12.109.2.1 L4_IPC_TIMEOUT_0

```
#define L4_IPC_TIMEOUT_0 ((l4\_timeout\_s) {0x0400})
```

Timeout constants.

0 timeout

Definition at line 77 of file [__timeout.h](#).

Referenced by [L4::lpc_svr::Timeout_queue_hooks< Br_manager_timeout_hooks, Br_manager >::timeout\(\)](#).

12.109.3 Typedef Documentation

12.109.3.1 l4_timeout_s

```
typedef struct l4_timeout_s l4_timeout_s
```

Basic timeout specification.

Basically a floating point number with 10 bits mantissa and 5 bits exponent ($t = m \cdot 2^e$).

The timeout can also specify an absolute point in time (bit 16 == 1).

12.109.3.2 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.

12.109.4 Enumeration Type Documentation

12.109.4.1 l4_timeout_abs_validity

```
enum l4_timeout_abs_validity
```

Intervals of validity for absolute timeouts

Times are actually 2^x values (e.g.

2ms -> 2048 μ s)

Definition at line 92 of file [__timeout.h](#).

12.109.5 Function Documentation

12.109.5.1 l4_ipc_timeout()

```
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 210 of file [__timeout.h](#).

References [l4_timeout_t::p](#), [l4_timeout_t::rcv](#), [l4_timeout_t::snd](#), and [l4_timeout_s::t](#).

12.109.5.2 l4_rcv_timeout()

```
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
------------	-----------------

Return values

<i>to</i>	L4 timeout
-----------	----------------------------

Definition at line 238 of file [__timeout.h](#).

12.109.5.3 l4_snd_timeout()

```
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
------------	--------------

Return values

<i>to</i>	L4 timeout
-----------	----------------------------

Definition at line 231 of file [__timeout.h](#).

12.109.5.4 l4_timeout()

```
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

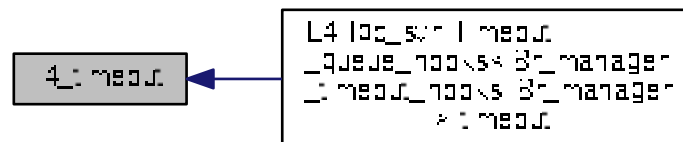
[L4](#) timeout

Definition at line 221 of file [__timeout.h](#).

References [l4_timeout_t::p](#), [l4_timeout_t::rcv](#), [l4_timeout_t::snd](#), and [l4_timeout_s::t](#).

Referenced by [L4::lpc_svr::Timeout_queue_hooks< Br_manager_timeout_hooks, Br_manager >::timeout\(\)](#).

Here is the caller graph for this function:



12.109.5.5 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.

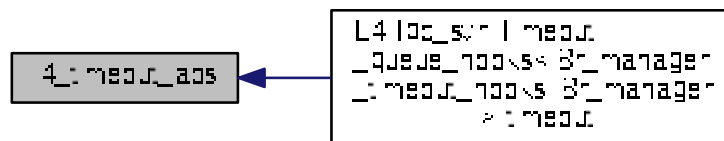
Returns

timeout value

Definition at line 383 of file [utcb.h](#).

Referenced by [L4::lpc_svr::Timeout_queue_hooks< Br_manager_timeout_hooks, Br_manager >::timeout\(\)](#).

Here is the caller graph for this function:



12.109.5.6 l4_timeout_get()

```
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

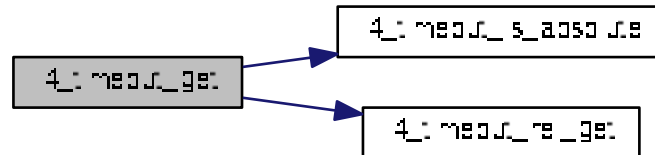
Returns

Clock sum

Definition at line 268 of file [__timeout.h](#).

References [l4_timeout_is_absolute\(\)](#), and [l4_timeout_rel_get\(\)](#).

Here is the call graph for this function:



12.109.5.7 l4_timeout_is_absolute()

```
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 261 of file [__timeout.h](#).

References [l4_timeout_s::t](#).

Referenced by [l4_timeout_get\(\)](#).

Here is the caller graph for this function:



12.109.5.8 l4_timeout_rel()

```
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 245 of file [__timeout.h](#).

12.109.5.9 l4_timeout_rel_get()

```
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 252 of file [__timeout.h](#).

References [l4_timeout_s::t](#).

Referenced by [l4_timeout_get\(\)](#).

Here is the caller graph for this function:



12.109.5.10 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

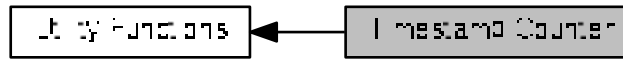
Index to 64bit message register alias

Definition at line [386](#) of file [utcb.h](#).

References [__END_DECLS](#).

12.110 Timestamp Counter

Collaboration diagram for Timestamp Counter:



Files

- file [rdtsc.h](#)
time stamp counter related functions
- file [rdtsc.h](#)
time stamp counter related functions

Macros

- `#define L4_TSC_INIT_AUTO 0`
Automatic init.
- `#define L4_TSC_INIT_KERNEL 1`
Initialized by kernel.
- `#define L4_TSC_INIT_CALIBRATE 2`
Initialized by user-level.
- `#define L4_TSC_INIT_AUTO 0`
Automatic init.
- `#define L4_TSC_INIT_KERNEL 1`
Initialized by kernel.
- `#define L4_TSC_INIT_CALIBRATE 2`
Initialized by user-level.

Functions

- `l4_cpu_time_t l4_rdtsc (void)`
Read current value of CPU-internal time stamp counter.
- `l4_uint32_t l4_rdtsc_32 (void)`
Read the lest significant 32 bit of the TSC.
- `l4_cpu_time_t l4_rdpmc (int nr)`
Return current value of CPU-internal performance measurement counter.
- `l4_uint32_t l4_rdpmc_32 (int nr)`
Return the least significant 32 bit of a performance counter.
- `l4_uint64_t l4_tsc_to_ns (l4_cpu_time_t tsc)`
Convert time stamp to ns value.
- `l4_uint64_t l4_tsc_to_us (l4_cpu_time_t tsc)`

Convert time stamp 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 time stamp 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](#) *kip)

Calibrate scalars for time stamp calculations.

- [l4_uint32_t](#) [l4_tsc_init](#) (int constraint, [l4_kernel_info_t](#) *kip)

Initialitze scaler for TSC calicatlions.

- [l4_uint32_t](#) [l4_get_hz](#) (void)

Get CPU frequency in Hz.

12.110.1 Detailed Description

12.110.2 Function Documentation

12.110.2.1 [l4_busy_wait_ns\(\)](#)

```
void l4_busy_wait_ns (
    l4\_uint64\_t ns ) [inline]
```

Wait busy for a small amount of time.

Parameters

ns	nano seconds to wait
--------------------	----------------------

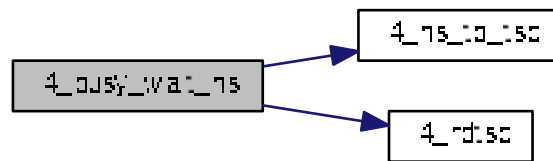
Attention

Not intendet for any use!

Definition at line [317](#) of file [rdtsc.h](#).

References [l4_ns_to_tsc\(\)](#), and [l4_rdtsc\(\)](#).

Here is the call graph for this function:



12.110.2.2 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
-----------	-----------------------

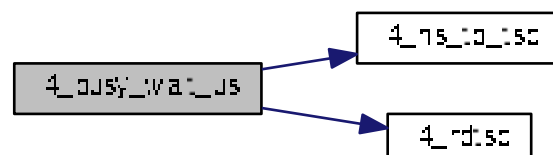
Attention

Not intendet for any use!

Definition at line 327 of file `rdtsc.h`.

References [EXTERN_C_END](#), [l4_ns_to_tsc\(\)](#), and [l4_rdtsc\(\)](#).

Here is the call graph for this function:



12.110.2.3 l4_calibrate_tsc()

```
l4_uint32_t l4_calibrate_tsc (
    l4_kernel_info_t * kip ) [inline]
```

Calibrate scalers for time stamp calculations.

Determine some scalers to be able to convert between real time and CPU ticks. This test uses channel 0 of the PIT (i8254) or the kernel KIP, depending on availability. Just calls l4_tsc_init(L4_TSC_INIT_AUTO).

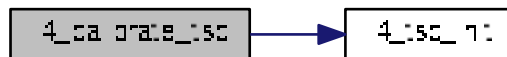
Examples:

[examples/sys/aliens/main.c](#).

Definition at line 179 of file [rdtsc.h](#).

References [l4_tsc_init\(\)](#), and [L4_TSC_INIT_AUTO](#).

Here is the call graph for this function:



12.110.2.4 l4_get_hz()

```
l4_uint32_t l4_get_hz (
    void )
```

Get CPU frequency in Hz.

Returns

frequency in Hz

12.110.2.5 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
-----------	--------------

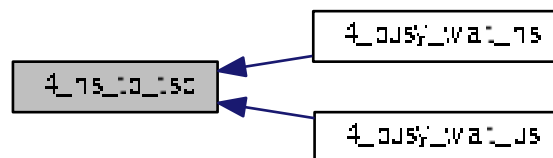
Returns

CPU ticks

Definition at line 303 of file [rdtsc.h](#).

Referenced by [l4_busy_wait_ns\(\)](#), and [l4_busy_wait_us\(\)](#).

Here is the caller graph for this function:

**12.110.2.6 l4_rdpmc()**

```
l4_cpu_time_t l4_rdpmc (
    int nr ) [inline]
```

Return current value of CPU-internal performance measurement counter.

Parameters

<i>nr</i>	Number of counter (0 or 1)
-----------	----------------------------

Returns

64-bit PMC

Definition at line 205 of file [rdtsc.h](#).

12.110.2.7 l4_rdpmc_32()

```
l4_uint32_t l4_rdpmc_32 (
    int nr ) [inline]
```

Return the least significant 32 bit of a performance counter.

Useful for smaller differences, needs less cycles.

Definition at line 227 of file [rdtsc.h](#).

12.110.2.8 l4_rdtsc()

```
l4_cpu_time_t l4_rdtsc (
    void ) [inline]
```

Read current value of CPU-internal time stamp counter.

Returns

64-bit time stamp

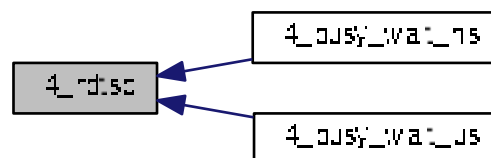
Examples:

[examples/sys/aliens/main.c](#).

Definition at line 185 of file [rdtsc.h](#).

Referenced by [l4_busy_wait_ns\(\)](#), and [l4_busy_wait_us\(\)](#).

Here is the caller graph for this function:



12.110.2.9 `l4_rdtsc_32()`

```
l4_uint32_t l4_rdtsc_32 (
    void ) [inline]
```

Read the lest significant 32 bit of the TSC.

Useful for smaller differences, needs less cycles.

Definition at line 246 of file `rdtsc.h`.

12.110.2.10 `l4_tsc_init()`

```
l4_uint32_t l4_tsc_init (
    int constraint,
    l4_kernel_info_t * kip )
```

Initialitze scaler for TSC calicaltions.

Initialize the scalers needed by `l4_tsc_to_ns()`/`l4_ns_to_tsc()` and so on. Current versions of Fiasco export these scalers from kernel into userland. The programmer may decide whether he allows to use these scalers or if an calibration should be performed.

Parameters

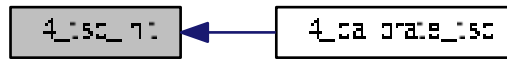
<i>constraint</i>	<p>programmers constraint:</p> <ul style="list-style-type: none"> • L4_TSC_INIT_AUTO if the kernel exports the scalers then use them. If not, perform calibration using channel 0 of the PIT (i8254). The latter case may lead into short (unpredictable) periods where interrupts are disabled. • L4_TSC_INIT_KERNEL depend on retrieving the scalers from kernel. If the scalers are not available, return 0. • L4_TSC_INIT_CALIBRATE Ignore possible scalers exported by the scaler, instead insist on calibration using the PIT.
<i>kip</i>	KIP pointer

Returns

0 on error (no scalers exported by kernel, calibrating failed ...) 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.

Referenced by [l4_calibrate_tsc\(\)](#).

Here is the caller graph for this function:



12.110.2.11 l4_tsc_to_ns()

```
l4_uint64_t l4_tsc_to_ns (
    l4_cpu_time_t tsc ) [inline]
```

Convert time stamp to ns value.

Parameters

<code>tsc</code>	time value in CPU ticks
------------------	-------------------------

Returns

time value in ns

Examples:

[examples/sys/aliens/main.c](#).

Definition at line 260 of file [rdtsc.h](#).

12.110.2.12 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 time stamp to s.ns value.

Parameters

<code>tsc</code>	time value in CPU ticks
------------------	-------------------------

Return values

<i>s</i>	seconds
<i>ns</i>	nano seconds

Definition at line 288 of file [rdtsc.h](#).

12.110.2.13 l4_tsc_to_us()

```
l4_uint64_t l4_tsc_to_us (
    l4_cpu_time_t tsc ) [inline]
```

Convert time stamp into micro seconds value.

Parameters

<i>tsc</i>	time value in CPU ticks
------------	-------------------------

Returns

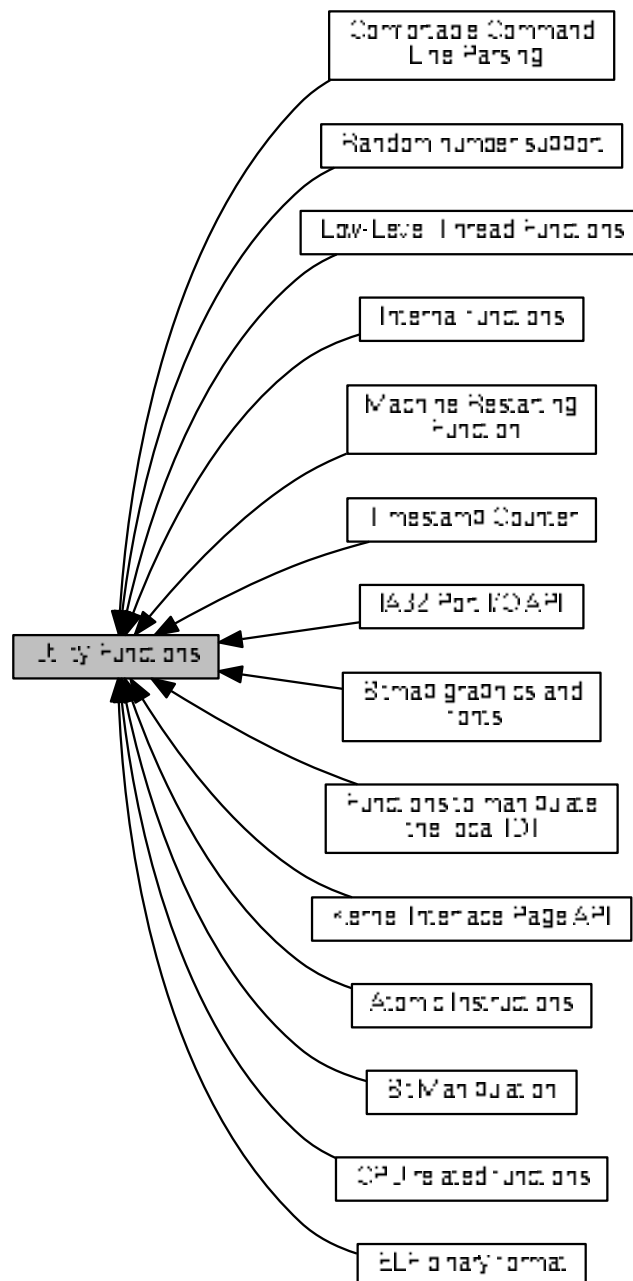
time value in micro seconds

Definition at line 274 of file [rdtsc.h](#).

12.111 Utility Functions

Utilities, generic file.

Collaboration diagram for Utility Functions:



Modules

- [Atomic Instructions](#)

- [Bit Manipulation](#)
- [Bitmap graphics and fonts](#)

This library provides some functions for bitmap handling in frame buffers.

- [CPU related functions](#)
- [Comfortable Command Line Parsing](#)
- [ELF binary format](#)

Functions and types related to ELF binaries.

- [Functions to manipulate the local IDT](#)
- [IA32 Port I/O API](#)
- [Internal functions](#)
- [Kernel Interface Page API](#)
- [Low-Level Thread Functions](#)
- [Machine Restarting Function](#)
- [Random number support](#)
- [Timestamp Counter](#)

Files

- file [rand.h](#)

Simple Pseudo-Random Number Generator.

Functions

- void [l4_sleep_forever](#) (void) [L4_NOTHROW](#)
Go sleep and never wake up.
- 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_timeout_s](#) [l4util_micros2l4to](#) (unsigned int mus) [L4_NOTHROW](#)
Calculate l4 timeouts.
- void [l4_sleep](#) (int ms) [L4_NOTHROW](#)
Suspend thread for a period of ms milliseconds.
- void [l4_usleep](#) (int us) [L4_NOTHROW](#)
Suspend thread for a period of us microseconds.
- 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)

12.111.1 Detailed Description

Utilities, generic file.

12.111.2 Function Documentation

12.111.2.1 [l4_sleep\(\)](#)

```
void l4_sleep (
    int ms )
```

Suspend thread for a period of *ms* milliseconds.

Parameters

<i>ms</i>	Time in milliseconds
-----------	----------------------

12.111.2.2 `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:

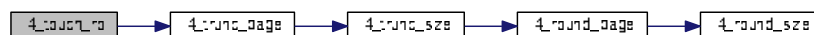
[examples/sys/singlestep/main.c](#).

Definition at line 94 of file [util.h](#).

References [L4_PAGESIZE](#), and [l4_trunc_page\(\)](#).

Referenced by [l4_sleep_forever\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



12.111.2.3 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:

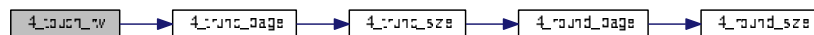
[examples/sys/aliens/main.c](#), and [examples/sys/singlestep/main.c](#).

Definition at line 107 of file [util.h](#).

References [EXTERN_C_END](#), [L4_PAGESIZE](#), and [l4_trunc_page\(\)](#).

Referenced by [l4_sleep_forever\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



12.111.2.4 l4_usleep()

```
void l4_usleep (
    int us )
```

Suspend thread for a period of *us* microseconds.

Parameters

<i>us</i>	Time in microseconds
-----------	----------------------

WARNING: This function is mostly bogus since the timer resolution of current [L4](#) implementations is about 1ms!

12.111.2.5 `l4util_micros2l4to()`

```
l4_timeout_s l4util_micros2l4to (
    unsigned int mus )
```

Calculate l4 timeouts.

Parameters

<i>mus</i>	time in microseconds. Special cases: <ul style="list-style-type: none"> • 0 -> timeout 0 • ~0U -> timeout NEVER
------------	---

Returns

the corresponding l4_timeout value

12.111.2.6 `l4util_splitlog2_hdl()`

```
long l4util_splitlog2_hdl (
    l4_addr_t start,
    l4_addr_t end,
    long(*) (l4_addr_t s, l4_addr_t e, int log2size) handler ) [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.

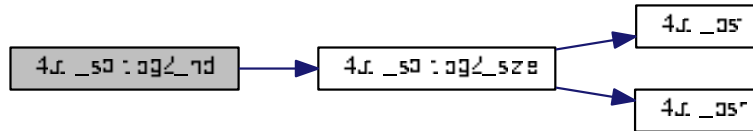
Returns

0 on success, != 0 otherwise

Definition at line [53](#) of file [splitlog2.h](#).

References [L4_EINVAL](#), and [l4util_splitlog2_size\(\)](#).

Here is the call graph for this function:



12.111.2.7 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

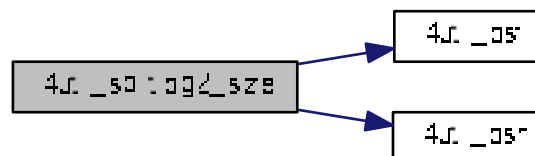
length of elements in log2size (length is $1 < \log_2 \text{size}$)

Definition at line 72 of file [splitlog2.h](#).

References [l4util_bsf\(\)](#), and [l4util_bsr\(\)](#).

Referenced by [l4util_splitlog2_hdl\(\)](#).

Here is the call graph for this function:



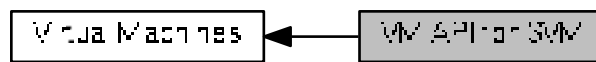
Here is the caller graph for this function:



12.112 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.

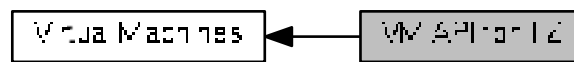
12.112.1 Detailed Description

Virtual machine API for SVM.

12.113 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

12.113.1 Detailed Description

Virtual Machine API for ARM TrustZone.

12.114 VM API for VMX

Virtual machine API for VMX.

Collaboration diagram for VM API for VMX:



Enumerations

- enum `L4_vm_vmx_caps_regs` {
`L4_VM_VMX_BASIC_REG` = 0, `L4_VM_VMX_TRUE_PINBASED_CTLTS_REG` = 1, `L4_VM_VMX_TRUE_PROCBASED_CTLTS_REG` = 2, `L4_VM_VMX_TRUE_EXIT_CTLTS_REG` = 3,
`L4_VM_VMX_TRUE_ENTRY_CTLTS_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` = 0xa, `L4_VM_VMX_PROCBASED_CTLTS2_REG` = 0xb,
`L4_VM_VMX_EPT_VPID_CAP_REG` = 0xc, `L4_VM_VMX_NUM_CAPS_REGS` }
Exported VMX capability registers.
- enum `L4_vm_vmx_dfl1_regs` {
`L4_VM_VMX_PINBASED_CTLTS_DFL1_REG` = 0x1, `L4_VM_VMX_PROCBASED_CTLTS_DFL1_REG` = 0x2, `L4_VM_VMX_EXIT_CTLTS_DFL1_REG` = 0x3, `L4_VM_VMX_ENTRY_CTLTS_DFL1_REG` = 0x4,
`L4_VM_VMX_NUM_DFL1_REGS` }
Exported VMX capability registers (default to 1 bits).
- enum {
`L4_VM_VMX_VMCS_CR2` = 0x683e, `L4_VM_VMX_VMCS_XCR0` = 0x2840, `L4_VM_VMX_VMCS_MSR_SYSCALL_MASK` = 0x2842, `L4_VM_VMX_VMCS_MSR_LSTAR` = 0x2844,
`L4_VM_VMX_VMCS_MSR_CSTAR` = 0x2846, `L4_VM_VMX_VMCS_MSR_TSC_AUX` = 0x2848, `L4_VM_VMX_VMCS_MSR_STAR` = 0x284a, `L4_VM_VMX_VMCS_MSR_KERNEL_GS_BASE` = 0x284c }
Additional (virtual) VMCS fields.

Functions

- `l4_uint64_t l4_vm_vmx_get_caps` (void const *vcpu_state, unsigned cap_msr) `L4_NOTHROW`
Get a capability register for VMX.
- `l4_uint32_t l4_vm_vmx_get_caps_default1` (void const *vcpu_state, unsigned cap_msr) `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` (void *vmcs, void *user_vmcs) `L4_NOTHROW`
Saves cached state from the kernel VMCS to the user VMCS.
- void `l4_vm_vmx_ptr_load` (void *vmcs, void *user_vmcs) `L4_NOTHROW`

- Loads the user_vmcs as the current VMCS.*

 - [l4_uint32_t l4_vm_vmx_get_cr2_index](#) (void const *vmcs) [L4_NOTHROW](#)
Get the VMCS field index of the virtual CR2 register.
 - [l4_umword_t l4_vm_vmx_read_nat](#) (void *vmcs, unsigned field) [L4_NOTHROW](#)
Read a natural width VMCS field.
 - [l4_uint16_t l4_vm_vmx_read_16](#) (void *vmcs, unsigned field) [L4_NOTHROW](#)
Read a 16bit VMCS field.
 - [l4_uint32_t l4_vm_vmx_read_32](#) (void *vmcs, unsigned field) [L4_NOTHROW](#)
Read a 32bit VMCS field.
 - [l4_uint64_t l4_vm_vmx_read_64](#) (void *vmcs, unsigned field) [L4_NOTHROW](#)
Read a 64bit VMCS field.
 - [l4_uint64_t l4_vm_vmx_read](#) (void *vmcs, unsigned field) [L4_NOTHROW](#)
Read any VMCS field.
 - void [l4_vm_vmx_write_nat](#) (void *vmcs, unsigned field, [l4_umword_t](#) val) [L4_NOTHROW](#)
Write to a natural width VMCS field.
 - void [l4_vm_vmx_write_16](#) (void *vmcs, unsigned field, [l4_uint16_t](#) val) [L4_NOTHROW](#)
Write to a 16bit VMCS field.
 - void [l4_vm_vmx_write_32](#) (void *vmcs, unsigned field, [l4_uint32_t](#) val) [L4_NOTHROW](#)
Write to a 32bit VMCS field.
 - void [l4_vm_vmx_write_64](#) (void *vmcs, unsigned field, [l4_uint64_t](#) val) [L4_NOTHROW](#)
Write to a 64bit VMCS field.
 - void [l4_vm_vmx_write](#) (void *vmcs, unsigned field, [l4_uint64_t](#) val) [L4_NOTHROW](#)
Write to an arbitrary VMCS field.

12.114.1 Detailed Description

Virtual machine API for VMX.

12.114.2 Enumeration Type Documentation

12.114.2.1 anonymous enum

anonymous enum

Additional (virtual) 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 version of the VMCS.

Enumerator

L4_VM_VMX_VMCS_CR2	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_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 105 of file [__vm-vmx.h](#).

12.114.2.2 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_CTLN_REG	True pin-based control caps.
L4_VM_VMX_TRUE_PROCBASED_CTLN_REG	True processor based control caps.
L4_VM_VMX_TRUE_EXIT_CTLN_REG	True exit control caps.
L4_VM_VMX_TRUE_ENTRY_CTLN_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_CTLN2_REG	Processor based control 2 caps.
L4_VM_VMX_EPT_VPID_CAP_REG	EPT and VPID caps.
L4_VM_VMX_NUM_CAPS_REGS	Total number of VMX capability registers.

Definition at line 39 of file [__vm-vmx.h](#).

12.114.2.3 L4_vm_vmx_dfl1_regs

enum [L4_vm_vmx_dfl1_regs](#)

Exported VMX capability registers (default to 1 bits).

Enumerator

L4_VM_VMX_PINBASED_CTL5_DFL1_REG	Default 1 bits in pin-based controls.
L4_VM_VMX_PROCBASED_CTL5_DFL1_REG	Default 1 bits in processor-based controls.
L4_VM_VMX_EXIT_CTL5_DFL1_REG	Default 1 bits in exit controls.
L4_VM_VMX_ENTRY_CTL5_DFL1_REG	Default 1 bits in entry controls.
L4_VM_VMX_NUM_DFL1_REGS	Total number of default on registers.

Definition at line 62 of file [__vm-vmx.h](#).

12.114.3 Function Documentation

12.114.3.1 l4_vm_vmx_clear()

```
void l4_vm_vmx_clear (
    void * vmcs,
    void * user_vmcs ) [inline]
```

Saves cached state from the kernel VMCS to the user VMCS.

Parameters

<i>vmcs</i>	Pointer to the kernel VMCS.
<i>user_vmcs</i>	Pointer to the user VMCS.

This function is comparable to VMX vmclear.

Definition at line 433 of file [__vm-vmx.h](#).

Referenced by [l4_vm_vmx_ptr_load\(\)](#).

Here is the caller graph for this function:



12.114.3.2 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 357 of file [__vm-vmx.h](#).

References [L4_NOTHROW](#), and [l4_vm_vmx_field_order\(\)](#).

Here is the call graph for this function:



12.114.3.3 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 342 of file [__vm-vmx.h](#).

Referenced by [l4_vm_vmx_field_len\(\)](#).

Here is the caller graph for this function:



12.114.3.4 l4_vm_vmx_get_caps()

```

l4_uint64_t l4_vm_vmx_get_caps (
    void const * vcpu_state,
    unsigned cap_msr ) [inline]
  
```

Get a capability register for VMX.

Parameters

<i>vcpu_state</i>	Pointer to the VCPU state of the VCPU.
<i>cap_msr</i>	Caps register index (see L4_vm_vmx_caps_regs).

Returns

The value of the capability register.

Definition at line 529 of file [__vm-vmx.h](#).

References [L4_VCPU_OFFSET_EXT_INFOS](#).

12.114.3.5 l4_vm_vmx_get_caps_default1()

```

l4_uint32_t l4_vm_vmx_get_caps_default1 (
    void const * vcpu_state,
    unsigned cap_msr ) [inline]
  
```

Get a default to one capability register for VMX.

Parameters

<i>vcpu_state</i>	Pointer to the VCPU state of the VCPU.
<i>cap_msr</i>	Default 1 caps register index (see L4_vm_vmx_dfl1_regs).

Returns

The value of the capability register.

Definition at line 537 of file [__vm-vmx.h](#).

References [L4_VCPU_OFFSET_EXT_INFOS](#), [L4_VM_VMX_NUM_CAPS_REGS](#), and [L4_VM_VMX_PINBASE_D_CTL5_DFL1_REG](#).

12.114.3.6 l4_vm_vmx_get_cr2_index()

```
l4_uint32_t l4_vm_vmx_get_cr2_index (
    void const * vmcs ) [inline]
```

Get the 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 version of the VMCS.

See also

[L4_VM_VMX_VMCS_CR2](#)

Definition at line 545 of file [__vm-vmx.h](#).

12.114.3.7 l4_vm_vmx_ptr_load()

```
void l4_vm_vmx_ptr_load (
    void * vmcs,
    void * user_vmcs ) [inline]
```

Loads the user_vmcs as the current VMCS.

Parameters

<i>vmcs</i>	Pointer to the kernel VMCS.
<i>user_vmcs</i>	Pointer to the user VMCS.

This function is comparable to VMX `vmptld`.

Definition at line 445 of file `__vm-vmx.h`.

References [l4_vm_vmx_clear\(\)](#).

Here is the call graph for this function:



12.114.3.8 `l4_vm_vmx_read()`

```
l4_uint64_t l4_vm_vmx_read (
    void * vmcs,
    unsigned field ) [inline]
```

Read any VMCS field.

Parameters

<i>vmcs</i>	Pointer to the software VMCS.
<i>field</i>	The VMCS field index as used on VMX hardware.

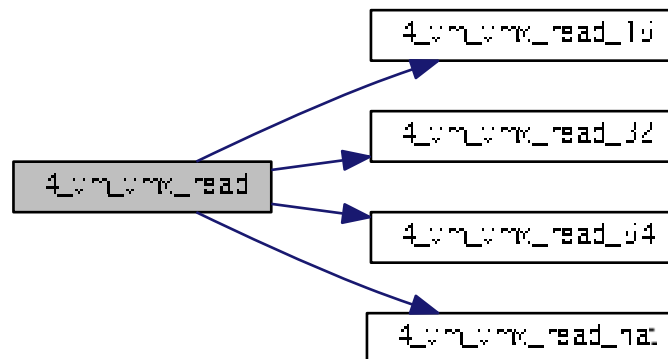
Returns

The value of the VMCS field with the given index.

Definition at line 481 of file `__vm-vmx.h`.

References [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:



12.114.3.9 l4_vm_vmx_read_16()

```

l4_uint16_t l4_vm_vmx_read_16 (
    void * vmcs,
    unsigned field ) [inline]
  
```

Read a 16bit 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 VMCS field with the given index.

Definition at line 466 of file [__vm-vmx.h](#).

Referenced by [l4_vm_vmx_read\(\)](#).

Here is the caller graph for this function:



12.114.3.10 `l4_vm_vmx_read_32()`

```
l4_uint32_t l4_vm_vmx_read_32 (  
    void * vmcs,  
    unsigned field ) [inline]
```

Read a 32bit 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 VMCS field with the given index.

Definition at line 471 of file [__vm-vmx.h](#).

Referenced by [l4_vm_vmx_read\(\)](#).

Here is the caller graph for this function:



12.114.3.11 `l4_vm_vmx_read_64()`

```
l4_uint64_t l4_vm_vmx_read_64 (  
    void * vmcs,  
    unsigned field ) [inline]
```

Read a 64bit VMCS field.

Parameters

<i>vmcs</i>	Pointer to the software VMCS.
<i>field</i>	The VMCS field index as used on VMX hardware.

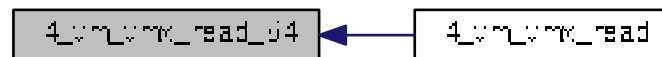
Returns

The value of the VMCS field with the given index.

Definition at line 476 of file [__vm-vmx.h](#).

Referenced by [l4_vm_vmx_read\(\)](#).

Here is the caller graph for this function:

**12.114.3.12 l4_vm_vmx_read_nat()**

```

l4_umword_t l4_vm_vmx_read_nat (
    void * vmcs,
    unsigned field ) [inline]
  
```

Read a natural width 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 VMCS field with the given index.

Definition at line 461 of file [__vm-vmx.h](#).

Referenced by [l4_vm_vmx_read\(\)](#).

Here is the caller graph for this function:



12.114.3.13 `l4_vm_vmx_write()`

```
void l4_vm_vmx_write (
    void * vmcs,
    unsigned field,
    l4_uint64_t val ) [inline]
```

Write to an arbitrary VMCS field.

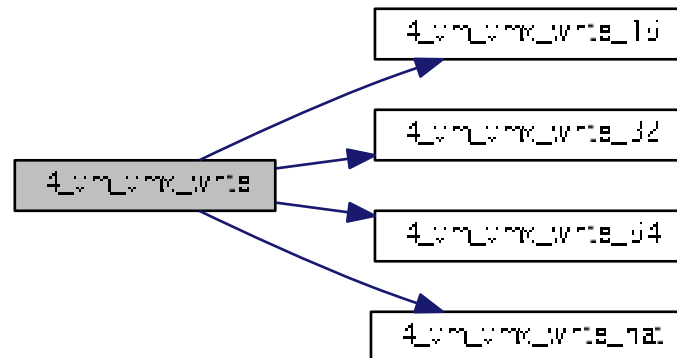
Parameters

<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 516 of file `__vm-vmx.h`.

References `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:

12.114.3.14 `l4_vm_vmx_write_16()`

```
void l4_vm_vmx_write_16 (
    void * vmcs,
    unsigned field,
    l4_uint16_t val ) [inline]
```

Write to a 16bit 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 500 of file [__vm-vmx.h](#).

Referenced by [l4_vm_vmx_write\(\)](#).

Here is the caller graph for this function:

12.114.3.15 `l4_vm_vmx_write_32()`

```
void l4_vm_vmx_write_32 (
    void * vmcs,
    unsigned field,
    l4_uint32_t val ) [inline]
```

Write to a 32bit VMCS field.

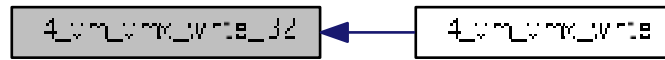
Parameters

<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 505 of file [__vm-vmx.h](#).

Referenced by [l4_vm_vmx_write\(\)](#).

Here is the caller graph for this function:



12.114.3.16 l4_vm_vmx_write_64()

```
void l4_vm_vmx_write_64 (
    void * vmcs,
    unsigned field,
    l4_uint64_t val ) [inline]
```

Write to a 64bit 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 510 of file [__vm-vmx.h](#).

Referenced by [l4_vm_vmx_write\(\)](#).

Here is the caller graph for this function:



12.114.3.17 l4_vm_vmx_write_nat()

```
void l4_vm_vmx_write_nat (
    void * vmcs,
```

```
    unsigned field,  
    l4_umword_t val ) [inline]
```

Write to a natural width 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 495 of file [__vm-vmx.h](#).

Referenced by [l4_vm_vmx_write\(\)](#).

Here is the caller graph for this function:



12.115 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 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_above](#) = 0x01000, [F_l4re_video_view_flags_mask](#) = 0xff000 }
Flags of information on a view.

Functions

- `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.
- `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.

12.115.1 Detailed Description

12.115.2 Typedef Documentation

12.115.2.1 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.

12.115.3 Enumeration Type Documentation

12.115.3.1 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 39 of file [goos.h](#).

12.115.3.2 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 33 of file [view.h](#).

12.115.4 Function Documentation

12.115.4.1 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

12.115.4.2 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.
-------------	--------------------------

Return values

<i>view</i>	the structure will be initialized for the new view.
-------------	---

12.115.4.3 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())

12.115.4.4 `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.

12.115.4.5 `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.

12.115.4.6 `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.

Return values

<i>view</i>	the structure will be 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\(\)](#).

12.115.4.7 l4re_video_goos_info()

```
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
-------------	-------------

Return values

<i>ginfo</i>	Pointer to goos information structure.
--------------	--

Returns

0 for success, <0 on error

- [-L4_ENODEV](#)
- IPC errors

12.115.4.8 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

12.115.4.9 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.
-------------	------------------------------------

Return values

<i>info</i>	a buffer receiving the information about the view.
-------------	--

12.115.4.10 l4re_video_view_refresh()

```
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

12.115.4.11 `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.

12.115.4.12 `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.

12.115.4.13 `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.

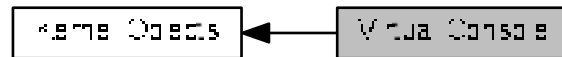
Parameters

<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.

12.116 Virtual Console

Virtual console for simple character based input and output.

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.

12.116.1 Detailed Description

Virtual console for simple character based input and output.

The interrupt for read events is provided by the virtual key interrupt.

Include File

```
#include <l4/sys/vcon.h>
```

See [L4::Vcon](#) for the C++ interface.

12.116.2 Enumeration Type Documentation

12.116.2.1 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 189 of file [vcon.h](#).

12.116.2.2 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 211 of file [vcon.h](#).

12.116.2.3 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 200 of file [vcon.h](#).

12.116.2.4 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](#).

12.116.3 Function Documentation

12.116.3.1 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.

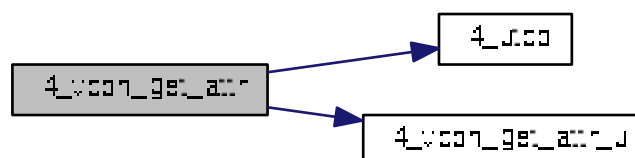
Returns

Syscall return tag

Definition at line 409 of file [vcon.h](#).

References [l4_utcb\(\)](#), and [l4_vcon_get_attr_u\(\)](#).

Here is the call graph for this function:



12.116.3.2 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 successfull call of this function.
	<i>utcb</i>	UTCB pointer of the calling thread.

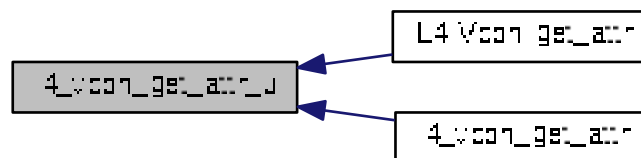
Returns

Syscall return tag.

Definition at line 391 of file [vcon.h](#).

Referenced by [L4::Vcon::get_attr\(\)](#), and [l4_vcon_get_attr\(\)](#).

Here is the caller graph for this function:



12.116.3.3 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><0</code>	Error code.
<code>>size</code>	If more bytes are to read, <code>size</code> bytes are in the buffer <code>buf</code> .
<code><=size</code>	Number of bytes read.

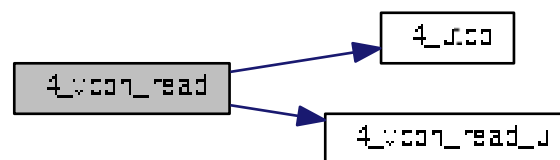
Note

Size must not exceed [L4_VCON_READ_SIZE](#).

Definition at line [365](#) of file [vcon.h](#).

References [l4_utcb\(\)](#), and [l4_vcon_read_u\(\)](#).

Here is the call graph for this function:

12.116.3.4 `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 pointer of the calling thread.

Return values

<code><0</code>	Error code.
--------------------	-------------

Return values

$>size$	More bytes to read, <code>size</code> bytes are in the buffer <code>buf</code> .
$\leq size$	Number of bytes read.

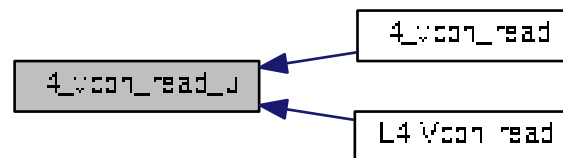
Note

Size must not exceed [L4_VCON_READ_SIZE](#).

Definition at line [355](#) of file [vcon.h](#).

Referenced by [l4_vcon_read\(\)](#), and [L4::Vcon::read\(\)](#).

Here is the caller graph for this function:



12.116.3.5 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

<code><0</code>	Error code.
<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.

Definition at line [349](#) of file [vcon.h](#).

12.116.3.6 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

<code>vcon</code>	Vcon object.
<code>buf</code>	Pointer to data buffer.
<code>size</code>	Size of buffer in bytes.

Returns

Syscall return tag

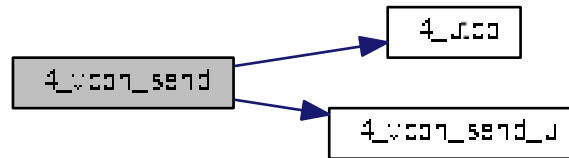
Note

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, do not use [l4_error\(\)](#), as [l4_error\(\)](#) will always return an error.

Definition at line [289](#) of file [vcon.h](#).

References [l4_utcb\(\)](#), and [l4_vcon_send_u\(\)](#).

Here is the call graph for this function:



12.116.3.7 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>	UTBC pointer of the calling thread.

Returns

Syscall return tag

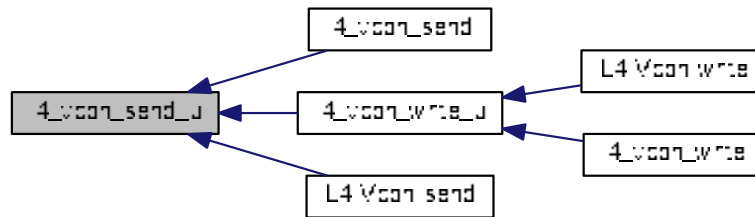
Note

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 [275](#) of file [vcon.h](#).

Referenced by [l4_vcon_send\(\)](#), [l4_vcon_write_u\(\)](#), and [L4::Vcon::send\(\)](#).

Here is the caller graph for this function:



12.116.3.8 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.

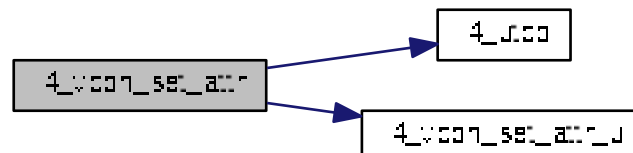
Returns

Syscall return tag

Definition at line 385 of file [vcon.h](#).

References [l4_utcb\(\)](#), and [l4_vcon_set_attr_u\(\)](#).

Here is the call graph for this function:



12.116.3.9 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 pointer of the calling thread.

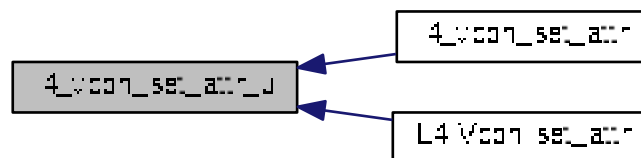
Returns

Syscall return tag.

Definition at line 371 of file [vcon.h](#).

Referenced by [l4_vcon_set_attr\(\)](#), and [L4::Vcon::set_attr\(\)](#).

Here is the caller graph for this function:



12.116.3.10 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.

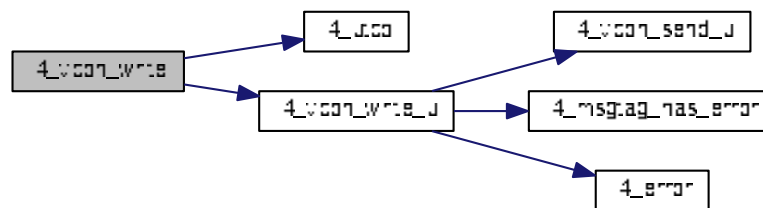
Return values

<0	Error.
≥ 0	Number of bytes written to the virtual console

Definition at line 310 of file [vcon.h](#).

References [l4_utcb\(\)](#), and [l4_vcon_write_u\(\)](#).

Here is the call graph for this function:



12.116.3.11 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 pointer of the calling thread.

Return values

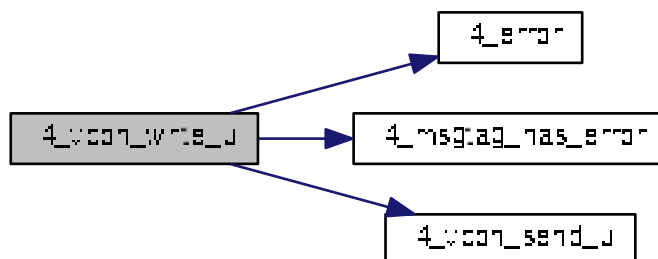
<0	Error.
≥ 0	Number of bytes written to the virtual console.

Definition at line 295 of file [vcon.h](#).

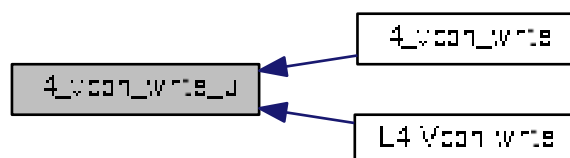
References [l4_error\(\)](#), [l4_msgtag_has_error\(\)](#), [l4_vcon_send_u\(\)](#), and [L4_VCON_WRITE_SIZE](#).

Referenced by [l4_vcon_write\(\)](#), and [L4::Vcon::write\(\)](#).

Here is the call graph for this function:



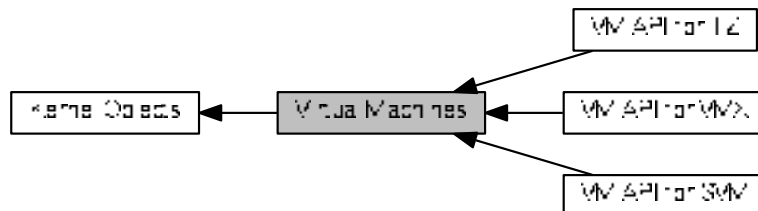
Here is the caller graph for this function:



12.117 Virtual Machines

Virtual Machine API.

Collaboration diagram for Virtual Machines:



Modules

- [VM API for SVM](#)
Virtual machine API for SVM.
- [VM API for TZ](#)
Virtual Machine API for ARM TrustZone.
- [VM API for VMX](#)
Virtual machine API for VMX.

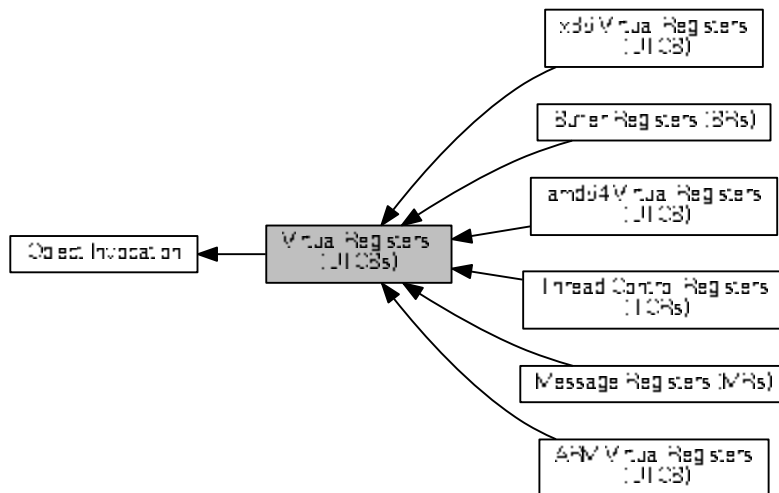
12.117.1 Detailed Description

Virtual Machine API.

12.118 Virtual Registers (UTCBs)

[L4 Virtual Registers \(UTCB\)](#).

Collaboration diagram for Virtual Registers (UTCBs):



Modules

- [ARM Virtual Registers \(UTCB\)](#)
- [Buffer Registers \(BRs\)](#)
- [Message Registers \(MRs\)](#)
- [Thread Control Registers \(TCRs\)](#)
- [amd64 Virtual Registers \(UTCB\)](#)
- [x86 Virtual Registers \(UTCB\)](#)

Files

- file [utcb.h](#)
UTCB definitions for ARM.
- file [utcb.h](#)
UTCB definitions for amd64.
- file [utcb.h](#)
UTCB definitions for X86.

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.

12.118.1 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.

12.118.2 Typedef Documentation

12.118.2.1 [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.

Definition at line 67 of file [utcb.h](#).

12.118.3 Function Documentation

12.118.3.1 l4_utcb_br()

```
l4_buf_regs_t * l4_utcb_br (
    void ) [inline]
```

Get the buffer-register block of a UTCB.

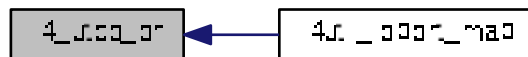
Returns

A pointer to the buffer-register block of u.

Definition at line 355 of file [utcb.h](#).

Referenced by [l4util_ioport_map\(\)](#).

Here is the caller graph for this function:



12.118.3.2 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 352 of file [utcb.h](#).

Referenced by [l4util_ioport_map\(\)](#).

Here is the caller graph for this function:



12.118.3.3 l4_tcb_tcr()

```
l4_thread_regs_t * l4_tcb_tcr (  
    void ) [inline]
```

Get the thread-control-register block of a UTCB.

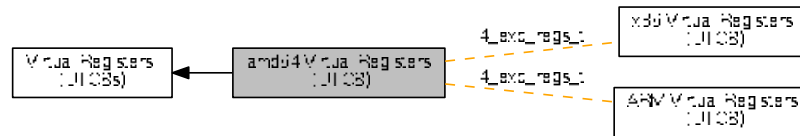
Returns

A pointer to the thread-control-register block of `u`.

Definition at line 358 of file [utcb.h](#).

12.119 amd64 Virtual Registers (UTCB)

Collaboration diagram for amd64 Virtual Registers (UTCB):



Data Structures

- struct [l4_exc_regs_t](#)
UTCB structure for exceptions.

Typedefs

- typedef struct [l4_exc_regs_t](#) [l4_exc_regs_t](#)
UTCB structure for exceptions.

Enumerations

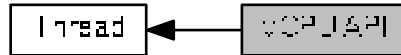
- enum [L4_utcb_consts_amd64](#)
UTCB constants for AMD64.

12.119.1 Detailed Description

12.120 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 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.

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_DEBUG_EXC](#) = 0x08,
[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](#) = 0x400, [L4_VCPU_OFFSET_EXT_INFOS](#) = 0x200 }
Offsets for vCPU state layouts.

12.120.1 Detailed Description

vCPU API

12.120.2 Enumeration Type Documentation

12.120.2.1 L4_vcpu_state_flags

enum [L4_vcpu_state_flags](#)

State flags of a vCPU.

Enumerator

L4_VCPU_F_IRQ	IRQs (events) enabled.
L4_VCPU_F_PAGE_FAULTS	Page faults enabled.
L4_VCPU_F_EXCEPTIONS	Exception enabled.
L4_VCPU_F_DEBUG_EXC	Debug exception enabled.
L4_VCPU_F_USER_MODE	User task will be used.
L4_VCPU_F_FPU_ENABLED	FPU enabled.

Definition at line 58 of file [vcpu.h](#).

12.120.2.2 L4_vcpu_state_offset

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 81 of file [vcpu.h](#).

12.120.2.3 L4_vcpu_sticky_flags

enum [L4_vcpu_sticky_flags](#)

Sticky flags of a vCPU.

Enumerator

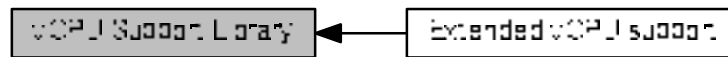
L4_VCPU_SF_IRQ_PENDING	An event (e.g. IRQ) is pending.
------------------------	---------------------------------

Definition at line 72 of file [vcpu.h](#).

12.121 vCPU Support Library

vCPU handling functionality.

Collaboration diagram for vCPU Support Library:



Modules

- [Extended vCPU support](#)
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.

12.121.1 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.

12.121.2 Function Documentation

12.121.2.1 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 208 of file [vcpu.h](#).

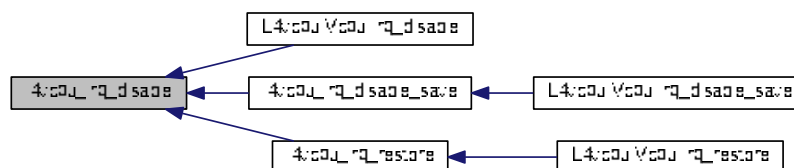
References [l4_barrier\(\)](#), [L4_CV](#), and [L4_VCPU_F_IRQ](#).

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:



12.121.2.2 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

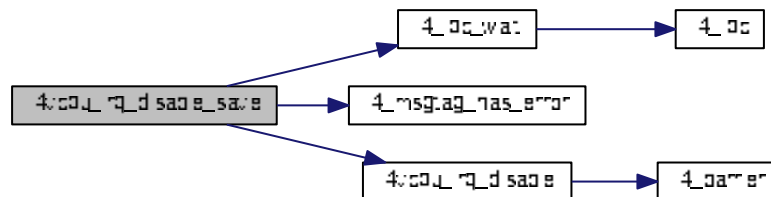
IRQ state before disabling IRQs.

Definition at line 216 of file [vcpu.h](#).

References [l4_vcpu_state_t::i](#), [L4_CV](#), [l4_ipc_wait\(\)](#), [L4_LIKELY](#), [l4_msgtag_has_error\(\)](#), [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:



12.121.2.3 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>	Utc b 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 239 of file [vcpu.h](#).

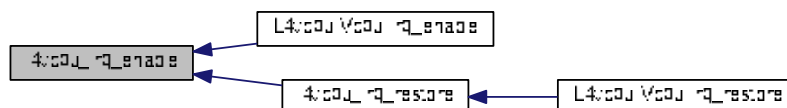
References [l4_barrier\(\)](#), [L4_CV](#), [L4_IPC_BOTH_TIMEOUT_0](#), [L4_LIKELY](#), [L4_VCPU_F_IRQ](#), [L4_VCPU_SF_IRQ_PENDING](#), [l4_vcpu_state_t::state](#), and [l4_vcpu_state_t::sticky_flags](#).

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:



12.121.2.4 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

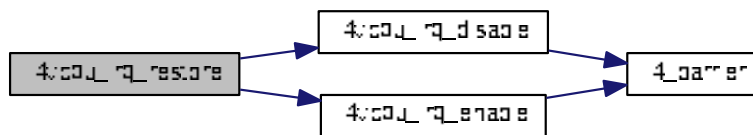
<i>vcpu</i>	Pointer to vCPU area.
<i>s</i>	IRQ state to be restored.
<i>utcb</i>	Utcbl pointer of the calling vCPU.
<i>do_event_work_cb</i>	Call-back function that is called in case an event (such as an interrupt) is pending after enabling.
<i>setup_ipc</i>	Function call-back that is called right before any IPC operation, and before event delivery is enabled.

Definition at line 264 of file [vcpu.h](#).

References [L4_CV](#), [L4_VCPU_F_IRQ](#), [l4vcpu_irq_disable\(\)](#), [l4vcpu_irq_enable\(\)](#), and [l4_vcpu_state_t::state](#).

Referenced by [L4vcpu::Vcpu::irq_restore\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



12.121.2.5 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.

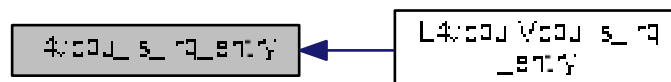
Parameters

<i>vcpu</i>	Pointer to vCPU area.
-------------	-----------------------

return 0 if not, !=0 otherwise.

Referenced by [L4vcpu::Vcpu::is_irq_entry\(\)](#).

Here is the caller graph for this function:



12.121.2.6 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.

Referenced by [L4vcpu::Vcpu::is_page_fault_entry\(\)](#).

Here is the caller graph for this function:



12.121.2.7 l4vcpu_print_state()

```
void l4vcpu_print_state (
    const l4_vcpu_state_t * vcpu,
    const char * prefix )
```

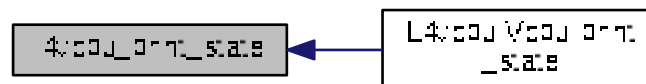
Print the state of a vCPU.

Parameters

<i>vcpu</i>	Pointer to vCPU area.
<i>prefix</i>	A prefix for each line printed.

Referenced by [L4vcpu::Vcpu::print_state\(\)](#).

Here is the caller graph for this function:



12.121.2.8 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>	UtcB 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.

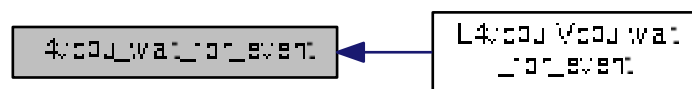
Note that event delivery remains disabled after this function returns.

Definition at line 277 of file [vcpu.h](#).

References [__END_DECLS](#), and [L4_IPC_NEVER](#).

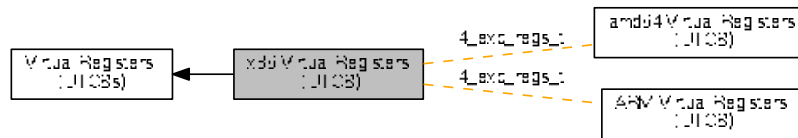
Referenced by [L4vcpu::Vcpu::wait_for_event\(\)](#).

Here is the caller graph for this function:



12.122 x86 Virtual Registers (UTCB)

Collaboration diagram for x86 Virtual Registers (UTCB):



Data Structures

- struct [l4_exc_regs_t](#)
UTCB structure for exceptions.

Typedefs

- typedef struct [l4_exc_regs_t](#) [l4_exc_regs_t](#)
UTCB structure for exceptions.

Enumerations

- enum [L4_utcb_consts_x86](#) {
[L4_UTCB_EXCEPTION_REGS_SIZE](#) = 19, [L4_UTCB_GENERIC_DATA_SIZE](#) = 63, [L4_UTCB_GENERIC_DATA_BUFFERS_SIZE](#) = 58, [L4_UTCB_MSG_REGS_OFFSET](#) = 0,
[L4_UTCB_BUF_REGS_OFFSET](#) = 64 * sizeof([l4_umword_t](#)), [L4_UTCB_THREAD_REGS_OFFSET](#) = 123
 * sizeof([l4_umword_t](#)), [L4_UTCB_INHERIT_FPU](#) = 1UL << 24, [L4_UTCB_OFFSET](#) = 512 }
UTCB constants for x86.

12.122.1 Detailed Description

12.122.2 Enumeration Type Documentation

12.122.2.1 L4_utcb_consts_x86

```
enum L4\_utcb\_consts\_x86
```

UTCB constants for x86.

Enumerator

L4_UTCB_EXCEPTION_REGS_SIZE	Number if message registers used for exception IPC.
L4_UTCB_GENERIC_DATA_SIZE	Total number of message register (MRs) available.
L4_UTCB_GENERIC_BUFFERS_SIZE	Total number of buffer registers (BRs) available.
L4_UTCB_MSG_REGS_OFFSET	Offset of MR[0] relative to the UTCB pointer.
L4_UTCB_BUF_REGS_OFFSET	Offset of BR[0] relative to the UTCB pointer.
L4_UTCB_THREAD_REGS_OFFSET	Offset of TCR[0] relative to the UTCB pointer.
L4_UTCB_INHERIT_FPU	BDR flag to accept reception of FPU state.
L4_UTCB_OFFSET	Offset of two consecutive UTCBs.

Definition at line 41 of file [utcb.h](#).

Chapter 13

Namespace Documentation

13.1 cxx Namespace Reference

Our C++ library.

Namespaces

- [Bits](#)

Internal helpers for the cxx package.

Data Structures

- class [Auto_ptr](#)
Smart pointer with automatic deletion.
- class [Avl_map](#)
AVL tree based associative container.
- class [Avl_set](#)
AVL set for simple comparable items.
- class [Avl_tree](#)
A generic AVL tree.
- class [Avl_tree_node](#)
Node of an AVL tree.
- class [Base_slab](#)
Basic slab allocator.
- class [Base_slab_static](#)
Merged slab allocator (allocators for objects of the same size are merged together).
- class [Bitfield](#)
Definition for a member (part) of a bit field.
- class [Bitmap](#)
A static bit map.
- class [Bitmap_base](#)
Basic bitmap abstraction.
- class [H_list](#)
General double-linked list of unspecified `cxx::H_list_item` elements.

- class [H_list_item_t](#)
Basic element type for a double-linked [H_list](#).
- struct [H_list_t](#)
Double-linked list of typed [H_list_item_t](#) elements.
- class [List](#)
Doubly linked list, with internal allocation.
- class [List_alloc](#)
Standard list-based allocator.
- class [List_item](#)
Basic list item.
- struct [Lt_functor](#)
Generic comparator class that defaults to the less-than operator.
- class [New_allocator](#)
Standard allocator based on `operator new ()`.
- class [Nothrow](#)
Helper type to distinguish the `operator new` version that does not throw exceptions.
- struct [Pair](#)
[Pair](#) of two values.
- class [Pair_first_compare](#)
Comparison functor for [Pair](#).
- class [Ref_ptr](#)
A reference-counting pointer with automatic cleanup.
- class [S_list](#)
Simple single-linked list.
- class [Slab](#)
[Slab](#) allocator for object of type `Type`.
- class [Slab_static](#)
Merged slab allocator (allocators for objects of the same size are merged together).
- class [static_vector](#)
Simple encapsulation for a dynamically allocated array.
- class [String](#)
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.
- class [Weak_ref](#)
Typed weak reference to an object of type `T`.
- class [Weak_ref_base](#)
Generic (base) weak reference to some object.

Functions

- `template<typename T1 >`
`T1 min (T1 a, T1 b)`
Get the minimum of `a` and `b`.
- `template<typename T1 >`
`T1 max (T1 a, T1 b)`
Get the maximum of `a` and `b`.

13.1.1 Detailed Description

Our C++ library.

Small Low-Level C++ Library.

Various kinds of C++ utilities.

13.2 cxx::Bits Namespace Reference

Internal helpers for the cxx package.

Data Structures

- struct [Avl_map_get_key](#)
Key-getter for [Avl_map](#).
- struct [Avl_set_get_key](#)
Internal, key-getter for [Avl_set](#) nodes.
- class [Base_avl_set](#)
Internal: AVL set with internally managed nodes.
- class [Basic_list](#)
Internal: Common functions for all head-based list implementations.
- class [Bst](#)
Basic binary search tree (BST).
- class [Bst_node](#)
Basic type of a node in a binary search tree (BST).
- struct [Direction](#)
The direction to go in a binary search tree.

13.2.1 Detailed Description

Internal helpers for the cxx package.

13.3 L4 Namespace Reference

[L4](#) low-level kernel interface.

Namespaces

- [lpc](#)
IPC related functionality.
- [lpc_svr](#)
Helper classes for [L4::Server](#) instantiation.
- [Typeid](#)
Definition of interface data-type helpers.
- [Types](#)
[L4](#) basic type helpers for C++.

Data Structures

- class [Alloc_list](#)
A simple list-based allocator.
- class [Base_exception](#)
Base class for all exceptions, thrown by the [L4Re](#) framework.
- class [Basic_registry](#)
This registry returns the corresponding server object based on the label of an [lpc_gate](#).
- class [Bounds_error](#)
Access out of bounds.
- class [Cap](#)
C++ interface for capabilities.
- class [Cap_base](#)
Base class for all kinds of capabilities.
- class [Com_error](#)
Error conditions during IPC.
- class [Debugger](#)
C++ debugger interface.
- class [Element_already_exists](#)
Exception for duplicate element insertions.
- class [Element_not_found](#)
Exception for a failed lookup (element not found).
- class [Exception_tracer](#)
Back-trace support for exceptions.
- class [Factory](#)
C++ [Factory](#) interface to create kernel objects.
- class [lcu](#)
C++ [lcu](#) interface.
- class [Invalid_capability](#)
Indicates that an invalid object was invoked.
- class [Io_pager](#)
[Io_pager](#) interface.
- class [Iommu](#)
Interface for IO-MMUs used for DMA remapping.
- class [IOModifier](#)
Modifier class for the IO stream.
- class [lpc_gate](#)
The C++ IPC gate interface.
- class [Irq](#)
C++ [Irq](#) interface.
- class [Irq_eoi](#)
Interface for sending an acknowledge message to an object.
- struct [Irq_handler_object](#)
[Server](#) object base class for handling IRQ messages.
- struct [Irq_mux](#)
IRQ multiplexer for shared IRQs.
- class [Kobject](#)
Base class for all kinds of kernel objects and remote objects, referenced by capabilities.
- 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...
 - class [Kobject_t](#)

Helper class to create an [L4Re](#) interface class that is derived from a single base class.
 - struct [Kobject_typeid](#)

Meta object for handling access to type information of Kobjects.
 - struct [Kobject_x](#)

Generic [Kobject](#) inheritance template.
 - class [Meta](#)

Meta interface that shall be implemented by each [L4Re](#) object and gives access to the dynamic type information for [L4Re](#) objects.
 - class [Out_of_memory](#)

Exception signalling insufficient memory.
 - class [Pager](#)

Pager interface including the [lo_pager](#) interface.
 - class [Platform_control](#)

L4 C++ interface for controlling platform-wide properties.
 - class [Poll_timeout_kipclock](#)

A polling timeout based on the [L4Re](#) clock.
 - struct [Proto_t](#)

Data type for defining protocol numbers.
 - class [Registry_iface](#)

Abstract interface for object registries.
 - class [Runtime_error](#)

Exception for an abstract runtime error.
 - class [Scheduler](#)

C++ interface of the [Scheduler](#) kernel object.
 - struct [Semaphore](#)

Kernel-provided semaphore object.
 - class [Server](#)

Basic server loop for handling client requests.
 - 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.
 - class [Smart_cap](#)

Smart capability class.
 - class [String](#)

A null-terminated string container class.
 - class [Task](#)

C++ interface of the [Task](#) kernel object.
 - class [Thread](#)

C++ [L4](#) kernel thread interface.
 - struct [Triggerable](#)

Interface that allows an object to be triggered by some source.
 - struct [Type_info](#)

Dynamic Type Information for [L4Re](#) Interfaces.
 - class [Unknown_error](#)

- *Exception for an unknown condition.*
- class [Vcon](#)
C++ L4 Vcon interface.
- class [Vm](#)
Virtual machine.

Typedefs

- typedef int [Opcode](#)
Data type for RPC opcodes.

Enumerations

- enum { [PROTO_ANY](#) = 0, [PROTO_EMPTY](#) = -19 }

Functions

- template<typename T >
[Type_info](#) const * [kobject_typeid](#) ()
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) throw ()
dynamic_cast for capabilities.
- template<typename T , typename F >
[Cap](#)< T > [cap_cast](#) ([Cap](#)< F > const &c) throw ()
static_cast for capabilities.
- template<typename T , typename F >
[Cap](#)< T > [cap_reinterpret_cast](#) ([Cap](#)< F > const &c) throw ()
reinterpret_cast for capabilities.
- template<typename T , typename F , typename SMART >
[Smart_cap](#)< T, SMART > [cap_cast](#) ([Smart_cap](#)< F, SMART > const &c) throw ()
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) throw ()
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.

13.3.1 Detailed Description

[L4](#) low-level kernel interface.

13.3.2 Enumeration Type Documentation

13.3.2.1 anonymous enum

anonymous enum

Enumerator

PROTO_ANY	Default protocol used by Kobject_t and Kobject_x .
PROTO_EMPTY	Empty protocol for empty APIs.

Definition at line 55 of file [__typeinfo.h](#).

13.3.3 Function Documentation

13.3.3.1 `cap_cast()` [1/2]

```
template<typename T , typename F , typename SMART >  
Smart\_cap<T, SMART> L4::cap_cast (  
    Smart\_cap< F, SMART > const & c ) throw )    [inline]
```

`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 203 of file [smart_capability](#).

13.3.3.2 cap_cast() [2/2]

```
template<typename T , typename F >
Cap<T> L4::cap_cast (
    Cap< F > const & c ) throw )    [inline]
```

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 376 of file [capability.h](#).

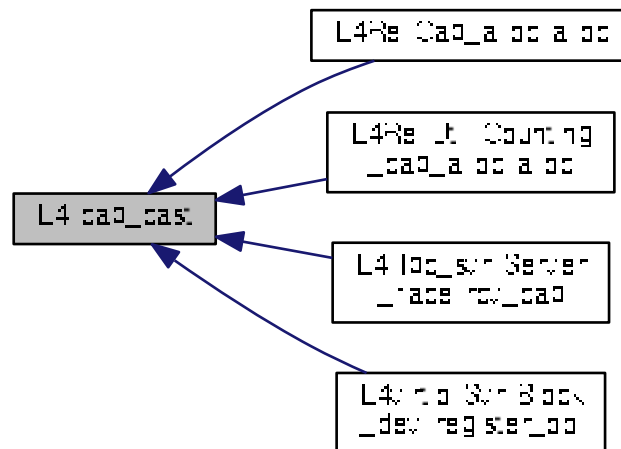
References [L4::Cap_base::cap\(\)](#).

Referenced by [L4Re::Cap_alloc::alloc\(\)](#), [L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::alloc\(\)](#), [L4::lpc_svr::Server_iface::rcv_cap\(\)](#), and [L4virtio::Svr::Block_dev< Ds_data >::register_obj\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



13.3.3.3 cap_dynamic_cast()

```

template<typename T , typename F >
Cap<T> L4::cap_dynamic_cast (
    Cap< F > const & c ) throw ()    [inline]
  
```

`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

<i>Cap<T></i>	Capability of target interface <i>T</i> .
<i>L4_INVALID_CAP</i>	<i>c</i> does not support the target interface <i>T</i> or the L4::Meta 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.

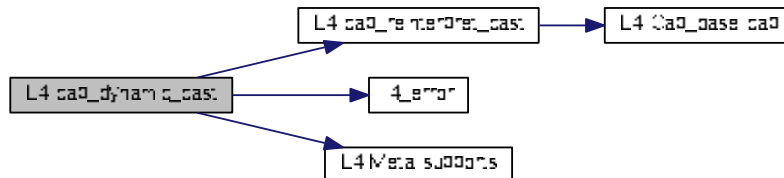
Example code:

```
L4::Cap<L4::Kobject> obj = ... ;
L4::Cap<L4::Icu> icu = L4::cap_dynamic_cast<L4::Icu>(obj);
```

Definition at line 119 of file [capability](#).

References [cap_reinterpret_cast\(\)](#), [l4_error\(\)](#), and [L4::Meta::supports\(\)](#).

Here is the call graph for this function:



13.3.3.4 cap_reinterpret_cast() [1/2]

```
template<typename T , typename F , typename SMART >
Smart_cap<T, SMART> L4::cap_reinterpret_cast (
    Smart_cap< F, SMART > const & c ) throw )    [inline]
```

`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 222 of file [smart_capability](#).

13.3.3.5 `cap_reinterpret_cast()` [2/2]

```
template<typename T , typename F >
Cap<T> L4::cap_reinterpret_cast (
    Cap< F > const & c ) throw ()    [inline]
```

`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 407 of file [capability.h](#).

References [L4::Cap_base::cap\(\)](#).

Referenced by [cap_dynamic_cast\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



```
void L4::throw_ipc_exception (
    L4::Cap< void > const & o,
    l4_msgtag_t const & err,
    l4_utcb_t * utcb ) [inline]
```

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, usually the UTCB of the calling thread.

Referenced by [throw_ipc_exception\(\)](#).

[illegible]

Here is the caller graph for this function:



13.3.3.7 `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, usually the UTCB of the calling thread.

Definition at line [61](#) of file [ipc_helper](#).

References [throw_ipc_exception\(\)](#).

- struct [Call_zero_send_timeout](#)
RPC attribute for an RPC call, with zero send timeout.
- class [Cap](#)
Capability type for RPC interfaces (see [L4::Cap<T>](#)).
- class [Gen_fpage](#)
Generic RPC wrapper for [L4](#) flex-pages.
- struct [In_out](#)
Mark an argument as in-out argument.
- class [lostream](#)
Input/Output stream for IPC [un]marshalling.
- class [lstream](#)
Input stream for IPC unmarshalling.
- class [Msg_ptr](#)
*Pointer to an element of type *T* in an [lpc::lstream](#).*
- struct [Opt](#)
Attribute for defining an optional RPC argument.
- class [Ostream](#)
Output stream for IPC marshalling.
- struct [Out](#)
Mark an argument as a output value in an RPC signature.
- struct [Ret_array](#)
*Dynamically sized output array of type *T*.*
- struct [Send_only](#)
RPC attribute for a send-only RPC.
- class [Small_buf](#)
A receive item for receiving a single capability.
- class [Snd_item](#)
RPC wrapper for a send item.
- class [Str_cp_in](#)
Abstraction for extracting a zero-terminated string from an [lpc::lstream](#).
- 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.

Typedefs

- typedef unsigned short [Array_len_default](#)
Default type for passing length of an array.
- typedef [Gen_fpage](#)< [Snd_item](#) > [Snd_fpage](#)
Send flex-page.
- typedef [Gen_fpage](#)< [Buf_item](#) > [Rcv_fpage](#)
Rcv flex-page.

Functions

- `template<typename T >`
`Cap< T > make_cap (L4::Cap< T > cap, unsigned rights)`
Make an L4::lpc::Cap<T> for the given capability and rights.
- `template<typename T >`
`Cap< T > make_cap_rw (L4::Cap< T > cap)`
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)`
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)`
Make an L4::IPC::Cap<T> for the given capability with full fpage and object-specific 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.

13.4.1 Detailed Description

IPC related functionality.

13.4.2 Function Documentation

13.4.2.1 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::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.

[buf_cp_in\(\)](#) can be used to extract an array from an [lpc::lstream](#). 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 172 of file [ipc_stream](#).

13.4.2.2 [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 114 of file [ipc_stream](#).

13.4.2.3 [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::lstream .
out	<i>size</i>	The number of elements found in the stream.

This routine provides a possibility to extract an array from an [lpc::lstream](#), 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 321 of file [ipc_stream](#).

13.4.2.4 make_cap()

```
template<typename T >
Cap<T> L4::Ipc::make_cap (
    L4::Cap< T > cap,
    unsigned rights )
```

Make an [L4::Ipc::Cap<T>](#) for the given capability and rights.

Template Parameters

<i>T</i>	(IMPLICIT) type of the referenced interface
----------	---

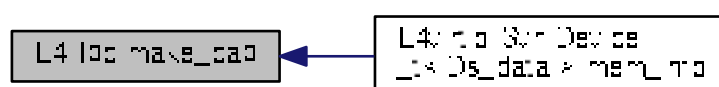
Parameters

<i>cap</i>	source capability (L4::Cap<T>)
<i>rights</i>	rights mask that shall be applied on transfer.

Definition at line 621 of file [ipc_types](#).

Referenced by [L4virtio::Svr::Device_t< Ds_data >::mem_info\(\)](#).

Here is the caller graph for this function:



13.4.2.5 make_cap_full()

```
template<typename T >
Cap<T> L4::ipc::make_cap_full (
    L4::Cap< T > cap )
```

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](#)
[L4_obj_fpage_ctl](#)

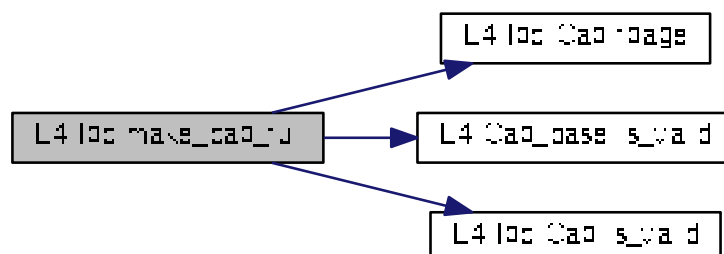
Note

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\(\)](#).

Definition at line 659 of file [ipc_types](#).

References [L4::ipc::Cap< T >::fpage\(\)](#), [L4::Cap_base::is_valid\(\)](#), [L4::ipc::Cap< T >::is_valid\(\)](#), [L4_CAP_FPAGE_C_OBJ_RIGHTS](#), [L4_CAP_FPAGE_C_RWSD](#), [L4_MSGMISSARG](#), [L4_FPAGE_C_OBJ_RIGHTS](#), and [L4_UNLIKELY](#).

Here is the call graph for this function:



Template Parameters

<i>T</i>	(IMPLICIT) type of the referenced interface
----------	---

Parameters

<i>cap</i>	source capability (L4::Cap<T>)
------------	--------------------------------

Definition at line 641 of file [ipc_types](#).

References [L4_CAP_FPAGE_RWS](#).

13.4.2.8 msg_ptr()

```
template<typename T >
Msg_ptr<T> L4::Ipc::msg_ptr (
    T *& p )
```

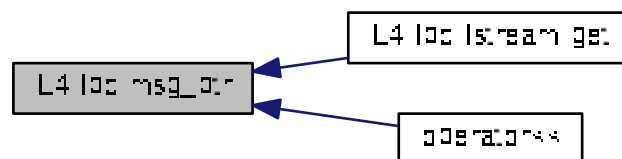
Create an [Msg_ptr](#) to adjust the given pointer.

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 264 of file [ipc_stream](#).

Referenced by [L4::Ipc::Istream::get\(\)](#), and [operator<<\(\)](#).

Here is the caller graph for this function:



13.4.2.9 read()

```
template<typename T >
T L4::Ipc::read (
    Istream & s ) [inline]
```

Read a value out of a stream.

Parameters

s	An Istream .
---	------------------------------

Returns

The value of type T.

The stream position is progressed accordingly.

Definition at line 1392 of file [ipc_stream](#).

13.4.2.10 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

	v	Pointer to the array that shall receive the values from the lpc::Istream .
in, out	size	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::Istream](#) (

See also

[Str_cp_in](#).)

Definition at line 225 of file [ipc_stream](#).

13.5 L4::Ipc::Msg Namespace Reference

IPC Message related functionality.

Data Structures

- struct [Clnt_val_ops](#)
Defines client-side handling of 'MTYPE' as RPC argument.
- struct [Cls_buffer](#)
Marker type for receive buffer values.

- struct [Cls_data](#)
Marker type for data values.
- struct [Cls_item](#)
Marker type for item values.
- struct [Dir_in](#)
Marker type for input values.
- struct [Dir_out](#)
Marker type for output values.
- struct [Do_in_data](#)
Marker for Input data.
- struct [Do_in_items](#)
Marker for Input items.
- struct [Do_out_data](#)
Marker for Output data.
- struct [Do_out_items](#)
Marker for Output items.
- struct [Do_rcv_buffers](#)
Marker for receive buffers.
- struct [Elem< Array< A, LEN > &>](#)
Array as output argument.
- struct [Elem< Array< A, LEN > >](#)
Array as input arguments.
- struct [Elem< Array_ref< A, LEN > &>](#)
Array_ref as output argument.
- 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 [Svr_val_ops](#)
Defines server-side handling for `MTYPE` server arguments.

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

- unsigned long [align_to](#) (unsigned long bytes, unsigned long align)
Pad bytes to the given alignment align (in bytes)
- template<typename T >
 unsigned long [align_to](#) (unsigned long bytes)
Pad bytes to the alignment of the type T.
- template<typename T >
 bool [check_size](#) (unsigned offset, unsigned limit)
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)`
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)`
Add some data to a message at offs.
- `template<typename T >`
`int msg_get (char *msg, unsigned offs, unsigned limit, T &v)`
Get some data from a message at offs.

13.5.1 Detailed Description

IPC Message related functionality.

13.5.2 Enumeration Type Documentation

13.5.2.1 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 96 of file [ipc_basics](#).

13.5.3 Function Documentation

13.5.3.1 [align_to\(\)](#) [1/2]

```
unsigned long L4::Ipc::Msg::align_to (
    unsigned long bytes,
    unsigned long align ) [inline]
```

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 41 of file [ipc_basics](#).

Referenced by [align_to\(\)](#).

Here is the caller graph for this function:

13.5.3.2 `align_to()` [2/2]

```
template<typename T >
unsigned long L4::ipc::Msg::align_to (
    unsigned long bytes ) [inline]
```

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 51 of file [ipc_basics](#).

References [align_to\(\)](#).

Here is the call graph for this function:



13.5.3.3 `check_size()` [1/2]

```

template<typename T >
bool L4::IpC::Msg::check_size (
    unsigned offset,
    unsigned limit ) [inline]
  
```

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 64 of file [ipc_basics](#).

13.5.3.4 `check_size()` [2/2]

```

template<typename T , typename CTYPE >
bool L4::IpC::Msg::check_size (
    unsigned offset,
    unsigned limit,
    CTYPE cnt ) [inline]
  
```

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 82 of file [ipc_basics](#).

References [L4_UNLIKELY](#).

13.5.3.5 msg_add()

```
template<typename T >
int L4::Ipc::Msg::msg_add (
    char * msg,
    unsigned offs,
    unsigned limit,
    T v ) [inline]
```

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 125 of file [ipc_basics](#).

References [L4_MSGTOOLONG](#), and [L4_UNLIKELY](#).

13.5.3.6 msg_get()

```
template<typename T >
int L4::Ipc::Msg::msg_get (
    char * msg,
    unsigned offs,
    unsigned limit,
    T & v ) [inline]
```

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 146 of file [ipc_basics](#).

References [L4_MSGTOOSHORT](#), and [L4_UNLIKELY](#).

13.6 L4::ipc_svr Namespace Reference

Helper classes for [L4::Server](#) instantiation.

Data Structures

- class [Br_manager_no_buffers](#)
Empty implementation of [Server_iface](#).
- struct [Compound_reply](#)
Mix in for `LOOP_HOOKS` to always use compound reply and wait.
- struct [Default_loop_hooks](#)
Default `LOOP_HOOKS`.
- struct [Default_setup_wait](#)

- *Mix in for LOOP_HOOKS for setup_wait no op.*
- struct [Default_timeout](#)
 - *Mix in for LOOP_HOOKS to use a 0 send and a infinite receive timeout.*
- struct [Direct_dispatch](#)
 - *Direct disptach helper, for forwarding dispatch calls a registry R.*
- struct [Direct_dispatch< R * >](#)
 - *Direct disptach 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 [Ignore_errors](#)
 - *Mix in for LOOP_HOOKS to ignore IPC errors.*
- class [Server_iface](#)
 - *Interface for server-loop related functions.*
- class [Timed_work](#)
 - *DEPRECATED.*
- 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.*

13.6.1 Detailed Description

Helper classes for [L4::Server](#) instantiation.

13.7 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_nocode](#)
 - *List of RPCs of an interface using a single operation without an opcode.*
- struct [Rpcs](#)
 - *Standard list of RPCs of an interface.*
- struct [Rpcs_code](#)
 - *List of RPCs of an interface using a special opcode type.*
- struct [Rpcs_sys](#)
 - *List of RPCs typically used for kernel interfaces.*

13.7.1 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>`.

13.8 L4::Types Namespace Reference

[L4](#) basic type helpers for C++.

Data Structures

- struct [Bool](#)
Boolean meta type.
- struct [False](#)
False meta value.
- class [Flags](#)
Template for defining typical [Flags](#) bitmaps.
- struct [Same](#)
Compare two data types for equality.
- struct [True](#)
True meta value.

13.8.1 Detailed Description

[L4](#) basic type helpers for C++.

13.9 L4Re Namespace Reference

[L4Re](#) C++ Interfaces.

Namespaces

- [Vfs](#)
Virtual file system for interfaces POSIX libc.

Data Structures

- class [Cap_alloc](#)
Capability allocator interface.
- class [Console](#)
Console class.
- class [Dataspace](#)
Interface for memory-like objects.
- class [Debug_obj](#)
Debug interface.
- class [Dma_space](#)
DMA Address Space.
- class [Env](#)
C++ interface of the initial environment that is provided to an [L4](#) task.
- class [Event](#)
Event class.
- class [Event_buffer_t](#)
Event buffer class.
- class [Inhibitor](#)
Set of inhibitor locks, which inhibit specific actions when held.
- 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.
- class [Rm](#)
Region map.
- class [Smart_cap_auto](#)
Helper for [Auto_cap](#) and [Auto_del_cap](#).

Functions

- long [chksys](#) (long err, char const *extra="", long ret=0)
Generate C++ exception on error.
- long [chksys](#) ([l4_msgtag_t](#) const &t, char const *extra="", [l4_utcb_t](#) *utcb=[l4_utcb](#)(), long ret=0)
Generate C++ exception on error.
- long [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="", long err=-[L4_ENOMEM](#))
Check for valid capability or raise C++ exception.

13.9.1 Detailed Description

[L4Re](#) C++ Interfaces.

[L4](#) Runtime Environment.

13.9.2.2 `chksys()` [1/3]

```

long L4Re::chksys (
    long err,
    char const * extra = "",
    long 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)

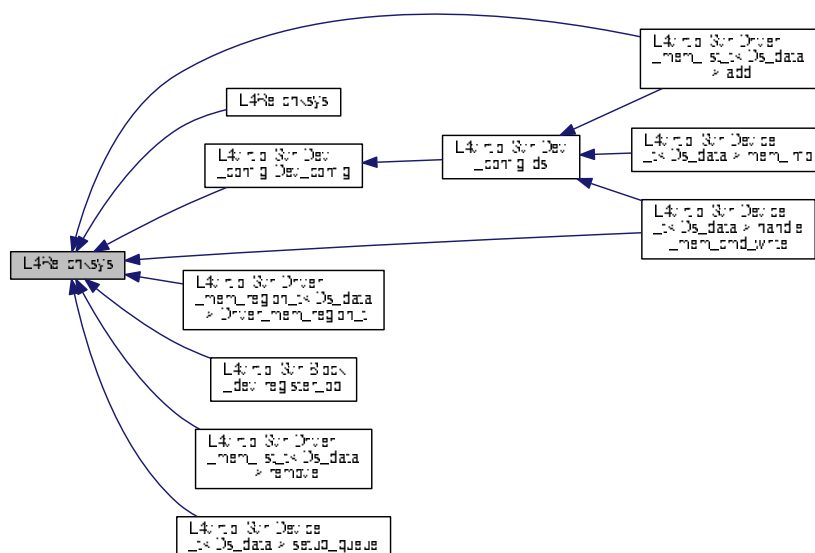
This function throws an exception if the `err` is negative and otherwise returns `err`.

Definition at line 62 of file [error_helper](#).

References [L4_UNLIKELY](#).

Referenced by [L4virtio::Svr::Driver_mem_list_t< Ds_data >::add\(\)](#), [chksys\(\)](#), [L4virtio::Svr::Dev_config::Dev_config\(\)](#), [L4virtio::Svr::Driver_mem_region_t< Ds_data >::Driver_mem_region_t\(\)](#), [L4virtio::Svr::Device_t< Ds_data >::handle_mem_cmd_write\(\)](#), [L4virtio::Svr::Block_dev< Ds_data >::register_obj\(\)](#), [L4virtio::Svr::Driver_mem_list_t< Ds_data >::remove\(\)](#), and [L4virtio::Svr::Device_t< Ds_data >::setup_queue\(\)](#).

Here is the caller graph for this function:



13.9.2.3 `chksys()` [2/3]

```

long L4Re::chksys (
    l4_msgtag_t const & t,
    char const * extra = "",
    l4_utcb_t * utcb = l4_utcb(),
    long ret = 0 ) [inline]

```

Generate C++ exception on error.

Parameters

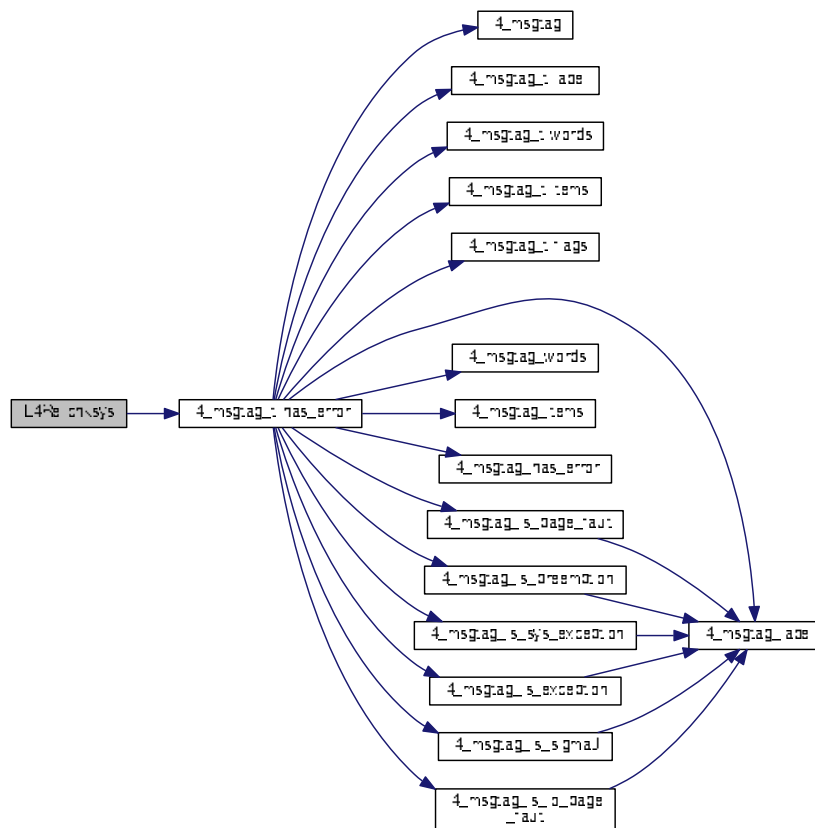
<i>t</i>	Message tag.
<i>extra</i>	Optional text for exception (default "")
<i>utcb</i>	Option UTCB
<i>ret</i>	Optional value for exception, default is error value (err)

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 83 of file [error_helper](#).

References [l4_msgtag_t::has_error\(\)](#), and [L4_UNLIKELY](#).

Here is the call graph for this function:



13.9.2.4 chksys() [3/3]

```
long 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 106 of file [error_helper](#).

References [chksys\(\)](#), and [L4_UNLIKELY](#).

Here is the call graph for this function:



13.10 L4Re::Vfs Namespace Reference

Virtual file system for interfaces 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 [Directory](#)
Interface for a POSIX file that is a directory.
- class [File](#)
The basic interface for an open POSIX file.
- class [File_system](#)
Basic interface for an [L4Re::Vfs](#) file system.

- class [Fs](#)
POSIX File-system related functionality.
- class [Generic_file](#)
The common interface for an open POSIX file.
- class [Mman](#)
Interface for the POSIX memory management.
- class [Ops](#)
Interface for the POSIX backends for an application.
- 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.

Functions

- [L4Re::Vfs::Ops](#) *vfs_ops [asm](#) ("l4re_env_posix_vfs_ops")
Reference to the applications [L4Re::Vfs::Ops](#) singleton.

13.10.1 Detailed Description

Virtual file system for interfaces POSIX libc.

13.11 L4vbus Namespace Reference

C++ interface of the [Vbus](#) API.

Data Structures

- class [Device](#)
[Device](#) on a virtual bus (V-BUS)
- class [Gpio_module](#)
A [Gpio_module](#) groups multiple GPIO pins together.
- class [Gpio_pin](#)
A GPIO pin.
- class [Icu](#)
V-BUS Interrupt controller API (ICU)
- class [Pci_dev](#)
A PCI device.
- class [Pci_host_bridge](#)
A Pci host bridge.
- class [Pm](#)
Power-management API mixin.
- class [Vbus](#)
The virtual BUS.

13.11.1 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 [lcu](#) ([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>
```

Refer to [L4 V-BUS functions](#) for the C API.

13.12 L4virtio Namespace Reference

L4-VIRTIO Transport C++ API.

Data Structures

- class [Ptr](#)
Pointer used in virtio descriptors.
- class [Virtqueue](#)
Low-level [Virtqueue](#).

13.12.1 Detailed Description

L4-VIRTIO Transport C++ API.

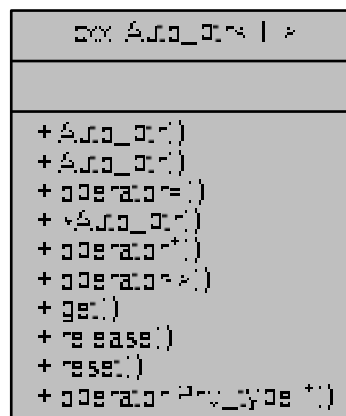
Chapter 14

Data Structure Documentation

14.1 `cxx::Auto_ptr< T >` Class Template Reference

Smart pointer with automatic deletion.

Collaboration diagram for `cxx::Auto_ptr< T >`:



Public Types

- typedef T [Ref_type](#)

The referenced type.

Public Member Functions

- [Auto_ptr](#) (T *p=0) throw ()
Construction by assignment of a normal pointer.
- [Auto_ptr](#) ([Auto_ptr](#) const &o) throw ()
Copy construction, releases the original pointer.
- [Auto_ptr](#) & [operator=](#) ([Auto_ptr](#) const &o) throw ()
Assignment from another smart pointer.
- [~Auto_ptr](#) () throw ()
Destruction, shall delete the object.
- T & [operator*](#) () const throw ()
Dereference the pointer.
- T * [operator->](#) () const throw ()
Member access for the object.
- T * [get](#) () const throw ()
Get the normal pointer.
- T * [release](#) () throw ()
Release the object and get the normal pointer back.
- void [reset](#) (T *p=0) throw ()
Delete the object and reset the smart pointer to NULL.
- [operator Priv_type *](#) () const throw ()
Operator for `if (!ptr) ...`

14.1.1 Detailed Description

```
template<typename T>
class cxx::Auto_ptr< T >
```

Smart pointer with automatic deletion.

Template Parameters

<i>T</i>	The type of the referenced object.
----------	------------------------------------

This smart pointer calls the delete operator when the destructor is called. This has the effect that the object the pointer points to will be deleted when the pointer goes out of scope, or a new value gets assigned. The smart pointer provides a [release\(\)](#) method to get a normal pointer to the object and set the smart pointer to NULL.

Definition at line 36 of file [auto_ptr](#).

14.1.2 Member Typedef Documentation

14.1.2.1 Ref_type

```
template<typename T >
typedef T cxx::Auto_ptr< T >::Ref_type
```

The referenced type.

Definition at line 41 of file [auto_ptr](#).

14.1.3 Constructor & Destructor Documentation

14.1.3.1 `Auto_ptr()` [1/2]

```
template<typename T >
cxx::Auto_ptr< T >::Auto_ptr (
    T * p = 0 ) throw ()    [inline], [explicit]
```

Construction by assignment of a normal pointer.

Parameters

<i>p</i>	The pointer to the object
----------	---------------------------

Definition at line 51 of file `auto_ptr`.

14.1.3.2 `Auto_ptr()` [2/2]

```
template<typename T >
cxx::Auto_ptr< T >::Auto_ptr (
    Auto_ptr< T > const & o ) throw ()    [inline]
```

Copy construction, releases the original pointer.

Parameters

<i>o</i>	The smart pointer, which shall be copied and released.
----------	--

Definition at line 57 of file `auto_ptr`.

14.1.3.3 `~Auto_ptr()`

```
template<typename T >
cxx::Auto_ptr< T >::~~Auto_ptr ( ) throw ()    [inline]
```

Destruction, shall delete the object.

Definition at line 76 of file `auto_ptr`.

14.1.4 Member Function Documentation

14.1.4.1 `get()`

```
template<typename T >
T* cxx::Auto_ptr< T >::get ( ) const throw ( )    [inline]
```

Get the normal pointer.

Attention

This function will not release the object, the object will be deleted by the smart pointer.

Definition at line 90 of file `auto_ptr`.

14.1.4.2 `operator Priv_type *()`

```
template<typename T >
cxx::Auto_ptr< T >::operator Priv_type * ( ) const throw ( )    [inline]
```

Operator for `if (!ptr)`

Definition at line 110 of file `auto_ptr`.

14.1.4.3 `operator*()`

```
template<typename T >
T& cxx::Auto_ptr< T >::operator* ( ) const throw ( )    [inline]
```

Dereference the pointer.

Definition at line 80 of file `auto_ptr`.

14.1.4.4 `operator->()`

```
template<typename T >
T* cxx::Auto_ptr< T >::operator-> ( ) const throw ( )    [inline]
```

Member access for the object.

Definition at line 83 of file `auto_ptr`.

14.1.4.5 `operator=()`

```
template<typename T >
Auto_ptr& cxx::Auto_ptr< T >::operator= (
    Auto_ptr< T > const & o ) throw ( )    [inline]
```

Assignment from another smart pointer.

Parameters

<code>o</code>	The source for the assignment (will be released).
----------------	---

Definition at line 65 of file [auto_ptr](#).

References [cxx::Auto_ptr< T >::release\(\)](#).

Here is the call graph for this function:

14.1.4.6 `release()`

```
template<typename T >
T* cxx::Auto_ptr< T >::release ( ) throw ( )    [inline]
```

Release the object and get the normal pointer back.

After calling this function the smart pointer will point to NULL and the object will not be deleted by the pointer anymore.

Definition at line 98 of file [auto_ptr](#).

Referenced by [cxx::Auto_ptr< T >::operator=\(\)](#).

Here is the caller graph for this function:



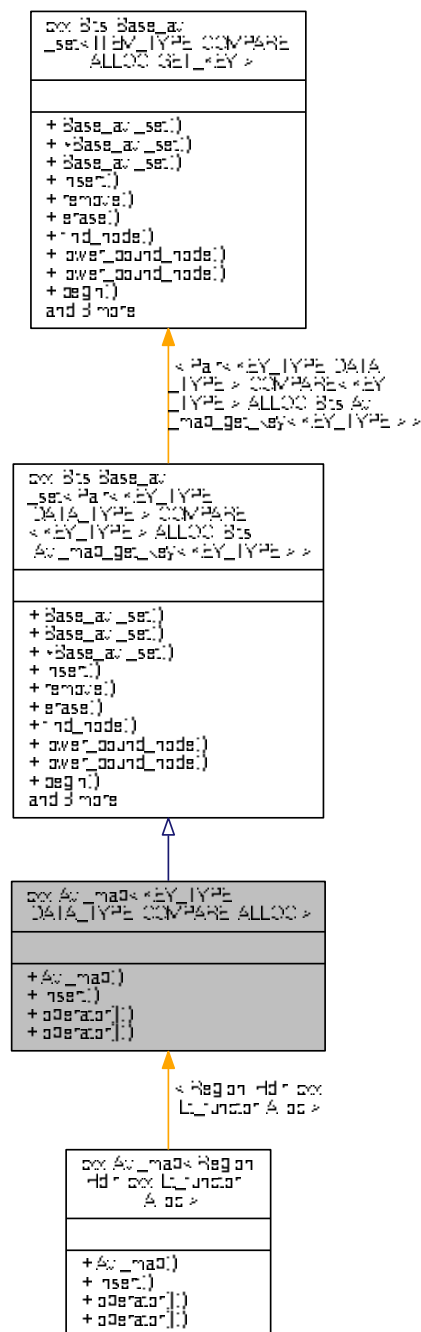
The documentation for this class was generated from the following file:

- `I4/cxx/auto_ptr`

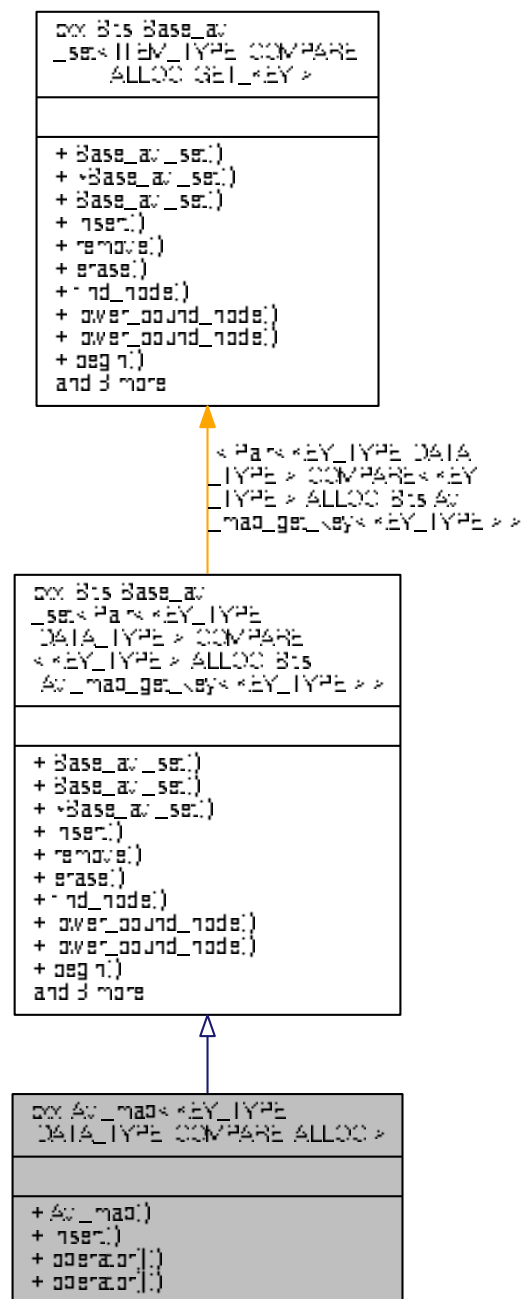
14.2 cxx::Avl_map< KEY_TYPE, DATA_TYPE, COMPARE, ALLOC > Class Template Reference

AVL tree based associative container.

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 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.
- [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.

14.2.1 Detailed Description

```
template<typename KEY_TYPE, typename DATA_TYPE, template< typename A > class COMPARE = Lt_functor, 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 56 of file [avl_map](#).

14.2.2 Constructor & Destructor Documentation

14.2.2.1 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 91 of file `avl_map`.

14.2.3 Member Function Documentation

14.2.3.1 `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 an element is already in the set. 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 109 of file `avl_map`.

14.2.3.2 `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 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 117 of file [avl_map](#).

14.2.3.3 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& 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 129 of file [avl_map](#).

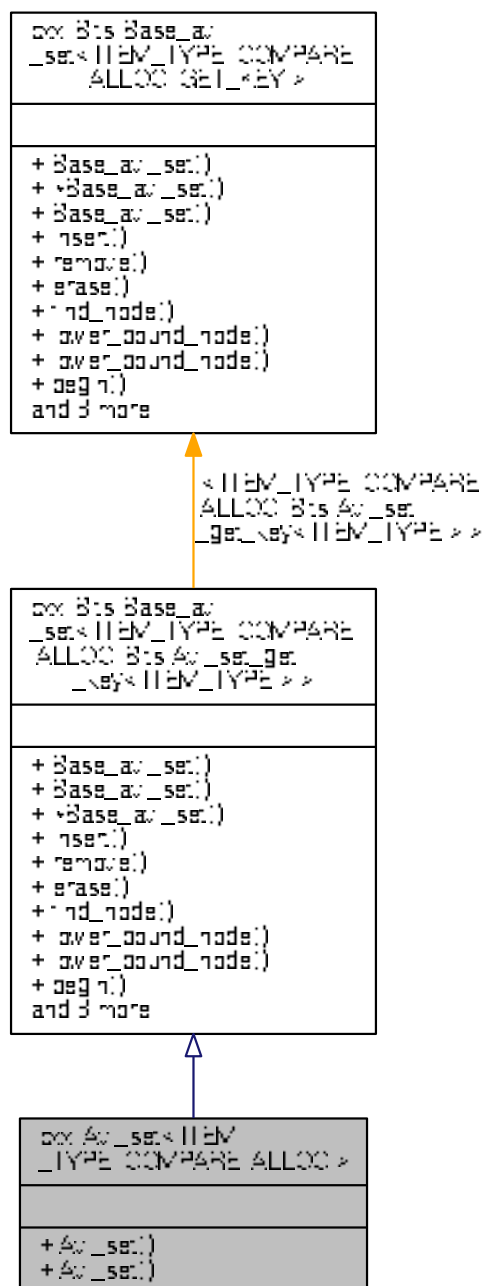
The documentation for this class was generated from the following file:

- I4/cxx/[avl_map](#)

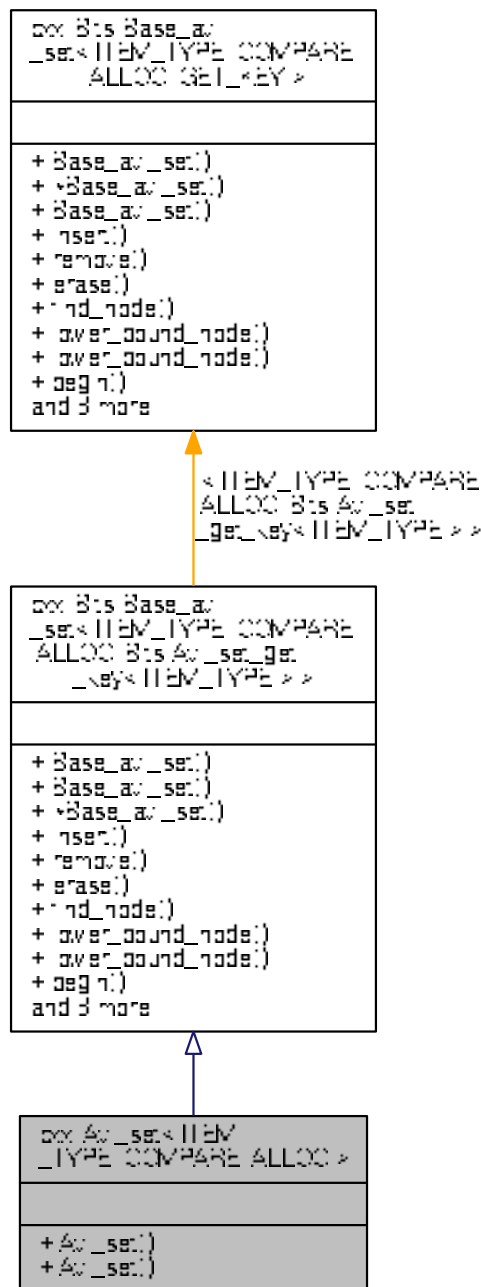
14.3 cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC > Class Template Reference

AVL set for simple compareable items.

Inheritance diagram for cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC >:



Collaboration diagram for `cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC >`:



Additional Inherited Members

14.3.1 Detailed Description

```
template<typename ITEM_TYPE, class COMPARE = Lt_functor<ITEM_TYPE>, template< typename A > class ALLOC = New_↵  
_allocator>  
class cxx::Avl_set< ITEM_TYPE, COMPARE, ALLOC >
```

AVL set for simple compareable 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 419 of file [avl_set](#).

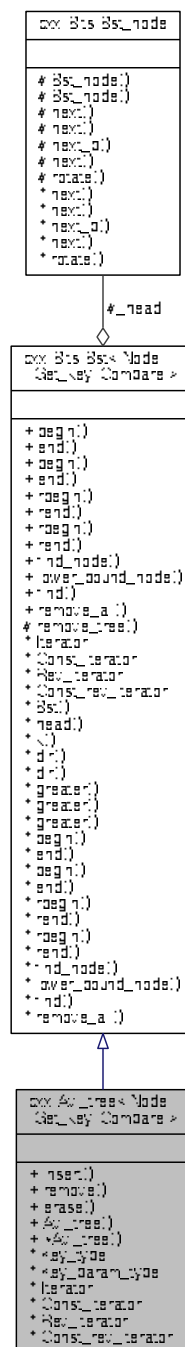
The documentation for this class was generated from the following file:

- [I4/cxx/avl_set](#)

14.4 `cxx::Avl_tree< Node, Get_key, Compare >` Class Template Reference

A generic AVL tree.

Collaboration diagram for cxx::Avl_tree< Node, Get_key, Compare >:



Public Types

- typedef [Bst::Iterator](#) `Iterator`

- *Forward iterator for the tree.*
- 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 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](#) ()
Create an empty AVL tree.
- [~Avl_tree](#) () noexcept
Destroy the tree.

Additional Inherited Members

14.4.1 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 <i>l</i> is smaller than the key <i>r</i> .

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 109 of file [avl_tree](#).

14.4.2 Member Function Documentation

14.4.2.1 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.
-----------------	----------------------------

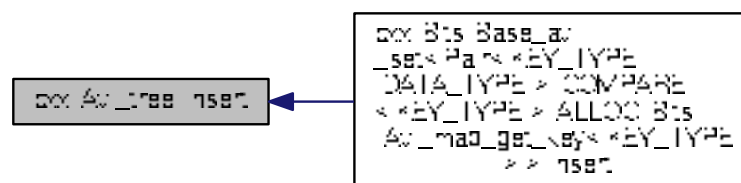
Returns

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 229 of file [avl_tree](#).

Referenced by [cxx::Bits::Base_avl_set< Pair< KEY_TYPE, DATA_TYPE >, COMPARE< KEY_TYPE >, ALLOC, Bits::Avl_map_get_key< KEY_TYPE > >::insert\(\)](#).

Here is the caller graph for this function:



14.4.2.2 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.
------------	--------------------------------

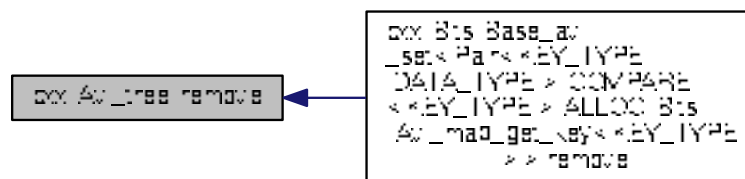
Returns

The pointer to the removed node on success, or 0 if no node with the *key* exists.

Definition at line 291 of file [avl_tree](#).

Referenced by [cxx::Bits::Base_avl_set< Pair< KEY_TYPE, DATA_TYPE >, COMPARE< KEY_TYPE >, ALLOC, Bits::Avl_map_get_key< KEY_TYPE > >::remove\(\)](#).

Here is the caller graph for this function:



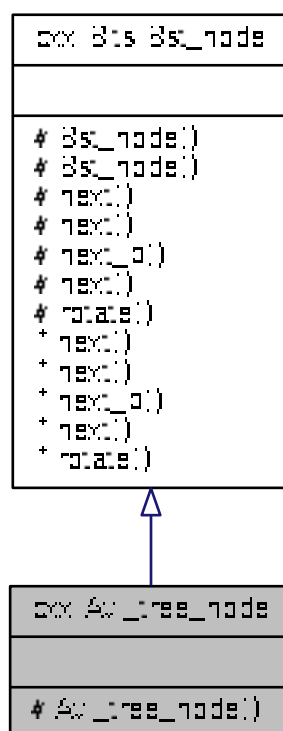
The documentation for this class was generated from the following file:

- [l4/cxx/avl_tree](#)

14.5 cxx::Avl_tree_node Class Reference

Node of an AVL tree.

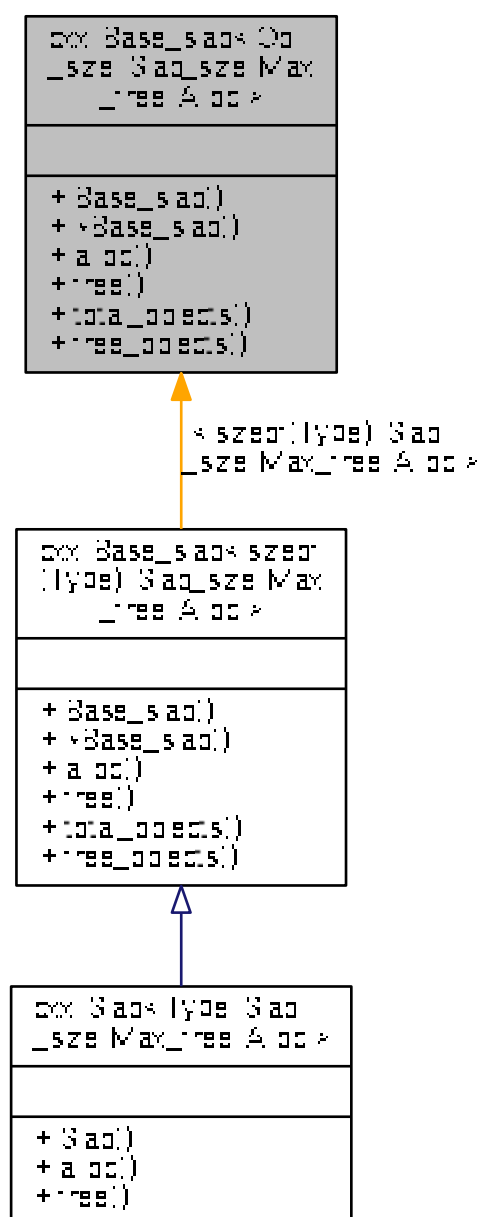
Inheritance diagram for cxx::Avl_tree_node:



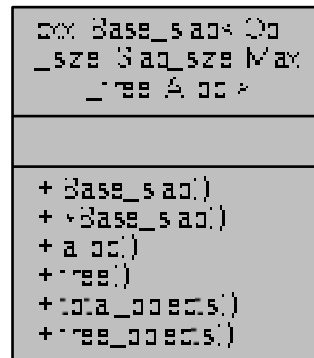
14.6 `cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >` Class Template Reference

Basic slab allocator.

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 >`:



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 allocator for the slab caches.

Public Member Functions

- unsigned `total_objects` () const throw ()
Get the total number of objects managed by the slab allocator.
- unsigned `free_objects` () const throw ()
Get the total number of objects managed by the slab allocator.

14.6.1 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.

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 cache (in bytes).
<i>Max_free</i>	The maximum number of free slab caches. When this limit is reached slab caches are freed.
<i>Alloc</i>	The allocator that is used to allocate the slab caches.

Definition at line 40 of file `slab_alloc`.

14.6.2 Member Enumeration Documentation

14.6.2.1 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 cache.
<code>objects_per_slab</code>	objects per slab cache.
<code>max_free_slabs</code>	maximum number of free slab caches.

Definition at line 63 of file [slab_alloc](#).

14.6.3 Member Function Documentation

14.6.3.1 `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 throw
() [inline]
```

Get the total number of objects managed by the slab allocator.

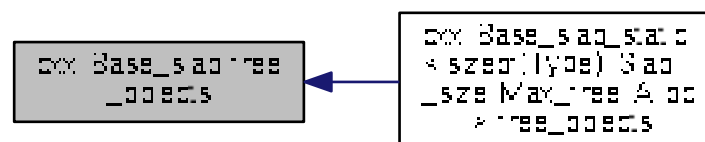
Returns

The number of objects managed by the allocator (including the free objects).

Definition at line 271 of file [slab_alloc](#).

Referenced by [cxx::Base_slab_static< sizeof\(Type\), Slab_size, Max_free, Alloc >::free_objects\(\)](#).

Here is the caller graph for this function:



14.6.3.2 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 throw
() [inline]
```

Get the total number of objects managed by the slab allocator.

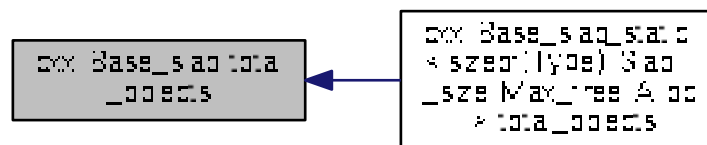
Returns

The number of objects managed by the allocator (including the free objects).

Definition at line 263 of file [slab_alloc](#).

Referenced by [cxx::Base_slab_static< sizeof\(Type\), Slab_size, Max_free, Alloc >::total_objects\(\)](#).

Here is the caller graph for this function:



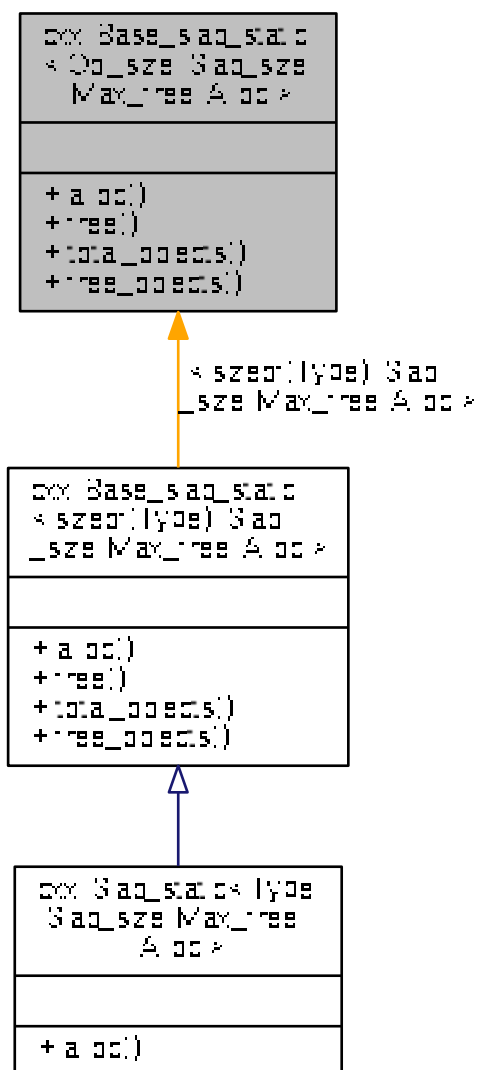
The documentation for this class was generated from the following file:

- `I4/cxx/slab_alloc`

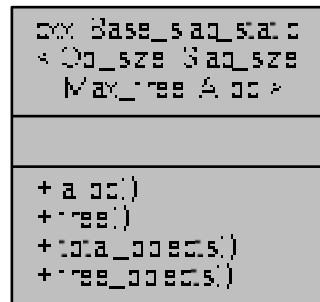
14.7 cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc > Class Template Reference

Merged slab allocator (allocators for objects of the same size are merged together).

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` () throw ()
Allocate an object.
- void `free` (void *p) throw ()
Free the given object (p).
- unsigned `total_objects` () const throw ()
Get the total number of objects managed by the slab allocator.
- unsigned `free_objects` () const throw ()
Get the number of free objects in the slab allocator.

14.7.1 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_static< Obj_size, Slab_size, Max_free, Alloc >
```

Merged slab allocator (allocators for objects of the same size are merged together).

Parameters

<i>Obj_size</i>	The size of an object managed by the slab allocator.
<i>Slab_size</i>	The size of a slab cache.
<i>Max_free</i>	The maximum number of free slab caches.
<i>Alloc</i>	The allocator for the slab caches.

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 348 of file `slab_alloc`.

14.7.2 Member Enumeration Documentation

14.7.2.1 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 cache.
<code>objects_per_slab</code>	number of objects per slab cache.
<code>max_free_slabs</code>	maximum number of free slab caches.

Definition at line 355 of file `slab_alloc`.

14.7.3 Member Function Documentation

14.7.3.1 `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_static< Obj_size, Slab_size, Max_free, Alloc >::alloc ( ) throw ( ) [inline]
```

Allocate an object.

Definition at line 365 of file `slab_alloc`.

14.7.3.2 `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_static< Obj_size, Slab_size, Max_free, Alloc >::free (
    void * p ) throw ( ) [inline]
```

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 371 of file [slab_alloc](#).

14.7.3.3 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\_static< Obj_size, Slab_size, Max_free, Alloc >::free_objects ( ) const
throw )    [inline]
```

Get the number of free objects in the slab allocator.

Returns

The number of free objects in all free and partially used slab caches managed by this allocator.

Note

The value is the merged value for all equal parameterized [Base_slab_static](#) instances.

Definition at line 389 of file [slab_alloc](#).

14.7.3.4 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\_static< Obj_size, Slab_size, Max_free, Alloc >::total_objects ( )
const throw )    [inline]
```

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 380 of file [slab_alloc](#).

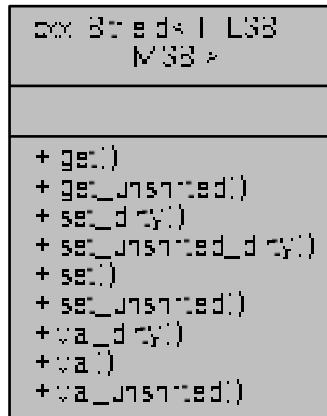
The documentation for this class was generated from the following file:

- [l4/cxx/slab_alloc](#)

14.8 `cxx::Bitfield< T, LSB, MSB >` Class Template Reference

Definition for a member (part) of a bit field.

Collaboration diagram for `cxx::Bitfield< T, LSB, MSB >`:



Data Structures

- class [Value](#)
Internal helper type.
- class [Value_base](#)
Internal helper type.
- class [Value_unshifted](#)
Internal helper type.

Public Types

- enum { `Bits` = `MSB + 1 - LSB`, `Lsb` = `LSB`, `Msb` = `MSB` }
- enum `Masks` : `T` { `Low_mask` = `((T)~0ULL) >> (sizeof(T)*8 - Bits)`, `Mask` = `Low_mask << Lsb` }
- 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< T & >` `Ref`
Reference type to access the bits inside a raw bit field.
- typedef `Value< T const >` `Val`
`Value` type to access the bits inside a raw bit field.
- typedef `Value_unshifted< T & >` `Ref_unshifted`
Reference type to access the bits inside a raw bit field (in place).
- typedef `Value_unshifted< T const >` `Val_unshifted`
`Value` type to access the bits inside a raw bit field (in place).

Static Public Member Functions

- static [Bits_type](#) [get](#) ([Shift_type](#) val)
Get the bits out of val.
- static [T](#) [get_unshifted](#) ([Shift_type](#) val)
Get the bits in place out of val.
- static [T](#) [set_dirty](#) ([T](#) dest, [Shift_type](#) val)
Set the bits corresponding to val.
- static [T](#) [set_unshifted_dirty](#) ([T](#) dest, [Shift_type](#) val)
Set the bits corresponding to val.
- static [T](#) [set](#) ([T](#) dest, [Bits_type](#) val)
Set the bits corresponding to val.
- static [T](#) [set_unshifted](#) ([T](#) dest, [Shift_type](#) val)
Set the bits corresponding to val.
- static [T](#) [val_dirty](#) ([Shift_type](#) val)
Get the shifted bits for val.
- static [T](#) [val](#) ([Bits_type](#) val)
Get the shifted bits for val.
- static [T](#) [val_unshifted](#) ([Shift_type](#) val)
Get the shifted bits for val.

14.8.1 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 mos significant bit if our bits.

Definition at line 34 of file [bitfield](#).

14.8.2 Member Typedef Documentation

14.8.2.1 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 78 of file [bitfield](#).

14.8.2.2 Ref

```
template<typename T , unsigned LSB, unsigned MSB>
typedef Value<T&> cxx::Bitfield< T, LSB, MSB >::Ref
```

Reference type to access the bits inside a raw bit field.

Definition at line 218 of file [bitfield](#).

14.8.2.3 Ref_unshifted

```
template<typename T , unsigned LSB, unsigned MSB>
typedef Value_unshifted<T&> cxx::Bitfield< T, LSB, MSB >::Ref_unshifted
```

Reference type to access the bits inside a raw bit field (in place).

Definition at line 223 of file [bitfield](#).

14.8.2.4 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 86 of file [bitfield](#).

14.8.2.5 Val

```
template<typename T , unsigned LSB, unsigned MSB>
typedef Value<T const> cxx::Bitfield< T, LSB, MSB >::Val
```

`Value` type to access the bits inside a raw bit field.

Definition at line 220 of file [bitfield](#).

14.8.2.6 Val_unshifted

```
template<typename T , unsigned LSB, unsigned MSB>
typedef Value_unshifted<T const> cxx::Bitfield< T, LSB, MSB >::Val_unshifted
```

`Value` type to access the bits inside a raw bit field (in place).

Definition at line 225 of file [bitfield](#).

14.8.3 Member Enumeration Documentation

14.8.3.1 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 58 of file [bitfield](#).

14.8.3.2 Masks

```
template<typename T , unsigned LSB, unsigned MSB>
enum cxx::Bitfield::Masks : T
```

Enumerator

Low_mask	Mask value to get Bits bits.
Mask	Mask value to the bits out of a <i>T</i> .

Definition at line 65 of file [bitfield](#).

14.8.4 Member Function Documentation**14.8.4.1 get()**

```
template<typename T , unsigned LSB, unsigned MSB>
static Bits\_type cxx::Bitfield< T, LSB, MSB >::get (
    Shift\_type val ) [inline], [static]
```

Get the bits out of *val*.

Parameters

<i>val</i>	the raw value of the whole bit field.
------------	---------------------------------------

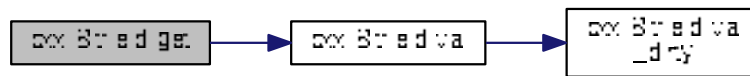
Returns

the bits form [Lsb](#) to [Msb](#) shifted to the right.

Definition at line 100 of file [bitfield](#).

References [cxx::Bitfield](#)< T, LSB, MSB >::Low_mask, [cxx::Bitfield](#)< T, LSB, MSB >::Lsb, and [cxx::Bitfield](#)< T, LSB, MSB >::val().

Here is the call graph for this function:



14.8.4.2 `get_unshifted()`

```
template<typename T , unsigned LSB, unsigned MSB>
static T cxx::Bitfield< T, LSB, MSB >::get_unshifted (
    Shift_type val ) [inline], [static]
```

Get the bits in place out of *val*.

Parameters

<i>val</i>	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 110 of file `bitfield`.

References `cxx::Bitfield< T, LSB, MSB >::Mask`.

14.8.4.3 `set()`

```
template<typename T , unsigned LSB, unsigned MSB>
static T cxx::Bitfield< T, LSB, MSB >::set (
    T 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.

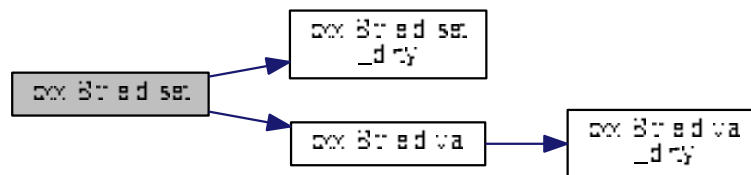
Returns

the new value of the whole bit field.

Definition at line 146 of file [bitfield](#).

References [cxx::Bitfield< T, LSB, MSB >::Low_mask](#), [cxx::Bitfield< T, LSB, MSB >::set_dirty\(\)](#), and [cxx::Bitfield< T, LSB, MSB >::val\(\)](#).

Here is the call graph for this function:

**14.8.4.4 set_dirty()**

```

template<typename T , unsigned LSB, unsigned MSB>
static T cxx::Bitfield< T, LSB, MSB >::set_dirty (
    T 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 to set into the bits.

Returns

the new value of the whole bit field.

Precondition

val must contain not more than bits than [Bits](#).

Note

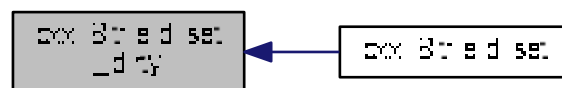
This function does not mask *val* to the right number of bits.

Definition at line 120 of file [bitfield](#).

References [cxx::Bitfield< T, LSB, MSB >::Lsb](#), and [cxx::Bitfield< T, LSB, MSB >::Mask](#).

Referenced by [cxx::Bitfield< T, LSB, MSB >::set\(\)](#).

Here is the caller graph for this function:

**14.8.4.5 `set_unshifted()`**

```

template<typename T , unsigned LSB, unsigned MSB>
static T cxx::Bitfield< T, LSB, MSB >::set_unshifted (
    T 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 Lsb bits to the left that shall be set into the bits.

Returns

the new value of the whole bit field.

Definition at line 155 of file [bitfield](#).

References [cxx::Bitfield< T, LSB, MSB >::Mask](#), and [cxx::Bitfield< T, LSB, MSB >::set_unshifted_dirty\(\)](#).

Here is the call graph for this function:



14.8.4.6 set_unshifted_dirty()

```
template<typename T , unsigned LSB, unsigned MSB>
static T cxx::Bitfield< T, LSB, MSB >::set_unshifted_dirty (
    T 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 Lsb bits to the left that shall be set into the bits.

Returns

the new value of the whole bit field.

Precondition

val must contain not more than bits than [Bits](#) shifted [Lsb](#) bits to the left.

Note

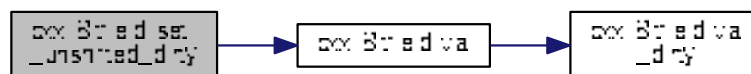
This function does not mask *val* to the right number of bits.

Definition at line [135](#) of file [bitfield](#).

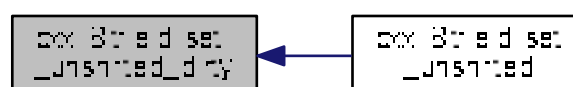
References [cxx::Bitfield< T, LSB, MSB >::Mask](#), and [cxx::Bitfield< T, LSB, MSB >::val\(\)](#).

Referenced by [cxx::Bitfield< T, LSB, MSB >::set_unshifted\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



14.8.4.7 `val()`

```
template<typename T , unsigned LSB, unsigned MSB>
static T cxx::Bitfield< T, LSB, MSB >::val (
    Bits_type val ) [inline], [static]
```

Get the shifted bits for *val*.

Parameters

<i>val</i>	the value to set into the bits.
------------	---------------------------------

Returns

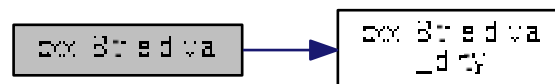
the raw bit field value containing.

Definition at line 170 of file `bitfield`.

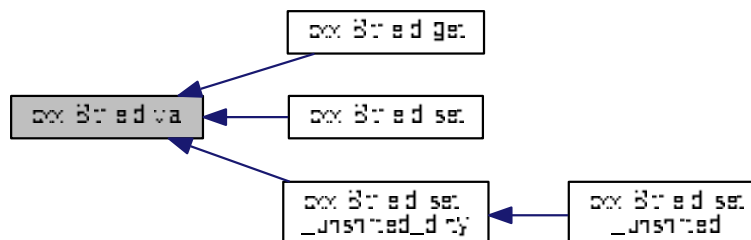
References `cxx::Bitfield< T, LSB, MSB >::Low_mask`, and `cxx::Bitfield< T, LSB, MSB >::val_dirty`.

Referenced by `cxx::Bitfield< T, LSB, MSB >::get()`, `cxx::Bitfield< T, LSB, MSB >::set()`, and `cxx::Bitfield< T, LSB, MSB >::set_unshifted_dirty()`.

Here is the call graph for this function:



Here is the caller graph for this function:



14.8.4.8 val_dirty()

```
template<typename T , unsigned LSB, unsigned MSB>
static T cxx::Bitfield< T, LSB, MSB >::val_dirty (
    Shift_type val ) [inline], [static]
```

Get the shifted bits for *val*.

Parameters

<i>val</i>	the value to set into the bits.
------------	---------------------------------

Returns

the raw bit field value containing.

Precondition

val must contain not more than bits than [Bits](#).

Note

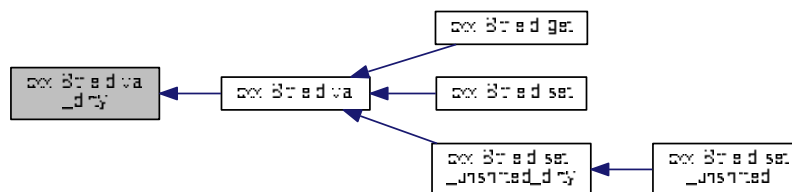
This function does not mask *val* to the right number of bits.

Definition at line [164](#) of file [bitfield](#).

References [cxx::Bitfield< T, LSB, MSB >::Lsb](#).

Referenced by [cxx::Bitfield< T, LSB, MSB >::val\(\)](#).

Here is the caller graph for this function:



14.8.4.9 val_unshifted()

```
template<typename T , unsigned LSB, unsigned MSB>
static T cxx::Bitfield< T, LSB, MSB >::val_unshifted (
    Shift_type val ) [inline], [static]
```

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 containing.

Definition at line 177 of file [bitfield](#).

References [cxx::Bitfield< T, LSB, MSB >::Mask](#).

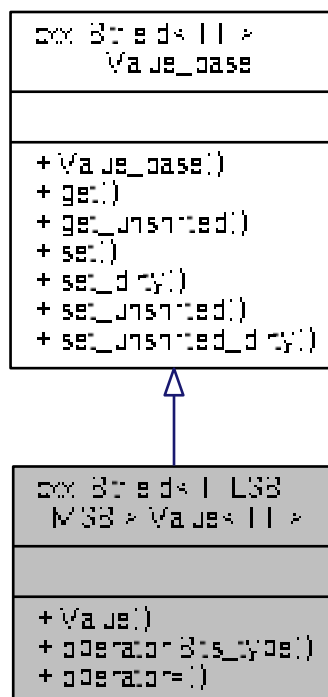
The documentation for this class was generated from the following file:

- `I4/cxx/bitfield`

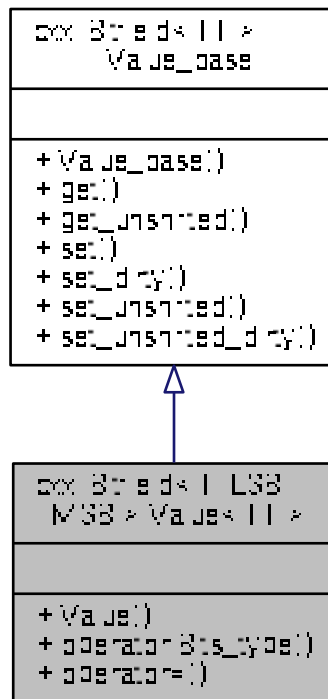
14.9 `cxx::Bitfield< T, LSB, MSB >::Value< TT >` Class Template Reference

Internal helper type.

Inheritance diagram for `cxx::Bitfield< T, LSB, MSB >::Value< TT >`:



Collaboration diagram for `cxx::Bitfield< T, LSB, MSB >::Value< TT >`:



14.9.1 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 199 of file [bitfield](#).

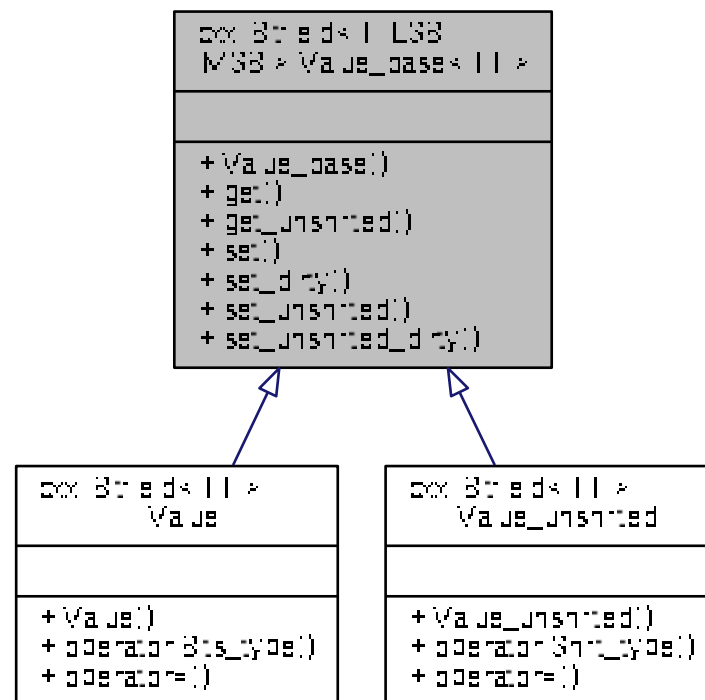
The documentation for this class was generated from the following file:

- `I4/cxx/bitfield`

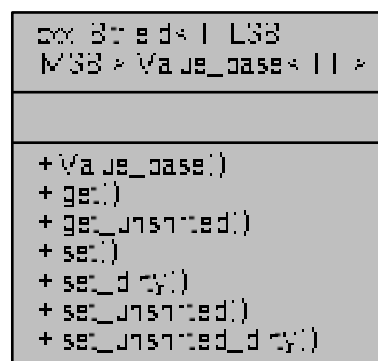
14.10 `cxx::Bitfield< T, LSB, MSB >::Value_base< TT >` Class Template Reference

Internal helper type.

Inheritance diagram for cxx::Bitfield< T, LSB, MSB >::Value_base< TT >:



Collaboration diagram for cxx::Bitfield< T, LSB, MSB >::Value_base< TT >:



14.10.1 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 181 of file [bitfield](#).

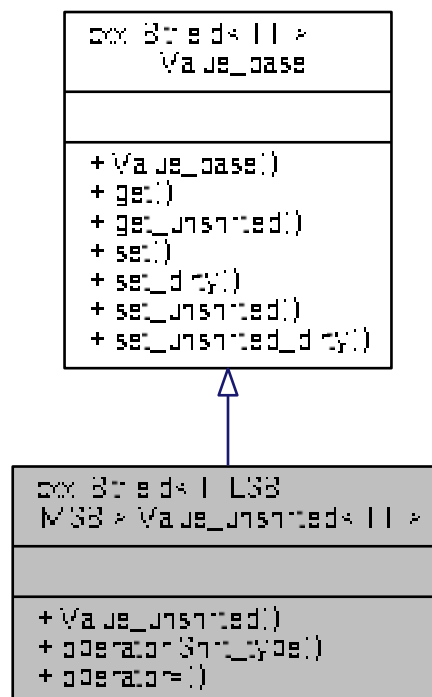
The documentation for this class was generated from the following file:

- I4/cxx/bitfield

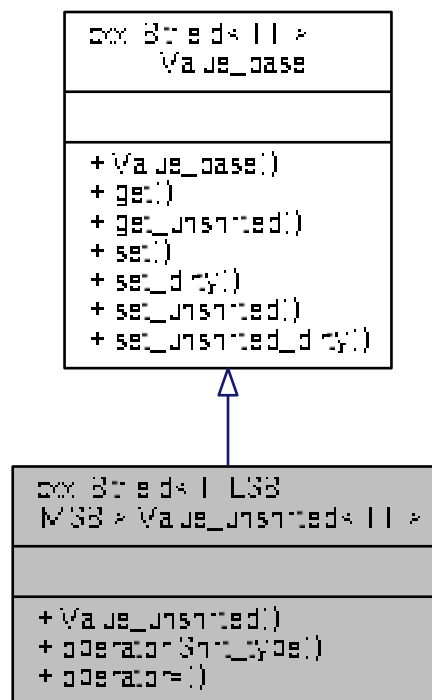
14.11 cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT > Class Template Reference

Internal helper type.

Inheritance diagram for cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT >:



Collaboration diagram for `cxx::Bitfield< T, LSB, MSB >::Value_unshifted< TT >`:



14.11.1 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 209 of file [bitfield](#).

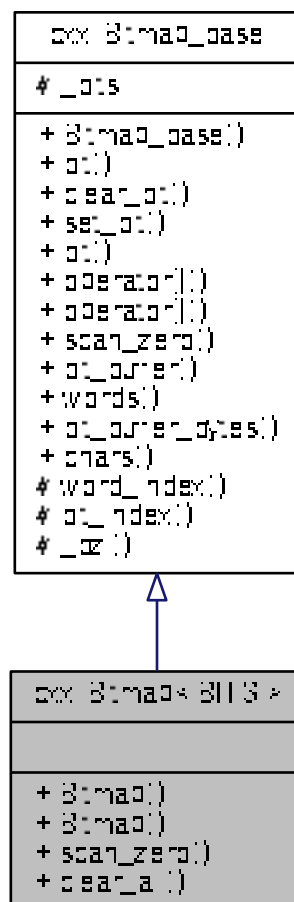
The documentation for this class was generated from the following file:

- `I4/cxx/bitfield`

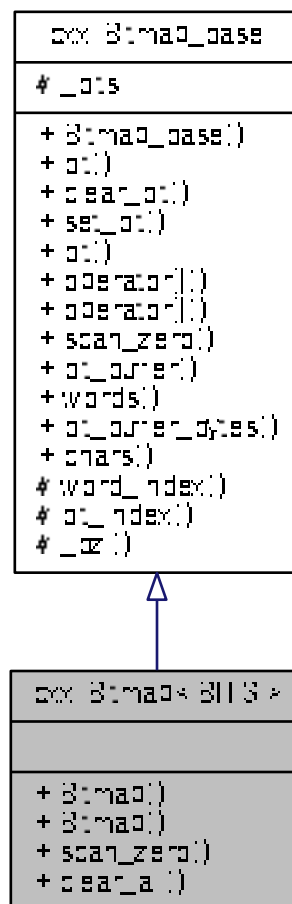
14.12 `cxx::Bitmap< BITS >` Class Template Reference

A static bit map.

Inheritance diagram for `cxx::Bitmap< BITS >`:



Collaboration diagram for `cx::Bitmap< BITS >`:



Public Member Functions

- `Bitmap ()` throw ()
Create a bitmap with *BITS* bits.
- long `scan_zero` (long start_bit=0) const throw ()
Scan for the first zero bit.

Additional Inherited Members

14.12.1 Detailed Description

```
template<int BITS>
class cx::Bitmap< BITS >
```

A static bit map.

Parameters

<i>BITS</i>	the number of bits that shall be in the bitmap.
-------------	---

Definition at line 180 of file [bitmap](#).

14.12.2 Constructor & Destructor Documentation

14.12.2.1 Bitmap()

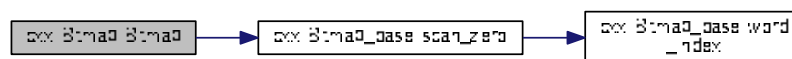
```
template<int BITS>
cxx::Bitmap< BITS >::Bitmap ( ) throw ( ) [inline]
```

Create a bitmap with *BITS* bits.

Definition at line 187 of file [bitmap](#).

References [cxx::Bitmap_base::scan_zero\(\)](#).

Here is the call graph for this function:



14.12.3 Member Function Documentation

14.12.3.1 scan_zero()

```
template<int BITS>
long cxx::Bitmap< BITS >::scan_zero (
    long start_bit = 0 ) const throw ( ) [inline]
```

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

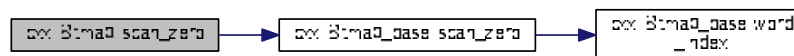
<code>>=</code>	0 Number of first zero bit found.
<code>-1</code>	All bits at <code>start_bit</code> or higher are set.

Compared to [Bitmap_base::scan_zero\(\)](#), the upper bound is set to BITS.

Definition at line 285 of file [bitmap](#).

References [cxx::Bitmap_base::scan_zero\(\)](#).

Here is the call graph for this function:



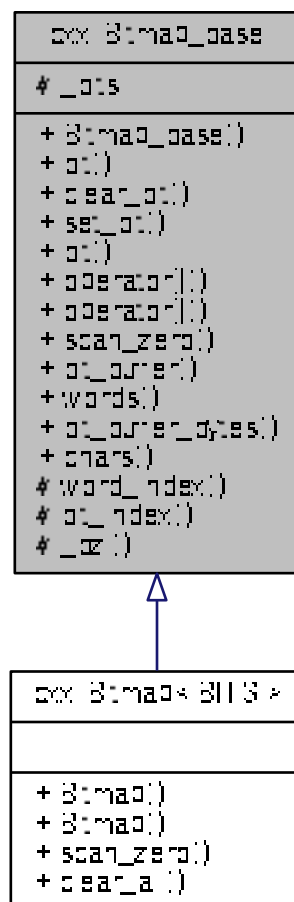
The documentation for this class was generated from the following file:

- `I4/cxx/bitmap`

14.13 `cxx::Bitmap_base` Class Reference

Basic bitmap abstraction.

Inheritance diagram for `cxx::Bitmap_base`:



Collaboration diagram for cxx::Bitmap_base:

cxx::Bitmap_base
_bits
+ Bitmap_base()
+ bit()
+ clear_bit()
+ set_bit()
+ bit()
+ operator[]()
+ operator[]()
+ scan_zero()
+ bit_index()
+ words()
+ bit_index_bytes()
+ chars()
word_index()
bit_index()
_ix()

Data Structures

- class [Bit](#)
A writeable bit in a bitmap.
- class [Char](#)
Helper abstraction for a byte contained in the bitmap.
- class [Word](#)
Helper abstraction for a word contained in the bitmap.

Public Member Functions

- void [bit](#) (long bit, bool on) throw ()
Set the value of bit bit to on.
- void [clear_bit](#) (long bit) throw ()
Clear bit bit.
- void [set_bit](#) (long bit) throw ()
Set bit bit.
- [word_type](#) [bit](#) (long bit) const throw ()
Get the truth value of a bit.
- [word_type](#) [operator\[\]](#) (long bit) const throw ()
Get the bit at index bit.
- [Bit](#) [operator\[\]](#) (long bit) throw ()
Get the lvalue for the bit at index bit.
- long [scan_zero](#) (long max_bit, long start_bit=0) const throw ()
Scan for the first zero bit.

Static Public Member Functions

- static long [words](#) (long bits) throw ()
Get the number of Words that are used for the bitmap.
- static long [chars](#) (long bits) throw ()
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.

14.13.1 Detailed Description

Basic bitmap abstraction.

This abstraction keeps a pointer to a memory area that is used as bitmap.

Definition at line 30 of file [bitmap](#).

14.13.2 Member Enumeration Documentation

14.13.2.1 anonymous enum

```
anonymous enum [protected]
```

Enumerator

W_bits	number of bits in word_type
C_bits	number of bits in char

Definition at line 38 of file [bitmap](#).

14.13.3 Member Function Documentation

14.13.3.1 bit() [1/2]

```
void cxx::Bitmap_base::bit (
    long bit,
    bool on ) throw ()    [inline]
```

Set the value of bit *bit* to *on*.

Parameters

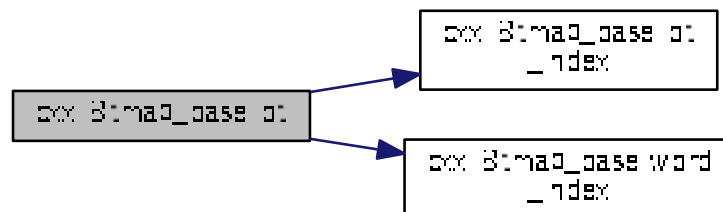
<i>bit</i>	the number of the bit
<i>on</i>	the boolean value that shall be assigned to the bit.

Definition at line 211 of file [bitmap](#).

References [bit_index\(\)](#), and [word_index\(\)](#).

Referenced by [operator\[\]\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



14.13.3.2 `bit()` [2/2]

```
unsigned long cxx::Bitmap_base::bit (
    long bit ) const throw ()    [inline]
```

Get the truth value of a bit.

Parameters

<i>bit</i>	the number of the bit to read.
------------	--------------------------------

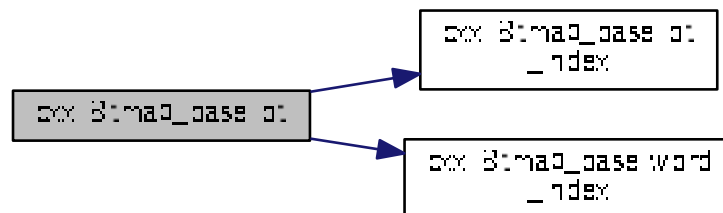
Returns

0 if *bit* is not set, != 0 if *bit* is set.

Definition at line 238 of file [bitmap](#).

References [bit_index\(\)](#), and [word_index\(\)](#).

Here is the call graph for this function:



14.13.3.3 `bit_index()`

```
static 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

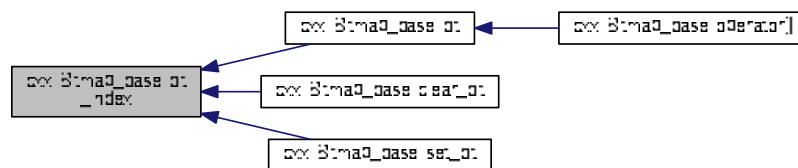
the bit index within word_type (bit % W_bits).

Definition at line 61 of file [bitmap](#).

References [W_bits](#).

Referenced by [bit\(\)](#), [clear_bit\(\)](#), and [set_bit\(\)](#).

Here is the caller graph for this function:

**14.13.3.4 chars()**

```
static long cxx::Bitmap_base::chars (
    long bits ) throw () [inline], [static]
```

Get the number of chars that are used for the bitmap.

Definition at line 98 of file [bitmap](#).

References [C_bits](#).

14.13.3.5 clear_bit()

```
void cxx::Bitmap_base::clear_bit (
    long bit ) throw () [inline]
```

Clear bit *bit*.

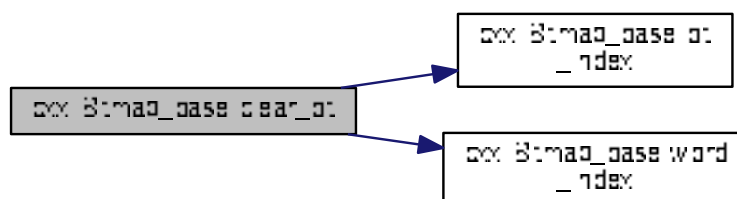
Parameters

<i>bit</i>	the number of the bit to clear.
------------	---------------------------------

Definition at line 220 of file [bitmap](#).

References [bit_index\(\)](#), and [word_index\(\)](#).

Here is the call graph for this function:



14.13.3.6 operator[]() [1/2]

```
word_type cxx::Bitmap_base::operator[] (
    long bit ) const throw ()    [inline]
```

Get the bit at index *bit*.

Parameters

<i>bit</i>	the number of the bit to read.
------------	--------------------------------

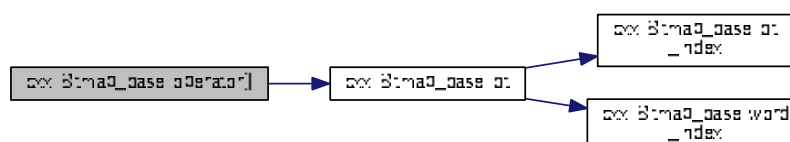
Returns

0 if *bit* is not set, != 0 if *bit* is set.

Definition at line 143 of file [bitmap](#).

References [bit\(\)](#).

Here is the call graph for this function:



14.13.3.7 operator[]() [2/2]

```
Bit cxx::Bitmap_base::operator[] (
    long bit ) throw )    [inline]
```

Get the lvalue for the bit at index *bit*.

Parameters

<i>bit</i>	the number.
------------	-------------

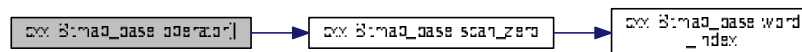
Returns

lvalue for *bit*

Definition at line 151 of file [bitmap](#).

References [_bits](#), and [scan_zero\(\)](#).

Here is the call graph for this function:



14.13.3.8 scan_zero()

```
long cxx::Bitmap_base::scan_zero (
    long max_bit,
    long start_bit = 0 ) const throw )    [inline]
```

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

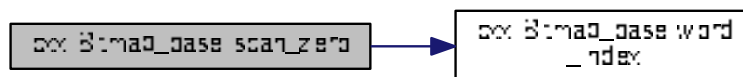
<i>>=</i>	0 Number of first zero bit found.
<i>-1</i>	All bits between <i>start_bit</i> and <i>max_bit</i> are set.

Definition at line 259 of file [bitmap](#).

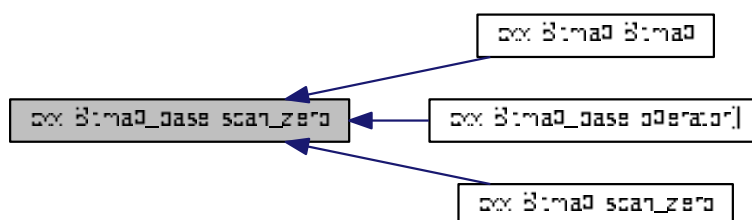
References [W_bits](#), and [word_index\(\)](#).

Referenced by [cxx::Bitmap< BITS >::Bitmap\(\)](#), [operator\[\]\(\)](#), and [cxx::Bitmap< BITS >::scan_zero\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



14.13.3.9 set_bit()

```
void cxx::Bitmap_base::set_bit (
    long bit ) throw () [inline]
```

Set bit *bit*.

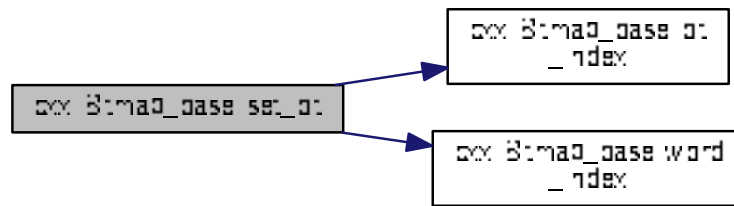
Parameters

<i>bit</i>	the number of the bit to set,
------------	-------------------------------

Definition at line 229 of file [bitmap](#).

References [bit_index\(\)](#), and [word_index\(\)](#).

Here is the call graph for this function:



14.13.3.10 word_index()

```
static 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

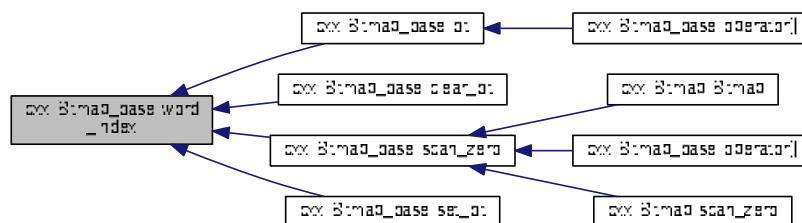
the index in [Bitmap_base::_bits](#) for the given bit (bit / W_bits).

Definition at line 54 of file [bitmap](#).

References [W_bits](#).

Referenced by [bit\(\)](#), [clear_bit\(\)](#), [scan_zero\(\)](#), and [set_bit\(\)](#).

Here is the caller graph for this function:



14.13.3.11 words()

```
static long cxx::Bitmap_base::words (
    long bits ) throw ()    [inline], [static]
```

Get the number of *Words* that are used for the bitmap.

Definition at line 81 of file [bitmap](#).

References [W_bits](#).

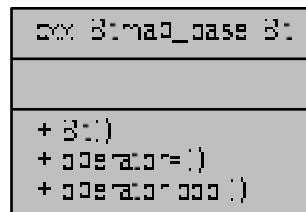
The documentation for this class was generated from the following file:

- I4/cxx/bitmap

14.14 cxx::Bitmap_base::Bit Class Reference

A writeable bit in a bitmap.

Collaboration diagram for cxx::Bitmap_base::Bit:



14.14.1 Detailed Description

A writeable bit in a bitmap.

Definition at line 66 of file [bitmap](#).

The documentation for this class was generated from the following file:

- I4/cxx/bitmap

14.15 `cxx::Bitmap_base::Char< BITS >` Class Template Reference

Helper abstraction for a byte contained in the bitmap.

Collaboration diagram for `cxx::Bitmap_base::Char< BITS >`:



14.15.1 Detailed Description

```
template<long BITS>
class cxx::Bitmap_base::Char< BITS >
```

Helper abstraction for a byte contained in the bitmap.

Definition at line [103](#) of file [bitmap](#).

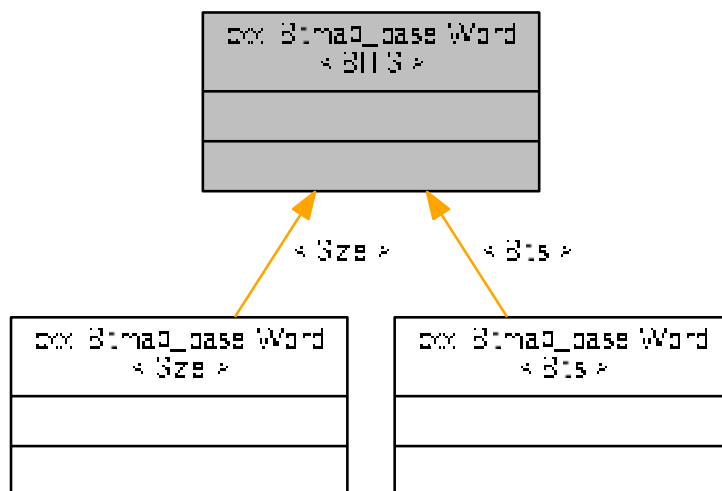
The documentation for this class was generated from the following file:

- `I4/cxx/bitmap`

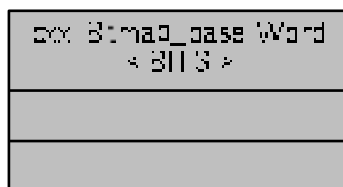
14.16 `cxx::Bitmap_base::Word< BITS >` Class Template Reference

Helper abstraction for a word contained in the bitmap.

Inheritance diagram for `cxx::Bitmap_base::Word< BITS >`:



Collaboration diagram for `cxx::Bitmap_base::Word< BITS >`:



14.16.1 Detailed Description

```
template<long BITS>
class cxx::Bitmap_base::Word< BITS >
```

Helper abstraction for a word contained in the bitmap.

Definition at line 87 of file [bitmap](#).

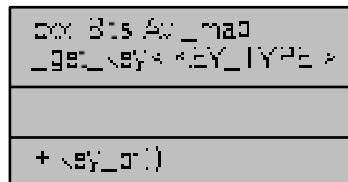
The documentation for this class was generated from the following file:

- `I4/cxx/bitmap`

14.17 cxx::Bits::Avl_map_get_key< KEY_TYPE > Struct Template Reference

Key-getter for [Avl_map](#).

Collaboration diagram for cxx::Bits::Avl_map_get_key< KEY_TYPE >:



14.17.1 Detailed Description

```
template<typename KEY_TYPE>
struct cxx::Bits::Avl_map_get_key< KEY_TYPE >
```

Key-getter for [Avl_map](#).

Definition at line 36 of file [avl_map](#).

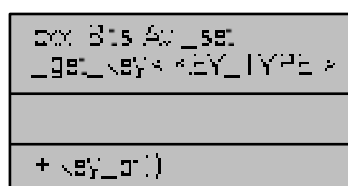
The documentation for this struct was generated from the following file:

- [l4/cxx/avl_map](#)

14.18 cxx::Bits::Avl_set_get_key< KEY_TYPE > Struct Template Reference

Internal, key-getter for [Avl_set](#) nodes.

Collaboration diagram for cxx::Bits::Avl_set_get_key< KEY_TYPE >:



14.18.1 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 93 of file [avl_set](#).

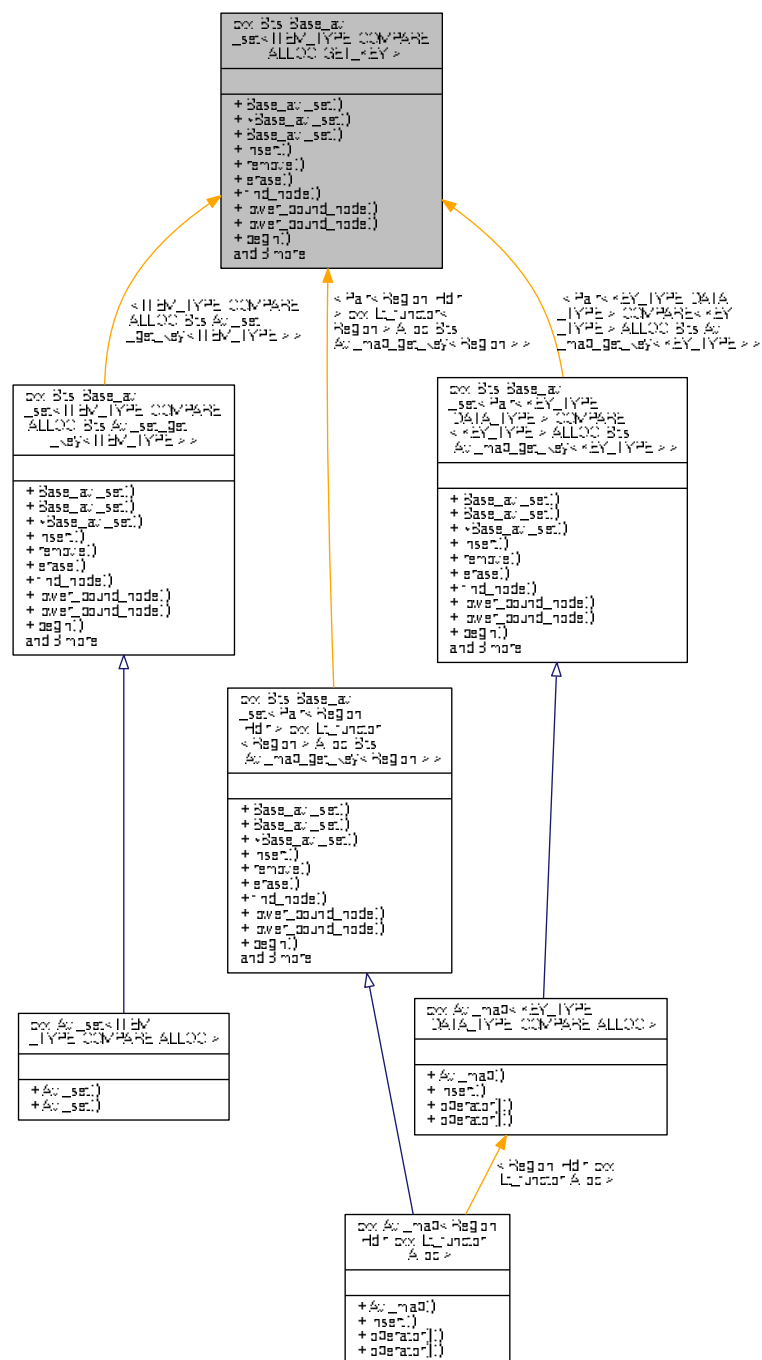
The documentation for this struct was generated from the following file:

- [l4/cxx/avl_set](#)

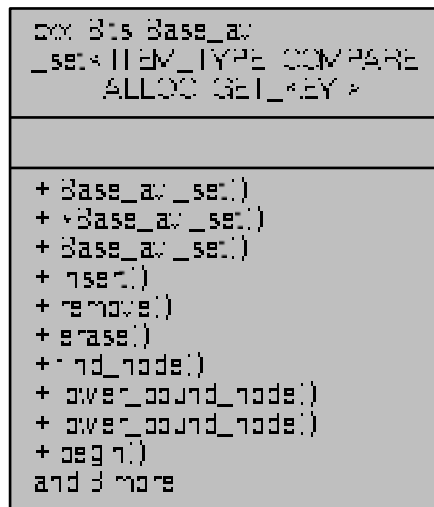
14.19 `cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >` Class Template Reference

Internal: AVL set with internally managed nodes.

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

- 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)
Create a copy of an AVL-tree based set.
- `cxx::Pair< Iterator, int >` [insert](#) ([Item_type](#) const &item)
Insert an item into 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.

14.19.1 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 >
```

Internal: AVL set with internally managed nodes.

Use [Avl_set](#), [Avl_map](#), or [Avl_tree](#) in applications.

Template Parameters

<code>ITEM_TYPE</code>	The type of the items to be stored in the set.
<code>COMPARE</code>	The relation to define the partial order, default is to use operator '<'.
<code>ALLOC</code>	The allocator to use for the nodes of the AVL set.
<code>GET_KEY</code>	Sort-key getter (must provide the <code>Key_type</code> and sort-key for an item (of <code>ITEM_TYPE</code>)).

Definition at line 117 of file [avl_set](#).

14.19.2 Constructor & Destructor Documentation

14.19.2.1 Base_avl_set() [1/2]

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GE←
T_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 229 of file [avl_set](#).

14.19.2.2 Base_avl_set() [2/2]

```
template<typename Item , class Compare , template< typename A > class Alloc, typename KEY_T←
YPE >
cxx::Bits::Base_avl_set< Item, Compare, Alloc, KEY_TYPE >::Base_avl_set (
    Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY > const & o ) [inline]
```

Create a copy of an AVL-tree based set.

Parameters

<i>o</i>	The set to copy.
----------	------------------

Creates a deep copy of the set with all its items.

Definition at line 380 of file [avl_set](#).

14.19.3 Member Function Documentation

14.19.3.1 `begin()` [1/2]

```
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.

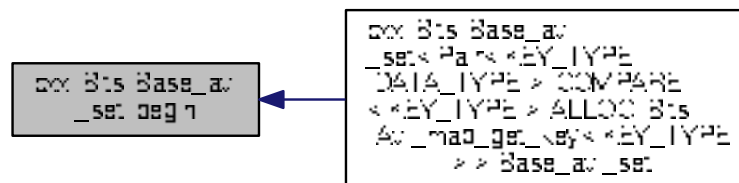
Returns

Constant forward iterator for the first element in the set.

Definition at line 330 of file [avl_set](#).

Referenced by [cxx::Bits::Base_avl_set< Pair< KEY_TYPE, DATA_TYPE >, COMPARE< KEY_TYPE >, ALLOC, Bits::Avl_map_get_key< KEY_TYPE > >::Base_avl_set\(\)](#).

Here is the caller graph for this function:

14.19.3.2 `begin()` [2/2]

```
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 341 of file [avl_set](#).

14.19.3.3 end() [1/2]

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GE←
T_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.

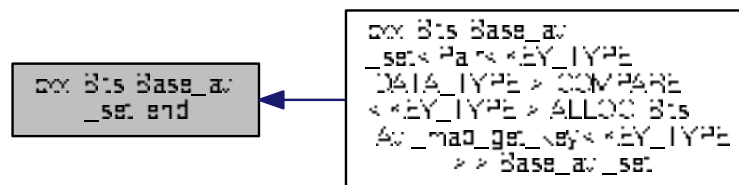
Returns

The end marker for the constant forward iterator.

Definition at line 335 of file [avl_set](#).

Referenced by [cxx::Bits::Base_avl_set< Pair< KEY_TYPE, DATA_TYPE >, COMPARE< KEY_TYPE >, ALLOC, Bits::Avl_map_get_key< KEY_TYPE > >::Base_avl_set\(\)](#).

Here is the caller graph for this function:



14.19.3.4 end() [2/2]

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GE←
T_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 346 of file [avl_set](#).

14.19.3.5 erase()

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GE←
T_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 298 of file [avl_set](#).

14.19.3.6 find_node()

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GE←
T_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 309 of file [avl_set](#).

14.19.3.7 insert()

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GE←
T_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 an element is already in the set. In both cases, *first* contains an iterator that points to the element. *second* may also be `-E_nomem` in 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 390 of file [avl_set](#).

14.19.3.8 lower_bound_node()

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GE↔
T_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 320 of file [avl_set](#).

14.19.3.9 rbegin() [1/2]

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GE↔
T_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 352 of file [avl_set](#).

14.19.3.10 rbegin() [2/2]

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GE↔
T_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 363 of file [avl_set](#).

14.19.3.11 remove()

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GE↵
T_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

0	Success
-E_noent	Item does not exist
-E_inval	Internal error

Definition at line 278 of file [avl_set](#).

14.19.3.12 rend() [1/2]

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GE↵
T_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 357 of file [avl_set](#).

14.19.3.13 rend() [2/2]

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GE↵
T_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 368 of file [avl_set](#).

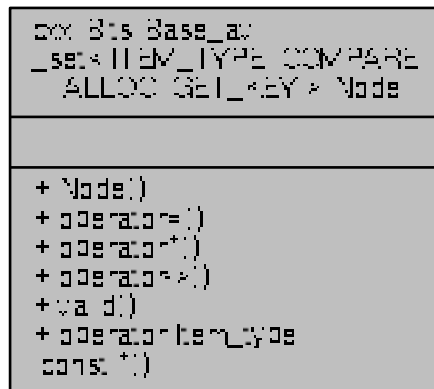
The documentation for this class was generated from the following file:

- [I4/cxx/avl_set](#)

14.20 cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node Class Reference

A smart pointer to a tree item.

Collaboration diagram for cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node:



Public Member Functions

- [Node](#) ()
Default construction for NIL pointer.
- [Node](#) & [operator=](#) ([Node](#) const &o)
Default assignment.
- [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.

14.20.1 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 155 of file [avl_set](#).

14.20.2 Member Function Documentation

14.20.2.1 `operator*()`

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GE←
T_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 175 of file [avl_set](#).

14.20.2.2 `operator->()`

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GE←
T_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 181 of file [avl_set](#).

14.20.2.3 `valid()`

```
template<typename ITEM_TYPE, class COMPARE, template< typename A > class ALLOC, typename GE←
T_KEY>
bool cxx::Bits::Base_avl_set< ITEM_TYPE, COMPARE, ALLOC, GET_KEY >::Node::valid ( ) const
[inline]
```

Validity check.

Returns

false if the pointer is NIL, true if valid.

Definition at line 187 of file [avl_set](#).

The documentation for this class was generated from the following file:

- [I4/cxx/avl_set](#)

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.

14.21.1 Detailed Description

```
template<typename POLICY>
class cxx::Bits::Basic_list< POLICY >
```

Internal: Common functions for all head-based list implementations.

Definition at line 50 of file [list_basics.h](#).

14.21.2 Member Function Documentation

14.21.2.1 `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.

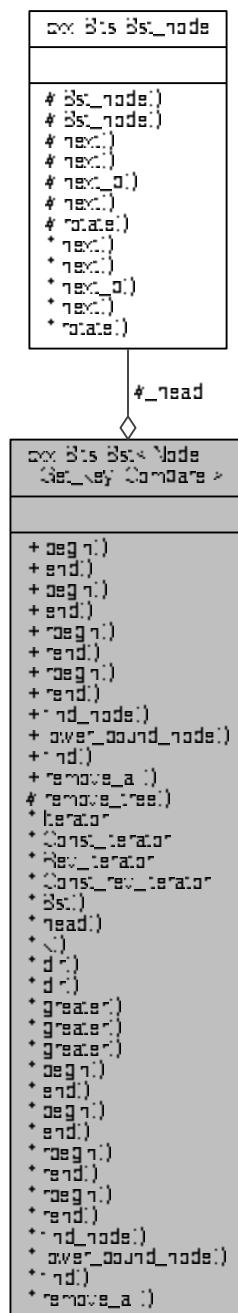
Definition at line 135 of file [list_basics.h](#).

14.21.2.2 `iter()`

```
template<typename POLICY>
static Const_iterator cxx::Bits::Basic_list< POLICY >::iter (
    Const_value_type c ) [inline], [static]
```

Return a const iterator that begins at the given element.

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.

14.22.1 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 40 of file [bst.h](#).

14.22.2 Member Function Documentation

14.22.2.1 `begin()` [1/2]

```
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.

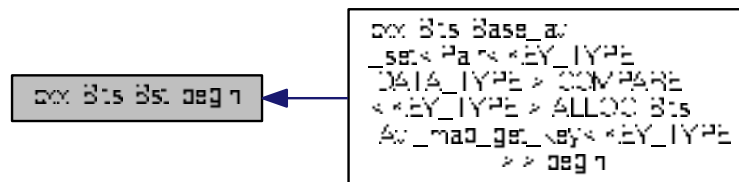
Returns

Constant forward iterator for the first element in the set.

Definition at line 179 of file [bst.h](#).

Referenced by [cxx::Bits::Base_avl_set< Pair< KEY_TYPE, DATA_TYPE >, COMPARE< KEY_TYPE >, ALLOC, Bits::Avl_map_get_key< KEY_TYPE > >::begin\(\)](#).

Here is the caller graph for this function:

14.22.2.2 `begin()` [2/2]

```
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 190 of file [bst.h](#).

14.22.2.3 `dir()` [1/2]

```
template<typename Node, typename Get_key, typename Compare>
static 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.

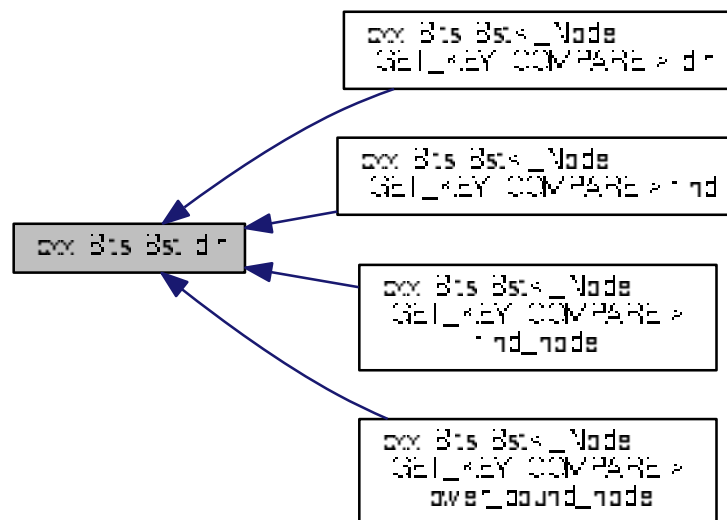
Return values

<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 118 of file [bst.h](#).

Referenced by [cxx::Bits::Bst< _Node, GET_KEY, COMPARE >::dir\(\)](#), [cxx::Bits::Bst< _Node, GET_KEY, COMPARE >::find\(\)](#), [cxx::Bits::Bst< _Node, GET_KEY, COMPARE >::find_node\(\)](#), and [cxx::Bits::Bst< _Node, GET_KEY, COMPARE >::lower_bound_node\(\)](#).

Here is the caller graph for this function:

14.22.2.4 `dir()` [2/2]

```

template<typename Node, typename Get_key, typename Compare>
static 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

<i>l</i>	is the key to look for.
<i>r</i>	is the node 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 135 of file [bst.h](#).

14.22.2.5 `end()` [1/2]

```
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.

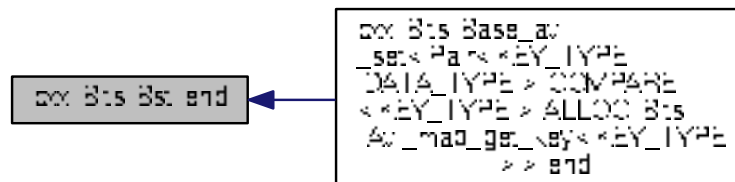
Returns

The end marker for the constant forward iterator.

Definition at line 184 of file [bst.h](#).

Referenced by [cxx::Bits::Base_avl_set< Pair< KEY_TYPE, DATA_TYPE >, COMPARE< KEY_TYPE >, ALLOC, Bits::Avl_map_get_key< KEY_TYPE > >::end\(\)](#).

Here is the caller graph for this function:



14.22.2.6 end() [2/2]

```
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 195 of file [bst.h](#).

14.22.2.7 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.
------------	---

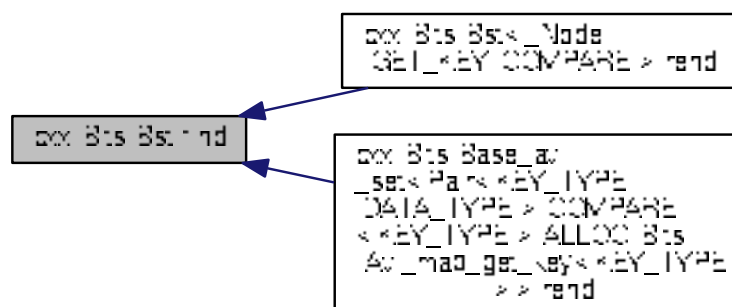
Returns

A valid iterator for the node with the given *key*, or an invalid iterator if *key* was not found.

Definition at line 312 of file [bst.h](#).

Referenced by [cxx::Bits::Bst< _Node, GET_KEY, COMPARE >::rend\(\)](#), and [cxx::Bits::Base_avl_set< Pair< KEY_TYPE, DATA_TYPE >, COMPARE< KEY_TYPE >, ALLOC, Bits::Avl_map_get_key< KEY_TYPE > >::rend\(\)](#).

Here is the caller graph for this function:



14.22.2.8 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.
------------	---

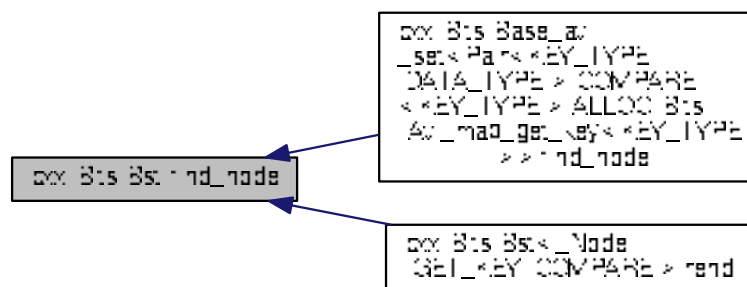
Returns

A pointer to the node with the given *key*, or NULL if *key* was not found.

Definition at line 276 of file [bst.h](#).

Referenced by [cxx::Bits::Base_avl_set< Pair< KEY_TYPE, DATA_TYPE >, COMPARE< KEY_TYPE >, ALLOC< KEY_TYPE >, Bits::Avl_map_get_key< KEY_TYPE > >::find_node\(\)](#), and [cxx::Bits::Bst< _Node, GET_KEY, COMPARE >::rend\(\)](#).

Here is the caller graph for this function:



14.22.2.9 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

<code>key</code>	The key value of the element to search.
------------------	---

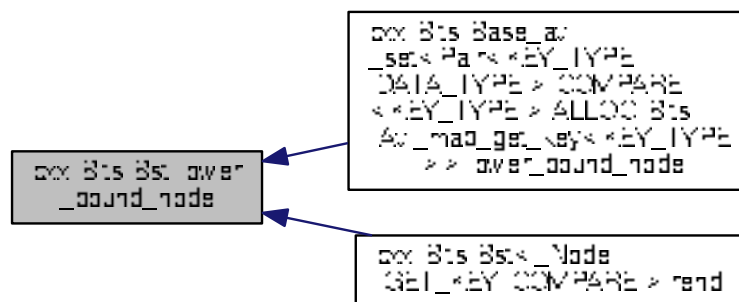
Returns

A pointer to the node with the given *key*, or NULL if *key* was not found.

Definition at line 292 of file `bst.h`.

Referenced by `cxx::Bits::Base_avl_set< Pair< KEY_TYPE, DATA_TYPE >, COMPARE< KEY_TYPE >, ALLOC, Bits::Avl_map_get_key< KEY_TYPE > >::lower_bound_node()`, and `cxx::Bits::Bst< _Node, GET_KEY, COMPARE >::rend()`.

Here is the caller graph for this function:

14.22.2.10 `rbegin()` [1/2]

```
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.

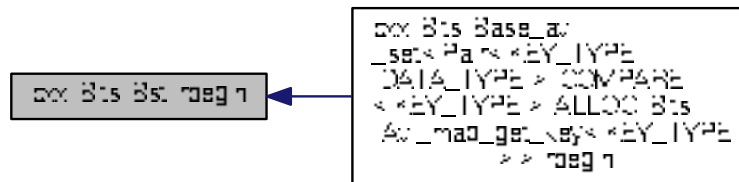
Returns

The constant backward iterator for the last element in the set.

Definition at line 201 of file `bst.h`.

Referenced by `cxx::Bits::Base_avl_set< Pair< KEY_TYPE, DATA_TYPE >, COMPARE< KEY_TYPE >, ALLOC, Bits::Avl_map_get_key< KEY_TYPE > >::rbegin()`.

Here is the caller graph for this function:



14.22.2.11 `rbegin()` [2/2]

```
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 212 of file [bst.h](#).

14.22.2.12 `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

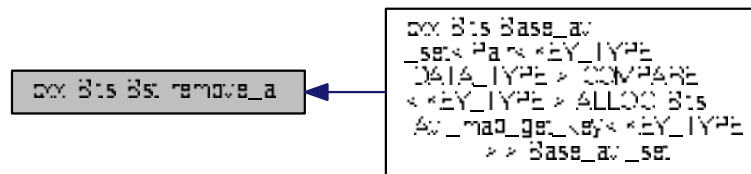
<i>callback</i>	Optional function to be called on each removed element.
-----------------	---

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 258 of file [bst.h](#).

Referenced by [cxx::Bits::Base_avl_set< Pair< KEY_TYPE, DATA_TYPE >, COMPARE< KEY_TYPE >, ALLOC, Bits::Avl_map_get_key< KEY_TYPE > >::Base_avl_set\(\)](#).

Here is the caller graph for this function:



14.22.2.13 remove_tree()

```

template<typename Node, typename Get_key, typename Compare>
template<typename FUNC >
static 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.

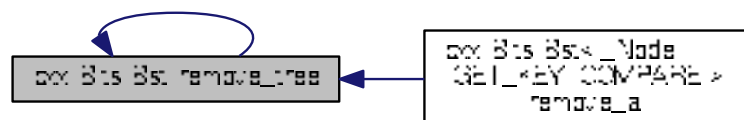
Parameters

<i>head</i>	Head of the the subtree to remove
<i>callback</i>	Optional function called on each removed element.

Definition at line 158 of file [bst.h](#).

Referenced by [cxx::Bits::Bst< _Node, GET_KEY, COMPARE >::remove_all\(\)](#), and [cxx::Bits::Bst< _Node, GET↔_KEY, COMPARE >::remove_tree\(\)](#).

Here is the caller graph for this function:



14.22.2.14 `rend()` [1/2]

```
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.

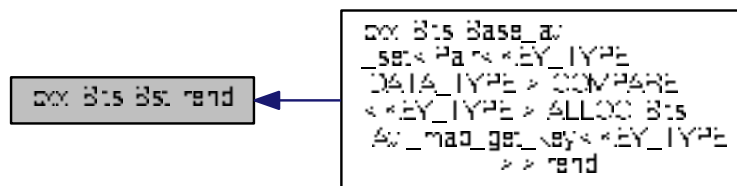
Returns

The end marker for the constant backward iterator.

Definition at line 206 of file [bst.h](#).

Referenced by [cxx::Bits::Base_avl_set< Pair< KEY_TYPE, DATA_TYPE >, COMPARE< KEY_TYPE >, ALLOC, Bits::Avl_map_get_key< KEY_TYPE > >::rend\(\)](#).

Here is the caller graph for this function:

**14.22.2.15** `rend()` [2/2]

```
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 217 of file [bst.h](#).

The documentation for this class was generated from the following file:

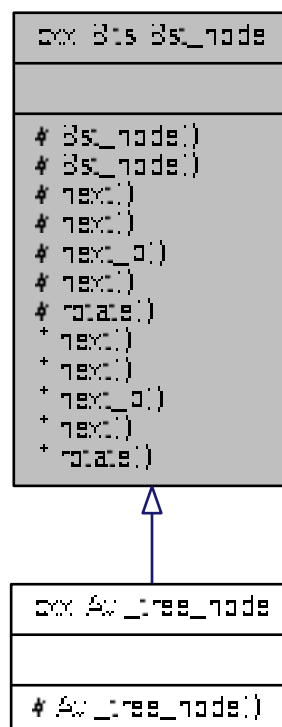
- [l4/cxx/bits/bst.h](#)

14.23 cxx::Bits::Bst_node Class Reference

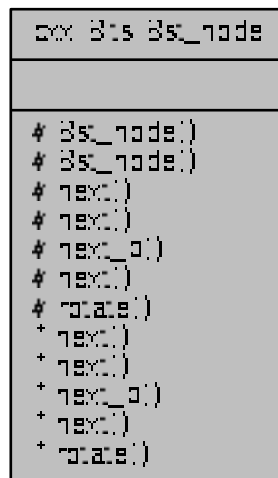
Basic type of a node in a binary search tree (BST).

```
#include <bst_base.h>
```

Inheritance diagram for cxx::Bits::Bst_node:



Collaboration diagram for `cxx::Bits::Bst_node`:



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.

14.23.1 Detailed Description

Basic type of a node in a binary search tree (BST).

Definition at line 77 of file [bst_base.h](#).

The documentation for this class was generated from the following file:

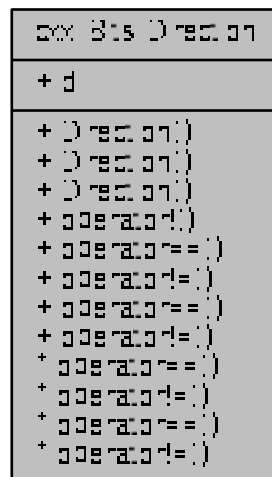
- [I4/cxx/bits/bst_base.h](#)

14.24 cxx::Bits::Direction Struct Reference

The direction to go in a binary search tree.

```
#include <bst_base.h>
```

Collaboration diagram for cxx::Bits::Direction:



Public Types

- enum [Direction_e](#) { [L](#) = 0, [R](#) = 1, [N](#) = 2 }

The literal direction values.

Public Member Functions

- [Direction](#) ()
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
- bool [operator!=](#) ([Direction_e](#) o) const
- bool [operator==](#) ([Direction](#) o) const
- bool [operator!=](#) ([Direction](#) o) const

14.24.1 Detailed Description

The direction to go in a binary search tree.

Definition at line 39 of file [bst_base.h](#).

14.24.2 Member Enumeration Documentation

14.24.2.1 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 42 of file [bst_base.h](#).

14.24.3 Member Function Documentation

14.24.3.1 operator!()

```
Direction cxx::Bits::Direction::operator! ( ) const [inline]
```

Negate the direction.

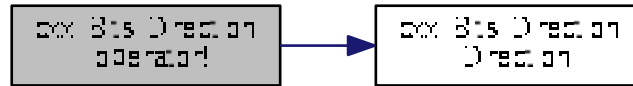
Note

This is only defined for a current value of [L](#) or [R](#)

Definition at line 63 of file [bst_base.h](#).

References [Direction\(\)](#).

Here is the call graph for this function:



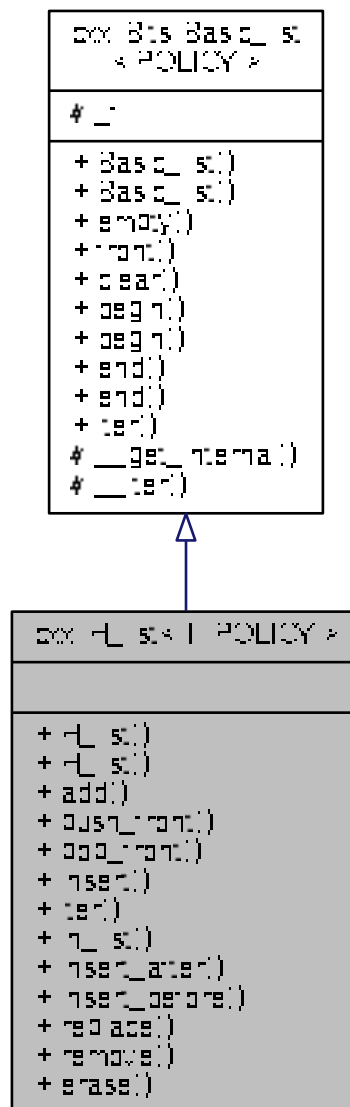
The documentation for this struct was generated from the following file:

- [l4/cxx/bits/bst_base.h](#)

14.25 `cxx::H_list< T, POLICY >` Class Template Reference

General double-linked list of unspecified `cxx::H_list_item` elements.

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.

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.

14.25.1 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 79 of file [hlist](#).

14.25.2 Member Function Documentation

14.25.2.1 erase()

```
template<typename T, typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
static 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 244 of file [hlist](#).

14.25.2.2 `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 end() is given, the element will be inserted at the beginning of the queue.

Returns

Iterator pointing to the newly inserted element.

Definition at line 143 of file [hlist](#).

14.25.2.3 `insert_after()`

```
template<typename T, typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
static 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 end() .

Returns

Iterator pointing to the newly inserted element.

Precondition

The list must not be empty.

Definition at line 170 of file [hlist](#).

14.25.2.4 insert_before()

```
template<typename T, typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
static 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 190 of file [hlist](#).

14.25.2.5 iter()

```
template<typename T, typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
static 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 103 of file [hlist](#).

14.25.2.6 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 126 of file [hlist](#).

14.25.2.7 remove()

```
template<typename T, typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
static 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 228 of file [hlist](#).

14.25.2.8 replace()

```
template<typename T, typename POLICY = Bits::Basic_list_policy< T, H_list_item>>
static 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.

After the operation the *p* element is no longer in the list and may be reused.

Definition at line 212 of file [hlist](#).

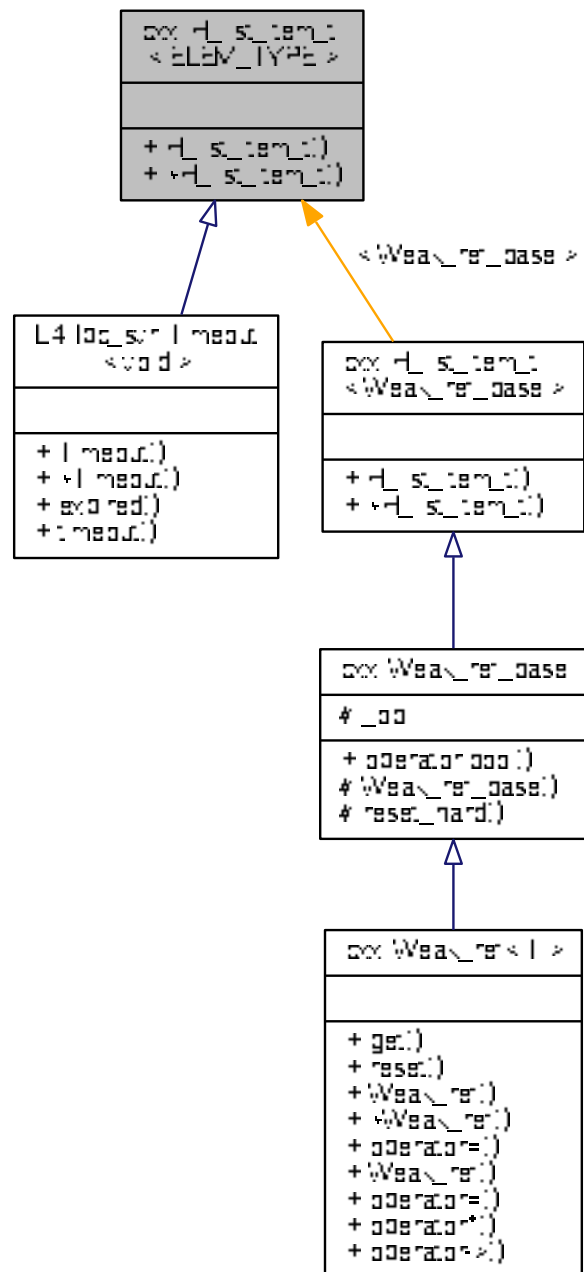
The documentation for this class was generated from the following file:

- I4/cxx/hlist

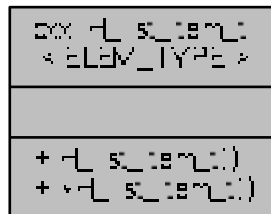
14.26 cxx::H_list_item_t< ELEM_TYPE > Class Template Reference

Basic element type for a double-linked [H_list](#).

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.

14.26.1 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

<i>ELEM_TYPE</i>	Base class of the list element.
------------------	---------------------------------

Definition at line 33 of file [hlist](#).

14.26.2 Constructor & Destructor Documentation

14.26.2.1 H_list_item_t()

```
template<typename ELEM_TYPE>
cxx::H_list_item_t< ELEM_TYPE >::H_list_item_t ( ) [inline]
```

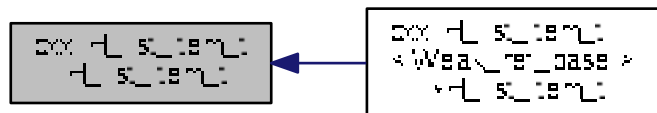
Constructor.

Creates an element that is not in any list.

Definition at line 41 of file [hlist](#).

Referenced by [cxx::H_list_item_t< Weak_ref_base >::~~H_list_item_t\(\)](#).

Here is the caller graph for this function:



14.26.2.2 ~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.

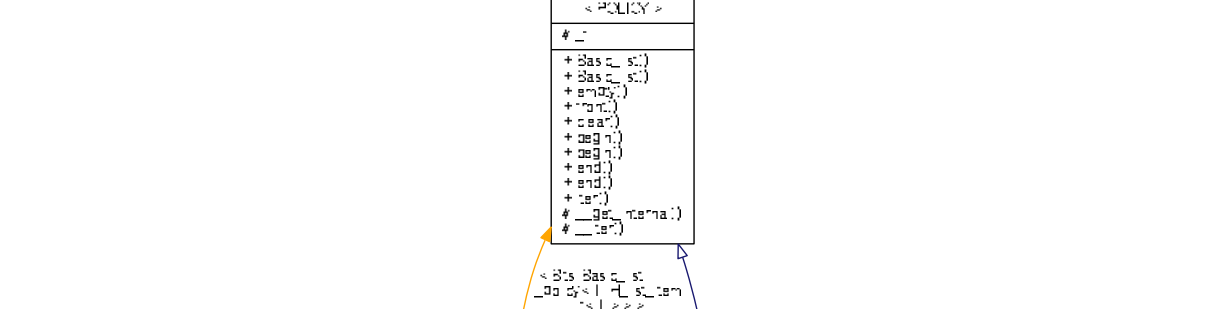
Definition at line 48 of file [hlist](#).

The documentation for this class was generated from the following file:

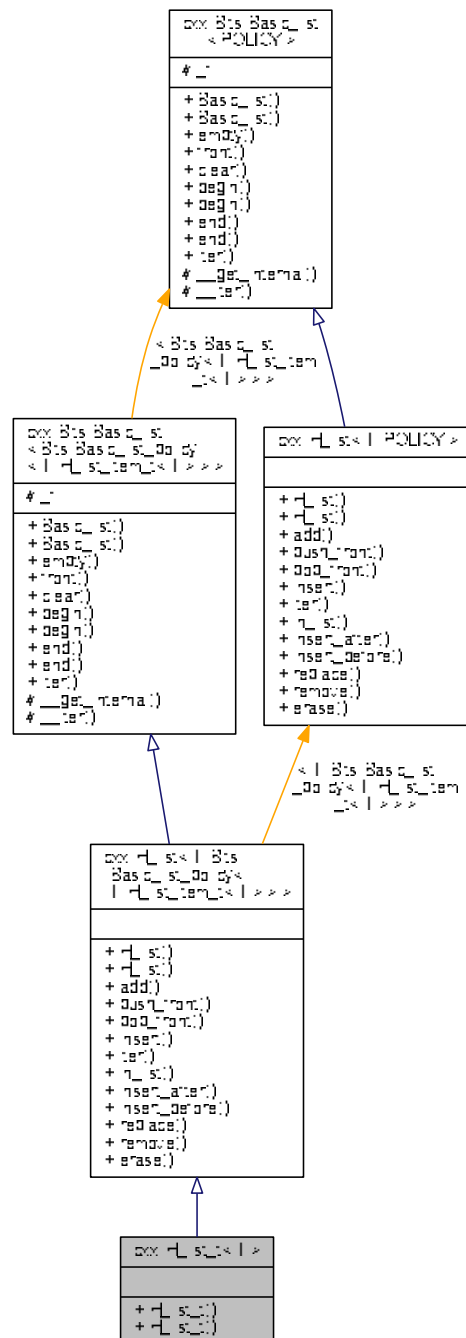
- `I4/cxx/hlist`

14.27 cxx::H_list_t< T > Struct Template Reference

Double-linked list of typed [H_list_item_t](#) elements.



Collaboration diagram for `cxx::H_list_t< T >`:



Additional Inherited Members

14.27.1 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 256 of file [hlist](#).

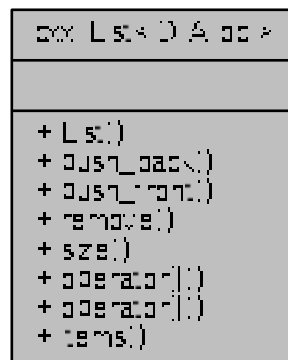
The documentation for this struct was generated from the following file:

- `l4/cxx/hlist`

14.28 `cxx::List< D, Alloc >` Class Template Reference

Doubly linked list, with internal allocation.

Collaboration diagram for `cxx::List< D, Alloc >`:



Data Structures

- class [Iter](#)
Iterator.

Public Member Functions

- void [push_back](#) (D const &d) throw ()
Add element at the end of the list.
- void [push_front](#) (D const &d) throw ()
Add element at the beginning of the list.
- void [remove](#) (Iter const &i) throw ()
Remove element pointed to by the iterator.
- unsigned long [size](#) () const throw ()
Get the length of the list.
- D const & [operator\[\]](#) (unsigned long idx) const throw ()
Random access.
- D & [operator\[\]](#) (unsigned long idx) throw ()
Random access.
- [Iter](#) [items](#) () throw ()
Get iterator for the list elements.

14.28.1 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 334 of file [list](#).

14.28.2 Member Function Documentation

14.28.2.1 items()

```
template<typename D , template< typename A > class Alloc = New_allocator>
Iter cxx::List< D, Alloc >::items ( ) throw )    [inline]
```

Get iterator for the list elements.

Definition at line 412 of file [list](#).

14.28.2.2 operator[]() [1/2]

```
template<typename D , template< typename A > class Alloc = New_allocator>
D const& cxx::List< D, Alloc >::operator[] (
    unsigned long idx ) const throw )    [inline]
```

Random access.

Complexity is O(n).

Definition at line 404 of file [list](#).

14.28.2.3 operator[]() [2/2]

```
template<typename D , template< typename A > class Alloc = New_allocator>
D& cxx::List< D, Alloc >::operator[] (
    unsigned long idx ) throw )    [inline]
```

Random access.

Complexity is O(n).

Definition at line 408 of file [list](#).

14.28.2.4 `push_back()`

```
template<typename D , template< typename A > class Alloc = New_allocator>
void cxx::List< D, Alloc >::push_back (
    D const & d ) throw )    [inline]
```

Add element at the end of the list.

Definition at line 379 of file [list](#).

14.28.2.5 `push_front()`

```
template<typename D , template< typename A > class Alloc = New_allocator>
void cxx::List< D, Alloc >::push_front (
    D const & d ) throw )    [inline]
```

Add element at the beginning of the list.

Definition at line 388 of file [list](#).

14.28.2.6 `remove()`

```
template<typename D , template< typename A > class Alloc = New_allocator>
void cxx::List< D, Alloc >::remove (
    Iter const & i ) throw )    [inline]
```

Remove element pointed to by the iterator.

Definition at line 397 of file [list](#).

14.28.2.7 `size()`

```
template<typename D , template< typename A > class Alloc = New_allocator>
unsigned long cxx::List< D, Alloc >::size ( ) const throw )    [inline]
```

Get the length of the list.

Definition at line 401 of file [list](#).

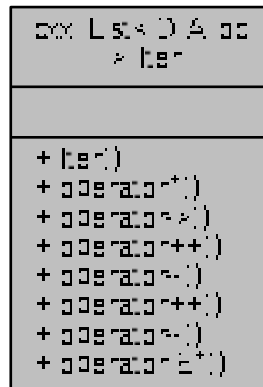
The documentation for this class was generated from the following file:

- `I4/cxx/list`

14.29 cxx::List< D, Alloc >::Iter Class Reference

Iterator.

Collaboration diagram for cxx::List< D, Alloc >::Iter:



Public Member Functions

- [operator E*](#) () const throw ()
operator for testing validity (syntactically equal to pointers)

14.29.1 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 [354](#) of file [list](#).

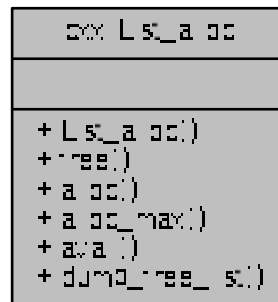
The documentation for this class was generated from the following file:

- [l4/cxx/list](#)

14.30 `cxx::List_alloc` Class Reference

Standard list-based allocator.

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 align)
Alloc a memory block.
- void * [alloc_max](#) (unsigned long min, unsigned long *max, unsigned align, unsigned granularity)
Allocate a memory block of $min \leq size \leq max$.
- unsigned long [avail](#) ()
Get the amount of available memory.

14.30.1 Detailed Description

Standard list-based allocator.

Definition at line 31 of file [list_alloc](#).

14.30.2 Constructor & Destructor Documentation

14.30.2.1 List_alloc()

```
cxx::List_alloc::List_alloc ( ) [inline]
```

Initializes an empty list allocator.

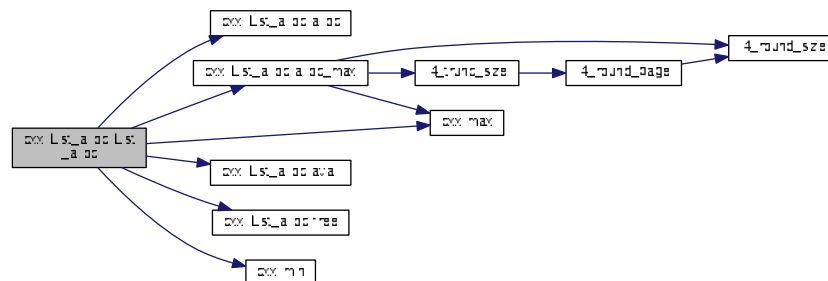
Note

To initialize the allocator with available memory use the [free\(\)](#) function.

Definition at line 56 of file [list_alloc](#).

References [alloc\(\)](#), [alloc_max\(\)](#), [avail\(\)](#), [L4::cerr](#), [free\(\)](#), [cxx::max\(\)](#), and [cxx::min\(\)](#).

Here is the call graph for this function:



14.30.3 Member Function Documentation

14.30.3.1 alloc()

```
void * cxx::List_alloc::alloc (
    unsigned long size,
    unsigned align ) [inline]
```

Alloc a memory block.

Parameters

<i>size</i>	Size of the memory block
<i>align</i>	Alignment constraint

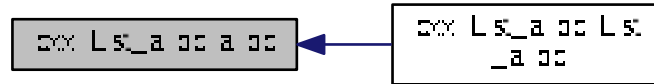
Returns

Pointer to memory block

Definition at line 354 of file [list_alloc](#).

Referenced by [List_alloc\(\)](#).

Here is the caller graph for this function:



14.30.3.2 alloc_max()

```
void * cxx::List_alloc::alloc_max (
    unsigned long min,
    unsigned long * max,
    unsigned align,
    unsigned granularity ) [inline]
```

Allocate a memory block of `min <= size <=max`.

Parameters

	<i>min</i>	Minimal size to allocate.
<i>in, out</i>	<i>max</i>	Maximum size to allocate. The actual allocated size is returned here.
	<i>align</i>	Alignment constraint.
	<i>granularity</i>	Granularity to use for the allocation.

Returns

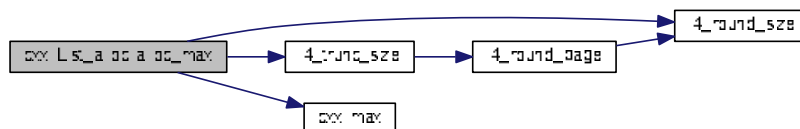
Pointer to memory block

Definition at line 257 of file [list_alloc](#).

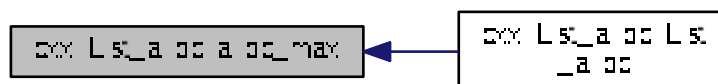
References [l4_round_size\(\)](#), [l4_trunc_size\(\)](#), and [cxx::max\(\)](#).

Referenced by [List_alloc\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



14.30.3.3 avail()

```
unsigned long cxx::List_alloc::avail ( ) [inline]
```

Get the amount of available memory.

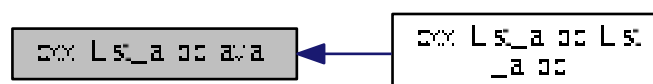
Returns

Available memory in bytes

Definition at line 425 of file [list_alloc](#).

Referenced by [List_alloc\(\)](#).

Here is the caller graph for this function:



14.30.3.4 free()

```
void cxx::List_alloc::free (
    void * block,
    unsigned long size,
    bool initial_free = false ) [inline]
```

Return a free memory block to the allocator.

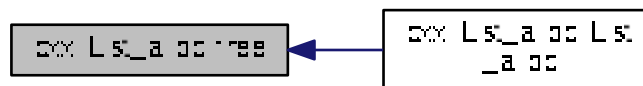
Parameters

<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.

Definition at line 218 of file [list_alloc](#).

Referenced by [List_alloc\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

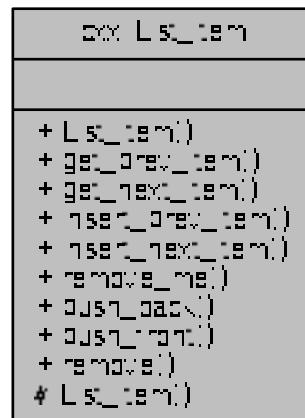
- `I4/cxx/list_alloc`

14.31 cxx::List_item Class Reference

Basic list item.

Inherited by `cxx::T_list_item< T >`.

Collaboration diagram for `cxx::List_item`:



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 throw ()
Get previous item.
- [List_item](#) * [get_next_item](#) () const throw ()
Get next item.
- void [insert_prev_item](#) ([List_item](#) *p) throw ()
Insert item p before this item.
- void [insert_next_item](#) ([List_item](#) *p) throw ()
Insert item p after this item.
- void [remove_me](#) () throw ()
Remove this item from the list.

Static Public Member Functions

- template<typename C , typename N >
static C * [push_back](#) (C *head, N *p) throw ()
Append item to a list.
- template<typename C , typename N >
static C * [push_front](#) (C *head, N *p) throw ()
Prepend item to a list.
- template<typename C , typename N >
static C * [remove](#) (C *head, N *p) throw ()
Remove item from a list.

14.31.1 Detailed Description

Basic list item.

Basic item that can be member of a doubly linked, cyclic list.

Definition at line 37 of file [list](#).

14.31.2 Member Function Documentation

14.31.2.1 `get_next_item()`

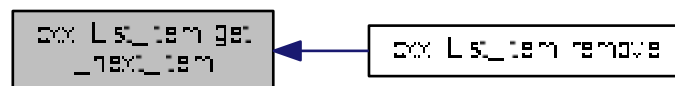
```
List_item* cxx::List_item::get_next_item ( ) const throw ( ) [inline]
```

Get next item.

Definition at line 176 of file [list](#).

Referenced by [remove\(\)](#).

Here is the caller graph for this function:



14.31.2.2 `get_prev_item()`

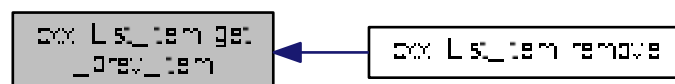
```
List_item* cxx::List_item::get_prev_item ( ) const throw ( ) [inline]
```

Get previous item.

Definition at line 173 of file [list](#).

Referenced by [remove\(\)](#).

Here is the caller graph for this function:



14.31.2.3 insert_next_item()

```
void cxx::List_item::insert_next_item (
    List_item * p ) throw ()    [inline]
```

Insert item p after this item.

Definition at line 189 of file [list](#).

14.31.2.4 insert_prev_item()

```
void cxx::List_item::insert_prev_item (
    List_item * p ) throw ()    [inline]
```

Insert item p before this item.

Definition at line 179 of file [list](#).

14.31.2.5 push_back()

```
template<typename C , typename N >
C * cxx::List_item::push_back (
    C * head,
    N * p ) throw ()    [inline], [static]
```

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.

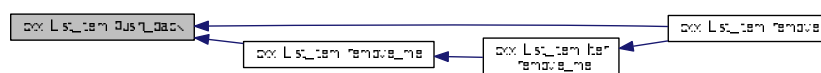
Returns

the pointer to the new head.

Definition at line 248 of file [list](#).

Referenced by [remove\(\)](#), and [remove_me\(\)](#).

Here is the caller graph for this function:



14.31.2.6 push_front()

```
template<typename C , typename N >
C * cxx::List_item::push_front (
    C * head,
    N * p ) throw ()    [inline], [static]
```

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.

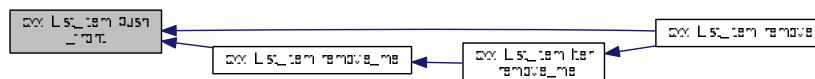
Returns

the pointer to the new head.

Definition at line 259 of file [list](#).

Referenced by [remove\(\)](#), and [remove_me\(\)](#).

Here is the caller graph for this function:



14.31.2.7 remove()

```
template<typename C , typename N >
C * cxx::List_item::remove (
    C * head,
    N * p ) throw ()    [inline], [static]
```

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.

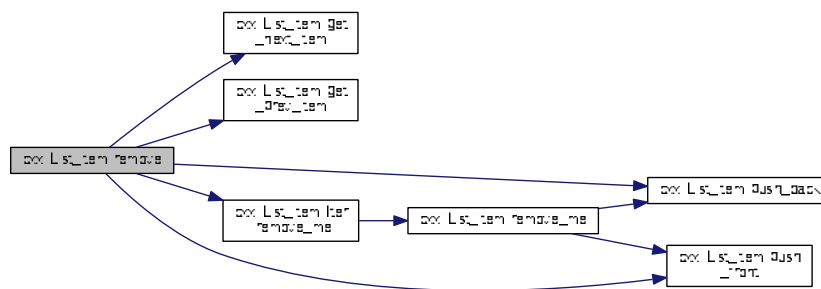
Returns

the pointer to the new head.

Definition at line 269 of file [list](#).

References [get_next_item\(\)](#), [get_prev_item\(\)](#), [push_back\(\)](#), [push_front\(\)](#), and [cxx::List_item::Iter::remove_me\(\)](#).

Here is the call graph for this function:

14.31.2.8 `remove_me()`

```
void cxx::List_item::remove_me ( ) throw ( ) [inline]
```

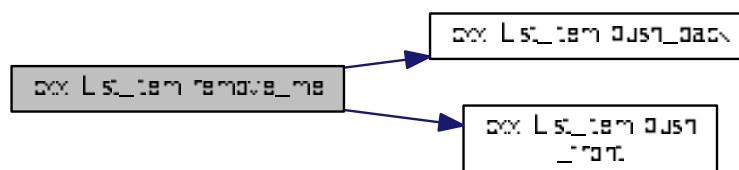
Remove this item from the list.

Definition at line 198 of file [list](#).

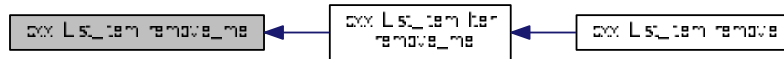
References [push_back\(\)](#), and [push_front\(\)](#).

Referenced by [cxx::List_item::Iter::remove_me\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



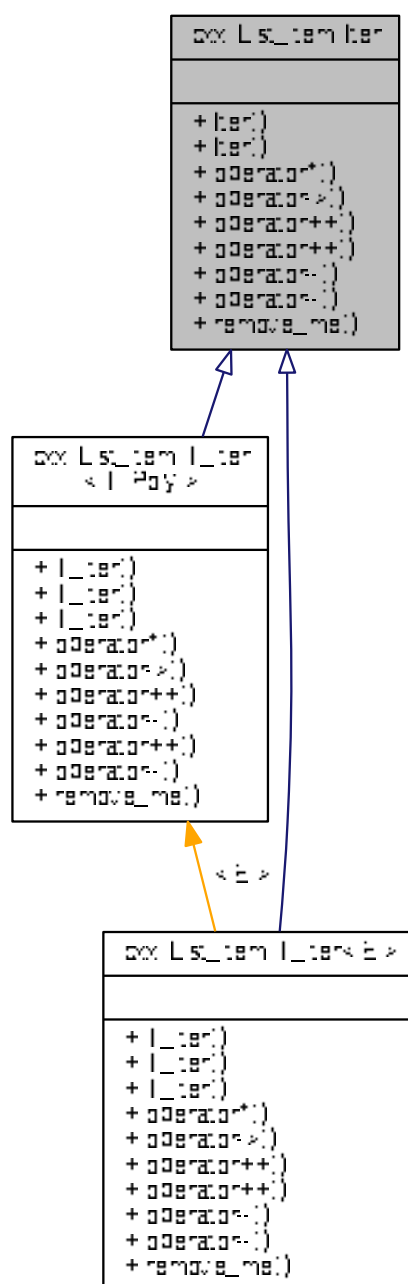
The documentation for this class was generated from the following file:

- `I4/cxx/list`

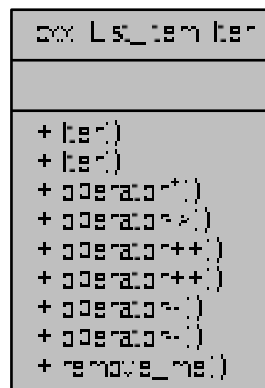
14.32 cxx::List_item::Iter Class Reference

Iterator for a list of ListItem-s.

Inheritance diagram for `cxx::List_item::Iter`:



Collaboration diagram for cxx::List_item::Iter:



Public Member Functions

- [List_item * remove_me \(\)](#) throw ()
Remove item pointed to by iterator, and return pointer to element.

14.32.1 Detailed Description

Iterator for a list of ListItem-s.

The Iterator iterates till it finds the first element again.

Definition at line 45 of file [list](#).

14.32.2 Member Function Documentation

14.32.2.1 remove_me()

```
List_item* cxx::List_item::Iter::remove_me ( ) throw () [inline]
```

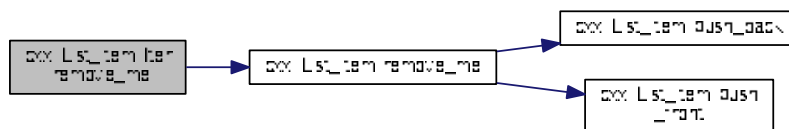
Remove item pointed to by iterator, and return pointer to element.

Definition at line 86 of file [list](#).

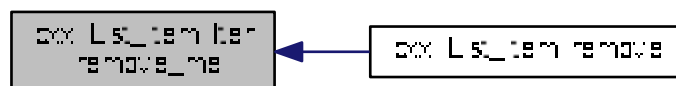
References [cxx::List_item::remove_me\(\)](#).

Referenced by [cxx::List_item::remove\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



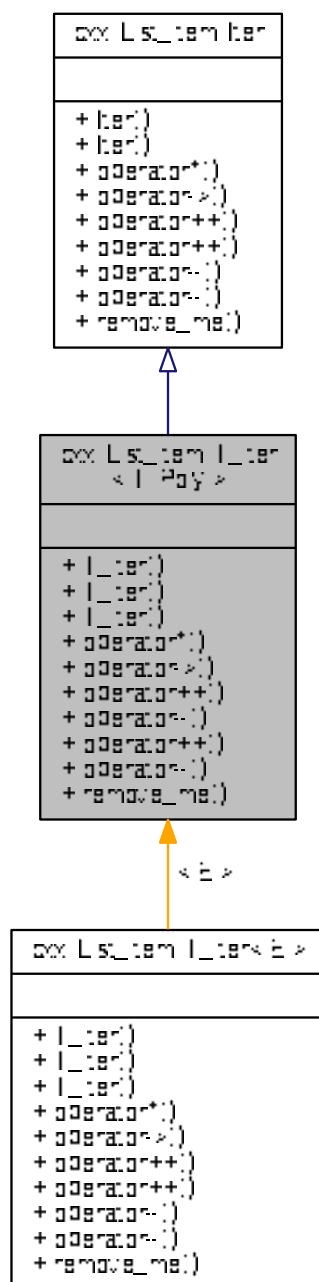
The documentation for this class was generated from the following file:

- `I4/cxx/list`

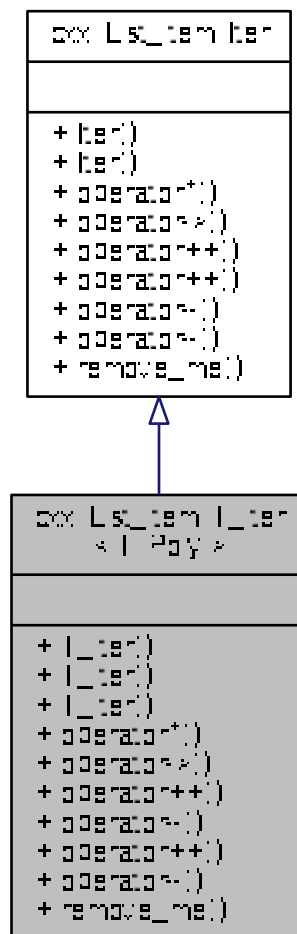
14.33 `cxx::List_item::T_iter< T, Poly >` Class Template Reference

Iterator for derived classes from `ListItem`.

Inheritance diagram for cxx::List_item::T_iter< T, Poly >:



Collaboration diagram for `cxx::List_item::T_iter< T, Poly >`:



Additional Inherited Members

14.33.1 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> lter; ... };`

Definition at line 119 of file [list](#).

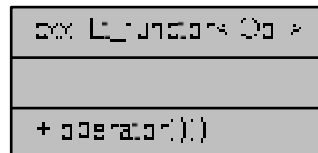
The documentation for this class was generated from the following file:

- `I4/cxx/list`

14.34 `cxx::Lt_functor< Obj >` Struct Template Reference

Generic comparator class that defaults to the less-than operator.

Collaboration diagram for `cxx::Lt_functor< Obj >`:



14.34.1 Detailed Description

```
template<typename Obj>
struct cxx::Lt_functor< Obj >
```

Generic comparator class that defaults to the less-than operator.

Definition at line 29 of file [std_ops](#).

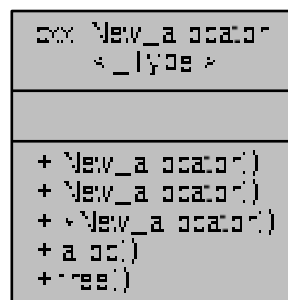
The documentation for this struct was generated from the following file:

- `I4/cxx/std_ops`

14.35 `cxx::New_allocator< _Type >` Class Template Reference

Standard allocator based on `operator new ()`.

Collaboration diagram for `cxx::New_allocator< _Type >`:



14.35.1 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 60 of file [std_alloc](#).

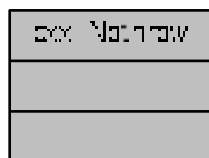
The documentation for this class was generated from the following file:

- `I4/cxx/std_alloc`

14.36 cxx::Nothrow Class Reference

Helper type to distinguish the `oeprator new` version that does not throw exceptions.

Collaboration diagram for `cxx::Nothrow`:



14.36.1 Detailed Description

Helper type to distinguish the `oeprator new` version that does not throw exceptions.

Definition at line 30 of file [std_alloc](#).

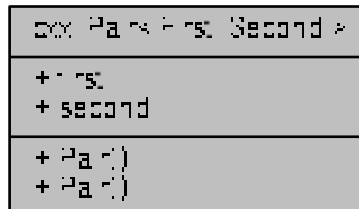
The documentation for this class was generated from the following file:

- `I4/cxx/std_alloc`

14.37 cxx::Pair< First, Second > Struct Template Reference

Pair of two values.

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.
- [Pair](#) ()
Default construction.

Data Fields

- First [first](#)
First value.
- Second [second](#)
Second value.

14.37.1 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 36 of file [pair](#).

14.37.2 Constructor & Destructor Documentation

14.37.2.1 Pair()

```
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 54 of file [pair](#).

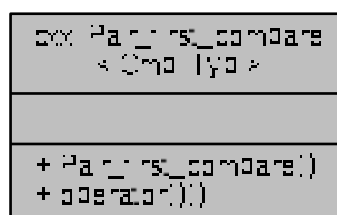
The documentation for this struct was generated from the following file:

- [l4/cxx/pair](#)

14.38 cxx::Pair_first_compare< Cmp, Typ > Class Template Reference

Comparison functor for [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 comaprison based on the first value.

14.38.1 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 75 of file [pair](#).

14.38.2 Constructor & Destructor Documentation

14.38.2.1 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 85 of file [pair](#).

14.38.3 Member Function Documentation

14.38.3.1 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 92 of file [pair](#).

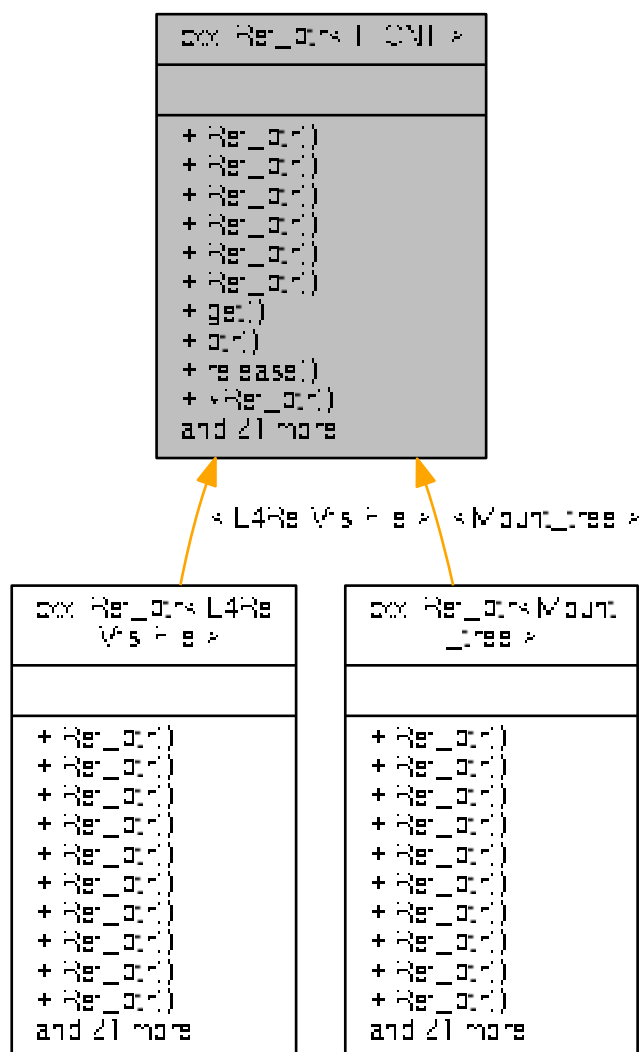
The documentation for this class was generated from the following file:

- [I4/cxx/pair](#)

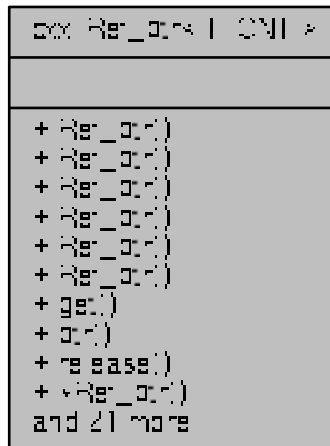
14.39 cxx::Ref_ptr< T, CNT > Class Template Reference

A reference-counting pointer with automatic cleanup.

Inheritance diagram for cxx::Ref_ptr< T, CNT >:



Collaboration diagram for `cxx::Ref_ptr< T, CNT >`:



Public Member Functions

- [Ref_ptr](#) () throw ()
Default constructor creates a pointer with no managed object.
- [Ref_ptr](#) (Wp const &o) throw ()
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) throw ()
Create a shared pointer from a raw pointer.
- [Ref_ptr](#) (T *o, bool d) throw ()
Create a shared pointer from a raw pointer without creating a new reference.
- T * [get](#) () const throw ()
Return a raw pointer to the object this shared pointer points to.
- T * [ptr](#) () const throw ()
Return a raw pointer to the object this shared pointer points to.
- T * [release](#) () throw ()
Release the shared pointer without removing the reference.

14.39.1 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 80 of file [ref_ptr](#).

14.39.2 Constructor & Destructor Documentation

14.39.2.1 `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 ) throw )    [inline]
```

Create a shared pointer from a weak pointer.

Increases references.

Definition at line 101 of file [ref_ptr](#).

14.39.2.2 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 ) throw )    [inline], [explicit]
```

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 115 of file `ref_ptr`.

14.39.2.3 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 ) throw )    [inline]
```

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 138 of file `ref_ptr`.

14.39.3 Member Function Documentation**14.39.3.1 get()**

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
T* cxx::Ref_ptr< T, CNT >::get ( ) const throw )    [inline]
```

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 145 of file `ref_ptr`.

14.39.3.2 `ptr()`

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
T* cxx::Ref_ptr< T, CNT >::ptr ( ) const throw ( ) [inline]
```

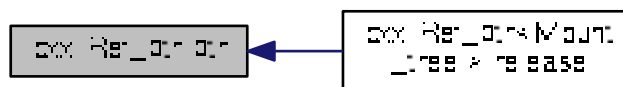
Return a raw pointer to the object this shared pointer points to.

This does not release the pointer or decrease the reference count.

Definition at line 151 of file [ref_ptr](#).

Referenced by [cxx::Ref_ptr< Mount_tree >::release\(\)](#).

Here is the caller graph for this function:

14.39.3.3 `release()`

```
template<typename T = void, template< typename X > class CNT = Default_ref_counter>
T* cxx::Ref_ptr< T, CNT >::release ( ) throw ( ) [inline]
```

Release the shared pointer without removing the reference.

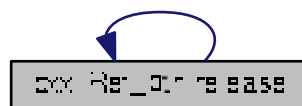
Returns

A raw pointer to the managed object.

Definition at line 162 of file [ref_ptr](#).

Referenced by [cxx::Ref_ptr< Mount_tree >::release\(\)](#).

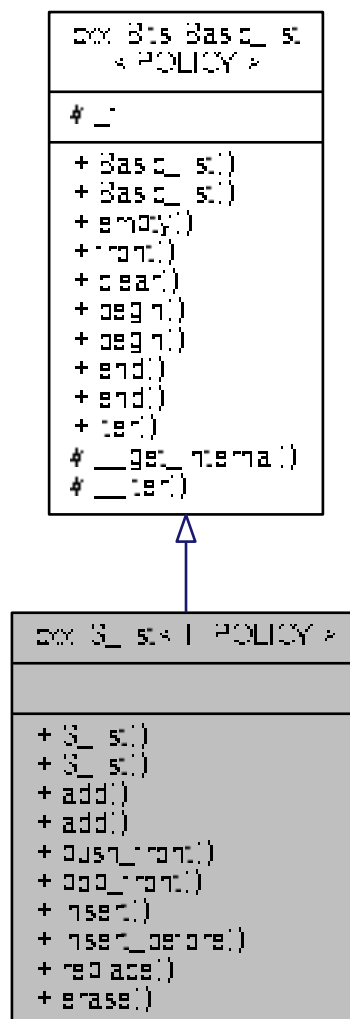
Here is the caller graph for this function:



The documentation for this class was generated from the following file:

- `I4/cxx/ref_ptr`

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.

Additional Inherited Members

14.40.1 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 50 of file [slist](#).

14.40.2 Member Function Documentation

14.40.2.1 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](#).

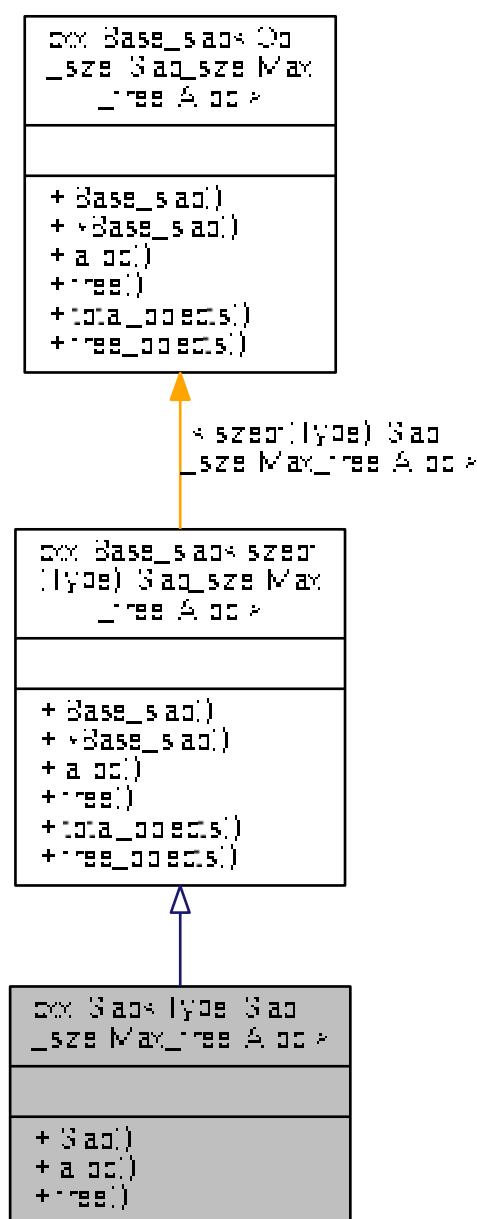
The documentation for this class was generated from the following file:

- I4/cxx/slist

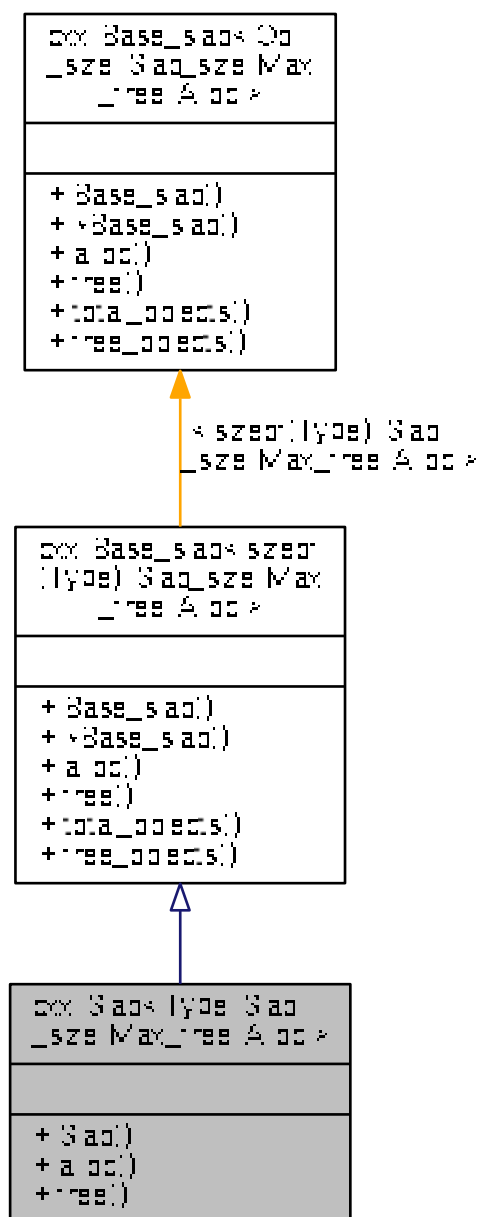
14.41 cxx::Slab< Type, Slab_size, Max_free, Alloc > Class Template Reference

[Slab](#) allocator for object of type *Type*.

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 ()` throw ()
Allocate an object of type Type.
- `void free (Type *o)` throw ()
Free the object addressed by o.

Additional Inherited Members

14.41.1 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*.

Parameters

<i>Type</i>	the type of the objects to manage.
<i>Slab_size</i>	size of a slab cache.
<i>Max_free</i>	the maximum number of free slab caches.
<i>Alloc</i>	the allocator for the slab caches.

Definition at line 297 of file [slab_alloc](#).

14.41.2 Member Function Documentation

14.41.2.1 `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 ( ) throw )    [inline]
```

Allocate an object of type *Type*.

Returns

A pointer to the object just allocated, or 0 on failure.

Definition at line 314 of file [slab_alloc](#).

14.41.2.2 `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 ) throw )    [inline]
```

Free the object addressed by *o*.

Parameters

<i>o</i>	The pointer to the object to free.
----------	------------------------------------

Precondition

The object must have been allocated with this allocator.

Definition at line [325](#) of file [slab_alloc](#).

References [L4_PAGESIZE](#).

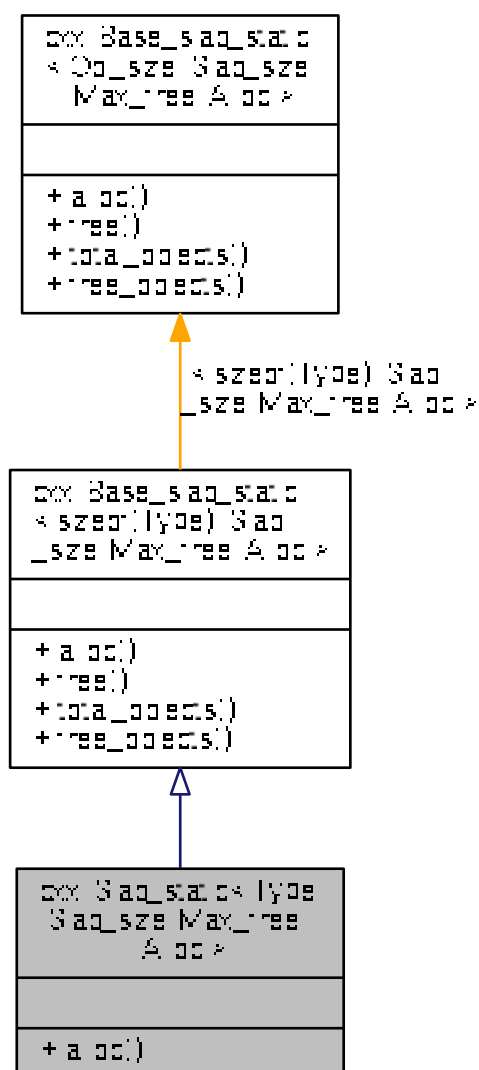
The documentation for this class was generated from the following file:

- `I4/cxx/slab_alloc`

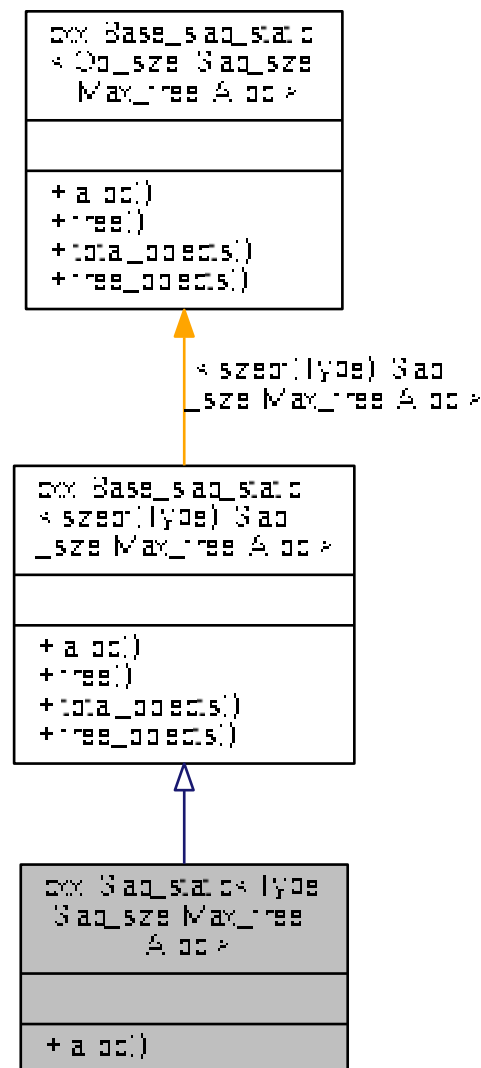
14.42 `cxx::Slab_static< Type, Slab_size, Max_free, Alloc >` Class Template Reference

Merged slab allocator (allocators for objects of the same size are merged together).

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 () throw ()`
Allocate an object of type *Type*.

14.42.1 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).

Parameters

<i>Type</i>	The type of the objects to manage.
<i>Slab_size</i>	The size of a slab cache.
<i>Max_free</i>	The maximum number of free slab caches.
<i>Alloc</i>	The allocator for the slab caches.

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 415 of file [slab_alloc](#).

14.42.2 Member Function Documentation

14.42.2.1 `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 ( ) throw ( )    [inline]
```

Allocate an object of type *Type*.

Returns

A pointer to the just allocated object, or 0 of failure.

Definition at line 425 of file [slab_alloc](#).

The documentation for this class was generated from the following file:

- `I4/cxx/slab_alloc`

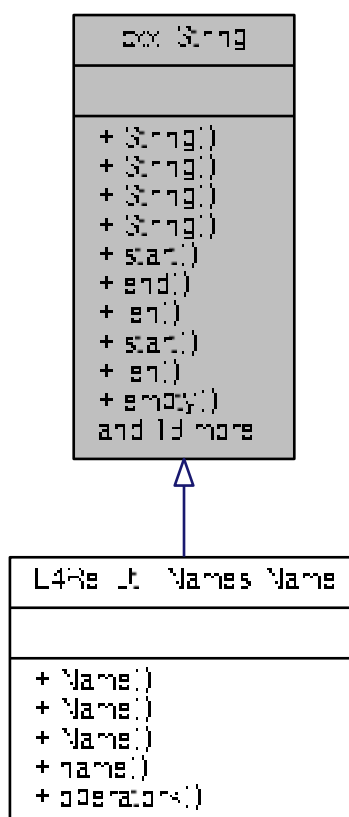
14.43 `cxx::static_vector< T, IDX >` Class Template Reference

Simple encapsulation for a dynamically allocated array.

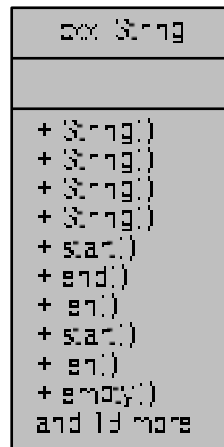
14.44 cxx::String Class Reference

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.

Inheritance diagram for cxx::String:



Collaboration diagram for `cxx::String`:



14.44.1 Detailed Description

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.

Examples:

[tmpfs/lib/src/fs.cc](#).

Definition at line 33 of file [string](#).

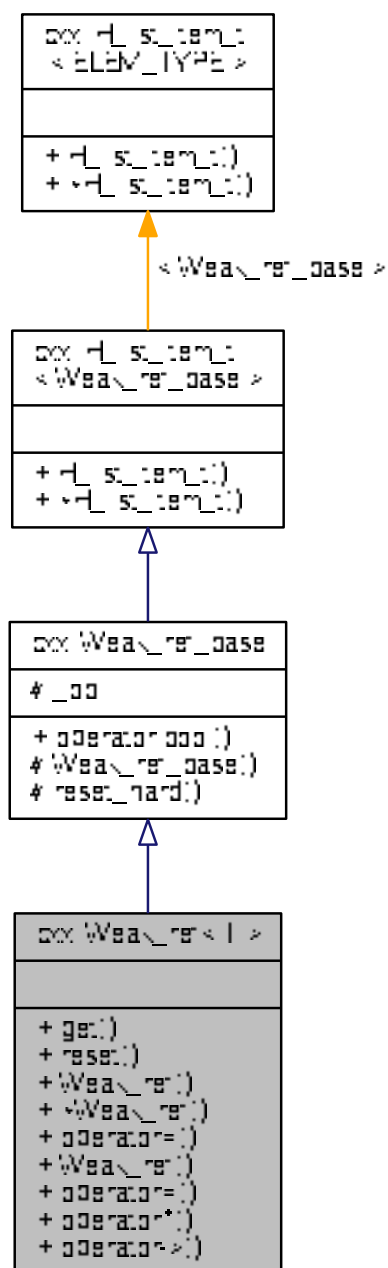
The documentation for this class was generated from the following file:

- [l4/cxx/string](#)

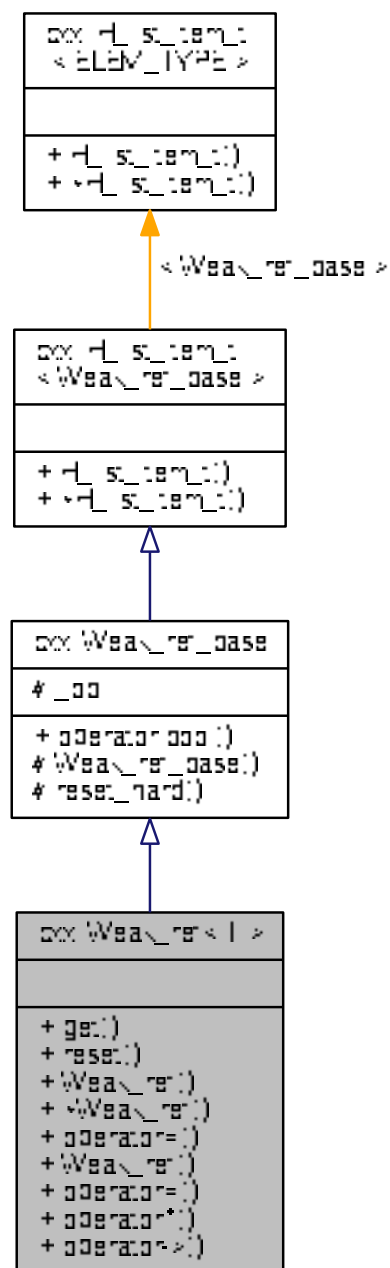
14.45 `cxx::Weak_ref< T >` Class Template Reference

Typed weak reference to an object of type `T`.

Inheritance diagram for cxx::Weak_ref< T >:



Collaboration diagram for `cxx::Weak_ref< T >`:



Additional Inherited Members

14.45.1 Detailed Description

```
template<typename T>
class cxx::Weak_ref< T >
```

Typed weak reference to an object of type `T`.

Template Parameters

<i>T</i>	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 and all the weak references are iterated and reset by the `Weak_ref_base::List` destructor or `Weak_ref_base::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.

Definition at line 67 of file [weak_ref](#).

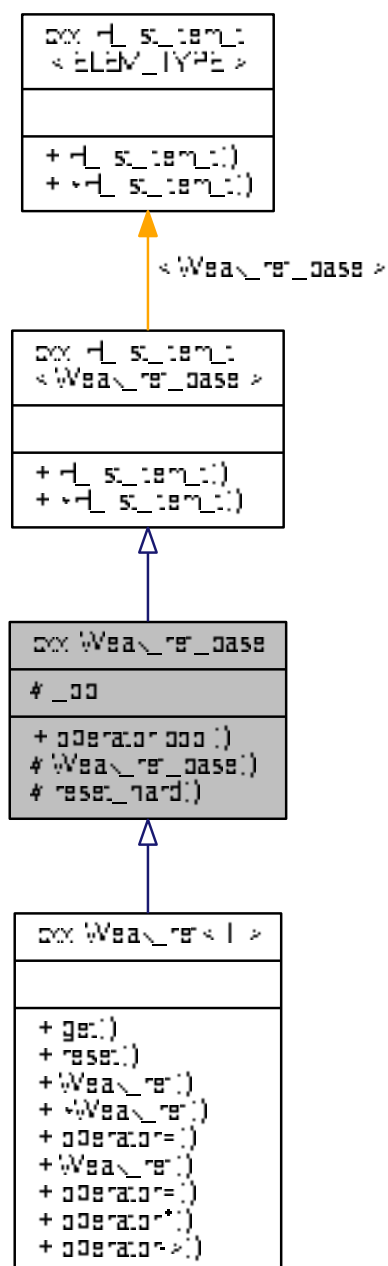
The documentation for this class was generated from the following file:

- `I4/cxx/weak_ref`

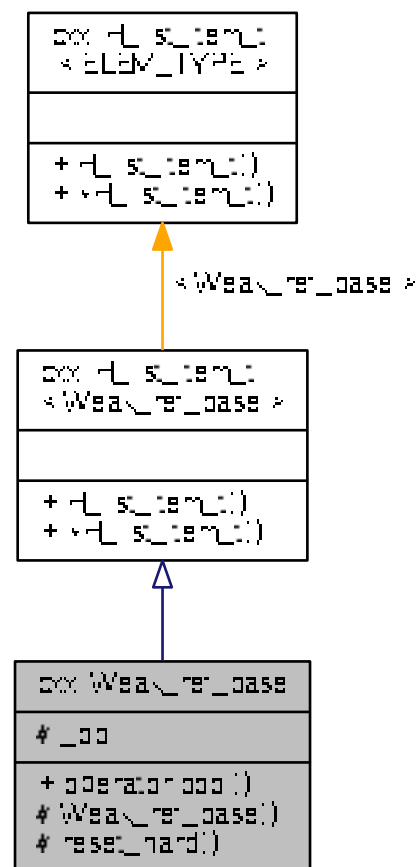
14.46 cxx::Weak_ref_base Class Reference

Generic (base) weak reference to some object.

Inheritance diagram for `cxx::Weak_ref_base`:



Collaboration diagram for cxx::Weak_ref_base:



Additional Inherited Members

14.46.1 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 25 of file [weak_ref](#).

The documentation for this class was generated from the following file:

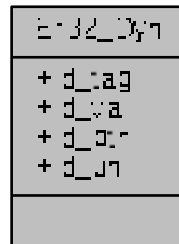
- `I4/cxx/weak_ref`

14.47 Elf32_Dyn Struct Reference

ELF32 dynamic entry.

```
#include <elf.h>
```

Collaboration diagram for Elf32_Dyn:



Data Fields

- [Elf32_Word d_tag](#)
see DT_ values
- [Elf32_Word d_val](#)
integer values with various interpret.
- [Elf32_Addr d_ptr](#)
program virtual addresses

14.47.1 Detailed Description

ELF32 dynamic entry.

Definition at line 500 of file [elf.h](#).

14.47.2 Field Documentation

14.47.2.1 d_val

[Elf32_Word](#) Elf32_Dyn::d_val

integer values with various interpret.

Definition at line 503 of file [elf.h](#).

The documentation for this struct was generated from the following file:

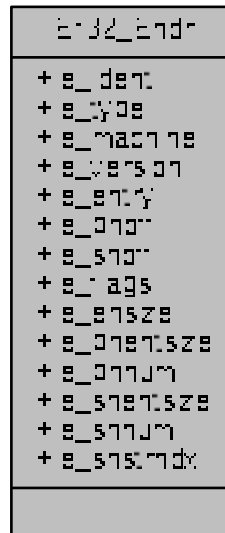
- [l4/util/elf.h](#)

14.48 Elf32_Ehdr Struct Reference

ELF32 header.

```
#include <elf.h>
```

Collaboration diagram for Elf32_Ehdr:



Data Fields

- [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 eip
- [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](#)

of entries in prog.

- [Elf32_Half e_shentsize](#)
size of section header entry
- [Elf32_Half e_shnum](#)

of entries in sect.

- [Elf32_Half e_shstrndx](#)
sect.head.tab.idx of strtab

14.48.1 Detailed Description

ELF32 header.

Definition at line 122 of file [elf.h](#).

14.48.2 Field Documentation

14.48.2.1 e_phnum

[Elf32_Half](#) Elf32_Ehdr::e_phnum

of entries in prog.

head. tab.

Definition at line 133 of file [elf.h](#).

14.48.2.2 e_shnum

[Elf32_Half](#) Elf32_Ehdr::e_shnum

of entries in sect.

head. tab.

Definition at line 135 of file [elf.h](#).

The documentation for this struct was generated from the following file:

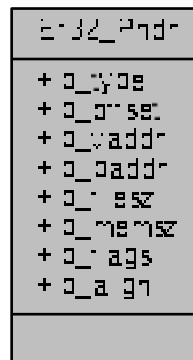
- [l4/util/elf.h](#)

14.49 Elf32_Phdr Struct Reference

ELF32 program header.

```
#include <elf.h>
```

Collaboration diagram for Elf32_Phdr:



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

14.49.1 Detailed Description

ELF32 program header.

Definition at line 419 of file [elf.h](#).

The documentation for this struct was generated from the following file:

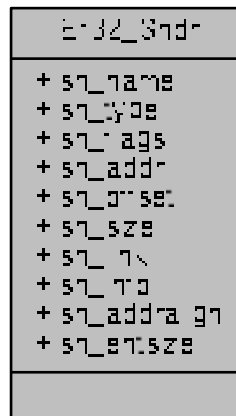
- [l4/util/elf.h](#)

14.50 Elf32_Shdr Struct Reference

ELF32 section header - figure 1-9, page 1-9.

```
#include <elf.h>
```

Collaboration diagram for Elf32_Shdr:



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

14.50.1 Detailed Description

ELF32 section header - figure 1-9, page 1-9.

Definition at line 342 of file [elf.h](#).

The documentation for this struct was generated from the following file:

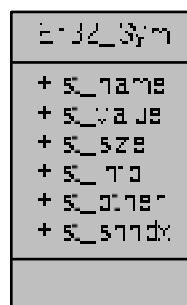
- [l4/util/elf.h](#)

14.51 Elf32_Sym Struct Reference

ELF32 symbol table entry.

```
#include <elf.h>
```

Collaboration diagram for Elf32_Sym:



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

14.51.1 Detailed Description

ELF32 symbol table entry.

Definition at line 801 of file [elf.h](#).

The documentation for this struct was generated from the following file:

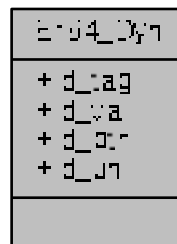
- [l4/util/elf.h](#)

14.52 Elf64_Dyn Struct Reference

ELF64 dynamic entry.

```
#include <elf.h>
```

Collaboration diagram for Elf64_Dyn:



Data Fields

- [Elf64_Sxword d_tag](#)
see DT_ values
- [Elf64_Xword d_val](#)
integer values with various interpret.
- [Elf64_Addr d_ptr](#)
program virtual addresses

14.52.1 Detailed Description

ELF64 dynamic entry.

Definition at line 509 of file [elf.h](#).

14.52.2 Field Documentation

14.52.2.1 d_val

`Elf64_Xword Elf64_Dyn::d_val`

integer values with various interpret.

Definition at line 512 of file [elf.h](#).

The documentation for this struct was generated from the following file:

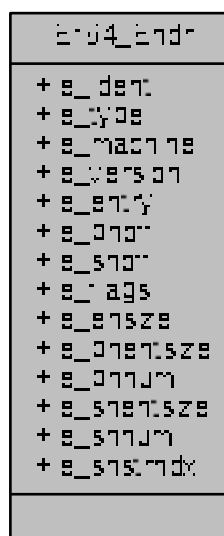
- [l4/util/elf.h](#)

14.53 Elf64_Ehdr Struct Reference

ELF64 header.

```
#include <elf.h>
```

Collaboration diagram for Elf64_Ehdr:



Data Fields

- [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 eip
- [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](#)

of entries in prog.

- [Elf64_Half e_shentsize](#)
size of section header entry
- [Elf64_Half e_shnum](#)

of entries in sect.

- [Elf64_Half e_shstrndx](#)
sect.head.tab.idx of strtb

14.53.1 Detailed Description

ELF64 header.

Definition at line 142 of file [elf.h](#).

14.53.2 Field Documentation

14.53.2.1 e_phnum

[Elf64_Half](#) [Elf64_Ehdr::e_phnum](#)

of entries in prog.

head. tab.

Definition at line 153 of file [elf.h](#).

14.53.2.2 e_shnum

[Elf64_Half](#) Elf64_Ehdr::e_shnum

of entries in sect.

head. tab.

Definition at line 155 of file [elf.h](#).

The documentation for this struct was generated from the following file:

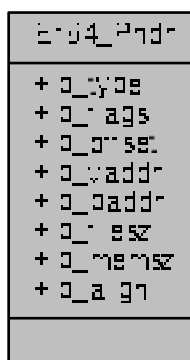
- [l4/util/elf.h](#)

14.54 Elf64_Phdr Struct Reference

ELF64 program header.

```
#include <elf.h>
```

Collaboration diagram for Elf64_Phdr:



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

14.54.1 Detailed Description

ELF64 program header.

Definition at line 431 of file [elf.h](#).

The documentation for this struct was generated from the following file:

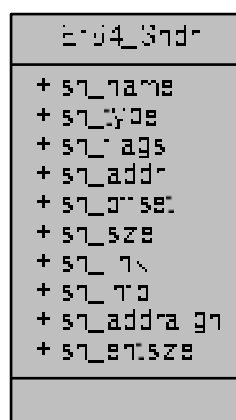
- [l4/util/elf.h](#)

14.55 Elf64_Shdr Struct Reference

ELF64 section header.

```
#include <elf.h>
```

Collaboration diagram for Elf64_Shdr:



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

14.55.1 Detailed Description

ELF64 section header.

Definition at line 356 of file [elf.h](#).

The documentation for this struct was generated from the following file:

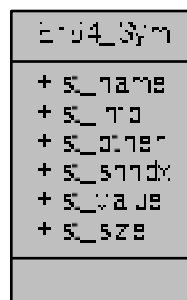
- [l4/util/elf.h](#)

14.56 Elf64_Sym Struct Reference

ELF64 symbol table entry.

```
#include <elf.h>
```

Collaboration diagram for Elf64_Sym:



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

14.56.1 Detailed Description

ELF64 symbol table entry.

Definition at line [811](#) of file [elf.h](#).

The documentation for this struct was generated from the following file:

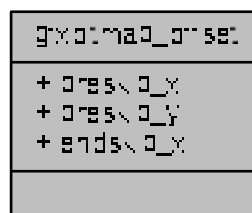
- [l4/util/elf.h](#)

14.57 gfxbitmap_offset Struct Reference

offsets in pmap[] and bmap[]

```
#include <bitmap.h>
```

Collaboration diagram for gfxbitmap_offset:



Data Fields

- `l4_uint32_t preskip_x`
skip pixels at beginning of line
- `l4_uint32_t preskip_y`
skip lines
- `l4_uint32_t endskip_x`
skip pixels at end of line

14.57.1 Detailed Description

offsets in pmap[] and bmap[]

Definition at line 69 of file [bitmap.h](#).

The documentation for this struct was generated from the following file:

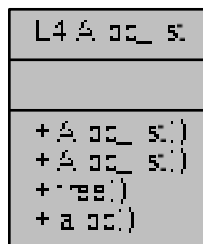
- `l4/libgfxbitmap/bitmap.h`

14.58 L4::Alloc_list Class Reference

A simple list-based allocator.

```
#include <alloc.h>
```

Collaboration diagram for L4::Alloc_list:



14.58.1 Detailed Description

A simple list-based allocator.

Definition at line 33 of file alloc.h.

The documentation for this class was generated from the following file:

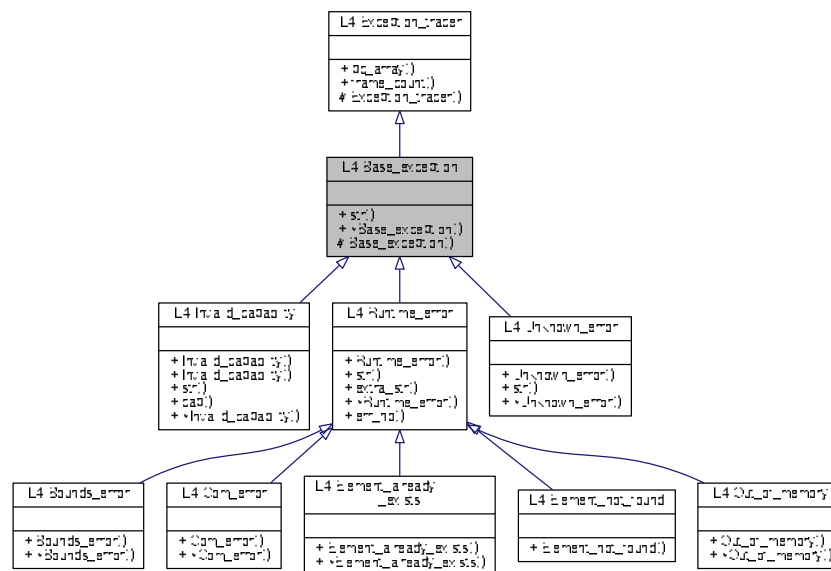
- `l4/cxx/alloc.h`

14.59 L4::Base_exception Class Reference

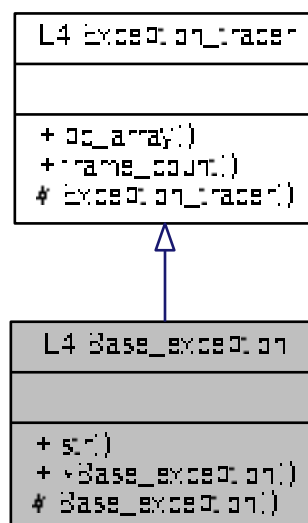
Base class for all exceptions, thrown by the [L4Re](#) framework.

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Base_exception:



Collaboration diagram for L4::Base_exception:



Public Member Functions

- virtual char const * [str](#) () const =0 throw ()
Return a human readable string for the exception.
- virtual [~Base_exception](#) () throw ()
Destruction.

Protected Member Functions

- [Base_exception](#) () throw ()
Create a base exception.

14.59.1 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 [116](#) of file [exceptions](#).

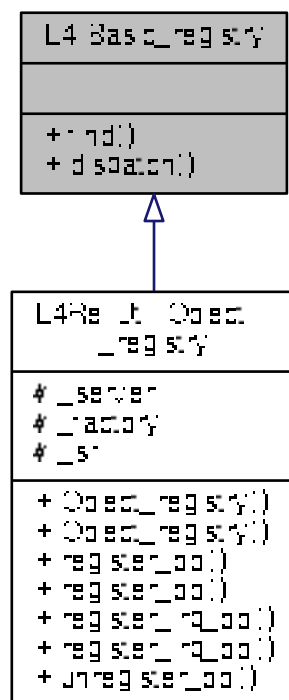
The documentation for this class was generated from the following file:

- [l4/cxx/exceptions](#)

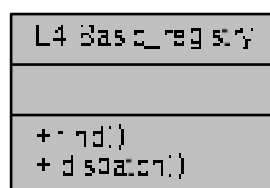
14.60 L4::Basic_registry Class Reference

This registry returns the corresponding server object based on the label of an [lpc_gate](#).

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.

14.60.1 Detailed Description

This registry returns the corresponding server object based on the label of an [lpc_gate](#).

Definition at line 469 of file [ipc_epiface](#).

14.60.2 Member Function Documentation

14.60.2.1 dispatch()

```
static 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.

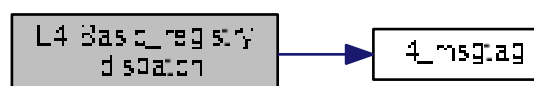
Returns

The return code from the object's dispatch function or -L4_ENOENT if the object does not exist.

Definition at line 494 of file [ipc_epiface](#).

References [L4_ENOENT](#), and [l4_msgtag\(\)](#).

Here is the call graph for this function:



14.60.2.2 find()

```
static 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 [478](#) of file [ipc_epiface](#).

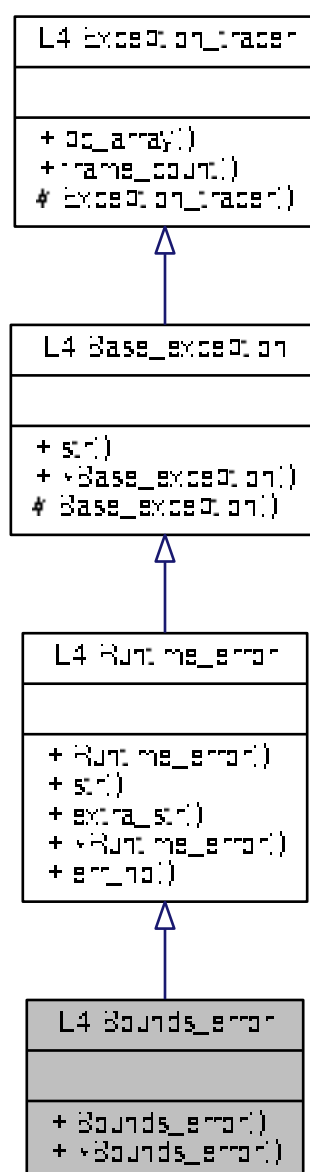
The documentation for this class was generated from the following file:

- [l4/sys/cxx/ipc_epiface](#)

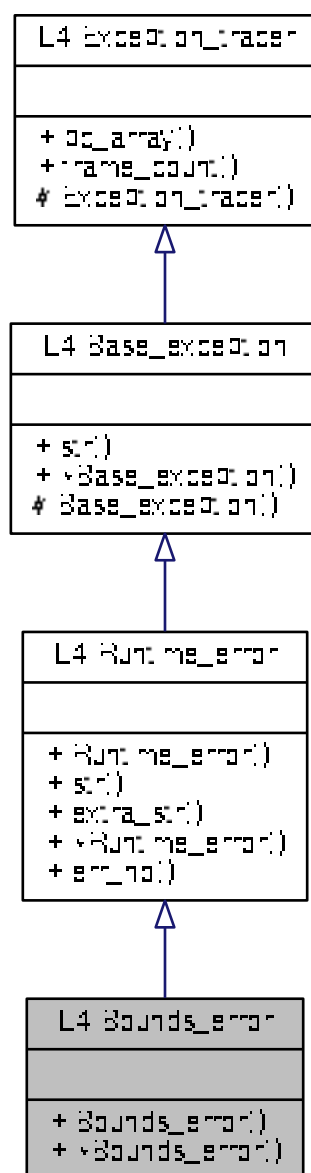
14.61 L4::Bounds_error Class Reference

Access out of bounds.

Inheritance diagram for L4::Bounds_error:



Collaboration diagram for L4::Bounds_error:



Additional Inherited Members

14.61.1 Detailed Description

Access out of bounds.

Definition at line 290 of file [exceptions](#).

The documentation for this class was generated from the following file:

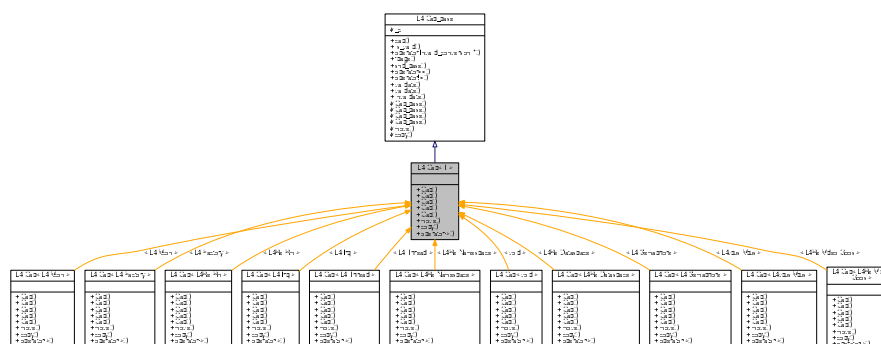
- [l4/cxx/exceptions](#)

14.62 L4::Cap< T > Class Template Reference

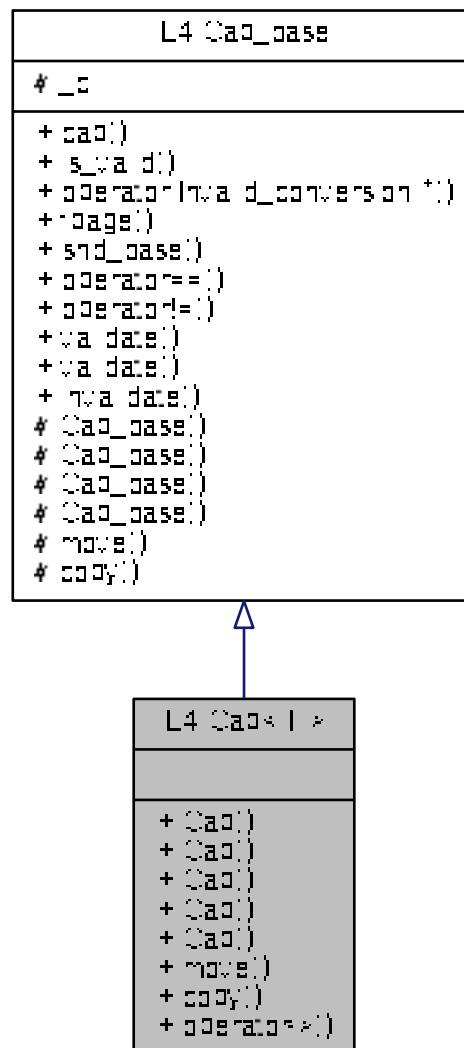
C++ interface for capabilities.

```
#include <capability.h>
```

Inheritance diagram for L4::Cap< T >:



Collaboration diagram for L4::Cap< T >:



Public Member Functions

- `template<typename O >`
`Cap (Cap< O > const &o) throw ()`
Create a copy from o, supporting implicit type casting.
- `Cap (Cap_type cap) throw ()`
Constructor to create an invalid capability selector.
- `Cap (l4_default_caps_t cap) throw ()`
Initialize capability with one of the default capability selectors.
- `Cap (l4_cap_idx_t idx=L4_INVALID_CAP) throw ()`
Initialize capability, defaults to the invalid capability selector.
- `Cap (No_init_type) throw ()`

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 throw ()

Member access of a T.

Friends

- class [L4::Kobject](#)

Additional Inherited Members

14.62.1 Detailed Description

```
template<typename T>
class L4::Cap< T >
```

C++ interface for capabilities.

Template Parameters

T	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/client.cc](#), [examples/libs/l4re/c++/shared_ds/ds_clnt.cc](#), and [examples/libs/l4re/streammap/client.cc](#).↔

Definition at line 13 of file [capability.h](#).

14.62.2 Constructor & Destructor Documentation

14.62.2.1 [Cap\(\)](#) [1/4]

```
template<typename T>
template<typename O >
L4::Cap< T >::Cap (
    Cap< O > const & o ) throw ()    [inline]
```

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 241 of file [capability.h](#).

14.62.2.2 **Cap()** [2/4]

```
template<typename T>
L4::Cap< T >::Cap (
    Cap_type cap ) throw )    [inline]
```

Constructor to create an invalid capability selector.

Parameters

<i>cap</i>	Capability selector.
------------	----------------------

Definition at line 248 of file [capability.h](#).

14.62.2.3 **Cap()** [3/4]

```
template<typename T>
L4::Cap< T >::Cap (
    l4_default_caps_t cap ) throw )    [inline]
```

Initialize capability with one of the default capability selectors.

Parameters

<i>cap</i>	Capability selector.
------------	----------------------

Definition at line 254 of file [capability.h](#).

14.62.2.4 **Cap()** [4/4]

```
template<typename T>
L4::Cap< T >::Cap (
    l4_cap_idx_t idx = L4_INVALID_CAP ) throw )    [inline], [explicit]
```

Initialize capability, defaults to the invalid capability selector.

Parameters

<i>idx</i>	Capability selector.
------------	----------------------

Definition at line 260 of file [capability.h](#).

14.62.3 Member Function Documentation**14.62.3.1 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 283 of file [capability.h](#).

14.62.3.2 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 273 of file [capability.h](#).

The documentation for this class was generated from the following file:

- [l4/sys/cxx/capability.h](#)

Public Member Functions

- [l4_cap_idx_t cap](#) () const throw ()
Return capability selector.
- bool [is_valid](#) () const throw ()
Test whether the capability is a valid capability index (i.e., not L4_INVALID_CAP).
- [l4_fpage_t fpage](#) (unsigned rights=[L4_FPAGE_RWX](#)) const throw ()
Return flex-page for the capability.
- [l4_umword_t snd_base](#) (unsigned grant=0, [l4_cap_idx_t](#) base=[L4_INVALID_CAP](#)) const throw ()
Return send base.
- bool [operator==](#) ([Cap_base](#) const &o) const throw ()
Test if two capabilities are equal.
- bool [operator!=](#) ([Cap_base](#) const &o) const throw ()
Test if two capabilities are not equal.
- [l4_msgtag_t validate](#) ([l4_utcb_t](#) *u=[l4_utcb\(\)](#)) const throw ()
Check whether a capability is present (refers to an object).
- [l4_msgtag_t validate](#) ([Cap](#)< [Task](#) > task, [l4_utcb_t](#) *u=[l4_utcb\(\)](#)) const throw ()
Check whether a capability is present (refers to an object).
- void [invalidate](#) () throw ()
Set this capability to invalid (L4_INVALID_CAP).

Protected Member Functions

- [Cap_base](#) ([l4_cap_idx_t](#) c) throw ()
Generate a capability from its C representation.
- [Cap_base](#) ([Cap_type](#) cap) throw ()
Constructor to create an invalid capability.
- [Cap_base](#) ([l4_default_caps_t](#) cap) throw ()
Initialize capability with one of the default capabilities.
- [Cap_base](#) () throw ()
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.

14.63.1 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](#).

14.63.2 Member Enumeration Documentation

14.63.2.1 Cap_type

```
enum L4::Cap_base::Cap_type
```

Invalid capability type.

Enumerator

Invalid	Invalid capability selector.
---------	------------------------------

Definition at line 42 of file [capability.h](#).

14.63.2.2 No_init_type

```
enum L4::Cap_base::No_init_type
```

Enumerator

No_init	Special value for constructing uninitialized Cap objects.
---------	---

Definition at line 31 of file [capability.h](#).

14.63.3 Constructor & Destructor Documentation

14.63.3.1 Cap_base() [1/2]

```
L4::Cap_base::Cap_base (
    l4_cap_idx_t c ) throw ()    [inline], [explicit], [protected]
```

Generate a capability from its C representation.

Parameters

c	The C capability
---	------------------

Definition at line 141 of file [capability.h](#).

14.63.3.2 Cap_base() [2/2]

```
L4::Cap_base::Cap_base (
    l4_default_caps_t cap ) throw ()    [inline], [explicit], [protected]
```

Initialize capability with one of the default capabilities.

Parameters

<i>cap</i>	Capability.
------------	-------------

Definition at line 152 of file [capability.h](#).

14.63.4 Member Function Documentation

14.63.4.1 cap()

```
l4_cap_idx_t L4::Cap_base::cap ( ) const throw ()    [inline]
```

Return capability selector.

Returns

Capability selector.

Definition at line 51 of file [capability.h](#).

Referenced by [L4Re::Util::Cap_alloc_base::alloc\(\)](#), [L4::lcu::bind\(\)](#), [L4::cap_cast\(\)](#), [L4::cap_reinterpret_cast\(\)](#), [L4::Irq_mux::chain\(\)](#), [L4Re::Util::Smart_count_cap< Unmap_flags >::copy\(\)](#), [L4::Factory::create_factory\(\)](#), [L4::Factory::create_gate\(\)](#), [L4::Factory::create_irq\(\)](#), [L4::Factory::create_task\(\)](#), [L4::Factory::create_thread\(\)](#), [L4::Factory::create_vm\(\)](#), [L4Re::Util::Smart_cap_auto< Unmap_flags >::free\(\)](#), [L4::Invalid_capability::Invalid_capability\(\)](#), [L4virtio::Svr::Device_t< Ds_data >::mem_info\(\)](#), [L4::Smart_cap< T, SMART >::operator->\(\)](#), [L4::Cap< L4Re::Video::Goos >::operator->\(\)](#), [L4Re::Util::Event_buffer_consumer_t< PAYLOAD >::process\(\)](#), [L4::lpc_svr::Server_iface::rcv_cap\(\)](#), [L4::Smart_cap< T, SMART >::Smart_cap\(\)](#), [L4::lcu::unbind\(\)](#), and [L4vbus::lcu::vicu\(\)](#).

Parameters

<code>src</code>	the source capability.
------------------	------------------------

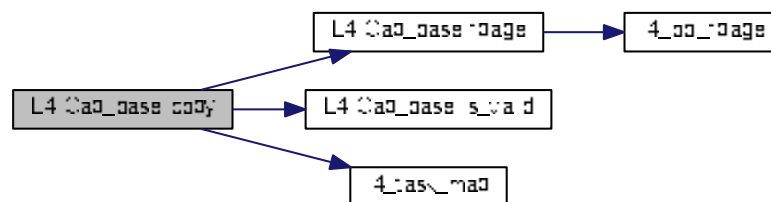
After this operation this capability refers to the same object as `src`.

Definition at line 184 of file `capability.h`.

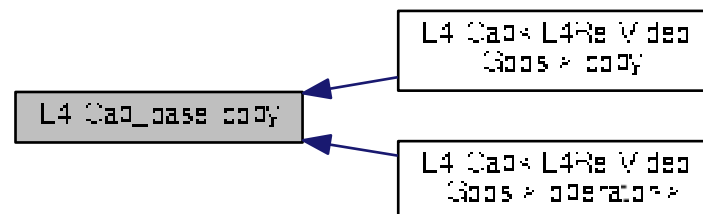
References `fpage()`, `is_valid()`, `L4_BASE_TASK_CAP`, `L4_CAP_FPAGE_RWSD`, and `l4_task_map()`.

Referenced by `L4::Cap< L4Re::Video::Goos >::copy()`, and `L4::Cap< L4Re::Video::Goos >::operator->()`.

Here is the call graph for this function:



Here is the caller graph for this function:

14.63.4.3 `fpage()`

```

l4_fpage_t L4::Cap_base::fpage (
    unsigned rights = L4_FPAGE_RWX ) const throw ()    [inline]

```

Return flex-page for the capability.

Parameters

<i>rights</i>	Rights, defaults to 'rwx'
---------------	---------------------------

Returns

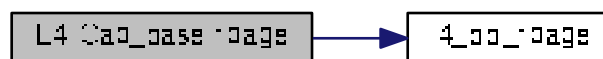
flex-page

Definition at line 71 of file [capability.h](#).

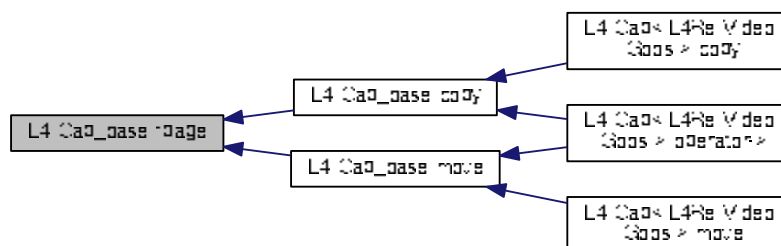
References [l4_obj_fpage\(\)](#).

Referenced by [copy\(\)](#), and [move\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



14.63.4.4 is_valid()

```
bool L4::Cap_base::is_valid ( ) const throw ( ) [inline]
```

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

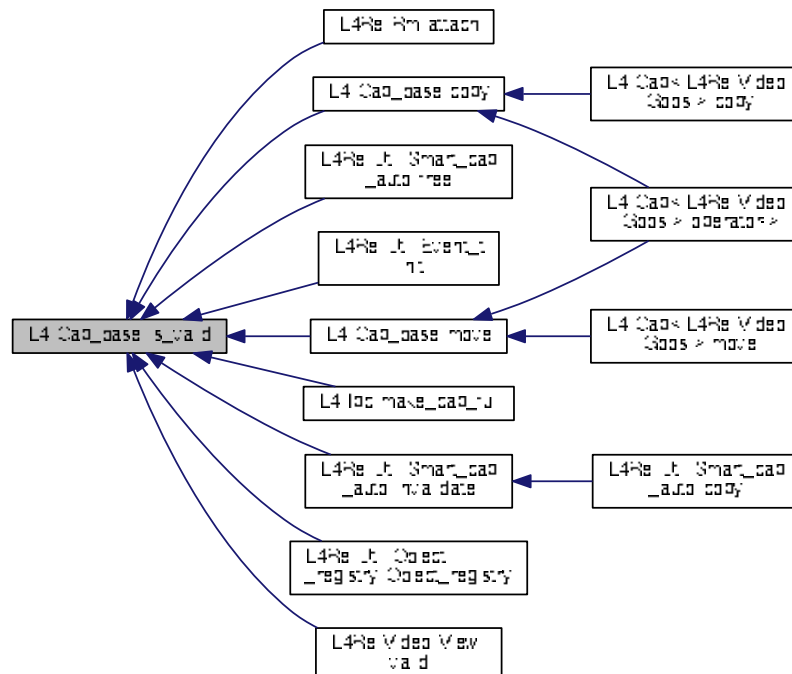
Examples:

[examples/libs/l4re/c++/mem_alloc/mem_rm.cc](#), and [examples/libs/l4re/c++/shared_ds/ds_clnt.cc](#).

Definition at line 59 of file [capability.h](#).

Referenced by [L4Re::Rm::attach\(\)](#), [copy\(\)](#), [L4Re::Util::Smart_cap_auto< Unmap_flags >::free\(\)](#), [L4Re::Util::Event_t< PAYLOAD >::init\(\)](#), [L4Re::Util::Smart_cap_auto< Unmap_flags >::invalidate\(\)](#), [L4::Ipc::make_cap_full\(\)](#), [move\(\)](#), [L4Re::Util::Object_registry::Object_registry\(\)](#), and [L4Re::Video::View::valid\(\)](#).

Here is the caller graph for this function:

**14.63.4.5 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.
------------------	------------------------

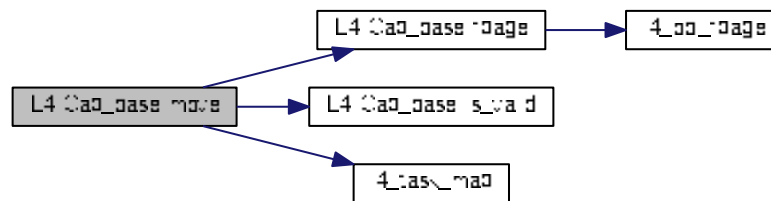
After the operation this capability refers to the object formerly referred to by the source capability `src`, and the source capability no longer refers to an object.

Definition at line 168 of file [capability.h](#).

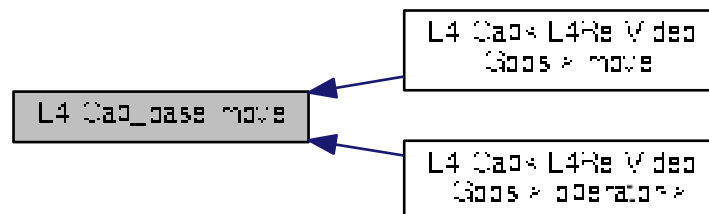
References [fpage\(\)](#), [is_valid\(\)](#), [L4_BASE_TASK_CAP](#), [L4_CAP_FPAGE_RWSD](#), [L4_MAP_ITEM_GRANT](#), and [l4_task_map\(\)](#).

Referenced by [L4::Cap< L4Re::Video::Goos >::move\(\)](#), and [L4::Cap< L4Re::Video::Goos >::operator->\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



14.63.4.6 `snd_base()`

```

l4_umword_t L4::Cap_base::snd_base (
    unsigned grant = 0,
    l4_cap_idx_t base = L4_INVALID_CAP ) const throw ()    [inline]
  
```

Return send base.

Parameters

<i>grant</i>	True object should be granted.
<i>base</i>	Base capability (first in a bundle of aligned capabilities)

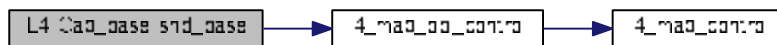
Returns

Map object.

Definition at line 82 of file [capability.h](#).

References [L4_INVALID_CAP](#), and [l4_map_obj_control\(\)](#).

Here is the call graph for this function:



14.63.4.7 validate() [1/2]

```
l4_msgtag_t L4::Cap_base::validate (
    l4_utcb_t * u = l4_utcb() ) const throw()    [inline]
```

Check whether a capability is present (refers to an object).

Parameters

<i>u</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.
----------	---

Return values

<i>tag.label()</i>	> 0 Capability is present (refers to an object).
<i>tag.label()</i>	== 0 No capability present (void object).

A capability is considered present when it refers to an existing kernel object.

Definition at line 87 of file [capability](#).

14.63.4.8 validate() [2/2]

```
l4_msgtag_t L4::Cap_base::validate (
    Cap< Task > task,
    l4_utcb_t * u = l4_utcb() ) const throw ()    [inline]
```

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, usually the UTCB of the calling thread.

Return values

<i>tag.label()</i>	> 0 Capability is present (refers to an object).
<i>tag.label()</i>	== 0 No capability present (void object).

A capability is considered present when it refers to an existing kernel object.

Definition at line 83 of file [capability](#).

14.63.5 Field Documentation**14.63.5.1 _c**

```
l4_cap_idx_t L4::Cap_base::_c    [protected]
```

The C representation of a capability selector.

Definition at line 195 of file [capability.h](#).

Referenced by [L4::Smart_cap< T, SMART >::operator->\(\)](#), and [L4::Smart_cap< T, SMART >::Smart_cap\(\)](#).

The documentation for this class was generated from the following files:

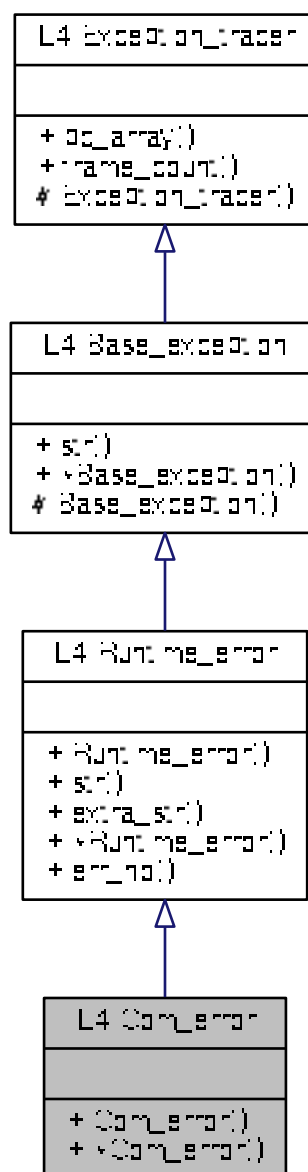
- I4/sys/cxx/capability.h
- I4/sys/[capability](#)

14.64 L4::Com_error Class Reference

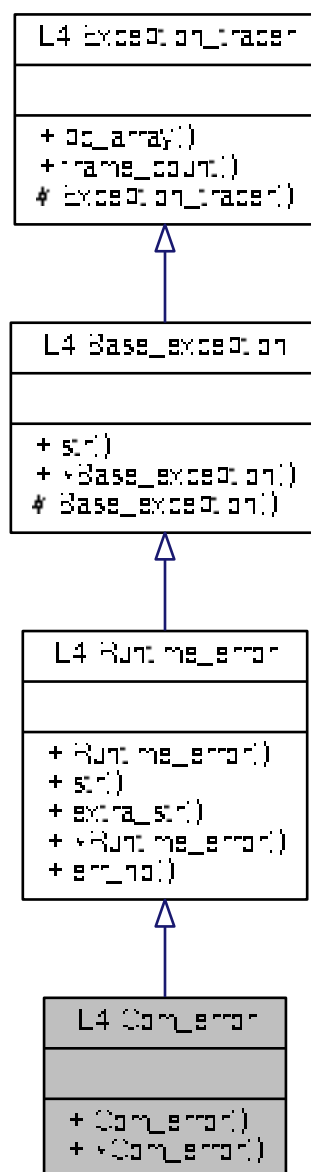
Error conditions during IPC.

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Com_error:



Collaboration diagram for L4::Com_error:



Public Member Functions

- [Com_error](#) (long err) throw ()

Create a [Com_error](#) for the given [L4](#) IPC error code.

Additional Inherited Members

14.64.1 Detailed Description

Error conditions during IPC.

This exception encapsulates all IPC error conditions of [L4 IPC](#).

Definition at line [275](#) of file [exceptions](#).

14.64.2 Constructor & Destructor Documentation

14.64.2.1 Com_error()

```
L4::Com_error::Com_error (
    long err ) throw ()    [inline], [explicit]
```

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 [282](#) of file [exceptions](#).

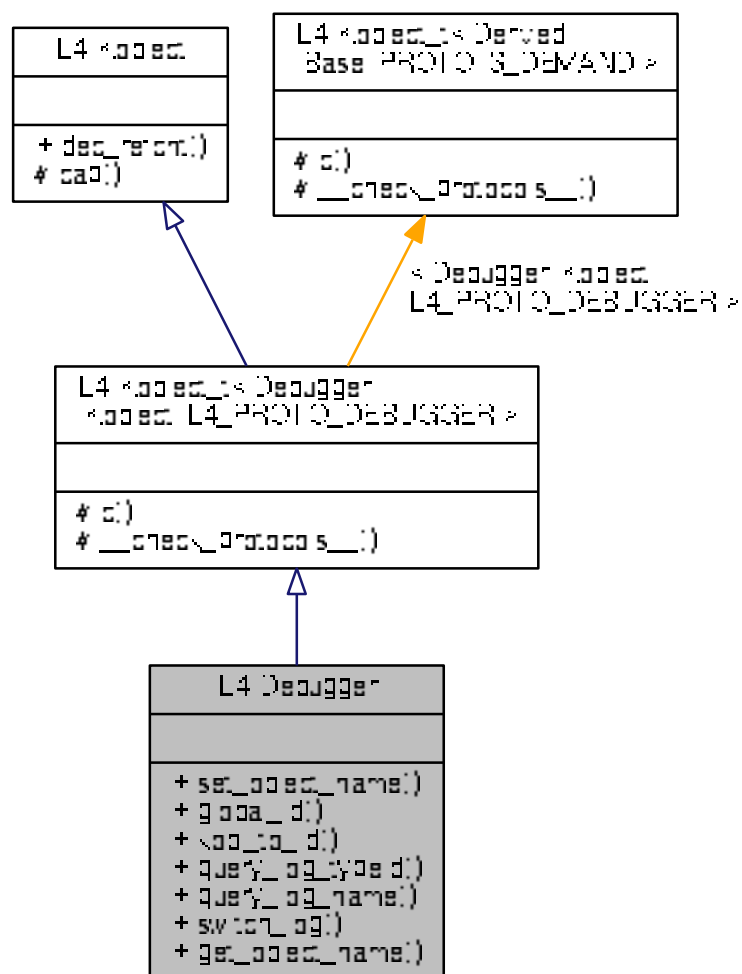
The documentation for this class was generated from the following file:

- [l4/cxx/exceptions](#)

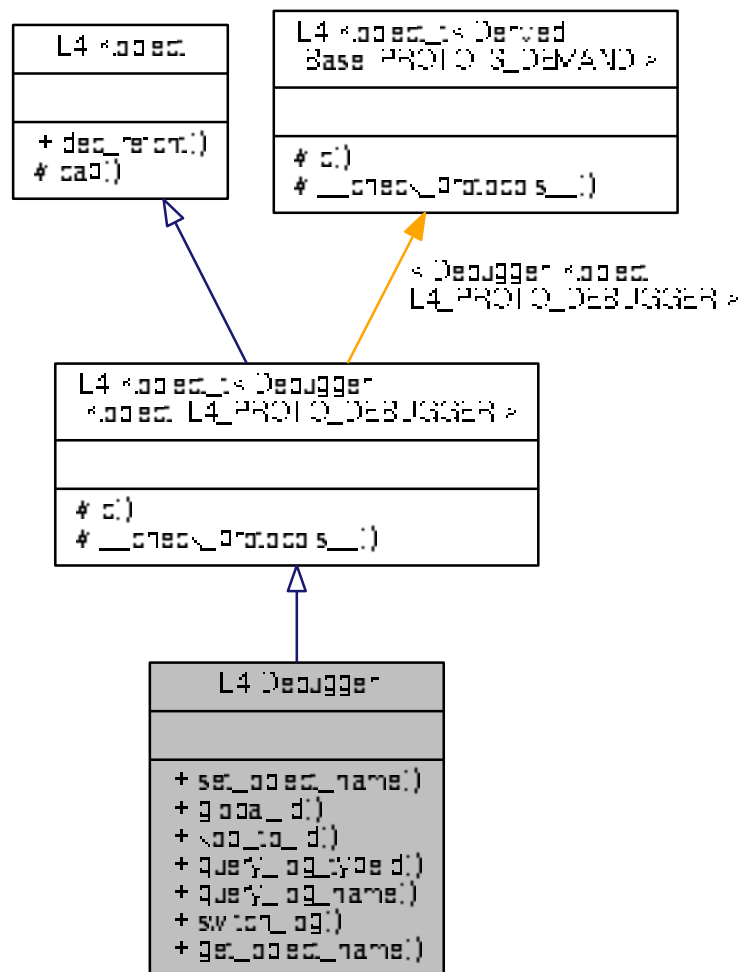
14.65 L4::Debugger Class Reference

C++ debugger interface.

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\(\)](#)) throw ()
Set the name of a kernel object.
- unsigned long [global_id](#) ([l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) throw ()
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\(\)](#)) throw ()
Get the globally unique ID of the object behind the kobject pointer.
- int [query_log_typeid](#) (const char *name, unsigned idx, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) throw ()
Query the log-id for a log type.
- int [query_log_name](#) (unsigned idx, char *name, unsigned namelen, char *shortname, unsigned shortname-len, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) throw ()
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\(\)](#)) throw ()

Set or unset log.

- [l4_msgtag_t get_object_name](#) (unsigned id, char *name, unsigned size, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) throw ()

Get name of object with Id `id`.

Additional Inherited Members

14.65.1 Detailed Description

C++ debugger interface.

Attention

This API is subject to change! Do not rely on it in production code.

Include File

```
#include <l4/sys/debugger>
```

Definition at line 39 of file [debugger](#).

14.65.2 Member Function Documentation

14.65.2.1 `get_object_name()`

```
l4\_msgtag\_t L4::Debugger::get_object_name (
    unsigned id,
    char * name,
    unsigned size,
    l4\_utcb\_t * utcb = l4\_utcb\(\) ) throw ()    [inline]
```

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 <code>name</code> buffer.
	<i>utcb</i>	The UTCB to use for the operation.

Returns

Syscall return tag

Definition at line 145 of file [debugger](#).

14.65.2.2 `global_id()`

```
unsigned long L4::Debugger::global_id (
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline]
```

Get the globally unique ID of the object behind a capability.

Parameters

<i>utcb</i>	The UTCB to use for the operation.
-------------	------------------------------------

Return values

$\sim 0UL$	The capability is invalid.
≥ 0	The global debugger id.

Definition at line 68 of file [debugger](#).

14.65.2.3 `kobj_to_id()`

```
unsigned long L4::Debugger::kobj_to_id (
    l4_addr_t kobjp,
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline]
```

Get the globally unique ID of the object behind the kobject pointer.

Parameters

<i>kobjp</i>	Kobject pointer
<i>utcb</i>	The UTCB to use for the operation.

Return values

$\sim 0UL$	The capability or the Kobject pointer are invalid.
≥ 0	The globally unique id.

Definition at line 80 of file [debugger](#).

14.65.2.4 `query_log_name()`

```
int L4::Debugger::query_log_name (
    unsigned idx,
    char * name,
    unsigned namelen,
```

```
char * shortname,
unsigned shortnamelen,
l4_utcb_t * utcb = l4_utcb() ) throw )    [inline]
```

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 <i>shortname</i> to. The buffer must be allocated by the caller.
	<i>shortnamelen</i>	Buffer length of <i>shortname</i> .
	<i>utcb</i>	The UTCB to use for the operation.

Return values

0	Success
<0	Error

Definition at line 113 of file [debugger](#).

14.65.2.5 query_log_typeid()

```
int L4::Debugger::query_log_typeid (
    const char * name,
    unsigned idx,
    l4_utcb_t * utcb = l4_utcb() ) throw )    [inline]
```

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>	The UTCB to use for the operation.

Return values

>=0	Id
<0	Error

Definition at line 94 of file [debugger](#).

14.65.2.6 `set_object_name()`

```
l4_msgtag_t L4::Debugger::set_object_name (
    const char * name,
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline]
```

Set the name of a kernel object.

Parameters

<i>name</i>	Name
<i>utcb</i>	The UTCB to use for the operation.

Returns

System call return tag.

Definition at line 56 of file [debugger](#).

14.65.2.7 `switch_log()`

```
l4_msgtag_t L4::Debugger::switch_log (
    const char * name,
    unsigned on_off,
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline]
```

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>	The UTCB to use for the operation.

Returns

Syscall return tag

Definition at line 130 of file [debugger](#).

The documentation for this class was generated from the following file:

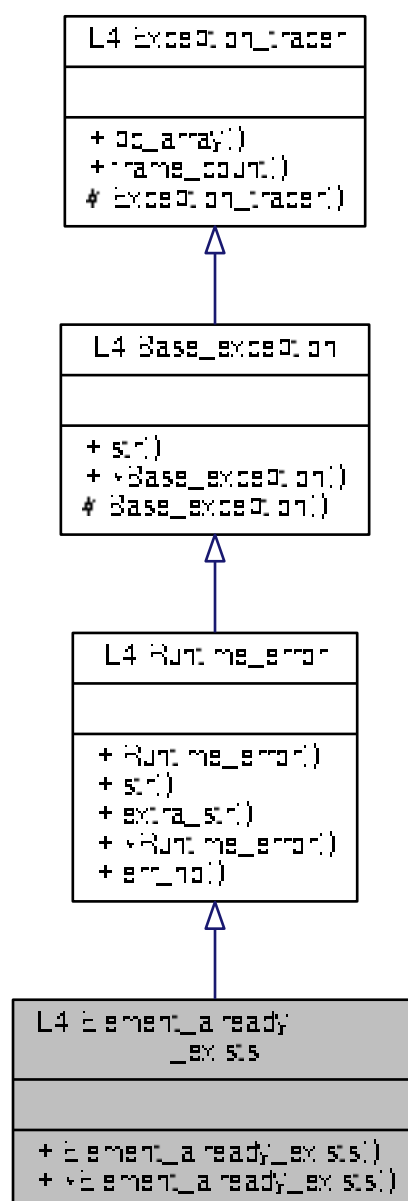
- [l4/sys/debugger](#)

14.66 L4::Element_already_exists Class Reference

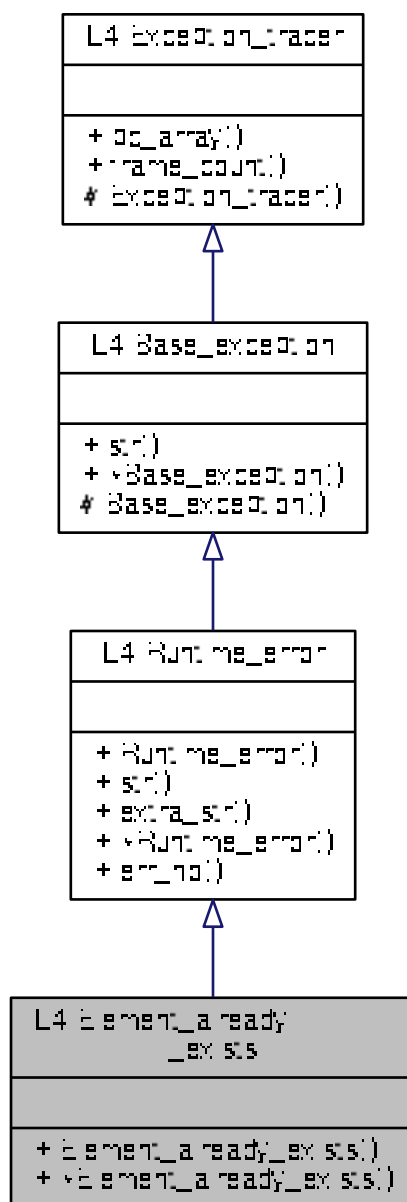
Exception for duplicate element insertions.

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Element_already_exists:



Collaboration diagram for L4::Element_already_exists:



Additional Inherited Members

14.66.1 Detailed Description

Exception for duplicate element insertions.

Definition at line 203 of file [exceptions](#).

The documentation for this class was generated from the following file:

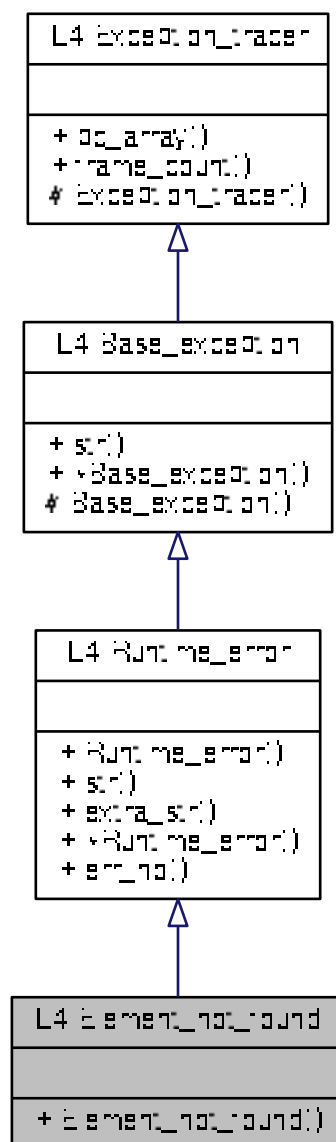
- [l4/cxx/exceptions](#)

14.67 L4::Element_not_found Class Reference

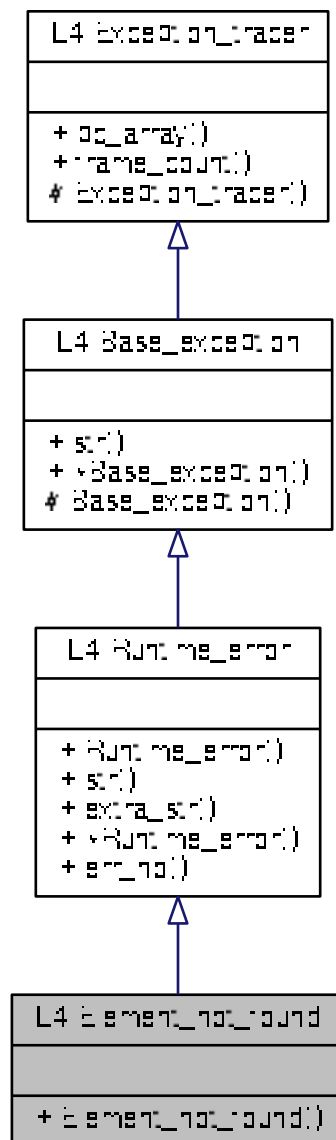
Exception for a failed lookup (element not found).

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Element_not_found:



Collaboration diagram for L4::Element_not_found:



Additional Inherited Members

14.67.1 Detailed Description

Exception for a failed lookup (element not found).

Definition at line 232 of file [exceptions](#).

The documentation for this class was generated from the following file:

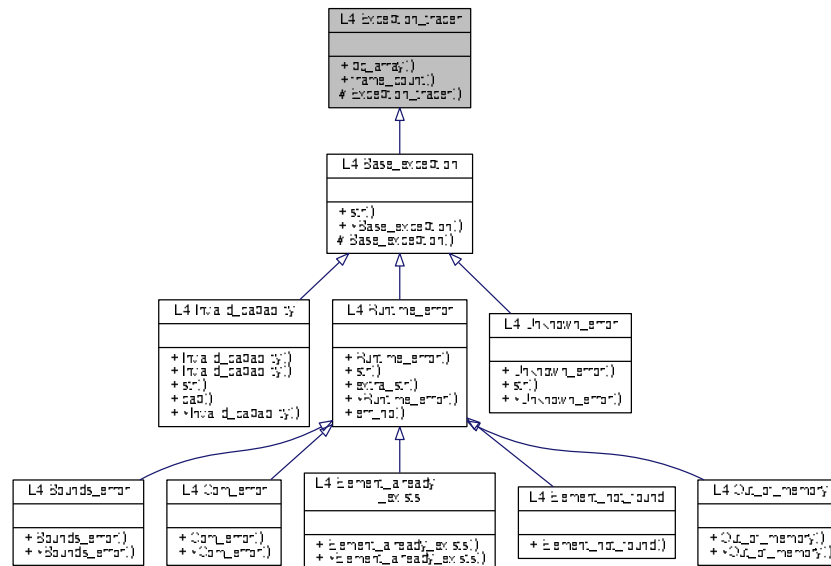
- [l4/cxx/exceptions](#)

14.68 L4::Exception_tracer Class Reference

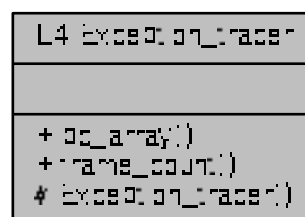
Back-trace support for exceptions.

```
#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 throw ()
Get the array containing the call trace.
- int [frame_count](#) () const throw ()
Get the number of entries that are valid in the call trace.

Protected Member Functions

- [Exception_tracer](#) () throw ()

Create a back trace.

14.68.1 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 63 of file [exceptions](#).

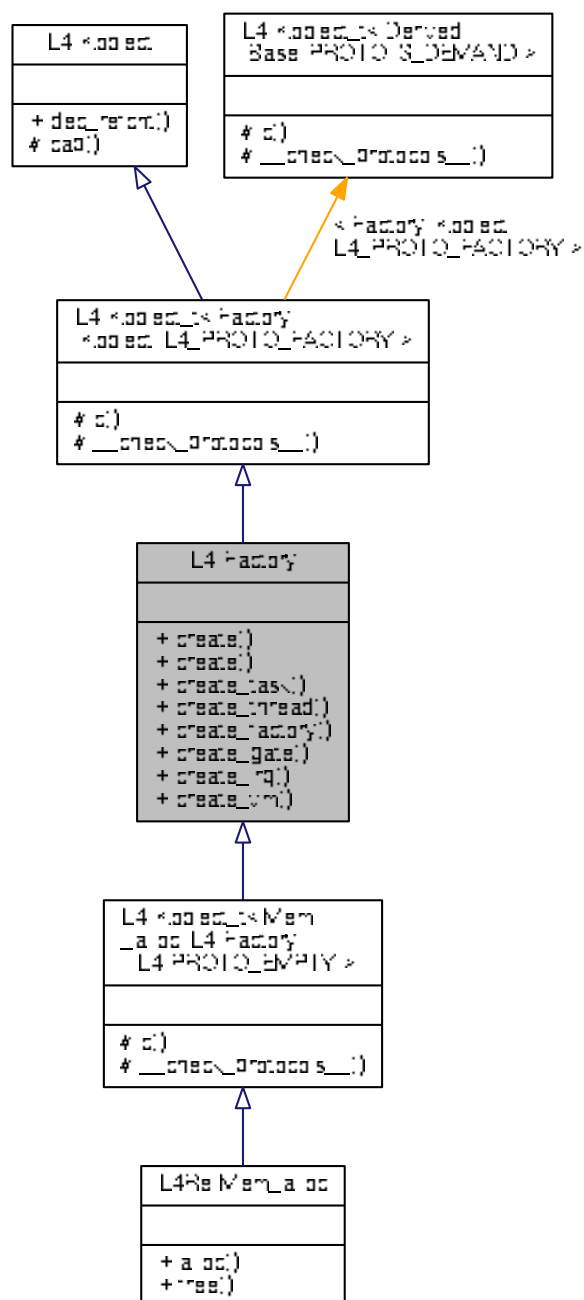
The documentation for this class was generated from the following file:

- [l4/cxx/exceptions](#)

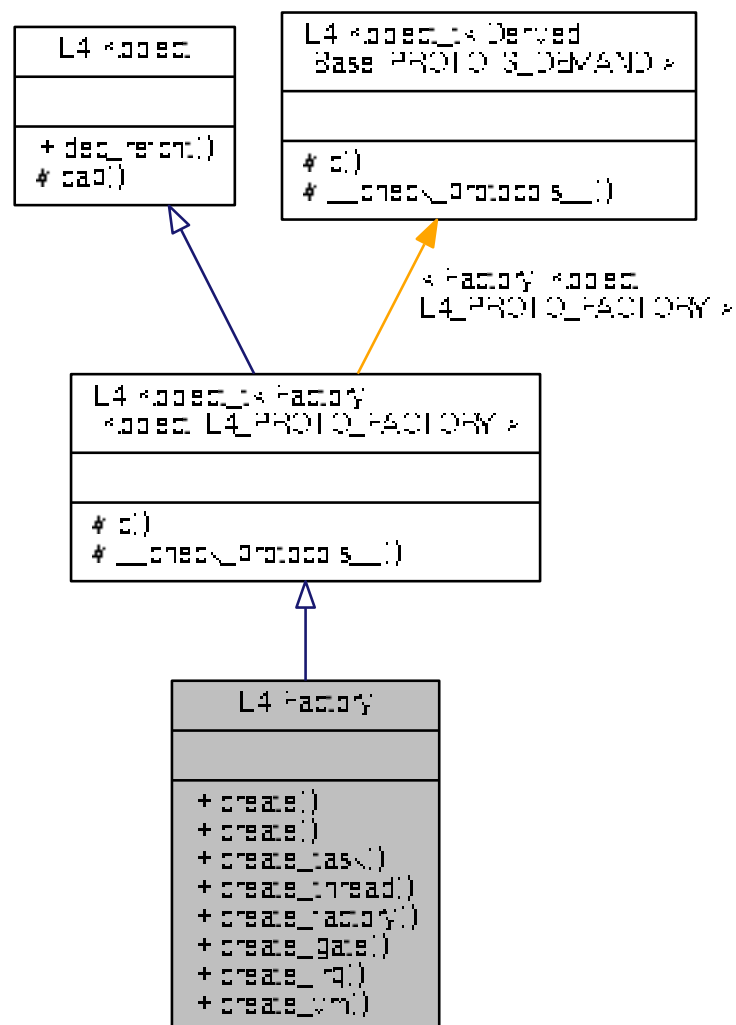
14.69 L4::Factory Class Reference

C++ [Factory](#) interface to create kernel objects.

Inheritance diagram for L4::Factory:



Collaboration diagram for L4::Factory:



Data Structures

- struct [Lstr](#)
Special type to add a pascal string into the factory create stream.
- struct [Nil](#)
Special type to add a void argument 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](#)()) throw ()

Generic create call to the factory.

- `template<typename OBJ >
S create (Cap< OBJ > target, l4_utcb_t *utcb=l4_utcb()) throw ()`
Create call for typed capabilities.
- `l4_msgtag_t create_task (Cap< Task > const &target_cap, l4_fpage_t const &utcb_area, l4_utcb_t *utcb=l4_utcb()) throw ()`
Create a new task.
- `l4_msgtag_t create_thread (Cap< Thread > const &target_cap, l4_utcb_t *utcb=l4_utcb())) throw ()`
Create a new thread.
- `l4_msgtag_t create_factory (Cap< Factory > const &target_cap, unsigned long limit, l4_utcb_t *utcb=l4_utcb()) throw ()`
Create a new factory.
- `l4_msgtag_t create_gate (Cap< Kobject > const &target_cap, Cap< Thread > const &thread_cap, l4_umword_t label, l4_utcb_t *utcb=l4_utcb()) throw ()`
Create a new IPC gate.
- `l4_msgtag_t create_irq (Cap< Irq > const &target_cap, l4_utcb_t *utcb=l4_utcb())) throw ()`
Create a new IRQ.
- `l4_msgtag_t create_vm (Cap< Vm > const &target_cap, l4_utcb_t *utcb=l4_utcb())) throw ()`
Create a new virtual machine.

Additional Inherited Members

14.69.1 Detailed Description

C++ [Factory](#) interface to create kernel objects.

A factory is used to create all kinds of kernel objects:

- [L4::Task](#)
- [L4::Thread](#)
- [L4::Factory](#)
- [L4::lpc_gate](#)
- [L4::Irq](#)
- [L4::Vm](#)

The factory is equipped with a limit that limits the amount of kernel memory available to that factory.

Note

The limit does not give any guarantee for the amount of available kernel memory.

Include File

```
#include <l4/sys/factory>
```

For the C interface refer to [Factory](#).

Definition at line 55 of file [factory](#).

14.69.2 Member Function Documentation

14.69.2.1 create() [1/2]

```
S L4::Factory::create (
    Cap< void > target,
    long obj,
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline]
```

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 objects's capability into this slot.
	<i>obj</i>	The protocol ID that specifies which kind of object shall be created.
	<i>utcb</i>	The UTCB to use for the operation.

Returns

A create stream that allows adding additional arguments to the [create\(\)](#) call.

This method does currently not directly invoke the factory. It returns a stream that shall invoke the factory after adding all additional arguments.

Usage:

```
L4::Cap<L4Re::Namespace> ns = L4Re::Util::cap_alloc.
    alloc<L4Re::Namespace>();
factory->create(ns, L4Re::Namespace::Protocol) << "Argument text";
```

Definition at line 253 of file [factory](#).

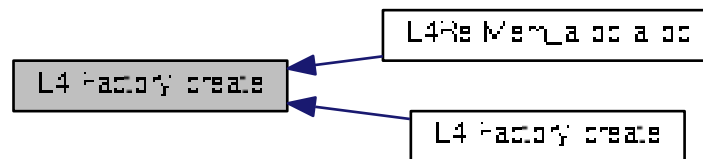
References [L4::Kobject::cap\(\)](#).

Referenced by [L4Re::Mem_alloc::alloc\(\)](#), and [create\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



14.69.2.2 create() [2/2]

```

template<typename OBJ >
S L4::Factory::create (
    Cap< OBJ > target,
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline]
  
```

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 use.

Returns

Argument stream to call the factory.

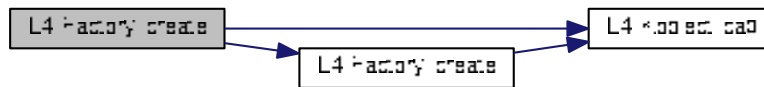
See also

[create](#)

Definition at line 270 of file [factory](#).

References [L4::Kobject::cap\(\)](#), [create\(\)](#), and [L4_INLINE_RPC_NF](#).

Here is the call graph for this function:



14.69.2.3 create_factory()

```

l4_msgtag_t L4::Factory::create_factory (
    Cap< Factory > const & target_cap,
    unsigned long limit,
    l4_utcb_t * utcb = l4_utcb() ) throw() [inline]
  
```

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>	The UTCB to use for the operation.

Returns

Syscall return tag

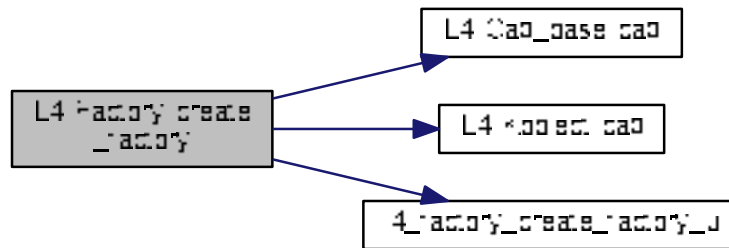
Note

The limit of the new factory is subtracted from the available amount of the factory used for creation.

Definition at line 336 of file [factory](#).

References [L4::Cap_base::cap\(\)](#), [L4::Kobject::cap\(\)](#), and [l4_factory_create_factory_u\(\)](#).

Here is the call graph for this function:



14.69.2.4 create_gate()

```

l4_msgtag_t L4::Factory::create_gate (
    Cap< Kobject > const & target_cap,
    Cap< Thread > const & thread_cap,
    l4_umword_t label,
    l4_utcb_t * utcb = l4_utcb() ) throw() [inline]
  
```

Create a new IPC gate.

Parameters

out	<i>target_cap</i>	The kernel stores the new IPC gate's capability into this slot.
	<i>thread_cap</i>	Optional capability selector of the thread to bind the gate to. Use L4_INVALID_CAP to create an unbound IPC gate.
	<i>label</i>	Optional label of the gate (is used if <i>thread_cap</i> is valid).
	<i>utcb</i>	The UTCB to use for the operation.

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_ENOENT</i>	<i>thread_cap</i> is void or points to something that is not a thread.
<i>-L4_EPERM</i>	No write rights on <i>thread_cap</i> .

An unbound IPC gate can be bound to a thread using [L4::lpc_gate::bind_thread\(\)](#).

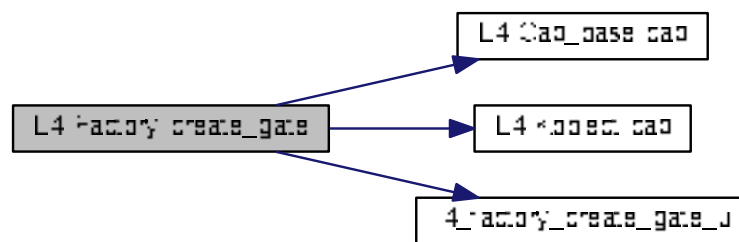
See also

[L4::lpc_gate](#)

Definition at line 366 of file [factory](#).

References [L4::Cap_base::cap\(\)](#), [L4::Kobject::cap\(\)](#), and [l4_factory_create_gate_u\(\)](#).

Here is the call graph for this function:



14.69.2.5 create_irq()

```

l4_msgtag_t L4::Factory::create_irq (
    Cap< Irq >const & target_cap,
    l4_utcb_t * utcb = l4_utcb() ) throw() [inline]
  
```

Create a new IRQ.

Parameters

out	<i>target_cap</i>	The kernel stores the new IRQ's capability into this slot.
	<i>utcb</i>	The UTCB to use for the operation.

Returns

Syscall return tag

Deprecated Use [create\(\)](#) with [Cap<Irq>](#) as argument instead.

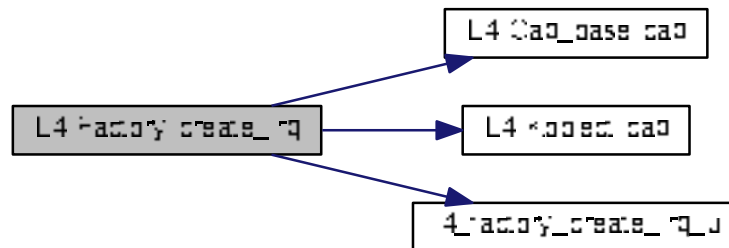
See also

[L4::Irq](#)

Definition at line 384 of file [factory](#).

References [L4::Cap_base::cap\(\)](#), [L4::Kobject::cap\(\)](#), and [l4_factory_create_irq_u\(\)](#).

Here is the call graph for this function:



14.69.2.6 create_task()

```

l4_msgtag_t L4::Factory::create_task (
    Cap< Task > const & target_cap,
    l4_fpage_t const & utcb_area,
    l4_utcb_t * utcb = l4_utcb() ) throw() [inline]
  
```

Create a new task.

Parameters

out	<i>target_cap</i>	The kernel stores the new task's capability into this slot.
	<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.
	<i>utcb</i>	The UTCB to use for the operation.

Returns

Syscall return tag

Note

The size of the UTCB area specifies indirectly the number of UTCBs available for this task. Refer to [L4::Task::add_ku_mem](#) / [l4_task_add_ku_mem\(\)](#) for adding more of this type of memory.

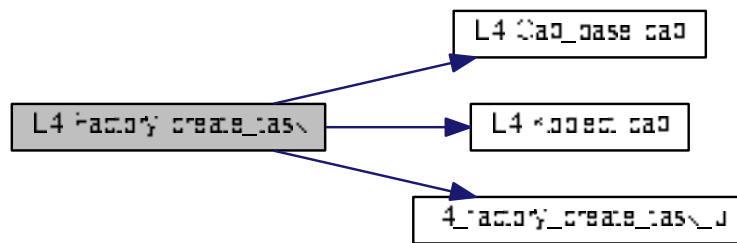
See also

[L4::Task](#)

Definition at line 300 of file [factory](#).

References [L4::Cap_base::cap\(\)](#), [L4::Kobject::cap\(\)](#), and [l4_factory_create_task_u\(\)](#).

Here is the call graph for this function:



14.69.2.7 create_thread()

```

l4_msgtag_t L4::Factory::create_thread (
    Cap< Thread > const & target_cap,
    l4_utcb_t * utcb = l4_utcb() ) throw() [inline]
  
```

Create a new thread.

Parameters

out	<i>target_cap</i>	The kernel stores the new thread's capability into this slot.
	<i>utcb</i>	The UTCB to use for the operation.

Returns

Syscall return tag

Deprecated Use [create\(\)](#) with [Cap<Thread>](#) as argument instead.

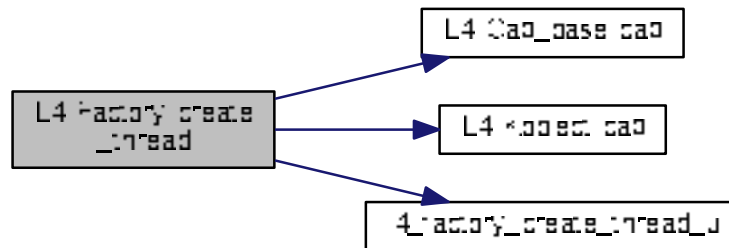
See also

[L4::Thread](#)

Definition at line 318 of file [factory](#).

References [L4::Cap_base::cap\(\)](#), [L4::Kobject::cap\(\)](#), and [l4_factory_create_thread_u\(\)](#).

Here is the call graph for this function:



14.69.2.8 create_vm()

```

l4_msgtag_t L4::Factory::create_vm (
    Cap< Vm >const & target_cap,
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline]
  
```

Create a new virtual machine.

Parameters

out	<i>target_cap</i>	The kernel stores the new VM's capability into this slot.
	<i>utcb</i>	The UTCB to use for the operation.

Returns

Syscall return tag

Deprecated Use `create()` with `Cap<Vm>` as argument instead.

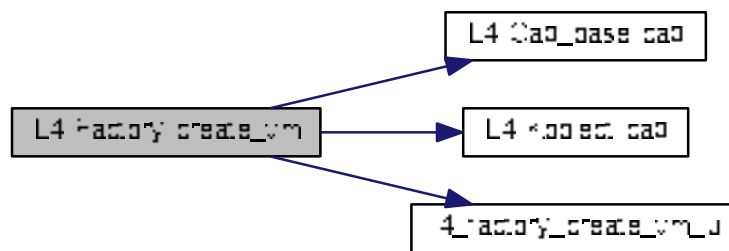
See also

[L4::Vm](#)

Definition at line 402 of file [factory](#).

References [L4::Cap_base::cap\(\)](#), [L4::Kobject::cap\(\)](#), and [l4_factory_create_vm_u\(\)](#).

Here is the call graph for this function:



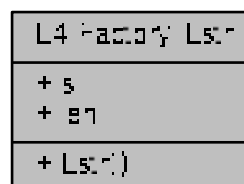
The documentation for this class was generated from the following file:

- [l4/sys/factory](#)

14.70 L4::Factory::Lstr Struct Reference

Special type to add a pascal string into the factory create stream.

Collaboration diagram for L4::Factory::Lstr:



Public Member Functions

- [Lstr](#) (char const *[s](#), int [len](#))

Data Fields

- char const * [s](#)
The character buffer.
- int [len](#)
The number of characters in the buffer.

14.70.1 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 71 of file [factory](#).

14.70.2 Constructor & Destructor Documentation

14.70.2.1 Lstr()

```
L4::Factory::Lstr::Lstr (  
    char const * s,  
    int len ) [inline]
```

Parameters

<i>s</i>	Pointer to the c-style string.
<i>len</i>	Length in number of characters of the string s.

Definition at line 87 of file [factory](#).

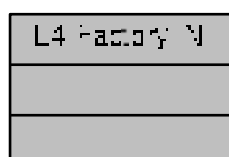
The documentation for this struct was generated from the following file:

- [l4/sys/factory](#)

14.71 L4::Factory::Nil Struct Reference

Special type to add a void argument into the factory create stream.

Collaboration diagram for L4::Factory::Nil:



14.71.1 Detailed Description

Special type to add a void argument into the factory create stream.

Definition at line 64 of file [factory](#).

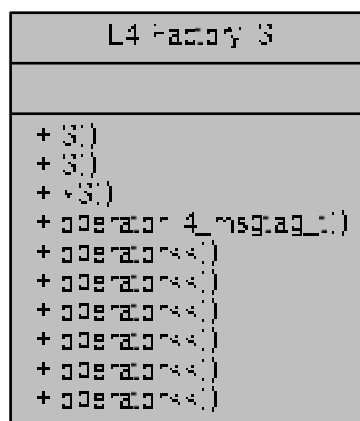
The documentation for this struct was generated from the following file:

- [l4/sys/factory](#)

14.72 L4::Factory::S Class Reference

Stream class for the [create\(\)](#) argument stream.

Collaboration diagram for L4::Factory::S:



Public Member Functions

- [S](#) ([S](#) const &o)
Create a copy.
- [S](#) ([l4_cap_idx_t](#) f, long obj, [L4::Cap](#)< void > target, [l4_utcb_t](#) *utcb) throw ()
Create a stream for a specific [create\(\)](#) call.
- [~S](#) ()
Commit the operation in the destructor to have a cool syntax for [create\(\)](#).
- [operator l4_msgtag_t](#) ()
Explicitly commits the operation and returns the result.
- [S](#) & [operator<<](#) ([l4_mword_t](#) i)
Put a single [l4_mword_t](#) as next argument.
- [S](#) & [operator<<](#) ([l4_umword_t](#) i)
Put a single [l4_umword_t](#) as next argument.

- [S](#) & [operator<<](#) (char const *s)
Add a zero-terminated string as next argument.
- [S](#) & [operator<<](#) ([Lstr](#) const &s)
Add a pascal string as next argument.
- [S](#) & [operator<<](#) ([Nil](#))
Add an empty argument.
- [S](#) & [operator<<](#) ([l4_fpage_t](#) d)
Add a flex page as next argument.

14.72.1 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 96 of file [factory](#).

14.72.2 Constructor & Destructor Documentation

14.72.2.1 [S\(\)](#) [1/2]

```
L4::Factory::S::S (
    S const & o ) [inline]
```

Create a copy.

Parameters

<i>o</i>	Instance of S to copy.
----------	--

Definition at line 109 of file [factory](#).

References [l4_msgtag_t::raw](#).

14.72.2.2 [S\(\)](#) [2/2]

```
L4::Factory::S::S (
    l4_cap_idx_t f,
    long obj,
    L4::Cap< void > target,
    l4_utcb_t * utcb ) throw () [inline]
```

Create a stream for a specific [create\(\)](#) call.

Parameters

	<i>f</i>	The capability for the factory object (L4::Factory).
	<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>	The UTCB to use for the operation.

Definition at line 124 of file [factory](#).

14.72.3 Member Function Documentation

14.72.3.1 operator l4_msgtag_t()

```
L4::Factory::S::operator l4_msgtag_t ( ) [inline]
```

Explicitly commits the operation and returns the result.

Returns

The result of the [create\(\)](#) operation.

Definition at line 144 of file [factory](#).

14.72.3.2 operator<<() [1/6]

```
S& L4::Factory::S::operator<< (
    l4_mword_t i ) [inline]
```

Put a single `l4_mword_t` as next argument.

Parameters

<i>i</i>	The value to add as next argument.
----------	------------------------------------

Returns

Reference to this stream.

Definition at line 158 of file [factory](#).

14.72.3.3 `operator<<()` [2/6]

```
S& L4::Factory::S::operator<< (
    l4_umword_t i ) [inline]
```

Put a single `l4_umword_t` as next argument.

Parameters

<code>i</code>	The value to add as next argument.
----------------	------------------------------------

Returns

Reference to this stream.

Definition at line 171 of file [factory](#).

14.72.3.4 `operator<<()` [3/6]

```
S& L4::Factory::S::operator<< (
    char const * s ) [inline]
```

Add a zero-terminated string as next argument.

Parameters

<code>s</code>	The string to add as next argument.
----------------	-------------------------------------

Returns

Reference to this stream.

Definition at line 184 of file [factory](#).

14.72.3.5 `operator<<()` [4/6]

```
S& L4::Factory::S::operator<< (
    Lstr const & s ) [inline]
```

Add a pascal string as next argument.

Parameters

<code>s</code>	The string to add as next argument.
----------------	-------------------------------------

Returns

Reference to this stream.

Definition at line 197 of file [factory](#).

14.72.3.6 operator<<() [5/6]

```
S& L4::Factory::S::operator<< (
    Nil ) [inline]
```

Add an empty argument.

Returns

Reference to this stream.

Definition at line 208 of file [factory](#).

14.72.3.7 operator<<() [6/6]

```
S& L4::Factory::S::operator<< (
    l4_fpage_t d ) [inline]
```

Add a flex page as next argument.

Parameters

<i>d</i>	The flex page to add (there will be no map operation).
----------	--

Returns

Reference to this stream.

Definition at line 221 of file [factory](#).

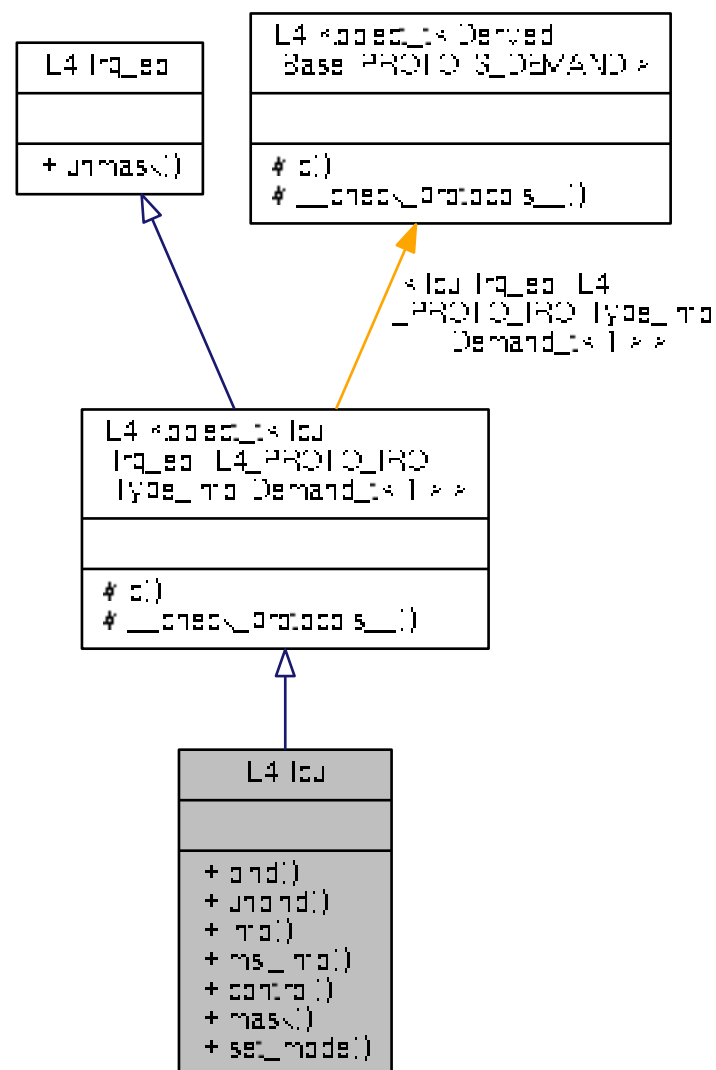
The documentation for this class was generated from the following file:

- [l4/sys/factory](#)

14.73 L4::lcu Class Reference

C++ [lcu](#) interface.

Collaboration diagram for L4::lcu:



Data Structures

- class [Info](#)

This class encapsulates information about an ICU.

Public Member Functions

- [l4_msgtag_t bind](#) (unsigned irqnum, [L4::Cap< Triggerable > irq](#), [l4_utcb_t *utcb=l4_utcb\(\)](#)) throw ()
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\(\)](#)) throw ()
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()) throw ()`
Get information about the capabilities of the ICU.
- `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()) throw ()`
Mask an IRQ line.
- `l4_msgtag_t set_mode (unsigned irqnum, l4_umword_t mode, l4_utcb_t *utcb=l4_utcb()) throw ()`
Set interrupt mode.

Additional Inherited Members

14.73.1 Detailed Description

C++ `l4cu` interface.

To setup an IRQ line the following steps are required:

1. `set_mode()` (optional if IRQ has a default mode)
2. `l4cu::attach()` to attach the IRQ capability to a thread
3. `bind()`
4. `unmask()` to receive the first IRQ

Include File

```
#include <l4/sys/icu>
```

Definition at line 242 of file `irq`.

14.73.2 Member Function Documentation

14.73.2.1 `bind()`

```
l4_msgtag_t L4::Icu::bind (
    unsigned irqnum,
    L4::Cap< Triggerable > irq,
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline]
```

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, usually the UTCB of the calling thread.

Returns

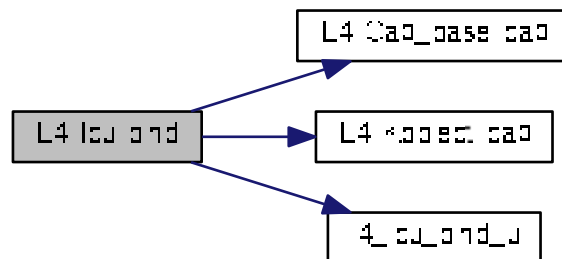
Syscall return tag. The caller should check the return value using [l4_error\(\)](#) to check for errors and to identify the correct method for unmasking the interrupt. Return values < 0 indicate an error. A return value of 0 means a direct unmask via the IRQ object using [L4::irq::unmask](#). A return value of 1 means that the interrupt has to be unmasked via the ICU using [L4::Icu::unmask](#).

Definition at line 290 of file [irq](#).

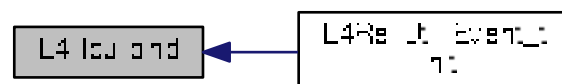
References [L4::Cap_base::cap\(\)](#), [L4::Kobject::cap\(\)](#), [l4_icu_bind_u\(\)](#), [L4_ICU_OP_BIND](#), and [L4_RPC_NF_OP](#).

Referenced by [L4Re::Util::Event_t< PAYLOAD >::init\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



14.73.2.2 info()

```

l4_msgtag_t L4::Icu::info (
    l4_icu_info_t * info,
    l4_utcb_t * utcb = l4_utcb() ) throw () [inline]
  
```

Get information about the capabilities of the ICU.

Parameters

<i>out</i>	<i>info</i>	Info structure to be filled with information.
	<i>utcb</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.

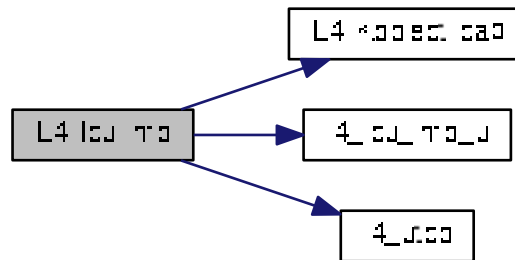
Returns

Syscall return tag

Definition at line 321 of file [irq](#).

References [L4::Kobject::cap\(\)](#), [l4_icu_info_u\(\)](#), [L4_ICU_OP_INFO](#), [L4_ICU_OP_MSI_INFO](#), [L4_INLINE_RPC_OP](#), [L4_RPC_NF_OP](#), and [l4_utcb\(\)](#).

Here is the call graph for this function:



14.73.2.3 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() ) throw ()    [inline]
  
```

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.
<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, usually the UTCB of the calling thread.

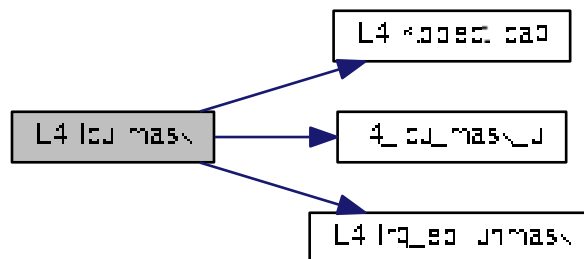
Returns

Syscall return tag

Definition at line 364 of file [irq](#).

References [L4::Kobject::cap\(\)](#), [l4_icu_mask_u\(\)](#), [L4_ICU_OP_MASK](#), [L4_ICU_OP_UNMASK](#), [L4_RPC_NF_OP](#), and [L4::Irq_eoi::unmask\(\)](#).

Here is the call graph for this function:



14.73.2.4 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

14.73.2.5 `set_mode()`

```

14_msgtag_t L4::Icu::set_mode (
    unsigned irqnum,
    14_umword_t mode,
    14_utcb_t * utcb = 14_utcb() ) throw ()    [inline]

```

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, usually the UTCB of the calling thread.

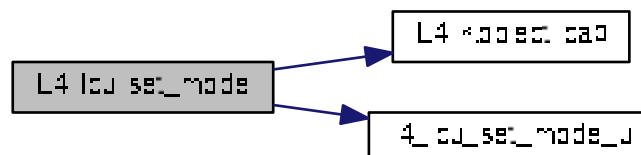
Returns

Syscall return tag

Definition at line 386 of file [irq](#).

References [L4::Kobject::cap\(\)](#), [L4_ICU_OP_SET_MODE](#), [l4_icu_set_mode_u\(\)](#), and [L4_RPC_NF_OP](#).

Here is the call graph for this function:

14.73.2.6 `unbind()`

```

14_msgtag_t L4::Icu::unbind (
    unsigned irqnum,
    L4::Cap< Triggerable > irq,
    14_utcb_t * utcb = 14_utcb() ) throw ()    [inline]

```

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, usually the UTCB of the calling thread.

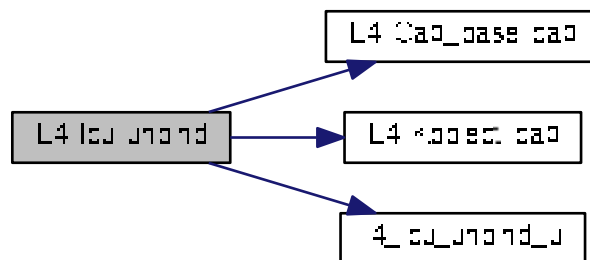
Returns

Syscall return tag

Definition at line 306 of file [irq](#).

References [L4::Cap_base::cap\(\)](#), [L4::Kobject::cap\(\)](#), [L4_ICU_OP_UNBIND](#), [I4_icu_unbind_u\(\)](#), and [L4_RPC_NF_OP](#).

Here is the call graph for this function:



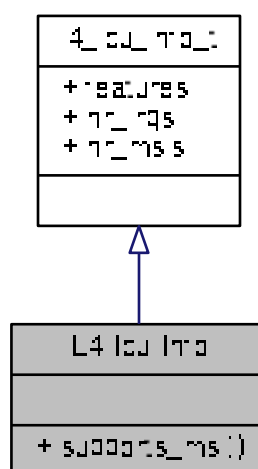
The documentation for this class was generated from the following file:

- [I4/sys/irq](#)

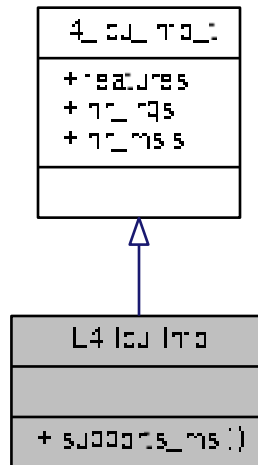
14.74 L4::Icu::Info Class Reference

This class encapsulates information about an ICU.

Inheritance diagram for L4::Icu::Info:



Collaboration diagram for L4::Icu::Info:



Additional Inherited Members

14.74.1 Detailed Description

This class encapsulates information about an ICU.

Definition at line 269 of file [irq](#).

The documentation for this class was generated from the following file:

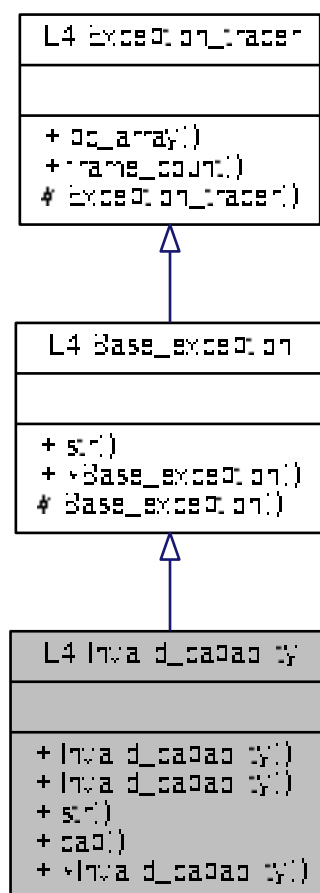
- [l4/sys/irq](#)

14.75 L4::Invalid_capability Class Reference

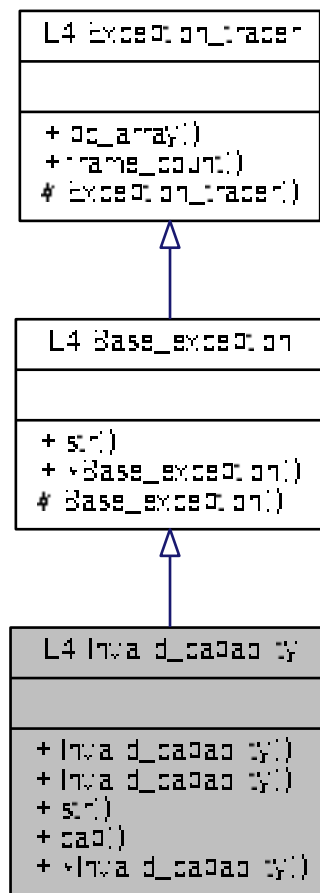
Indicates that an invalid object was invoked.

```
#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) throw ()
Create an *Invalid_obejct* exception for the Object o.
- char const * [str](#) () const throw ()
Return a human readable string for the exception.
- [Cap](#)< void > const & [cap](#) () const throw ()
Get the object that caused the error.

Additional Inherited Members

14.75.1 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 [246](#) of file [exceptions](#).

14.75.2 Constructor & Destructor Documentation

14.75.2.1 Invalid_capability()

```
L4::Invalid_capability::Invalid_capability (
    Cap< void > const & o ) throw ()    [inline], [explicit]
```

Create an Invalid_obejct exception for the Object o.

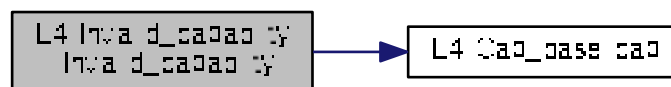
Parameters

<i>o</i>	The object that caused the server side error.
----------	---

Definition at line 256 of file [exceptions](#).

References [L4::Cap_base::cap\(\)](#).

Here is the call graph for this function:



14.75.3 Member Function Documentation

14.75.3.1 cap()

```
Cap<void> const& L4::Invalid_capability::cap ( ) const throw ()    [inline]
```

Get the object that caused the error.

Returns

The object that caused the error on invocation.

Definition at line 265 of file [exceptions](#).

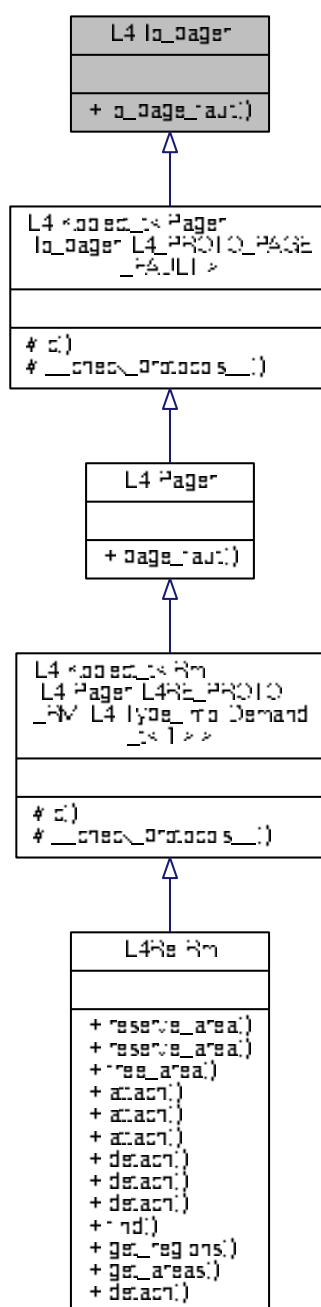
The documentation for this class was generated from the following file:

- [l4/cxx/exceptions](#)

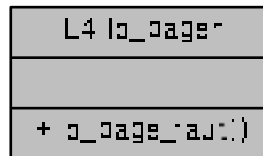
14.76 L4::lo_pager Class Reference

[lo_pager](#) interface.

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::Opt](#)< [l4_mword_t](#) &> result, [L4::lpc::Rcv_fpage](#) rwin, [L4::lpc::Opt](#)< [L4::lpc::Snd_fpage](#) &> fp)
IO page fault protocol message.

14.76.1 Detailed Description

[io_pager](#) interface.

Definition at line 35 of file [pager](#).

14.76.2 Member Function Documentation

14.76.2.1 io_page_fault()

```

l4_msgtag_t L4::Io_pager::io_page_fault (
    l4_fpage_t io_pfa,
    l4_umword_t pc,
    L4::lpc::Opt< l4_mword_t &> result,
    L4::lpc::Rcv_fpage rwin,
    L4::lpc::Opt< L4::lpc::Snd_fpage &> fp )
  
```

IO page fault protocol message.

Parameters

	<i>io_pfa</i>	Flex-page describing the faulting IO-port.
	<i>pc</i>	Faulting program counter.
out	<i>result</i>	Optional: handling result value.
	<i>rwin</i>	The receive window for a flex-page mapping.
out	<i>fp</i>	Optional: flex-page 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 by filling in `result` and / or `fp`.

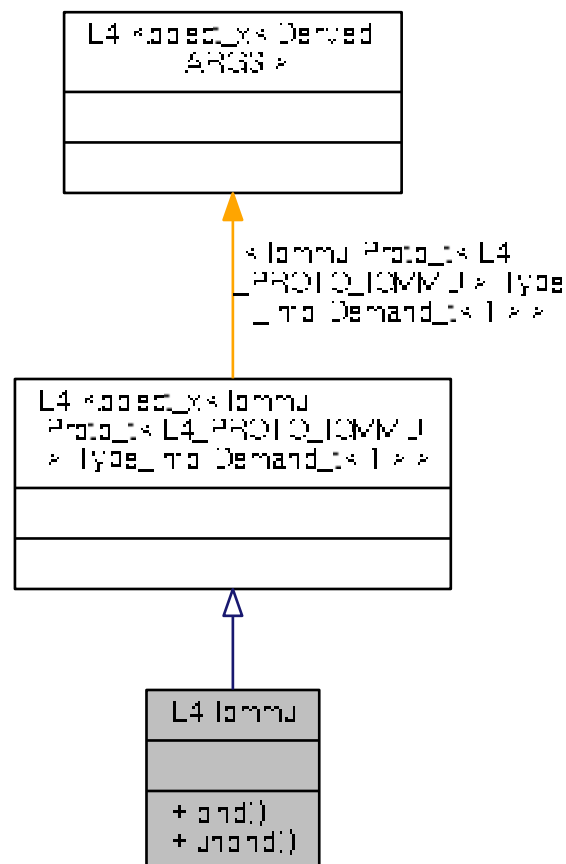
The documentation for this class was generated from the following file:

- `l4/sys/pager`

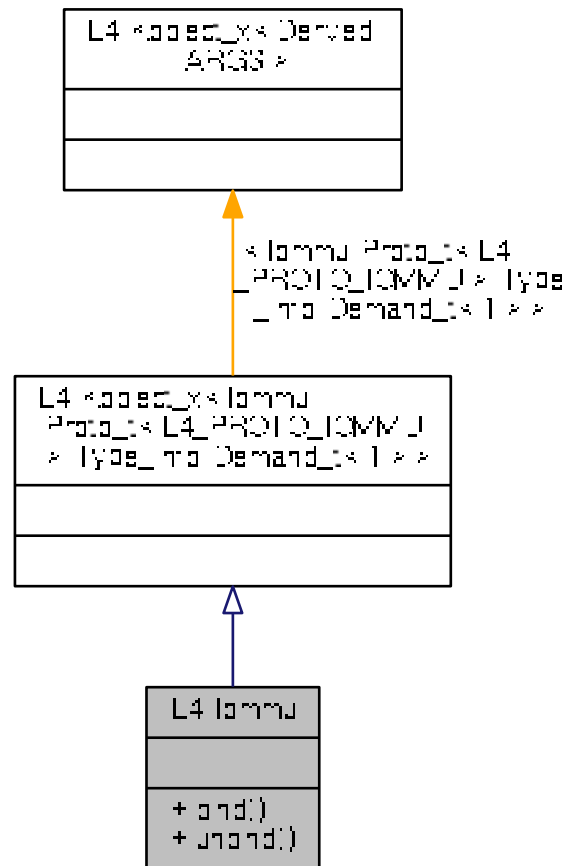
14.77 L4::lommu Class Reference

Interface for IO-MMUs used for DMA remapping.

Inheritance diagram for L4::lommu:



Collaboration diagram for L4::iommu:



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*.

14.77.1 Detailed Description

Interface for IO-MMUs used for DMA remapping.

This interface allows to associate a DMA address space with a platform dependent set of devices.

Definition at line 16 of file [iommu](#).

14.77.2 Member Function Documentation

14.77.2.1 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`.

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).

14.77.2.2 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`.

Parameters

<i>src_id</i>	Platform dependent source ID specifying the set of devices that shall no longer use <code>dma_space</code> for DMA remapping.
<i>dma_space</i>	The DMA space formerly associated with <code>associate()</code> .

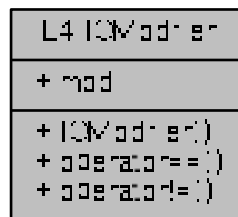
The documentation for this class was generated from the following file:

- `l4/sys/iommu`

14.78 L4::IOModifier Class Reference

Modifier class for the IO stream.

Collaboration diagram for L4::IOModifier:



14.78.1 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 33 of file [basic_ostream](#).

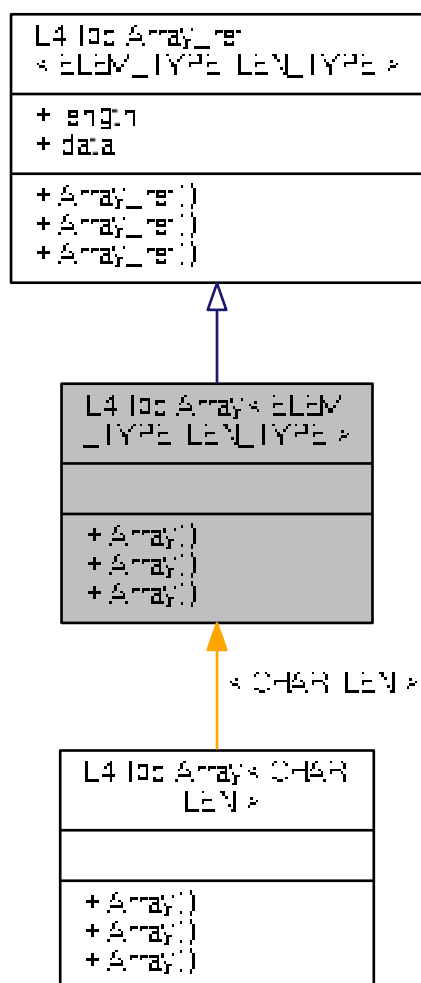
The documentation for this class was generated from the following file:

- [l4/cxx/basic_ostream](#)

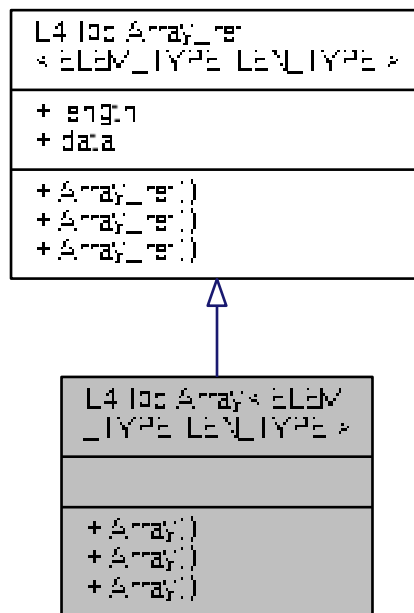
14.79 L4::lpc::Array< ELEM_TYPE, LEN_TYPE > Struct Template Reference

[Array](#) data type for dynamically sized arrays in RPCs.

Inheritance diagram for L4::ipc::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.

14.79.1 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 85 of file [ipc_array](#).

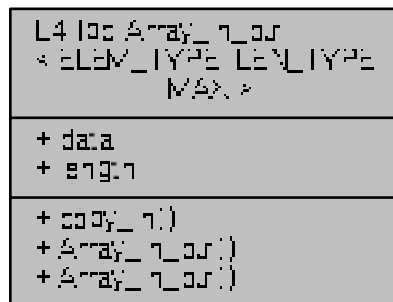
The documentation for this struct was generated from the following file:

- `l4/sys/cxx/ipc_array`

14.80 L4::lpc::Array_in_buf< ELEM_TYPE, LEN_TYPE, MAX > Struct Template Reference

Server-side copy in buffer for [Array](#).

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.

14.80.1 Detailed Description

```
template<typename ELEM_TYPE, typename LEN_TYPE = Array_len_default, LEN_TYPE MAX = (L4_UTCB_GENERIC_DATA_SIZE * sizeof(l4_umword_t)) / sizeof(ELEM_TYPE)>
struct L4::ipc::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 [123](#) of file [ipc_array](#).

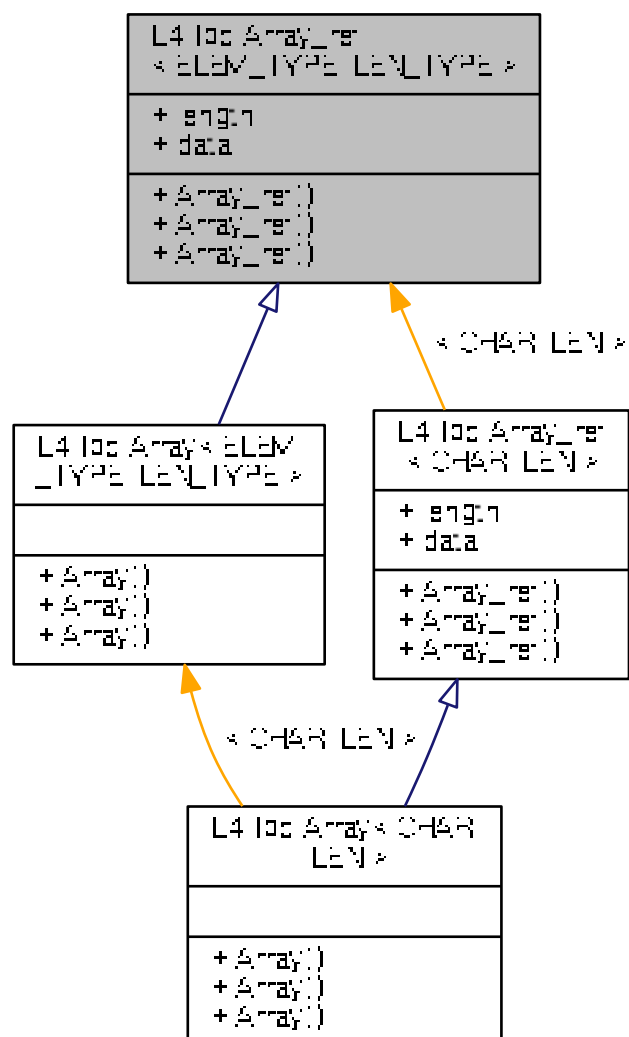
The documentation for this struct was generated from the following file:

- `l4/sys/cxx/ipc_array`

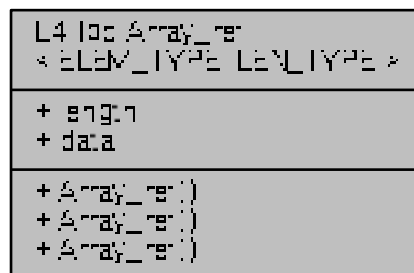
14.81 L4::ipc::Array_ref< ELEM_TYPE, LEN_TYPE > Struct Template Reference

[Array](#) reference data type for arrays located in the message.

Inheritance diagram for L4::lpc::Array_ref< ELEM_TYPE, LEN_TYPE >:



Collaboration diagram for L4::lpc::Array_ref< ELEM_TYPE, LEN_TYPE >:



14.81.1 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 39 of file [ipc_array](#).

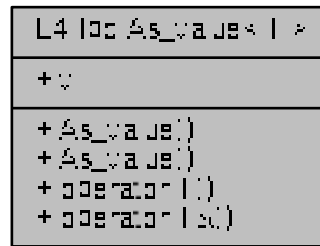
The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_array

14.82 L4::lpc::As_value< T > Struct Template Reference

Pass the argument as plain data value.

Collaboration diagram for L4::ipc::As_value< T >:



14.82.1 Detailed Description

```
template<typename T>
struct L4::ipc::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 flex page. When using `As_value<>` then the argument is transmitted as plain data element.

Definition at line 127 of file [ipc_types](#).

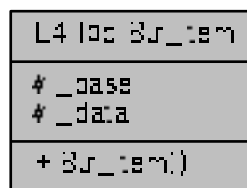
The documentation for this struct was generated from the following file:

- [I4/sys/cxx/ipc_types](#)

14.83 L4::ipc::Buf_item Class Reference

RPC warpper for a receive item.

Collaboration diagram for L4::lpc::Buf_item:



14.83.1 Detailed Description

RPC warpper for a receive item.

Definition at line 305 of file [ipc_types](#).

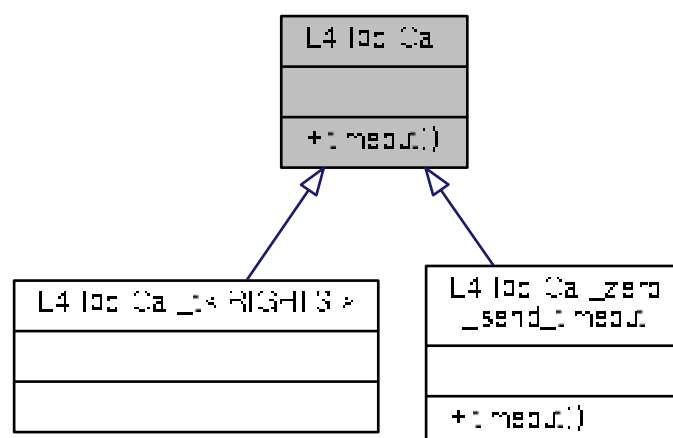
The documentation for this class was generated from the following file:

- [l4/sys/cxx/ipc_types](#)

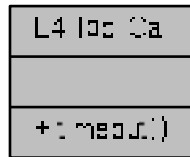
14.84 L4::lpc::Call Struct Reference

RPC attribute for a standard RPC call.

Inheritance diagram for L4::lpc::Call:



Collaboration diagram for L4::ipc::Call:



14.84.1 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(long, send, (unsigned value), L4::Ipc::Call);
```

which is equivalent to:

```
L4_RPC(long, send, (unsigned value));
```

Definition at line 215 of file [ipc_iface](#).

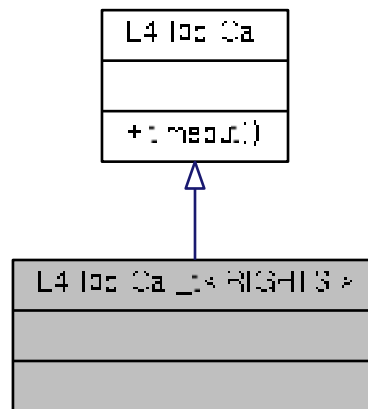
The documentation for this struct was generated from the following file:

- [l4/sys/cxx/ipc_iface](#)

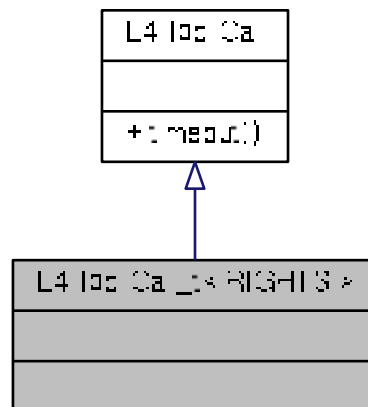
14.85 L4::ipc::Call_t< RIGHTS > Struct Template Reference

RPC attribute for an RPC call with required rights.

Inheritance diagram for L4::ipc::Call_t< RIGHTS >:



Collaboration diagram for L4::ipc::Call_t< RIGHTS >:



14.85.1 Detailed Description

```
template<unsigned RIGHTS>
struct L4::ipc::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(long, func, (unsigned value), L4::Ipc::Call_t<L4_CAP_FPAGE_RW>
);
```

Definition at line 246 of file [ipc_iface](#).

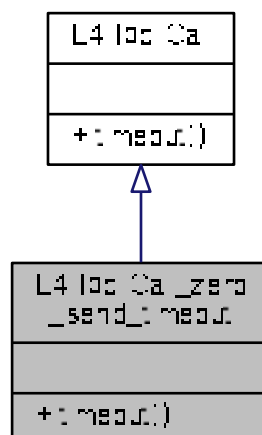
The documentation for this struct was generated from the following file:

- [l4/sys/cxx/ipc_iface](#)

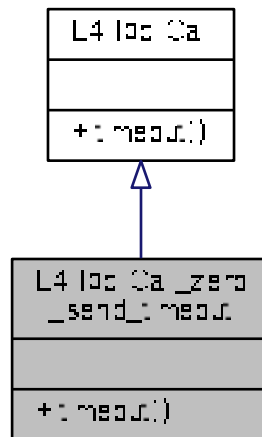
14.86 L4::Ipc::Call_zero_send_timeout Struct Reference

RPC attribute for an RPC call, with zero send timeout.

Inheritance diagram for L4::Ipc::Call_zero_send_timeout:



Collaboration diagram for L4::ipc::Call_zero_send_timeout:



14.86.1 Detailed Description

RPC attribute for an RPC call, with zero send timeout.

Definition at line 225 of file [ipc_iface](#).

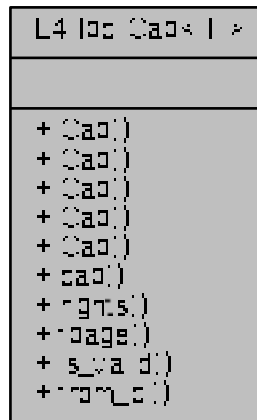
The documentation for this struct was generated from the following file:

- [l4/sys/cxx/ipc_iface](#)

14.87 L4::ipc::Cap< T > Class Template Reference

Capability type for RPC interfaces (see [L4::Cap<T>](#)).

Collaboration diagram for L4::ipc::Cap< T >:



Public Types

- enum { [Rights_mask](#) = 0xff, [Cap_mask](#) = L4_CAP_MASK }

Public Member Functions

- template<typename O >
[Cap](#) ([Cap](#)< O > const &o)
Make copy with conversion.
- [Cap](#) (L4::Cap< T > cap)
Make a [Cap](#) from L4::Cap< T >, with minimal rights.
- template<typename O >
[Cap](#) (L4::Cap< O > cap)
Make IPC [Cap](#) from L4::Cap with conversion (and minimal rights).
- [Cap](#) ()
Make an invalid cap.
- [Cap](#) (L4::Cap< T > cap, unsigned char rights)
Make a [Cap](#) from L4::Cap< T > with the given rights.
- L4::Cap< T > cap () const
Return the L4::Cap< T > of this [Cap](#).
- unsigned rights () const
Return the rights bits stored in this IPC cap.
- L4::ipc::Snd_fpage fpage () const
Return the send flexpage for this [Cap](#) (see [l4_fpage_t](#))
- bool is_valid () const throw ()
Return true if this [Cap](#) is valid.

Static Public Member Functions

- static [Cap from_ci](#) ([l4_cap_idx_t](#) c)
Create an IPC capability from a C capability index plus rights.

14.87.1 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 541 of file [ipc_types](#).

14.87.2 Member Enumeration Documentation

14.87.2.1 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).
Cap_mask	Mask for significant capability bits. (incl. the invalid bit to support invalid caps)

Definition at line 547 of file [ipc_types](#).

14.87.3 Constructor & Destructor Documentation

14.87.3.1 Cap()

```
template<typename T>
L4::Ipc::Cap< T >::Cap (
    L4::Cap< T > cap,
    unsigned char rights ) [inline]
```

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 L4_obj_fpage_ctl .

Definition at line 586 of file [ipc_types](#).

14.87.4 Member Function Documentation

14.87.4.1 from_ci()

```
template<typename T>
static Cap L4::Ipc::Cap< T >::from_ci (
    l4_cap_idx_t c ) [inline], [static]
```

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 594 of file [ipc_types](#).

The documentation for this class was generated from the following file:

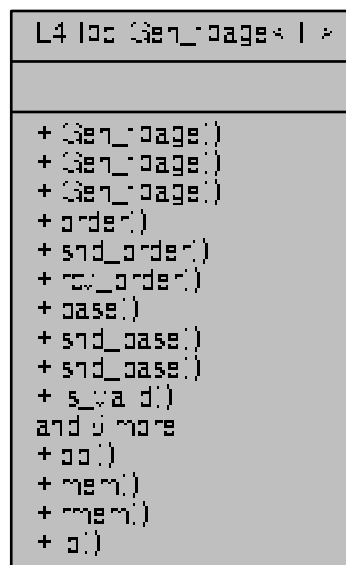
- [l4/sys/cxx/ipc_types](#)

14.88 L4::Ipc::Gen_fpage< T > Class Template Reference

Generic RPC wrapper for [L4](#) flex-pages.

Inherits [T](#).

Collaboration diagram for L4::lpc::Gen_fpage< T >:



Public Types

- enum [Type](#)
Type of mapping object, see L4_fpage_type.
- enum [Map_type](#)
Kind of mapping.
- enum [Cacheopt](#)
Caching options, see l4_fpage_cacheability_opt_t.

Public Member Functions

- bool [is_valid](#) () const
Check if the capability is valid.
- bool [cap_received](#) () const
Check if the capability has been mapped.
- bool [id_received](#) () const
Check if a label was received instead of a mapping.
- bool [local_id_received](#) () const
Check if a local capability id has been received.
- bool [is_compound](#) () const
Check if the received item has the compound bit set.
- [l4_umword_t data](#) () const
Return the raw flex page descriptor.
- [l4_umword_t base_x](#) () const
Return the raw base descriptor.

14.88.1 Detailed Description

```
template<typename T>
class L4::Ipc::Gen_fpage< T >
```

Generic RPC wrapper for [L4](#) flex-pages.

Template Parameters

<i>T</i>	Underlying specific flexpage type.
----------	------------------------------------

Definition at line [321](#) of file [ipc_types](#).

14.88.2 Member Function Documentation

14.88.2.1 cap_received()

```
template<typename T>
bool L4::Ipc::Gen_fpage< T >::cap_received ( ) const [inline]
```

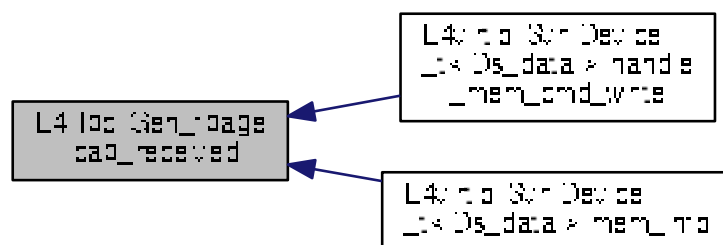
Check if the capability has been mapped.

The capability itself can then be retrieved from the cap slot that has been provided in the receive operation.

Definition at line [438](#) of file [ipc_types](#).

Referenced by [L4virtio::Svr::Device_t< Ds_data >::handle_mem_cmd_write\(\)](#), and [L4virtio::Svr::Device_t< Ds_data >::mem_info\(\)](#).

Here is the caller graph for this function:



14.88.2.2 id_received()

```
template<typename T>
bool L4::Ipc::Gen_fpage< T >::id_received ( ) const [inline]
```

Check if a label was received instead of a mapping.

For IPC gates, if the L4_RCV_ITEM_LOCAL_ID has been set, then only the label of the IPC gate will be provided if the gate is local to the receiver, i.e. the target thread of the IPC gate is in the same task as the receiving thread.

The label can be retrieved with [Gen_fpage::data\(\)](#).

Definition at line 450 of file [ipc_types](#).

Referenced by [L4Re::Util::Dataspace_svr::is_static\(\)](#).

Here is the caller graph for this function:



14.88.2.3 is_compound()

```
template<typename T>
bool L4::Ipc::Gen_fpage< T >::is_compound ( ) const [inline]
```

Check if the received item has the compound bit set.

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 468 of file [ipc_types](#).

14.88.2.4 local_id_received()

```
template<typename T>
bool L4::Ipc::Gen_fpage< T >::local_id_received ( ) const [inline]
```

Check if a local capability id has been received.

If the L4_RCV_ITEM_LOCAL_ID flag has been set by the receiver, and sender and receiver are in the same task, then only the capability index is transferred.

The capability can be retrieved with [Gen_fpage::data\(\)](#).

Definition at line 460 of file [ipc_types](#).

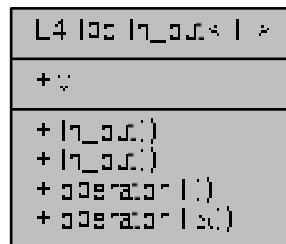
The documentation for this class was generated from the following file:

- [l4/sys/cxx/ipc_types](#)

14.89 L4::ipc::In_out< T > Struct Template Reference

Mark an argument as in-out argument.

Collaboration diagram for L4::ipc::In_out< T >:



14.89.1 Detailed Description

```
template<typename T>
struct L4::ipc::In_out< T >
```

Mark an argument as in-out argument.

Template Parameters

<i>T</i>	The original argument type, usually a pointer or a reference.
----------	---

In_out<> is used when an otherwise output-only value shall also be used as input value.

Definition at line 52 of file [ipc_types](#).

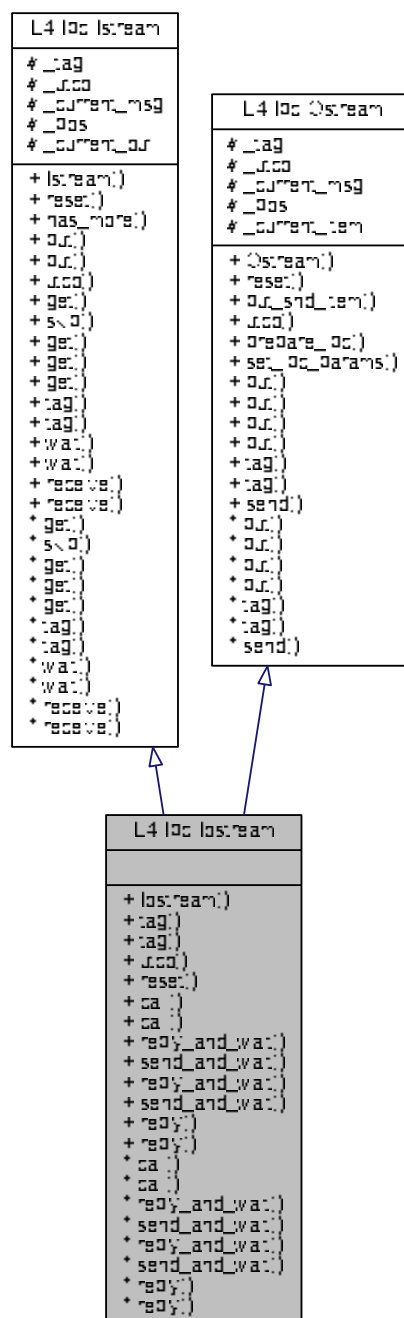
The documentation for this struct was generated from the following file:

- [l4/sys/cxx/ipc_types](#)

14.90 L4::ipc::lostream Class Reference

Input/Output stream for IPC [un]marshalling.

Inheritance diagram for L4::lpc::lostream:



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 receiving to the input stream.
- [l4_msgtag_t call](#) ([l4_cap_idx_t](#) dst, long proto=0)
- [l4_msgtag_t reply_and_wait](#) ([l4_umword_t](#) *src_dst, long proto=0)
Do an IPC reply and wait.
- [l4_msgtag_t send_and_wait](#) ([l4_cap_idx_t](#) dest, [l4_umword_t](#) *src, long proto=0)
- [l4_msgtag_t reply_and_wait](#) ([l4_umword_t](#) *src_dst, [l4_timeout_t](#) timeout, long proto=0)
Do an IPC reply and wait.
- [l4_msgtag_t send_and_wait](#) ([l4_cap_idx_t](#) dest, [l4_umword_t](#) *src, [l4_timeout_t](#) timeout, long proto=0)
- [l4_msgtag_t reply](#) ([l4_timeout_t](#) timeout, long proto=0)
- [l4_msgtag_t reply](#) (long proto=0)

14.90.1 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 791 of file [ipc_stream](#).

14.90.2 Constructor & Destructor Documentation**14.90.2.1 lostream()**

```
L4::Ipc::Iostream::Iostream (
    l4\_utcb\_t * utcb ) [inline], [explicit]
```

Create an IPC IO stream with a single message buffer.

Parameters

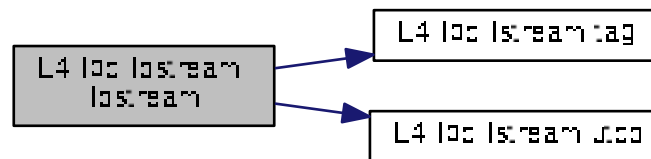
utcb	The message buffer used as backing store.
----------------------	---

The created IO stream uses the same message buffer for sending and receiving IPC messages.

Definition at line 803 of file [ipc_stream](#).

References [L4::lpc::lstream::tag\(\)](#), and [L4::lpc::lstream::utcb\(\)](#).

Here is the call graph for this function:



14.90.3 Member Function Documentation

14.90.3.1 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 receiving to the input stream.

Parameters

<i>dst</i>	The destination L4 UID (thread) 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 dope 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.

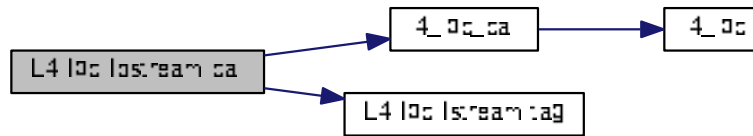
Examples:

[examples/libs/l4re/streammap/client.cc](#).

Definition at line [962](#) of file [ipc_stream](#).

References [l4_ipc_call\(\)](#), [L4_IPC_NEVER](#), and [L4::Ipc::Iostream::tag\(\)](#).

Here is the call graph for this function:



14.90.3.2 reply_and_wait() [1/2]

```
l4_msgtag_t L4::Ipc::Iostream::reply_and_wait (
    l4_umword_t * src_dst,
    long proto = 0 ) [inline]
```

Do an IPC reply and wait.

Parameters

in, out	<i>src_dst</i>	Input: the destination for the send operation. Output: the source of the received message.
	<i>proto</i>	Protocol to use.

Returns

the result code 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 876 of file [ipc_stream](#).

References [L4_IPC_SEND_TIMEOUT_0](#).

14.90.3.3 reply_and_wait() [2/2]

```
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

<code>in, out</code>	<code>src_dst</code>	Input: the destination for the send operation. Output: the source of the received message.
	<code>timeout</code>	Timeout used for IPC.
	<code>proto</code>	Protocol to use.

Returns

the result of the IPC operation.

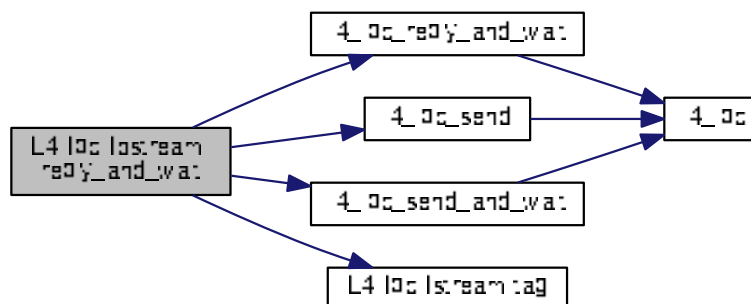
This is a combined IPC operation consisting of a send operation and an open wait for any message.

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 977 of file [ipc_stream](#).

References [L4_INVALID_CAP](#), [l4_ipc_reply_and_wait\(\)](#), [l4_ipc_send\(\)](#), [l4_ipc_send_and_wait\(\)](#), [L4_SYSF_REPLY](#), and [L4::ipc::lstream::tag\(\)](#).

Here is the call graph for this function:



14.90.3.4 reset()

```
void L4::Ipc::Iostream::reset ( ) [inline]
```

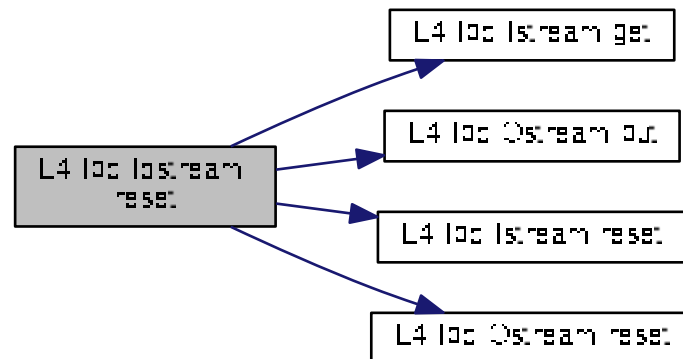
Reset the stream to its initial state.

Input as well as the output stream are reset.

Definition at line 817 of file [ipc_stream](#).

References [L4::ipc::lstream::get\(\)](#), [L4::ipc::Ostream::put\(\)](#), [L4::ipc::lstream::reset\(\)](#), and [L4::ipc::Ostream::reset\(\)](#).

Here is the call graph for this function:



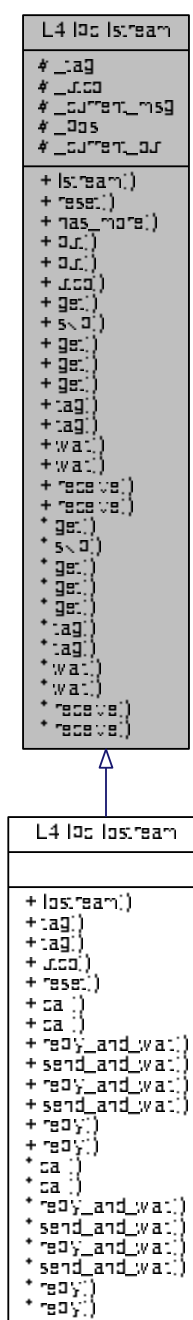
The documentation for this class was generated from the following file:

- [l4/cxx/ipc_stream](#)

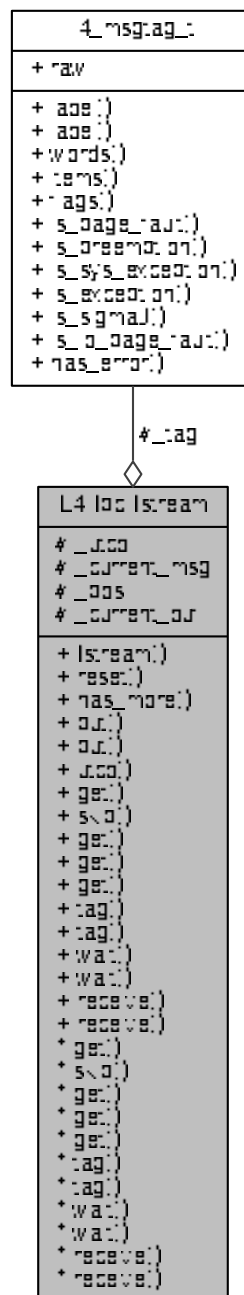
14.91 L4::ipc::Istream Class Reference

Input stream for IPC unmarshalling.

Inheritance diagram for L4::lpc::lstream:



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.
- `bool get (lpc::Varg *va)`
- `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.
- `l4_msgtag_t receive (l4_cap_idx_t src, l4_timeout_t timeout)`

14.91.1 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 345 of file `ipc_stream`.

14.91.2 Constructor & Destructor Documentation

14.91.2.1 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 [359](#) of file [ipc_stream](#).

14.91.3 Member Function Documentation

14.91.3.1 get() [1/3]

```
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 <code>T</code> .
<i>elems</i>	Number of elements of type <code>T</code> to copy out.

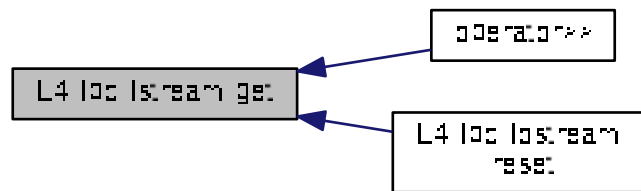
See [Istream::operator>>\(\)](#)

Definition at line [403](#) of file [ipc_stream](#).

References [L4_UNLIKELY](#).

Referenced by [operator>>\(\)](#), and [L4::ipc::Iostream::reset\(\)](#).

Here is the caller graph for this function:



14.91.3.2 `get()` [2/3]

```

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).

In contrast to a normal `get`, this version does actually not copy the data but returns a pointer to the data.

See [Istream::operator>>\(\)](#)

Definition at line 445 of file [ipc_stream](#).

References [L4_UNLIKELY](#).

14.91.3.3 `get()` [3/3]

```

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.
-----	---	--------------

See [Istream::operator>>\(\)](#)

Definition at line 467 of file [ipc_stream](#).

References [L4_UNLIKELY](#), and [L4::lpc::msg_ptr\(\)](#).

Here is the call graph for this function:



14.91.3.4 receive()

```
l4_msgtag_t L4::Ipc::Istream::receive (
    l4_cap_idx_t src ) [inline]
```

Wait for a message from the specified sender.

Parameters

src	The sender id to receive from.
-----	--------------------------------

Returns

The IPC result code ([l4_msgtag_t](#)).

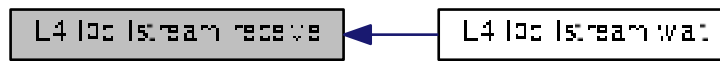
This is commonly known as 'closed wait'.

Definition at line 575 of file [ipc_stream](#).

References [L4_IPC_NEVER](#).

Referenced by [wait\(\)](#).

Here is the caller graph for this function:



14.91.3.5 reset()

```
void L4::Ipc::Istream::reset ( ) [inline]
```

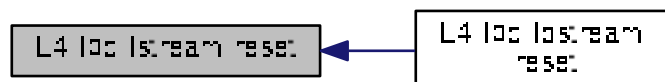
Reset the stream to empty, and ready for [receive\(\)](#)/[wait\(\)](#).

The stream is reset to the same state as on its creation.

Definition at line [369](#) of file [ipc_stream](#).

Referenced by [L4::Ipc::Iostream::reset\(\)](#).

Here is the caller graph for this function:



14.91.3.6 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 422 of file [ipc_stream](#).

References [L4_UNLIKELY](#).

14.91.3.7 tag() [1/2]

```
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.

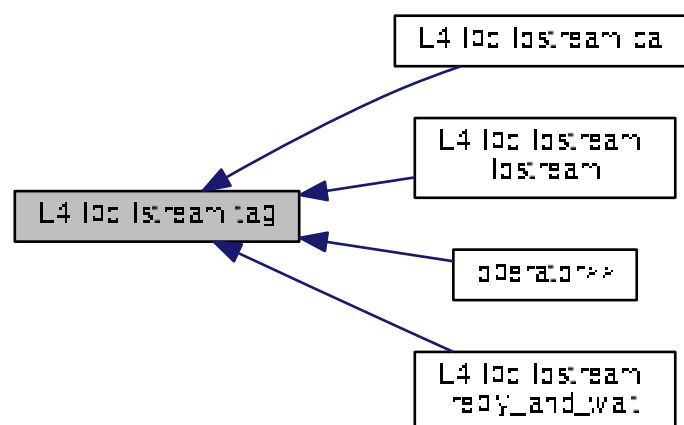
This is in particular useful for handling page faults or exceptions.

See [Istream::operator>>\(\)](#)

Definition at line 508 of file [ipc_stream](#).

Referenced by [L4::ipc::Istream::call\(\)](#), [L4::ipc::Istream::Istream\(\)](#), [operator>>\(\)](#), and [L4::ipc::Istream::reply_and_wait\(\)](#).

Here is the caller graph for this function:



14.91.3.8 tag() [2/2]

```
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 [Istream::operator>>\(\)](#)

Definition at line [520](#) of file [ipc_stream](#).

14.91.3.9 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 [551](#) of file [ipc_stream](#).

References [L4_IPC_NEVER](#).

14.91.3.10 wait() [2/2]

```
l4_msgtag_t L4::Ipc::Istream::wait (
    l4_umword_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

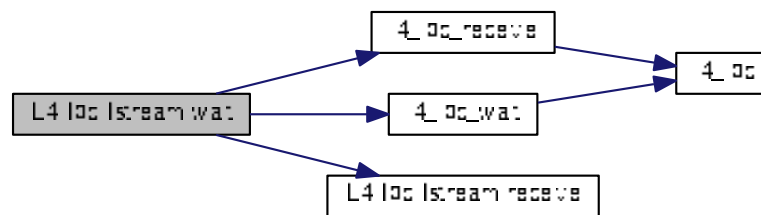
The IPC result dope ([l4_msgtag_t](#)).

This wait is actually known as 'open wait'.

Definition at line 1009 of file [ipc_stream](#).

References [l4_ipc_receive\(\)](#), [l4_ipc_wait\(\)](#), and [receive\(\)](#).

Here is the call graph for this function:



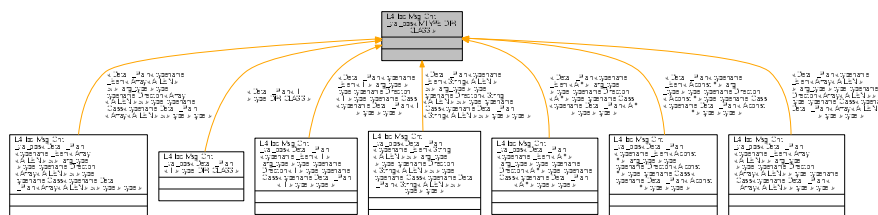
The documentation for this class was generated from the following file:

- [l4/cxx/ipc_stream](#)

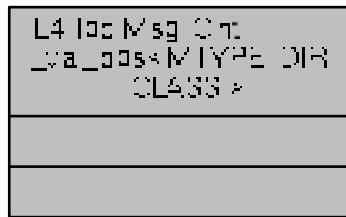
14.92 L4::lpc::Msg::Clnt_val_ops< MTYPE, DIR, CLASS > Struct Template Reference

Defines client-side handling of 'MTYPE' as RPC argument.

Inheritance diagram for L4::lpc::Msg::Clnt_val_ops< MTYPE, DIR, CLASS >:



Collaboration diagram for L4::ipc::Msg::Cnt_val_ops< MTYPE, DIR, CLASS >:



14.92.1 Detailed Description

```
template<typename MTYPE, typename DIR, typename CLASS>
struct L4::ipc::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 [221](#) of file [ipc_basics](#).

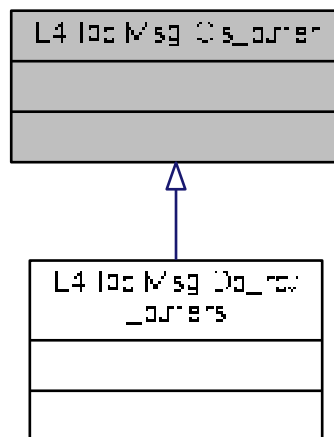
The documentation for this struct was generated from the following file:

- [l4/sys/cxx/ipc_basics](#)

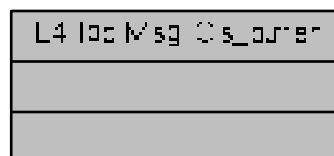
14.93 L4::ipc::Msg::Cls_buffer Struct Reference

Marker type for receive buffer values.

Inheritance diagram for L4::ipc::Msg::Cls_buffer:



Collaboration diagram for L4::ipc::Msg::Cls_buffer:



14.93.1 Detailed Description

Marker type for receive buffer values.

Definition at line 165 of file [ipc_basics](#).

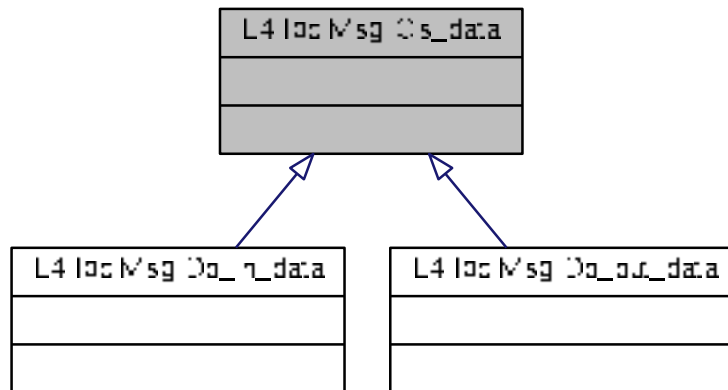
The documentation for this struct was generated from the following file:

- `l4/sys/cxx/ipc_basics`

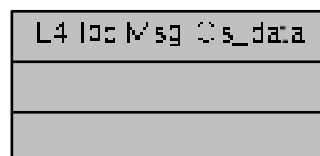
14.94 L4::ipc::Msg::Cls_data Struct Reference

Marker type for data values.

Inheritance diagram for L4::ipc::Msg::Cls_data:



Collaboration diagram for L4::ipc::Msg::Cls_data:



14.94.1 Detailed Description

Marker type for data values.

Definition at line 161 of file [ipc_basics](#).

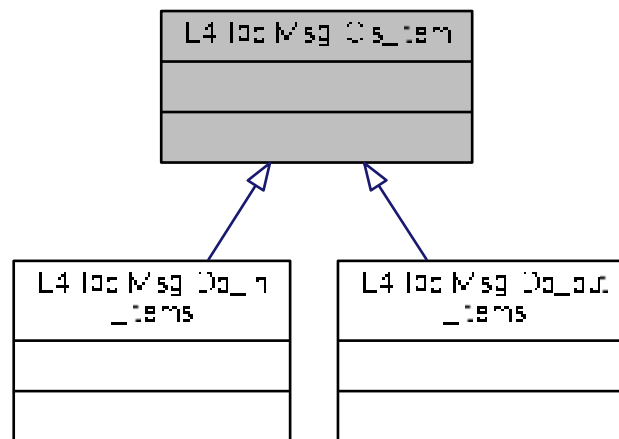
The documentation for this struct was generated from the following file:

- `l4/sys/cxx/ipc_basics`

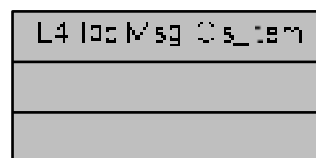
14.95 L4::lpc::Msg::Cls_item Struct Reference

Marker type for item values.

Inheritance diagram for L4::lpc::Msg::Cls_item:



Collaboration diagram for L4::lpc::Msg::Cls_item:



14.95.1 Detailed Description

Marker type for item values.

Definition at line 163 of file [ipc_basics](#).

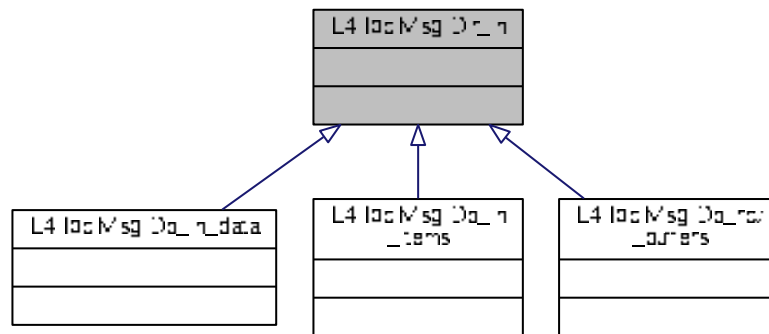
The documentation for this struct was generated from the following file:

- `l4/sys/cxx/ipc_basics`

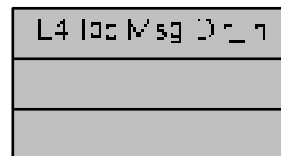
14.96 L4::ipc::Msg::Dir_in Struct Reference

Marker type for input values.

Inheritance diagram for L4::ipc::Msg::Dir_in:



Collaboration diagram for L4::ipc::Msg::Dir_in:



14.96.1 Detailed Description

Marker type for input values.

Definition at line 156 of file [ipc_basics](#).

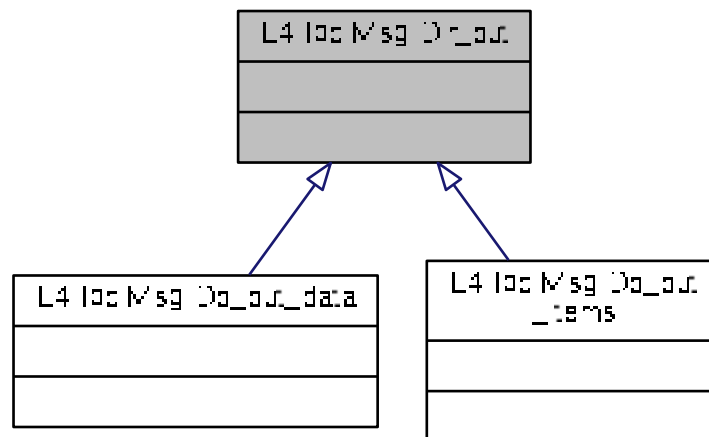
The documentation for this struct was generated from the following file:

- `l4/sys/cxx/ipc_basics`

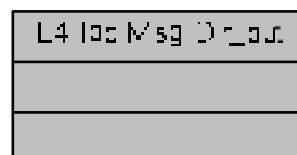
14.97 L4::lpc::Msg::Dir_out Struct Reference

Marker type for output values.

Inheritance diagram for L4::lpc::Msg::Dir_out:



Collaboration diagram for L4::lpc::Msg::Dir_out:



14.97.1 Detailed Description

Marker type for output values.

Definition at line 158 of file [ipc_basics](#).

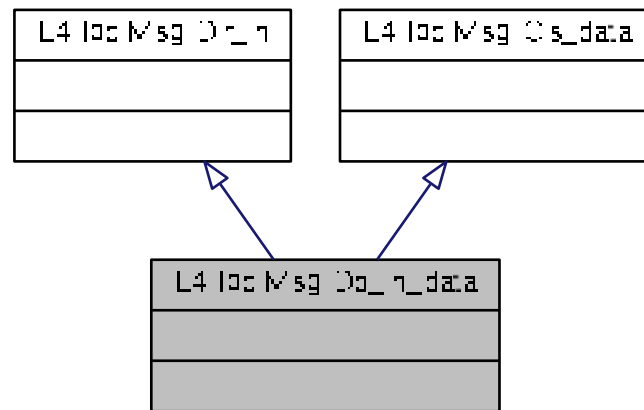
The documentation for this struct was generated from the following file:

- `l4/sys/cxx/ipc_basics`

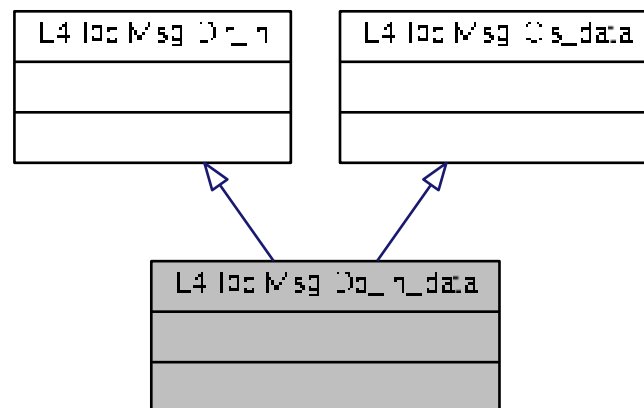
14.98 L4::ipc::Msg::Do_in_data Struct Reference

Marker for Input data.

Inheritance diagram for L4::ipc::Msg::Do_in_data:



Collaboration diagram for L4::ipc::Msg::Do_in_data:



14.98.1 Detailed Description

Marker for Input data.

Definition at line 169 of file [ipc_basics](#).

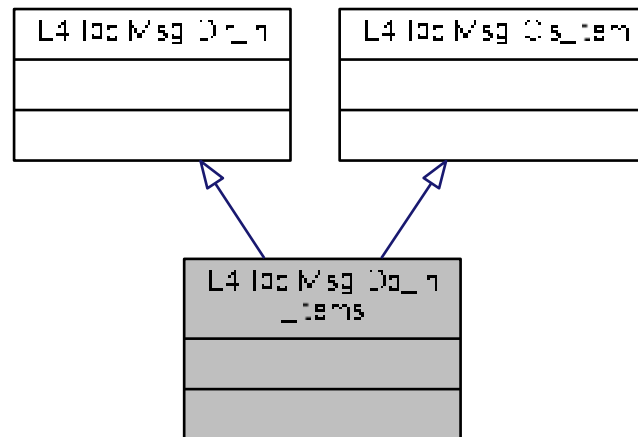
The documentation for this struct was generated from the following file:

- `I4/sys/cxx/ipc_basics`

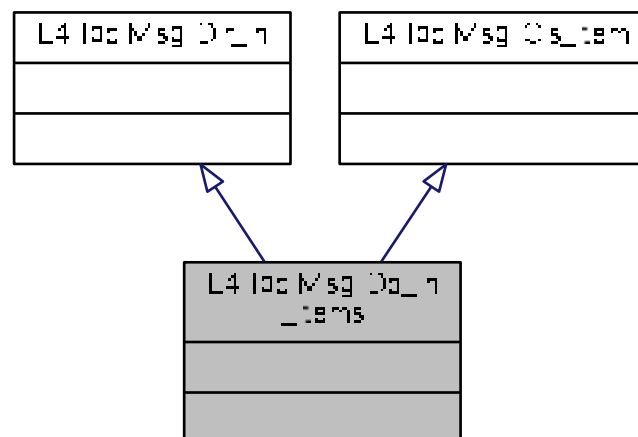
14.99 L4::lpc::Msg::Do_in_items Struct Reference

Marker for Input items.

Inheritance diagram for L4::lpc::Msg::Do_in_items:



Collaboration diagram for L4::lpc::Msg::Do_in_items:



14.99.1 Detailed Description

Marker for Input items.

Definition at line 173 of file [ipc_basics](#).

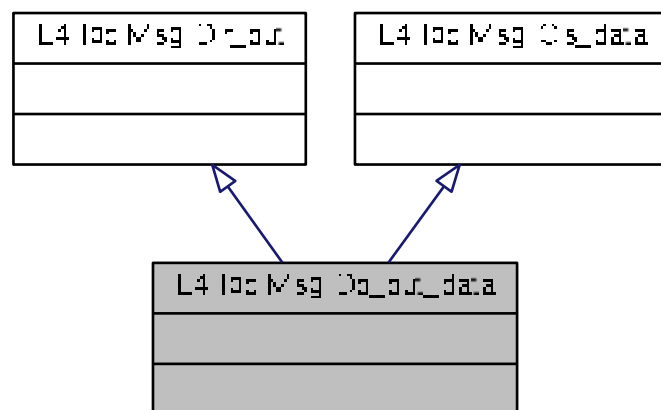
The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_basics

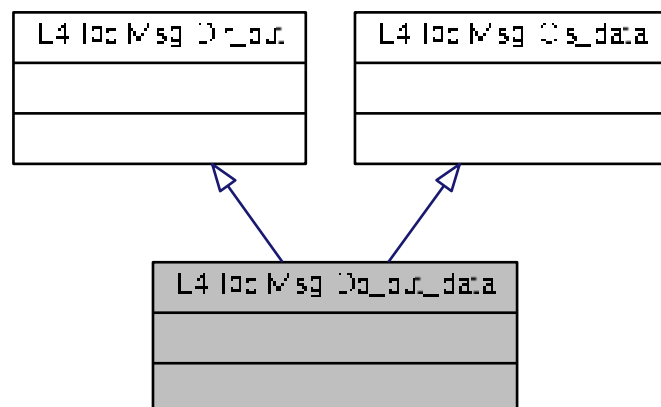
14.100 L4::ipc::Msg::Do_out_data Struct Reference

Marker for Output data.

Inheritance diagram for L4::ipc::Msg::Do_out_data:



Collaboration diagram for L4::ipc::Msg::Do_out_data:



14.100.1 Detailed Description

Marker for Output data.

Definition at line 171 of file [ipc_basics](#).

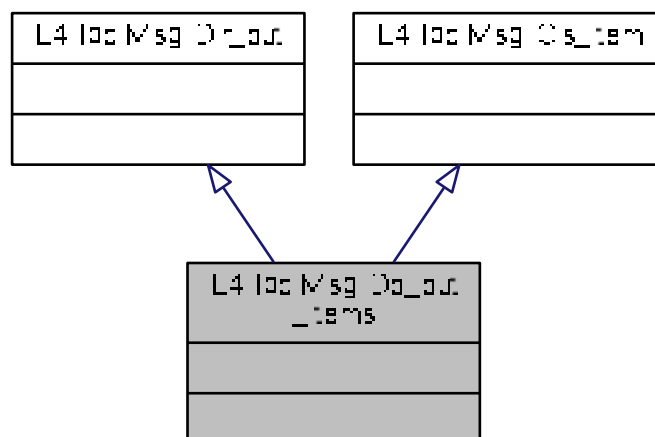
The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_basics

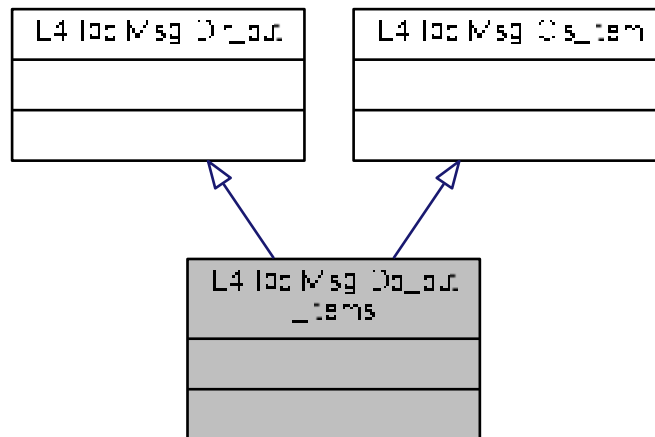
14.101 L4::lpc::Msg::Do_out_items Struct Reference

Marker for Output items.

Inheritance diagram for L4::lpc::Msg::Do_out_items:



Collaboration diagram for L4::ipc::Msg::Do_out_items:



14.101.1 Detailed Description

Marker for Output items.

Definition at line [175](#) of file [ipc_basics](#).

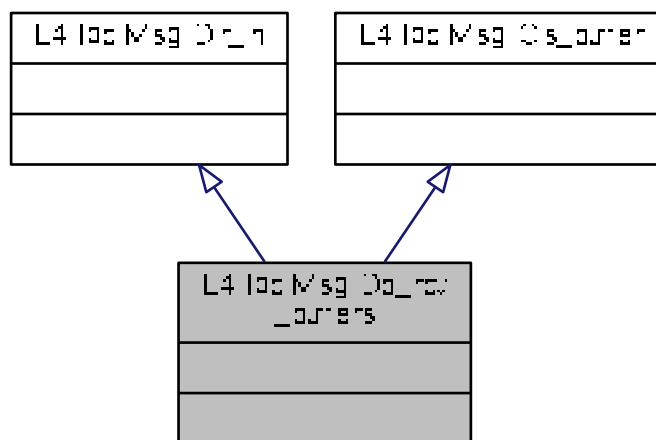
The documentation for this struct was generated from the following file:

- [l4/sys/cxx/ipc_basics](#)

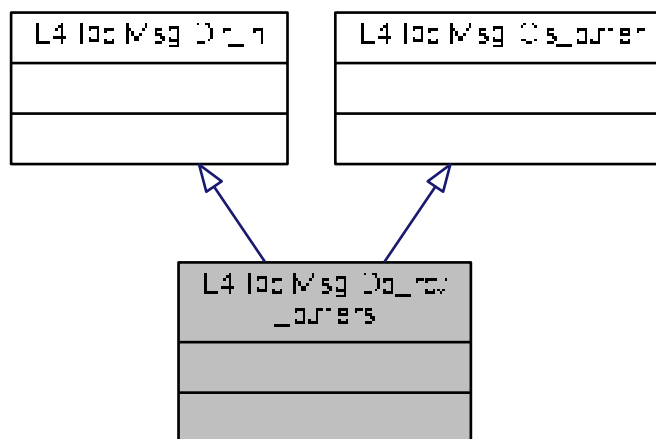
14.102 L4::ipc::Msg::Do_rcv_buffers Struct Reference

Marker for receive buffers.

Inheritance diagram for L4::ipc::Msg::Do_rcv_buffers:



Collaboration diagram for L4::ipc::Msg::Do_rcv_buffers:



14.102.1 Detailed Description

Marker for receive buffers.

Definition at line 177 of file [ipc_basics](#).

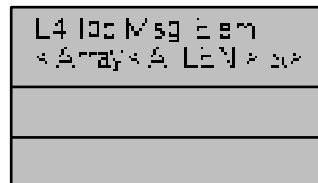
The documentation for this struct was generated from the following file:

- `l4/sys/cxx/ipc_basics`

14.103 L4::ipc::Msg::Elem< Array< A, LEN > &> Struct Template Reference

[Array](#) as output argument.

Collaboration diagram for L4::ipc::Msg::Elem< Array< A, LEN > &>:



Public Types

- typedef [Array](#)< A, LEN > & [arg_type](#)
Array<> & at the interface.
- typedef [Array_ref](#)< A, LEN > [svr_type](#)
Array_ref<> as server storage type.
- typedef [svr_type](#) & [svr_arg_type](#)
Array_ref<> & at the server side.

14.103.1 Detailed Description

```
template<typename A, typename LEN>
struct L4::ipc::Msg::Elem< Array< A, LEN > &>
```

[Array](#) as output argument.

Definition at line 167 of file [ipc_array](#).

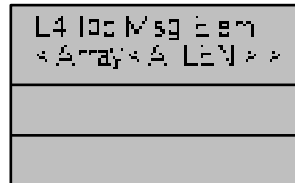
The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_array

14.104 L4::lpc::Msg::Elem< Array< A, LEN > > Struct Template Reference

[Array](#) as input arguments.

Collaboration diagram for L4::lpc::Msg::Elem< Array< A, LEN > >:



Public Types

- typedef [Array](#)< A, LEN > [arg_type](#)
Array<> as argument at the interface.
- typedef [Array_ref](#)< A, LEN > [svr_type](#)
Array_ref<> at the server side.

14.104.1 Detailed Description

```
template<typename A, typename LEN>
struct L4::lpc::Msg::Elem< Array< A, LEN > >
```

[Array](#) as input arguments.

Definition at line [155](#) of file [ipc_array](#).

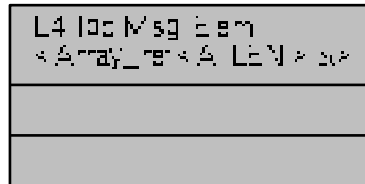
The documentation for this struct was generated from the following file:

- I4/sys/cxx/ipc_array

14.105 L4::ipc::Msg::Elem< Array_ref< A, LEN > &> Struct Template Reference

[Array_ref](#) as output argument.

Collaboration diagram for L4::ipc::Msg::Elem< Array_ref< A, LEN > &>:



Public Types

- typedef [Array_ref](#)< A, LEN > & [arg_type](#)
Array_ref<> at the interface.
- typedef [Array_ref](#)< typename L4::Types::Remove_const< A >::type, LEN > [svr_type](#)
Array_ref<> as server storage.
- typedef [svr_type](#) & [svr_arg_type](#)
Array_ref<> & as server argument.

14.105.1 Detailed Description

```
template<typename A, typename LEN>
struct L4::ipc::Msg::Elem< Array_ref< A, LEN > &>
```

[Array_ref](#) as output argument.

Definition at line 180 of file [ipc_array](#).

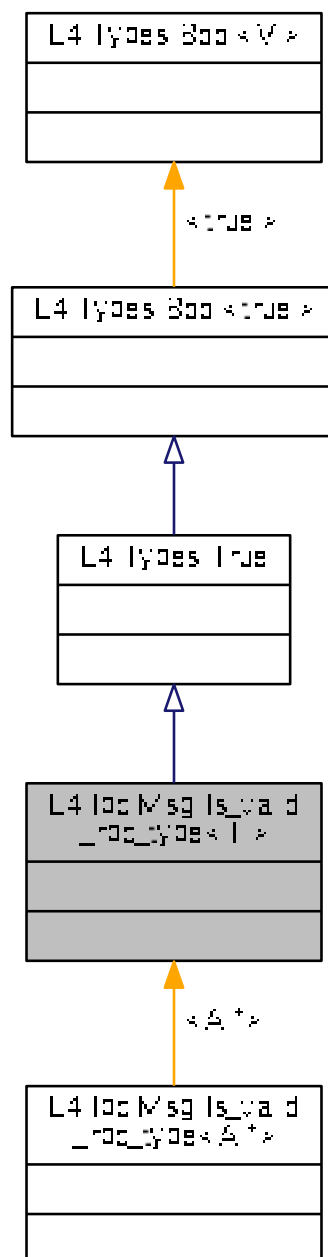
The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_array

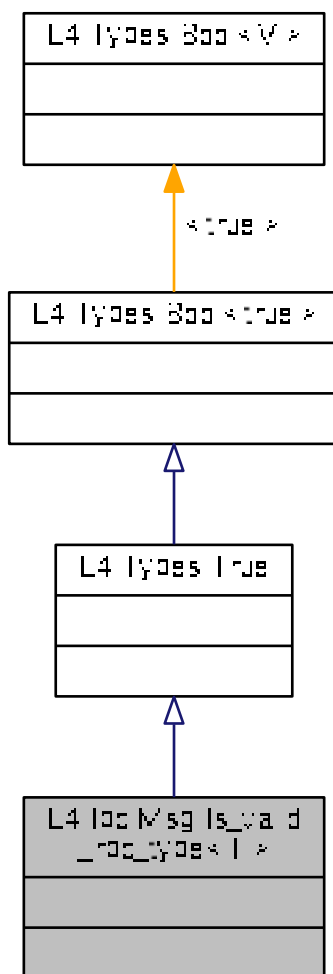
14.106 L4::lpc::Msg::ls_valid_rpc_type< T > Struct Template Reference

Type trait defining a valid RPC parameter type.

Inheritance diagram for L4::lpc::Msg::ls_valid_rpc_type< T >:



Collaboration diagram for L4::ipc::Msg::Is_valid_rpc_type< T >:



Additional Inherited Members

14.106.1 Detailed Description

```
template<typename T>
struct L4::ipc::Msg::Is_valid_rpc_type< T >
```

Type trait defining a valid RPC parameter type.

Definition at line 350 of file [ipc_basics](#).

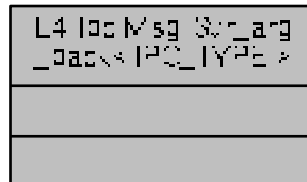
The documentation for this struct was generated from the following file:

- `I4/sys/cxx/ipc_basics`

14.107 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 >:



14.107.1 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 155 of file [ipc_server](#).

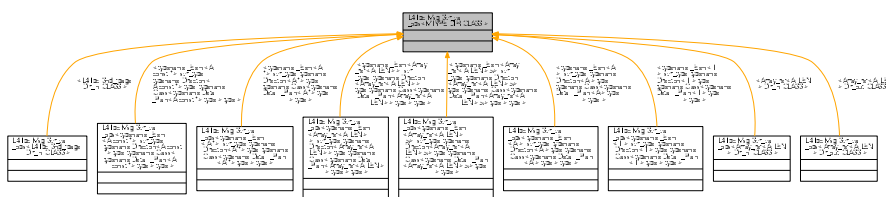
The documentation for this struct was generated from the following file:

- `l4/sys/cxx/ipc_server`

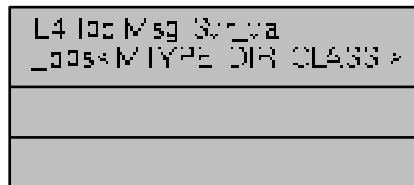
14.108 L4::lpc::Msg::Svr_val_ops< MTYPE, DIR, CLASS > Struct Template Reference

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 >`:



14.108.1 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>	<code>Elem<T>::svr_type</code> (where <code>T</code> is the type used in the RPC definition)
<i>DIR</i>	<code>Dir_in</code> (client -> server), or <code>Dir_out</code> (server -> client)
<i>CLASS</i>	<code>Cls_data</code> , <code>Cls_item</code> , or <code>Cls_buffer</code>

Definition at line 275 of file `ipc_basics`.

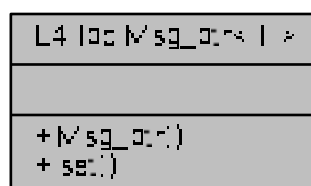
The documentation for this struct was generated from the following file:

- `l4/sys/cxx/ipc_basics`

14.109 L4::lpc::Msg_ptr< T > Class Template Reference

Pointer to an element of type `T` in an `lpc::lstream`.

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.

14.109.1 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 241 of file [ipc_stream](#).

14.109.2 Constructor & Destructor Documentation

14.109.2.1 Msg_ptr()

```
template<typename T>
L4::lpc::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 252 of file [ipc_stream](#).

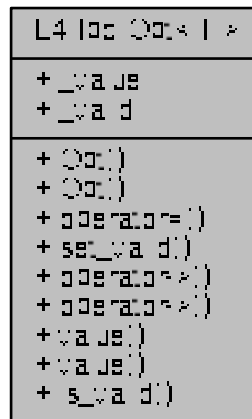
The documentation for this class was generated from the following file:

- [l4/cxx/ipc_stream](#)

14.110 L4::lpc::Opt< T > Struct Template Reference

Attribute for defining an optional RPC argument.

Collaboration diagram for L4::ipc::Opt< T >:



Public Member Functions

- [Opt \(\)](#)
Make an absent optional argument.
- [Opt \(T value\)](#)
Make a present optional argument with the given value.
- [Opt & operator= \(T value\)](#)
Assign a value to the optional argument (makes the argument present)
- void [set_valid](#) (bool valid=true)
Set the argument to present or absent.
- T * [operator-> \(\)](#)
Get the pointer to the value.
- T const * [operator-> \(\) const](#)
Get the const pointer to the value.
- T [value \(\) const](#)
Get the value.
- T & [value \(\)](#)
Get the value.
- bool [is_valid \(\) const](#)
Get true if present, false if not.

Data Fields

- T [_value](#)
The value.
- bool [_valid](#)
True if the optional argument is present, false else.

14.110.1 Detailed Description

```
template<typename T>  
struct L4::lpc::Opt< T >
```

Attribute for defining an optional RPC argument.

Definition at line [147](#) of file [ipc_types](#).

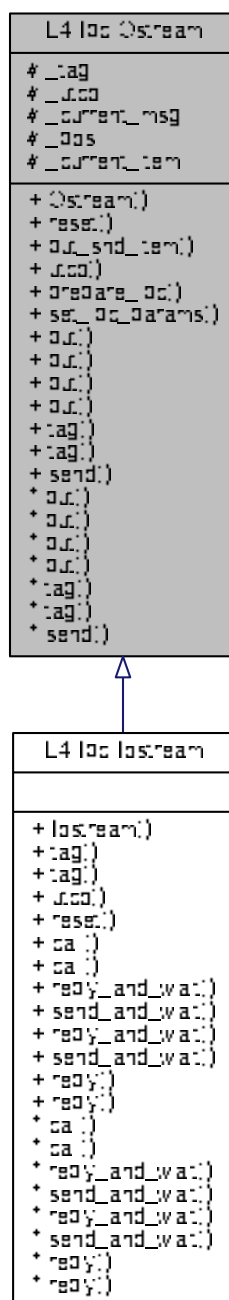
The documentation for this struct was generated from the following file:

- [l4/sys/cxx/ipc_types](#)

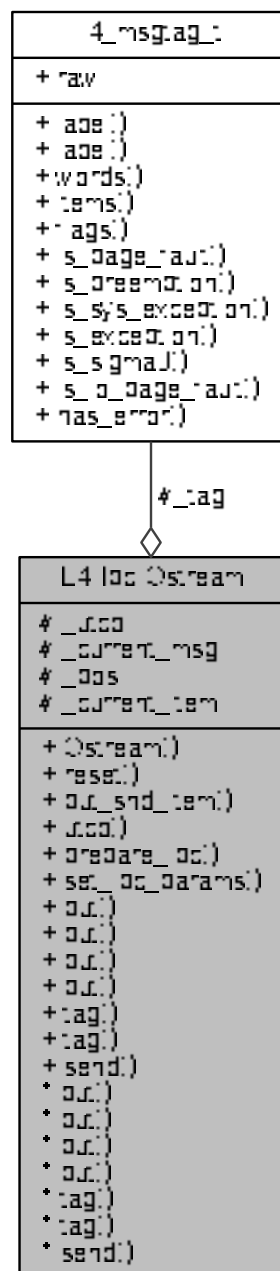
14.111 L4::lpc::Ostream Class Reference

Output stream for IPC marshalling.

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.
- `int put (Varg const &va)`
- `template<typename T >`
`int put (Varg_t< T > const &va)`
- `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.

14.111.1 Detailed Description

Output stream for IPC marshalling.

`lpc::Ostream` is part of the dynamic IPC marshalling infrastructure, as well as `lpc::Istream` and `lpc::Iostream`.

`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`.

14.111.2 Member Function Documentation

14.111.2.1 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.

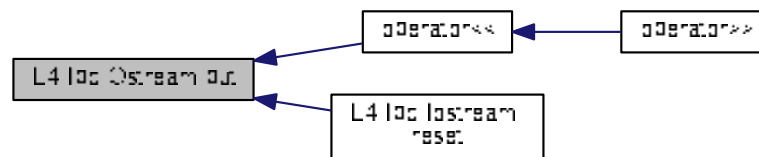
Parameters

<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<<\(\)](#), and [L4::Ipc::Iostream::reset\(\)](#).

Here is the caller graph for this function:

14.111.2.2 `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](#).

14.111.2.3 `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

Syscall return tag.

Definition at line 955 of file [ipc_stream](#).

References [L4_IPC_NEVER](#), [l4_ipc_send\(\)](#), and [L4_MSGTAG_FLAGS](#).

Here is the call graph for this function:

14.111.2.4 `tag()` [1/2]

```
l4_msgtag_t L4::Ipc::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<<\(\)](#).

Here is the caller graph for this function:



14.111.2.5 tag() [2/2]

```
l4_msgtag_t& L4::Ipc::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](#).

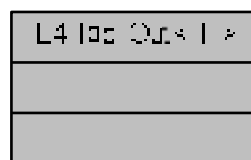
The documentation for this class was generated from the following file:

- [l4/cxx/ipc_stream](#)

14.112 L4::lpc::Out< T > Struct Template Reference

Mark an argument as a output value in an RPC signature.

Collaboration diagram for L4::lpc::Out< T >:



14.112.1 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 out-put 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 42 of file [ipc_types](#).

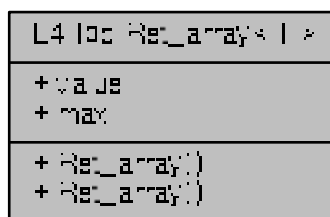
The documentation for this struct was generated from the following file:

- [l4/sys/cxx/ipc_types](#)

14.113 L4::ipc::Ret_array< T > Struct Template Reference

Dynamically sized output array of type T.

Collaboration diagram for L4::ipc::Ret_array< T >:



14.113.1 Detailed Description

```
template<typename T>
struct L4::ipc::Ret_array< T >
```

Dynamically sized output array of type T.

Template Parameters

<code>T</code>	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 34 of file [ipc_ret_array](#).

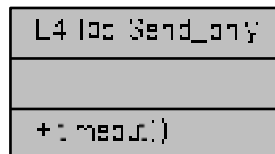
The documentation for this struct was generated from the following file:

- [l4/sys/cxx/ipc_ret_array](#)

14.114 L4::lpc::Send_only Struct Reference

RPC attribute for a send-only RPC.

Collaboration diagram for L4::lpc::Send_only:



14.114.1 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(long, send, (unsigned value), L4::lpc::Send_only);
```

Definition at line 263 of file [ipc_iface](#).

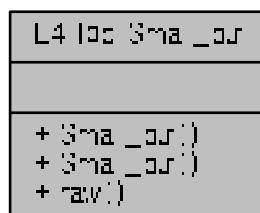
The documentation for this struct was generated from the following file:

- [l4/sys/cxx/ipc_iface](#)

14.115 L4::lpc::Small_buf Class Reference

A receive item for receiving a single capability.

Collaboration diagram for L4::lpc::Small_buf:



Public Member Functions

- [Small_buf](#) ([L4::Cap](#)< void > cap, unsigned long flags=0)
Create a receive item from a C++ cap.
- [Small_buf](#) ([l4_cap_idx_t](#) cap, unsigned long flags=0)
Create a receive item from a C cap.

14.115.1 Detailed Description

A receive item for receiving a single capability.

This class is the main abstraction for receiving capabilities via [lpc::lstream](#). To receive a 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 268 of file [ipc_types](#).

14.115.2 Constructor & Destructor Documentation

14.115.2.1 [Small_buf\(\)](#) [1/2]

```
L4::Ipc::Small_buf::Small_buf (
    L4::Cap< void > cap,
    unsigned long flags = 0 ) [inline], [explicit]
```

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 278 of file [ipc_types](#).

14.115.2.2 [Small_buf\(\)](#) [2/2]

```
L4::Ipc::Small_buf::Small_buf (
    l4\_cap\_idx\_t cap,
    unsigned long flags = 0 ) [inline], [explicit]
```

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 285 of file [ipc_types](#).

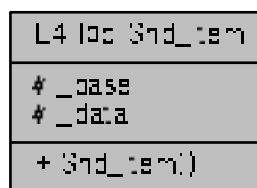
The documentation for this class was generated from the following file:

- [l4/sys/cxx/ipc_types](#)

14.116 L4::lpc::Snd_item Class Reference

RPC wrapper for a send item.

Collaboration diagram for L4::lpc::Snd_item:



14.116.1 Detailed Description

RPC wrapper for a send item.

Definition at line 294 of file [ipc_types](#).

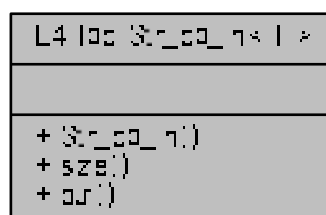
The documentation for this class was generated from the following file:

- [l4/sys/cxx/ipc_types](#)

14.117 L4::lpc::Str_cp_in< T > Class Template Reference

Abstraction for extracting a zero-terminated string from an [lpc::lstream](#).

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](#).

14.117.1 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 191 of file [ipc_stream](#).

14.117.2 Constructor & Destructor Documentation

14.117.2.1 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::lstream](#).

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 202 of file [ipc_stream](#).

The documentation for this class was generated from the following file:

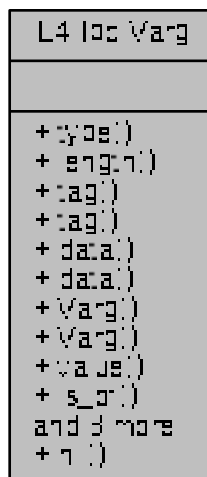
- [l4/cxx/ipc_stream](#)

14.118 L4::lpc::Varg Class Reference

Variably sized RPC argument.

Inherited by L4::lpc::Varg_t< T >.

Collaboration diagram for L4::lpc::Varg:



Public Types

- typedef [l4_umword_t](#) Tag
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, typename L4::Types::Enable_if< sizeof(T)<=sizeof(char const *) , bool >::type=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, typename L4::Types::Enable_if< sizeof(T)<=sizeof(char const *) , bool >::type=true)`
Make Varg from direct value.

14.118.1 Detailed Description

Variably sized RPC argument.

Definition at line 96 of file [ipc_varg](#).

14.118.2 Member Function Documentation

14.118.2.1 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](#).

14.118.2.2 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 184 of file [ipc_varg](#).

14.118.2.3 is_nil()

```
bool L4::Ipc::Varg::is_nil ( ) const [inline]
```

Returns

true if the [Varg](#) is of nil type.

Definition at line 171 of file [ipc_varg](#).

14.118.2.4 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 168 of file [ipc_varg](#).

14.118.2.5 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 174 of file [ipc_varg](#).

14.118.2.6 length()

```
unsigned L4::Ipc::Varg::length ( ) const [inline]
```

Get the size of the RPC argument.

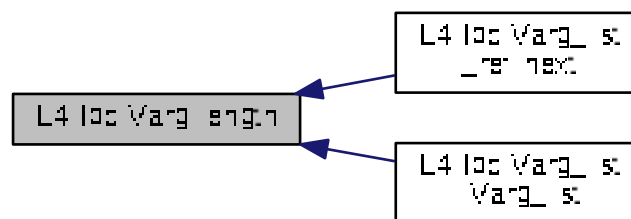
Returns

The size of the RPC argument

Definition at line 114 of file [ipc_varg](#).

Referenced by [L4::Ipc::Varg_list_ref::next\(\)](#), and [L4::Ipc::Varg_list< MAX >::Varg_list\(\)](#).

Here is the caller graph for this function:



14.118.2.7 tag()

```
Tag L4::Ipc::Varg::tag ( ) const [inline]
```

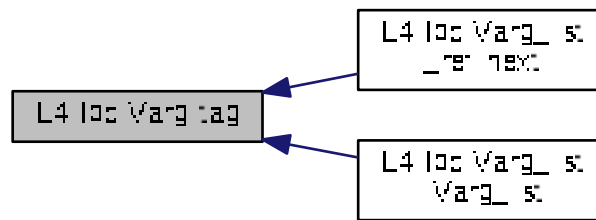
Returns

the tag value (the `Direct_data` bit masked)

Definition at line 116 of file [ipc_varg](#).

Referenced by [L4::Ipc::Varg_list_ref::next\(\)](#), and [L4::Ipc::Varg_list< MAX >::Varg_list\(\)](#).

Here is the caller graph for this function:



14.118.2.8 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](#).

14.118.2.9 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 154 of file [ipc_varg](#).

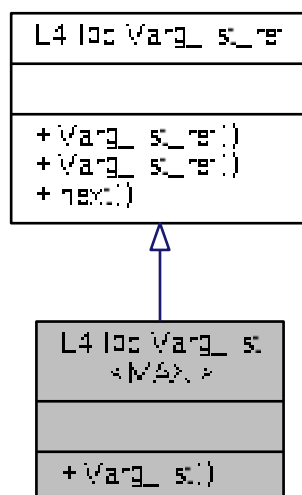
The documentation for this class was generated from the following file:

- [l4/sys/cxx/ipc_varg](#)

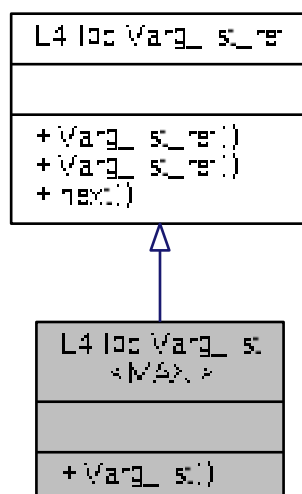
14.119 L4::ipc::Varg_list< MAX > Class Template Reference

Self-contained list of variable-sized RPC parameters.

Inheritance diagram for L4::ipc::Varg_list< MAX >:



Collaboration diagram for L4::ipc::Varg_list< MAX >:



Public Member Functions

- [Varg_list](#) ([Varg_list_ref](#) const &r)

Create a parameter list as a copy from a referencing list.

14.119.1 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 239 of file [ipc_varg](#).

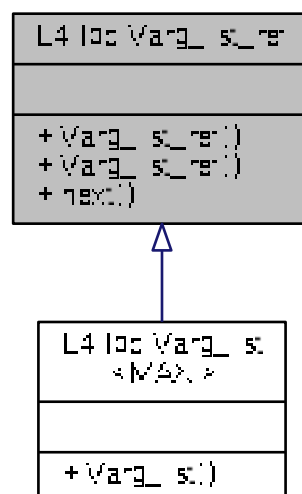
The documentation for this class was generated from the following file:

- [l4/sys/cxx/ipc_varg](#)

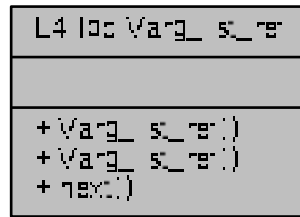
14.120 L4::lpc::Varg_list_ref Class Reference

List of variable-sized RPC parameters as received by the server.

Inheritance diagram for L4::lpc::Varg_list_ref:



Collaboration diagram for L4::Ipc::Varg_list_ref:



Public Member Functions

- [Varg_list_ref](#) ()
Create an empty parameter list.
- [Varg_list_ref](#) (void const *start, void const *end)
Create a parameter list over a given memory region.
- [Varg_next](#) ()
Get the next parameter in the list.

14.120.1 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 252 of file [ipc_varg](#).

14.120.2 Constructor & Destructor Documentation

14.120.2.1 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 269 of file ipc_varg.

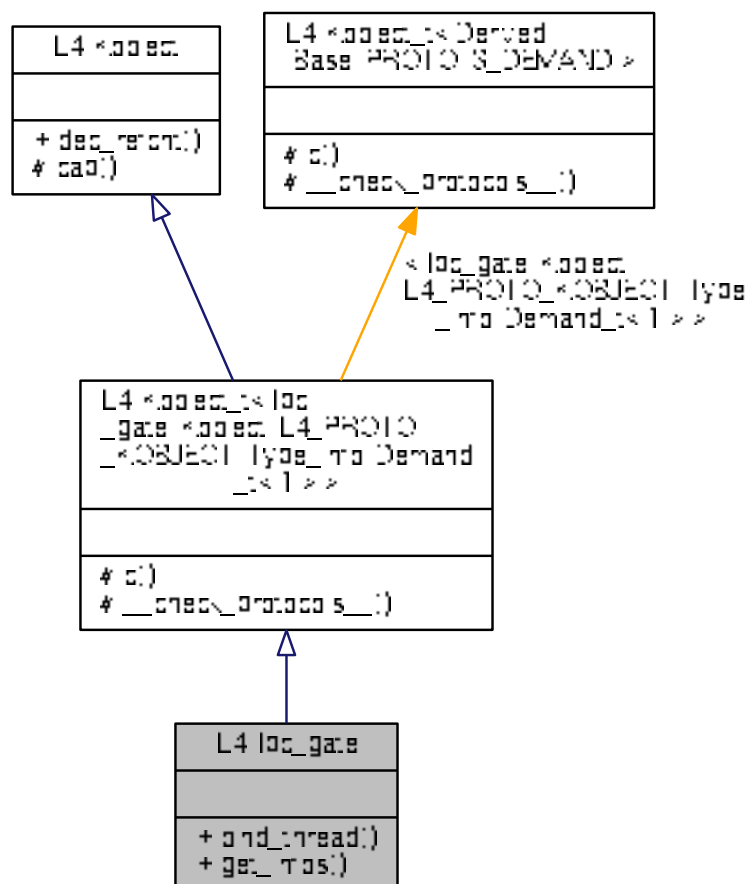
The documentation for this class was generated from the following file:

- l4/sys/cxx/ipc_varg

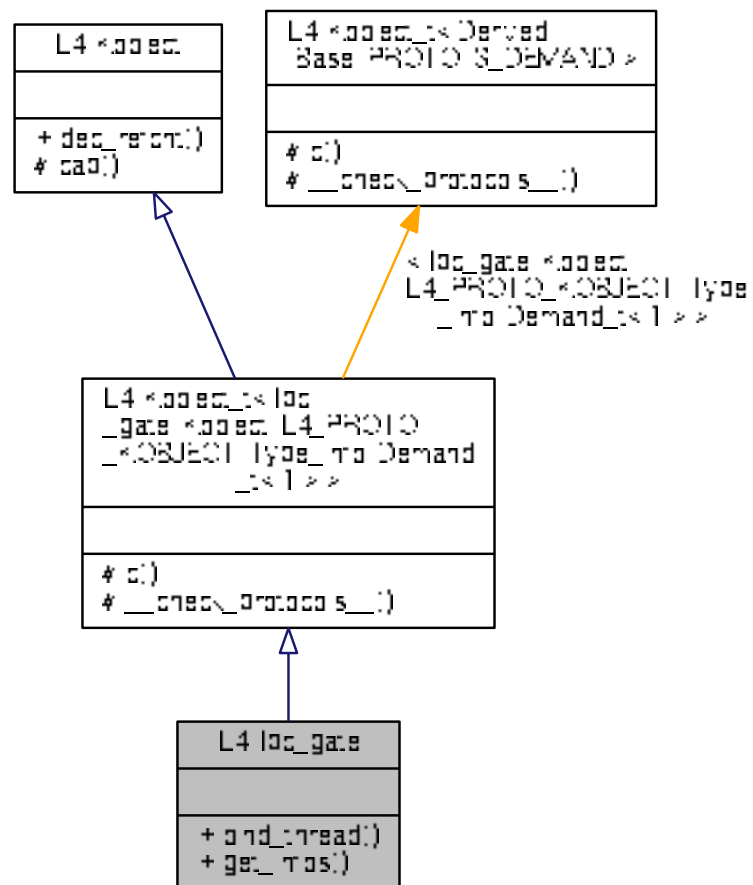
14.121 L4::lpc_gate Class Reference

The C++ IPC gate interface.

Inheritance diagram for L4::lpc_gate:



Collaboration diagram for L4::ipc_gate:



Public Member Functions

- `L4::msgtag_t bind_thread (Ipc::Cap< Thread > t, l4_umword_t label)`
Bind a thread to an IPC gate.
- `L4::msgtag_t get_infos (l4_umword_t *label)`
Get information about the IPC-gate.

Additional Inherited Members

14.121.1 Detailed Description

The C++ IPC gate interface.

IPC gates are used to create secure communication channels between protection domains. An IPC gate can be created using the `L4::Factory` interface. `L4::ipc_gate::bind_thread()` binds an `L4::Thread` as the receiver of all messages to an IPC gate.

The `bind_thread()` call allows to assign each IPC gate a kernel protected, machine-word sized payload called a *label*. It securely identifies the gate. The two least significant bits of the *label* are ORed with the `L4_CAP_FPAGE_S` and `L4_CAP_FPAGE_W` bits stored in the capability when transferred to the receiver. This means the *label* should usually have its two least significant bits set to zero. The *label* is only visible in the `L4::Task` which is running the thread the IPC gate was bound to and cannot be altered by the sender.

Include File

```
#include <l4/sys/ipc_gate>
```

For the C interface refer to the C [IPC-Gate API](#).

Definition at line 55 of file `ipc_gate`.

14.121.2 Member Function Documentation

14.121.2.1 bind_thread()

```
l4_msgtag_t L4::IpC_gate::bind_thread (
    IpC::Cap< Thread > t,
    l4_umword_t label )
```

Bind a thread to an IPC gate.

Parameters

<i>t</i>	Thread object that shall be bound to this IPC gate.
<i>label</i>	Label to assign to this IPC gate. The two least significant bits should usually be set to zero.

Returns

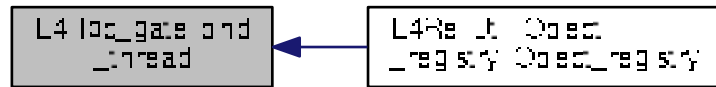
Syscall return tag containing one of the following return codes.

Return values

<code>L4_EOK</code>	Operation successful.
<code>-L4_EINVAL</code>	<i>t</i> is not a thread object or other arguments were malformed.
<code>-L4_EPERM</code>	<i>t</i> is missing L4_CAP_FPAGE_S right.

Referenced by `L4Re::Util::Object_registry::Object_registry()`.

Here is the caller graph for this function:



14.121.2.2 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.

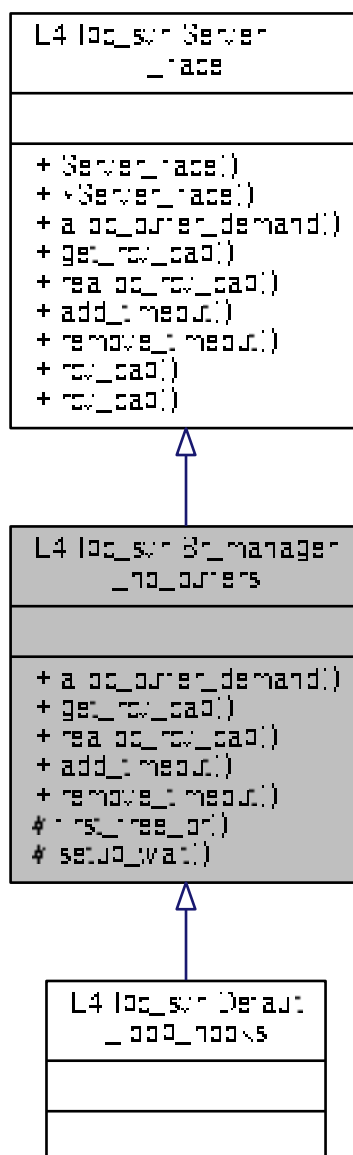
The documentation for this class was generated from the following file:

- [l4/sys/ipc_gate](#)

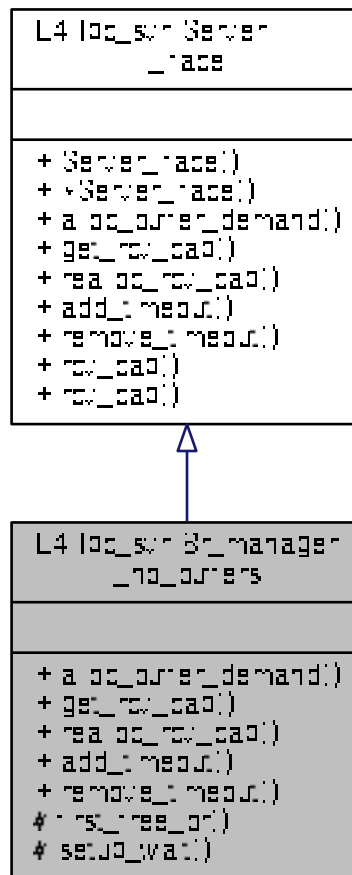
14.122 L4::ipc_svr::Br_manager_no_buffers Class Reference

Empty implementation of [Server_iface](#).

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)
Tells the server to allocate buffers for the given demand.
- [L4::Cap](#)< void > [get_rcv_cap](#) (int) const
Returns [L4::Cap](#)< void> ::Invalid, we have no buffer management.
- int [realloc_rcv_cap](#) (int)
Returns -L4_ENOMEM, we have no buffer management.
- int [add_timeout](#) (Timeout *, [l4_kernel_clock_t](#))
Returns -L4_ENOSYS, we have no timeout queue.
- int [remove_timeout](#) (Timeout *)
Returns -L4_ENOSYS, we have no timeout queue.

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

14.122.1 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 216 of file [ipc_server_loop](#).

14.122.2 Member Function Documentation

14.122.2.1 `alloc_buffer_demand()`

```
int L4::Ipc_svr::Br_manager_no_buffers::alloc_buffer_demand (
    Demand const & demand ) [inline], [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

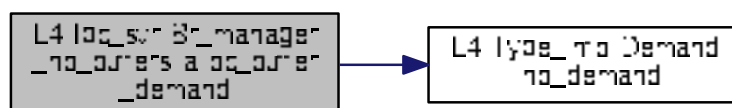
success (0) if demand is empty, -L4_ENOMEM else.

Implements [L4::lpc_svr::Server_iface](#).

Definition at line 223 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:



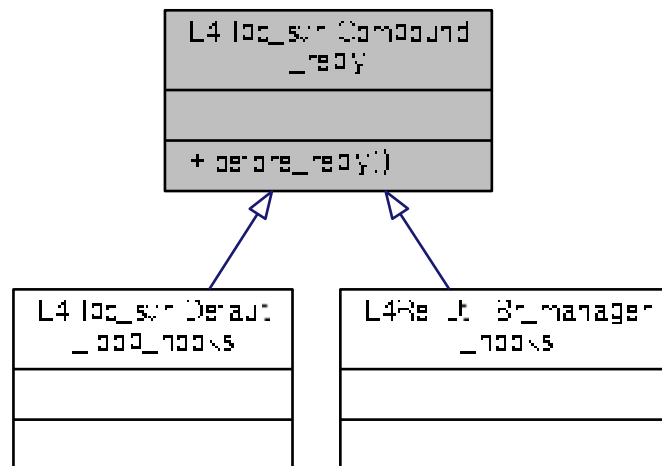
The documentation for this class was generated from the following file:

- l4/sys/cxx/ipc_server_loop

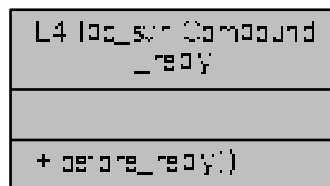
14.123 L4::lpc_svr::Compound_reply Struct Reference

Mix in for LOOP_HOOKS to always use compound reply and wait.

Inheritance diagram for L4::lpc_svr::Compound_reply:



Collaboration diagram for L4::lpc_svr::Compound_reply:



14.123.1 Detailed Description

Mix in for LOOP_HOOKS to always use compound reply and wait.

Definition at line 77 of file [ipc_server_loop](#).

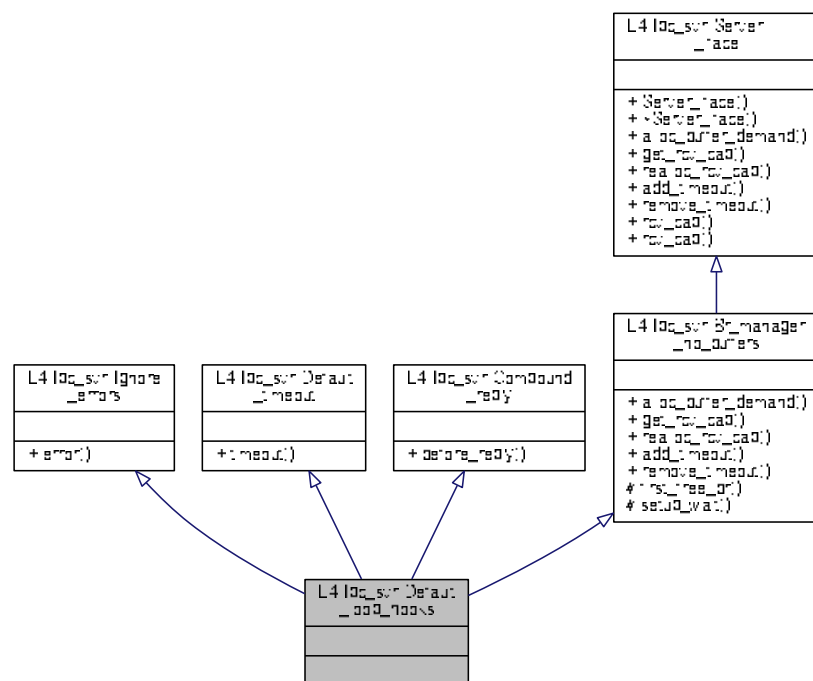
The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_server_loop

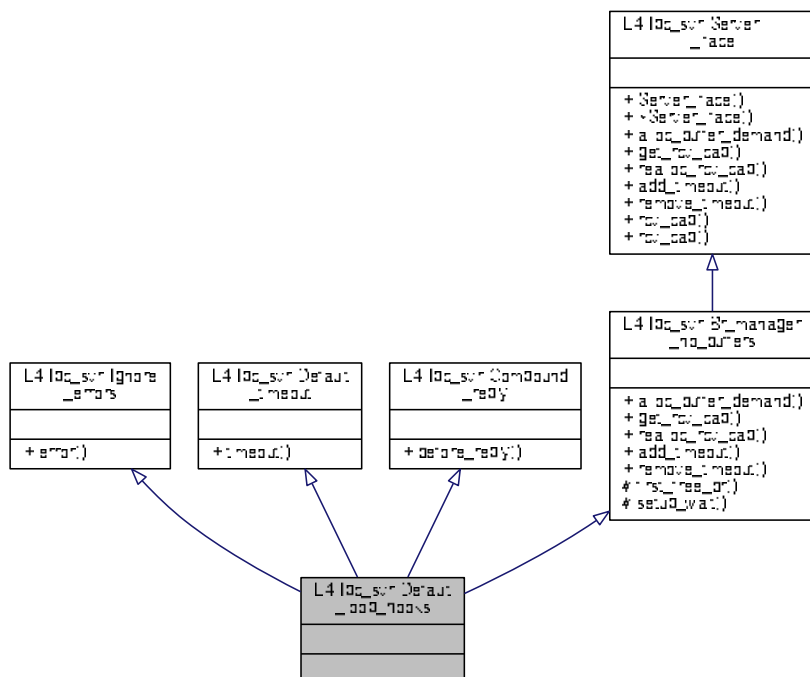
14.124 L4::lpc_svr::Default_loop_hooks Struct Reference

Default LOOP_HOOKS.

Inheritance diagram for L4::lpc_svr::Default_loop_hooks:



Collaboration diagram for L4::ipc_srvr::Default_loop_hooks:



Additional Inherited Members

14.124.1 Detailed Description

Default LOOP_HOOKS.

Combination of [Ignore_errors](#), [Default_timeout](#), [Compound_reply](#), and [Br_manager_no_buffers](#).

Definition at line 268 of file [ipc_server_loop](#).

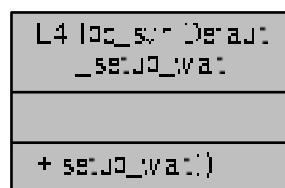
The documentation for this struct was generated from the following file:

- [l4/sys/cxx/ipc_server_loop](#)

14.125 L4::ipc_srvr::Default_setup_wait Struct Reference

Mix in for LOOP_HOOKS for setup_wait no op.

Collaboration diagram for L4::lpc_svr::Default_setup_wait:



14.125.1 Detailed Description

Mix in for LOOP_HOOKS for setup_wait no op.

Definition at line 88 of file [ipc_server_loop](#).

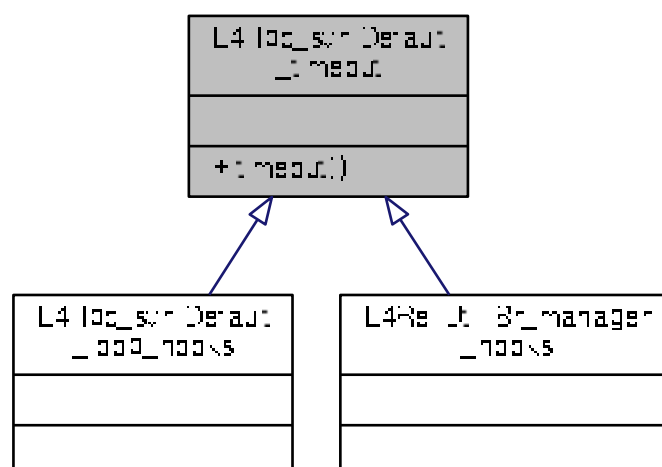
The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_server_loop

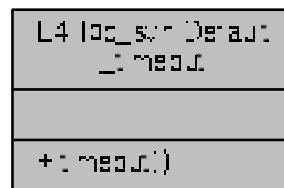
14.126 L4::lpc_svr::Default_timeout Struct Reference

Mix in for LOOP_HOOKS to use a 0 send and a infinite receive timeout.

Inheritance diagram for L4::lpc_svr::Default_timeout:



Collaboration diagram for L4::lpc_svr::Default_timeout:



14.126.1 Detailed Description

Mix in for LOOP_HOOKS to use a 0 send and a infinite receive timeout.

Definition at line 69 of file [ipc_server_loop](#).

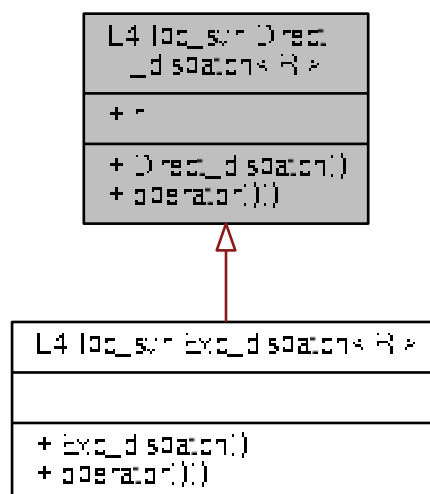
The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_server_loop

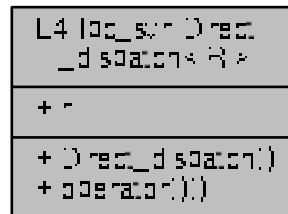
14.127 L4::lpc_svr::Direct_dispatch< R > Struct Template Reference

Direct disptach helper, for forwarding dispatch calls a registry *R*.

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

14.127.1 Detailed Description

```
template<typename R>
struct L4::lpc_svr::Direct_dispatch< R >
```

Direct dispatch helper, for forwarding dispatch calls 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 139 of file [ipc_server_loop](#).

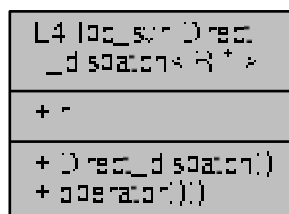
The documentation for this struct was generated from the following file:

- l4/sys/cxx/ipc_server_loop

14.128 L4::lpc_svr::Direct_dispatch< R * > Struct Template Reference

Direct disptach helper, for forwarding dispatch calls via a pointer to a registry *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

14.128.1 Detailed Description

```
template<typename R>
struct L4::lpc_svr::Direct_dispatch< R * >
```

Direct disptach 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 160 of file [ipc_server_loop](#).

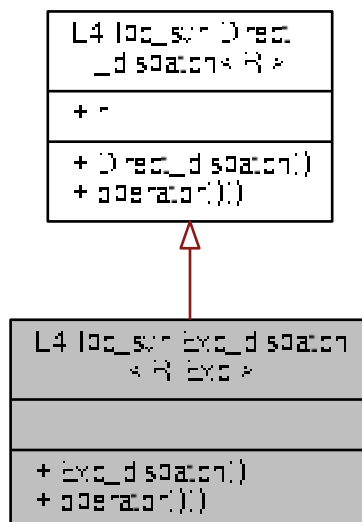
The documentation for this struct was generated from the following file:

- `l4/sys/cxx/ipc_server_loop`

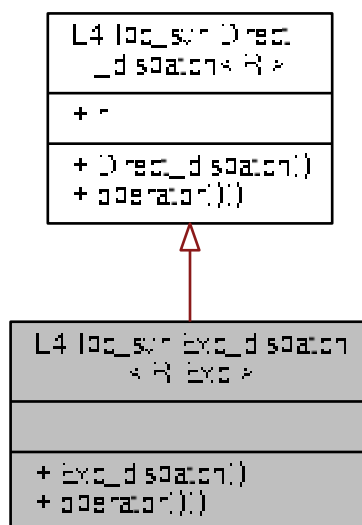
14.129 L4::lpc_svr::Exc_dispatch< R, Exc > Struct Template Reference

Dispatch helper wrapping try {} catch {} around the dispatch call.

Inheritance diagram for L4::lpc_svr::Exc_dispatch< R, Exc >:



Collaboration diagram for L4::lpc_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.

14.129.1 Detailed Description

```
template<typename R, typename Exc>
struct L4::lpc_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 184 of file [ipc_server_loop](#).

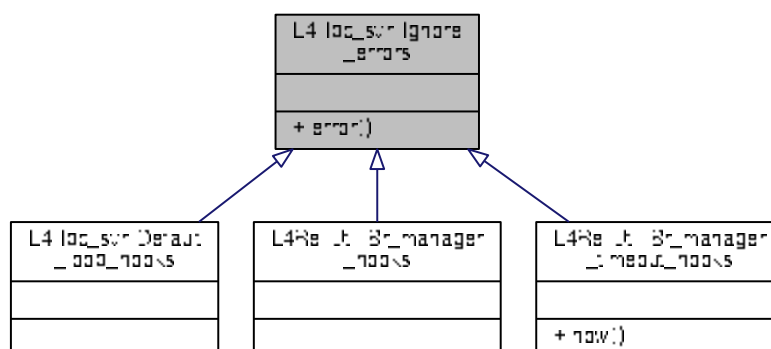
The documentation for this struct was generated from the following file:

- `l4/sys/cxx/ipc_server_loop`

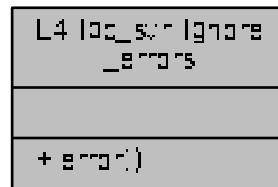
14.130 L4::lpc_svr::Ignore_errors Struct Reference

Mix in for LOOP_HOOKS to ignore IPC errors.

Inheritance diagram for `L4::lpc_svr::Ignore_errors`:



Collaboration diagram for L4::lpc_svr::Ignore_errors:



14.130.1 Detailed Description

Mix in for LOOP_HOOKS to ignore IPC errors.

Definition at line 61 of file [ipc_server_loop](#).

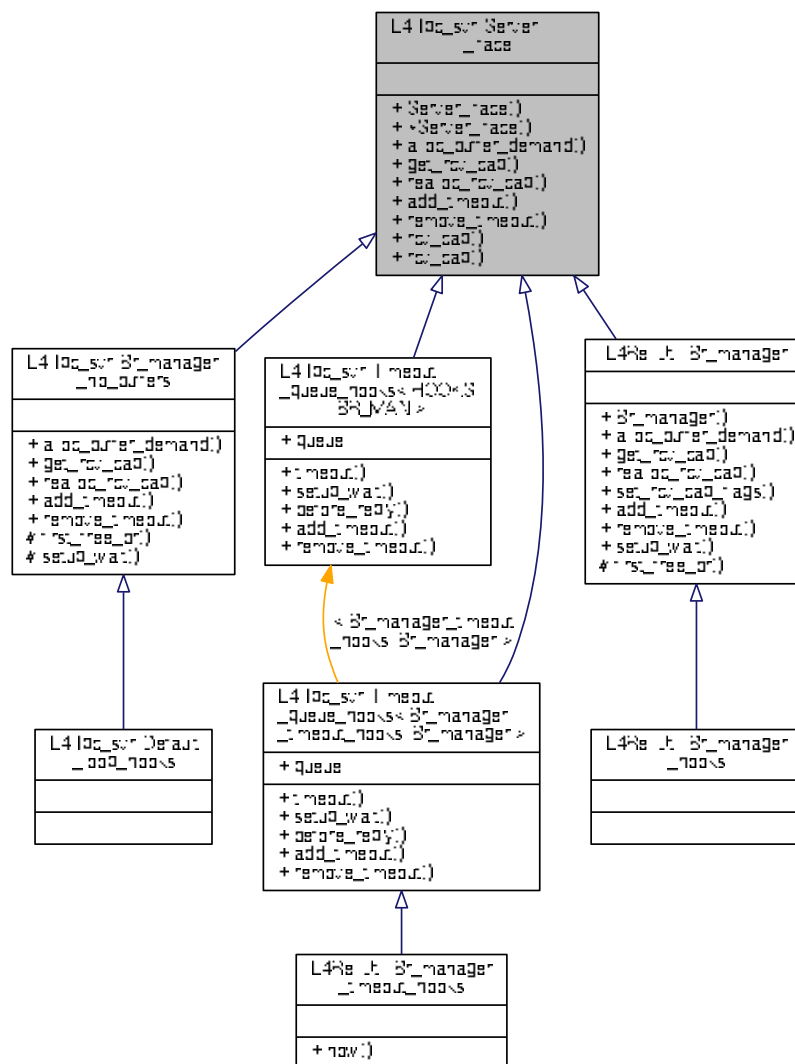
The documentation for this struct was generated from the following file:

- `l4/sys/cxx/ipc_server_loop`

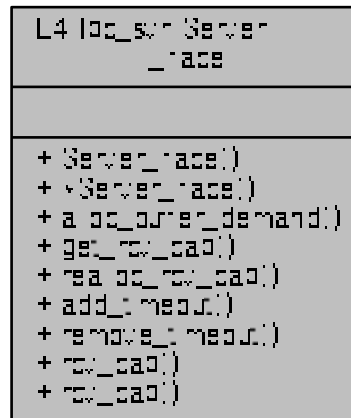
14.131 L4::lpc_svr::Server_iface Class Reference

Interface for server-loop related functions.

Inheritance diagram for L4::lpc_svr::Server_iface:



Collaboration diagram for L4::lpc_svr::Server_iface:



Public Types

- typedef [L4::Type_info::Demand](#) 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 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.

14.131.1 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 45 of file [ipc_epiface](#).

14.131.2 Member Function Documentation

14.131.2.1 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.

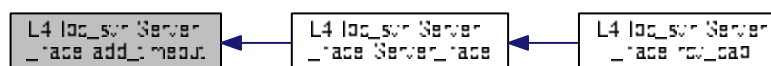
Returns

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](#).

Referenced by [Server_iface\(\)](#).

Here is the caller graph for this function:



14.131.2.2 `alloc_buffer_demand()`

```
virtual int L4::lpc_svr::Server_iface::alloc_buffer_demand (
    Demand const & demand ) [pure virtual]
```

Tells the server to allocate buffers for the given demand.

Parameters

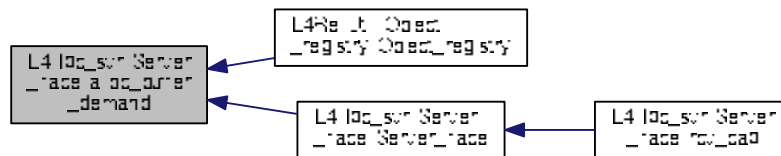
<i>demand</i>	The total server-side demand of receive buffers needed for a given interface, see Demand.
---------------	---

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](#).

Referenced by [L4Re::Util::Object_registry::Object_registry\(\)](#), and [Server_iface\(\)](#).

Here is the caller graph for this function:

14.131.2.3 `get_rcv_cap()`

```
virtual L4::Cap<void> L4::lpc_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\(\)](#)

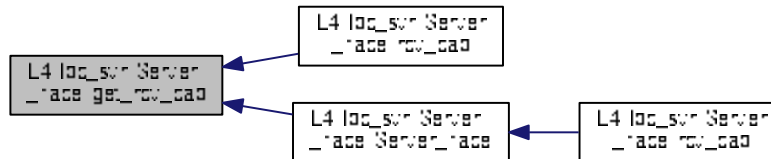
Returns

Capability slot currently allocated to the given receive buffer.

Implemented in [L4::Ipc_svr::Br_manager_no_buffers](#), and [L4Re::Util::Br_manager](#).

Referenced by [rcv_cap\(\)](#), and [Server_iface\(\)](#).

Here is the caller graph for this function:

**14.131.2.4 rcv_cap()** [1/2]

```
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.

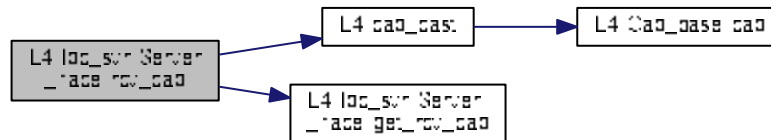
Note

This is a convenience wrapper for [get_rcv_cap\(\)](#) to avoid [L4::cap_cast<>\(\)](#).

Definition at line 120 of file [ipc_epiface](#).

References [L4::cap_cast\(\)](#), and [get_rcv_cap\(\)](#).

Here is the call graph for this function:

**14.131.2.5 rcv_cap()** [2/2]

```
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.

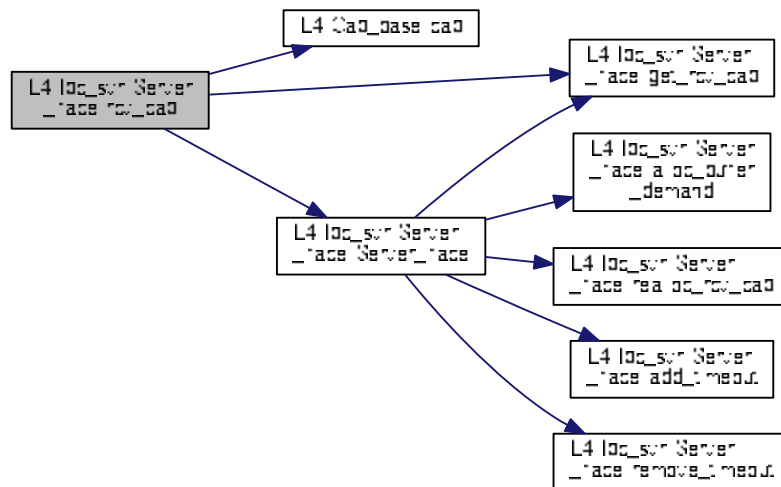
Note

This is a convenience wrapper for [get_rcv_cap\(\)](#).

Definition at line 132 of file [ipc_epiface](#).

References [L4::Cap_base::cap\(\)](#), [get_rcv_cap\(\)](#), [L4_CAP_MASK](#), and [Server_iface\(\)](#).

Here is the call graph for this function:



14.131.2.6 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\(\)](#)

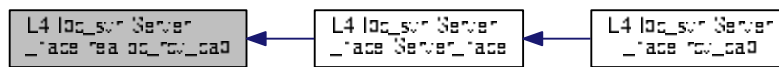
Returns

0 on success, < 0 on error.

Implemented in [L4::ipc_svr::Br_manager_no_buffers](#), and [L4Re::Util::Br_manager](#).

Referenced by [Server_iface\(\)](#).

Here is the caller graph for this function:



14.131.2.7 remove_timeout()

```
virtual int L4::lpc_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.
----------------	-------------------------------

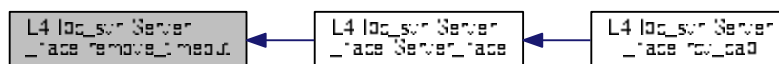
Returns

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](#).

Referenced by [Server_iface\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

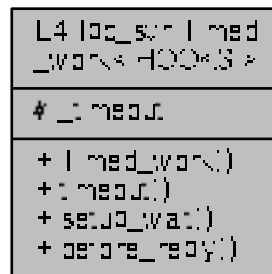
- `l4/sys/cxx/ipc_epiface`

14.132 L4::ipc_svr::Timed_work< HOOKS > Class Template Reference

DEPRECATED.

Inherits [HOOKS](#).

Collaboration diagram for L4::ipc_svr::Timed_work< HOOKS >:



14.132.1 Detailed Description

```
template<typename HOOKS>
class L4::ipc_svr::Timed_work< HOOKS >
```

DEPRECATED.

Deprecated Use [L4::ipc_svr::Timeout_queue_hooks](#)

Definition at line 97 of file [ipc_server_loop](#).

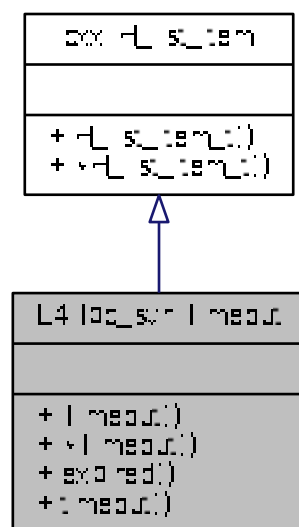
The documentation for this class was generated from the following file:

- `l4/sys/cxx/ipc_server_loop`

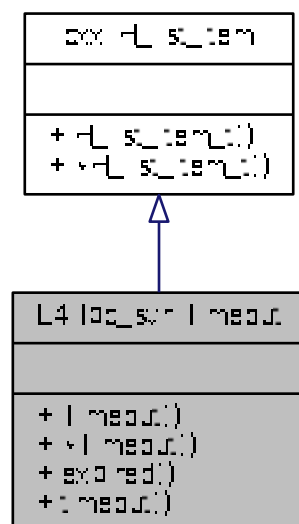
14.133 L4::ipc_svr::Timeout Class Reference

Callback interface for [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.

14.133.1 Detailed Description

Callback interface for `Timeout_queue`.

Definition at line 20 of file `ipc_timeout_queue`.

14.133.2 Member Function Documentation

14.133.2.1 `expired()`

```
virtual void L4::Ipc_svr::Timeout::expired ( ) [pure virtual]
```

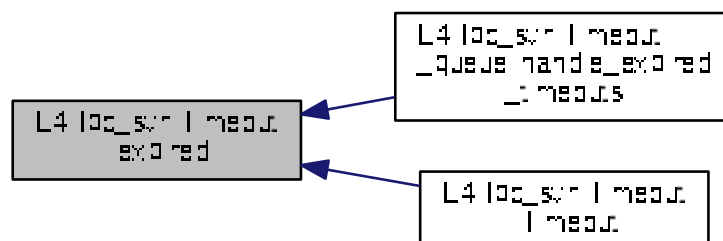
callback function to be called when timeout happened

Note

The timeout object is already dequeued when this function is called, this means the timeout may be safely again within the `expired()` function.

Referenced by `L4::Ipc_svr::Timeout_queue::handle_expired_timeouts()`, and `Timeout()`.

Here is the caller graph for this function:



14.133.2.2 timeout()

```
l4_kernel_clock_t L4::lpc_svr::Timeout::timeout ( ) const [inline]
```

return absolute timeout of this callback.

Returns

absolute timeout for this instance of the timeout.

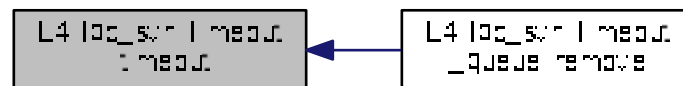
Precondition

The timeout object must have been in a queue before, otherwise the timeout is not set.

Definition at line 43 of file [ipc_timeout_queue](#).

Referenced by [L4::lpc_svr::Timeout_queue::remove\(\)](#).

Here is the caller graph for this function:



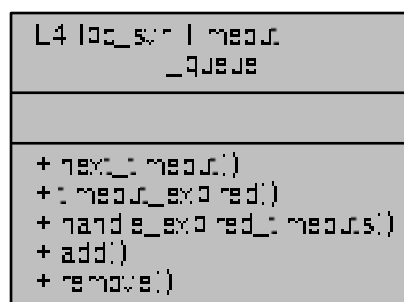
The documentation for this class was generated from the following file:

- `I4/cxx/ipc_timeout_queue`

14.134 L4::lpc_svr::Timeout_queue Class Reference

[Timeout](#) queue to be used in I4re server loop.

Collaboration diagram for L4::lpc_svr::Timeout_queue:



Public Types

- typedef [L4::lpc_svr::Timeout](#) [Timeout](#)
Provide a local definition of [Timeout](#) for backward compat.

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.

14.134.1 Detailed Description

[Timeout](#) queue to be used in l4re server loop.

Definition at line 56 of file [ipc_timeout_queue](#).

14.134.2 Member Function Documentation

14.134.2.1 [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 112 of file [ipc_timeout_queue](#).

14.134.2.2 handle_expired_timeouts()

```
void L4::Ipc_svr::Timeout_queue::handle_expired_timeouts (
    l4_kernel_clock_t now ) [inline]
```

run the callbacks of expired timeouts

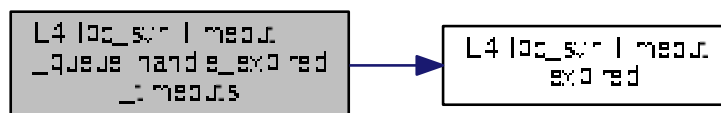
Parameters

<i>now</i>	the current time.
------------	-------------------

Definition at line 92 of file [ipc_timeout_queue](#).

References [L4::Ipc_svr::Timeout::expired\(\)](#).

Here is the call graph for this function:



14.134.2.3 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 66 of file [ipc_timeout_queue](#).

14.134.2.4 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
----------------	--------------------------------------

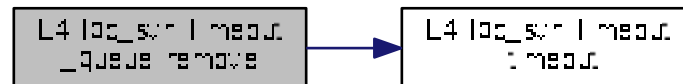
Precondition

timeout must be in this queue

Definition at line 127 of file [ipc_timeout_queue](#).

References [L4::ipc_svr::Timeout::timeout\(\)](#).

Here is the call graph for this function:

**14.134.2.5 timeout_expired()**

```
bool L4::Ipc_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 82 of file [ipc_timeout_queue](#).

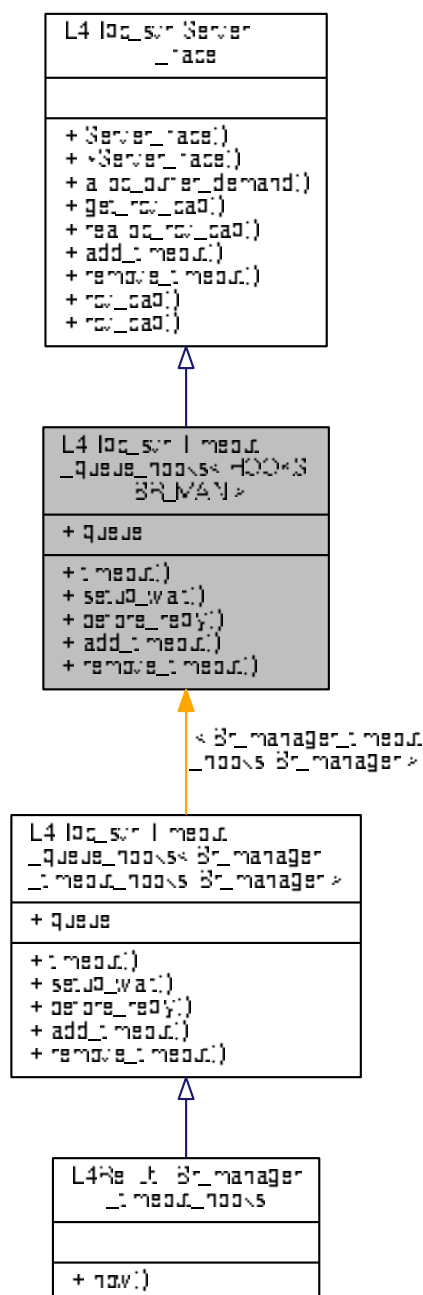
The documentation for this class was generated from the following file:

- `l4/cxx/ipc_timeout_queue`

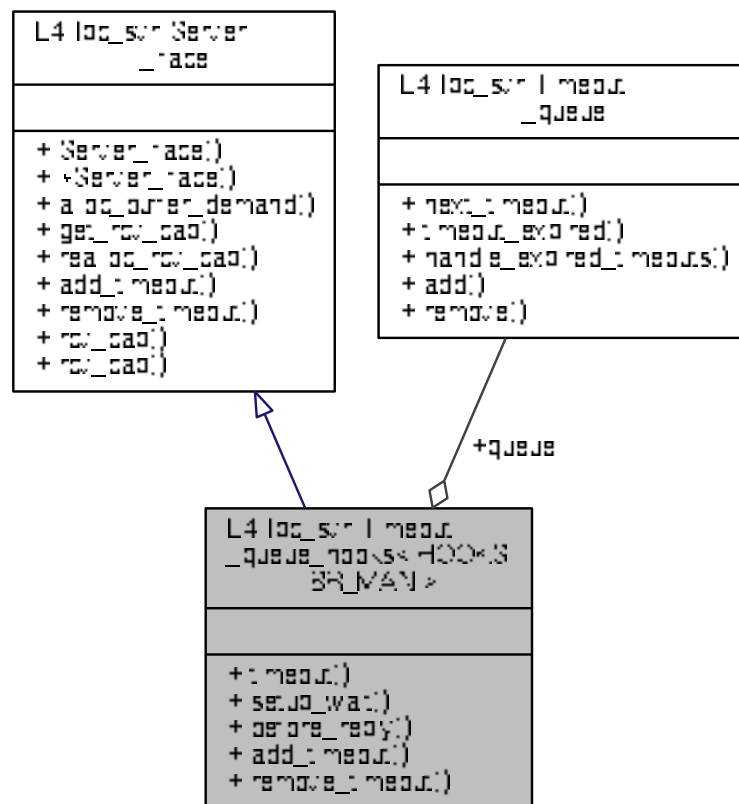
14.135 L4::lpc_svr::Timeout_queue_hooks< HOOKS, BR_MAN > Class Template Reference

Loop hooks mixin for integrating a timeout queue into the server loop.

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)`
Add a timeout to the queue for time time.
- `int remove_timeout (Timeout *timeout)`
Remove timeout from the queue.

Data Fields

- `Timeout_queue queue`
Use this timeout queue.

Additional Inherited Members

14.135.1 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 <code>Timeout_queue_hooks<></code> 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 151 of file [ipc_timeout_queue](#).

14.135.2 Member Function Documentation

14.135.2.1 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], [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 203 of file [ipc_timeout_queue](#).

14.135.2.2 `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], [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 216 of file [ipc_timeout_queue](#).

The documentation for this class was generated from the following file:

- `I4/cxx/ipc_timeout_queue`

14.136 L4::Irq Class Reference

C++ [Irq](#) interface.

- `l4_msgtag_t receive (l4_timeout_t timeout=L4_IPC_NEVER, l4_utcb_t *utcb=l4_utcb()) throw ()`
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()) throw ()`
Unmask IRQ and (open) wait for any message.
- `l4_msgtag_t unmask (l4_utcb_t *utcb=l4_utcb()) throw ()`
Unmask IRQ.

Additional Inherited Members

14.136.1 Detailed Description

C++ [Irq](#) interface.

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)

[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 111 of file [irq](#).

14.136.2 Member Function Documentation

14.136.2.1 attach()

```
l4_msgtag_t L4::Irq::attach (
    l4_umword_t label,
    Cap< Thread > const & thread = Cap<Thread>::Invalid,
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline]
```

Attach a thread to this interrupt.

Parameters

<i>label</i>	Identifier of the IRQ (<i>protected label</i> used for messages)
<i>thread</i>	Capability of the thread to attach the IRQ to.
<i>utcb</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

Syscall return tag

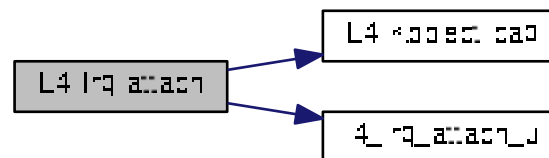
The *protected label* is stored in the kernel and sent to the attached thread with the IRQ-triggered notification. It allows the receiver thread to securely identify the IRQ.

Definition at line 130 of file `irq`.

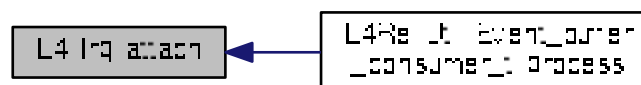
References `L4::Kobject::cap()`, and `l4_irq_attach_u()`.

Referenced by `L4Re::Util::Event_buffer_consumer_t< PAYLOAD >::process()`.

Here is the call graph for this function:



Here is the caller graph for this function:



14.136.2.2 detach()

```

l4_msgtag_t L4::Irq::detach (
    l4_utcb_t * utcb = l4_utcb() ) throw() [inline]
  
```

Detach from this interrupt.

Parameters

<i>utcb</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.
-------------	---

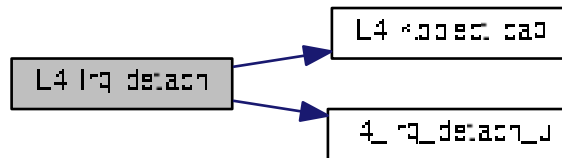
Returns

Syscall return tag

Definition at line 142 of file [irq](#).

References [L4::Kobject::cap\(\)](#), and [l4_irq_detach_u\(\)](#).

Here is the call graph for this function:



14.136.2.3 receive()

```

l4_msgtag_t L4::Irq::receive (
    l4_timeout_t timeout = L4_IPC_NEVER,
    l4_utcb_t * utcb = l4_utcb() ) throw() [inline]
  
```

Unmask and wait for this IRQ.

Parameters

<i>timeout</i>	Timeout.
<i>utcb</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

Syscall return tag

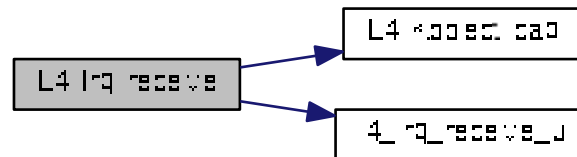
Note

If this is the function normally used for your IRQs consider using [L4::Semaphore](#) instead of [L4::Irq](#).

Definition at line 157 of file [irq](#).

References [L4::Kobject::cap\(\)](#), and [l4_irq_receive_u\(\)](#).

Here is the call graph for this function:



14.136.2.4 unmask()

```
l4_msgtag_t L4::Irq::unmask (
    l4_utcb_t * utcb = l4_utcb() ) throw() [inline]
```

Unmask IRQ.

Parameters

<i>utcb</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.
-------------	---

Returns

Syscall return tag for a send-only operation, use [l4_ipc_error\(\)](#) to check for errors (**do not** use [l4_error\(\)](#)).

Note

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 errors, **do not** use [l4_error\(\)](#), because [l4_error\(\)](#) will always return an error.

[Irq::wait\(\)](#) and [Irq::receive\(\)](#) operations already include an [unmask\(\)](#), do not use an extra [unmask\(\)](#) in these cases.

Deprecated Use [L4::Irq_eoi::unmask\(\)](#)

Definition at line 193 of file [irq](#).

References [L4_IPC_NEVER](#), and [L4::Irq_eoi::unmask\(\)](#).

Here is the call graph for this function:



14.136.2.5 wait()

```
l4_msgtag_t L4::Irq::wait (
    l4_umword_t * label,
    l4_timeout_t timeout = L4_IPC_NEVER,
    l4_utcb_t * utcb = l4_utcb() ) throw() [inline]
```

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, usually the UTCB of the calling thread.

Returns

Syscall return tag

Definition at line 170 of file [irq](#).

References [L4::Irq_eoi::unmask\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

- [l4/sys/irq](#)

14.137.1 Detailed Description

Interface for sending an acknowledge message to an object.

The object is usually an ICU or an IRQ.

Definition at line 40 of file [irq](#).

14.137.2 Member Function Documentation

14.137.2.1 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() ) throw ()    [inline]
```

Acknowledge the given interrupt line.

Parameters

	<i>irqnum</i>	The interrupt line that shall be acknowledged.
out	<i>label</i>	If NULL this is a send-only unmask, if not NULL then 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, usually the UTCB of the calling thread.

Returns

Syscall return tag.

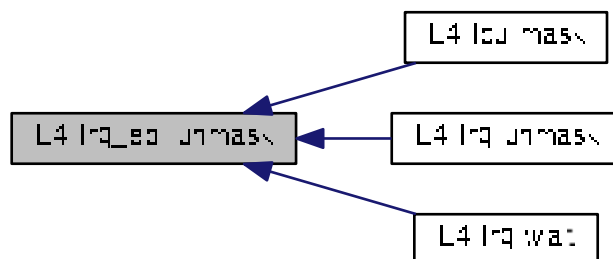
Note

If *label* is NULL this function is a send-only operation and there is no return value except for a failed send operation. In this case use [l4_ipc_error\(\)](#) to check for errors, **do not** use [l4_error\(\)](#), because [l4_error\(\)](#) will always return an error.

Definition at line 62 of file [irq](#).

Referenced by [L4::Icu::mask\(\)](#), [L4::Irq::unmask\(\)](#), and [L4::Irq::wait\(\)](#).

Here is the caller graph for this function:



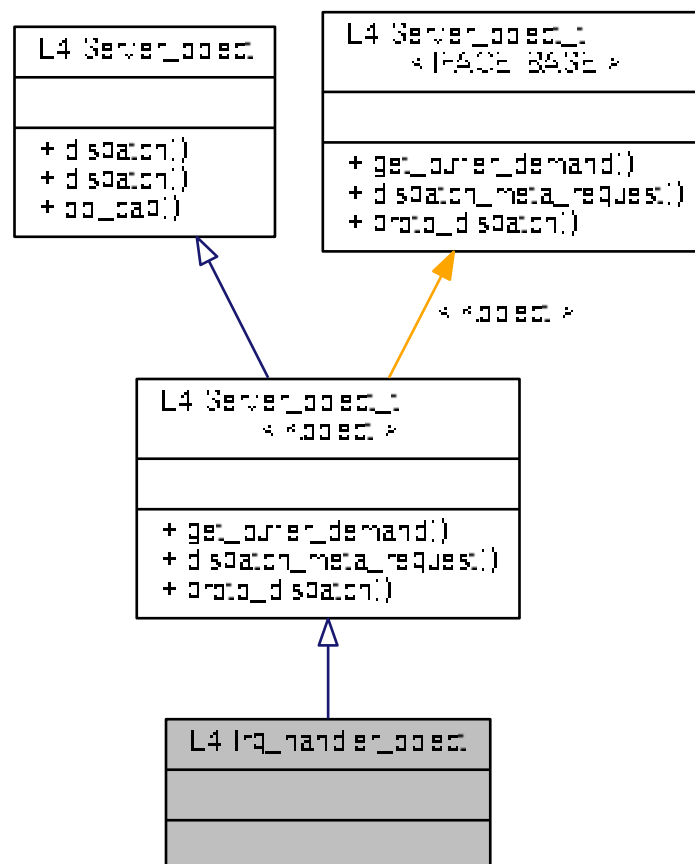
The documentation for this class was generated from the following file:

- [l4/sys/irq](#)

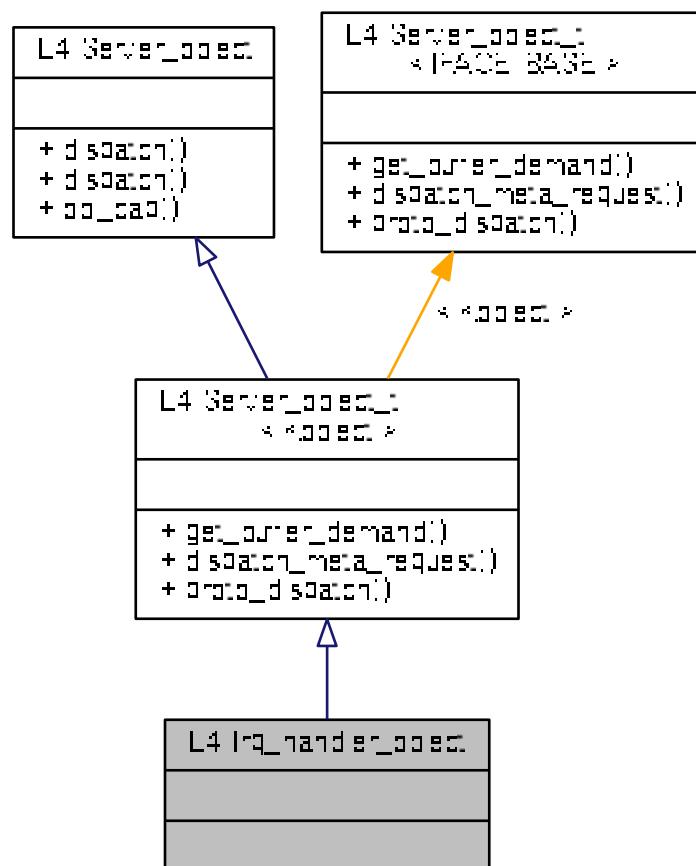
14.138 L4::Irq_handler_object Struct Reference

[Server](#) object base class for handling IRQ messages.

Inheritance diagram for L4::Irq_handler_object:



Collaboration diagram for L4::Irq_handler_object:



Additional Inherited Members

14.138.1 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:

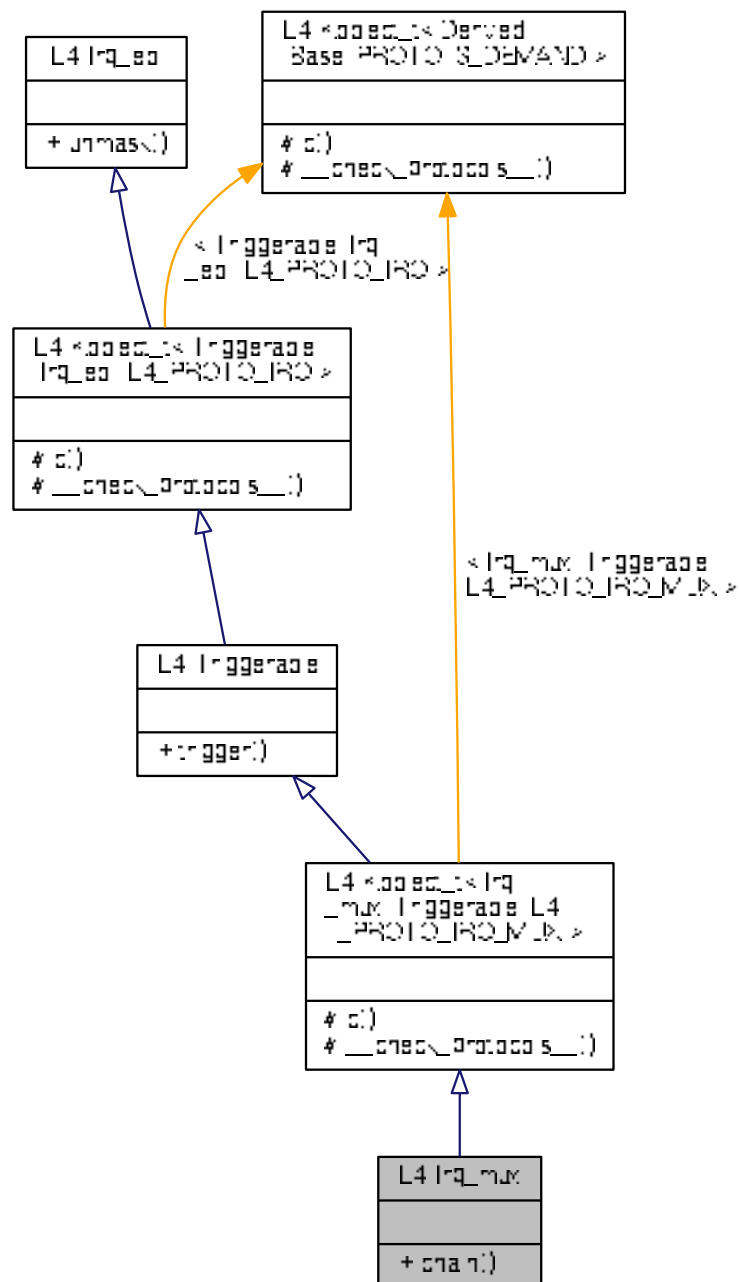
[examples/libs/l4re/c++/shared_ds/ds_srv.cc](#).

Definition at line 173 of file [ipc_server](#).

The documentation for this struct was generated from the following file:

- [l4/cxx/ipc_server](#)

Collaboration diagram for L4::Irq_mux:



Public Member Functions

- `l4_msgtag_t chain (Cap< Triggerable > const &slave, l4_utcb_t *utcb=l4_utcb()) throw ()`

Attach an IRQ to this multiplexer.

Additional Inherited Members

14.139.1 Detailed Description

IRQ multiplexer for shared IRQs.

This interface allows broadcasting of shared IRQs to multiple triggerables. The IRQ multiplexer is responsible for the correct mask and unmask logic for such shared IRQs.

The semantics are that each of the slave IRQs is triggered whenever the multiplexer IRQ is triggered. As shared IRQs are usually level-triggered, the real IRQ source will be masked automatically when an IRQ is delivered and shall be unmasked when all attached slave IRQs are acknowledged.

Definition at line 210 of file [irq](#).

14.139.2 Member Function Documentation

14.139.2.1 chain()

```
l4_msgtag_t L4::Irq_mux::chain (
    Cap< Triggerable > const & slave,
    l4_utcb_t * utcb = l4_utcb() ) throw () [inline]
```

Attach an IRQ to this multiplexer.

Parameters

<i>slave</i>	The slave that shall be attached to the master.
<i>utcb</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.

Returns

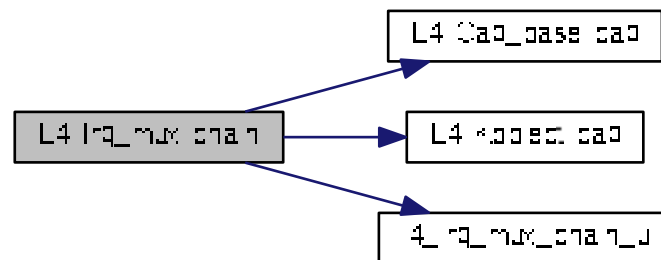
Syscall return tag

The chaining feature of IRQ objects allows to deal with shared IRQs. For chaining IRQs there must be an IRQ multiplexer ([Irq_mux](#)) bound to the real IRQ source. This function allows to add slave IRQs to this multiplexer.

Definition at line 225 of file [irq](#).

References [L4::Cap_base::cap\(\)](#), [L4::Kobject::cap\(\)](#), and [l4_irq_mux_chain_u\(\)](#).

Here is the call graph for this function:



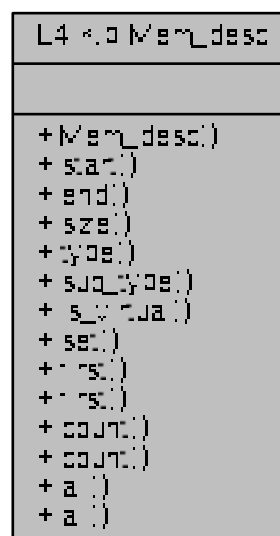
The documentation for this struct was generated from the following file:

- [l4/sys/irq](#)

14.140 L4::Kip::Mem_desc Class Reference

Memory descriptors stored in the kernel interface page.

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_rsdp](#) = 0 }
Memory sub types for the Mem_type::Info type.

Public Member Functions

- [Mem_desc](#) (unsigned long [start](#), unsigned long [end](#), [Mem_type](#) t, unsigned char st=0, bool virt=false) throw ()
Initialize memory descriptor.
- unsigned long [start](#) () const throw ()
Return start address of memory descriptor.
- unsigned long [end](#) () const throw ()
Return end address of memory descriptor.
- unsigned long [size](#) () const throw ()
Return size of region described by the memory descriptor.
- [Mem_type](#) type () const throw ()
Return type of the memory descriptor.
- unsigned char [sub_type](#) () const throw ()
Return sub-type of the memory descriptor.
- unsigned [is_virtual](#) () const throw ()
Return whether the memory descriptor describes a virtual or physical region.
- void [set](#) (unsigned long [start](#), unsigned long [end](#), [Mem_type](#) t, unsigned char st=0, bool virt=false) throw ()
Set values of a memory descriptor.

Static Public Member Functions

- static [Mem_desc](#) * [first](#) (void *kip) throw ()
Get first memory descriptor.
- static unsigned long [count](#) (void const *kip) throw ()
Return number of memory descriptors stored in the kernel info page.
- static void [count](#) (void *kip, unsigned count) throw ()
Set number of memory descriptors.
- static [cxx::static_vector](#)< [Mem_desc](#) const > [all](#) (void const *kip)
Return enumerable list of memory descriptors.
- static [cxx::static_vector](#)< [Mem_desc](#) > [all](#) (void *kip)
Return enumerable list of memory descriptors.

14.140.1 Detailed Description

Memory descriptors stored in the kernel interface page.

Include File

```
#include <l4/sys/kip>
```

Definition at line 53 of file [kip](#).

14.140.2 Member Enumeration Documentation

14.140.2.1 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.
----------------	--

Definition at line 75 of file [kip](#).

14.140.2.2 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 59 of file [kip](#).

14.140.3 Constructor & Destructor Documentation

14.140.3.1 Mem_desc()

```
L4::Kip::Mem_desc::Mem_desc (
    unsigned long start,
    unsigned long end,
```

```
Mem_type t,
unsigned char st = 0,
bool virt = false ) throw )    [inline]
```

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

Definition at line 166 of file [kip](#).

14.140.4 Member Function Documentation

14.140.4.1 all() [1/2]

```
static cxx::static_vector<Mem_desc const> L4::Kip::Mem_desc::all (
    void const * kip )    [inline], [static]
```

Return enumerable list of memory descriptors.

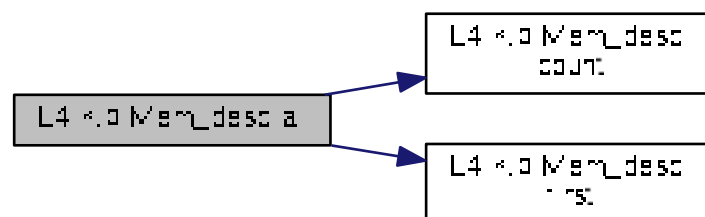
Parameters

<i>kip</i>	Pointer to the kernel info page.
------------	----------------------------------

Definition at line 139 of file [kip](#).

References [count\(\)](#), and [first\(\)](#).

Here is the call graph for this function:



14.140.4.2 `all()` [2/2]

```
static cxx::static_vector<Mem_desc> L4::Kip::Mem_desc::all (
    void * kip ) [inline], [static]
```

Return enumerable list of memory descriptors.

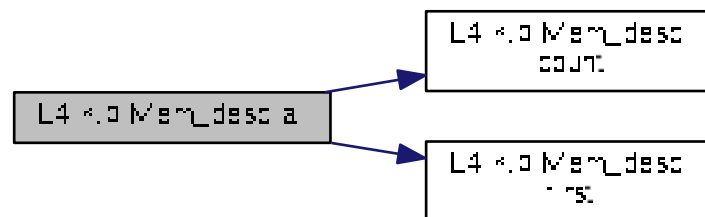
Parameters

<i>kip</i>	Pointer to the kernel info page.
------------	----------------------------------

Definition at line 150 of file [kip](#).

References [count\(\)](#), and [first\(\)](#).

Here is the call graph for this function:



14.140.4.3 `count()` [1/2]

```
static unsigned long L4::Kip::Mem_desc::count (
    void const * kip ) throw ) [inline], [static]
```

Return number of memory descriptors stored in the kernel info page.

Parameters

<i>kip</i>	Pointer to the kernel info page
------------	---------------------------------

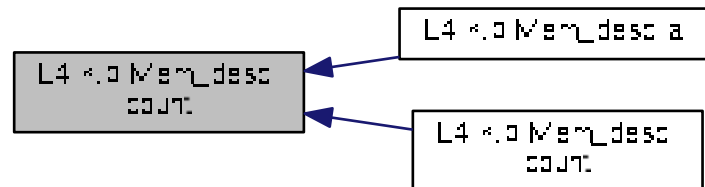
Returns

Number of memory descriptors in the kernel info page.

Definition at line 116 of file [kip](#).

Referenced by [all\(\)](#), and [count\(\)](#).

Here is the caller graph for this function:



14.140.4.4 `count()` [2/2]

```
static void L4::Kip::Mem_desc::count (
    void * kip,
    unsigned count ) throw ()    [inline], [static]
```

Set number of memory descriptors.

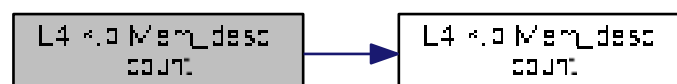
Parameters

<i>kip</i>	Pointer to the kernel info page
<i>count</i>	Number of memory descriptors

Definition at line 128 of file [kip](#).

References [count\(\)](#).

Here is the call graph for this function:



14.140.4.5 `end()`

```
unsigned long L4::Kip::Mem_desc::end ( ) const throw ( )    [inline]
```

Return end address of memory descriptor.

Returns

End address of memory descriptor

Definition at line 184 of file [kip](#).

Referenced by [size\(\)](#).

Here is the caller graph for this function:

14.140.4.6 `first()`

```
static Mem_desc* L4::Kip::Mem_desc::first (
    void * kip ) throw ( )    [inline], [static]
```

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 97 of file [kip](#).

Referenced by [all\(\)](#).

Here is the caller graph for this function:



14.140.4.7 `is_virtual()`

```
unsigned L4::Kip::Mem_desc::is_virtual ( ) const throw ( ) [inline]
```

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](#).

14.140.4.8 `set()`

```
void L4::Kip::Mem_desc::set (
    unsigned long start,
    unsigned long end,
    Mem_type t,
    unsigned char st = 0,
    bool virt = false ) throw ( ) [inline]
```

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

Definition at line 224 of file [kip](#).

14.140.4.9 size()

```
unsigned long L4::Kip::Mem_desc::size ( ) const throw ( ) [inline]
```

Return size of region described by the memory descriptor.

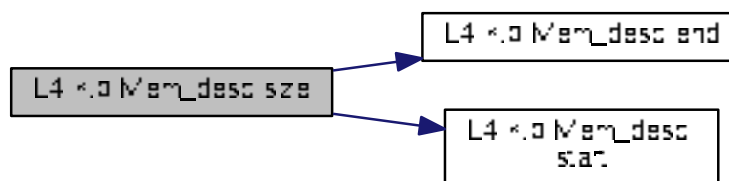
Returns

Size of the region described by the memory descriptor

Definition at line 191 of file [kip](#).

References [end\(\)](#), and [start\(\)](#).

Here is the call graph for this function:



14.140.4.10 start()

```
unsigned long L4::Kip::Mem_desc::start ( ) const throw ( ) [inline]
```

Return start address of memory descriptor.

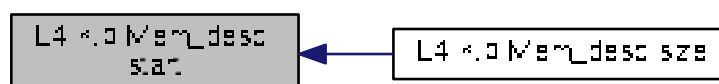
Returns

Start address of memory descriptor

Definition at line 177 of file [kip](#).

Referenced by [size\(\)](#).

Here is the caller graph for this function:



14.140.4.11 sub_type()

```
unsigned char L4::Kip::Mem_desc::sub_type ( ) const throw ( ) [inline]
```

Return sub-type of the memory descriptor.

Returns

Sub-type of the memory descriptor

Definition at line 205 of file [kip](#).

14.140.4.12 type()

```
Mem_type L4::Kip::Mem_desc::type ( ) const throw ( ) [inline]
```

Return type of the memory descriptor.

Returns

Type of the memory descriptor

Definition at line 198 of file [kip](#).

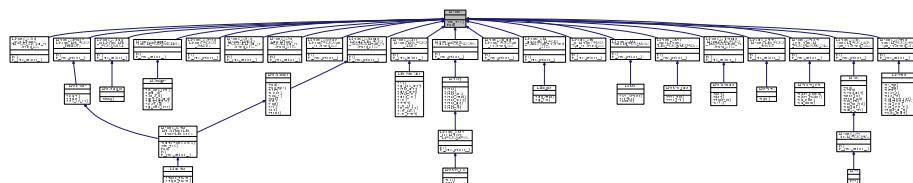
The documentation for this class was generated from the following file:

- [l4/sys/kip](#)

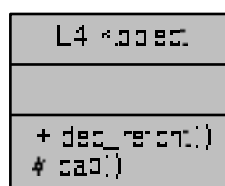
14.141 L4::Kobject Class Reference

Base class for all kinds of kernel objects and remote objects, referenced by capabilities.

Inheritance diagram for L4::Kobject:



Collaboration diagram for L4::Kobject:



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 throw ()

Return capability selector.

14.141.1 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 46 of file [kobject](#).

14.141.2 Member Function Documentation

14.141.2.1 cap()

```
l4\_cap\_idx\_t L4::Kobject::cap ( ) const throw ()    [inline], [protected]
```

Return capability selector.

Returns

Capability selector.

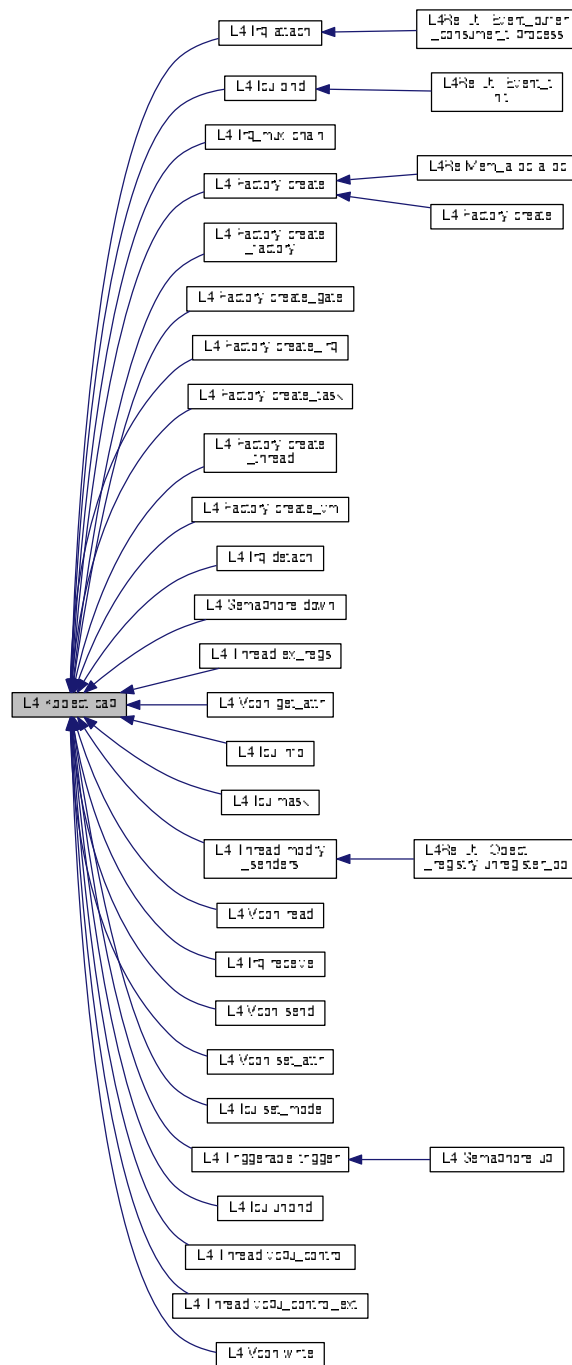
This method is for derived classes to gain access to the actual capability selector.

Definition at line 79 of file [kobject](#).

References [L4_CAP_MASK](#).

Referenced by [L4::Irq::attach\(\)](#), [L4::Icu::bind\(\)](#), [L4::Irq_mux::chain\(\)](#), [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_vm\(\)](#), [L4::Irq::detach\(\)](#), [L4::Semaphore::down\(\)](#), [L4::Thread::ex_regs\(\)](#), [L4::Vcon↵::get_attr\(\)](#), [L4::Icu::info\(\)](#), [L4::Icu::mask\(\)](#), [L4::Thread::modify_senders\(\)](#), [L4::Vcon::read\(\)](#), [L4::Irq::receive\(\)](#), [L4↵::Vcon::send\(\)](#), [L4::Vcon::set_attr\(\)](#), [L4::Icu::set_mode\(\)](#), [L4::Triggerable::trigger\(\)](#), [L4::Icu::unbind\(\)](#), [L4::Thread↵::vcpu_control\(\)](#), [L4::Thread::vcpu_control_ext\(\)](#), and [L4::Vcon::write\(\)](#).

Here is the caller graph for this function:



14.141.2.2 dec_refcnt()

```

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, usually the UTCB of the calling thread.

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.

Definition at line 104 of file [kobject](#).

The documentation for this class was generated from the following file:

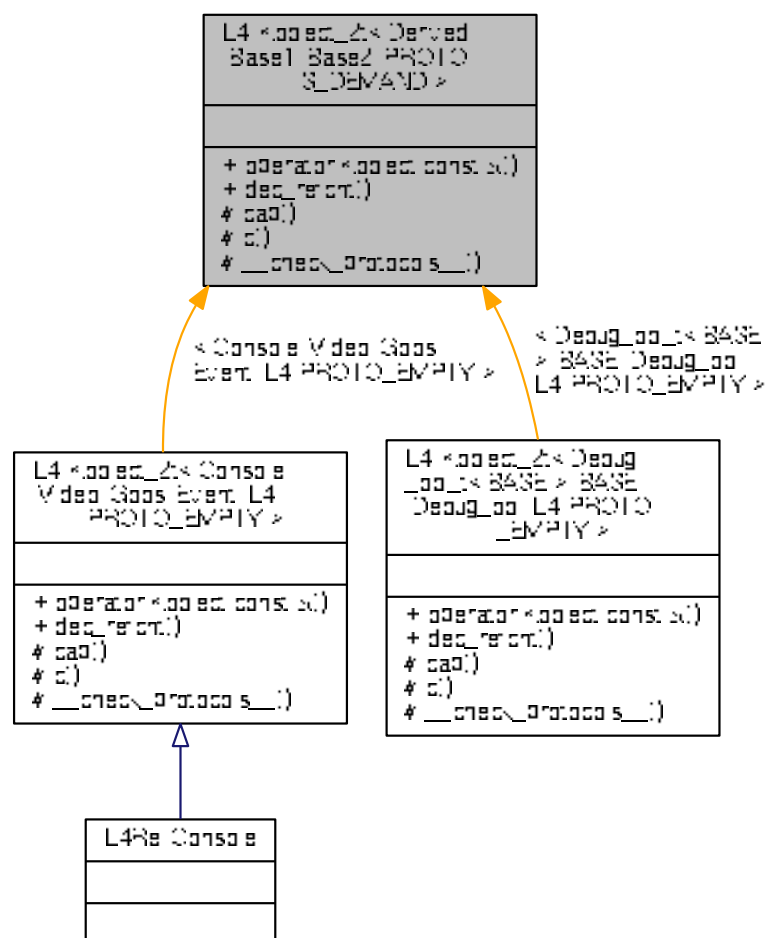
- l4/sys/kobject

14.142 L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND > Class Template Reference

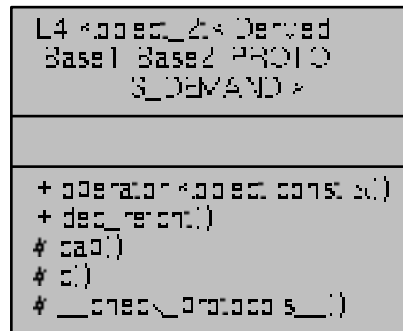
Helper class to create an [L4Re](#) interface class that is derived from two base classes (see [L4::Kobject_t](#)).

```
#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_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
Get the capability to ourselves.

Static Protected Member Functions

- static void [__check_protocols__](#) ()

14.142.1 Detailed Description

```

template<typename Derived, typename Base1, typename Base2, long PROTO = PROTO_ANY, typename 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 [829](#) of file [__typeinfo.h](#).

14.142.2 Member Typedef Documentation

14.142.2.1 __iface

```
template<typename Derived, typename Base1, typename Base2, long 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 [835](#) of file [__typeinfo.h](#).

14.142.2.2 __iface_list

```
template<typename Derived, typename Base1, typename Base2, long 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, PRO↵
TO, S_DEMAND >::__Iface_list [protected]
```

The list of all RPC interfaces provided directly or through inheritance.

Definition at line [843](#) of file [__typeinfo.h](#).

14.142.2.3 Class

```
template<typename Derived, typename Base1, typename Base2, long 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 833 of file [__typeinfo.h](#).

14.142.3 Member Function Documentation

14.142.3.1 __check_protocols__()

```
template<typename Derived, typename Base1, typename Base2, long PROTO = PROTO_ANY, typename
S_DEMAND = Type_info::Demand_t<>>
static void L4::Kobject_2t< Derived, Base1, Base2, PROTO, S_DEMAND >::__check_protocols__ ( )
[inline], [static], [protected]
```

Definition at line 846 of file [__typeinfo.h](#).

14.142.3.2 c()

```
template<typename Derived, typename Base1, typename Base2, long 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]
```

Get the capability to ourselves.

Definition at line 865 of file [__typeinfo.h](#).

The documentation for this class was generated from the following file:

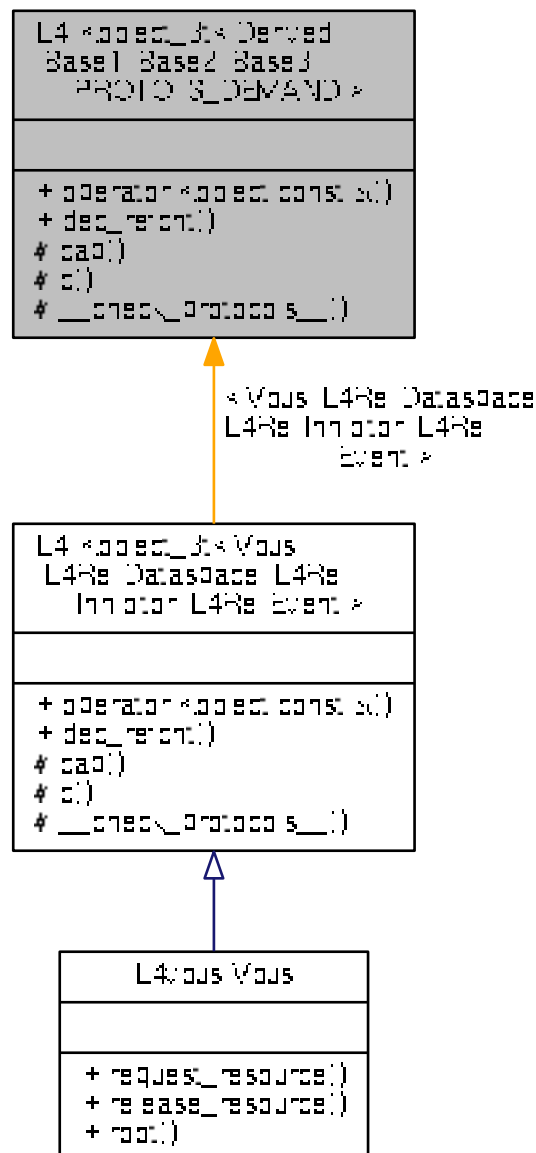
- [l4/sys/__typeinfo.h](#)

14.143 L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND > Struct Template Reference

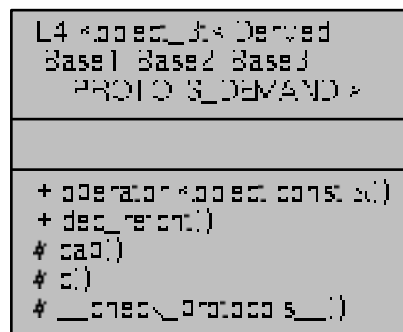
Helper class to create an [L4Re](#) interface class that is derived from three base classes (see [L4::Kobject_t](#)).

```
#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
Get the capability to ourselves.

Static Protected Member Functions

- static void [__check_protocols__](#) ()

14.143.1 Detailed Description

```

template<typename Derived, typename Base1, typename Base2, typename Base3, long 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 929 of file [__typeinfo.h](#).

14.143.2 Member Typedef Documentation

14.143.2.1 [__iface](#)

```
template<typename Derived, typename Base1, typename Base2, typename Base3, long PROTO = PROTO_↵
O_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 935 of file [__typeinfo.h](#).

14.143.2.2 [__iface_list](#)

```
template<typename Derived, typename Base1, typename Base2, typename Base3, long PROTO = PROTO_↵
O_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 946 of file [__typeinfo.h](#).

14.143.2.3 Class

```
template<typename Derived, typename Base1, typename Base2, typename Base3, long PROTO = PROTO_↵
O_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 933 of file [__typeinfo.h](#).

14.143.3 Member Function Documentation

14.143.3.1 __check_protocols__()

```
template<typename Derived, typename Base1, typename Base2, typename Base3, long PROTO = PROTO_↵
O_ANY, typename S_DEMAND = Type_info::Demand_t<>>
static void L4::Kobject_3t< Derived, Base1, Base2, Base3, PROTO, S_DEMAND >::__check_protocols_↵
__ ( ) [inline], [static], [protected]
```

Definition at line 949 of file [__typeinfo.h](#).

14.143.3.2 c()

```
template<typename Derived, typename Base1, typename Base2, typename Base3, long PROTO = PROTO_↵
O_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]
```

Get the capability to ourselves.

Definition at line 977 of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

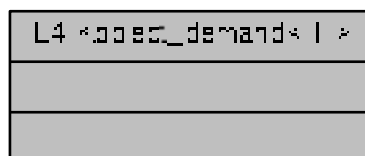
- [l4/sys/__typeinfo.h](#)

14.144 L4::Kobject_demand< T > Struct Template Reference

Get the combined server-side resource requirements for all type T...

```
#include <l4/sys/capability>
```

Collaboration diagram for L4::Kobject_demand< T >:



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
Get the capability to ourselves.

Static Protected Member Functions

- static void [__check_protocols__](#) ()
Helper to check for protocol conflicts.

14.145.1 Detailed Description

```
template<typename Derived, typename Base, long 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>
{
    ...
};
```

Definition at line [753](#) of file [__typeinfo.h](#).

The documentation for this class was generated from the following file:

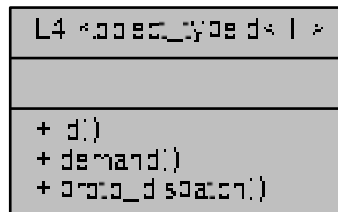
- [l4/sys/__typeinfo.h](#)

14.146 L4::Kobject_typeid< T > Struct Template Reference

[Meta](#) object for handling access to type information of Kobjects.

```
#include <__typeinfo.h>
```

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](#) ()
Get a pointer to the [Kobject](#) type information of T.
- static [Type_info::Demand](#) [demand](#) ()
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, long proto, A1 a1, A2 &a2)
Protocol based server-side dispatch function.

14.146.1 Detailed Description

```
template<typename T>
struct L4::Kobject_typeid< T >
```

[Meta](#) object for handling access to type information of Kobjects.

Template Parameters

T	The data type derived from Kobject , usually using Kobject_t .
-------------------	--

Definition at line 620 of file [__typeinfo.h](#).

14.146.2 Member Typedef Documentation

14.146.2.1 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 632 of file [__typeinfo.h](#).

14.146.3 Member Function Documentation

14.146.3.1 demand()

```
template<typename T>
static Type_info::Demand L4::Kobject_typeid< T >::demand ( ) [inline], [static]
```

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 649 of file [__typeinfo.h](#).

14.146.3.2 id()

```
template<typename T>
static Type\_info const* L4::Kobject\_typeid< T >::id ( ) [inline], [static]
```

Get a pointer to the [Kobject](#) type information of T.

Returns

a pointer to the [Kobject](#) typeinfo of T.

Definition at line 640 of file [__typeinfo.h](#).

Referenced by [L4::kobject_typeid\(\)](#).

Here is the caller graph for this function:



14.146.3.3 proto_dispatch()

```
template<typename T>
template<typename THIS , typename A1 , typename A2 >
static int L4::Kobject\_typeid< T >::proto_dispatch (
    THIS * self,
    long proto,
    A1 a1,
    A2 & a2 ) [inline], [static]
```

Protocol based server-side dispatch function.

Template Parameters

$T \leftrightarrow$ <i>HIS</i>	Data type of the server-side object implementing the interface T.
<i>A1</i>	Data type of second argument for <code>p_dispatch()</code>
<i>A2</i>	Data type of third argument for <code>p_dispatch()</code>

Parameters

<i>self</i>	The pointer to the server object
<i>proto</i>	The protocol number used by the caller

Parameters

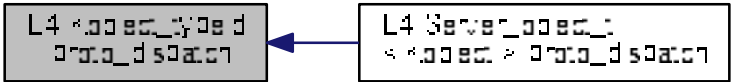
a1	The second argument passed to self->p_dispatch()
a2	The third argument passed to self->p_dispatch()

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 670 of file [__typeinfo.h](#).

Referenced by [L4::Server_object_t< Kobject >::proto_dispatch\(\)](#).

Here is the caller graph for this function:



The documentation for this struct was generated from the following file:

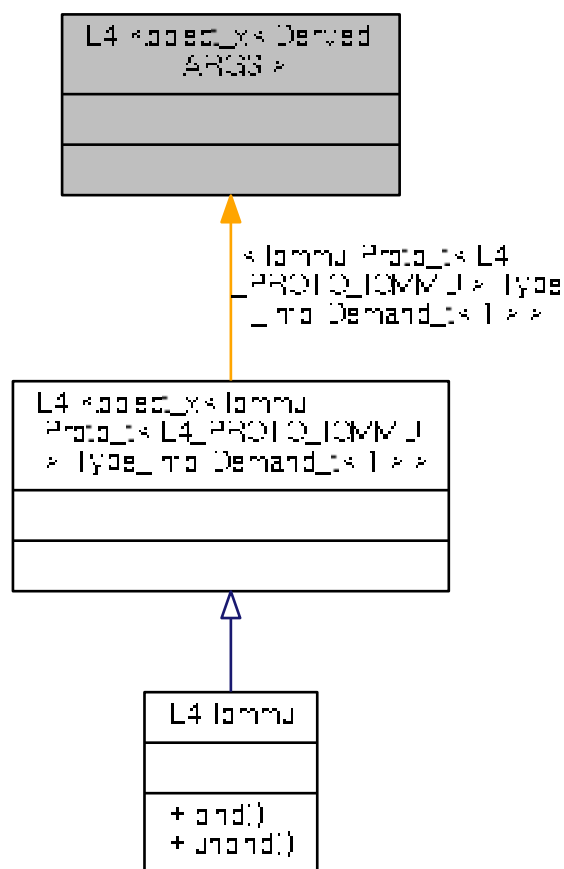
- [l4/sys/__typeinfo.h](#)

14.147 L4::Kobject_x< Derived, ARGS > Struct Template Reference

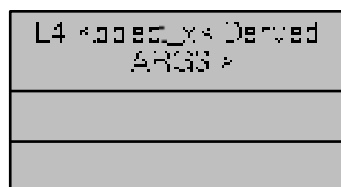
Generic [Kobject](#) inheritance template.

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Kobject_x< Derived, ARGS >:



Collaboration diagram for L4::Kobject_x< Derived, ARGS >:



14.147.1 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 [1189](#) of file [__typeinfo.h](#).

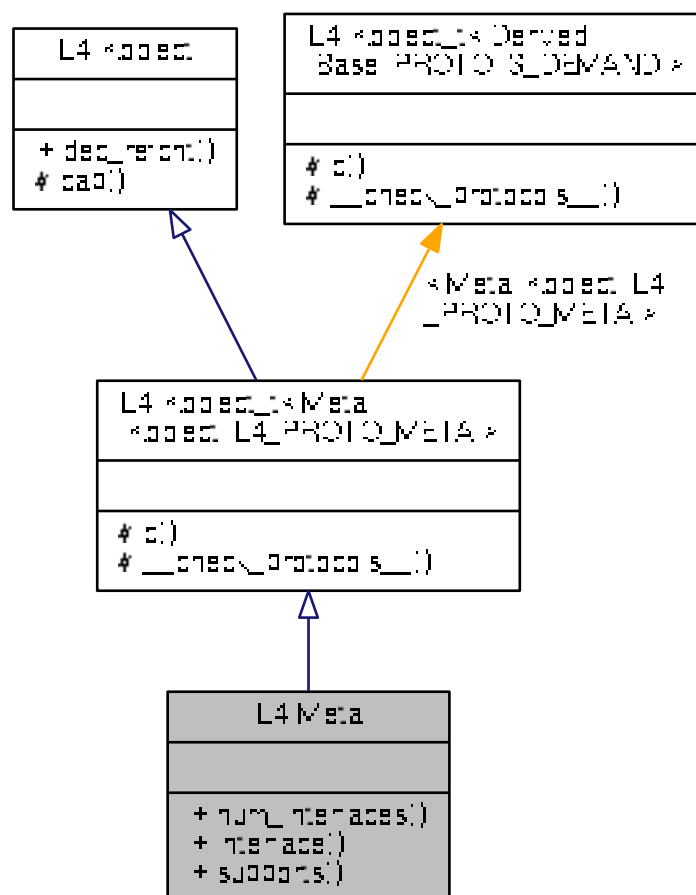
The documentation for this struct was generated from the following file:

- [l4/sys/__typeinfo.h](#)

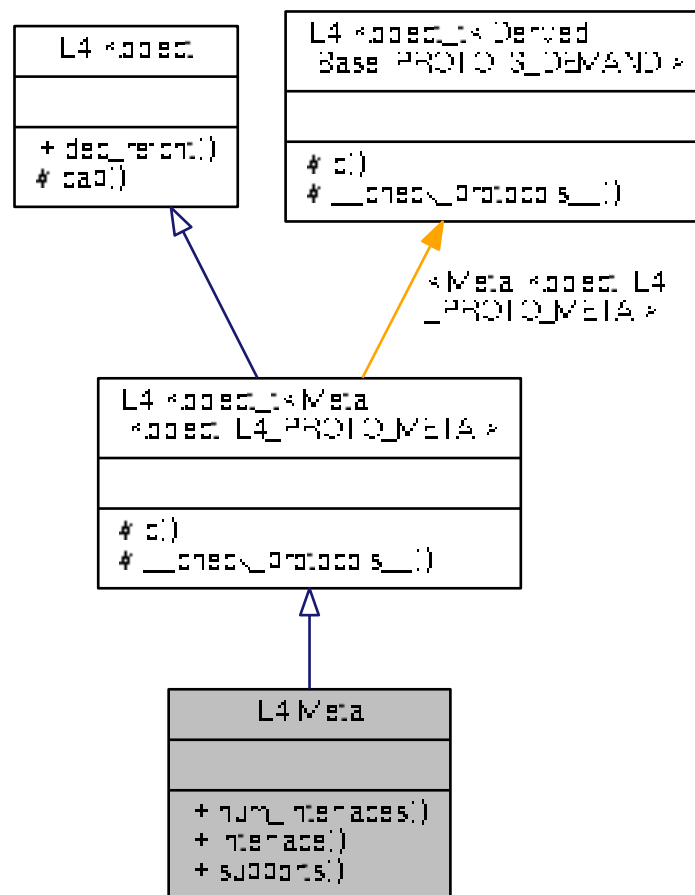
14.148 L4::Meta Class Reference

[Meta](#) interface that shall be implemented by each [L4Re](#) object and gives access to the dynamic type information for [L4Re](#) objects.

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).

Additional Inherited Members

14.148.1 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 37 of file [meta](#).

14.148.2 Member Function Documentation

14.148.2.1 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 num_interfaces() .
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

l4_msgtag_t::label()	≥ 0 Successful; see <code>proto</code> and <code>name</code> .
l4_msgtag_t::label()	< 0 Error code.

14.148.2.2 num_interfaces()

```
l4_msgtag_t L4::Meta::num_interfaces ( )
```

Get the number of interfaces implemented by this object.

Return values

l4_msgtag_t::label()	≥ 0 The number of supported interfaces.
l4_msgtag_t::label()	< 0 Error code of the occurred error.

14.148.2.3 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

l4_msgtag_t::label()	== 1 protocol is supported.
l4_msgtag_t::label()	== 0 protocol is not supported.

This method is intended to be used for statically assigned protocol numbers.

Referenced by [L4::cap_dynamic_cast\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

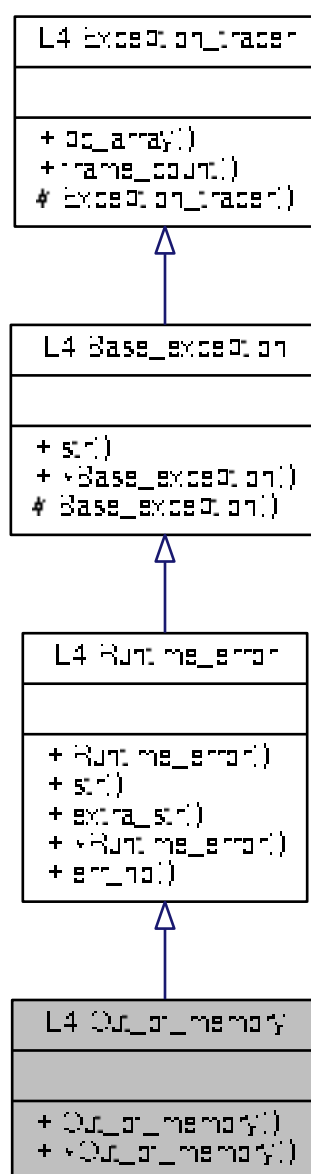
- [l4/sys/meta](#)

14.149 L4::Out_of_memory Class Reference

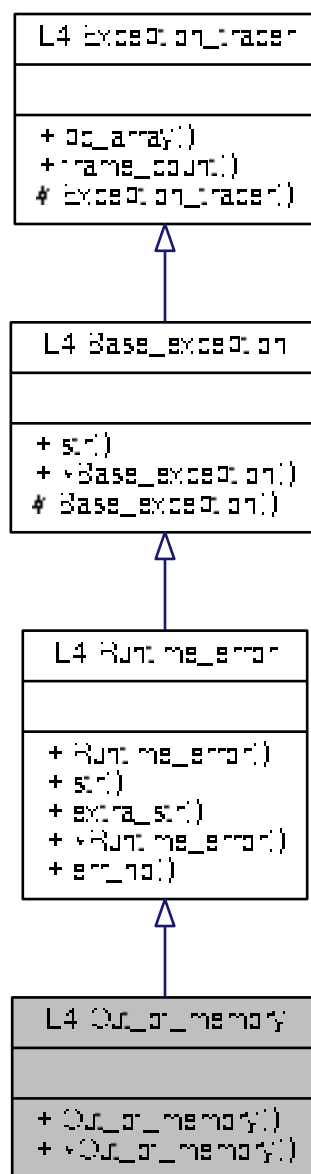
Exception signalling insufficient memory.

```
#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="") throw ()

Create an out-of-memory exception.

- [~Out_of_memory](#) () throw ()

Destruction.

Additional Inherited Members

14.149.1 Detailed Description

Exception signalling insufficient memory.

Definition at line [188](#) of file [exceptions](#).

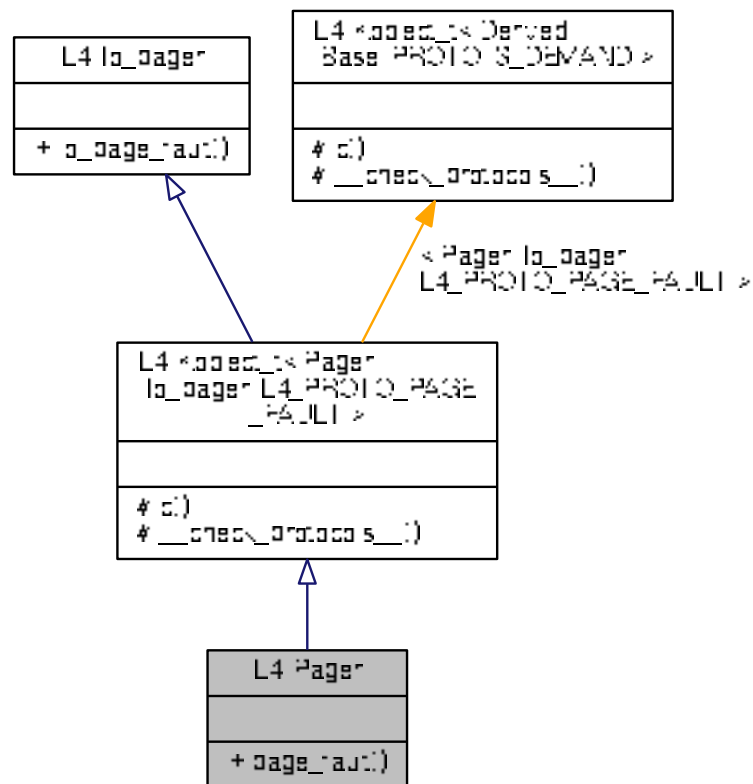
The documentation for this class was generated from the following file:

- [l4/cxx/exceptions](#)

14.150 L4::Pager Class Reference

[Pager](#) interface including the [lo_pager](#) interface.

Collaboration diagram for L4::Pager:



Public Member Functions

- `l4_msgtag_t page_fault (l4_umword_t pfa, l4_umword_t pc, L4::lpc::Opt< l4_mword_t > result, L4::lpc::Rcv_fpage rwin, L4::lpc::Opt< L4::lpc::Snd_fpage > fp)`
Page-fault protocol message.

Additional Inherited Members

14.150.1 Detailed Description

`Pager` interface including the `lo_pager` interface.

Definition at line 67 of file `pager`.

14.150.2 Member Function Documentation

14.150.2.1 `page_fault()`

```
l4_msgtag_t L4::Pager::page_fault (
    l4_umword_t pfa,
    l4_umword_t pc,
    L4::Ipc::Opt< l4_mword_t > result,
    L4::Ipc::Rcv_fpage rwin,
    L4::Ipc::Opt< L4::Ipc::Snd_fpage > fp )
```

Page-fault protocol message.

Parameters

	<i>pfa</i>	Faulting address including failure reason: bits [0:2]
	<i>pc</i>	Faulting program counter.
out	<i>result</i>	Optional: handling result value.
	<i>rwin</i>	Receive window for a flex-page mapping resolving the page fault
out	<i>fp</i>	Optional: flex-page 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 that fills in *result* and / or *fp* for the reply.

pfa encoding is as shown:

[63/31 .. 3]	2	1	0
PFA	X	W	T

- **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* is set to 1 for a page fault during instruction fetch.
- **W** Bit 1 of *pfa* is set to 1 for a page fault due to a write operation.
- **T** Bit 0 of *pfa* is set for translation faults (no mapping was present).

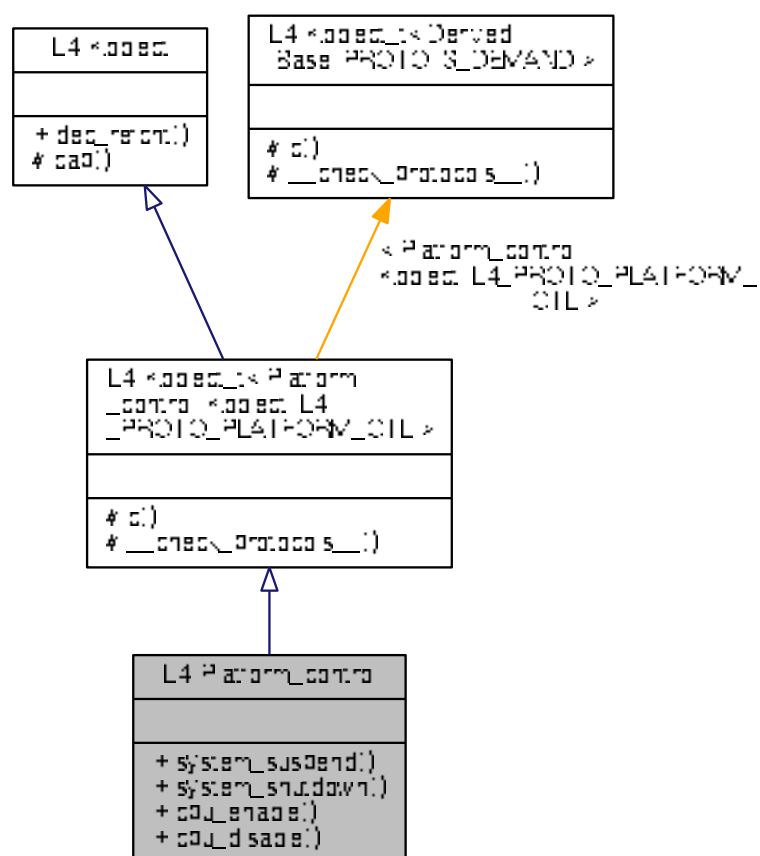
The documentation for this class was generated from the following file:

- [l4/sys/pager](#)

14.151 L4::Platform_control Class Reference

[L4](#) C++ interface for controlling platform-wide properties.

Inheritance diagram for L4::Platform_control:



- Opcodes for platform-control object.*

Additional Inherited Members

14.151.1 Detailed Description

[L4](#) C++ interface for controlling platform-wide properties.

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 [46](#) of file [platform_control](#).

14.151.2 Member Enumeration Documentation

14.151.2.1 Opcode

```
enum L4::Platform\_control::Opcode
```

Opcodes for platform-control object.

Enumerator

Suspend	Opcode for suspend to RAM.
Shutdown	Opcode for shutdown / reboot.
Cpu_enable	Opcode to enable a CPU.
Cpu_disable	Opcode to disable a CPU.

Definition at line [51](#) of file [platform_control](#).

14.151.3 Member Function Documentation

14.151.3.1 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

14.151.3.2 `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

14.151.3.3 `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
---------------	-------------------------------

14.151.3.4 `system_suspend()`

```
l4_msgtag_t L4::Platform_control::system_suspend (
    l4_umword_t extras )
```

Enter suspend to RAM.

Parameters

<i>extras</i>	some extra platform-specific information needed to enter suspend to RAM.
---------------	--

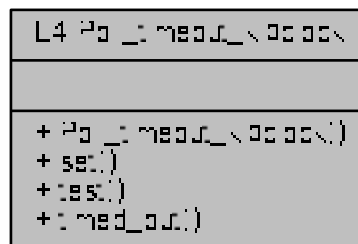
The documentation for this class was generated from the following file:

- [l4/sys/platform_control](#)

14.152 L4::Poll_timeout_kipclock Class Reference

A polling timeout based on the [L4Re](#) clock.

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.

14.152.1 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 37 of file [poll_timeout_kipclock](#).

14.152.2 Constructor & Destructor Documentation

14.152.2.1 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 44 of file [poll_timeout_kipclock](#).

14.152.3 Member Function Documentation

14.152.3.1 set()

```
void L4::Poll_timeout_kipclock::set (
    unsigned poll_time_us ) [inline]
```

(Re-)Set relative timeout in microseconds

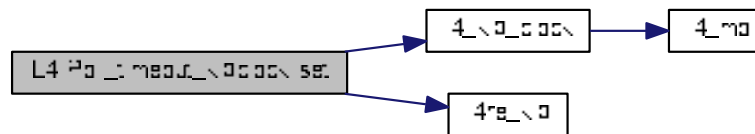
Parameters

<i>poll_time_us</i>	Polling timeout in microseconds.
---------------------	----------------------------------

Definition at line 53 of file [poll_timeout_kipclock](#).

References [l4_kip_clock\(\)](#), and [l4re_kip\(\)](#).

Here is the call graph for this function:



14.152.3.2 test()

```
bool L4::Poll_timeout_kipclock::test (
    bool expression = true ) [inline]
```

Test whether timeout has expired.

Parameters

<i>expression</i>	Optional expression.
-------------------	----------------------

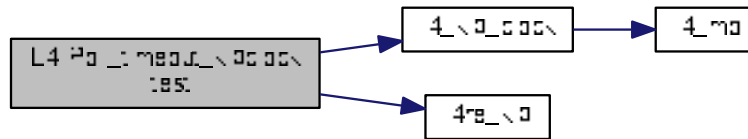
Return values

<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 67 of file [poll_timeout_kipclock](#).

References [l4_kip_clock\(\)](#), and [l4re_kip\(\)](#).

Here is the call graph for this function:



14.152.3.3 timed_out()

```
bool L4::Poll_timeout_kipclock::timed_out ( ) const [inline]
```

Query whether timeout has expired.

Returns

Expiry state of timeout

Definition at line 79 of file [poll_timeout_kipclock](#).

The documentation for this class was generated from the following file:

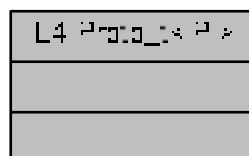
- `l4/re/util/poll_timeout_kipclock`

14.153 L4::Proto_t< P > Struct Template Reference

Data type for defining protocol numbers.

```
#include <__typeinfo.h>
```

Collaboration diagram for L4::Proto_t< P >:



14.153.1 Detailed Description

```
template<long 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 1173 of file [__typeinfo.h](#).

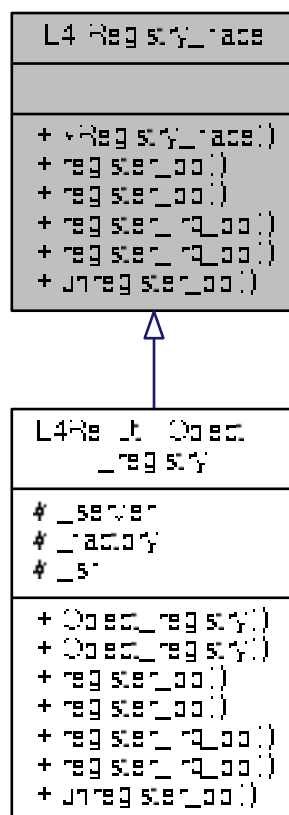
The documentation for this struct was generated from the following file:

- [l4/sys/__typeinfo.h](#)

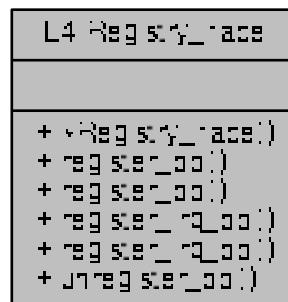
14.154 L4::Registry_iface Class Reference

Abstract interface for object registries.

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 o as server-side object 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::Irq > register_irq_obj](#) (L4::Epiface *o, [L4::Cap< L4::Irq > const &irq](#))=0
Register o as server-side object for asynchronous IRQs.
- virtual void [unregister_obj](#) (L4::Epiface *o, bool unmap=true)=0
Unregister the given object o from the server.

14.154.1 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 285 of file [ipc_epiface](#).

14.154.2 Member Function Documentation

14.154.2.1 [register_irq_obj\(\)](#) [1/2]

```
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

<i>o</i>	Pointer to an Epiface object that shall be registered as server-side object for IRQs.
----------	---

Return values

<i>L4::Cap<L4::Irq></i>	Capability to a new IRQ object on success.
<i>L4::Cap<L4::Irq>::Invalid</i>	The allocation of the IRQ has failed.

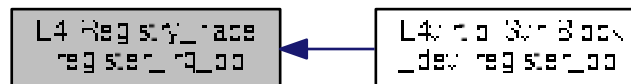
After successful registration `o->obj_cap()` will be the capability of the allocated IRQ object.

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`.

Referenced by `L4virtio::Svr::Block_dev<Ds_data>::register_obj()`.

Here is the caller graph for this function:



14.154.2.2 register_irq_obj() [2/2]

```
virtual L4::Cap<L4::Irq> L4::Registry_iface::register_irq_obj (
    L4::Epiface * o,
    L4::Cap< L4::Irq > const & irq ) [pure virtual]
```

Register `o` as server-side object for asynchronous IRQs.

Parameters

<i>o</i>	Pointer to an Epiface object that shall be registered as server-side object for IRQs.
<i>irq</i>	Capability to an already allocated IRQ object where <code>o</code> shall be attached as server-side handler.

Return values

<i>L4::Cap<L4::Irq></i>	Capability to the given IRQ object on success.
<i>L4::Cap<L4::Irq>::Invalid</i>	The IRQ attach operation has failed.

After successful registration `o->obj_cap()` will be equal to `irq`.

Implemented in [L4Re::Util::Object_registry](#).

14.154.2.3 `register_obj()` [1/2]

```
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 <code>Epiface</code> object that shall be registered.
<i>service</i>	Name of the capability that shall be used to connect <code>o</code> to as a server-side object.

Return values

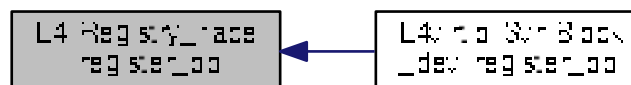
<code>L4::Cap<void></code>	The capability known as <code>service</code> on success.
<code>L4::Cap<void>::Invalid</code>	No capability with the given name found.

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](#).

Referenced by [L4virtio::Svr::Block_dev<Ds_data>::register_obj\(\)](#).

Here is the caller graph for this function:



14.154.2.4 `register_obj()` [2/2]

```
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

<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.

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`.

14.154.2.5 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 <code>Registry_iface</code> .
<i>unmap</i>	If true the capability <code>o->obj_cap()</code> 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:

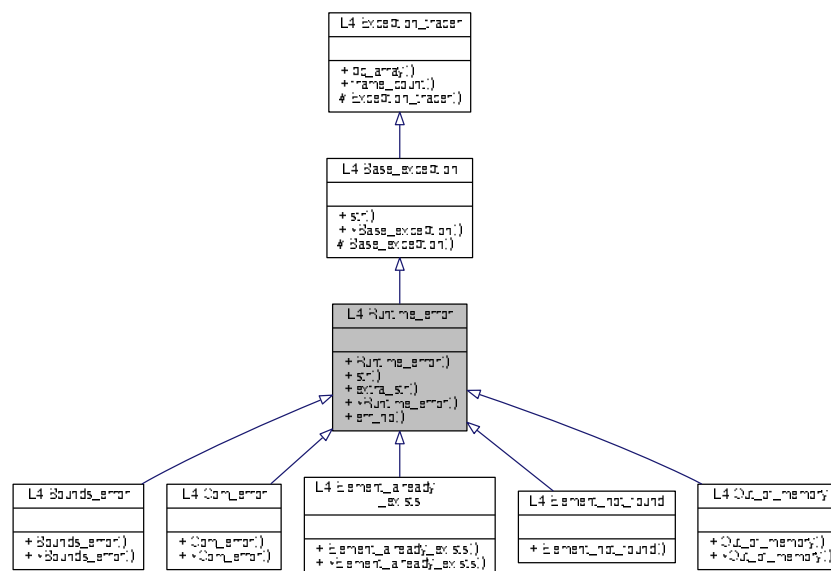
- `l4/sys/cxx/ipc_epiface`

14.155 L4::Runtime_error Class Reference

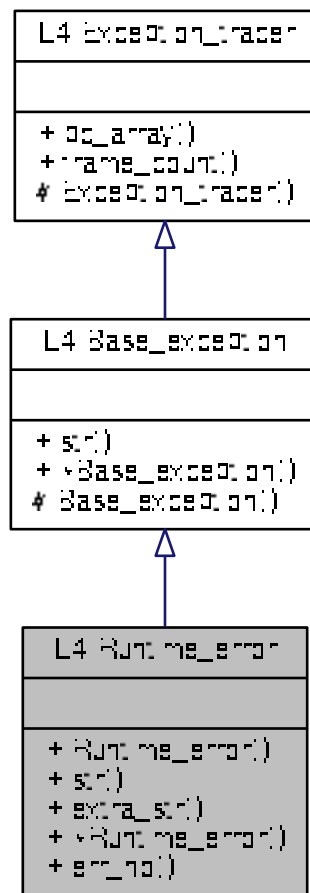
Exception for an abstract runtime error.

```
#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) throw ()
Create a new [Runtime_error](#).
- char const * [str](#) () const throw ()
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 throw ()
Get the error value for this runtime error.

Additional Inherited Members

14.155.1 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 139 of file [exceptions](#).

14.155.2 Constructor & Destructor Documentation

14.155.2.1 Runtime_error()

```
L4::Runtime_error::Runtime_error (
    long err_no,
    char const * extra = 0 ) throw ()    [inline], [explicit]
```

Create a new [Runtime_error](#).

Parameters

<i>err_no</i>	Error value for this runtime error.
<i>extra</i>	Description of what was happening while the error occurred.

Definition at line [152](#) of file [exceptions](#).

14.155.3 Member Function Documentation

14.155.3.1 err_no()

```
long L4::Runtime_error::err_no ( ) const throw ()    [inline]
```

Get the error value for this runtime error.

Returns

Error value.

Definition at line [181](#) of file [exceptions](#).

14.155.3.2 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 [173](#) of file [exceptions](#).

The documentation for this class was generated from the following file:

- [l4/cxx/exceptions](#)

C++ interface of the [Scheduler](#) kernel object.

[illegible]

- `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 throw ()`
Query if a CPU is online.

Additional Inherited Members

14.156.1 Detailed Description

C++ interface of the [Scheduler](#) kernel object.

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.

Include File

```
#include <l4/sys/scheduler>
```

Definition at line 42 of file [scheduler](#).

14.156.2 Member Function Documentation

14.156.2.1 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.

14.156.2.2 info()

```
l4_msgtag_t L4::Scheduler::info (
    l4_umword_t * cpu_max,
    l4_sched_cpu_set_t * cpus,
    l4_utcb_t * utcb = l4_utcb() ) const throw()    [inline]
```

Get scheduler information.

Parameters

out	<i>cpu_max</i>	Maximum number of CPUs ever available.
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.
	<i>utcb</i>	UTCB pointer of the calling thread. This defaults to the UTCB of the current thread.

Return values

0	Success.
<code>-L4_ERANGE</code>	The given CPU offset is larger than the maximum number of CPUs.

Definition at line 66 of file [scheduler](#).

References [l4_sched_cpu_set_t::gran_offset](#), [L4_INLINE_RPC_OP](#), [L4_SCHEDULER_IDLE_TIME_OP](#), [L4_SCHEDULER_RUN_THREAD_OP](#), [l4_sched_cpu_set_t::map](#), and [cxx::max\(\)](#).

Here is the call graph for this function:



14.156.2.3 `is_online()`

```
bool L4::Scheduler::is_online (
    l4_umword_t cpu,
    l4_utcb_t * utcb = l4_utcb() ) const throw ()    [inline]
```

Query if a CPU is online.

Parameters

<i>cpu</i>	CPU number whose online status should be queried.
<i>utcb</i>	UTCB pointer 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 139 of file [scheduler](#).

14.156.2.4 `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

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 physical CPUs.

- Two threads with a single identical CPU selected in the CPU set must be scheduled to the same physical CPU.

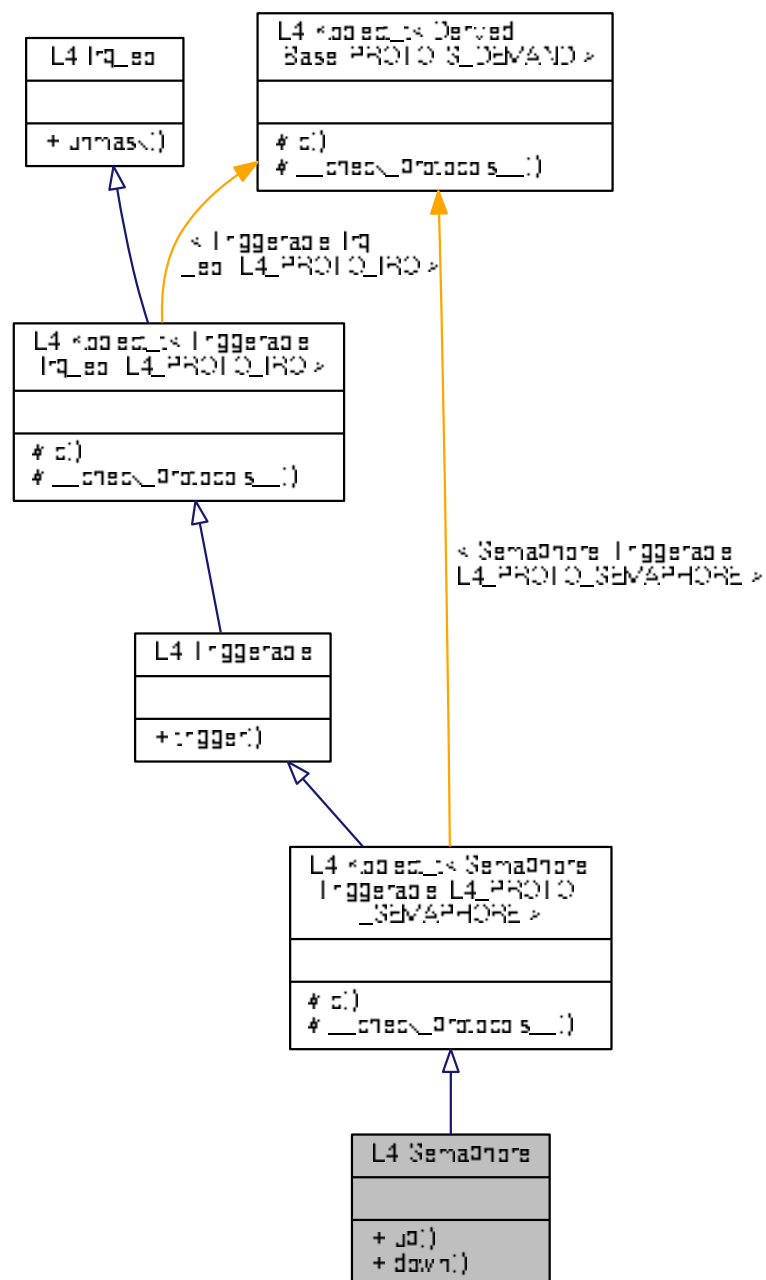
The documentation for this class was generated from the following file:

- [l4/sys/scheduler](#)

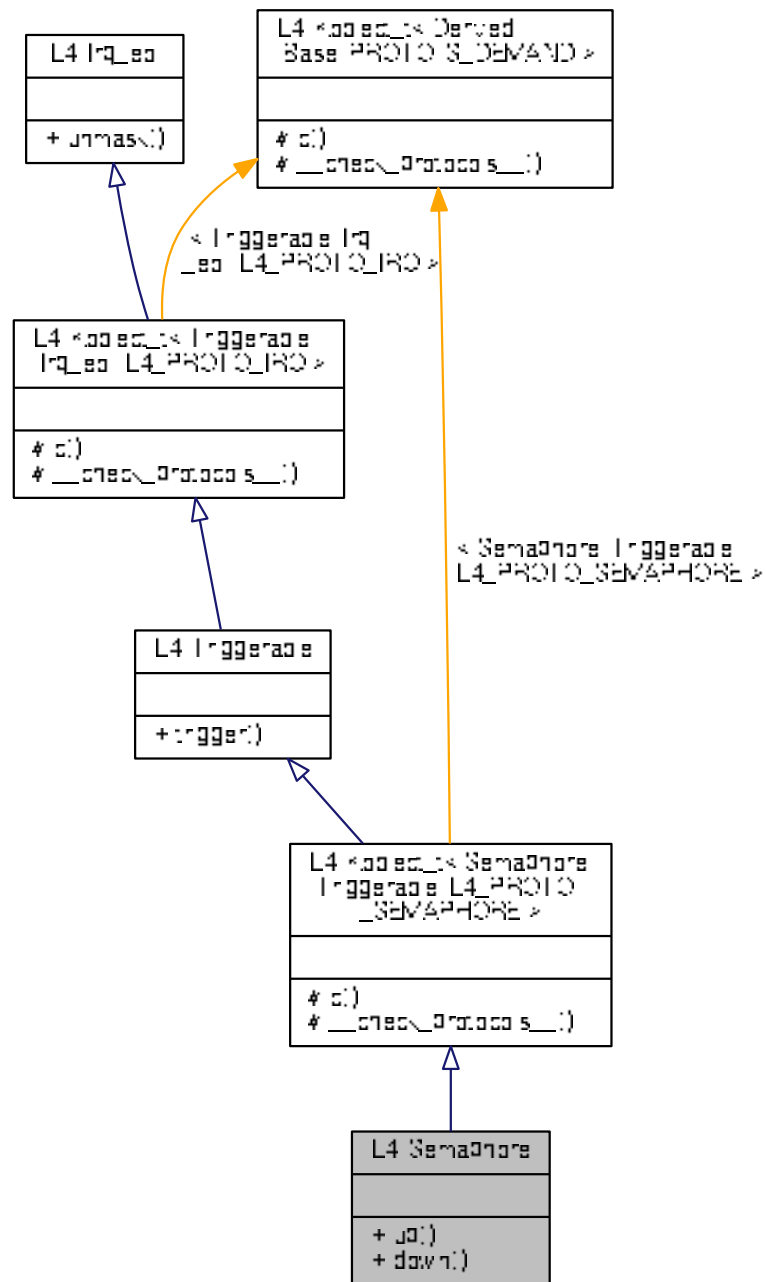
14.157 L4::Semaphore Struct Reference

Kernel-provided semaphore object.

Inheritance diagram for L4::Semaphore:



Collaboration diagram for L4::Semaphore:



Public Member Functions

- `l4_msgtag_t up (l4_utcb_t *utcb=l4_utcb()) throw ()`
Semaphore up operation (wrapper for `trigger()`).
- `l4_msgtag_t down (l4_timeout_t timeout=L4_IPC_NEVER, l4_utcb_t *utcb=l4_utcb()) throw ()`
Semaphore down operation.

Additional Inherited Members

14.157.1 Detailed Description

Kernel-provided semaphore object.

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.

The main reason for using a semaphore instead of an [L4::lrc](#) is to ensure that incoming trigger signals do not interfere with any open-wait operations, as used for example in a server loop.

Definition at line 51 of file [semaphore](#).

14.157.2 Member Function Documentation

14.157.2.1 down()

```
l4_msgtag_t L4::Semaphore::down (
    l4_timeout_t timeout = L4_IPC_NEVER,
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline]
```

[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, usually the UTCB of the calling thread.

Returns

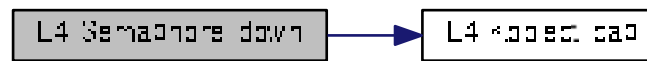
IPC message return tag. Use [l4_ipc_error\(\)](#) to check for a timeout or a cancel condition.

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 84 of file [semaphore](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



14.157.2.2 up()

```

l4_msgtag_t L4::Semaphore::up (
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline]
  
```

Semaphore up operation (wrapper for [trigger\(\)](#)).

Parameters

<i>utcb</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.
-------------	---

Returns

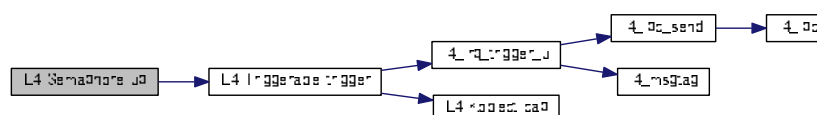
Send-only IPC message return tag. Use [l4_ipc_error\(\)](#) to check for errors, do **not** use [l4_error\(\)](#).

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 65 of file [semaphore](#).

References [L4::Triggerable::trigger\(\)](#).

Here is the call graph for this function:



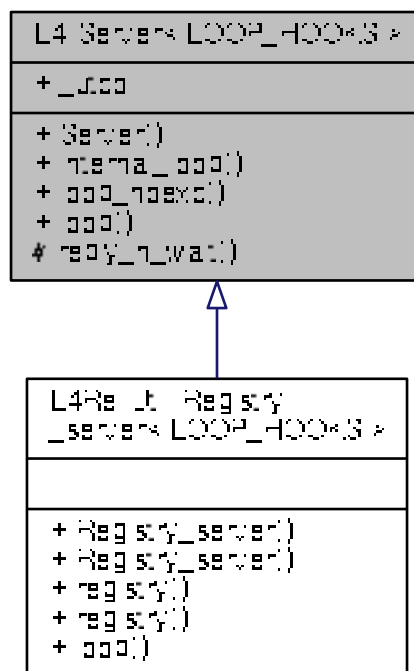
The documentation for this struct was generated from the following file:

- [l4/sys/semaphore](#)

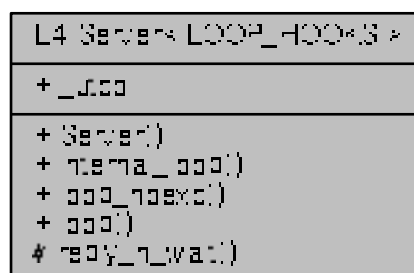
14.158 L4::Server< LOOP_HOOKS > Class Template Reference

Basic server loop for handling client requests.

Inheritance diagram for L4::Server< LOOP_HOOKS >:



Collaboration diagram for L4::Server< LOOP_HOOKS >:



Public Member Functions

- [Server](#) ([l4_utcb_t](#) *utcb)
Initializes the server loop.
- `template<typename DISPATCH >`
`L4_NORETURN void internal_loop (DISPATCH dispatch)`
The server loop.
- `template<typename R >`
`L4_NORETURN void loop_noexc (R r)`
Server loop without exception handling.
- `template<typename EXC , typename R >`
`L4_NORETURN void loop (R r)`
Server loop with internal exception handling.

Protected Member Functions

- `l4_msgtag_t reply_n_wait (l4_msgtag_t reply, l4_umword_t *p)`
Internal implementation for reply and wait.

14.158.1 Detailed Description

```
template<typename LOOP_HOOKS = lpc_svr::Default_loop_hooks>
class L4::Server< LOOP_HOOKS >
```

Basic server loop for handling client requests.

Parameters

<code>LOOP_HOOKS</code>	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 290 of file [ipc_server_loop](#).

14.158.2 Constructor & Destructor Documentation

14.158.2.1 [Server\(\)](#)

```
template<typename LOOP_HOOKS = lpc_svr::Default_loop_hooks>
L4::Server< LOOP_HOOKS >::Server (
    l4\_utcb\_t * utcb ) [inline], [explicit]
```

Initializes the server loop.

Parameters

<i>utcb</i>	The UTCB of the thread running the server loop.
-------------	---

14.158.3.2 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 ) [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 323 of file [ipc_server_loop](#).

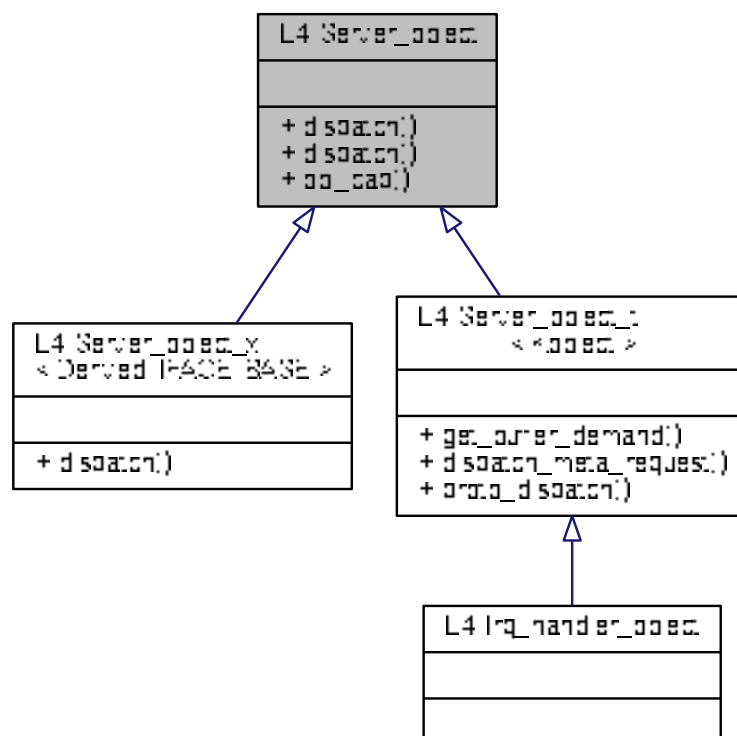
The documentation for this class was generated from the following file:

- [l4/sys/cxx/ipc_server_loop](#)

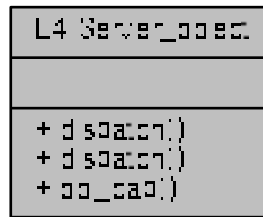
14.159 L4::Server_object Class Reference

Abstract server object to be used with [L4::Server](#) and [L4::Basic_registry](#).

Inheritance diagram for L4::Server_object:



Collaboration diagram for L4::Server_object:



Public Member Functions

- virtual int [dispatch](#) (unsigned long rights, [ipc::Iostream](#) &ios)=0
The abstract handler for client requests to the object.

14.159.1 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 [50](#) of file [ipc_server](#).

14.159.2 Member Function Documentation

14.159.2.1 dispatch()

```
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 lpc::lostream for reading the request and writing the reply.

Return values

<code>-L4_ENOREPLY</code>	No reply message is send.
<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 >](#).

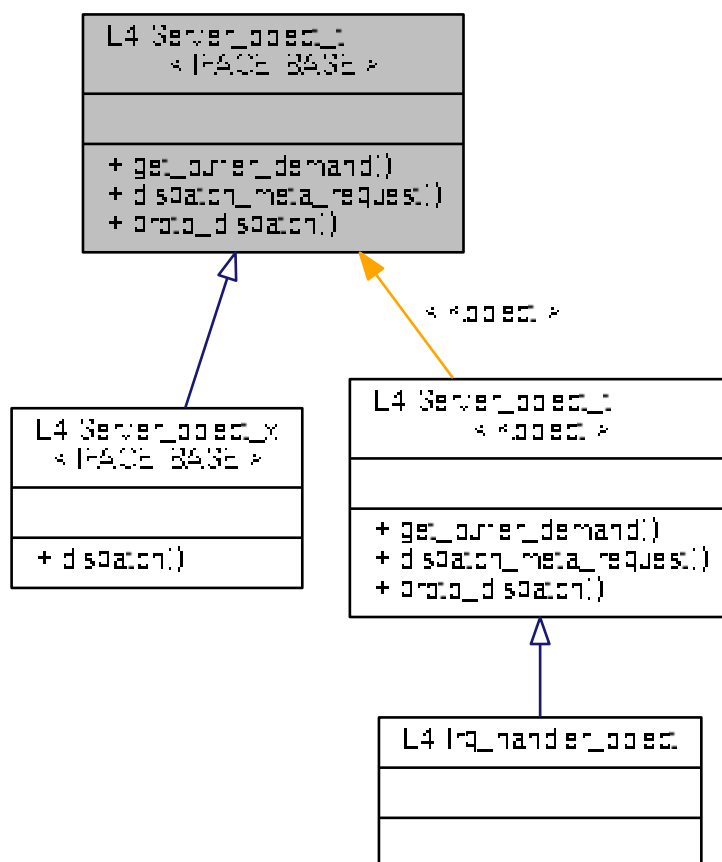
The documentation for this class was generated from the following file:

- [l4/cxx/ipc_server](#)

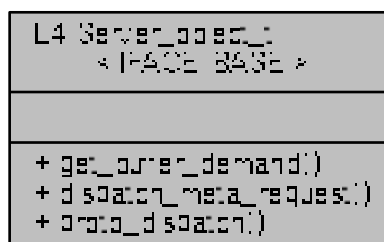
14.160 L4::Server_object_t< IFACE, BASE > Struct Template Reference

Base class (template) for server implementing server objects.

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 Member Functions

- BASE::Demand [get_buffer_demand](#) () const
- int [dispatch_meta_request](#) (L4::lpc::lostream &ios)
Implementation of the meta protocol based on IFACE.

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.

14.160.1 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 92 of file [ipc_server](#).

14.160.2 Member Function Documentation

14.160.2.1 dispatch_meta_request()

```
template<typename IFACE, typename BASE = L4::Server_object>
int L4::Server_object_t< IFACE, BASE >::dispatch_meta_request (
    L4::lpc::lostream & 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 111 of file [ipc_server](#).

14.160.2.2 get_buffer_demand()

```
template<typename IFACE, typename BASE = L4::Server_object>
BASE::Demand L4::Server_object_t< IFACE, BASE >::get_buffer_demand ( ) const [inline]
```

Returns

the server-side buffer demand based in *IFACE*.

Definition at line 98 of file [ipc_server](#).

14.160.2.3 proto_dispatch()

```
template<typename IFACE, typename BASE = L4::Server_object>
template<typename THIS >
static 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

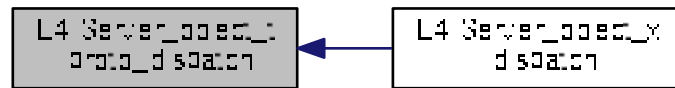
<i>self</i>	The this pointer for teh object (inheritits from Server_object_t).
<i>rights</i>	The rights from the received IPC (forwarded to p_dispatch()).
<i>ios</i>	The message stream for the incoming and the reply message.

[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 126 of file [ipc_server](#).

Referenced by [L4::Server_object_x< Derived, IFACE, BASE >::dispatch\(\)](#).

Here is the caller graph for this function:



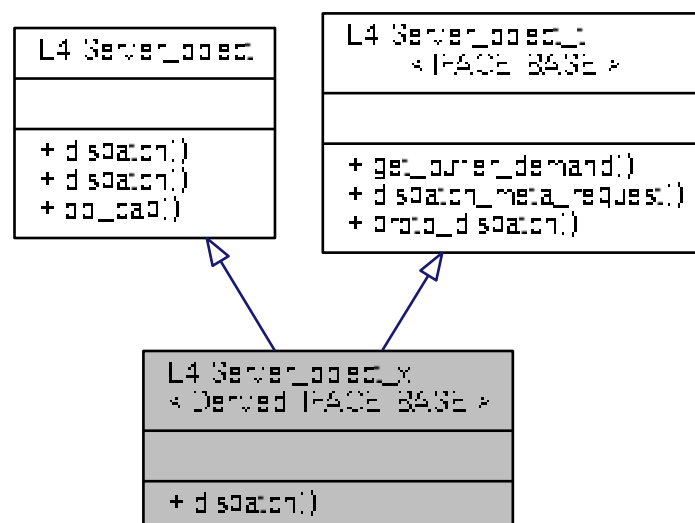
The documentation for this struct was generated from the following file:

- [l4/cxx/ipc_server](#)

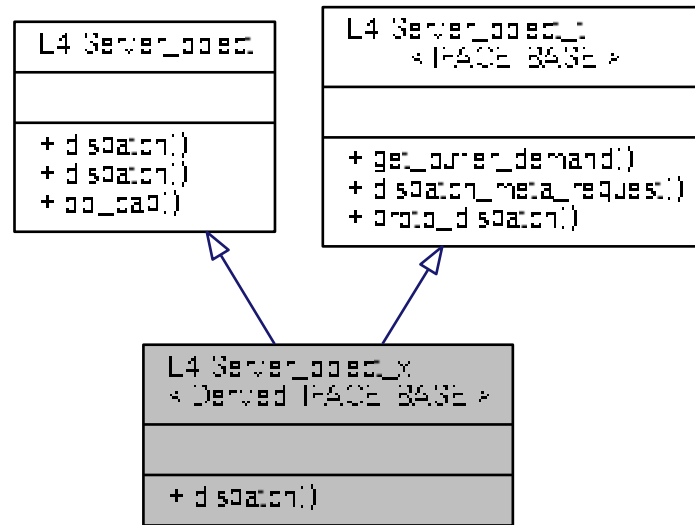
14.161 L4::Server_object_x< Derived, IFACE, BASE > Struct Template Reference

Helper class to implement p_dispatch based server objects.

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::Iostream &ios)`
Implementation forwarding to `p_dispatch()`.

Additional Inherited Members

14.161.1 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::lpc::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 155 of file `ipc_server`.

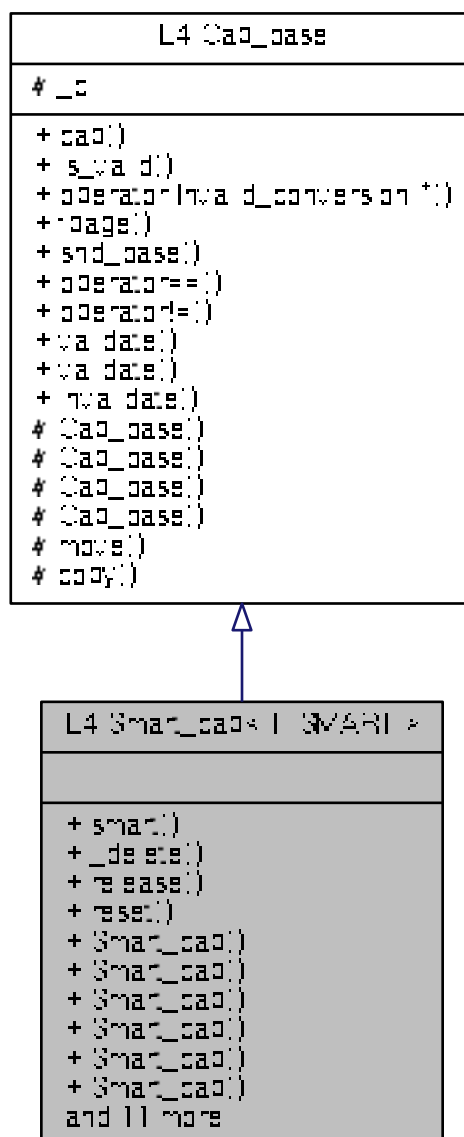
The documentation for this struct was generated from the following file:

- `I4/cxx/ipc_server`

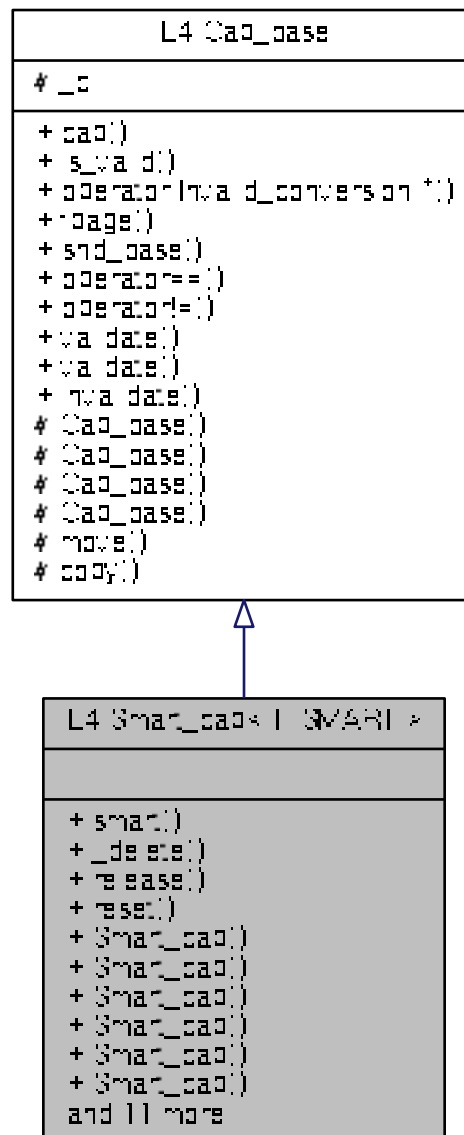
14.162 L4::Smart_cap< T, SMART > Class Template Reference

Smart capability class.

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) throw ()`
Internal constructor, use to generate a capability from a `this` pointer.
- `Cap< T > operator-> () const throw ()`
Member access of a `T`.

Additional Inherited Members

14.162.1 Detailed Description

```
template<typename T, typename SMART>
class L4::Smart_cap< T, SMART >
```

Smart capability class.

Definition at line 36 of file [smart_capability](#).

14.162.2 Constructor & Destructor Documentation

14.162.2.1 Smart_cap()

```
template<typename T, typename SMART>
template<typename O >
L4::Smart_cap< T, SMART >::Smart_cap (
    Cap< O > const & p ) throw ()    [inline]
```

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 objected that is an [L4::Kobject](#). Do never use this constructor for something else!

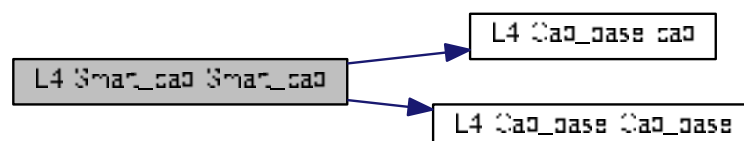
Parameters

<i>p</i>	The <code>this</code> pointer of the Kobject or derived object
----------	--

Definition at line 73 of file [smart_capability](#).

References [L4::Cap_base::_c](#), [L4::Cap_base::cap\(\)](#), and [L4::Cap_base::Cap_base\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

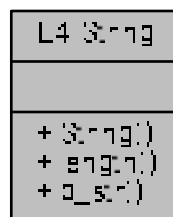
- [l4/sys/smart_capability](#)

14.163 L4::String Class Reference

A null-terminated string container class.

```
#include <string.h>
```

Collaboration diagram for L4::String:



14.163.1 Detailed Description

A null-terminated string container class.

Definition at line 35 of file [string.h](#).

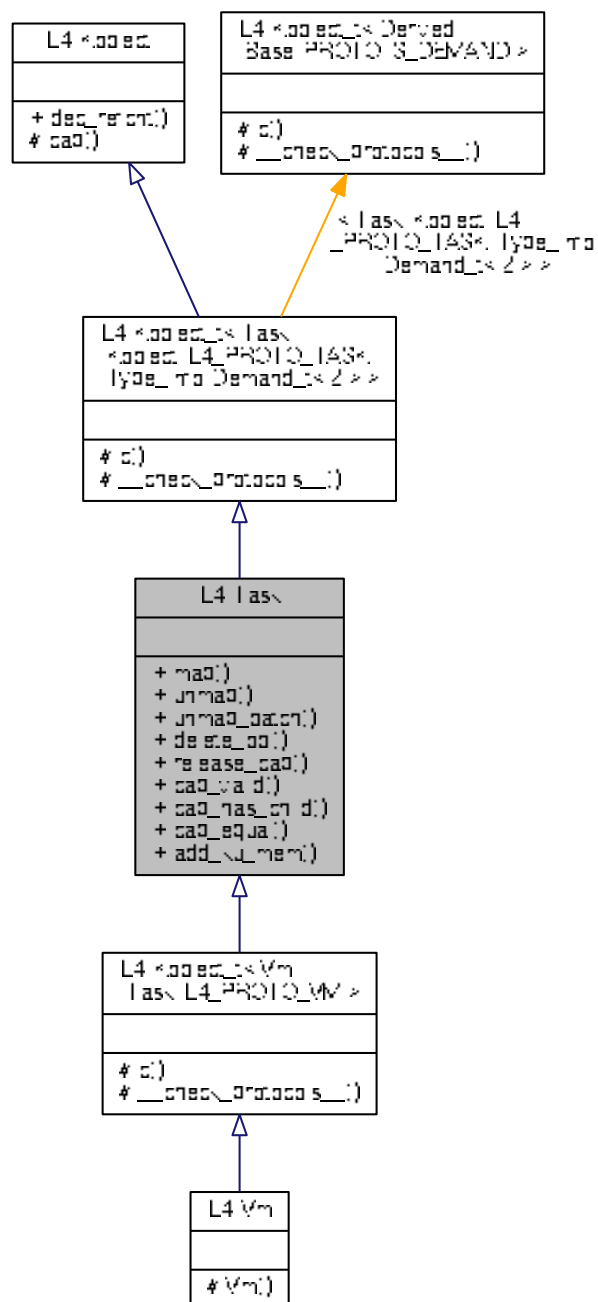
The documentation for this class was generated from the following file:

- [l4/cxx/string.h](#)

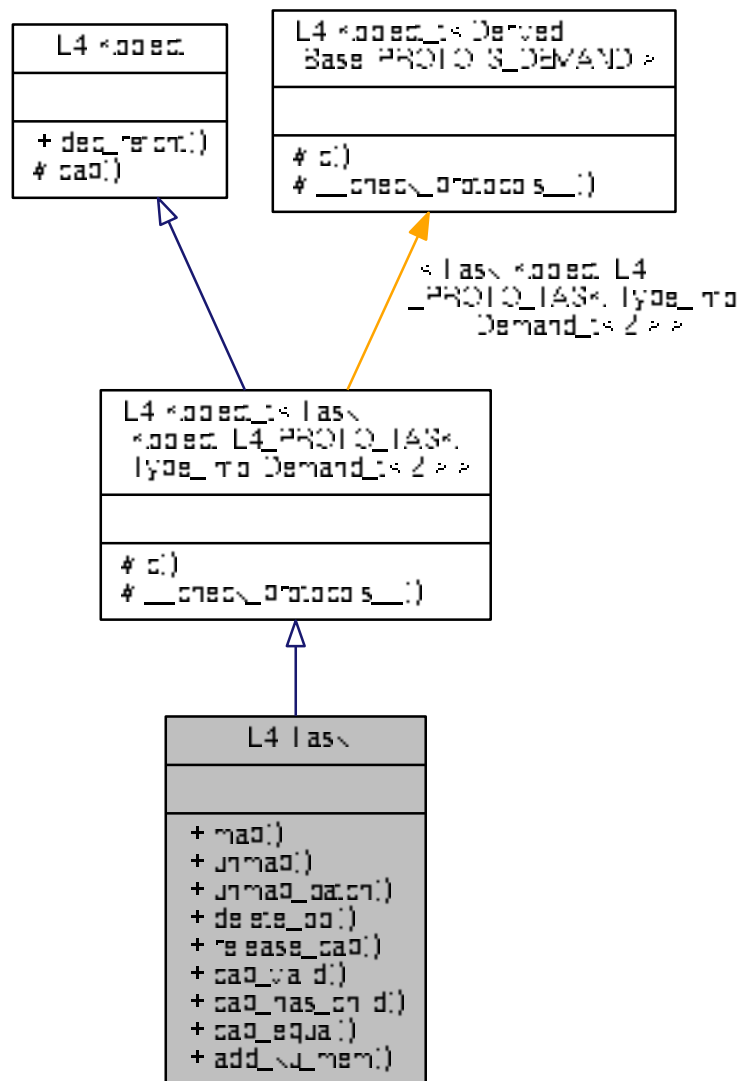
14.164 L4::Task Class Reference

C++ interface of the [Task](#) kernel object.

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 &src_fpage, l4_addr_t src_base, l4_< utcb_t *utcb=l4_< utcb()) throw ()`
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()) throw ()`
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()) throw ()`
Revoke rights from a task.
- `l4_msgtag_t delete_obj (L4::Cap< void > obj, l4_< utcb_t *utcb=l4_< utcb()) throw ()`
Release capability and delete object.

- `l4_msgtag_t release_cap (L4::Cap< void > cap, l4_utcb_t *utcb=l4_utcb()) throw ()`
Release capability.
- `l4_msgtag_t cap_valid (Cap< void > const &cap, l4_utcb_t *utcb=l4_utcb()) throw ()`
Check whether a capability is present (refers to an object).
- `l4_msgtag_t cap_has_child (Cap< void > const &cap, l4_utcb_t *utcb=l4_utcb())) throw ()`
Test whether a capability has child mappings (in another task).
- `l4_msgtag_t cap_equal (Cap< void > const &cap_a, Cap< void > const &cap_b, l4_utcb_t *utcb=l4_utcb()) throw ()`
Test whether two capabilities point to the same object with the same rights.
- `l4_msgtag_t add_ku_mem (l4_fpage_t const &fpage, l4_utcb_t *utcb=l4_utcb()) throw ()`
Add kernel-user memory.

Additional Inherited Members

14.164.1 Detailed Description

C++ interface of the [Task](#) kernel object.

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 43 of file [task](#).

14.164.2 Member Function Documentation

14.164.2.1 add_ku_mem()

```
l4_msgtag_t L4::Task::add_ku_mem (
    l4_fpage_t const & fpage,
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline]
```

Add kernel-user memory.

Parameters

<i>fpage</i>	Flexpage describing the virtual area the memory goes to.
<i>utcb</i>	UTCP pointer of the calling thread.

Returns

Syscall return tag

Definition at line 209 of file [task](#).

14.164.2.2 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() ) throw ()    [inline]
```

Test whether two capabilities point to the same object with the same rights.

Parameters

<i>cap_a</i>	First capability selector to compare.
<i>cap_b</i>	Second capability selector to compare.
<i>utcb</i>	Optional: UTCB pointer of the calling thread.

Return values

<i>tag.label()</i>	= 1: <i>cap_a</i> and <i>cap_b</i> point to the same object.
<i>tag.label()</i>	= 0: The two caps do not point to the same object.

Definition at line 196 of file [task](#).

14.164.2.3 cap_has_child()

```
l4_msgtag_t L4::Task::cap_has_child (
    Cap< void > const & cap,
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline]
```

Test whether a capability has child mappings (in another task).

Parameters

<i>cap</i>	Capability selector to look up in the destination task.
<i>utcb</i>	UTCB pointer of the calling thread.

Return values

<i>tag.label()</i>	= 1: The capability has at least on child mapping in another task.
--------------------	--

Return values

<i>tag.label()</i>	= 0: No child mappings.
--------------------	-------------------------

Deprecated Do not use. Undetermined future, might be removed.

Definition at line 179 of file [task](#).

14.164.2.4 cap_valid()

```
l4_msgtag_t L4::Task::cap_valid (
    Cap< void > const & cap,
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline]
```

Check whether a capability is present (refers to an object).

Parameters

<i>cap</i>	Capability to check for presence.
<i>utcb</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.

Return values

<i>tag.label()</i>	> 0 Capability is present (refers to an object).
<i>tag.label()</i>	== 0 No capability present (void object).

A capability is considered present when it refers to an existing kernel object.

Definition at line 163 of file [task](#).

14.164.2.5 delete_obj()

```
l4_msgtag_t L4::Task::delete_obj (
    L4::Cap< void > obj,
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline]
```

Release capability and delete object.

Parameters

<i>obj</i>	Capability selector of the object to delete.
<i>utcb</i>	UTCB pointer of the calling thread.

Returns

Syscall return tag

The object will be deleted if the `obj` has sufficient rights. No error will be reported if the rights are insufficient, however, the capability is removed in all cases.

Definition at line 133 of file [task](#).

14.164.2.6 map()

```
l4_msgtag_t L4::Task::map (
    Cap< Task > const & src_task,
    l4_fpage_t const & snd_fpage,
    l4_addr_t snd_base,
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline]
```

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.
<i>utcb</i>	UTCB pointer of the calling thread.

Returns

Syscall return tag.

This method allows for asynchronous rights delegation from one task to another. It can be used to share memory as well as to delegate access to objects. The destination task is the task referenced by the capability invoking map and the receive window is the whole address space of said task.

Definition at line 67 of file [task](#).

14.164.2.7 release_cap()

```
l4_msgtag_t L4::Task::release_cap (
    L4::Cap< void > cap,
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline]
```

Release capability.

Parameters

<i>cap</i>	Capability selector to release.
<i>utcb</i>	UTCB pointer of the calling thread.

Returns

Syscall return tag.

This operation unmaps the capability from `this` task.

Definition at line 147 of file [task](#).

14.164.2.8 unmap()

```
l4_msgtag_t L4::Task::unmap (
    l4_fpage_t const & fpage,
    l4_umword_t map_mask,
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline]
```

Revoke rights from the task.

Parameters

<i>fpage</i>	Flexpage that describes an area in the address space or object space of <code>this</code> task
<i>map_mask</i>	Unmap mask, see l4_unmap_flags_t
<i>utcb</i>	UTCB pointer of the calling thread.

Returns

Syscall return tag

This method allows to revoke rights from the destination task and from all the tasks that got the rights delegated from that task (i.e., this operation does a recursive rights revocation).

Note

If the capability possesses delete rights or if it is the last capability pointing to the object, calling this function might destroy the object itself.

Definition at line 90 of file [task](#).

Referenced by [L4Re::Rm::attach\(\)](#).

Here is the caller graph for this function:



14.164.2.9 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() ) throw ()    [inline]
```

Revoke rights from a task.

Parameters

<i>fpages</i>	An array of flexpages. Each item describes an area in the address or object space of <code>this</code> task.
<i>num_fpages</i>	Number of fpages in the <i>fpages</i> array.
<i>map_mask</i>	Unmap mask, see l4_unmap_flags_t .
<i>utcb</i>	UTCB pointer of the calling thread.

This method allows to revoke rights from the destination task and from all the tasks that got the rights delegated from that task (i.e., this operation does a recursive rights revocation).

Precondition

The caller needs to take care that `num_fpages` is not bigger than `L4_UTCB_GENERIC_DATA_SIZE - 2`.

Note

If the capability possesses delete rights or if it is the last capability pointing to the object, calling this function might destroy the object itself.

Definition at line 115 of file [task](#).

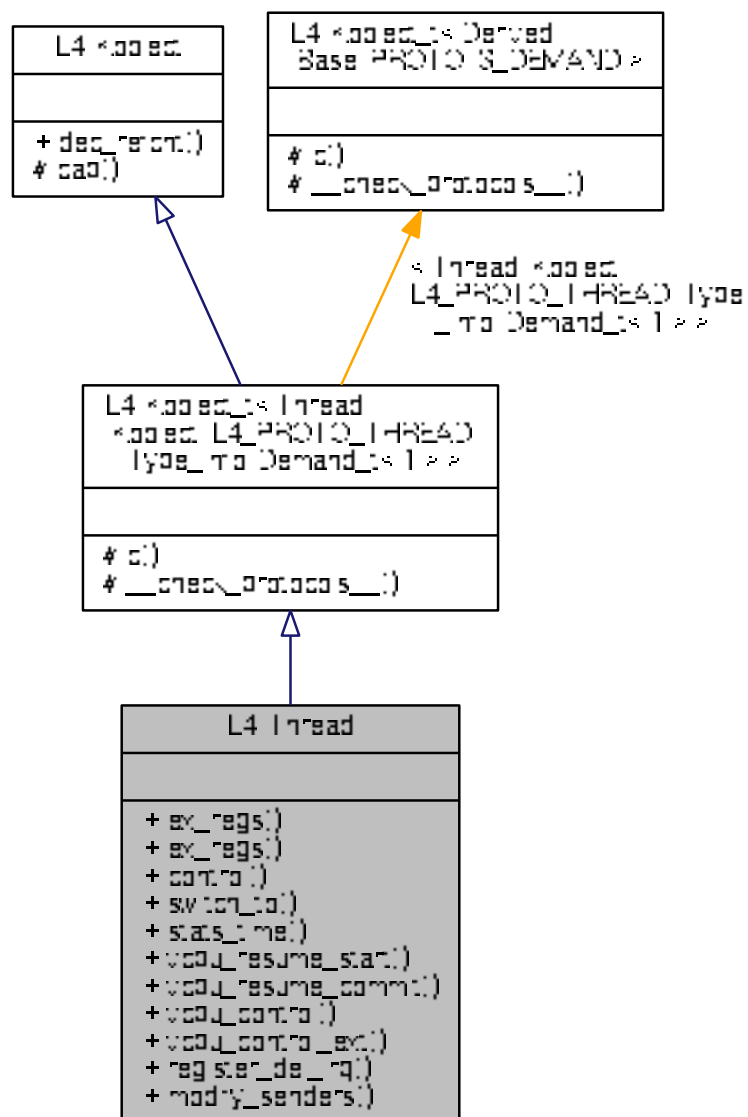
The documentation for this class was generated from the following file:

- [l4/sys/task](#)

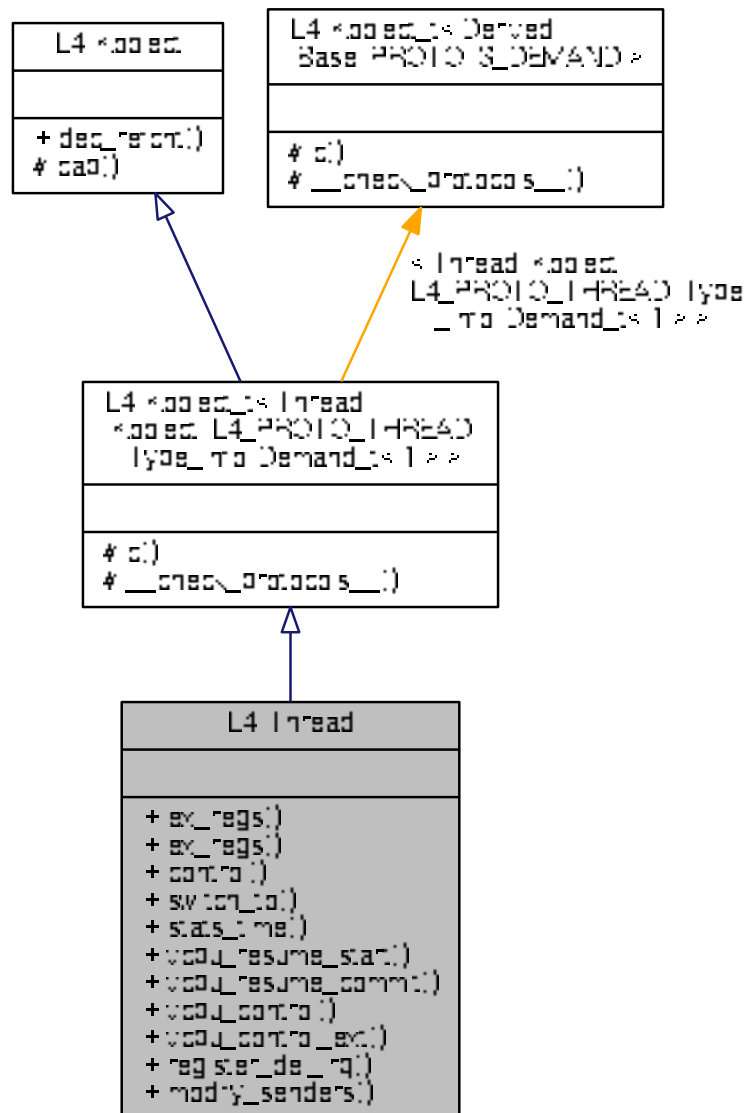
14.165 L4::Thread Class Reference

C++ [L4](#) kernel thread interface.

Inheritance diagram for L4::Thread:



Collaboration diagram for L4::Thread:



Data Structures

- class [Attr](#)
Thread attributes used for control_commit().
- class [Modify_senders](#)
Wrapper class for modifying senders.

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\(\)](#)) throw ()

- *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\(\)](#)) throw ()
- *Exchange basic thread registers and return previous values.*
[l4_msgtag_t control](#) ([Attr](#) const &attr) throw ()
- *Commit the given thread-attributes object.*
[l4_msgtag_t switch_to](#) ([l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) throw ()
- *Switch execution to this thread.*
[l4_msgtag_t stats_time](#) ([l4_kernel_clock_t](#) *us, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) throw ()
- *Get consumed time of thread in us.*
[l4_msgtag_t vcpu_resume_start](#) ([l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) throw ()
- *vCPU resume, start.*
[l4_msgtag_t vcpu_resume_commit](#) ([l4_msgtag_t](#) tag, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) throw ()
- *vCPU resume, commit.*
[l4_msgtag_t vcpu_control](#) ([l4_addr_t](#) vcpu_state, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) throw ()
- *Enable or disable the vCPU feature for the thread.*
[l4_msgtag_t vcpu_control_ext](#) ([l4_addr_t](#) ext_vcpu_state, [l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) throw ()
- *Enable or disable the extended vCPU feature for the thread.*
[l4_msgtag_t register_del_irq](#) ([Cap](#) < [Irq](#) > [irq](#), [l4_utcb_t](#) *u=[l4_utcb\(\)](#)) throw ()
- *Register an IRQ that will trigger upon deletion events.*
[l4_msgtag_t modify_senders](#) ([Modify_senders](#) const &todo) throw ()
- *Apply sender modification rules.*

Additional Inherited Members

14.165.1 Detailed Description

C++ [L4](#) kernel thread 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.

Include File

```
#include <l4/sys/thread>
```

For the C interface see the [Thread](#) API.

Definition at line 58 of file [thread](#).

14.165.2 Member Function Documentation

14.165.2.1 control()

```
l4_msgtag_t L4::Thread::control (
    Attr const & attr ) throw ()    [inline]
```

Commit the given thread-attributes object.

Parameters

<i>attr</i>	the attribute object to commit to the thread.
-------------	---

Definition at line 215 of file [thread](#).

14.165.2.2 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() ) throw ()    [inline]
```

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 use for this operation.

Returns

System call return tag

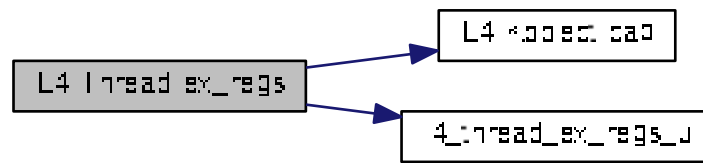
This method allows to manipulate a thread. The basic functionality is to set the instruction pointer and the stack pointer of a thread. Additionally, this method allows also to cancel ongoing IPC operations and to force the thread to raise an artificial exception (see `flags`).

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 85 of file [thread](#).

References [L4::Kobject::cap\(\)](#), and [l4_thread_ex_regs_u\(\)](#).

Here is the call graph for this function:



14.165.2.3 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() ) throw() [inline]
  
```

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 use for this operation.

Returns

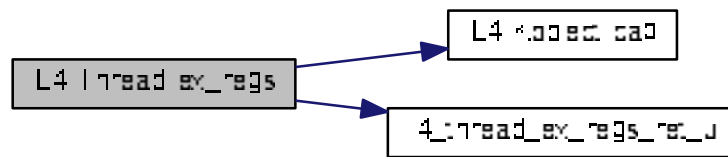
System call return tag. [out] parameters are only valid if the function returns successfully. Use [l4_error\(\)](#) to check.

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*).

Definition at line 111 of file [thread](#).

References [L4::Kobject::cap\(\)](#), and [l4_thread_ex_regs_ret_u\(\)](#).

Here is the call graph for this function:



14.165.2.4 modify_senders()

```

l4_msgtag_t L4::Thread::modify_senders (
    Modify_senders const & todo ) throw ()    [inline]
  
```

Apply sender modification rules.

Parameters

<i>todo</i>	Prepared sender modification rules.
-------------	-------------------------------------

Returns

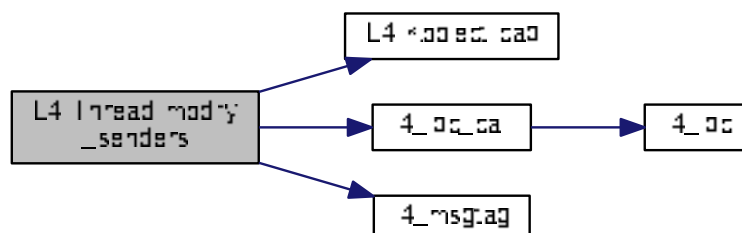
System call return tag.

Definition at line 373 of file [thread](#).

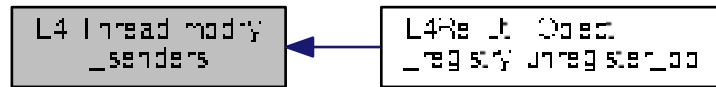
References [L4::Kobject::cap\(\)](#), [l4_ipc_call\(\)](#), [L4_IPC_NEVER](#), [l4_msgtag\(\)](#), and [L4_PROTO_THREAD](#).

Referenced by [L4Re::Util::Object_registry::unregister_obj\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



14.165.2.5 register_del_irq()

```
l4_msgtag_t L4::Thread::register_del_irq (
    Cap< Irq > irq,
    l4_utcb_t * u = l4_utcb() ) throw ()    [inline]
```

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, usually the UTCB of the calling thread.

Returns

System call return tag containing the return code.

An example of a deletion event is the removal of an IPC gate that is bound to this thread.

See also

[l4_thread_register_del_irq](#)

Definition at line 313 of file [thread](#).

14.165.2.6 stats_time()

```
l4_msgtag_t L4::Thread::stats_time (
    l4_kernel_clock_t * us,
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline]
```

Get consumed time of thread in us.

Parameters

	<i>utcb</i>	UTCB of the current thread.
out	<i>us</i>	Consumed time in μ s.

Returns

Syscall return tag.

Definition at line 236 of file [thread](#).

14.165.2.7 switch_to()

```
l4_msgtag_t L4::Thread::switch_to (
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline]
```

Switch execution to this thread.

Parameters

<i>utcb</i>	the UTCB of the current thread.
-------------	---------------------------------

Note

The current time slice is inherited to this thread.

Definition at line 225 of file [thread](#).

14.165.2.8 vcpu_control()

```
l4_msgtag_t L4::Thread::vcpu_control (
    l4_addr_t vcpu_state,
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline]
```

Enable or disable the vCPU feature for the thread.

Parameters

<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()).
<i>utcb</i>	UTCB to use for this operation.

Returns

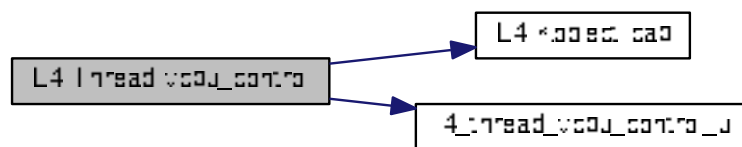
Syscall return tag.

This function enables the vCPU feature of `this` thread if `vcpu_state` is set to a valid kernel-user-memory address, or disables the vCPU feature if `vcpu_state` is 0. (Disable: optional, currently unsupported.)

Definition at line 272 of file [thread](#).

References [L4::Kobject::cap\(\)](#), and [l4_thread_vcpu_control_u\(\)](#).

Here is the call graph for this function:

**14.165.2.9 vcpu_control_ext()**

```
l4_msgtag_t L4::Thread::vcpu_control_ext (
    l4_addr_t ext_vcpu_state,
    l4_utcb_t * utcb = l4_utcb() ) throw() [inline]
```

Enable or disable 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 use for this operation.

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 `this` thread if `ext_vcpu_state` is set to a valid kernel-user-memory address, or disables the vCPU feature if `ext_vcpu_state` is 0.

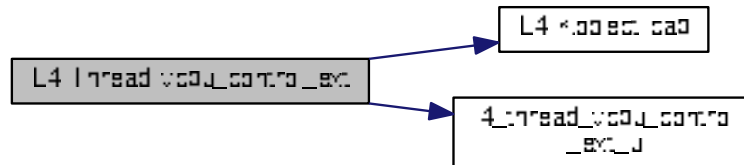
Note

The extended vCPU mode includes the normal vCPU mode.

Definition at line 296 of file [thread](#).

References [L4::Kobject::cap\(\)](#), and [l4_thread_vcpu_control_ext_u\(\)](#).

Here is the call graph for this function:



14.165.2.10 vcpu_resume_commit()

```

l4_msgtag_t L4::Thread::vcpu_resume_commit (
    l4_msgtag_t tag,
    l4_utcb_t * utcb = l4_utcb() ) throw() [inline]
  
```

vCPU resume, commit.

See also

[l4_thread_vcpu_resume_commit](#)

Definition at line 253 of file [thread](#).

14.165.2.11 vcpu_resume_start()

```

l4_msgtag_t L4::Thread::vcpu_resume_start (
    l4_utcb_t * utcb = l4_utcb() ) throw() [inline]
  
```

vCPU resume, start.

See also

[l4_thread_vcpu_resume_start](#)

Definition at line 245 of file [thread](#).

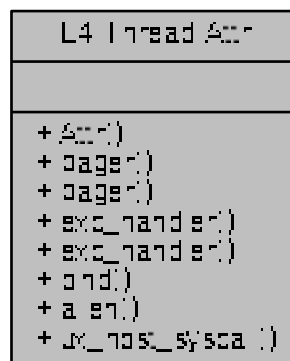
The documentation for this class was generated from the following file:

- [l4/sys/thread](#)

14.166 L4::Thread::Attr Class Reference

[Thread](#) attributes used for `control_commit()`.

Collaboration diagram for L4::Thread::Attr:



Public Member Functions

- [Attr](#) ([l4_utcb_t](#) *utcb=[l4_utcb\(\)](#)) throw ()
Create a thread-attribute object with the given UTCB.
- void [pager](#) ([Cap](#)< void > const &pager) throw ()
Set the pager capability selector.
- [Cap](#)< void > [pager](#) () throw ()
Get the capability selector used for page-fault messages.
- void [exc_handler](#) ([Cap](#)< void > const &exc_handler) throw ()
Set the exception-handler capability selector.
- [Cap](#)< void > [exc_handler](#) () throw ()
Get the capability selector used for exception messages.
- void [bind](#) ([l4_utcb_t](#) *thread_utcb, [Cap](#)< [Task](#) > const &task) throw ()
Bind the thread to a task.
- void [alien](#) (int on) throw ()
Set the thread to alien mode.
- void [ux_host_syscall](#) (int on) throw ()
Allow host system calls on Fiasco-UX.

Friends

- class [L4::Thread](#)

14.166.1 Detailed Description

[Thread](#) attributes used for `control_commit()`.

This class is responsible for initializing various attributes of a thread in a UTCB for the `control_commit()` method.

See also

[Thread control](#) for some more details.

Definition at line 125 of file [thread](#).

14.166.2 Constructor & Destructor Documentation

14.166.2.1 Attr()

```
L4::Thread::Attr::Attr (
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline], [explicit]
```

Create a thread-attribute object with the given UTCB.

Parameters

<i>utcb</i>	The UTCB to use for the later <code>L4::Thread::control_commit()</code> function. Usually this is the UTCB of the calling thread.
-------------	---

Definition at line 138 of file [thread](#).

14.166.3 Member Function Documentation

14.166.3.1 bind()

```
void L4::Thread::Attr::bind (
    l4_utcb_t * thread_utcb,
    Cap< Task > const & task ) throw ()    [inline]
```

Bind the thread to a task.

Parameters

<i>thread_utcb</i>	The UTCB address of the thread within the task specified by <i>task</i> .
<i>task</i>	The capability selector for the task the thread shall be bound to.

Binding a thread to a task means that the thread shall afterwards execute in the given task. To actually start execution you need to use [L4::Thread::ex_regs\(\)](#).

Definition at line 191 of file [thread](#).

14.166.3.2 `exc_handler()` [1/2]

```
void L4::Thread::Attr::exc_handler (
    Cap< void > const & exc_handler ) throw ()    [inline]
```

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 167 of file [thread](#).

14.166.3.3 `exc_handler()` [2/2]

```
Cap<void> L4::Thread::Attr::exc_handler ( ) throw ()    [inline]
```

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 176 of file [thread](#).

14.166.3.4 `pager()` [1/2]

```
void L4::Thread::Attr::pager (
    Cap< void > const & pager ) throw ()    [inline]
```

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 148 of file [thread](#).

14.166.3.5 pager() [2/2]

```
Cap<void> L4::Thread::Attr::pager ( ) throw ( ) [inline]
```

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 157 of file [thread](#).

14.166.3.6 ux_host_syscall()

```
void L4::Thread::Attr::ux_host_syscall (
    int on ) throw ( ) [inline]
```

Allow host system calls on Fiasco-UX.

Precondition

Running on Fiasco-UX.

Definition at line 205 of file [thread](#).

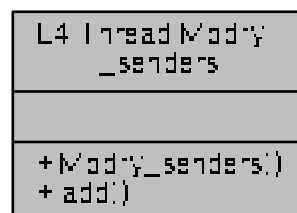
The documentation for this class was generated from the following file:

- [l4/sys/thread](#)

14.167 L4::Thread::Modify_senders Class Reference

Wrapper class for modifying senders.

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) throw ()`

Add a rule.

14.167.1 Detailed Description

Wrapper class for modifying senders.

Use the `add()` function to add modification rules, and use `modify_senders()` to commit. Do not use the UTCB inbetween as it is used by `add()` and `modify_senders()`.

Definition at line 323 of file `thread`.

14.167.2 Member Function Documentation

14.167.2.1 add()

```
int L4::Thread::Modify_senders::add (
    l4_umword_t match_mask,
    l4_umword_t match,
    l4_umword_t del_bits,
    l4_umword_t add_bits ) throw ()    [inline]
```

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 <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 sucess, <0 on error

Only the first match is applied.

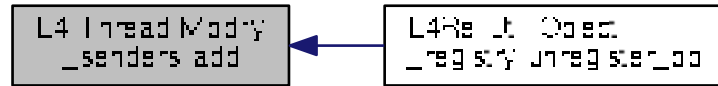
See also

[l4_thread_modify_sender_add\(\)](#)

Definition at line 352 of file `thread`.

Referenced by `L4Re::Util::Object_registry::unregister_obj()`.

Here is the caller graph for this function:



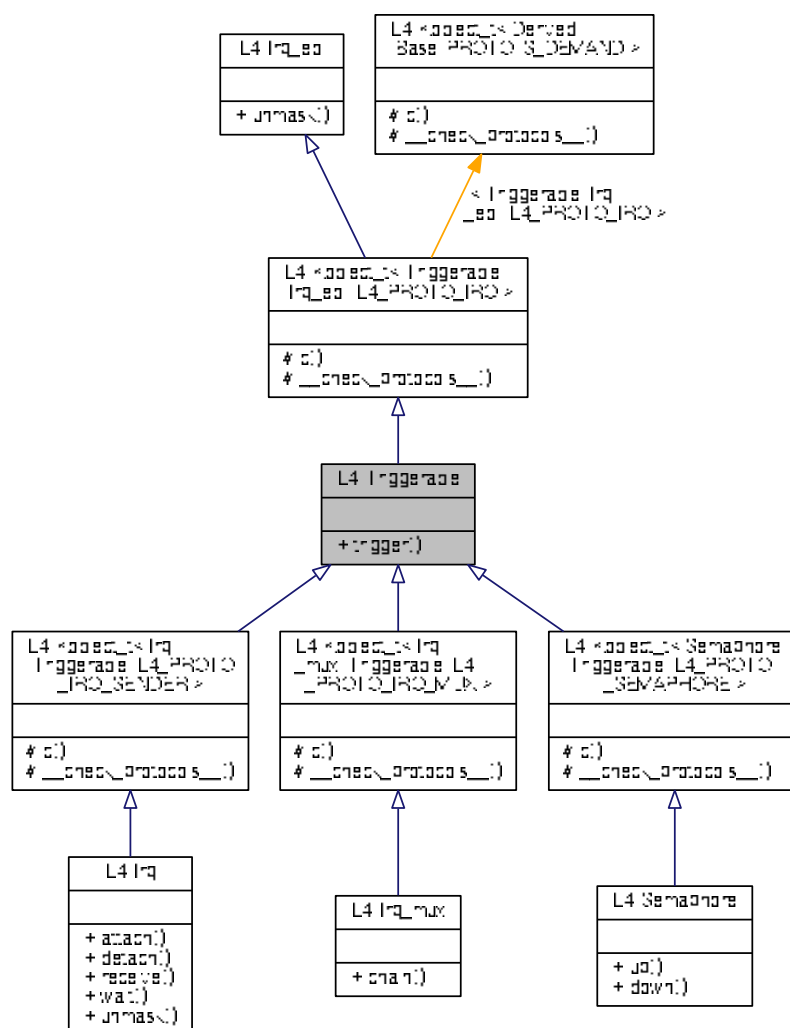
The documentation for this class was generated from the following file:

- [l4/sys/thread](#)

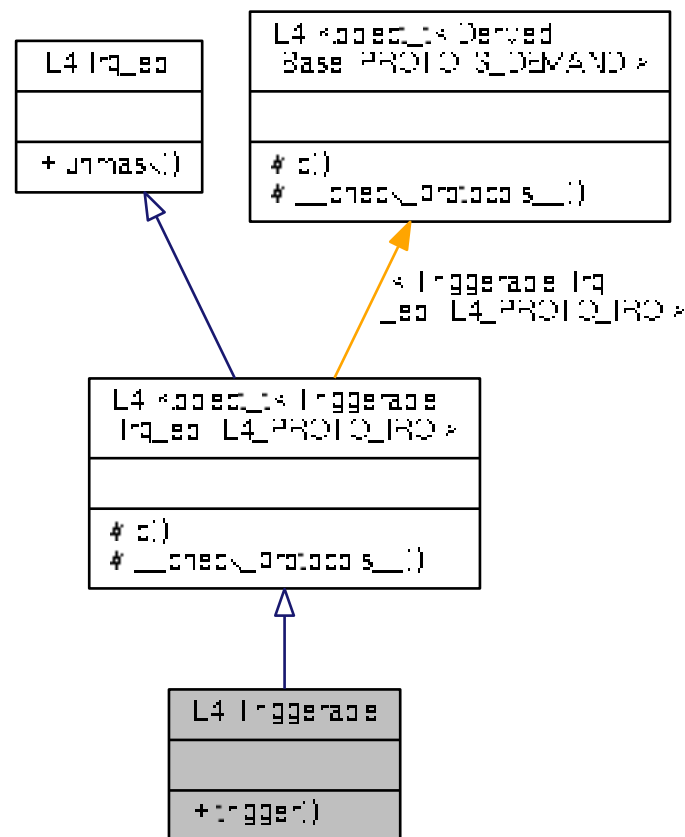
14.168 L4::Triggerable Struct Reference

Interface that allows an object to be triggered by some source.

Inheritance diagram for L4::Triggerable:



Collaboration diagram for L4::Triggerable:



Public Member Functions

- `l4_msgtag_t trigger (l4_utcb_t *utcb=l4_utcb()) throw ()`
Trigger.

Additional Inherited Members

14.168.1 Detailed Description

Interface that allows an object to be triggered by some source.

This interface is usually used in conjunction with [L4::lcu](#).

Definition at line 75 of file [irq](#).

14.168.2 Member Function Documentation

14.168.2.1 trigger()

```
l4_msgtag_t L4::Triggerable::trigger (
    l4_utcb_t * utcb = l4_utcb() ) throw ()    [inline]
```

Trigger.

Parameters

<i>utcb</i>	UTCB to be used for this operation, usually the UTCB of the calling thread.
-------------	---

Returns

Syscall return tag for a send-only operation, use [l4_ipc_error\(\)](#) to check for errors (**do not** use [l4_error\(\)](#)).

Note

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 errors, **do not** use [l4_error\(\)](#), because [l4_error\(\)](#) will always return an error.

Examples:

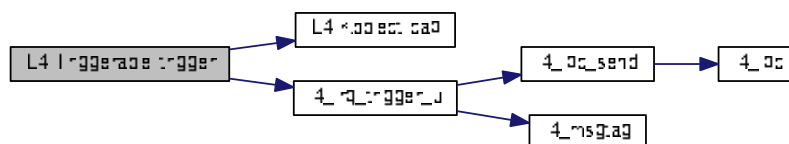
[examples/libs/l4re/c++/shared_ds/ds_clnt.cc](#).

Definition at line 90 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:

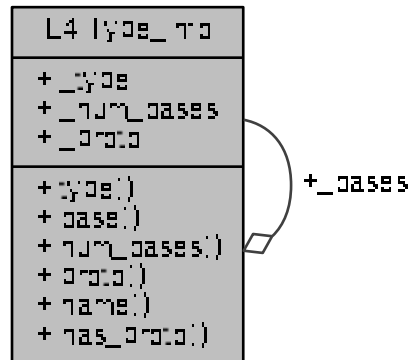
- [l4/sys/irq](#)

14.169 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.

14.169.1 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 509 of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

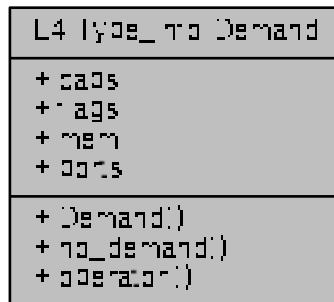
- [l4/sys/__typeinfo.h](#)

Data type for expressing the needed receive buffers at the server-side of an interface.

Inheritance diagram for L4::Type_info::Demand:



Collaboration diagram for L4::Type_info::Demand:



Public Member Functions

- [Demand](#) (unsigned char [caps](#)=0, unsigned char [flags](#)=0, unsigned char [mem](#)=0, unsigned char [ports](#)=0)
Make [Demand](#) object.
- bool [no_demand](#) () const
- [Demand operator|](#) ([Demand](#) const &rhs) const
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](#)
number of memory receive buffers.
- unsigned char [ports](#)
number of IO-port receive buffers.

14.170.1 Detailed Description

Data type for expressing the needed receive buffers at the server-side of an interface.

Definition at line 516 of file [__typeinfo.h](#).

14.170.2 Constructor & Destructor Documentation

14.170.2.1 Demand()

```
L4::Type_info::Demand::Demand (
    unsigned char caps = 0,
    unsigned char flags = 0,
    unsigned char mem = 0,
    unsigned char ports = 0 ) [inline], [explicit]
```

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>	number of memory receive windows.
<i>ports</i>	number of IO-port receive windows.

Definition at line 537 of file [__typeinfo.h](#).

14.170.3 Member Function Documentation

14.170.3.1 no_demand()

```
bool L4::Type_info::Demand::no_demand ( ) const [inline]
```

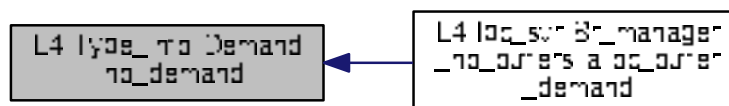
Returns

true if there is no demand at all

Definition at line 542 of file [__typeinfo.h](#).

Referenced by [L4::lpc_svr::Br_manager_no_buffers::alloc_buffer_demand\(\)](#).

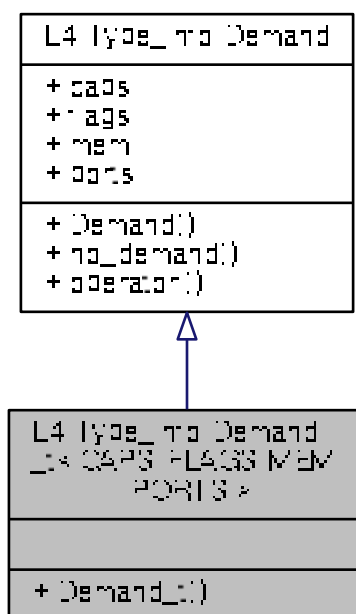
Here is the caller graph for this function:



The documentation for this class was generated from the following file:

- [l4/sys/__typeinfo.h](#)

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

14.171.1 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>	number of memory receive windows needed.
<i>PORTS</i>	number of IO-port receive windwows needed.

Definition at line 563 of file `__typeinfo.h`.

14.171.2 Member Enumeration Documentation

14.171.2.1 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	number of memory receive windows.
Ports	number of IO-port receive windows.

Definition at line 565 of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

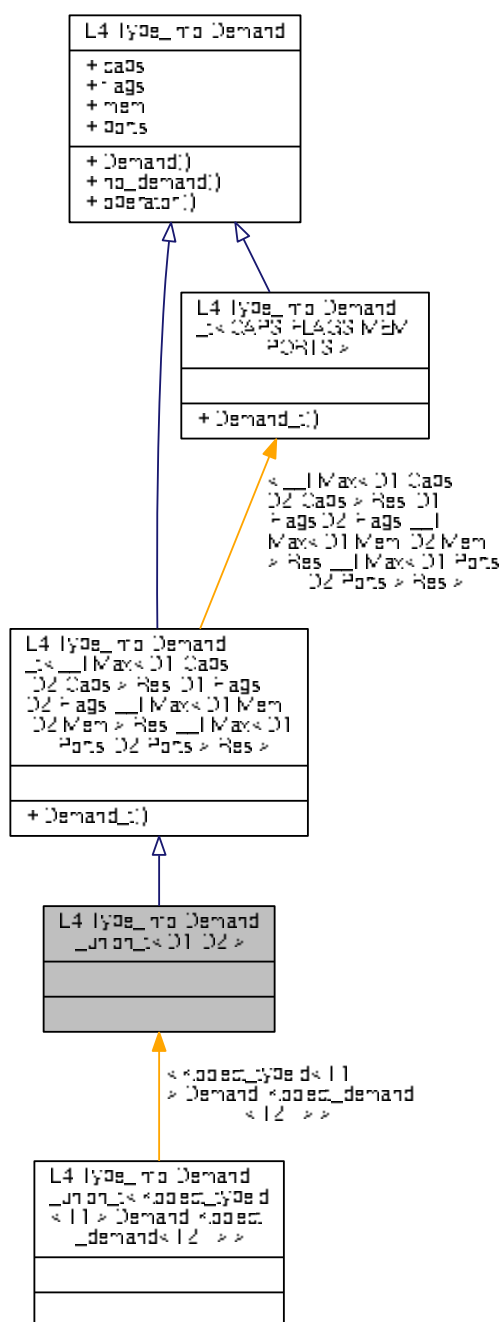
- [l4/sys/__typeinfo.h](#)

14.172 L4::Type_info::Demand_union_t< D1, D2 > Struct Template Reference

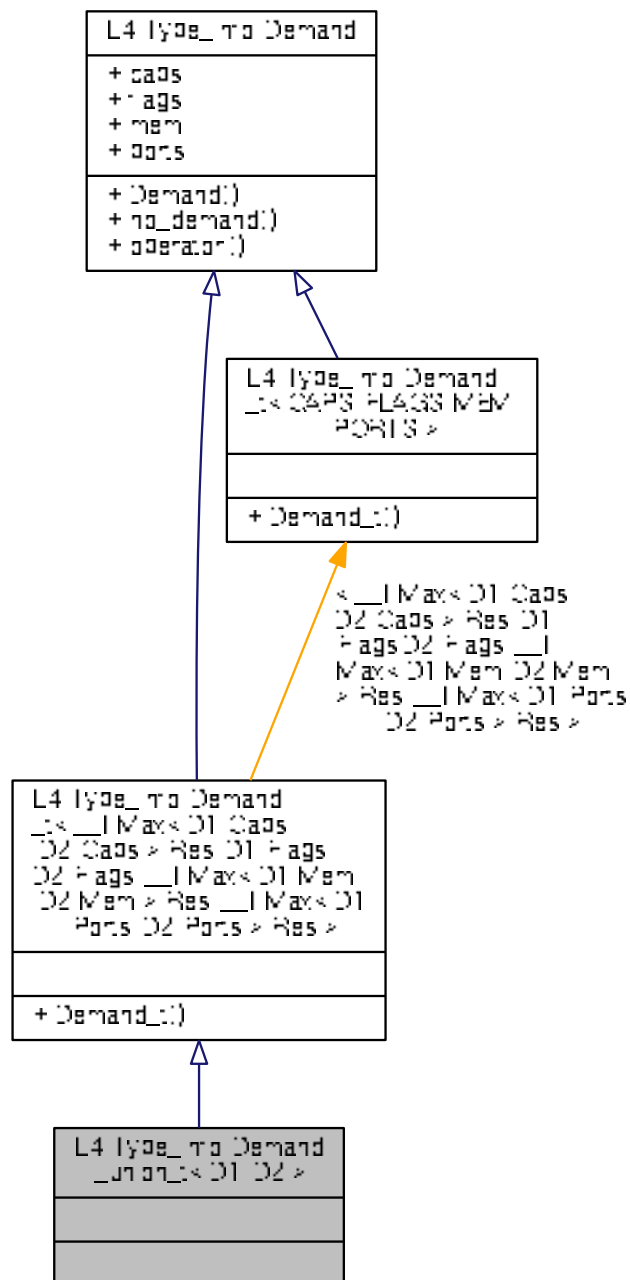
Template type statically describing the combination of two [Demand](#) object.

```
#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

14.172.1 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

$D1$	first demand object.
$D2$	second demand object.

Definition at line 583 of file __typeinfo.h.

The documentation for this struct was generated from the following file:

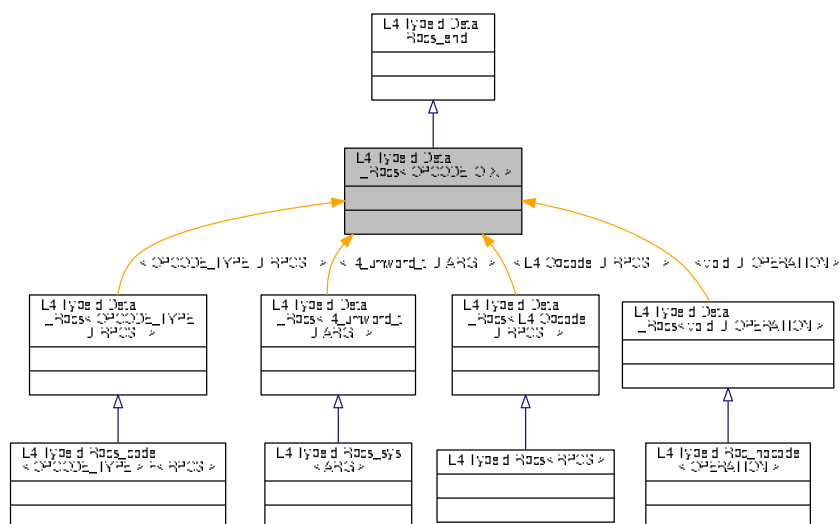
- `l4/sys/__typeinfo.h`

14.173 L4::Typeid::Detail:: Rpcs< OPCODE, O, X > Struct Template Reference

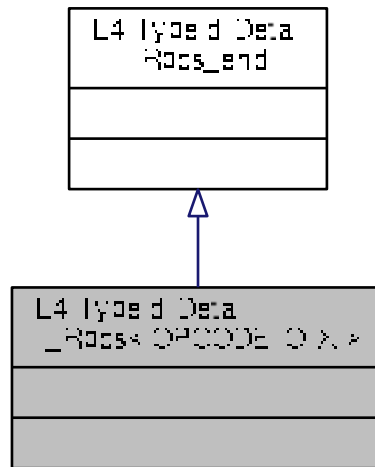
Empty list of RPCs.

```
#include <__typeinfo.h>
```

Inheritance diagram for L4::Typeid::Detail::_Rpc< OPCODE, O, X >:



Collaboration diagram for L4::Typeid::Detail::_Rpc< OPCODE, O, X >:



14.173.1 Detailed Description

```
template<typename OPCODE, unsigned O, typename ... X>
struct L4::Typeid::Detail::_Rpc< OPCODE, O, X >
```

Empty list of RPCs.

Definition at line 375 of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

- [l4/sys/__typeinfo.h](#)

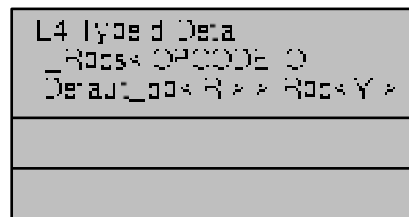
14.174 L4::Typeid::Detail::_Rpc< OPCODE, O, Default_op< R > >::Rpc< Y > Struct Template Reference

Find the given RPC in the list.

```
#include <__typeinfo.h>
```

Inherits L4::Typeid::Detail::Rpc< OP, RPCS >.

Collaboration diagram for L4::Typeid::Detail::_Rpc< OPCODE, O, Default_op< R > >::Rpc< Y >:



14.174.1 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 [409](#) of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

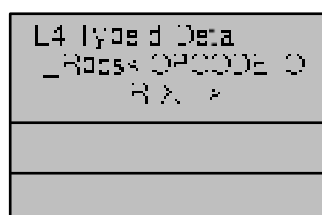
- [l4/sys/__typeinfo.h](#)

14.175 L4::Typeid::Detail::_Rpc< OPCODE, O, R, X... > Struct Template Reference

Non-empty list of RPCs.

```
#include <__typeinfo.h>
```

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 [_Rpcs](#) type
The list element itself.
- typedef OP CODE [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 [_Rpcs](#)< OP CODE, _Get_opcode< R, O >::value+1, X... >::type next
The next RPC in the list or [Rpcs_end](#) if this is the last.

14.175.1 Detailed Description

```
template<typename OP CODE, unsigned O, typename R, typename ... X>
struct L4::Typeid::Detail::_Rpcs< OP CODE, O, R, X... >
```

Non-empty list of RPCs.

Definition at line 379 of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

- l4/sys/[__typeinfo.h](#)

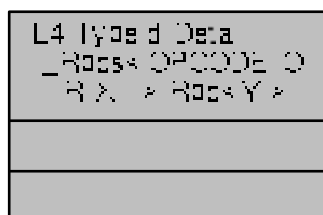
14.176 L4::Typeid::Detail::_Rpcs< OP CODE, O, R, X... >::Rpc< Y > Struct Template Reference

Find the given RPC in the list.

```
#include <__typeinfo.h>
```

Inherits L4::Typeid::Detail::Rpc< OP, RPCS >.

Collaboration diagram for L4::Typeid::Detail::_Rpcs< OP CODE, O, R, X... >::Rpc< Y >:



14.176.1 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 392 of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

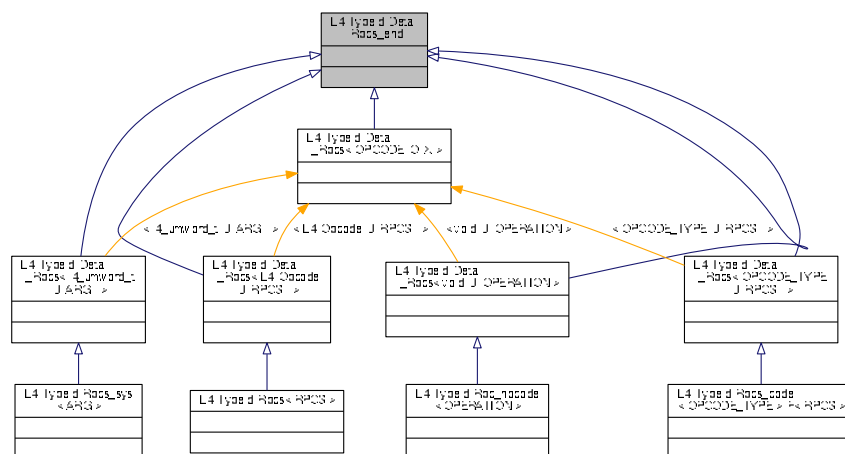
- [l4/sys/__typeinfo.h](#)

14.177 L4::Typeid::Detail::Rpc_end Struct Reference

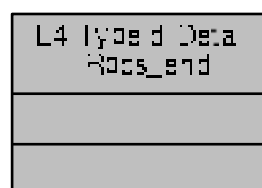
Internal end-of-list marker.

```
#include <__typeinfo.h>
```

Inheritance diagram for L4::Typeid::Detail::Rpc_end:



Collaboration diagram for L4::Typeid::Detail::Rpc_end:



14.177.1 Detailed Description

Internal end-of-list marker.

Definition at line 327 of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

- [l4/sys/__typeinfo.h](#)

14.178 L4::Typeid::P_dispatch< LIST > Struct Template Reference

Use for protocol based dispatch stage.

```
#include <__typeinfo.h>
```

Inherits L4::Typeid::_P_dispatch< LIST >.

Collaboration diagram for L4::Typeid::P_dispatch< LIST >:



14.178.1 Detailed Description

```
template<typename LIST>
struct L4::Typeid::P_dispatch< LIST >
```

Use for protocol based dispatch stage.

Definition at line 318 of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

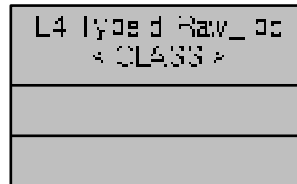
- [l4/sys/__typeinfo.h](#)

14.179 L4::Typeid::Raw_ipc< CLASS > Struct Template Reference

RPCs list for passing raw incoming IPC to the server object.

```
#include <l4/sys/capability>
```

Collaboration diagram for L4::Typeid::Raw_ipc< CLASS >:



14.179.1 Detailed Description

```
template<typename CLASS>
struct L4::Typeid::Raw_ipc< CLASS >
```

RPCs list for passing raw incoming IPC to the server object.

Template Parameters

<i>CLASS</i>	The type of the interface (e.g., L4::lcu)
--------------	--

This template allows to have fully handcrafted IPC protocols.

Definition at line [422](#) of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

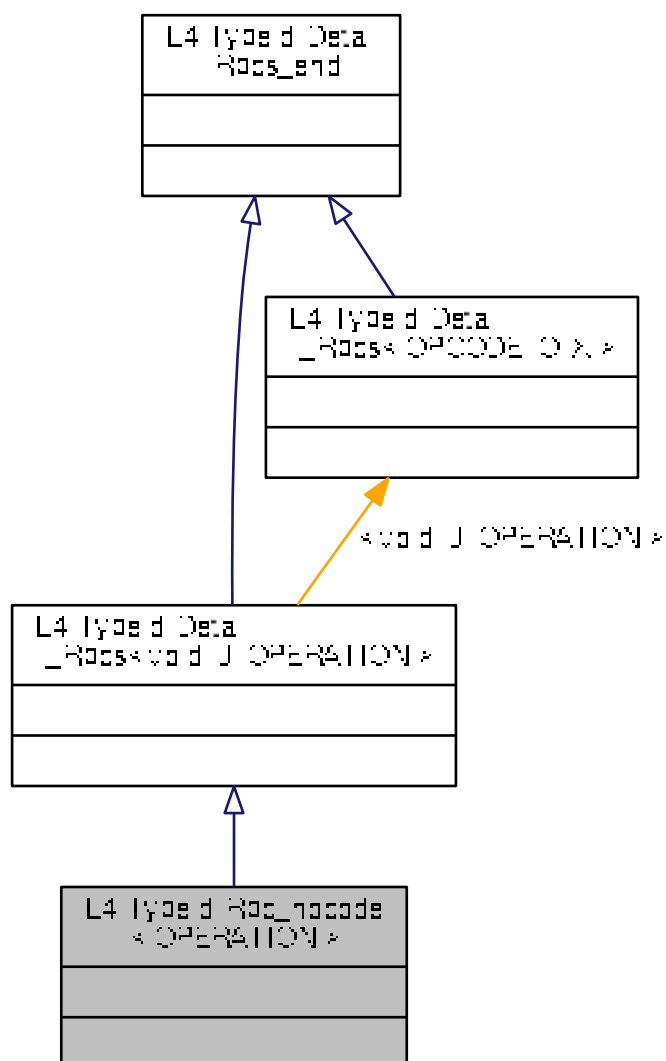
- [l4/sys/__typeinfo.h](#)

14.180 L4::Typeid::Rpc_nocode< OPERATION > Struct Template Reference

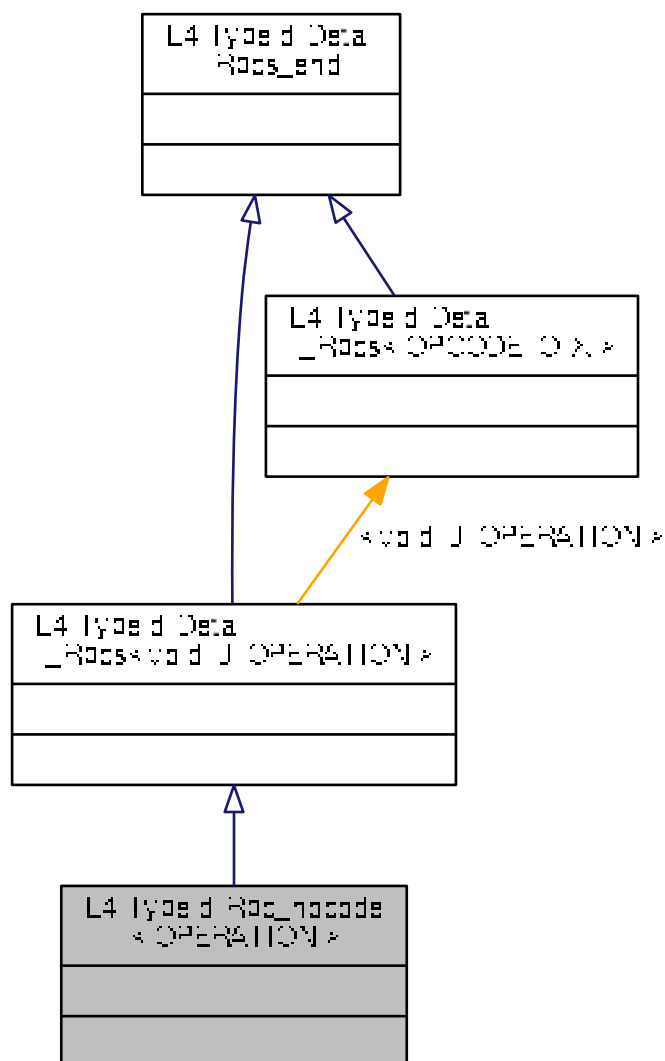
List of RPCs of an interface using a single operation without an opcode.

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Typeid::Rpc_nocode< OPERATION >:



Collaboration diagram for L4::Typeid::Rpc_nocode< OPERATION >:



14.180.1 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

<code>OPERATION</code>	The RPC operation as defined by <code>L4_RPC</code> etc.
------------------------	--

Definition at line 464 of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

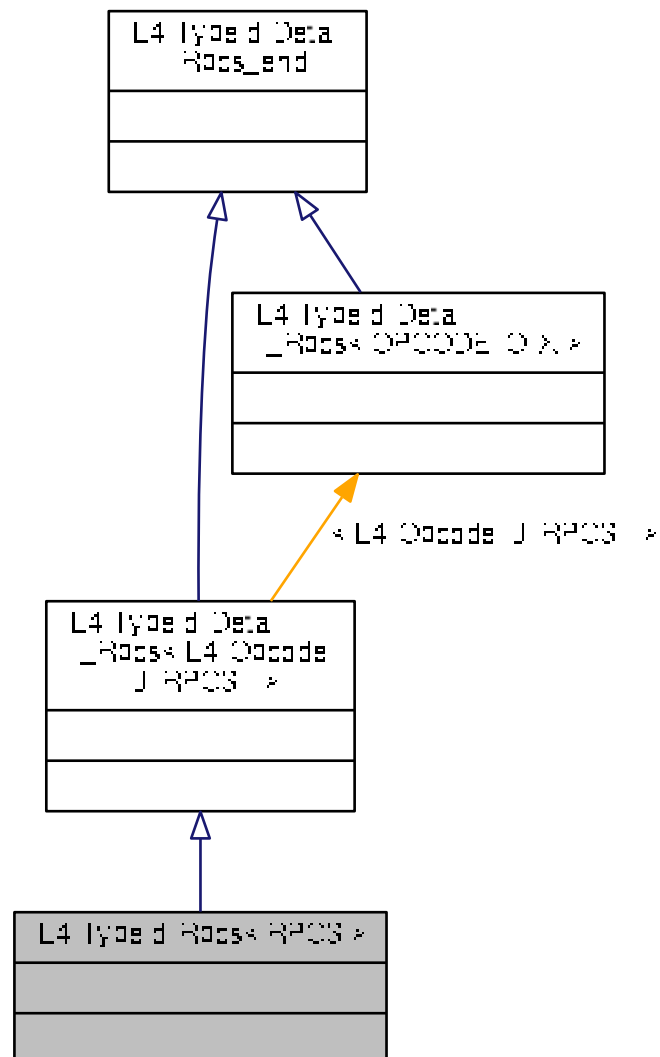
- [l4/sys/__typeinfo.h](#)

14.181 L4::Typeid::Rpc< RPCS > Struct Template Reference

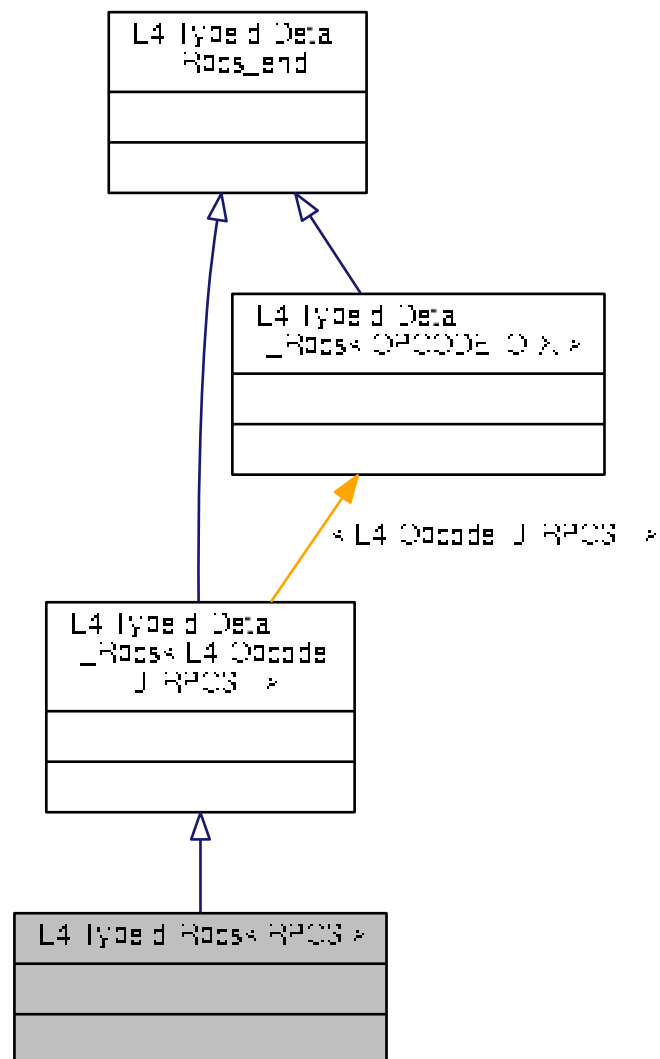
Standard list of RPCs of an interface.

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Typeid::Rpc< RPCS >:



Collaboration diagram for L4::Typeid::Rpc< RPCS >:



14.181.1 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.

Definition at line [438](#) of file [__typeinfo.h](#).

The documentation for this struct was generated from the following file:

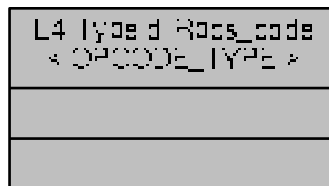
- [l4/sys/__typeinfo.h](#)

14.182 L4::Typeid::Rpc_code< OPCODE_TYPE > Struct Template Reference

List of RPCs of an interface using a special opcode type.

```
#include <l4/sys/capability>
```

Collaboration diagram for L4::Typeid::Rpc_code< OPCODE_TYPE >:



Data Structures

- struct [F](#)

14.182.1 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 [449](#) of file [__typeinfo.h](#).

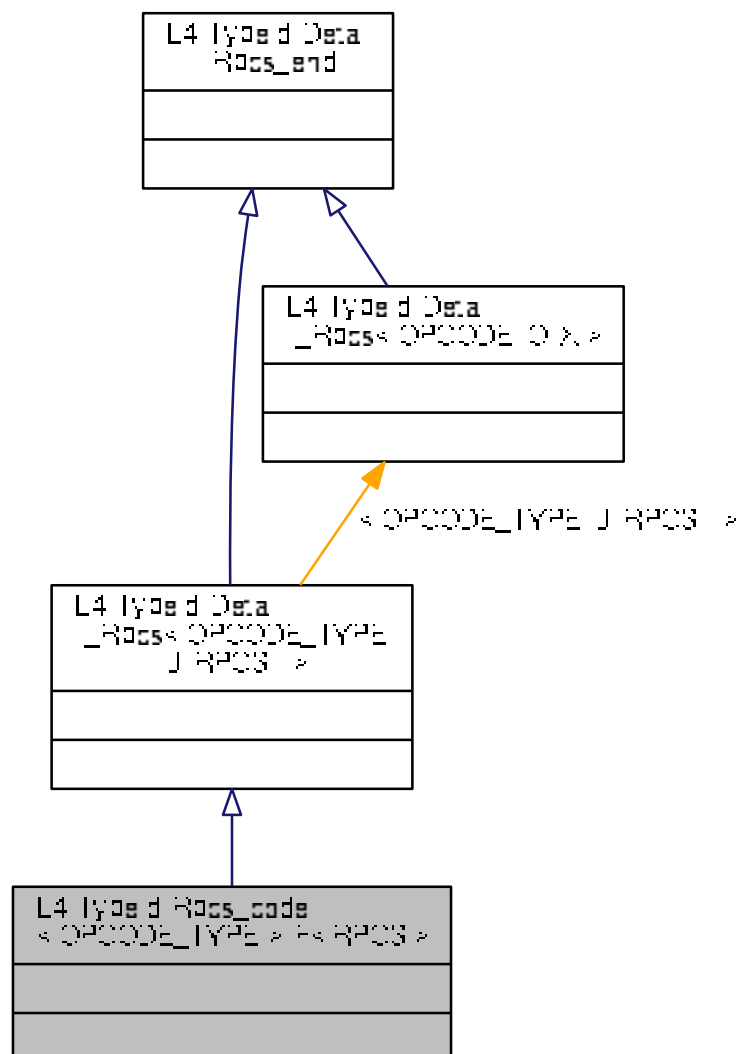
The documentation for this struct was generated from the following file:

- [l4/sys/__typeinfo.h](#)

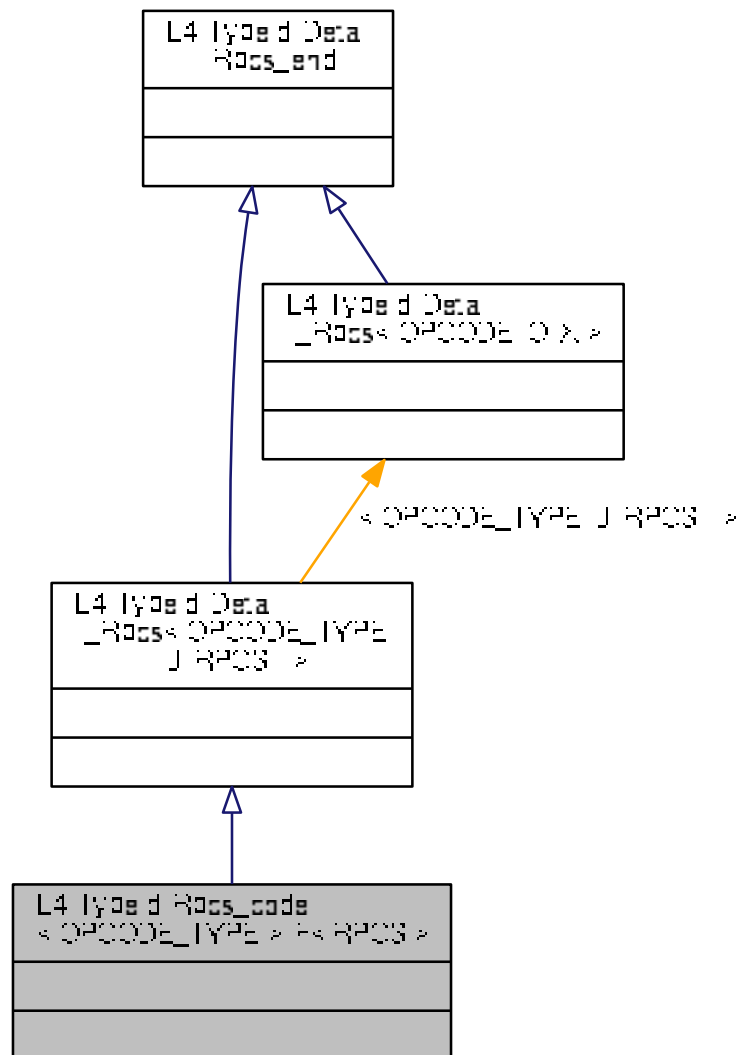
14.183 L4::Typeid::Rpc_code< OPCODE_TYPE >::F< RPCS > Struct Template Reference

```
#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 >:



14.183.1 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 455 of file `__typeinfo.h`.

The documentation for this struct was generated from the following file:

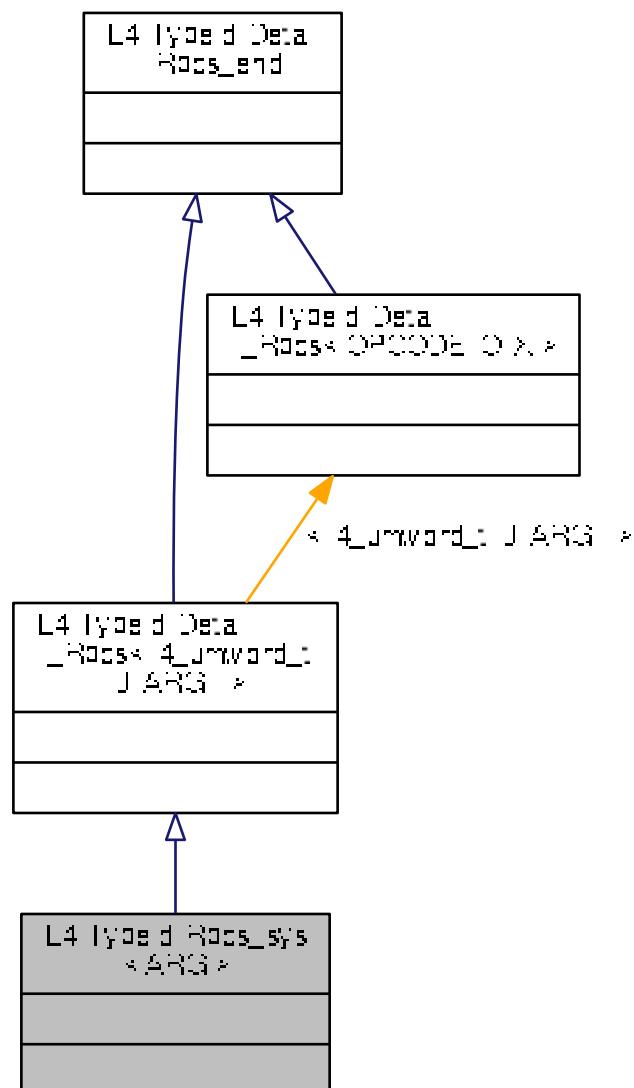
- [l4/sys/__typeinfo.h](#)

14.184 L4::Typeid::Rpcsys< ARG > Struct Template Reference

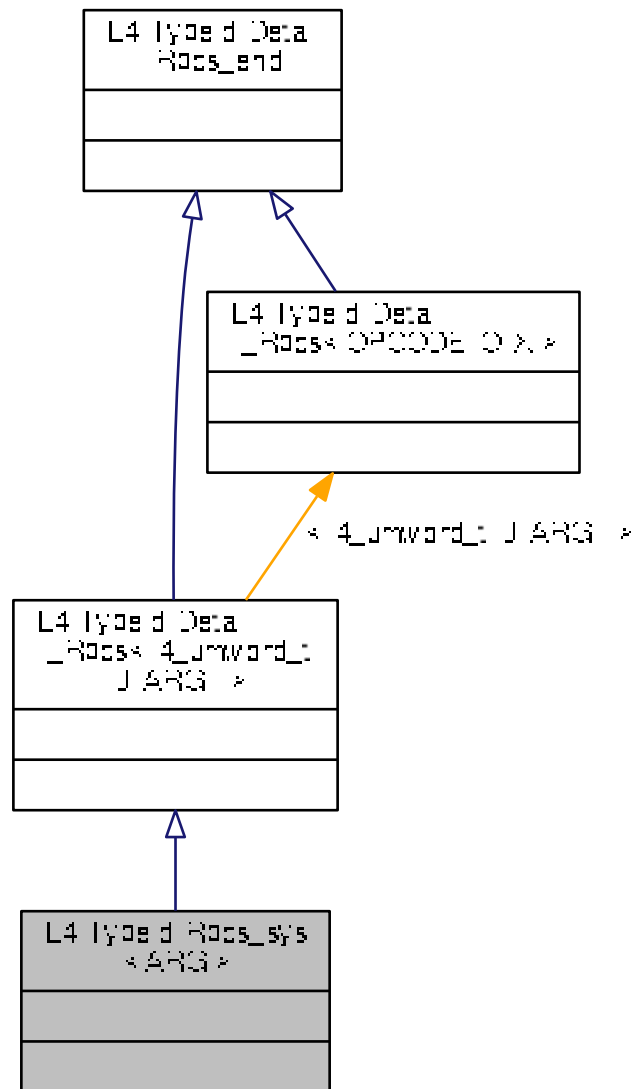
List of RPCs typically used for kernel interfaces.

```
#include <l4/sys/capability>
```

Inheritance diagram for L4::Typeid::Rpcsys< ARG >:



Collaboration diagram for L4::Typeid::Rpcsys< ARG >:



14.184.1 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.
-------------	---

Definition at line 475 of file `__typeinfo.h`.

- `l4/sys/__typeinfo.h`

Boolean meta type.

The diagram illustrates the L4 types hierarchy. At the top is a box labeled "L4 types Base = V". Below it are six boxes, each representing a different L4 type. These boxes are connected to the top box by arrows. Below these six boxes are more boxes, representing further refinements or specializations of the L4 types. Arrows point from the six middle boxes to these bottom boxes. At the bottom are two boxes, representing the final L4 types in the hierarchy. Arrows point from the bottom boxes to these final boxes.

L4 Types Baa < V >

- `typedef Bool< V > type`
The meta type itself.

14.185.1 Detailed Description

```
template<bool V>  
struct L4::Types::Bool< V >
```

Boolean meta type.

Template Parameters

<i>V</i>	The boolean value
----------	-------------------

Definition at line [165](#) of file [types](#).

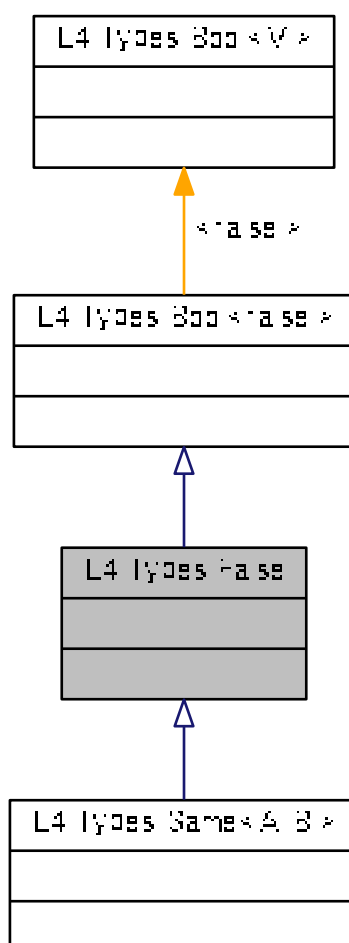
The documentation for this struct was generated from the following file:

- [l4/sys/cxx/types](#)

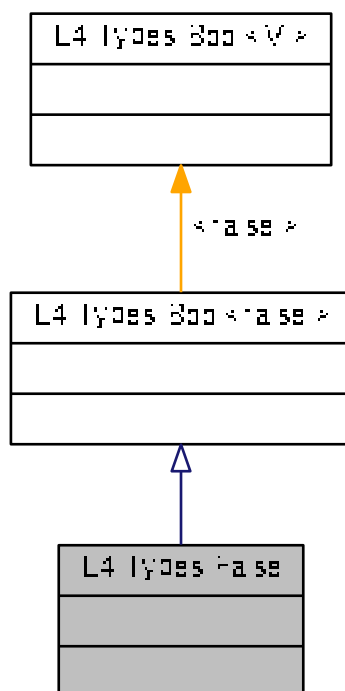
14.186 L4::Types::False Struct Reference

[False](#) meta value.

Inheritance diagram for L4::Types::False:



Collaboration diagram for L4::Types::False:



Additional Inherited Members

14.186.1 Detailed Description

[False](#) meta value.

Definition at line [173](#) of file [types](#).

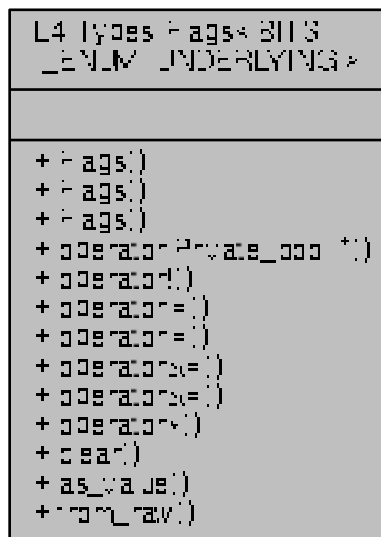
The documentation for this struct was generated from the following file:

- [l4/sys/cxx/types](#)

14.187 L4::Types::Flags< BITS_ENUM, UNDERLYING > Class Template Reference

Template for defining typical [Flags](#) bitmaps.

Collaboration diagram for L4::Types::Flags< BITS_ENUM, UNDERLYING >:



Public Types

- enum [None_type](#) { [None](#) }
The none type to get an empty bitmap.
- typedef UNDERLYING [value_type](#)
type of the underlying value
- typedef BITS_ENUM [bits_enum_type](#)
enum type defining a name for each bit
- typedef [Flags](#)< BITS_ENUM, UNDERLYING > [type](#)
the Flags<> type itself

Public Member Functions

- [Flags](#) ([None_type](#))
Make an empty bitmap.
- [Flags](#) ()
Make default [Flags](#).
- [Flags](#) (BITS_ENUM e)
Make flags from bit name.
- [operator Private_bool *](#) () const
Support for `if (flags)` syntax (test for non-empty flags).
- [bool operator!](#) () const
Support for `if (!flags)` syntax (test for empty flags).
- [type & operator|=](#) ([type](#) rhs)
Support `|=` of two compatible [Flags](#) types.

- `type & operator| = (bits_enum_type rhs)`
Support `| =` of *Flags* type and bit name.
- `type & operator& = (type rhs)`
Support `& =` of two compatible *Flags* types.
- `type & operator& = (bits_enum_type rhs)`
Support `& =` of *Flags* type and bit name.
- `type operator~ () const`
Support `~` for *Flags* types.
- `type & clear (bits_enum_type flag)`
Clear the given flag.
- `value_type as_value () const`
Get the underlying value.

Static Public Member Functions

- static `type from_raw (value_type v)`
Make flags from a raw value of *value_type*.

Friends

- `type operator| (type lhs, type rhs)`
Support `|` of two compatible *Flags* types.
- `type operator| (type lhs, bits_enum_type rhs)`
Support `|` of *Flags* type and bit name.
- `type operator& (type lhs, type rhs)`
Support `&` of two compatible *Flags* types.
- `type operator& (type lhs, bits_enum_type rhs)`
Support `&` of *Flags* type and bit name.

14.187.1 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
};

typedef L4::Types::Flags<Test_flag> Test_flags;

Test_flags x = Do_weak_tests;

if (x & Do_strong_tests) { ... }
x |= Do_strong_tests;
if (x & Do_strong_tests) { ... }
```

Definition at line 63 of file [types](#).

14.187.2 Member Enumeration Documentation

14.187.2.1 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 80 of file [types](#).

14.187.3 Constructor & Destructor Documentation

14.187.3.1 Flags() [1/2]

```
template<typename BITS_ENUM , typename UNDERLYING = unsigned long>
L4::Types::Flags< BITS_ENUM, UNDERLYING >::Flags (
    None_type ) [inline]
```

Make an empty bitmap.

Usually used for implicit conversion from [Flags::None](#).

```
Flags x = Flags::None;
```

Definition at line 90 of file [types](#).

14.187.3.2 **Flags()** [2/2]

```
template<typename BITS_ENUM , typename UNDERLYING = unsigned long>
L4::Types::Flags< BITS_ENUM, UNDERLYING >::Flags (
    BITS_ENUM e ) [inline]
```

Make flags from bit name.

Usually used for implicit conversion for a bit name.

```
Test_flags f = Do_strong_tests;
```

Definition at line 103 of file [types](#).

14.187.4 **Member Function Documentation****14.187.4.1** **clear()**

```
template<typename BITS_ENUM , typename UNDERLYING = unsigned long>
type& L4::Types::Flags< BITS_ENUM, UNDERLYING >::clear (
    bits_enum_type flag ) [inline]
```

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 154 of file [types](#).

References [L4::Types::Flags< BITS_ENUM, UNDERLYING >::operator&=\(\)](#).

Here is the call graph for this function:



14.187.4.2 from_raw()

```
template<typename BITS_ENUM , typename UNDERLYING = unsigned long>
static type L4::Types::Flags< BITS_ENUM, UNDERLYING >::from_raw (
    value_type v ) [inline], [static]
```

Make flags from a raw value of *value_type*.

This function may be used for example in C wrapper code.

Definition at line 110 of file [types](#).

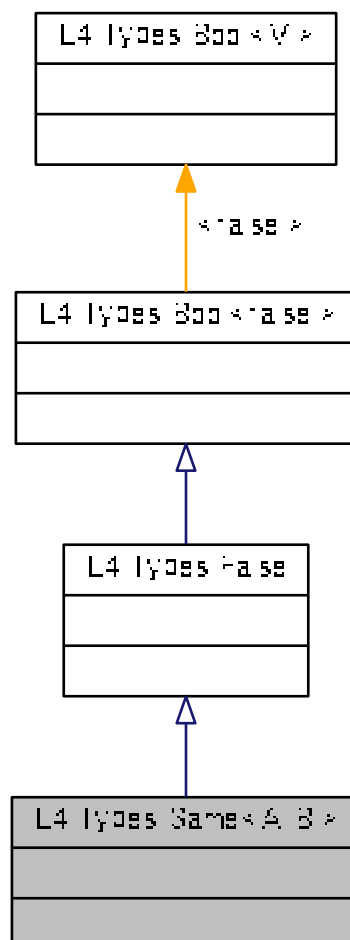
The documentation for this class was generated from the following file:

- [l4/sys/cxx/types](#)

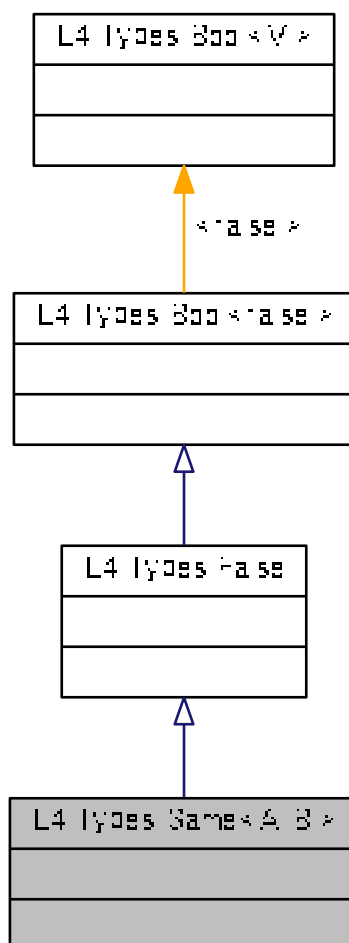
14.188 L4::Types::Same< A, B > Struct Template Reference

Compare two data types for equality.

Inheritance diagram for L4::Types::Same< A, B >:



Collaboration diagram for L4::Types::Same< A, B >:



Additional Inherited Members

14.188.1 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 [189](#) of file [types](#).

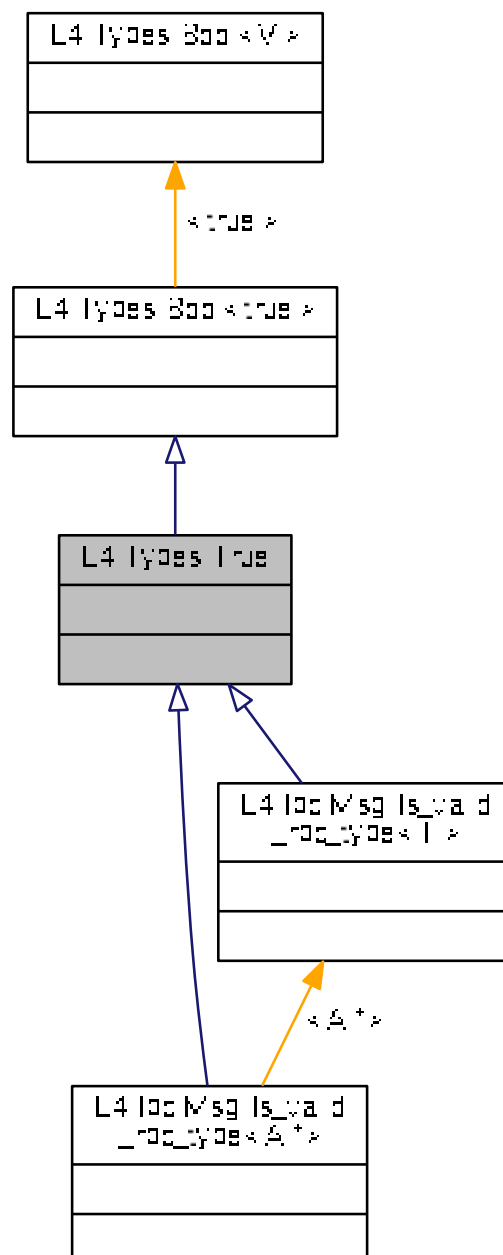
The documentation for this struct was generated from the following file:

- [l4/sys/cxx/types](#)

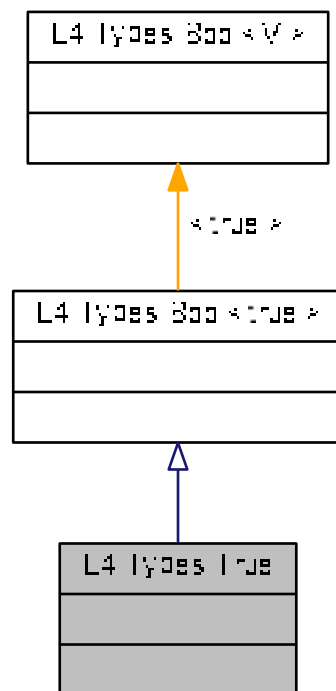
14.189 L4::Types::True Struct Reference

[True](#) meta value.

Inheritance diagram for L4::Types::True:



Collaboration diagram for L4::Types::True:



Additional Inherited Members

14.189.1 Detailed Description

[True](#) meta value.

Definition at line 177 of file [types](#).

The documentation for this struct was generated from the following file:

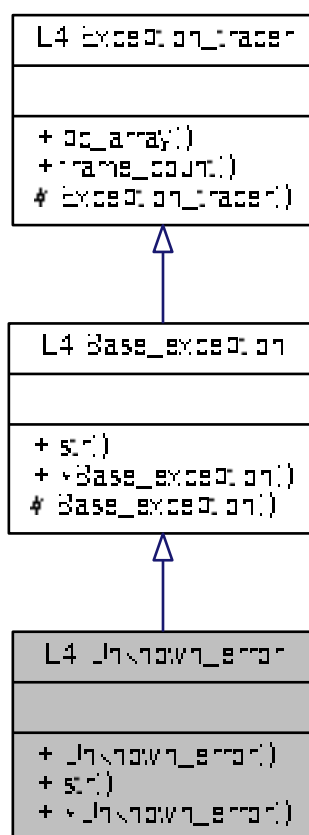
- [l4/sys/cxx/types](#)

14.190 L4::Unknown_error Class Reference

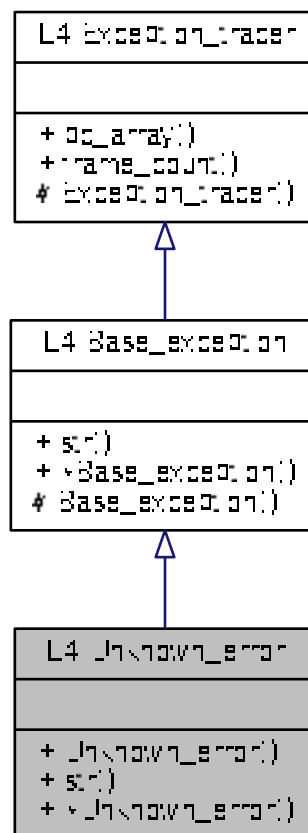
Exception for an unknown condition.

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Unknown_error:



Collaboration diagram for L4::Unknown_error:



Public Member Functions

- `char const * str () const throw ()`
Return a human readable string for the exception.

Additional Inherited Members

14.190.1 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 219 of file [exceptions](#).

The documentation for this class was generated from the following file:

- [l4/cxx/exceptions](#)

C++ L4 Vcon interface.

```

classDiagram
    class L4Rs_Log {
        + send()
        + recv()
    }
    class L4Rs_Log {
        + send()
        + recv()
    }
    class L4_Abstract_Log_L4_Vcon_L4_PROTO_divPIV {
        # c()
        # __create_protocols__()
    }
    class L4_Vcon {
        + send()
        + write()
        + read()
        + read_write_tags()
        + set_attr()
        + get_attr()
    }
    class L4_Abstract_Vcon_lcu_L4_PROTO_LOG {
        # c()
        # __create_protocols__()
    }
    class L4_lcu {
        + send()
        + write()
        + read()
        + read_write_tags()
        + set_attr()
        + get_attr()
    }
    class L4_Abstract_lcu_lcu_L4_PROTO_log_lcu_Demand_1 {
        # c()
        # __create_protocols__()
    }
    class L4_req_ua {
        + write()
        + read()
    }
    class L4_Abstract_Derived_Base_PROTO_S_DEMAND {
        # c()
        # __create_protocols__()
    }

    L4Rs_Log --> L4Rs_Log
    L4Rs_Log --> L4_Abstract_Log_L4_Vcon_L4_PROTO_divPIV
    L4_Abstract_Log_L4_Vcon_L4_PROTO_divPIV --> L4_Vcon
    L4_Vcon --> L4_Abstract_Vcon_lcu_L4_PROTO_LOG
    L4_Abstract_Vcon_lcu_L4_PROTO_LOG --> L4_lcu
    L4_lcu --> L4_Abstract_lcu_lcu_L4_PROTO_log_lcu_Demand_1
    L4_Abstract_lcu_lcu_L4_PROTO_log_lcu_Demand_1 --> L4_req_ua
    L4_Abstract_lcu_lcu_L4_PROTO_log_lcu_Demand_1 --> L4_Abstract_Derived_Base_PROTO_S_DEMAND
  
```


[illegible]

- `l4_msgtag_t send` (char const *buf, unsigned size, `l4_utcb_t *utcb=l4_utcb()`) const throw ()
*Send data to **this** virtual console.*
- `long write` (char const *buf, unsigned size, `l4_utcb_t *utcb=l4_utcb()`) const throw ()
*Write data to **this** virtual console.*
- `int read` (char *buf, unsigned size, `l4_utcb_t *utcb=l4_utcb()`) const throw ()

Read data from *this* virtual console.

- `int read_with_flags (char *buf, unsigned size, l4_utcb_t *utcb=l4_utcb()) const throw ()`

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 throw ()`

Set the attributes of *this* virtual console.

- `l4_msgtag_t get_attr (l4_vcon_attr_t *attr, l4_utcb_t *utcb=l4_utcb()) const throw ()`

Get attributes of *this* virtual console.

Additional Inherited Members

14.191.1 Detailed Description

C++ [L4 Vcon](#) 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.

Include File

```
#include <l4/sys/vcon>
```

See the [Virtual Console](#) for the C interface.

Definition at line [43](#) of file [vcon](#).

14.191.2 Member Function Documentation

14.191.2.1 get_attr()

```
l4_msgtag_t L4::Vcon::get_attr (
    l4_vcon_attr_t * attr,
    l4_utcb_t * utcb = l4_utcb() ) const throw ()    [inline]
```

Get attributes of *this* virtual console.

Parameters

out	<i>attr</i>	Attribute structure. Contains the attributes after a successfull call of this function.
	<i>utcb</i>	UTCB pointer of the calling thread.

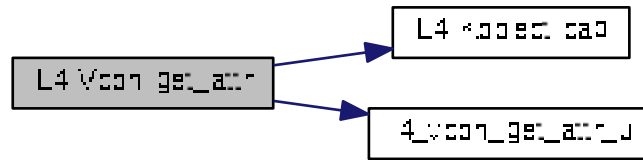
Returns

Syscall return tag.

Definition at line [145](#) of file [vcon](#).

References [L4::Kobject::cap\(\)](#), and [l4_vcon_get_attr_u\(\)](#).

Here is the call graph for this function:



14.191.2.2 read()

```
int L4::Vcon::read (
    char * buf,
    unsigned size,
    l4_utcb_t * utcb = l4_utcb() ) const throw ()    [inline]
```

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 pointer of the calling thread.

Return values

<0	Error code.
> <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.

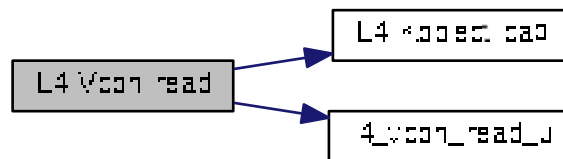
Note

Size must not exceed [L4_VCON_READ_SIZE](#).

Definition at line 94 of file [vcon](#).

References [L4::Kobject::cap\(\)](#), and [l4_vcon_read_u\(\)](#).

Here is the call graph for this function:



14.191.2.3 read_with_flags()

```

int L4::Vcon::read_with_flags (
    char * buf,
    unsigned size,
    l4_utcb_t * utcb = l4_utcb() ) const throw()    [inline]
  
```

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 pointer of the calling thread.

Return values

<0	Error code.
> <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.

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 [119](#) of file [vcon](#).

14.191.2.4 send()

```
l4_msgtag_t L4::Vcon::send (
    char const * buf,
    unsigned size,
    l4_utcb_t * utcb = l4_utcb() ) const throw ()    [inline]
```

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>	UTBC pointer of the calling thread.

Returns

Syscall return tag

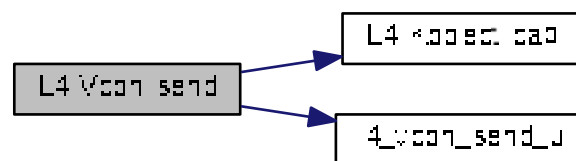
Note

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 63 of file [vcon](#).

References [L4::Kobject::cap\(\)](#), and [l4_vcon_send_u\(\)](#).

Here is the call graph for this function:



14.191.2.5 set_attr()

```
l4_msgtag_t L4::Vcon::set_attr (
    l4_vcon_attr_t const * attr,
    l4_utcb_t * utcb = l4_utcb() ) const throw ()    [inline]
```

Set the attributes of this virtual console.

Parameters

<i>attr</i>	Attribute structure with the attributes for the virtual console.
<i>utcb</i>	UTCB pointer of the calling thread.

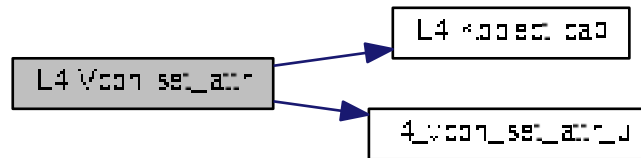
Returns

Syscall return tag.

Definition at line 132 of file [vcon](#).

References [L4::Kobject::cap\(\)](#), and [l4_vcon_set_attr_u\(\)](#).

Here is the call graph for this function:



14.191.2.6 write()

```

long L4::Vcon::write (
    char const * buf,
    unsigned size,
    l4_utcb_t * utcb = l4_utcb() ) const throw()    [inline]
  
```

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 pointer of the calling thread.

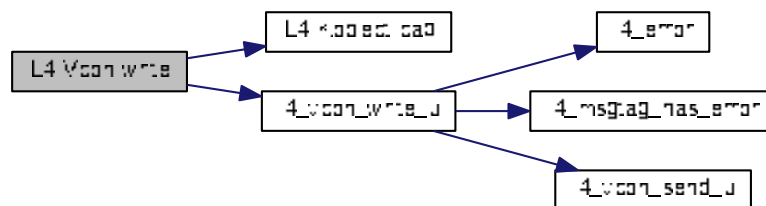
Return values

<0	Error.
>=0	Number of bytes written to the virtual console.

Definition at line 77 of file [vcon](#).

References [L4::Kobject::cap\(\)](#), and [l4_vcon_write_u\(\)](#).

Here is the call graph for this function:



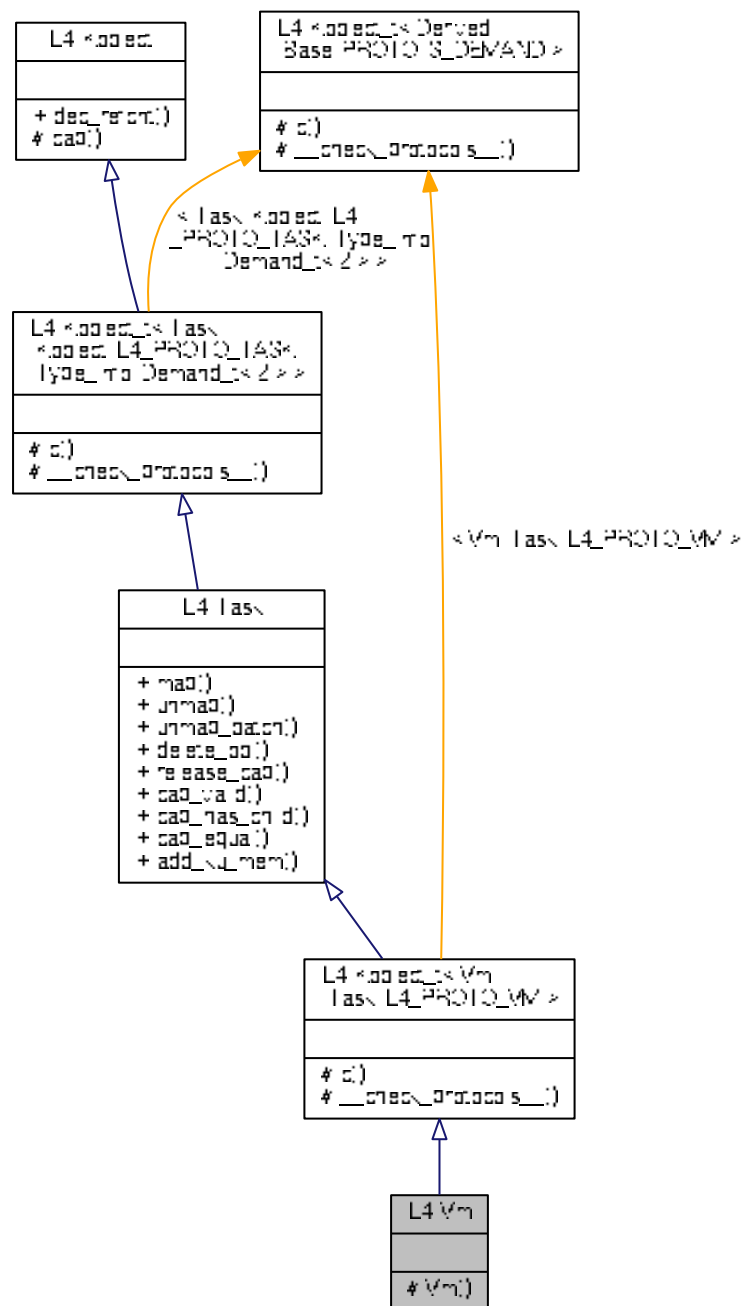
The documentation for this class was generated from the following file:

- [l4/sys/vcon](#)

14.192 L4::Vm Class Reference

Virtual machine.

Collaboration diagram for L4::Vm:



Additional Inherited Members

14.192.1 Detailed Description

Virtual machine.

Definition at line 35 of file [vm](#).

The documentation for this class was generated from the following file:

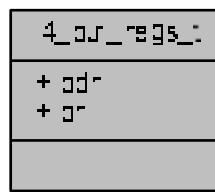
- [l4/sys/vm](#)

14.193 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.

14.193.1 Detailed Description

Encapsulation of the buffer-registers block in the UTCB.

Definition at line 93 of file [utcb.h](#).

The documentation for this struct was generated from the following file:

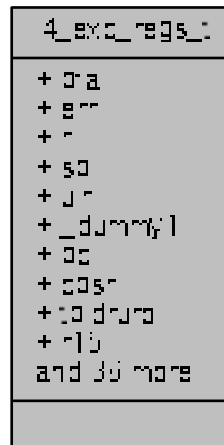
- [l4/sys/utcb.h](#)

14.194 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
- [l4_umword_t pc](#)
pc
- [l4_umword_t cpsr](#)
cpsr
- [l4_umword_t tpidruro](#)
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 ip](#)
instruction pointer
- [l4_umword_t dummy1](#)
dummy
- [l4_umword_t flags](#)
rflags
- [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

14.194.1 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 38 of file [utcb.h](#).

14.194.2 Field Documentation

14.194.2.1 flags

[l4_umword_t](#) l4_exc_regs_t::flags

rflags

eflags

Definition at line 80 of file [utcb.h](#).

14.194.2.2 ss

[l4_umword_t](#) l4_exc_regs_t::ss

stack segment register

ss register

Definition at line 82 of file [utcb.h](#).

The documentation for this struct was generated from the following file:

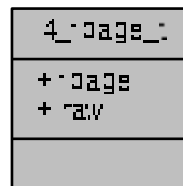
- [arm/l4/sys/utcb.h](#)

14.195 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.

14.195.1 Detailed Description

L4 flexpage type.

Definition at line 81 of file [__l4_fpage.h](#).

The documentation for this union was generated from the following file:

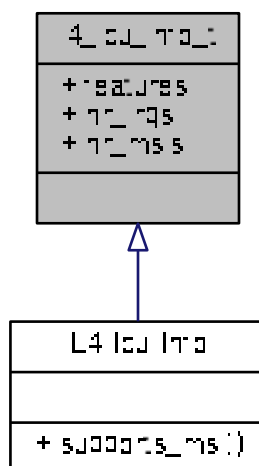
- l4/sys/__l4_fpage.h

14.196 l4_icu_info_t Struct Reference

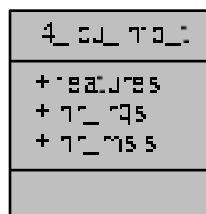
Info structure for an ICU.

```
#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,.

14.196.1 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 159 of file [icu.h](#).

14.196.2 Field Documentation

14.196.2.1 features

```
unsigned l4_icu_info_t::features
```

Feature flags.

If [L4_ICU_FLAG_MSI](#) is set the ICU supports MSIs.

Definition at line 166 of file [icu.h](#).

The documentation for this struct was generated from the following file:

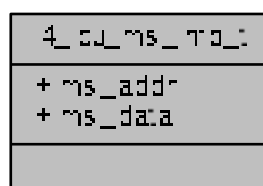
- [l4/sys/icu.h](#)

14.197 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.

14.197.1 Detailed Description

Info to use for a specific MSI.

Definition at line 180 of file [icu.h](#).

14.197.2 Field Documentation

14.197.2.1 msi_data

```
l4\_uint32\_t l4_icu_msi_info_t::msi_data
```

Value to use as data written to msi_addr, when sending this MSI.

Definition at line 185 of file [icu.h](#).

The documentation for this struct was generated from the following file:

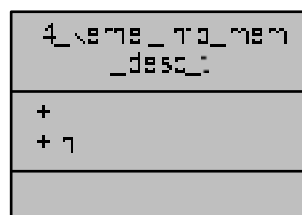
- [l4/sys/icu.h](#)

14.198 l4_kernel_info_mem_desc_t Struct Reference

Memory descriptor data structure.

```
#include <memdesc.h>
```

Collaboration diagram for `l4_kernel_info_mem_desc_t`:



14.198.1 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 74 of file [memdesc.h](#).

The documentation for this struct was generated from the following file:

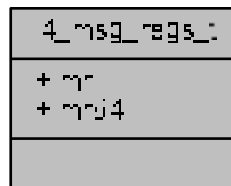
- [l4/sys/memdesc.h](#)

14.199 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.

14.199.1 Detailed Description

Encapsulation of the message-register block in the UTCB.

Examples:

[examples/sys/utcb-ipc/main.c](#).

Definition at line 78 of file [utcb.h](#).

The documentation for this union was generated from the following file:

- [l4/sys/utcb.h](#)

14.200 l4_msgtag_t Struct Reference

Message tag data structure.

```
#include <types.h>
```

Collaboration diagram for l4_msgtag_t:

l4_msgtag_t
+ raw
+ add ()
+ add ()
+ words ()
+ items ()
+ flags ()
+ is_page_fault ()
+ is_preemption ()
+ is_sys_exception ()
+ is_exception ()
+ is_sigma ()
+ is_io_page_fault ()
+ has_error ()

Public Member Functions

- long [label](#) () const throw ()
Get the protocol value.
- void [label](#) (long v) throw ()
Set the protocol value.
- unsigned [words](#) () const throw ()
Get the number of untyped words.
- unsigned [items](#) () const throw ()
Get the number of typed items.
- unsigned [flags](#) () const throw ()
Get the flags value.
- bool [is_page_fault](#) () const throw ()
Test if protocol indicates page-fault protocol.
- bool [is_preemption](#) () const throw ()
Test if protocol indicates preemption protocol.
- bool [is_sys_exception](#) () const throw ()
Test if protocol indicates system-exception protocol.
- bool [is_exception](#) () const throw ()
Test if protocol indicates exception protocol.
- bool [is_sigma0](#) () const throw ()
Test if protocol indicates sigma0 protocol.
- bool [is_io_page_fault](#) () const throw ()
Test if protocol indicates IO-page-fault protocol.
- unsigned [has_error](#) () const throw ()
Test if flags indicate an error.

14.201 l4_sched_cpu_set_t Struct Reference

CPU sets.

```
#include <scheduler.h>
```

Collaboration diagram for l4_sched_cpu_set_t:

<code>l4_sched_cpu_set_t</code>
<ul style="list-style-type: none"> + <code>gran_offset</code> + <code>map</code>
<ul style="list-style-type: none"> + <code>granularity()</code> + <code>offset()</code> + <code>set()</code>

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.

14.201.1 Detailed Description

CPU sets.

Examples:

[examples/sys/migrate/thread_migrate.cc](#).

Definition at line [44](#) of file [scheduler.h](#).

14.201.2 Member Function Documentation

14.201.2.1 granularity()

```
unsigned char l4_sched_cpu_set_t::granularity ( ) const [inline]
```

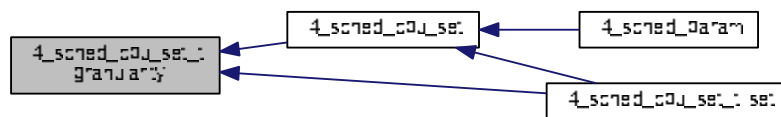
Returns

Get granularity value

Definition at line 66 of file [scheduler.h](#).

Referenced by [l4_sched_cpu_set\(\)](#), and [set\(\)](#).

Here is the caller graph for this function:



14.201.2.2 offset()

```
unsigned l4_sched_cpu_set_t::offset ( ) const [inline]
```

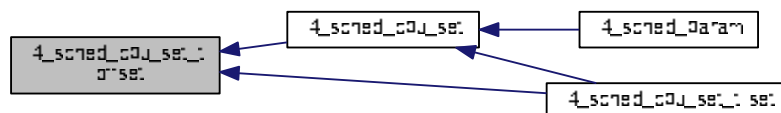
Returns

Get offset value

Definition at line 68 of file [scheduler.h](#).

Referenced by [l4_sched_cpu_set\(\)](#), and [set\(\)](#).

Here is the caller graph for this function:



14.201.3 Field Documentation

14.201.3.1 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 numer 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 57 of file [scheduler.h](#).

Referenced by [L4::Scheduler::info\(\)](#), and [l4_sched_cpu_set\(\)](#).

The documentation for this struct was generated from the following file:

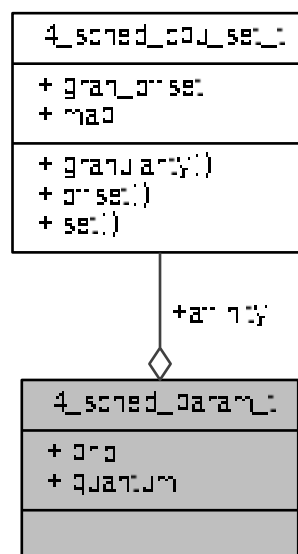
- [l4/sys/scheduler.h](#)

14.202 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.

14.202.1 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 119 of file [scheduler.h](#).

The documentation for this struct was generated from the following file:

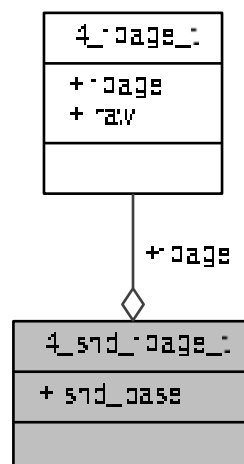
- [l4/sys/scheduler.h](#)

14.203 l4_snd_fpage_t Struct Reference

Send-flex-page 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 flex-page descriptor.

14.203.1 Detailed Description

Send-flex-page types.

Definition at line 98 of file [__l4_fpage.h](#).

The documentation for this struct was generated from the following file:

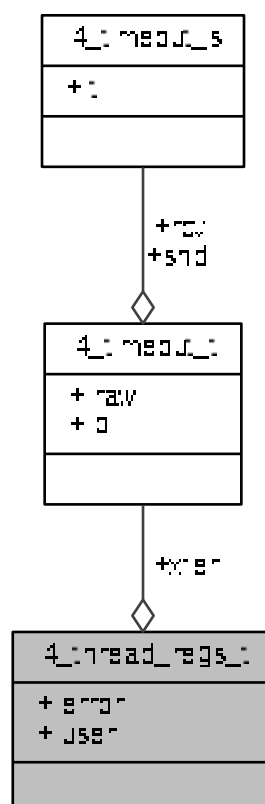
- l4/sys/__l4_fpage.h

14.204 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 codes.
- [l4_timeout_t xfer](#)
Message transfer timeout.
- [l4_umword_t user](#) [3]
User values (ignored and preserved by the kernel)

14.204.1 Detailed Description

Encapsulation of the thread-control-register block of the UTCB.

Definition at line 110 of file [utcb.h](#).

The documentation for this struct was generated from the following file:

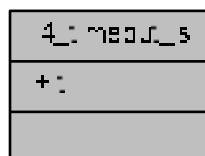
- [l4/sys/utcb.h](#)

14.205 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

14.205.1 Detailed Description

Basic timeout specification.

Basically a floating point number with 10 bits mantissa and 5 bits exponent ($t = m \cdot 2^e$).

The timeout can also specify an absolute point in time (bit 16 == 1).

Definition at line 45 of file [__timeout.h](#).

The documentation for this struct was generated from the following file:

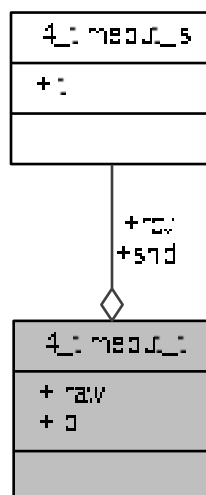
- l4/sys/__timeout.h

14.206 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*

### 14.206.1 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 57 of file [\\_\\_timeout.h](#).

The documentation for this union was generated from the following file:

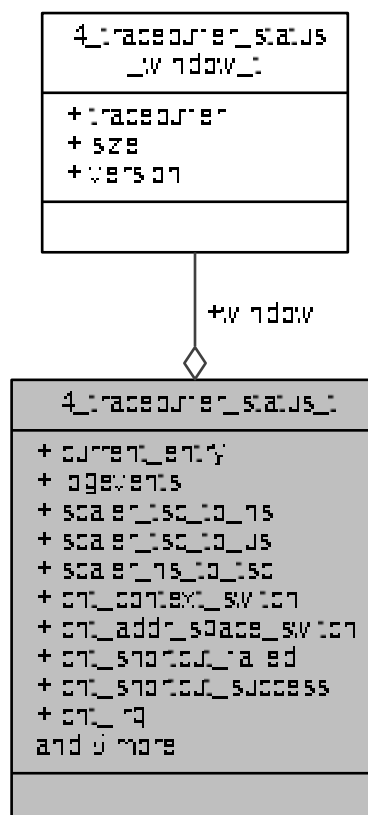
- l4/sys/\_\_timeout.h

### 14.207 l4\_tracebuffer\_status\_t Struct Reference

Trace-buffer status.

```
#include <ktrace.h>
```

Collaboration diagram for l4\_tracebuffer\_status\_t:



## Data Fields

- volatile `l4_tracebuffer_entry_t` \* `current_entry`  
*Address of the most current event in trace-buffer.*
- `l4_uint32_t` `logevents` [`LOG_EVENT_MAX_EVENTS`]  
*Available LOG events.*
- `l4_uint32_t` `scaler_tsc_to_ns`  
*Scaler used for translation of CPU cycles to nano seconds.*
- `l4_uint32_t` `scaler_tsc_to_us`  
*Scaler used for translation of CPU cycles to micro seconds.*
- `l4_uint32_t` `scaler_ns_to_tsc`  
*Scaler used for translation of nano seconds to CPU cycles.*
- volatile `l4_uint32_t` `cnt_context_switch`  
*Number of context switches (intra AS or inter AS)*
- volatile `l4_uint32_t` `cnt_addr_space_switch`  
*Number of inter AS context switches.*
- volatile `l4_uint32_t` `cnt_shortcut_failed`  
*How often was the IPC shortcut taken.*
- volatile `l4_uint32_t` `cnt_shortcut_success`  
*How often was the IPC shortcut not taken.*
- volatile `l4_uint32_t` `cnt_irq`  
*Number of hardware interrupts (without kernel scheduling interrupt)*
- volatile `l4_uint32_t` `cnt_ipc_long`  
*Number of long IPCs.*
- volatile `l4_uint32_t` `cnt_page_fault`  
*Number of page faults.*
- volatile `l4_uint32_t` `cnt_io_fault`  
*Number of faults (application runs at IOPL 0 and tries to execute cli, sti, in, or out but does not have a sufficient right in the I/O bitmap)*
- volatile `l4_uint32_t` `cnt_task_create`  
*Number of tasks created.*
- volatile `l4_uint32_t` `cnt_schedule`  
*Number of reschedules.*
- volatile `l4_uint32_t` `cnt_jobmap_tlb_flush`  
*Number of flushes of the I/O bitmap.*

### 14.207.1 Detailed Description

Trace-buffer status.

Definition at line 67 of file [ktrace.h](#).

### 14.207.2 Field Documentation

### 14.207.2.1 cnt\_iobmap\_tlb\_flush

```
volatile l4_uint32_t l4_tracebuffer_status_t::cnt_iobmap_tlb_flush
```

Number of flushes of the I/O bitmap.

Increases on context switches between two small address spaces if at least one of the spaces has an I/O bitmap allocated.

Definition at line 106 of file [ktrace.h](#).

The documentation for this struct was generated from the following file:

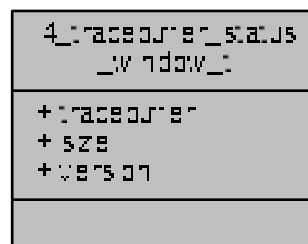
- [l4/sys/ktrace.h](#)

## 14.208 l4\_tracebuffer\_status\_window\_t Struct Reference

Trace-buffer status window descriptor.

```
#include <ktrace.h>
```

Collaboration diagram for `l4_tracebuffer_status_window_t`:



### Data Fields

- `l4_tracebuffer_entry_t * tracebuffer`  
*Address of trace-buffer.*
- `l4_umword_t size`  
*Size of trace-buffer.*
- `volatile l4_uint64_t version`  
*Version number of trace-buffer (incremented if trace-buffer overruns)*

### 14.208.1 Detailed Description

Trace-buffer status window descriptor.

Definition at line 52 of file [ktrace.h](#).

The documentation for this struct was generated from the following file:

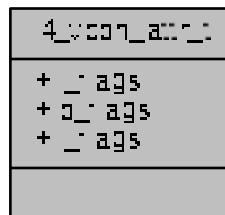
- [l4/sys/ktrace.h](#)

## 14.209 l4\_vcon\_attr\_t Struct Reference

Vcon attribute structure.

```
#include <vcon.h>
```

Collaboration diagram for l4\_vcon\_attr\_t:



### Data Fields

- [l4\\_umword\\_t i\\_flags](#)  
*input flags*
- [l4\\_umword\\_t o\\_flags](#)  
*output flags*
- [l4\\_umword\\_t l\\_flags](#)  
*local flags*

### 14.209.1 Detailed Description

Vcon attribute structure.

Definition at line 178 of file [vcon.h](#).

The documentation for this struct was generated from the following file:

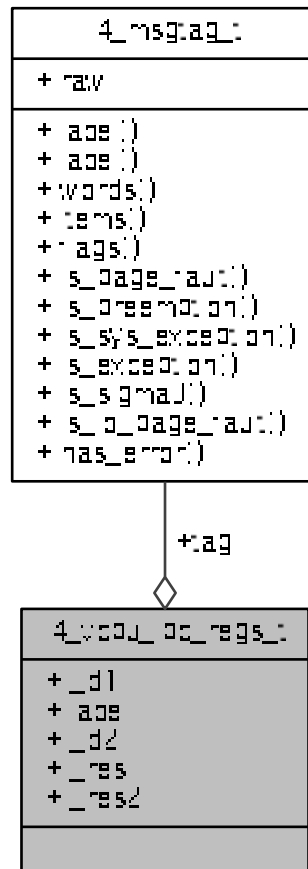
- [l4/sys/vcon.h](#)

## 14.210 l4\_vcpu\_ipc\_regs\_t Struct Reference

vCPU message registers.

```
#include <__vcpu-arch.h>
```

Collaboration diagram for l4\_vcpu\_ipc\_regs\_t:



### 14.210.1 Detailed Description

vCPU message registers.

Definition at line 62 of file [\\_\\_vcpu-arch.h](#).

The documentation for this struct was generated from the following file:

- arm/l4/sys/\_\_vcpu-arch.h

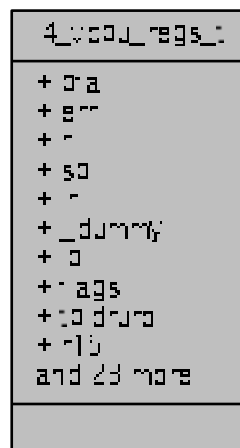


## 14.211 l4\_vcpu\_regs\_t Struct Reference

vCPU registers.

```
#include <__vcpu-arch.h>
```

Collaboration diagram for l4\_vcpu\_regs\_t:



### 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 eflags](#)  
*eflags*
- [l4\\_umword\\_t tpidruro](#)  
*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 rdi](#)  
*rdi register*
- [l4\\_umword\\_t rsi](#)  
*rsi register*
- [l4\\_umword\\_t rbp](#)  
*rbp register*
- [l4\\_umword\\_t rbx](#)  
*rbx register*
- [l4\\_umword\\_t rdx](#)  
*rdx register*
- [l4\\_umword\\_t rcx](#)  
*rcx register*
- [l4\\_umword\\_t rax](#)  
*rax register*
- [l4\\_umword\\_t trapno](#)  
*trap number*
- [l4\\_umword\\_t cs](#)  
*dummy*
- [l4\\_umword\\_t ss](#)  
*ss register*
- [l4\\_umword\\_t es](#)  
*gs register*
- [l4\\_umword\\_t ds](#)  
*fs register*
- [l4\\_umword\\_t gs](#)  
*gs register*
- [l4\\_umword\\_t fs](#)  
*fs register*
- [l4\\_umword\\_t dummy1](#)  
*dummy*

### 14.211.1 Detailed Description

vCPU registers.

Definition at line 38 of file [\\_\\_vcpu-arch.h](#).

### 14.211.2 Field Documentation

#### 14.211.2.1 ax

`l4_umword_t l4_vcpu_regs_t::ax`

rax register

eax register

Definition at line 68 of file [\\_\\_vcpu-arch.h](#).

#### 14.211.2.2 bp

`l4_umword_t l4_vcpu_regs_t::bp`

rbp register

ebp register

Definition at line 63 of file [\\_\\_vcpu-arch.h](#).

#### 14.211.2.3 bx

`l4_umword_t l4_vcpu_regs_t::bx`

rbx register

ebx register

Definition at line 65 of file [\\_\\_vcpu-arch.h](#).

#### 14.211.2.4 cx

`l4_umword_t l4_vcpu_regs_t::cx`

rcx register

ecx register

Definition at line 67 of file [\\_\\_vcpu-arch.h](#).

#### 14.211.2.5 di

`l4_umword_t l4_vcpu_regs_t::di`

rdi register

edi register

Definition at line 61 of file [\\_\\_vcpu-arch.h](#).

#### 14.211.2.6 dx

`l4_umword_t l4_vcpu_regs_t::dx`

rdx register

edx register

Definition at line 66 of file [\\_\\_vcpu-arch.h](#).

#### 14.211.2.7 si

`l4_umword_t l4_vcpu_regs_t::si`

rsi register

esi register

Definition at line 62 of file [\\_\\_vcpu-arch.h](#).

The documentation for this struct was generated from the following file:

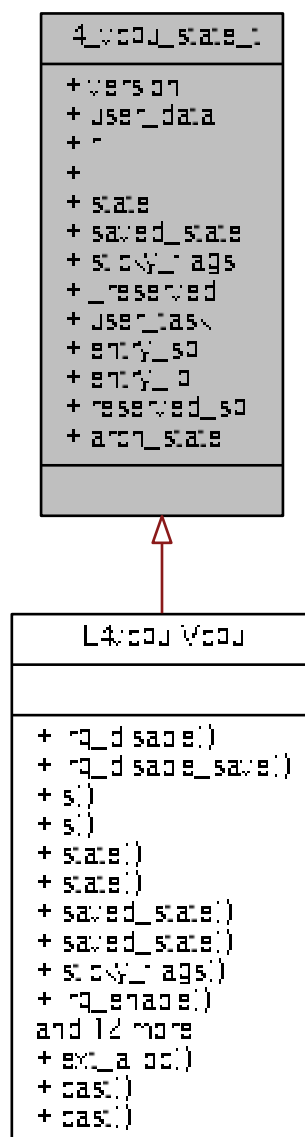
- [arm/l4/sys/\\_\\_vcpu-arch.h](#)

## 14.212 l4\_vcpu\_state\_t Struct Reference

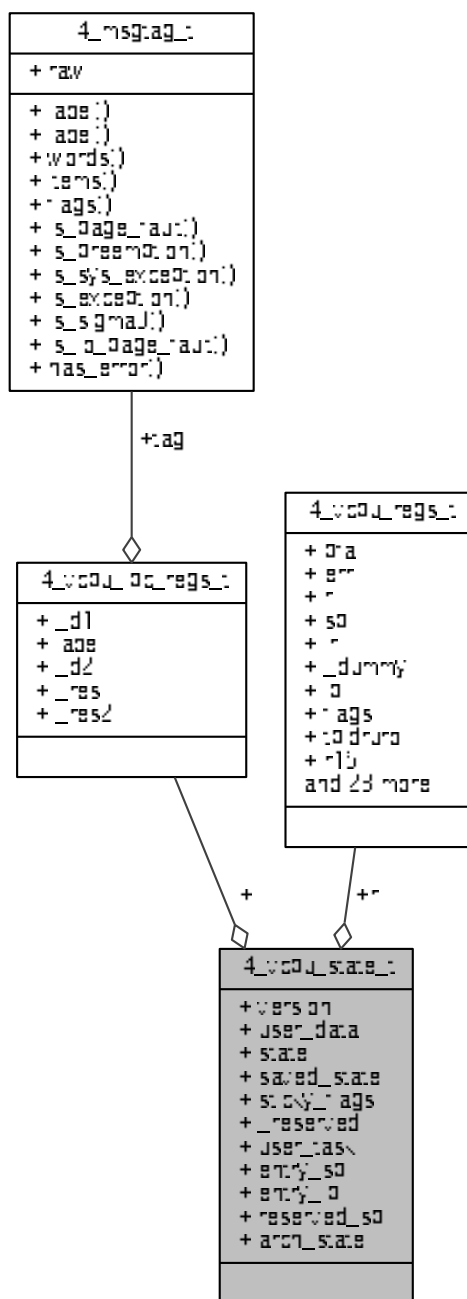
State of a vCPU.

```
#include <vcpu.h>
```

Inheritance diagram for l4\_vcpu\_state\_t:



Collaboration diagram for `l4_vcpu_state_t`:



## Data Fields

- [l4\\_vcpu\\_regs\\_t r](#)  
*Register state.*
- [l4\\_vcpu\\_ipc\\_regs\\_t i](#)  
*IPC state.*
- [l4\\_uint16\\_t state](#)

*Current vCPU state.*

- [l4\\_uint16\\_t saved\\_state](#)

*Saved vCPU state.*

- [l4\\_uint16\\_t sticky\\_flags](#)

*Pending 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.*

### 14.212.1 Detailed Description

State of a vCPU.

Definition at line 34 of file [vcpu.h](#).

The documentation for this struct was generated from the following file:

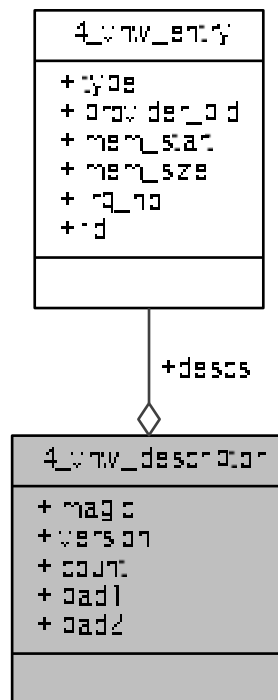
- [l4/sys/vcpu.h](#)

## 14.213 l4\_vhw\_descriptor Struct Reference

Virtual hardware devices description.

```
#include <vhw.h>
```

Collaboration diagram for `l4_vhw_descriptor`:



## Data Fields

- `l4_uint32_t magic`  
*Magic.*
- `l4_uint8_t version`  
*Version of the descriptor.*
- `l4_uint8_t count`  
*Number of entries.*
- `l4_uint8_t pad1`  
*padding*
- `l4_uint8_t pad2`  
*padding*
- `struct l4_vhw_entry desc[]`  
*Array of device descriptions.*

### 14.213.1 Detailed Description

Virtual hardware devices description.

Examples:

[examples/sys/ux-vhw/main.c](#).

Definition at line 70 of file [vhw.h](#).



## 14.213.2 Field Documentation

### 14.213.2.1 count

```
l4_uint8_t l4_vhw_descriptor::count
```

Number of entries.

Examples:

[examples/sys/ux-vhw/main.c](#).

Definition at line 73 of file [vhw.h](#).

### 14.213.2.2 descs

```
struct l4_vhw_entry l4_vhw_descriptor::descs[]
```

Array of device descriptions.

Definition at line 77 of file [vhw.h](#).

### 14.213.2.3 magic

```
l4_uint32_t l4_vhw_descriptor::magic
```

Magic.

Examples:

[examples/sys/ux-vhw/main.c](#).

Definition at line 71 of file [vhw.h](#).

#### 14.213.2.4 version

```
l4_uint8_t l4_vhw_descriptor::version
```

Version of the descriptor.

Examples:

[examples/sys/ux-vhw/main.c](#).

Definition at line 72 of file [vhw.h](#).

The documentation for this struct was generated from the following file:

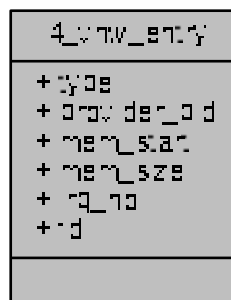
- [l4/sys/vhw.h](#)

## 14.214 l4\_vhw\_entry Struct Reference

Description of a device.

```
#include <vhw.h>
```

Collaboration diagram for l4\_vhw\_entry:



### Data Fields

- enum [l4\\_vhw\\_entry\\_type](#) type  
*Type of virtual hardware.*
- [l4\\_uint32\\_t](#) provider\_pid  
*Host PID of the VHW provider.*
- [l4\\_addr\\_t](#) mem\_start  
*Start of memory region.*
- [l4\\_addr\\_t](#) mem\_size  
*Size of memory region.*
- [l4\\_uint32\\_t](#) irq\_no  
*IRQ number.*
- [l4\\_uint32\\_t](#) fd  
*File descriptor.*

### 14.214.1 Detailed Description

Description of a device.

Examples:

[examples/sys/ux-vhw/main.c](#).

Definition at line 55 of file [vhw.h](#).

### 14.214.2 Field Documentation

#### 14.214.2.1 fd

```
l4_uint32_t l4_vhw_entry::fd
```

File descriptor.

Definition at line 63 of file [vhw.h](#).

#### 14.214.2.2 irq\_no

```
l4_uint32_t l4_vhw_entry::irq_no
```

IRQ number.

Examples:

[examples/sys/ux-vhw/main.c](#).

Definition at line 62 of file [vhw.h](#).

#### 14.214.2.3 mem\_size

```
l4_addr_t l4_vhw_entry::mem_size
```

Size of memory region.

Examples:

[examples/sys/ux-vhw/main.c](#).

Definition at line 60 of file [vhw.h](#).

#### 14.214.2.4 mem\_start

```
l4_addr_t l4_vhw_entry::mem_start
```

Start of memory region.

Examples:

[examples/sys/ux-vhw/main.c](#).

Definition at line 59 of file [vhw.h](#).

#### 14.214.2.5 provider\_pid

```
l4_uint32_t l4_vhw_entry::provider_pid
```

Host PID of the VHW provider.

Examples:

[examples/sys/ux-vhw/main.c](#).

Definition at line 57 of file [vhw.h](#).

#### 14.214.2.6 type

```
enum l4_vhw_entry_type l4_vhw_entry::type
```

Type of virtual hardware.

Examples:

[examples/sys/ux-vhw/main.c](#).

Definition at line 56 of file [vhw.h](#).

The documentation for this struct was generated from the following file:

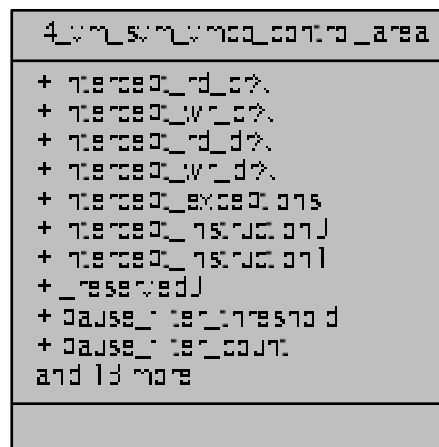
- [l4/sys/vhw.h](#)

### 14.215 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:



### 14.215.1 Detailed Description

VMCB structure for SVM VMs.

Definition at line 39 of file \_\_vm-svm.h.

The documentation for this struct was generated from the following file:

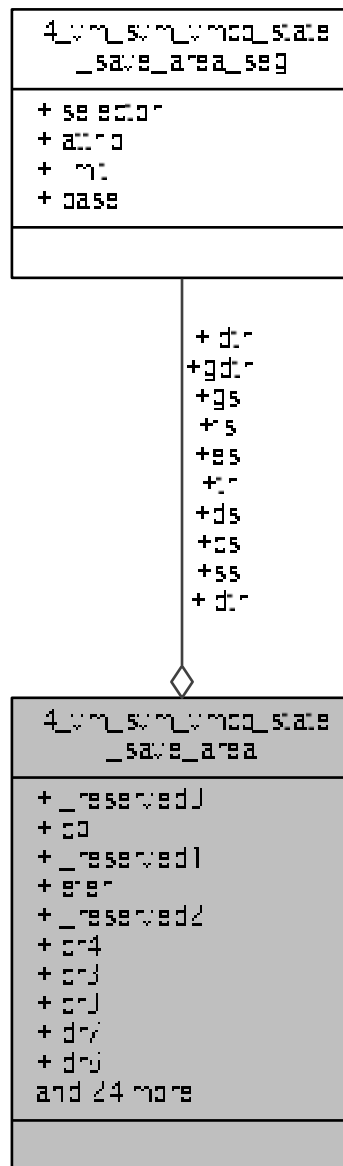
- `l4/sys/__vm-svm.h`

#### 14.216 `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`:



### 14.216.1 Detailed Description

State save area structure for SVM VMs.

Definition at line 96 of file [\\_\\_vm-svm.h](#).

The documentation for this struct was generated from the following file:

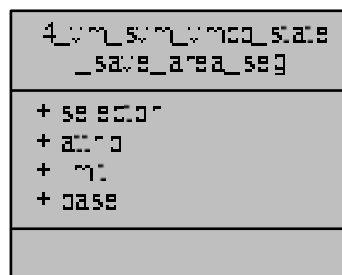
- `l4/sys/__vm-svm.h`

## 14.217 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:



### 14.217.1 Detailed Description

State save area segment selector struct.

Definition at line 84 of file [\\_\\_vm-svm.h](#).

The documentation for this struct was generated from the following file:

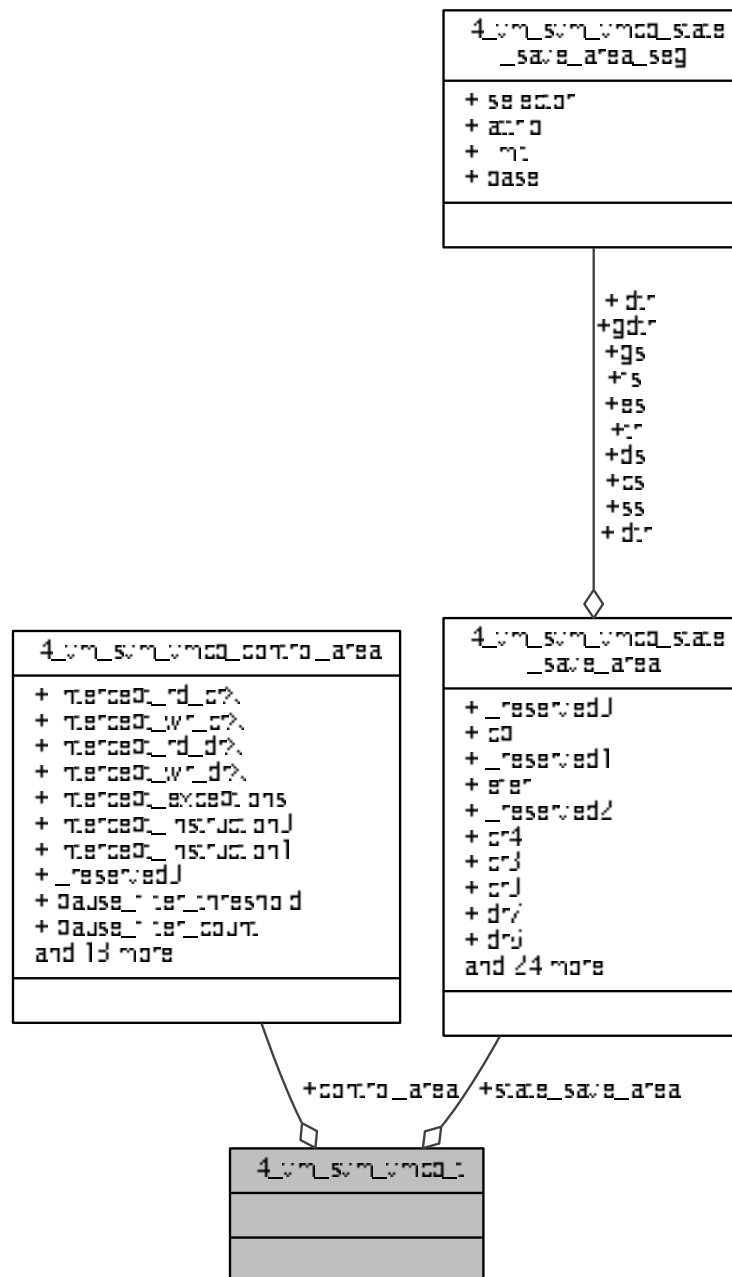
- l4/sys/\_\_vm-svm.h

## 14.218 l4\_vm\_svm\_vmcb\_t Struct Reference

Control structure for SVM VMs.

```
#include <__vm-svm.h>
```

Collaboration diagram for `l4_vm_svm_vmcb_t`:



### 14.218.1 Detailed Description

Control structure for SVM VMs.

Definition at line 165 of file `__vm-svm.h`.

The documentation for this struct was generated from the following file:

- `l4/sys/__vm-svm.h`

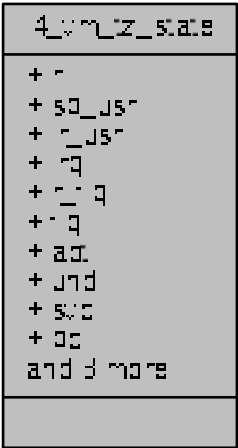


## 14.219 l4\_vm\_tz\_state Struct Reference

state structure for TrustZone VMs

```
#include <vm.h>
```

Collaboration diagram for l4\_vm\_tz\_state:



### 14.219.1 Detailed Description

state structure for TrustZone VMs

Definition at line 52 of file [vm.h](#).

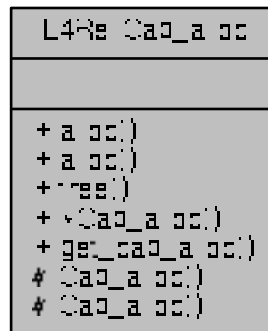
The documentation for this struct was generated from the following file:

- [arm/l4/sys/vm.h](#)

## 14.220 L4Re::Cap\_alloc Class Reference

Capability allocator interface.

Collaboration diagram for L4Re::Cap\_alloc:



## Public Member Functions

- virtual [L4::Cap](#)< void > [alloc](#) ()=0 throw ()  
*Allocate a capability.*
- template<typename T >  
[L4::Cap](#)< T > [alloc](#) () throw ()  
*Allocate a capability.*
- virtual void [free](#) ([L4::Cap](#)< void > cap)=0 throw ()  
*Free a capability.*
- virtual [~Cap\\_alloc](#) ()=0  
*Destructor.*

## Static Public Member Functions

- template<typename CAP\_ALLOC >  
static [L4Re::Cap\\_alloc](#) \* [get\\_cap\\_alloc](#) (CAP\_ALLOC &ca)  
*Construct an instance of a capability allocator.*

### 14.220.1 Detailed Description

Capability allocator interface.

Definition at line 40 of file [cap\\_alloc](#).

### 14.220.2 Member Function Documentation

## 14.220.2.1 alloc() [1/2]

```
virtual L4::Cap<void> L4Re::Cap_alloc::alloc () throw () [pure virtual]
```

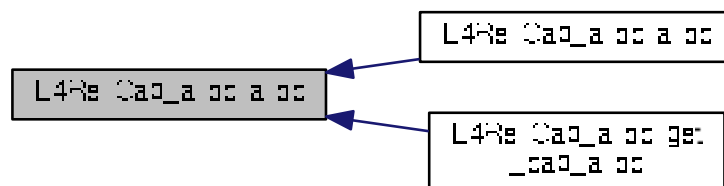
Allocate a capability.

## Returns

Capability of type void

Referenced by [alloc\(\)](#), and [get\\_cap\\_alloc\(\)](#).

Here is the caller graph for this function:



## 14.220.2.2 alloc() [2/2]

```
template<typename T >
L4::Cap<T> L4Re::Cap_alloc::alloc () throw () [inline]
```

Allocate a capability.

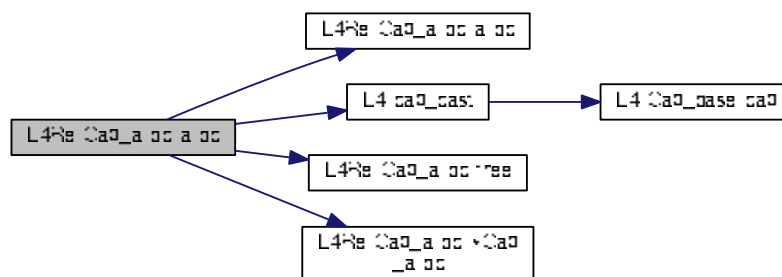
## Returns

Capability of type T

Definition at line 62 of file [cap\\_alloc](#).

References [alloc\(\)](#), [L4::cap\\_cast\(\)](#), [free\(\)](#), and [~Cap\\_alloc\(\)](#).

Here is the call graph for this function:



## 14.220.2.3 free()

```
virtual void L4Re::Cap_alloc::free (
 L4::Cap< void > cap) throw () [pure virtual]
```

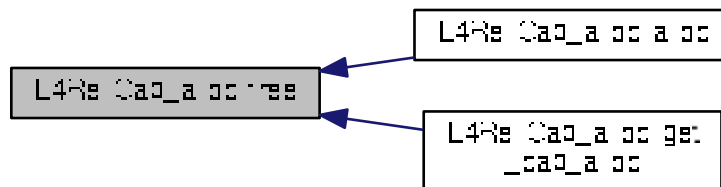
Free a capability.

## Parameters

|            |                     |
|------------|---------------------|
| <i>cap</i> | Capability to free. |
|------------|---------------------|

Referenced by [alloc\(\)](#), and [get\\_cap\\_alloc\(\)](#).

Here is the caller graph for this function:



## 14.220.2.4 get\_cap\_alloc()

```
template<typename CAP_ALLOC >
static L4Re::Cap_alloc* L4Re::Cap_alloc::get_cap_alloc (
 CAP_ALLOC & ca) [inline], [static]
```

Construct an instance of a capability allocator.

## Parameters

|           |                      |
|-----------|----------------------|
| <i>ca</i> | Capability allocator |
|-----------|----------------------|

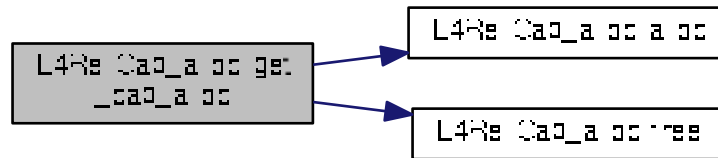
## Returns

Instance of a capability allocator.

Definition at line 85 of file [cap\\_alloc](#).

References [alloc\(\)](#), and [free\(\)](#).

Here is the call graph for this function:



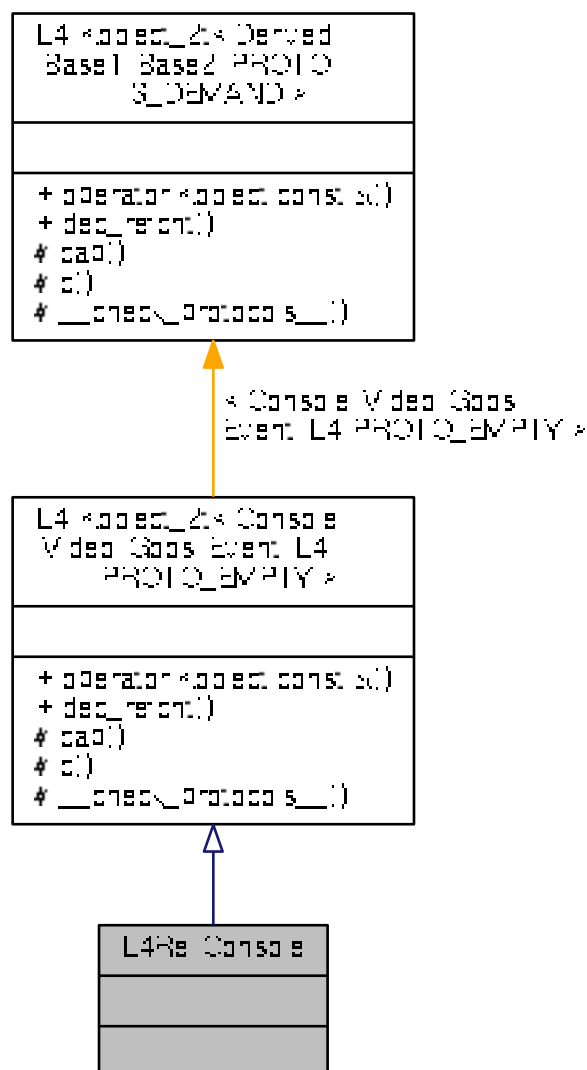
The documentation for this class was generated from the following file:

- [l4/re/cap\\_alloc](#)

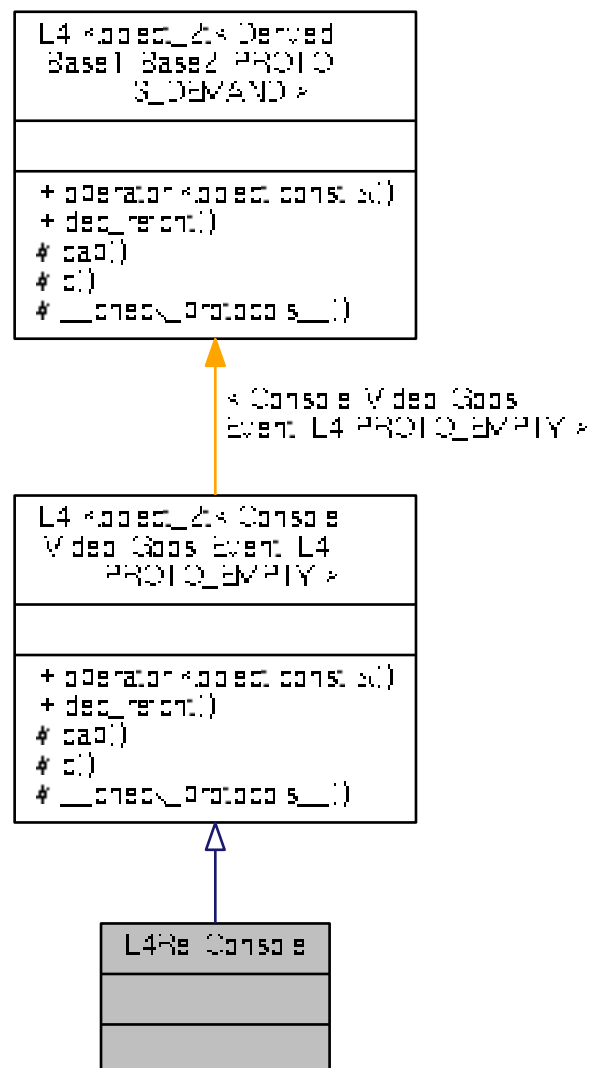
## 14.221 L4Re::Console Class Reference

[Console](#) class.

Inheritance diagram for L4Re::Console:



Collaboration diagram for L4Re::Console:



## Additional Inherited Members

### 14.221.1 Detailed Description

[Console](#) class.

Definition at line 39 of file [console](#).

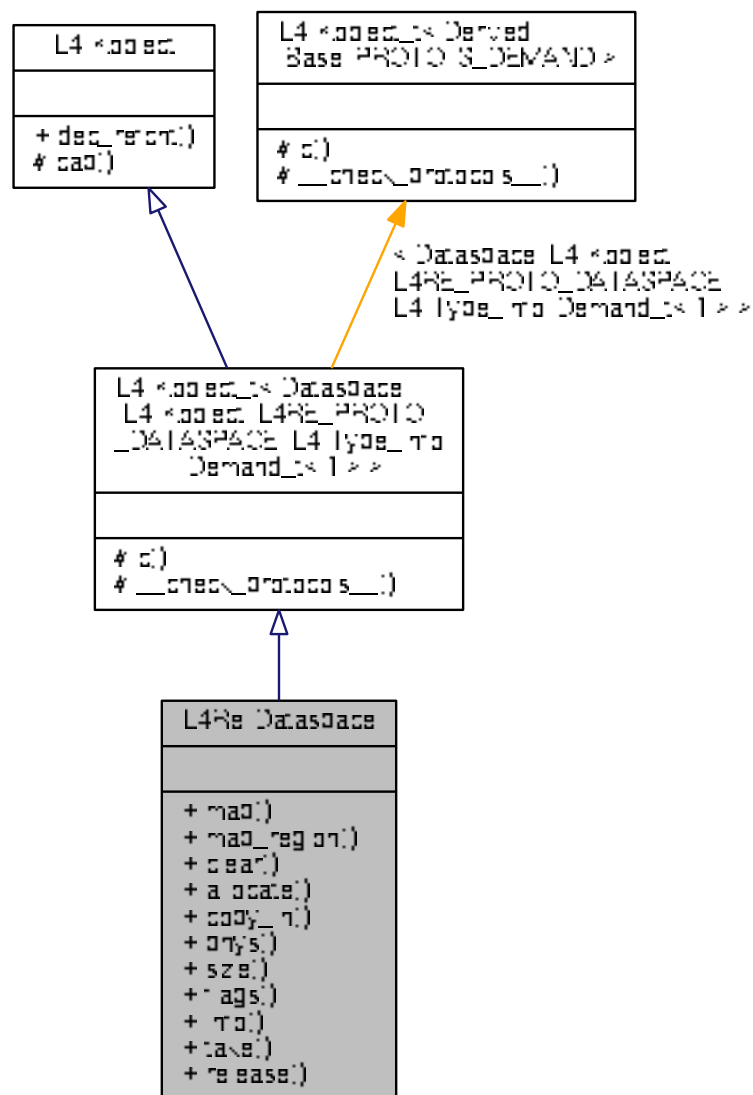
The documentation for this class was generated from the following file:

- [l4/re/console](#)





Collaboration diagram for L4Re::Dataspace:



## Data Structures

- struct [Stats](#)

*Information about the dataspace.*

## Public Types

- enum [Map\\_flags](#) {  
[Map\\_ro](#) = 0, [Map\\_rw](#) = 1, [Map\\_normal](#) = 0x00, [Map\\_cacheable](#) = [Map\\_normal](#),  
[Map\\_bufferable](#) = 0x10, [Map\\_uncacheable](#) = 0x20, [Map\\_caching\\_mask](#) = 0x30, [Map\\_caching\\_shift](#) = 4 }

*Flags for map operations.*

## Public Member Functions

- long [map](#) ([l4\\_addr\\_t](#) offset, unsigned long [flags](#), [l4\\_addr\\_t](#) local\_addr, [l4\\_addr\\_t](#) min\_addr, [l4\\_addr\\_t](#) max\_↵  
addr) const throw ()  
*Request a flex-page mapping from the dataspace.*
- long [map\\_region](#) ([l4\\_addr\\_t](#) offset, unsigned long [flags](#), [l4\\_addr\\_t](#) min\_addr, [l4\\_addr\\_t](#) max\_addr) const throw  
( )  
*Map a part of a dataspace completely.*
- long [clear](#) ([l4\\_addr\\_t](#) offset, unsigned long [size](#))  
*Clear parts of a dataspace.*
- long [allocate](#) ([l4\\_addr\\_t](#) offset, [l4\\_size\\_t](#) size)  
*Allocate a range in the dataspace.*
- long [copy\\_in](#) ([l4\\_addr\\_t](#) dst\_offs, [L4::lpc::Cap](#)< [Dataspace](#) > src, [l4\\_addr\\_t](#) src\_offs, unsigned long [size](#))  
*Copy contents from another dataspace.*
- long [phys](#) ([l4\\_addr\\_t](#) offset, [l4\\_addr\\_t](#) &phys\_addr, [l4\\_size\\_t](#) &phys\_size)  
*Get the physical addresses of a dataspace.*
- unsigned long [size](#) () const throw ()  
*Get size of a dataspace.*
- long [flags](#) () const throw ()  
*Get flags of the dataspace.*
- long [info](#) ([Stats](#) \*stats)  
*Get information on the dataspace.*

## Additional Inherited Members

### 14.222.1 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+\\_ds/ds\\_srv.cc](#).

Definition at line 59 of file [dataspace](#).

### 14.222.2 Member Enumeration Documentation

#### 14.222.2.1 Map\_flags

```
enum L4Re::Dataspace::Map_flags
```

Flags for map operations.

## Enumerator

|                   |                                              |
|-------------------|----------------------------------------------|
| Map_ro            | Request read-only mapping.                   |
| Map_rw            | Request writable mapping.                    |
| Map_normal        | request normal memory mapping                |
| Map_cacheable     | request normal memory mapping                |
| Map_bufferable    | request bufferable (write buffered) mappings |
| Map_uncacheable   | request uncacheable memory mappings          |
| Map_caching_mask  | mask for caching flags                       |
| Map_caching_shift | shift value for caching flags                |

Definition at line 68 of file [dataspace](#).

## 14.222.3 Member Function Documentation

## 14.222.3.1 allocate()

```
long L4Re::Dataspace::allocate (
 l4_addr_t offset,
 l4_size_t 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                                                                                                            |
| <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>&lt;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.

## 14.222.3.2 clear()

```
long L4Re::Dataspace::clear (
 l4_addr_t offset,
 unsigned long 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

|                          |                                                                                                                    |
|--------------------------|--------------------------------------------------------------------------------------------------------------------|
| $\geq 0$                 | 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> | <a href="#">Dataspace</a> is read-only.                                                                            |
| $< 0$                    | 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.

#### 14.222.3.3 `copy_in()`

```
long L4Re::Dataspace::copy_in (
 l4_addr_t dst_offs,
 L4::Ipc::Cap< Dataspace > src,
 l4_addr_t src_offs,
 unsigned long 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

|                          |                                     |
|--------------------------|-------------------------------------|
| <code>L4_EOK</code>      | Success                             |
| <code>-L4_EACCESS</code> | Destination dataspace not writable. |
| <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.

#### 14.222.3.4 `flags()`

```
long L4Re::Dataspace::flags () const throw)
```

Get flags of the dataspace.

## Return values

|          |                        |
|----------|------------------------|
| $\geq 0$ | Flags of the dataspace |
| $< 0$    | IPC errors             |

## See also

[L4Re::Dataspace::Map\\_flags](#)

Definition at line 116 of file [dataspace\\_impl.h](#).

## 14.222.3.5 info()

```
long L4Re::Dataspace::info (
 Stats * stats)
```

Get information on the dataspace.

## Parameters

|     |       |                                       |
|-----|-------|---------------------------------------|
| out | stats | <a href="#">Dataspace</a> information |
|-----|-------|---------------------------------------|

## Return values

|       |            |
|-------|------------|
| 0     | Success    |
| $< 0$ | IPC errors |

## 14.222.3.6 map()

```
long L4Re::Dataspace::map (
 l4_addr_t offset,
 unsigned long flags,
 l4_addr_t local_addr,
 l4_addr_t min_addr,
 l4_addr_t max_addr) const throw ()
```

Request a flex-page mapping from the dataspace.

## Parameters

|                   |                                                               |
|-------------------|---------------------------------------------------------------|
| <i>offset</i>     | Offset to start within dataspace                              |
| <i>flags</i>      | map flags, see <a href="#">Map_flags</a> .                    |
| <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.)     |

## Return values

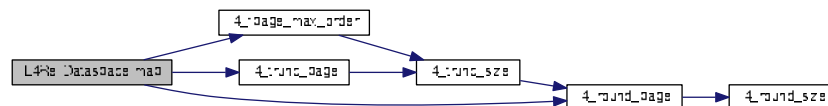
|                   |                                                       |
|-------------------|-------------------------------------------------------|
| <i>L4_EOK</i>     | Success                                               |
| <i>-L4_ERANGE</i> | Invalid offset.                                       |
| <i>-L4_EPERM</i>  | Insufficient permission to map with requested rights. |
| <i>&lt;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 92 of file [dataspace\\_impl.h](#).

References [l4\\_fpage\\_max\\_order\(\)](#), [L4\\_LOG2\\_PAGESIZE](#), [l4\\_round\\_page\(\)](#), and [l4\\_trunc\\_page\(\)](#).

Here is the call graph for this function:



## 14.222.3.7 map\_region()

```

long L4Re::Dataspace::map_region (
 l4_addr_t offset,
 unsigned long flags,
 l4_addr_t min_addr,
 l4_addr_t max_addr) const throw ()

```

Map a part of a dataspace completely.

## Parameters

|                 |                                                               |
|-----------------|---------------------------------------------------------------|
| <i>offset</i>   | Offset to start within dataspace                              |
| <i>flags</i>    | map flags, see <a href="#">Map_flags</a> .                    |
| <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.)     |

## Return values

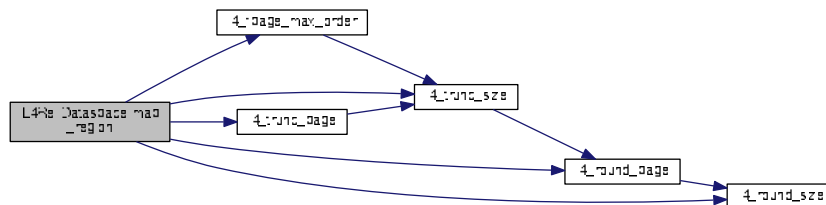
|                   |                                                       |
|-------------------|-------------------------------------------------------|
| <i>L4_EOK</i>     | Success                                               |
| <i>-L4_ERANGE</i> | Invalid offset.                                       |
| <i>-L4_EPERM</i>  | Insufficient permission to map with requested rights. |
| <i>&lt;0</i>      | IPC errors                                            |

Definition at line 55 of file [dataspace\\_impl.h](#).

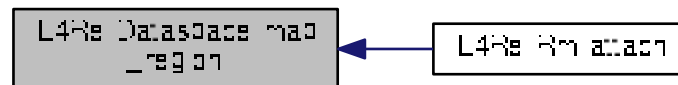
References [l4\\_fpage\\_max\\_order\(\)](#), [L4\\_LOG2\\_PAGESIZE](#), [l4\\_round\\_page\(\)](#), [l4\\_round\\_size\(\)](#), [l4\\_trunc\\_page\(\)](#), [l4\\_trunc\\_size\(\)](#), and [L4\\_UNLIKELY](#).

Referenced by [L4Re::Rm::attach\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 14.222.3.8 phys()

```

long L4Re::Dataspace::phys (
 l4_addr_t offset,
 l4_addr_t & phys_addr,
 l4_size_t & phys_size)

```

Get the physical addresses of a dataspace.

#### Parameters

|               |                     |
|---------------|---------------------|
| <i>offset</i> | Offset in dataspace |
|---------------|---------------------|

#### Return values

|                  |                                                                                                  |
|------------------|--------------------------------------------------------------------------------------------------|
| <i>phys_addr</i> | Physical address.                                                                                |
| <i>phys_size</i> | Size of largest physically contiguous region in the dataspace after <i>phys_addr</i> (in bytes). |



## Return values

|                   |                                          |
|-------------------|------------------------------------------|
| <i>L4_EOK</i>     | Success                                  |
| <i>-L4_EINVAL</i> | <a href="#">Dataspace</a> is not pinned. |
| <i>&lt;0</i>      | IPC errors                               |

This call will only succeed on pinned memory dataspaces.

**Deprecated** Use [L4Re::Dma\\_space](#) instead. This function will be removed in future releases.

## 14.222.3.9 size()

```
unsigned long L4Re::Dataspace::size () const throw ()
```

Get size of a dataspace.

## Returns

Size of the dataspace in bytes.

## Examples:

[examples/libs/l4re/c++/shared\\_ds/ds\\_clnt.cc](#).

Definition at line [106](#) of file [dataspace\\_impl.h](#).

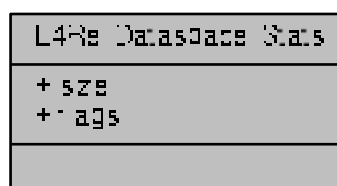
The documentation for this class was generated from the following files:

- [l4re/dataspace](#)
- [l4re/impl/dataspace\\_impl.h](#)

## 14.223 L4Re::Dataspace::Stats Struct Reference

Information about the dataspace.

Collaboration diagram for L4Re::Dataspace::Stats:



## Data Fields

- unsigned long [size](#)
- unsigned long [flags](#)

### 14.223.1 Detailed Description

Information about the dataspace.

Definition at line 85 of file [dataspace](#).

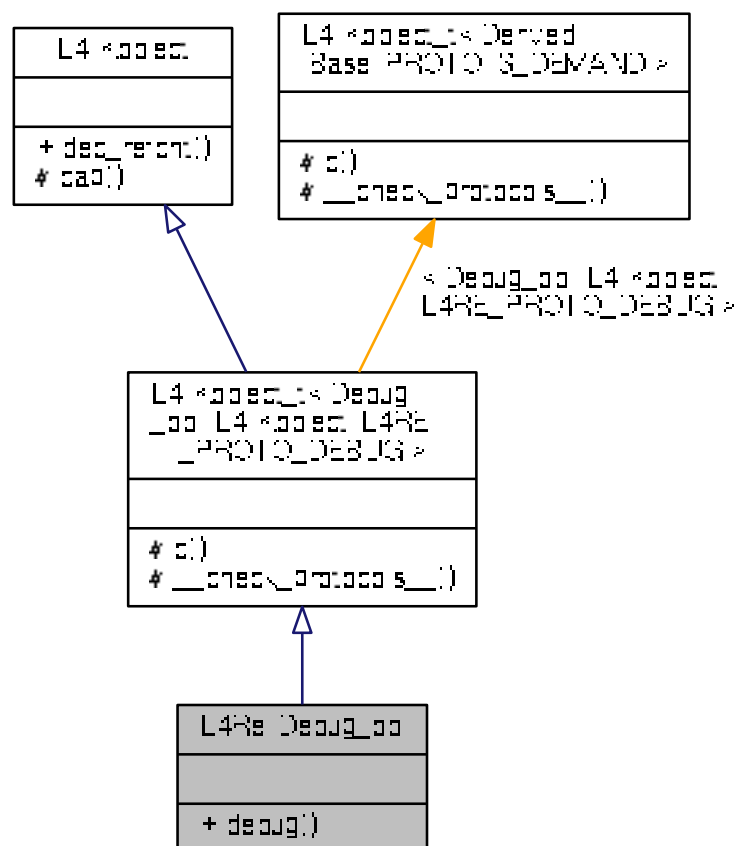
The documentation for this struct was generated from the following file:

- [l4/re/dataspace](#)

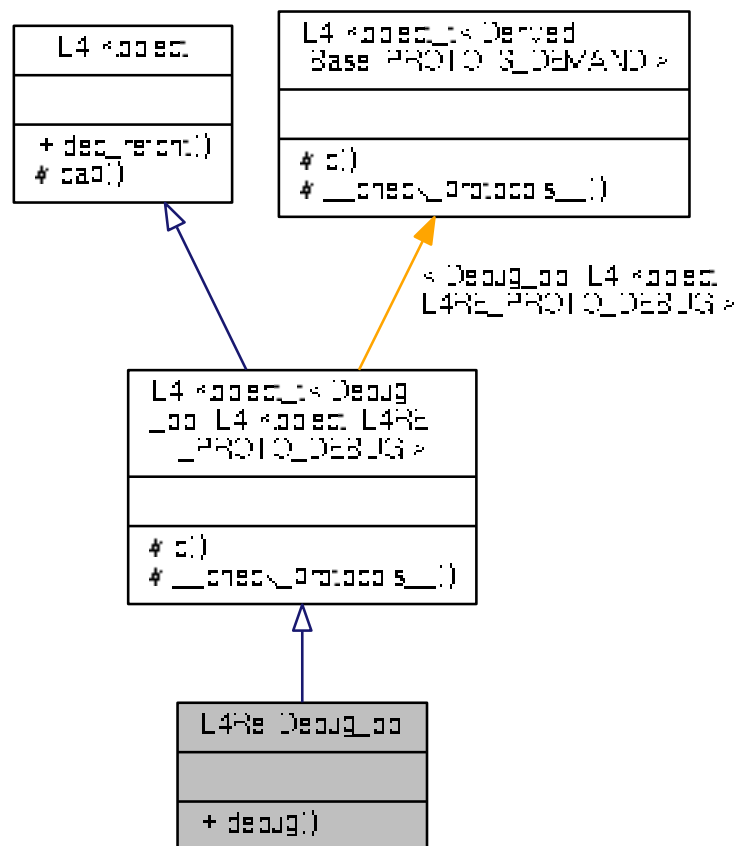
## 14.224 L4Re::Debug\_obj Class Reference

Debug interface.

Inheritance diagram for L4Re::Debug\_obj:



Collaboration diagram for L4Re::Debug\_obj:



## Public Member Functions

- long [debug](#) (unsigned long function)  
*Debug call.*

## Additional Inherited Members

### 14.224.1 Detailed Description

Debug interface.

See also

[Debugging API](#) .

Definition at line 51 of file [debug](#).

## 14.224.2 Member Function Documentation

### 14.224.2.1 debug()

```
long L4Re::Debug_obj::debug (
 unsigned long function)
```

Debug call.

#### Parameters

|                 |                   |
|-----------------|-------------------|
| <i>function</i> | Function to call. |
|-----------------|-------------------|

#### Returns

- L4\_EOK
- IPC errors

The documentation for this class was generated from the following file:

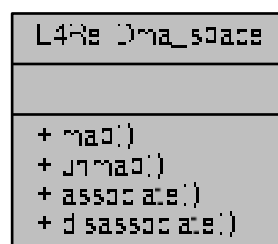
- [l4/re/debug](#)

## 14.225 L4Re::Dma\_space Class Reference

DMA Address Space.

Inherits L4::Kobject\_0t< Derived, PROTO, S\_DEMAND >.

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

- long [map](#) ([L4::lpc::Cap](#)< [L4Re::Dataspace](#) > src, [l4\\_addr\\_t](#) 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.*
- long [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.*
- long [associate](#) ([L4::lpc::Opt](#)< [L4::lpc::Cap](#)< [L4::Task](#) > > dma\_task, [Space\\_attribs](#) attr)  
*Associate a DMA task for a device to this [Dma\\_space](#).*
- long [disassociate](#) ()  
*Disassociate the DMA task from this [Dma\\_space](#).*

### 14.225.1 Detailed Description

DMA Address Space.

A [Dma\\_space](#) represents the DMA address space of a device. Whenever a device needs direct access to parts of an [L4Re::Dataspace](#), that part of the data space must be mapped to the DMA address space that is assigned to that device. After the DMA accesses to the memory are finished the memory must be unmapped from the device's DMA address space.

Mapping to a 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 61 of file [dma\\_space](#).

### 14.225.2 Member Typedef Documentation

### 14.225.2.1 Attributes

```
typedef L4::Types::Flags<Attribute> L4Re::Dma_space::Attributes
```

Attributes for DMA mappings.

See also

[Attribute](#)

Definition at line 106 of file [dma\\_space](#).

## 14.225.3 Member Enumeration Documentation

### 14.225.3.1 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 <a href="#">Dma_space</a> . 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 <a href="#">map()</a> or <a href="#">unmap()</a> operation of the same part of the data space (without the <a href="#">No_sync</a> attribute). |
|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Definition at line 85 of file [dma\\_space](#).

### 14.225.3.2 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 73 of file [dma\\_space](#).

### 14.225.3.3 Space\_attrb

enum [L4Re::Dma\\_space::Space\\_attrb](#)

Attributes assigned to the DMA space when associated with a specific device.

See also

[Space\\_attrbs](#)

#### Enumerator

|            |                                                                                                                                                                               |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Coherent   | The device is connected coherently with the cache. This means that the <a href="#">map()</a> and <a href="#">unmap()</a> 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 113 of file [dma\\_space](#).

## 14.225.4 Member Function Documentation

### 14.225.4.1 associate()

```
long L4Re::Dma_space::associate (
 L4::Ipc::Opt< L4::Ipc::Cap< L4::Task > > dma_task,
 Space_attrbs attr)
```

Associate a DMA task for a device to this [Dma\\_space](#).

#### Precondition

requires capability rights: {RW}

#### Parameters

|    |                 |                                                                                                                                                                                                                                                               |
|----|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| in | <i>dma_task</i> | The DMA task used for the device that shall be associated with this DMA space. The <i>dma_task</i> might be an invalid capability when <a href="#">Phys_space</a> 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 <a href="#">Space_attrb</a> .                                                                                                                                                                                              |

#### 14.225.4.2 disassociate()

```
long L4Re::Dma_space::disassociate ()
```

Disassociate the DMA task from this [Dma\\_space](#).

##### Precondition

requires capability rights: {RW}

#### 14.225.4.3 map()

```
long L4Re::Dma_space::map (
 L4::Ipc::Cap< L4Re::Dataspace > src,
 l4_addr_t 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.

##### Precondition

requires capability rights: {R}

##### Parameters

|         |                 |                                                                                                                                                                                                                                                                                                                         |
|---------|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| in      | <i>src</i>      | Source data space (that describes the memory). Caller needs write rights 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 of the region to be mapped to for DMA, after successful mapping the size returned is the size mapped for DMA as single block, this size might be smaller than the original input size, in this case the caller might call <a href="#">map()</a> again with a new offset and the remaining size. |
| in      | <i>attrs</i>    | The attributes used for this DMA mapping (a combination of <a href="#">Dma_space::Attribute</a> values).                                                                                                                                                                                                                |
| in      | <i>dir</i>      | The direction of the DMA transfer issued with this mapping. The same value must later be passed to <a href="#">unmap()</a> .                                                                                                                                                                                            |
| out     | <i>dma_addr</i> | The DMA address to use for DMA with the associated device.                                                                                                                                                                                                                                                              |

##### Returns

0 in the case of success, a negative error code otherwise.



```
long L4Re::Dma_space::unmap (
 Dma_addr dma_addr,
 l4_size_t size,
 Attributes attrs,
 Direction dir)
```

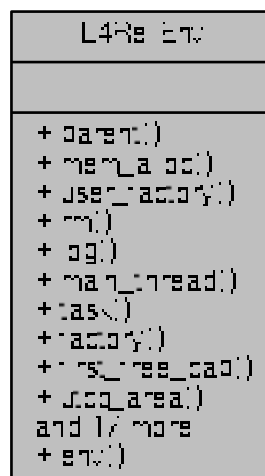
### Precondition

## Parameters

|                 |                                                               |
|-----------------|---------------------------------------------------------------|
| <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.                  |

- l4/re/dma\_space

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 throw ()  
*Object-capability to the parent.*
- [L4::Cap](#)< [Mem\\_alloc](#) > [mem\\_alloc](#) () const throw ()  
*Object-capability to the memory allocator.*
- [L4::Cap](#)< [L4::Factory](#) > [user\\_factory](#) () const throw ()  
*Object-capability to the user-level object factory.*
- [L4::Cap](#)< [Rm](#) > [rm](#) () const throw ()  
*Object-capability to the region map.*
- [L4::Cap](#)< [Log](#) > [log](#) () const throw ()  
*Object-capability to the logging service.*
- [L4::Cap](#)< [L4::Thread](#) > [main\\_thread](#) () const throw ()  
*Object-capability of the first user thread.*
- [L4::Cap](#)< [L4::Task](#) > [task](#) () const throw ()  
*Object-capability of the user task.*
- [L4::Cap](#)< [L4::Factory](#) > [factory](#) () const throw ()  
*Object-capability to the factory object available to the task.*
- [l4\\_cap\\_idx\\_t](#) [first\\_free\\_cap](#) () const throw ()  
*First available capability selector.*
- [l4\\_fpage\\_t](#) [utcb\\_area](#) () const throw ()  
*UTCB area of the task.*
- [l4\\_addr\\_t](#) [first\\_free\\_utcb](#) () const throw ()  
*First free UTCB.*
- [Cap\\_entry](#) const \* [initial\\_caps](#) () const throw ()  
*Get a pointer to the first entry in the initial objects array.*
- [Cap\\_entry](#) const \* [get](#) (char const \*name, unsigned l) const throw ()  
*Get the Cap\_entry for the object named name.*
- template<typename T >  
[L4::Cap](#)< T > [get\\_cap](#) (char const \*name, unsigned l) const throw ()  
*Get the capability selector for the object named name.*
- template<typename T >  
[L4::Cap](#)< T > [get\\_cap](#) (char const \*name) const throw ()  
*Get the capability selector for the object named name.*
- void [parent](#) ([L4::Cap](#)< [Parent](#) > const &c) throw ()  
*Set parent object-capability.*
- void [mem\\_alloc](#) ([L4::Cap](#)< [Mem\\_alloc](#) > const &c) throw ()  
*Set memory allocator object-capability.*
- void [rm](#) ([L4::Cap](#)< [Rm](#) > const &c) throw ()  
*Set region map object-capability.*
- void [log](#) ([L4::Cap](#)< [Log](#) > const &c) throw ()  
*Set log object-capability.*
- void [main\\_thread](#) ([L4::Cap](#)< [L4::Thread](#) > const &c) throw ()  
*Set object-capability of first user thread.*
- void [factory](#) ([L4::Cap](#)< [L4::Factory](#) > const &c) throw ()

- Set factory object-capability.*
- void [first\\_free\\_cap](#) ([l4\\_cap\\_idx\\_t](#) c) throw ()
- Set first available capability selector.*
- void [utcb\\_area](#) ([l4\\_fpage\\_t](#) utcb) throw ()
- Set UTCB area of the task.*
- void [first\\_free\\_utcb](#) ([l4\\_addr\\_t](#) u) throw ()
- Set first free UTCB.*
- [L4::Cap](#)< [L4::Scheduler](#) > [scheduler](#) () const throw ()
- Get the scheduler capability for the task.*
- void [scheduler](#) ([L4::Cap](#)< [L4::Scheduler](#) > const &c) throw ()
- Set the scheduler capability.*
- void [initial\\_caps](#) ([Cap\\_entry](#) \*first) throw ()
- Set the pointer to the first Cap\_entry in the initial objects array.*

### Static Public Member Functions

- static [Env](#) const \* [env](#) () throw ()
- Returns the initial environment for the current task.*

#### 14.226.1 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       | <a href="#">L4Re::Parent</a>  | <a href="#">L4Re::Env::parent()</a>       |
| user_factory | <a href="#">L4::Factory</a>   | <a href="#">L4Re::Env::user_factory()</a> |
| log          | <a href="#">L4Re::Log</a>     | <a href="#">L4Re::Env::log()</a>          |
| main_thread  | <a href="#">L4::Thread</a>    | <a href="#">L4Re::Env::main_thread()</a>  |
| rm           | <a href="#">L4Re::Rm</a>      | <a href="#">L4Re::Env::rm()</a>           |
| factory      | <a href="#">L4::Factory</a>   | <a href="#">L4Re::Env::factory()</a>      |
| task         | <a href="#">L4::Task</a>      | <a href="#">L4Re::Env::task()</a>         |
| scheduler    | <a href="#">L4::Scheduler</a> | <a href="#">L4Re::Env::scheduler()</a>    |

Additional information found in the initial environment is:

- First free entry in capability table
- The [UTCB](#) area (as flex page)
- First free UTCB (address in the UTCB area)

## Include File

```
#include <l4/re/env>
```

For the C interface refer to [Initial Environment](#).

Definition at line 82 of file [env](#).

## 14.226.2 Member Function Documentation

### 14.226.2.1 env()

```
static Env const* L4Re::Env::env () throw () [inline], [static]
```

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>()

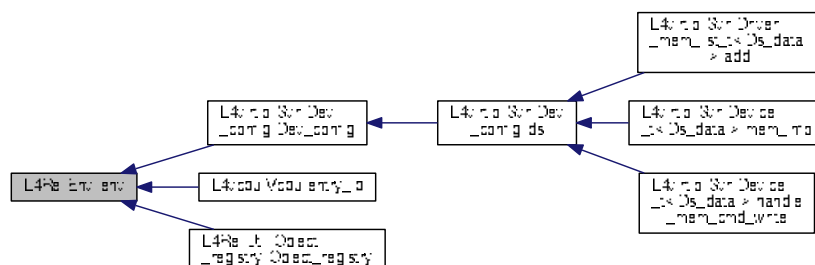
#### Examples:

[examples/clntsrv/client.cc](#), [examples/libs/l4re/c++/mem\\_alloc/marm.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 100 of file [env](#).

Referenced by [L4virtio::Svr::Dev\\_config::Dev\\_config\(\)](#), [L4vcpu::Vcpu::entry\\_ip\(\)](#), and [L4Re::Util::Object\\_registry::Object\\_registry\(\)](#).

Here is the caller graph for this function:



**14.226.2.2** `factory()` [1/2]

```
L4::Cap<L4::Factory> L4Re::Env::factory () const throw () [inline]
```

Object-capability to the factory object available to the task.

**Returns**

Factory object-capability

Definition at line 148 of file [env](#).

References [l4re\\_env\\_t::factory](#).

**14.226.2.3** `factory()` [2/2]

```
void L4Re::Env::factory (
 L4::Cap< L4::Factory > const & c) throw () [inline]
```

Set factory object-capability.

**Parameters**

|                |                           |
|----------------|---------------------------|
| <code>c</code> | Factory object-capability |
|----------------|---------------------------|

Definition at line 253 of file [env](#).

References [l4re\\_env\\_t::factory](#).

**14.226.2.4** `first_free_cap()` [1/2]

```
l4_cap_idx_t L4Re::Env::first_free_cap () const throw () [inline]
```

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](#).

References [l4re\\_env\\_t::first\\_free\\_cap](#).

**14.226.2.5** `first_free_cap()` [2/2]

```
void L4Re::Env::first_free_cap (
 l4_cap_idx_t c) throw () [inline]
```

Set first available capability selector.

**Parameters**

|                |                                                         |
|----------------|---------------------------------------------------------|
| <code>c</code> | First capability selector available to the application. |
|----------------|---------------------------------------------------------|

Definition at line 259 of file [env](#).

References [l4re\\_env\\_t::first\\_free\\_cap](#).

**14.226.2.6 first\_free\_utcb()** [1/2]

```
l4_addr_t L4Re::Env::first_free_utcb () const throw () [inline]
```

First free UTCB.

**Returns**

object-capability

First free UTCB within the UTCB area available for the application to use.

Definition at line 171 of file [env](#).

References [l4re\\_env\\_t::first\\_free\\_utcb](#).

**14.226.2.7 first\_free\_utcb()** [2/2]

```
void L4Re::Env::first_free_utcb (
 l4_addr_t u) throw () [inline]
```

Set first free UTCB.

**Parameters**

|                |                                                  |
|----------------|--------------------------------------------------|
| <code>u</code> | First UTCB available for the application to use. |
|----------------|--------------------------------------------------|

Definition at line 271 of file [env](#).

References [l4re\\_env\\_t::first\\_free\\_utcb](#).

**14.226.2.8 get()**

```
Cap_entry const* L4Re::Env::get (
 char const * name,
 unsigned l) const throw () [inline]
```

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 189 of file `env`.

References `l4re_env_get_cap_l()`.

Here is the call graph for this function:

14.226.2.9 `get_cap()` [1/2]

```

template<typename T >
L4::Cap<T> L4Re::Env::get_cap (
 char const * name,
 unsigned l) const throw () [inline]

```

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:

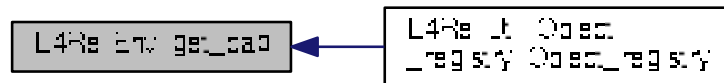
[examples/clntsrv/client.cc](#), [examples/libs/l4re/c++/shared\\_ds/ds\\_clnt.cc](#), and [examples/libs/l4re/streammap/client.cc](#).↔

Definition at line 201 of file [env](#).

References [L4\\_ENOENT](#).

Referenced by [L4Re::Util::Object\\_registry::Object\\_registry\(\)](#).

Here is the caller graph for this function:



#### 14.226.2.10 `get_cap()` [2/2]

```
template<typename T >
L4::Cap<T> L4Re::Env::get_cap (
 char const * name) const throw () [inline]
```

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 216 of file [env](#).

#### 14.226.2.11 `initial_caps()` [1/2]

```
Cap_entry const* L4Re::Env::initial_caps () const throw () [inline]
```

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 178 of file [env](#).



14.226.2.12 `initial_caps()` [2/2]

```
void L4Re::Env::initial_caps (
 Cap_entry * first) throw () [inline]
```

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 293 of file `env`.

14.226.2.13 `log()` [1/2]

```
L4::Cap<Log> L4Re::Env::log () const throw () [inline]
```

Object-capability to the logging service.

## Returns

`Log` object-capability

Definition at line 130 of file `env`.

References `l4re_env_t::log`.

14.226.2.14 `log()` [2/2]

```
void L4Re::Env::log (
 L4::Cap< Log > const & c) throw () [inline]
```

Set log object-capability.

## Parameters

|          |                                    |
|----------|------------------------------------|
| <i>c</i> | <code>Log</code> object-capability |
|----------|------------------------------------|

Definition at line 241 of file `env`.

References `l4re_env_t::log`.

**14.226.2.15** `main_thread()` [1/2]

```
L4::Cap<L4::Thread> L4Re::Env::main_thread () const throw () [inline]
```

Object-capability of the first user thread.

**Returns**

Object-capability of the first user thread.

Definition at line 136 of file [env](#).

References [l4re\\_env\\_t::main\\_thread](#).

**14.226.2.16** `main_thread()` [2/2]

```
void L4Re::Env::main_thread (
 L4::Cap< L4::Thread > const & c) throw () [inline]
```

Set object-capability of first user thread.

**Parameters**

|                |                                  |
|----------------|----------------------------------|
| <code>c</code> | First thread's object-capability |
|----------------|----------------------------------|

Definition at line 247 of file [env](#).

References [l4re\\_env\\_t::main\\_thread](#).

**14.226.2.17** `mem_alloc()` [1/2]

```
L4::Cap<Mem_alloc> L4Re::Env::mem_alloc () const throw () [inline]
```

Object-capability to the memory allocator.

**Returns**

Memory allocator object-capability

**Examples:**

[examples/libs/l4re/c++/shared\\_ds/ds\\_srv.cc](#).

Definition at line 113 of file [env](#).

References [l4re\\_env\\_t::mem\\_alloc](#).

**14.226.2.18** `mem_alloc()` [2/2]

```
void L4Re::Env::mem_alloc (
 L4::Cap< Mem_alloc > const & c) throw () [inline]
```

Set memory allocator object-capability.

## Parameters

|                |                                    |
|----------------|------------------------------------|
| <code>c</code> | Memory allocator object-capability |
|----------------|------------------------------------|

Definition at line 229 of file [env](#).

References [l4re\\_env\\_t::mem\\_alloc](#).

14.226.2.19 `parent()` [1/2]

```
L4::Cap<Parent> L4Re::Env::parent () const throw () [inline]
```

Object-capability to the parent.

## Returns

[Parent](#) object-capability

Definition at line 107 of file [env](#).

References [l4re\\_env\\_t::parent](#).

14.226.2.20 `parent()` [2/2]

```
void L4Re::Env::parent (
 L4::Cap< Parent > const & c) throw () [inline]
```

Set parent object-capability.

## Parameters

|                |                                          |
|----------------|------------------------------------------|
| <code>c</code> | <a href="#">Parent</a> object-capability |
|----------------|------------------------------------------|

Definition at line 223 of file [env](#).

References [l4re\\_env\\_t::parent](#).

14.226.2.21 `rm()` [1/2]

```
L4::Cap<Rm> L4Re::Env::rm () const throw () [inline]
```

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 124 of file [env](#).

References [l4re\\_env\\_t::rm](#).

**14.226.2.22 rm()** [2/2]

```
void L4Re::Env::rm (
 L4::Cap< Rm > const & c) throw () [inline]
```

Set region map object-capability.

**Parameters**

|          |                              |
|----------|------------------------------|
| <b>c</b> | Region map object-capability |
|----------|------------------------------|

Definition at line 235 of file [env](#).

References [l4re\\_env\\_t::rm](#).

**14.226.2.23 scheduler()** [1/2]

```
L4::Cap<L4::Scheduler> L4Re::Env::scheduler () const throw () [inline]
```

Get the scheduler capability for the task.

**Returns**

The capability selector for the default scheduler used for this task.

**Examples:**

[examples/sys/migrate/thread\\_migrate.cc](#).

Definition at line 279 of file [env](#).

References [l4re\\_env\\_t::scheduler](#).

**14.226.2.24 scheduler()** [2/2]

```
void L4Re::Env::scheduler (
 L4::Cap< L4::Scheduler > const & c) throw () [inline]
```

Set the scheduler capability.

## Parameters

|                |                                           |
|----------------|-------------------------------------------|
| <code>c</code> | is the capability to be set as scheduler. |
|----------------|-------------------------------------------|

Definition at line 286 of file [env](#).

References [l4re\\_env\\_t::scheduler](#).

## 14.226.2.25 task()

```
L4::Cap<L4::Task> L4Re::Env::task () const throw () [inline]
```

Object-capability of the user task.

## Returns

Object-capability of the user task.

Definition at line 142 of file [env](#).

## 14.226.2.26 utcb\_area() [1/2]

```
l4_fpage_t L4Re::Env::utcb_area () const throw () [inline]
```

UTCB area of the task.

## Returns

UTCB area

Definition at line 162 of file [env](#).

References [l4re\\_env\\_t::utcb\\_area](#).

## 14.226.2.27 utcb\_area() [2/2]

```
void L4Re::Env::utcb_area (
 l4_fpage_t utcbs) throw () [inline]
```

Set UTCB area of the task.

## Parameters

|               |           |
|---------------|-----------|
| <i>utcb</i> s | UTCB area |
|---------------|-----------|

Definition at line 265 of file [env](#).

References [l4re\\_env\\_t::utcb\\_area](#).

The documentation for this class was generated from the following file:

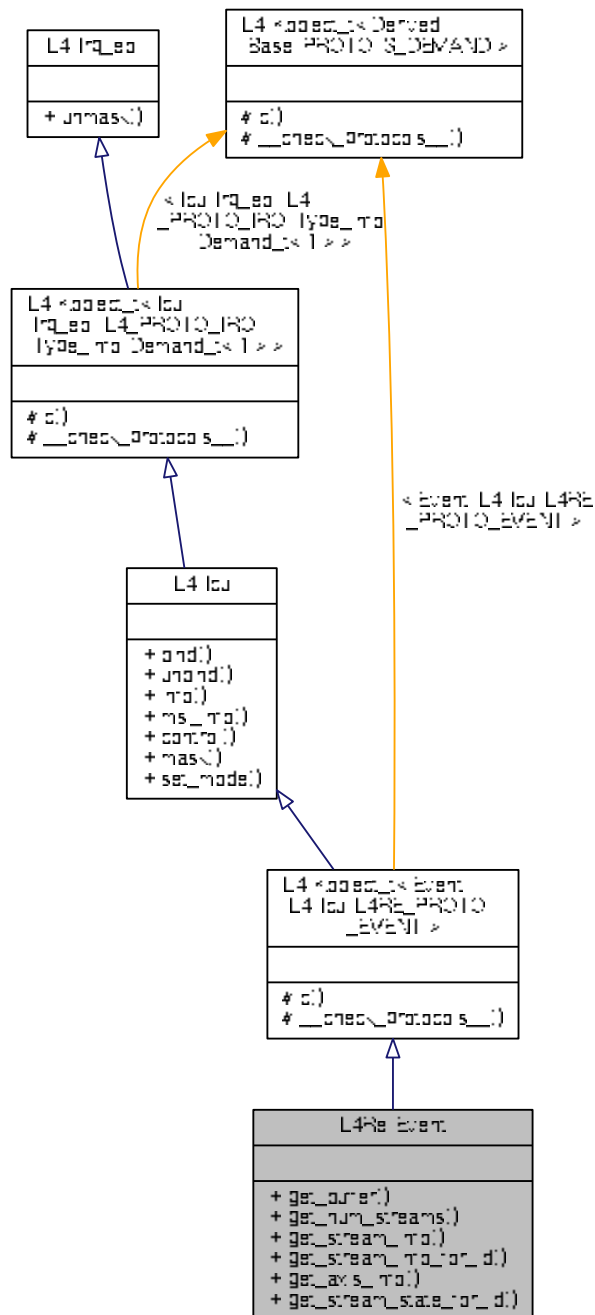
- [l4/re/env](#)

## 14.227 L4Re::Event Class Reference

[Event](#) class.







- Get event signal buffer.

---

## Additional Inherited Members

### 14.227.1 Detailed Description

[Event](#) class.

Definition at line 133 of file [event](#).

### 14.227.2 Member Function Documentation

#### 14.227.2.1 `get_buffer()`

```
long L4Re::Event::get_buffer (
 L4::Ipc::Out< L4::Cap< Dataspace > > ds)
```

Get event signal buffer.

#### Return values

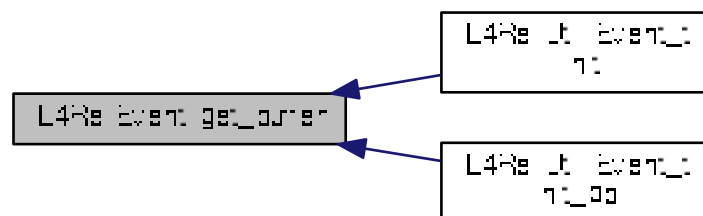
|           |                               |
|-----------|-------------------------------|
| <i>ds</i> | <a href="#">Event</a> buffer. |
|-----------|-------------------------------|

#### Returns

0 on success, negative error code otherwise.

Referenced by [L4Re::Util::Event\\_t< PAYLOAD >::init\(\)](#), and [L4Re::Util::Event\\_t< PAYLOAD >::init\\_poll\(\)](#).

Here is the caller graph for this function:



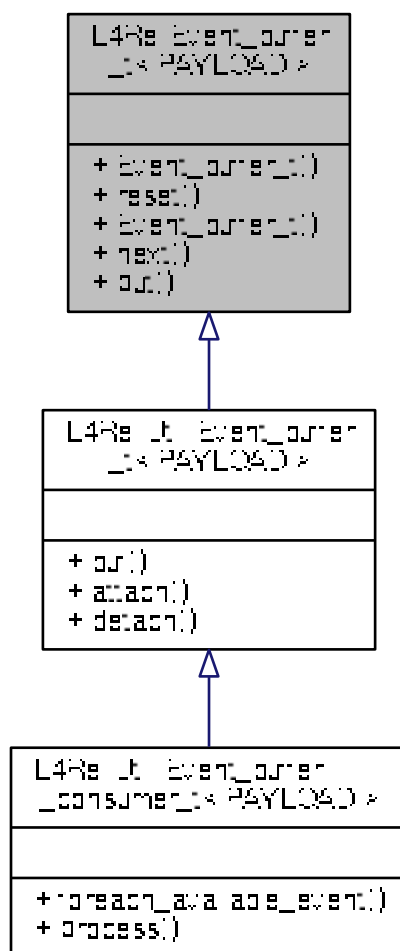
The documentation for this class was generated from the following file:

- `l4/re/event`

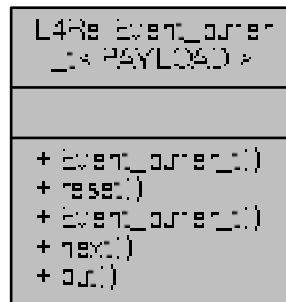
## 14.228 L4Re::Event\_buffer\_t< PAYLOAD > Class Template Reference

Event buffer class.

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](#) () throw ()  
*Next event in buffer.*
- bool [put](#) ([Event](#) const &ev) throw ()  
*Put event into buffer at current position.*

### 14.228.1 Detailed Description

```
template<typename PAYLOAD = Default_event_payload>
class L4Re::Event_buffer_t< PAYLOAD >
```

[Event](#) buffer class.

Definition at line [187](#) of file [event](#).

### 14.228.2 Constructor & Destructor Documentation

#### 14.228.2.1 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 234 of file [event](#).

### 14.228.3 Member Function Documentation

#### 14.228.3.1 next()

```
template<typename PAYLOAD = Default_event_payload>
Event* L4Re::Event_buffer_t< PAYLOAD >::next () throw () [inline]
```

Next event in buffer.

## Returns

0 if no event available, event otherwise.

Definition at line 244 of file [event](#).

References [L4Re::Event\\_buffer\\_t< PAYLOAD >::Event::time](#).

#### 14.228.3.2 put()

```
template<typename PAYLOAD = Default_event_payload>
bool L4Re::Event_buffer_t< PAYLOAD >::put (
 Event const & ev) throw () [inline]
```

Put event into buffer at current position.

## Parameters

|           |                                               |
|-----------|-----------------------------------------------|
| <i>ev</i> | <a href="#">Event</a> to put into the buffer. |
|-----------|-----------------------------------------------|

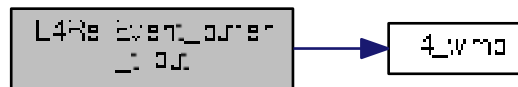
## Returns

false if buffer is full and entry could not be added.

Definition at line 261 of file [event](#).

References [l4\\_wmb\(\)](#), and [L4Re::Event\\_buffer\\_t< PAYLOAD >::Event::time](#).

Here is the call graph for this function:



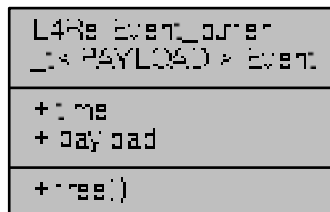
The documentation for this class was generated from the following file:

- `l4/re/event`

## 14.229 L4Re::Event\_buffer\_t< PAYLOAD >::Event Struct Reference

[Event](#) structure used in buffer.

Collaboration diagram for L4Re::Event\_buffer\_t< PAYLOAD >::Event:



### Public Member Functions

- `void free () throw ()`  
Free the entry.

### Data Fields

- `long long time`  
*Event time stamp.*

### 14.229.1 Detailed Description

```
template<typename PAYLOAD = Default_event_payload>
struct L4Re::Event_buffer_t< PAYLOAD >::Event
```

[Event](#) structure used in buffer.

Definition at line [194](#) of file [event](#).

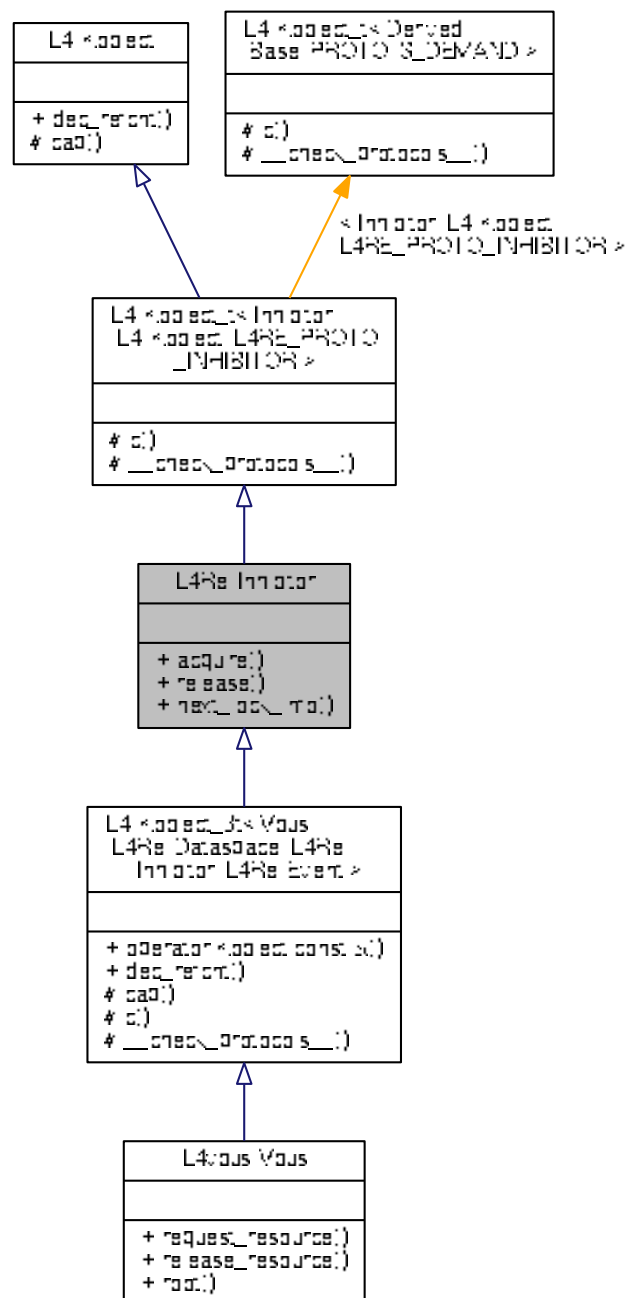
The documentation for this struct was generated from the following file:

- [l4/re/event](#)

## 14.230 L4Re::Inhibitor Class Reference

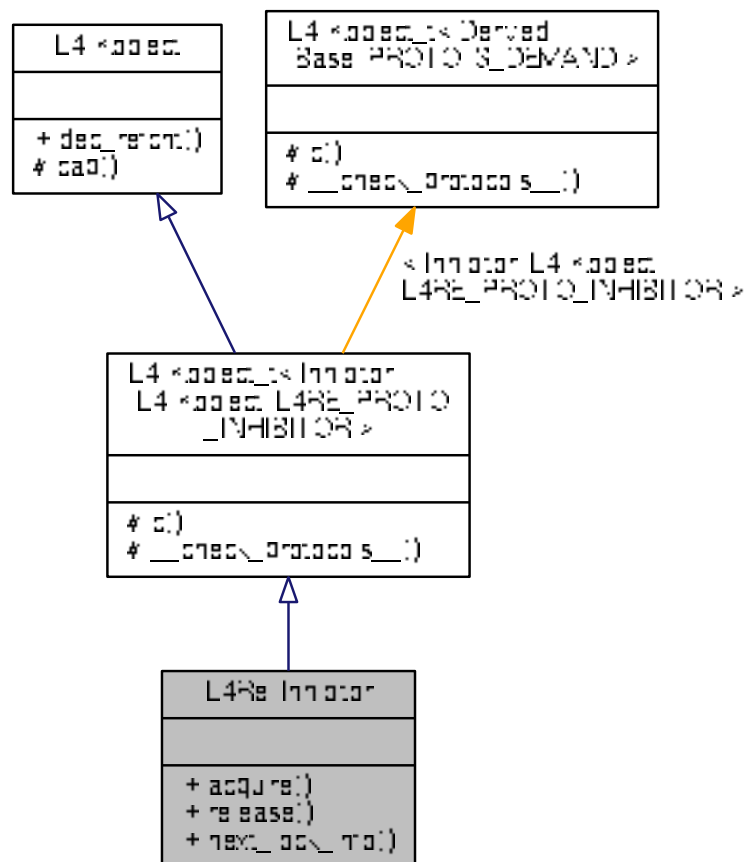
Set of inhibitor locks, which inhibit specific actions when held.

Inheritance diagram for L4Re::Inhibitor:





Collaboration diagram for L4Re::Inhibitor:



## Public Types

- enum { `Name_max` = 20 }

## Public Member Functions

- long `acquire` (`l4_umword_t` id, `L4::lpc::String<>` reason)  
*Acquire a specific inhibitor lock.*
- long `release` (`l4_umword_t` id)  
*Release a specific inhibitor lock.*
- long `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.*

## Additional Inherited Members

### 14.230.1 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 [40](#) of file [inhibitor](#).

### 14.230.2 Member Enumeration Documentation

#### 14.230.2.1 anonymous enum

anonymous enum

##### Enumerator

|          |                                      |
|----------|--------------------------------------|
| Name_max | The maximum length of a lock's name. |
|----------|--------------------------------------|

Definition at line [44](#) of file [inhibitor](#).

### 14.230.3 Member Function Documentation

#### 14.230.3.1 acquire()

```
long 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. |

## 14.230.3.2 next\_lock\_info()

```
long 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 <a href="#">Name_max</a> 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.                                              |

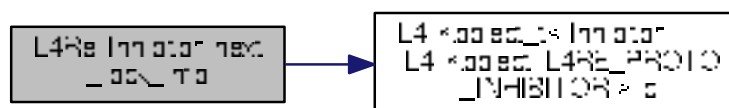
## Return values

|            |                                                                                                                            |
|------------|----------------------------------------------------------------------------------------------------------------------------|
| >0         | 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). |
| -L4_ENODEV | There are no more locks.                                                                                                   |

Definition at line 86 of file [inhibitor](#).

References [L4::Kobject\\_t< Inhibitor](#), [L4::Kobject](#), [L4RE\\_PROTO\\_INHIBITOR >::c\(\)](#), and [L4\\_INLINE\\_RPC\\_NF](#).

Here is the call graph for this function:



### 14.230.3.3 release()

```
long 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. |

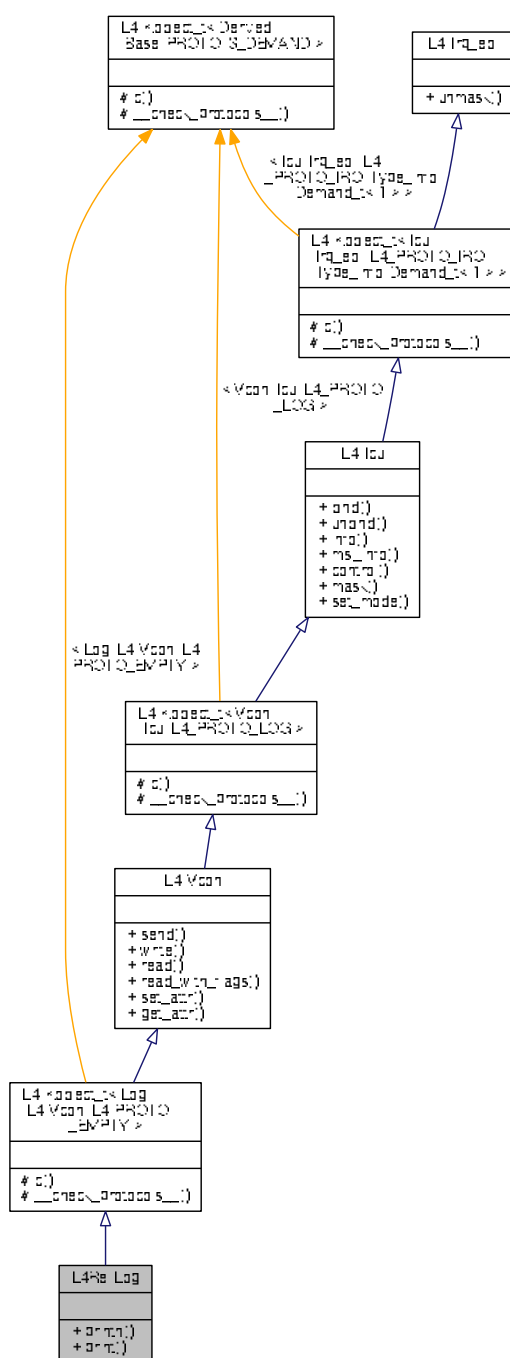
The documentation for this class was generated from the following file:

- l4/re/inhibitor

## 14.231 L4Re::Log Class Reference

[Log](#) interface class.

Inheritance diagram for L4Re::Log:





## Additional Inherited Members

### 14.231.1 Detailed Description

[Log](#) interface class.

Definition at line 44 of file [log](#).

### 14.231.2 Member Function Documentation

#### 14.231.2.1 print()

```
void L4Re::Log::print (
 char const * string) const throw)
```

Print NULL-terminated string.

##### Parameters

|               |                 |
|---------------|-----------------|
| <i>string</i> | string to print |
|---------------|-----------------|

#### 14.231.2.2 printn()

```
void L4Re::Log::printn (
 char const * string,
 int len) const throw)
```

Print string with length len, NULL characters don't matter.

##### Parameters

|               |                  |
|---------------|------------------|
| <i>string</i> | string to print  |
| <i>len</i>    | length of string |

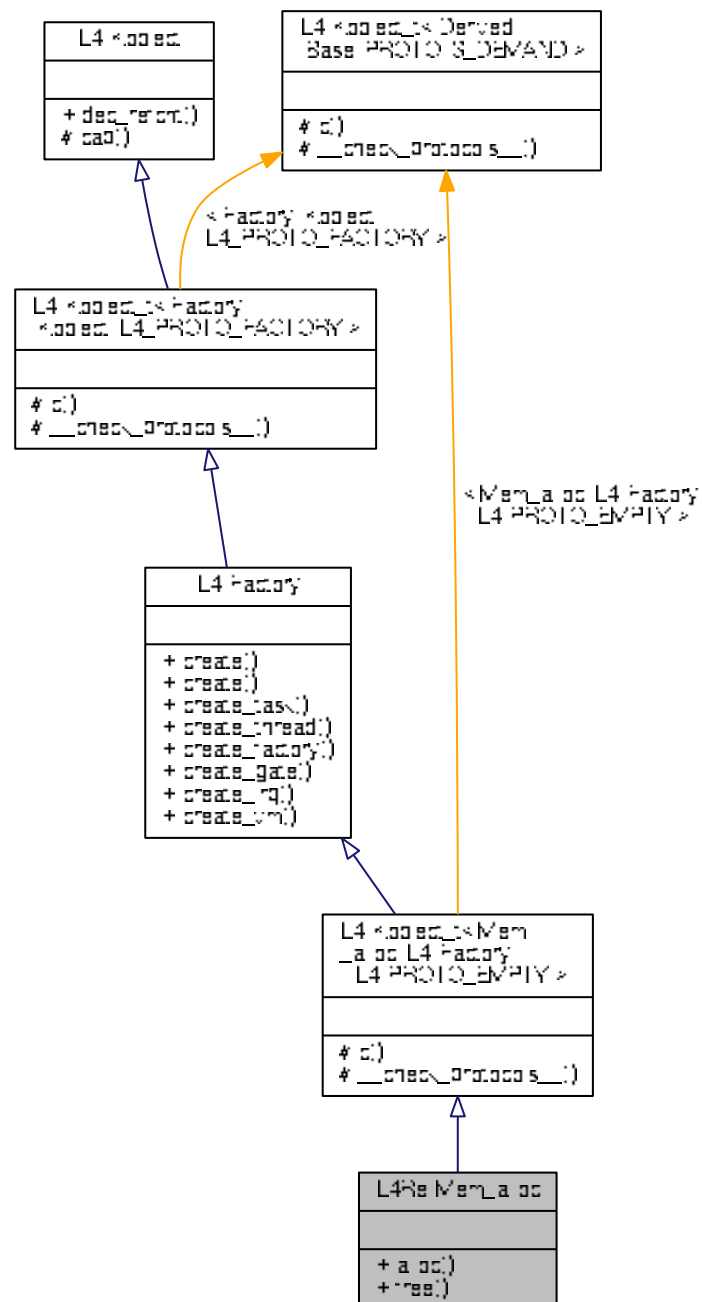
The documentation for this class was generated from the following file:

- [l4/re/log](#)

## 14.232 L4Re::Mem\_alloc Class Reference

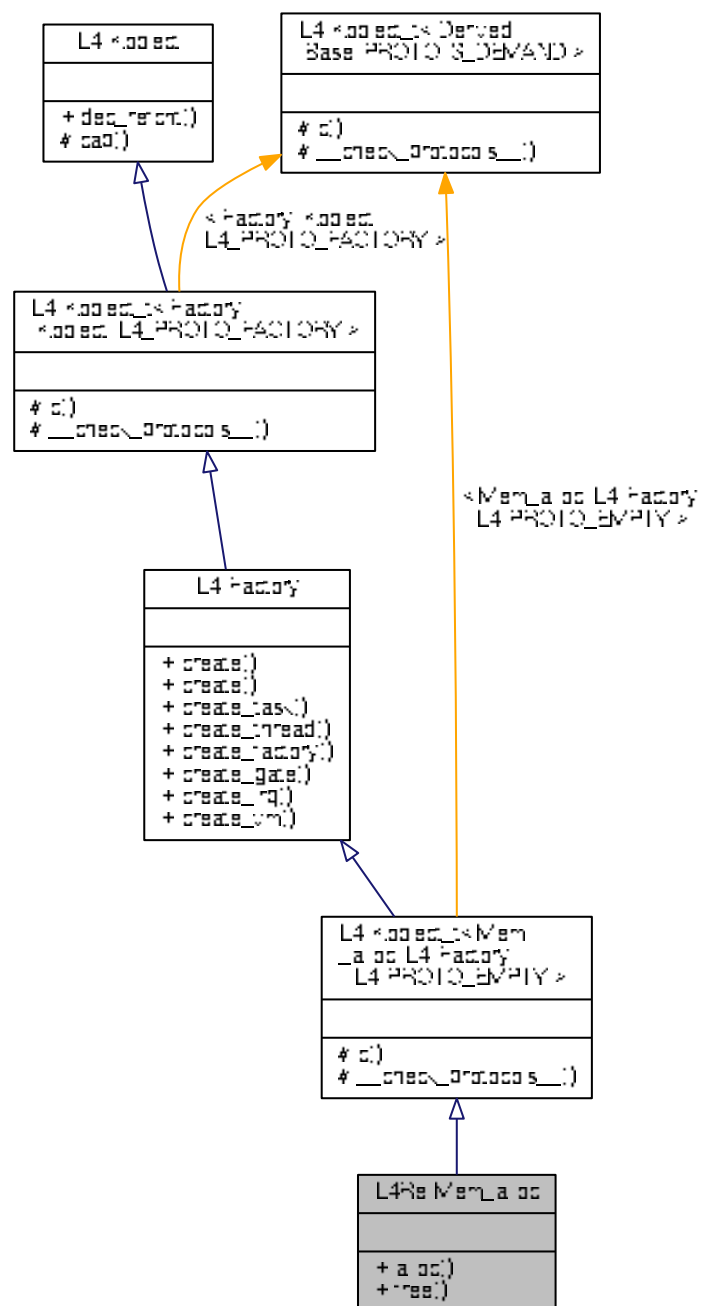
Memory allocation interface.

Inheritance diagram for L4Re::Mem\_alloc:





Collaboration diagram for L4Re::Mem\_alloc:



## Public Types

- enum `Mem_alloc_flags` { `Continuous` = 0x01, `Pinned` = 0x02, `Super_pages` = 0x04 }

*Flags for the allocator.*

## Public Member Functions

- long `alloc` (long size, `L4::Cap< Dataspace >` mem, unsigned long flags=0, unsigned long align=0) const throw ()  
*Allocate anonymous memory.*
- long `free` (`L4::Cap< Dataspace >` mem) const) throw ()  
*Free dataspace.*

## Additional Inherited Members

### 14.232.1 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 61 of file `mem_alloc`.

### 14.232.2 Member Enumeration Documentation

#### 14.232.2.1 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 <code>L4Re::Dma_space</code> instead. |
| Super_pages | Allocate super pages.                                 |

Definition at line 71 of file `mem_alloc`.

### 14.232.3 Member Function Documentation

#### 14.232.3.1 alloc()

```
long L4Re::Mem_alloc::alloc (
 long size,
 L4::Cap< Dataspace > mem,
 unsigned long flags = 0,
 unsigned long align = 0) const throw ()
```

Allocate anonymous memory.

##### Parameters

|     |              |                                                                                                                                                                                                                                                                                                                                                                     |
|-----|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|     | <i>size</i>  | Size in bytes to be requested (granularity is (super)pages and the size is rounded up to this granularity). 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. |
| out | <i>mem</i>   | Capability slot where the capability to the dataspace is received.                                                                                                                                                                                                                                                                                                  |
|     | <i>flags</i> | Special dataspace properties, see <a href="#">Mem_alloc_flags</a>                                                                                                                                                                                                                                                                                                   |
|     | <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>                                                                                                                                                                                  |

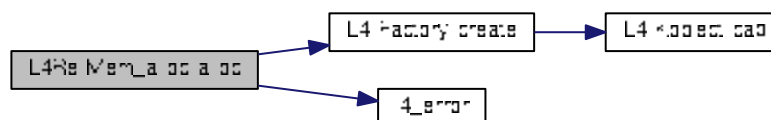
##### Return values

|            |                              |
|------------|------------------------------|
| 0          | Success                      |
| -L4_ERANGE | Given size not supported.    |
| -L4_ENOMEM | Not enough memory available. |
| <0         | IPC error                    |

Definition at line 35 of file [mem\\_alloc\\_impl.h](#).

References [L4::Factory::create\(\)](#), and [l4\\_error\(\)](#).

Here is the call graph for this function:



## 14.232.3.2 free()

```
long L4Re::Mem_alloc::free (
 L4::Cap< Dataspace > mem) const throw ()
```

Free dataspace.

## Parameters

|            |                           |
|------------|---------------------------|
| <i>mem</i> | Dataspace to be released. |
|------------|---------------------------|

## Return values

|   |  |
|---|--|
| 0 |  |
|---|--|

**Deprecated** This function is an empty stub which remains here for backward compatibility only. Use `L4::Task↵::unmap(mem, L4_FP_DELETE_OBJ)` or other similar means to remove a dataspace.

Definition at line 47 of file [mem\\_alloc\\_impl.h](#).

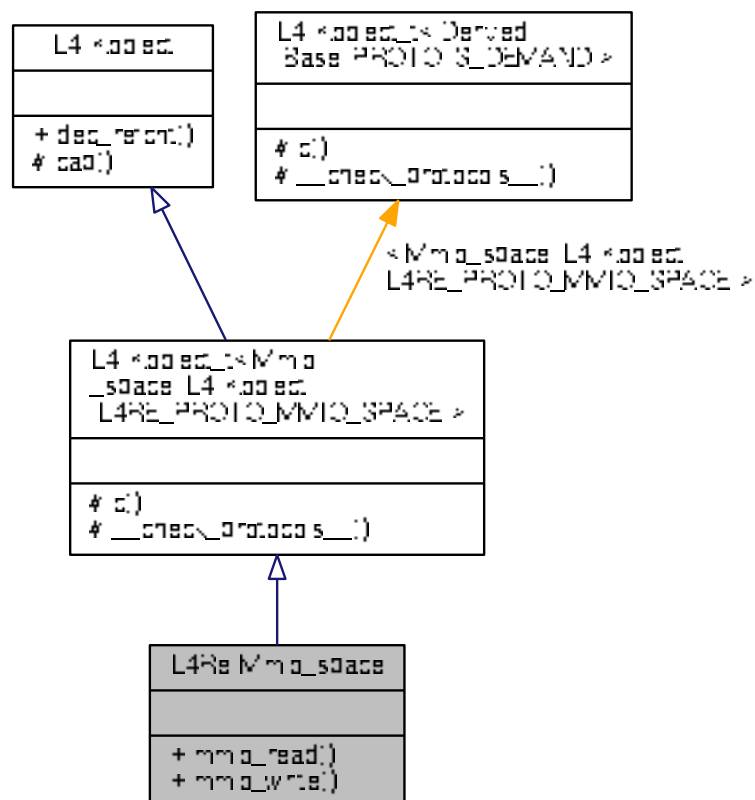
The documentation for this class was generated from the following files:

- [l4/re/mem\\_alloc](#)
- [l4/re/impl/mem\\_alloc\\_impl.h](#)

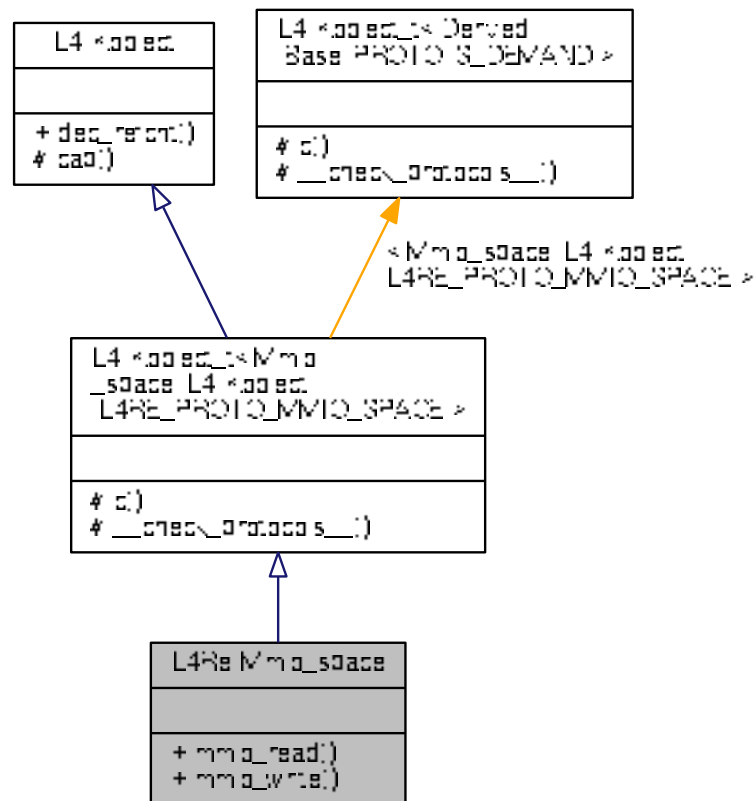
## 14.233 L4Re::Mmio\_space Struct Reference

Interface for memory-like address space accessible via IPC.

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

- long `mmio_read` (`Addr addr`, char width, `l4_uint64_t *value`)  
*Read a value from the given address.*
- long `mmio_write` (`Addr addr`, char width, `l4_uint64_t value`)  
*Write a value to the given address.*

## Additional Inherited Members

### 14.233.1 Detailed Description

Interface for memory-like address space accessible via IPC.

#### Include File

```
#include <l4/re/mmio_space>
```

Definition at line 25 of file [mmio\\_space](#).

### 14.233.2 Member Enumeration Documentation

#### 14.233.2.1 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 29 of file [mmio\\_space](#).

### 14.233.3 Member Function Documentation

#### 14.233.3.1 mmio\_read()

```
long 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 <a href="#">Access_width</a> .                           |
| out | <i>value</i> | Return value. If width is smaller than 64 bit, the upper bits are guaranteed to be 0.          |

## Return values

|                   |                                                                    |
|-------------------|--------------------------------------------------------------------|
| <i>L4_OK</i>      | Success.                                                           |
| <i>-L4_EPERM</i>  | Insufficient read rights.                                          |
| <i>-L4_EINVAL</i> | Address does not exist or cannot be accessed with the given width. |

## 14.233.3.2 mmio\_write()

```
long 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 <a href="#">Access_width</a> .                            |
| <i>value</i> | Value to write. If width is smaller than 64 bit, the upper bits are ignored.                  |

## Return values

|                   |                                                                    |
|-------------------|--------------------------------------------------------------------|
| <i>L4_OK</i>      | Success.                                                           |
| <i>-L4_EPERM</i>  | Insufficient write rights.                                         |
| <i>-L4_EINVAL</i> | Address does not exist or cannot be accessed with the given width. |

The documentation for this struct was generated from the following file:

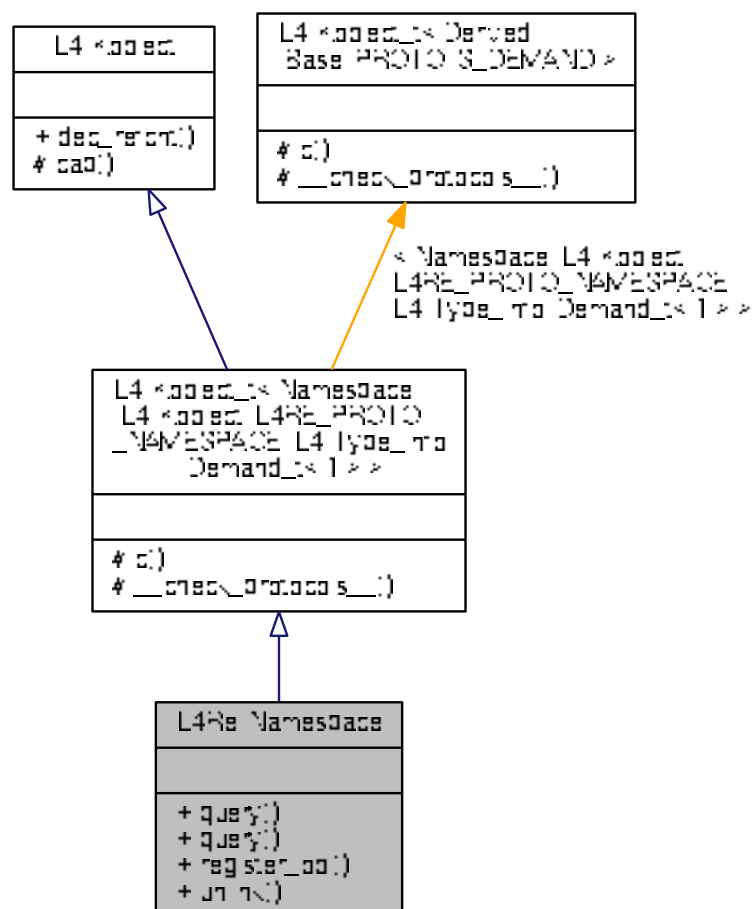
- l4/re/mmio\_space

## 14.234 L4Re::Namespace Class Reference

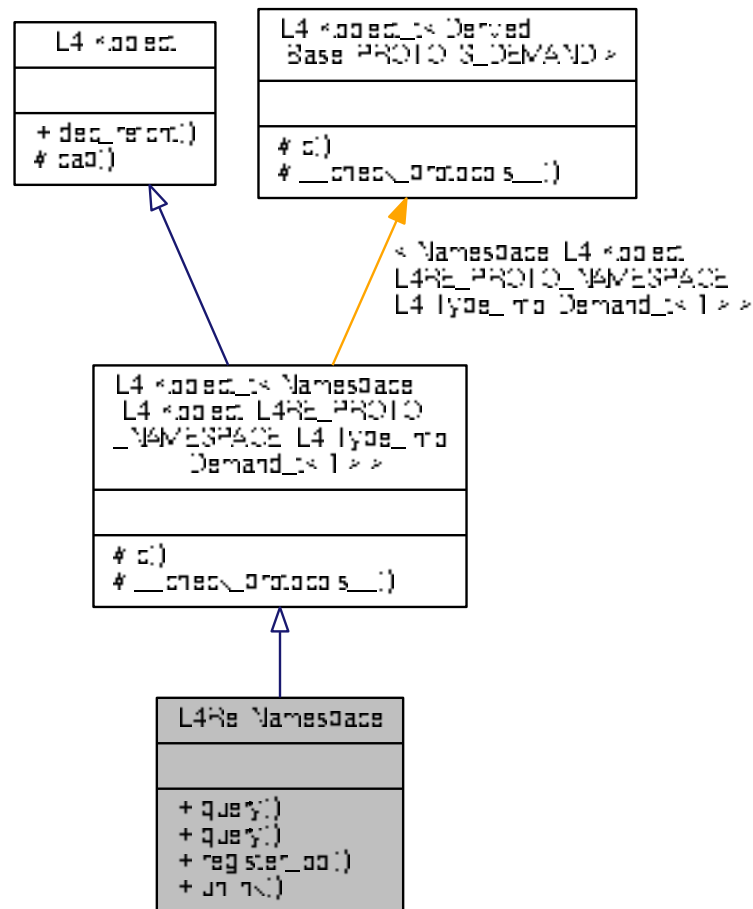
Name-space interface.



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`, `Link` = `0x100`, `Overwrite` = `0x200` }  
*Flags for registering name spaces.*
- enum `Query_result_flags` { `Partly_resolved` = `0x020` }  
*Flags returned by query IPC, only used internally.*

## Public Member Functions

- long `query` (char const \*name, `L4::Cap`< void > const &cap, int timeout=To\_default, `l4_umword_t` \*local\_id=0, bool iterate=true) const throw ()  
*Query the name space for a named object.*
- long `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 throw ()

*Query the name space for a named object.*

- long `register_obj` (char const \*name, `L4::lpc::Cap`< void > obj, unsigned flags=`Rw`) const throw ()

*Register an object with a name.*

- long `unlink` (char const \*name)

*Remove an entry from the name space.*

## Additional Inherited Members

### 14.234.1 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 60 of file `namespace`.

### 14.234.2 Member Enumeration Documentation

#### 14.234.2.1 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 88 of file `namespace`.

#### 14.234.2.2 Register\_flags

```
enum L4Re::Namespace::Register_flags
```

Flags for registering name spaces.

Enumerator

|    |            |
|----|------------|
| Ro | Read-only. |
|----|------------|

## Enumerator

|           |                                        |
|-----------|----------------------------------------|
| 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 68 of file [namespace](#).

## 14.234.3 Member Function Documentation

14.234.3.1 `query()` [1/2]

```
long 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 throw ()
```

Query the name space for a named object.

## Parameters

|     |                 |                                                                                                                                                                                                                  |
|-----|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| in  | <i>name</i>     | String to query (without any leading slashes).                                                                                                                                                                   |
| in  | <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, <a href="#">L4_RCV_ITEM_LOCAL_ID</a> 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

|            |                                                                                     |
|------------|-------------------------------------------------------------------------------------|
| 0          | Name could be fully resolved.                                                       |
| >0         | Name could only be partly resolved. The number of remaining characters is returned. |
| -L4_ENOENT | Entry could not be found.                                                           |
| -L4_EAGAIN | Entry exists but no object is yet attached. Try again later.                        |
| <0         | IPC errors, see <a href="#">l4_error_code_t</a> .                                   |

Definition at line 118 of file [namespace\\_impl.h](#).

## 14.234.3.2 query() [2/2]

```

long 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 throw ()

```

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).                                                                                                                                                                   |
| in  | <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, <a href="#">L4_RCV_ITEM_LOCAL_ID</a> 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.                                                                                           |

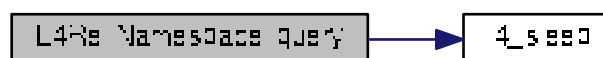
## Return values

|            |                                                                                     |
|------------|-------------------------------------------------------------------------------------|
| 0          | Name could be fully resolved.                                                       |
| >0         | Name could only be partly resolved. The number of remaining characters is returned. |
| -L4_ENOENT | Entry could not be found.                                                           |
| -L4_EAGAIN | Entry exists but no object is yet attached. Try again later.                        |
| <0         | IPC errors, see <a href="#">l4_error_code_t</a> .                                   |

Definition at line 77 of file [namespace\\_impl.h](#).

References [L4\\_EAGAIN](#), [L4\\_EINVAL](#), [l4\\_sleep\(\)](#), and [L4\\_UNLIKELY](#).

Here is the call graph for this function:



### 14.234.3.3 register\_obj()

```
long L4Re::Namespace::register_obj (
 char const * name,
 L4::Ipc::Cap< void > obj,
 unsigned flags = Rw) const throw () [inline]
```

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 <a href="#">L4Re::Namespace::Register_flags</a> . 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

|            |                                                       |
|------------|-------------------------------------------------------|
| 0          | Object was successfully registered with <i>name</i> . |
| -L4_EEXIST | Name already registered.                              |
| -L4_EPERM  | Caller doesn't have necessary permissions.            |
| -L4_ENOMEM | Server has insufficient resources.                    |
| -L4_EINVAL | Invalid parameter.                                    |
| <0         | IPC errors, see <a href="#">l4_error_code_t</a> .     |

#### Precondition

requires capability rights: {RW}

Definition at line 176 of file [namespace](#).

References [L4\\_RPC\\_NF\\_OP](#).

### 14.234.3.4 unlink()

```
long 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   | Caller does not have write permission.            |
| -L4_EACCESS | Name cannot be removed.                           |
| <0          | IPC errors, see <a href="#">l4_error_code_t</a> . |

## Precondition

requires capability rights: {RW}

Definition at line 202 of file [namespace](#).

The documentation for this class was generated from the following files:

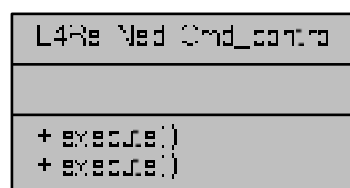
- [l4/re/namespace](#)
- [l4/re/impl/namespace\\_impl.h](#)

## 14.235 L4Re::Ned::Cmd\_control Class Reference

Direct control interface for Ned.

Inherits L4::Kobject\_0t< Derived, PROTO, S\_DEMAND >.

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.*

### 14.235.1 Detailed Description

Direct control interface for Ned.

Definition at line 20 of file [cmd\\_control](#).

### 14.235.2 Member Function Documentation

#### 14.235.2.1 `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 43 of file [cmd\\_control](#).

#### 14.235.2.2 `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

|                         |                                 |
|-------------------------|---------------------------------|
| <code>L4_EOK</code>     | Code was successfully executed. |
| <code>-L4_EINVAL</code> | Code could not be parsed.       |
| <code>-L4_EIO</code>    | 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 65 of file [cmd\\_control](#).

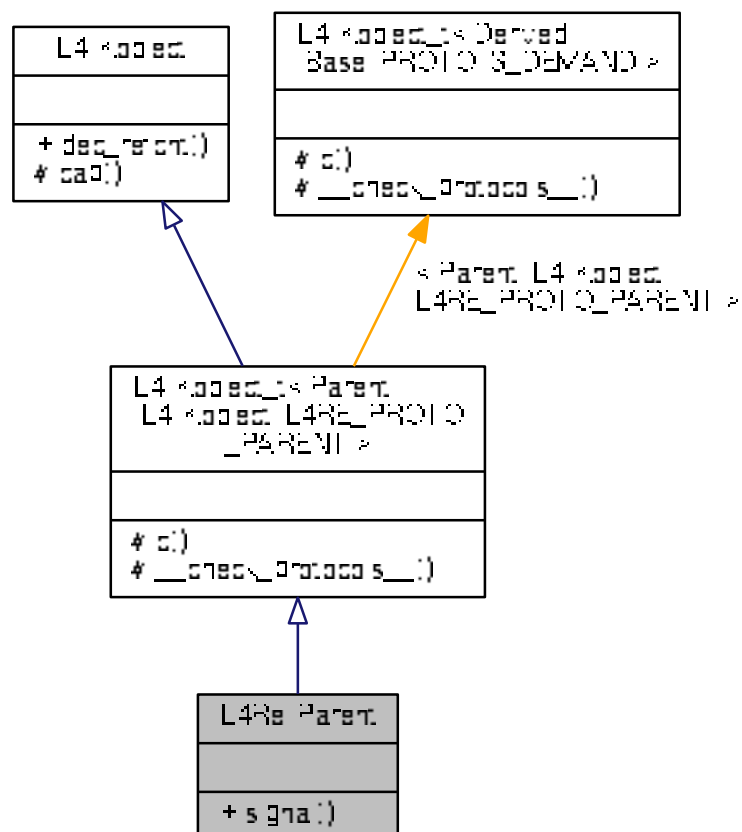
The documentation for this class was generated from the following file:

- `pkg/l4re-core/ned/lib/include/cmd_control`

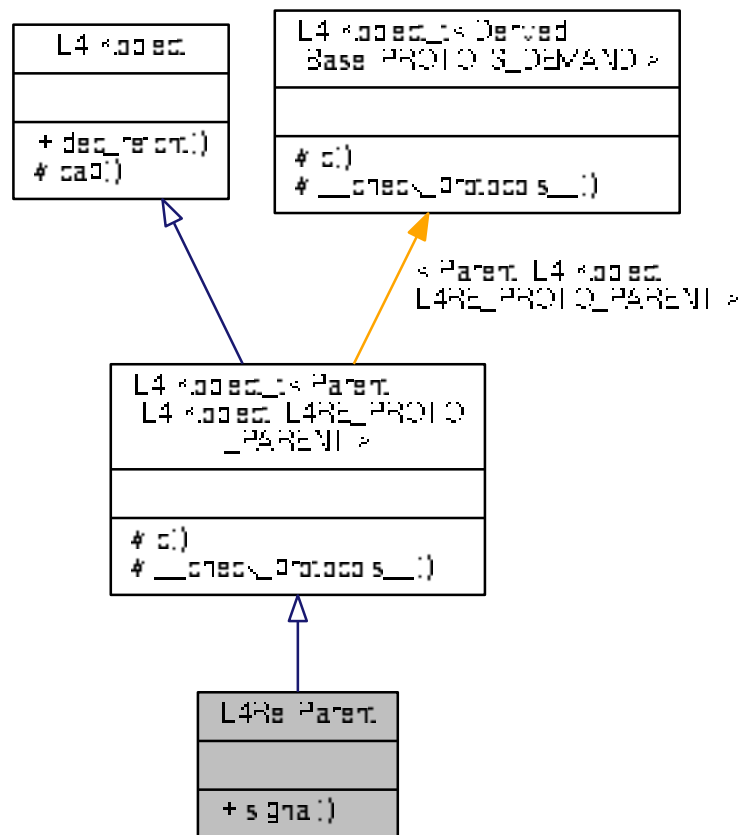
## 14.236 L4Re::Parent Class Reference

[Parent](#) interface.

Inheritance diagram for L4Re::Parent:



Collaboration diagram for L4Re::Parent:



## Public Member Functions

- long [signal](#) (unsigned long sig, unsigned long val)  
*Send a signal to the parent.*

## Additional Inherited Members

### 14.236.1 Detailed Description

[Parent](#) interface.

See also

[Parent API](#) for more details about the purpose.

Definition at line 51 of file [parent](#).

## 14.236.2 Member Function Documentation

### 14.236.2.1 signal()

```
long 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 |

#### Returns

0 on success, <0 on error

- [-L4\\_ENOREPLY](#)
- IPC errors

The documentation for this class was generated from the following file:

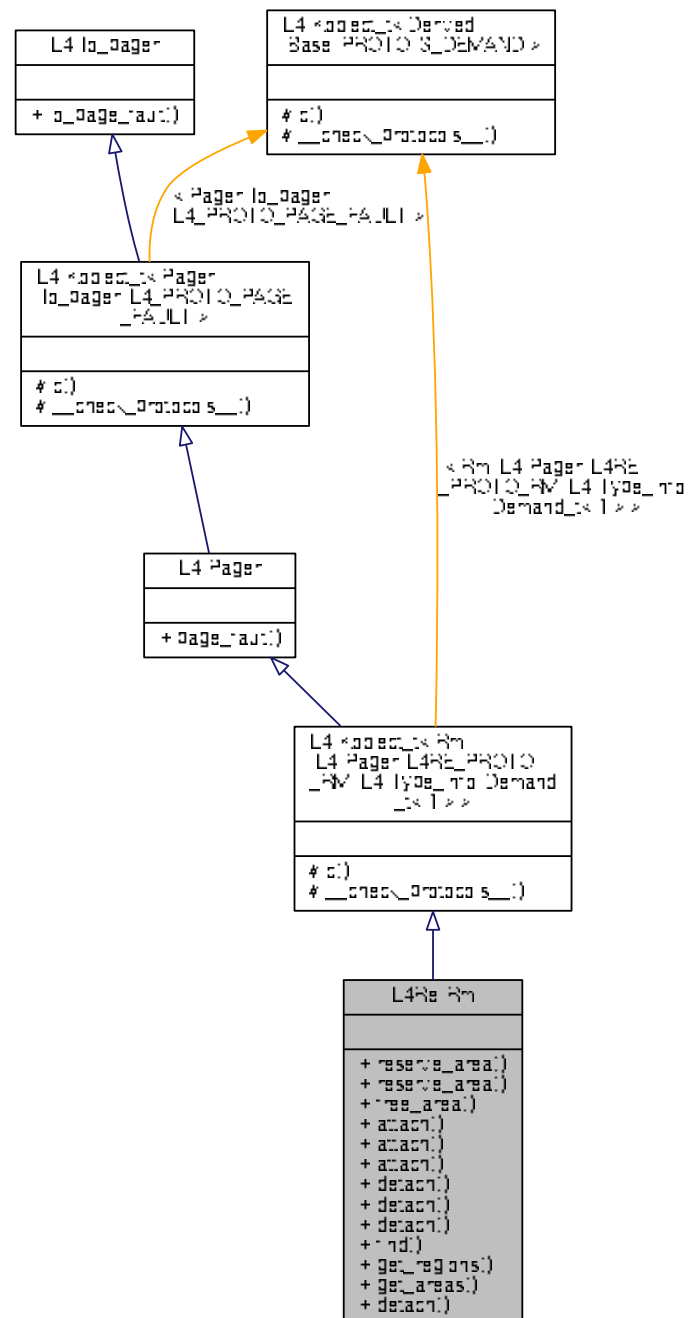
- [l4/re/parent](#)

## 14.237 L4Re::Rm Class Reference

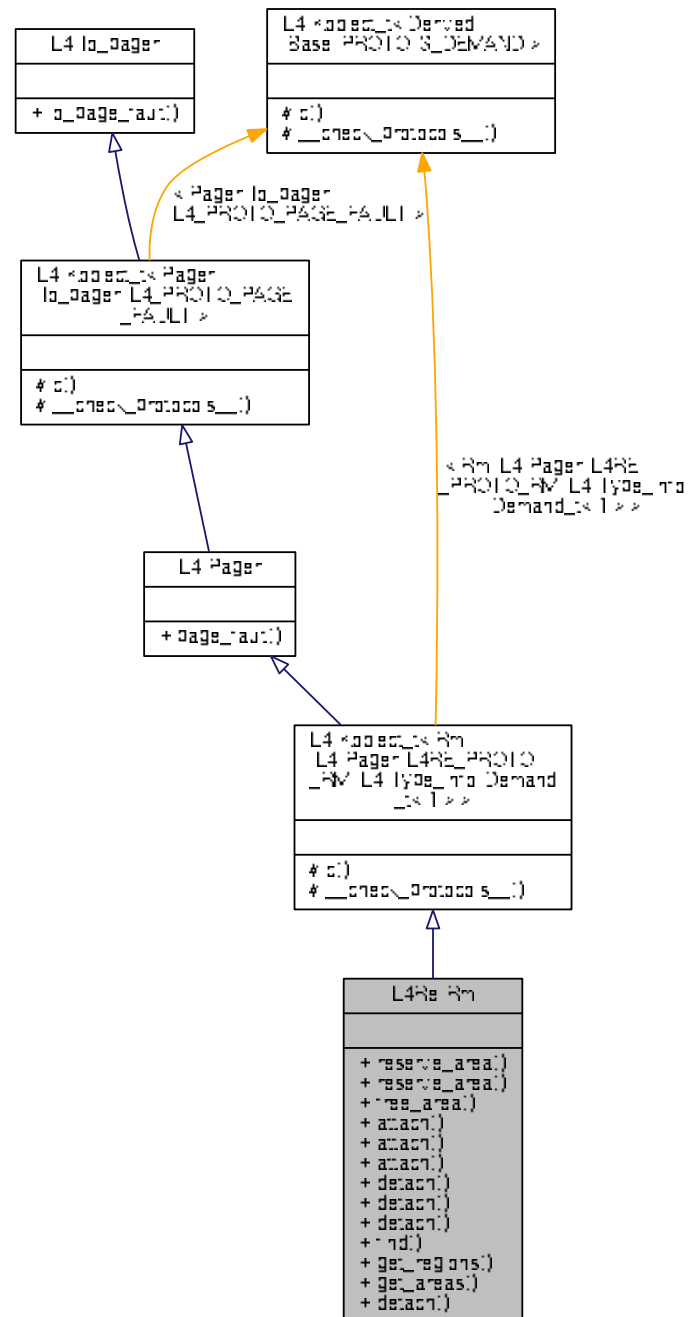
Region map.

```
#include <l4/re/rm>
```

Inheritance diagram for L4Re::Rm:



Collaboration diagram for L4Re::Rm:



## Public Types

- enum [Detach\\_result](#) { [Detached\\_ds](#) = 0, [Kept\\_ds](#) = 1, [Split\\_ds](#) = 2, [Detach\\_again](#) = 4 }
- Result values for detach operation.*
- enum [Region\\_flags](#) {  
[Read\\_only](#) = 0x01, [Detach\\_free](#) = 0x02, [Pager](#) = 0x04, [Reserved](#) = 0x08,  
[Caching\\_shift](#) = 8, [Caching\\_ds\\_shift](#) = [Caching\\_shift](#) - [Dataspace::Map\\_caching\\_shift](#), [Caching](#) =

```
Dataspace::Map_caching_mask << Caching_ds_shift, Cache_normal = Dataspace::Map_normal <<
Caching_ds_shift,
Cache_buffered = Dataspace::Map_bufferable << Caching_ds_shift, Cache_uncached = Dataspace::
Map_uncacheable << Caching_ds_shift, Region_flags = Caching | 0x0f }
```

*Flags for regions.*

- enum `Attach_flags` { `Search_addr` = 0x20, `In_area` = 0x40, `Eager_map` = 0x80, `Attach_flags` = 0xf0 }

*Flags for attach operation.*

- enum `Detach_flags` { `Detach_exact` = 1, `Detach_overlap` = 2, `Detach_keep` = 4 }

*Flags for detach operation.*

## Public Member Functions

- long `reserve_area` (`l4_addr_t` \*start, unsigned long size, unsigned flags=0, unsigned char align=`L4_PAGE←SHIFT`) const throw ()

*Reserve the given area in the region map.*

- template<typename T >  
long `reserve_area` (T \*\*start, unsigned long size, unsigned flags=0, unsigned char align=`L4_PAGESHIFT`) const throw ()

*Reserve the given area in the region map.*

- long `free_area` (`l4_addr_t` addr)

*Free an area from the region map.*

- long `attach` (`l4_addr_t` \*start, unsigned long size, unsigned long flags, `L4::lpc::Cap`< `Dataspace` > mem, `l4_addr_t` offs=0, unsigned char align=`L4_PAGESHIFT`) const throw ()

*Attach a data space to a region.*

- template<typename T >  
long `attach` (T \*\*start, unsigned long size, unsigned long flags, `L4::lpc::Cap`< `Dataspace` > mem, `l4_addr_t` offs=0, unsigned char align=`L4_PAGESHIFT`) const throw ()

*Attach a data space to a region.*

- int `detach` (`l4_addr_t` addr, `L4::Cap`< `Dataspace` > \*mem, `L4::Cap`< `L4::Task` > const &task=`This_task`) const throw ()

*Detach a region from the address space.*

- int `detach` (void \*addr, `L4::Cap`< `Dataspace` > \*mem, `L4::Cap`< `L4::Task` > const &task=`This_task`) const throw ()

*Detach a region from the address space.*

- int `detach` (`l4_addr_t` start, unsigned long size, `L4::Cap`< `Dataspace` > \*mem, `L4::Cap`< `L4::Task` > const &task) const throw ()

*Detach all regions of the specified interval.*

- int `find` (`l4_addr_t` \*addr, unsigned long \*size, `l4_addr_t` \*offset, unsigned \*flags, `L4::Cap`< `Dataspace` > \*m) throw ()

*Find a region given an address and size.*

## Additional Inherited Members

### 14.237.1 Detailed Description

Region map.

Definition at line 69 of file `rm`.

## 14.237.2 Member Enumeration Documentation

### 14.237.2.1 Attach\_flags

enum [L4Re::Rm::Attach\\_flags](#)

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. |
| Attach_flags | Mask of all attach flags.               |

Definition at line 111 of file [rm](#).

### 14.237.2.2 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 <a href="#">Detach_free</a> . |

Definition at line 121 of file [rm](#).

### 14.237.2.3 Detach\_result

enum [L4Re::Rm::Detach\\_result](#)

Result values for detach operation.

#### Enumerator

|              |                                  |
|--------------|----------------------------------|
| Detached_ds  | Detached data sapce.             |
| Kept_ds      | Kept data space.                 |
| Split_ds     | Splitted data space, and done.   |
| Detach_again | Detached data space, more to do. |

Definition at line 75 of file [rm](#).

#### 14.237.2.4 Region\_flags

```
enum L4Re::Rm::Region_flags
```

Flags for regions.

##### Enumerator

|                  |                                                                             |
|------------------|-----------------------------------------------------------------------------|
| Read_only        | Region is read-only.                                                        |
| Detach_free      | Free the portion of the data space after detach.                            |
| Pager            | Region has a pager.                                                         |
| Reserved         | Region is reserved (blocked)                                                |
| Caching_shift    | Start of <a href="#">Rm</a> cache bits.                                     |
| Caching_ds_shift | Shift value for <a href="#">Dataspace</a> to <a href="#">Rm</a> cache bits. |
| Caching          | Mask of all <a href="#">Rm</a> cache bits.                                  |
| Cache_normal     | Cache bits for normal cacheable memory.                                     |
| Cache_buffered   | Cache bits for buffered (write combining) memory.                           |
| Cache_uncached   | Cache bits for uncached memory.                                             |
| Region_flags     | Mask of all region flags.                                                   |

Definition at line 86 of file [rm](#).

### 14.237.3 Member Function Documentation

#### 14.237.3.1 attach() [1/2]

```
long L4Re::Rm::attach (
 l4_addr_t * start,
 unsigned long size,
 unsigned long flags,
 L4::Ipc::Cap< Dataspace > mem,
 l4_addr_t offs = 0,
 unsigned char align = L4_PAGESHIFT) const throw ()
```

Attach a data space to a region.

##### Parameters

|         |       |                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|---------|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| in, out | start | Virtual start address where the region manager shall attach the data space. If <a href="#">L4Re::Rm::Search_addr</a> 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 <a href="#">L4Re::Rm::In_area</a> is given the value is used as a selector for the area (see <a href="#">L4Re::Rm::reserve_area</a> ) to attach the data space to. |
|---------|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|



## Parameters

|  |              |                                                                                                                                                                                                                                               |
|--|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|  | <i>size</i>  | Size of the data space to attach (in bytes)                                                                                                                                                                                                   |
|  | <i>flags</i> | Flags, see <a href="#">L4Re::Rm::Attach_flags</a> and <a href="#">L4Re::Rm::Region_flags</a> . If the <code>Eager_map</code> flag is set this function may also return <a href="#">L4Re::Dataspace::map</a> 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 ( <a href="#">L4_PAGESHIFT</a> ). This is only meaningful if the <a href="#">L4Re::Rm::Search_addr</a> flag is used.                                                              |

## Return values

|                   |                                                                 |
|-------------------|-----------------------------------------------------------------|
| 0                 | Success                                                         |
| -L4_ENOENT        | No area could be found (see <a href="#">L4Re::Rm::In_area</a> ) |
| -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

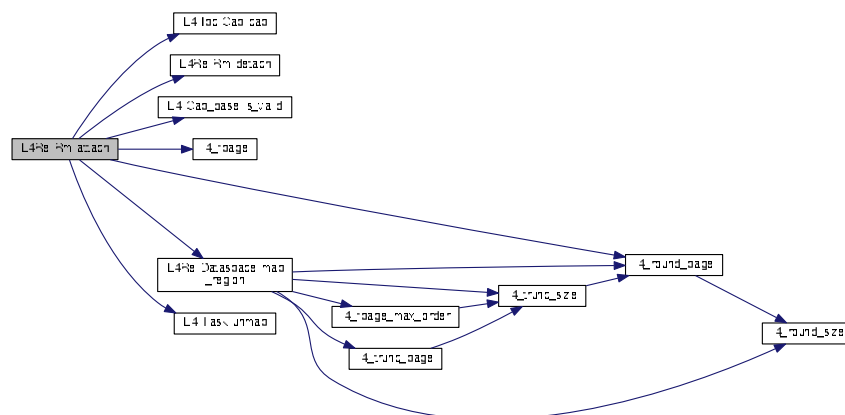
When searching for a free place in the virtual address space, the space between *start* and the end of the virtual address space is searched.

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 43 of file [rm\\_impl.h](#).

References [L4::ipc::Cap< T >::cap\(\)](#), [detach\(\)](#), [L4::Cap\\_base::is\\_valid\(\)](#), [L4\\_FP\\_ALL\\_SPACES](#), [l4\\_fpage\(\)](#), [L4\\_FPAGE\\_RWX](#), [L4\\_INVALID\\_CAP](#), [L4\\_LOG2\\_PAGESIZE](#), [l4\\_round\\_page\(\)](#), [L4\\_UNLIKELY](#), [L4Re::Dataspace::map\\_region\(\)](#), [L4Re::Dataspace::Map\\_ro](#), [L4Re::Dataspace::Map\\_rw](#), and [L4::Task::unmap\(\)](#).

Here is the call graph for this function:



**14.237.3.2 attach()** [2/2]

```
template<typename T >
long L4Re::Rm::attach (
 T ** start,
 unsigned long size,
 unsigned long flags,
 L4::Ipc::Cap< Dataspace > mem,
 l4_addr_t offs = 0,
 unsigned char align = L4_PAGESHIFT) const throw () [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. If <a href="#">L4Re::Rm::Search_addr</a> 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 <a href="#">L4Re::Rm::In_area</a> is given the value is used as a selector for the area (see <a href="#">L4Re::Rm::reserve_area</a> ) to attach the data space to. |
|                | <i>size</i>  | Size of the data space to attach (in bytes)                                                                                                                                                                                                                                                                                                                                                                                        |
|                | <i>flags</i> | Flags, see <a href="#">L4Re::Rm::Attach_flags</a> and <a href="#">L4Re::Rm::Region_flags</a> . If the <code>Eager_map</code> flag is set this function may also return <a href="#">L4Re::Dataspace::map</a> 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 ( <a href="#">L4_PAGESHIFT</a> ). This is only meaningful if the <a href="#">L4Re::Rm::Search_addr</a> flag is used.                                                                                                                                                                                                                                                   |

**Return values**

|                          |                                                                 |
|--------------------------|-----------------------------------------------------------------|
| <i>0</i>                 | Success                                                         |
| <i>-L4_ENOENT</i>        | No area could be found (see <a href="#">L4Re::Rm::In_area</a> ) |
| <i>-L4_EPERM</i>         | Operation not allowed.                                          |
| <i>-L4_EINVAL</i>        |                                                                 |
| <i>-L4_EADDRNOTAVAIL</i> | The given address is not available.                             |
| <i>&lt;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 345 of file [rm](#).

References [L4\\_PAGESHIFT](#).

## 14.237.3.3 detach() [1/3]

```
int L4Re::Rm::detach (
 l4_addr_t addr,
 L4::Cap< Dataspace > * mem,
 L4::Cap< L4::Task > const & task = This_task) const throw () [inline]
```

Detach 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>  | <a href="#">Dataspace</a> that is affected. Give 0 if not interested.                                                                                              |
|     | <i>task</i> | This argument specifies the task where the pages are unmapped. Provide <a href="#">L4::Cap&lt;L4::Task&gt;::Invalid</a> for none. The default is the current task. |

## Return values

|                               |                  |
|-------------------------------|------------------|
| <a href="#">Detach_result</a> | On success.      |
| <a href="#">-L4_ENOENT</a>    | 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 497 of file [rm](#).

Referenced by [attach\(\)](#).

Here is the caller graph for this function:



## 14.237.3.4 detach() [2/3]

```
int L4Re::Rm::detach (
 void * addr,
 L4::Cap< Dataspace > * mem,
 L4::Cap< L4::Task > const & task = This_task) const throw () [inline]
```

Detach 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>  | <a href="#">Dataspace</a> that is affected. Give 0 if not interested.                                                                                              |
|     | <i>task</i> | This argument specifies the task where the pages are unmapped. Provide <a href="#">L4::Cap&lt;L4::Task&gt;::Invalid</a> for none. The default is the current task. |

## Return values

|                               |                  |
|-------------------------------|------------------|
| <a href="#">Detach_result</a> | On success.      |
| <a href="#">-L4_ENOENT</a>    | 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 502 of file [rm](#).

## 14.237.3.5 detach() [3/3]

```
int L4Re::Rm::detach (
 l4_addr_t start,
 unsigned long size,
 L4::Cap< Dataspace > * mem,
 L4::Cap< L4::Task > const & task) const throw () [inline]
```

Detach all regions of 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>   | <a href="#">Dataspace</a> that is affected. Give 0 if not interested.                                                                                              |
|     | <i>task</i>  | This argument specifies the task where the pages are unmapped. Provide <a href="#">L4::Cap&lt;L4::Task&gt;::Invalid</a> for none. The default is the current task. |

## Return values

|                               |                  |
|-------------------------------|------------------|
| <a href="#">Detach_result</a> | On success.      |
| <a href="#">-L4_ENOENT</a>    | 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 507 of file [rm](#).

## 14.237.3.6 find()

```
int L4Re::Rm::find (
 l4_addr_t * addr,
 unsigned long * size,
 l4_addr_t * offset,
 unsigned * flags,
 L4::Cap< Dataspace > * m) throw () [inline]
```

Find a region given an address and size.

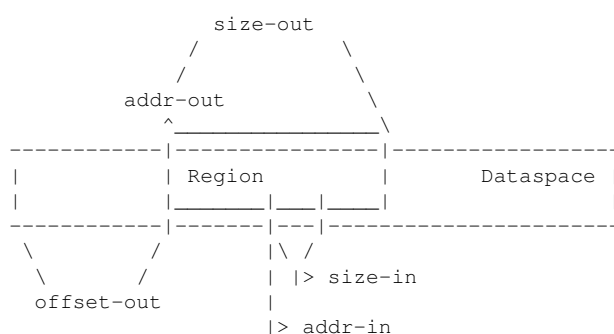
## Parameters

|     |               |                                                                                |
|-----|---------------|--------------------------------------------------------------------------------|
|     | <i>addr</i>   | Address to look for                                                            |
|     | <i>size</i>   | Size of the area to look for (in bytes).                                       |
| out | <i>addr</i>   | Start address of the found region.                                             |
| out | <i>size</i>   | 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 <a href="#">Region_flags</a> (and <a href="#">ln_area</a> ). |
| 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 [ln\\_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 461 of file [rm](#).

References [L4\\_RPC](#), and [L4\\_RPC\\_NF](#).

14.237.3.7 `free_area()`

```
long 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

|                   |                |
|-------------------|----------------|
| <i>0</i>          | Success        |
| <i>-L4_ENOENT</i> | No area found. |
| <i>&lt;0</i>      | 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.

14.237.3.8 `reserve_area()` [1/2]

```
long L4Re::Rm::reserve_area (
 l4_addr_t * start,
 unsigned long size,
 unsigned flags = 0,
 unsigned char align = L4_PAGESHIFT) const throw () [inline]
```

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 <a href="#">L4Re::Rm::Region_flags</a> and <a href="#">L4Re::Rm::Attach_flags</a> ). |
|                | <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>&lt;0</i>             | IPC errors                         |

This function reserves an area within the virtual address space implemented by the region map. There are two kinds of areas available:

- Reserved areas (*flags* = [Reserved](#)), where no data spaces can be attached
- Special purpose areas (*flags* = 0), where data spaces can be attached to the area via the [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* = [Search\\_addr](#)), the space between *start* and the end of the virtual address space is searched.

Definition at line [239](#) of file [rm](#).

References [L4\\_RPC\\_NF](#).

#### 14.237.3.9 `reserve_area()` [2/2]

```
template<typename T >
long L4Re::Rm::reserve_area (
 T ** start,
 unsigned long size,
 unsigned flags = 0,
 unsigned char align = L4_PAGESHIFT) const throw () [inline]
```

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 <a href="#">Region_flags</a> and <a href="#">Attach_flags</a> ). |
|                | <i>align</i> | Alignment of area if searched as bits (log2 value).                                               |

#### Return values

|                                   |                                    |
|-----------------------------------|------------------------------------|
| 0                                 | Success                            |
| <a href="#">-L4_EADDRNOTAVAIL</a> | 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 [265](#) of file [rm](#).

References [L4\\_PAGESHIFT](#), [L4\\_RPC](#), and [L4\\_RPC\\_NF](#).

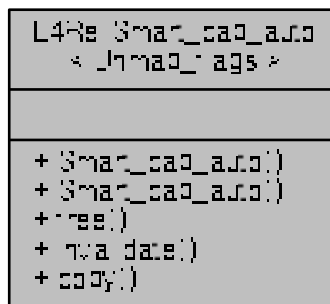
The documentation for this class was generated from the following files:

- [l4/re/rm](#)
- [l4/re/impl/rm\\_impl.h](#)

## 14.238 L4Re::Smart\_cap\_auto< Unmap\_flags > Class Template Reference

Helper for Auto\_cap and Auto\_del\_cap.

Collaboration diagram for L4Re::Smart\_cap\_auto< Unmap\_flags >:



### 14.238.1 Detailed Description

```
template<unsigned long Unmap_flags = L4_FP_ALL_SPACES>
class L4Re::Smart_cap_auto< Unmap_flags >
```

Helper for Auto\_cap and Auto\_del\_cap.

Definition at line 110 of file [cap\\_alloc](#).

The documentation for this class was generated from the following file:

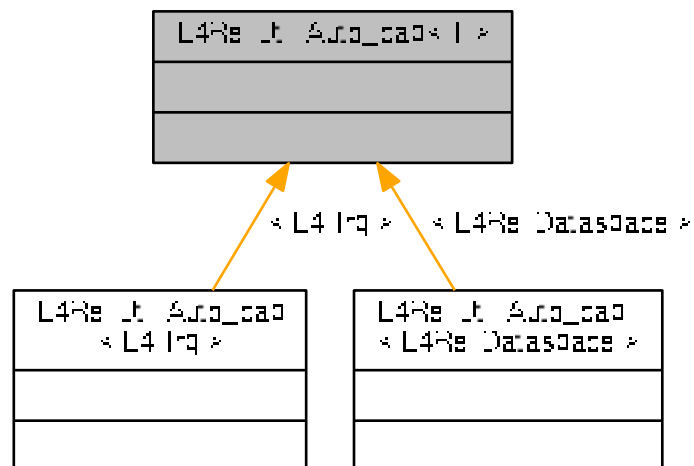
- [l4/re/cap\\_alloc](#)



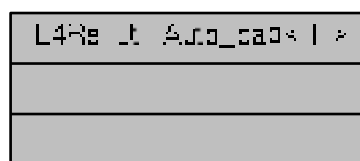
## 14.239 L4Re::Util::Auto\_cap< T > Struct Template Reference

Automatic capability that implements automatic free and unmap of the capability selector.

Inheritance diagram for L4Re::Util::Auto\_cap< T >:



Collaboration diagram for L4Re::Util::Auto\_cap< T >:



### 14.239.1 Detailed Description

```

template<typename T>
struct L4Re::Util::Auto_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 is useful for capabilities that shall have a lifetime that is strictly coupled to one C++ scope.

## Usage:

```
{
 L4Re::Util::Auto_cap<L4Re::Dataspace>::Cap
 ds_cap(L4Re::Util::cap_alloc.alloc<L4Re::Dataspace>());

 // use the dataspace cap
 L4Re::chksys(mem_alloc->alloc(4096, ds_cap.get()));

 ...

 // At the end of the scope ds_cap is unmapped and the capability selector
 // is freed.
}
```

Definition at line 161 of file [cap\\_alloc](#).

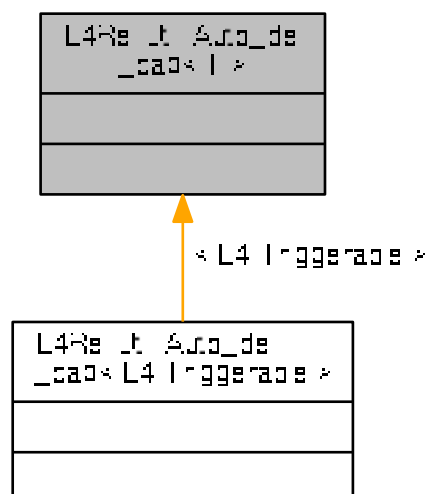
The documentation for this struct was generated from the following file:

- [l4/re/util/cap\\_alloc](#)

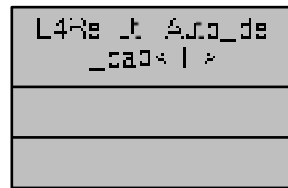
## 14.240 L4Re::Util::Auto\_del\_cap< T > Struct Template Reference

Automatic capability that implements automatic free and unmap+delete of the capability selector.

Inheritance diagram for L4Re::Util::Auto\_del\_cap< T >:



Collaboration diagram for L4Re::Util::Auto\_del\_cap< T >:



### 14.240.1 Detailed Description

```
template<typename T>
struct L4Re::Util::Auto_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 is useful for capabilities with that shall have a lifetime that is strictly coupled to one C++ scope. The main difference to [Auto\\_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::Auto_del_cap<L4Re::Dataspace>::Cap
 ds_cap(L4Re::Util::cap_alloc.alloc<L4Re::Dataspace>());

 // use the dataspace cap
 L4Re::chksys(mem_alloc->alloc(4096, 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 be
 // also deleted (even if there are other references to this data space).
}
```

Definition at line 196 of file [cap\\_alloc](#).

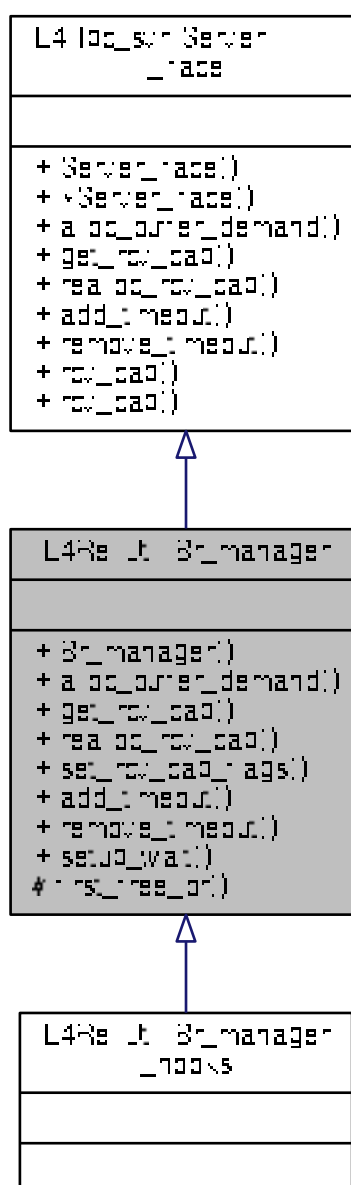
The documentation for this struct was generated from the following file:

- [l4/re/util/cap\\_alloc](#)

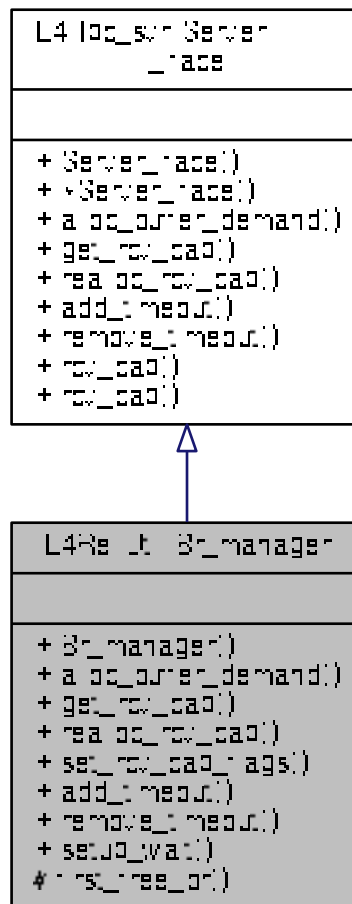
## 14.241 L4Re::Util::Br\_manager Class Reference

Buffer-register (BR) manager for [L4::Server](#).

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)  
*Tells the server to allocate buffers for the given demand.*
- `L4::Cap< void >` [get\\_rcv\\_cap](#) (int i) const  
*Get capability slot allocated to the given receive buffer.*
- `int` [realloc\\_rcv\\_cap](#) (int i)  
*Allocate a new capability for the given receive buffer.*
- `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`)  
*No timeouts handled by us.*
- `int` [remove\\_timeout](#) (`L4::lpc_svr::Timeout *`)  
*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)*

## Protected Member Functions

- unsigned [first\\_free\\_br](#) () const  
*Used for assigning BRs for a timeout.*

## Additional Inherited Members

### 14.241.1 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 36 of file [br\\_manager](#).

### 14.241.2 Member Function Documentation

#### 14.241.2.1 [alloc\\_buffer\\_demand\(\)](#)

```
int L4Re::Util::Br_manager::alloc_buffer_demand (
 Demand const & demand) [inline], [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 50 of file [br\\_manager](#).

#### 14.241.2.2 [get\\_rcv\\_cap\(\)](#)

```
L4::Cap<void> L4Re::Util::Br_manager::get_rcv_cap (
 int index) const [inline], [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 <a href="#">alloc_buffer_demand()</a> ). |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|

**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 81 of file [br\\_manager](#).

References [L4\\_CAP\\_MASK](#).

**14.241.2.3 realloc\_rcv\_cap()**

```
int L4Re::Util::Br_manager::realloc_rcv_cap (
 int index) [inline], [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 <a href="#">alloc_buffer_demand()</a> ). |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|

**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 89 of file [br\\_manager](#).

#### 14.241.2.4 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 <a href="#">l4_msg_item_consts_t</a> . |
|--------------|--------------------------------------------------------------------------|

Definition at line 113 of file [br\\_manager](#).

References [l4\\_assert](#).

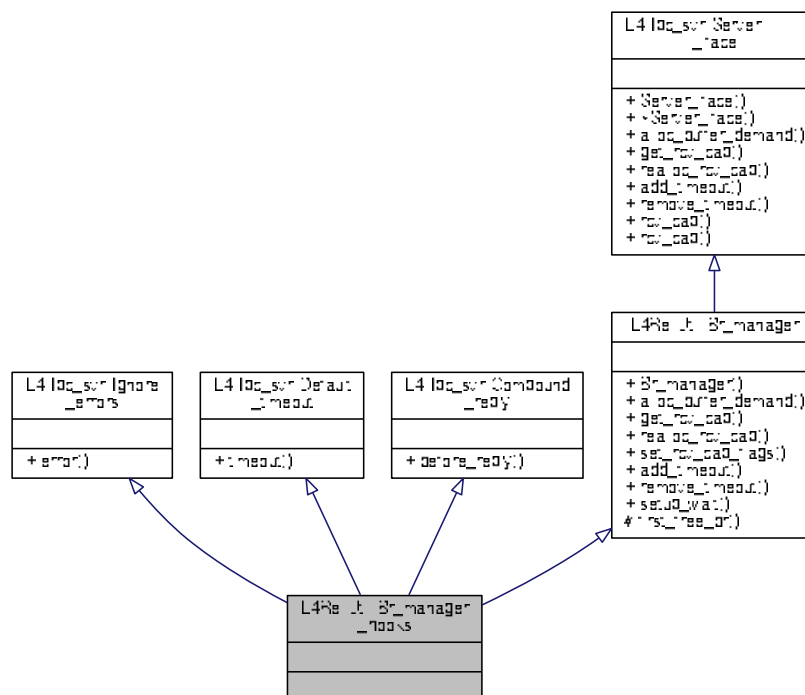
The documentation for this class was generated from the following file:

- l4/re/util/br\_manager

## 14.242 L4Re::Util::Br\_manager\_hooks Struct Reference

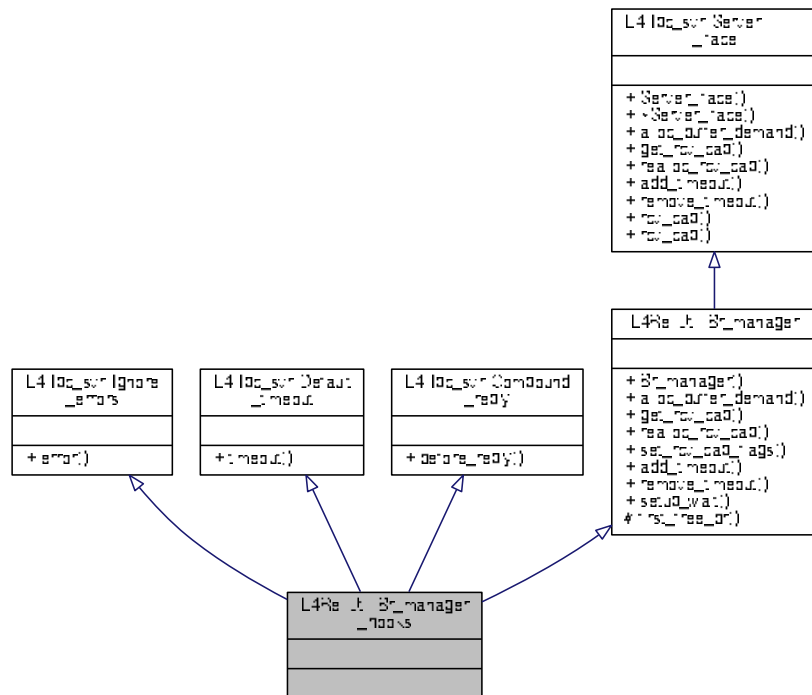
Predefined server-loop hooks for a server loop using the [Br\\_manager](#).

Inheritance diagram for L4Re::Util::Br\_manager\_hooks:





Collaboration diagram for L4Re::Util::Br\_manager\_hooks:



## Additional Inherited Members

### 14.242.1 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 160 of file [br\\_manager](#).

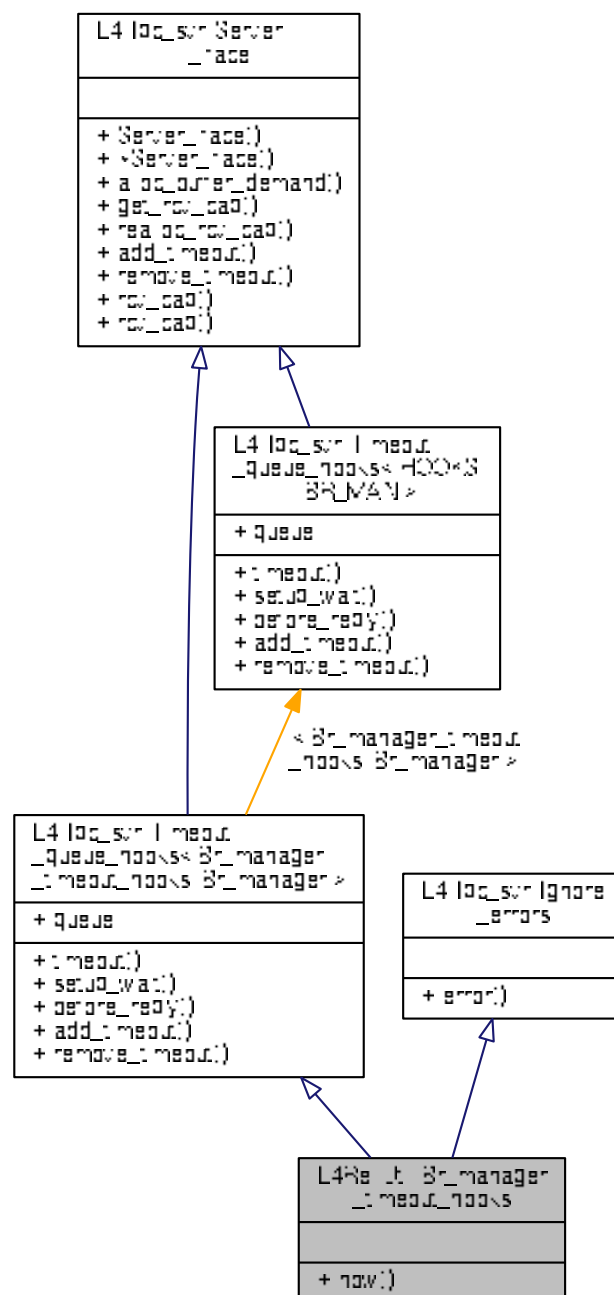
The documentation for this struct was generated from the following file:

- [l4/re/util/br\\_manager](#)

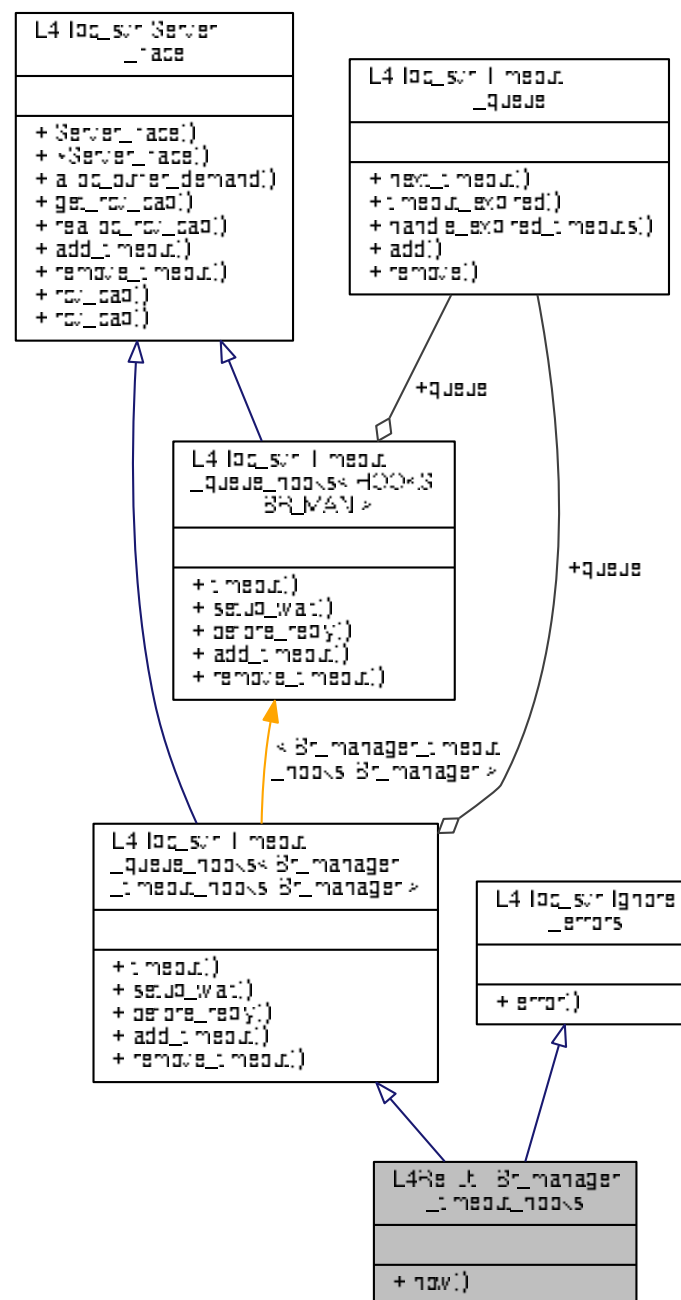
## 14.243 L4Re::Util::Br\_manager\_timeout\_hooks Struct Reference

Predefined server-loop hooks for a server with using the [Br\\_manager](#) and a timeout queue.

Inheritance diagram for L4Re::Util::Br\_manager\_timeout\_hooks:



Collaboration diagram for L4Re::Util::Br\_manager\_timeout\_hooks:



## Additional Inherited Members

### 14.243.1 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 174 of file [br\\_manager](#).

The documentation for this struct was generated from the following file:

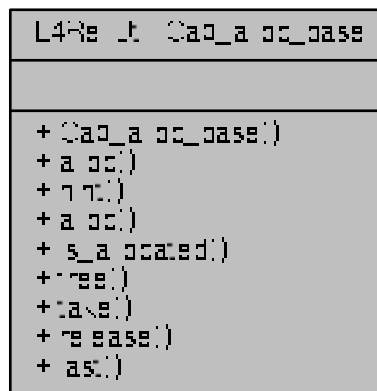
- [l4/re/util/br\\_manager](#)

## 14.244 L4Re::Util::Cap\_alloc\_base Class Reference

Capability allocator.

Inherited by [L4Re::Util::Cap\\_alloc< Size >](#).

Collaboration diagram for [L4Re::Util::Cap\\_alloc\\_base](#):



### Public Member Functions

- `template<typename T >`  
[L4::Cap< T >](#) [alloc](#) () throw ()  
*Allocate a capability slot.*
- `template<typename T >`  
 void [free](#) ([L4::Cap< T >](#) const &cap, [l4\\_cap\\_idx\\_t](#) task=-1UL, [l4\\_umword\\_t](#) unmap\_flags=L4\_FP\_ALL\_S←  
 PACES) throw ()  
*Free a capability slot.*

### 14.244.1 Detailed Description

Capability allocator.

Definition at line 38 of file [bitmap\\_cap\\_alloc](#).

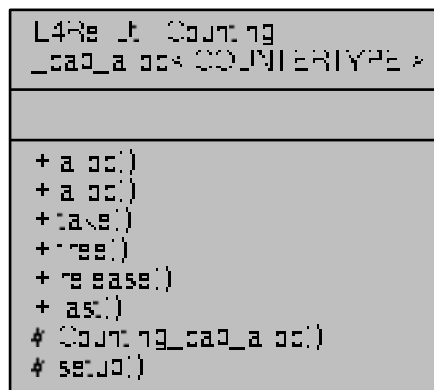
The documentation for this class was generated from the following file:

- [l4/re/util/bitmap\\_cap\\_alloc](#)

## 14.245 L4Re::Util::Counting\_cap\_alloc< COUNTERTYPE > Class Template Reference

Reference-counting cap allocator.

Collaboration diagram for L4Re::Util::Counting\_cap\_alloc< COUNTERTYPE >:



### Public Member Functions

- `L4::Cap< void > alloc () throw ()`  
*Allocated a new capability.*
- `template<typename T >`  
`L4::Cap< T > alloc () throw ()`  
*Allocated a new capability.*
- `void take (L4::Cap< void > cap) throw ()`  
*Increase the reference counter for the capability.*
- `bool free (L4::Cap< void > cap, l4_cap_idx_t task=L4_INVALID_CAP, unsigned unmap_flags=L4_FP_ALL_SPACES) throw ()`  
*Free the capability.*
- `bool release (L4::Cap< void > cap, l4_cap_idx_t task=L4_INVALID_CAP, unsigned unmap_flags=L4_FP_ALL_SPACES) throw ()`  
*Decrease the reference counter for a capability.*
- `long last () throw ()`  
*Return highest capability id managed by this allocator.*

### Protected Member Functions

- `Counting_cap_alloc () throw ()`  
*Create a new, empty allocator.*
- `void setup (void *m, long capacity, long bias) throw ()`  
*Set up the backing memory for the allocator.*

### 14.245.1 Detailed Description

```
template<typename COUNTERTYPE = L4Re::Util::Counter<unsigned char>>
class L4Re::Util::Counting_cap_alloc< COUNTERTYPE >
```

Reference-counting cap allocator.

#### Note

The operations in this class are not thread-safe.

Definition at line 56 of file [counting\\_cap\\_alloc](#).

### 14.245.2 Constructor & Destructor Documentation

#### 14.245.2.1 Counting\_cap\_alloc()

```
template<typename COUNTERTYPE = L4Re::Util::Counter<unsigned char>>
L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::Counting_cap_alloc () throw () [inline],
[protected]
```

Create a new, empty allocator.

Needs to be initialized with [setup\(\)](#) before it can be used.

Definition at line 85 of file [counting\\_cap\\_alloc](#).

### 14.245.3 Member Function Documentation

#### 14.245.3.1 alloc() [1/2]

```
template<typename COUNTERTYPE = L4Re::Util::Counter<unsigned char>>
L4::Cap<void> L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::alloc () throw () [inline]
```

Allocated a new capability.

#### Returns

The newly allocated capability, invalid if the allocator was exhausted.

#### 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+\\_ds/ds\\_srv.cc](#).

Definition at line 110 of file [counting\\_cap\\_alloc](#).

References [L4::Cap\\_base::Invalid](#), and [L4\\_CAP\\_SHIFT](#).

## 14.245.3.2 alloc() [2/2]

```
template<typename COUNTERTYPE = L4Re::Util::Counter<unsigned char>>
template<typename T >
L4::Cap<T> L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::alloc () throw () [inline]
```

Allocated a new capability.

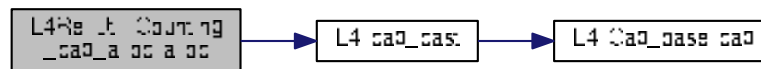
## Returns

The newly allocated capability, invalid if the allocator was exhausted.

Definition at line 131 of file [counting\\_cap\\_alloc](#).

References [L4::cap\\_cast\(\)](#).

Here is the call graph for this function:



## 14.245.3.3 free()

```
template<typename COUNTERTYPE = L4Re::Util::Counter<unsigned char>>
bool L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::free (
 L4::Cap< void > cap,
 l4_cap_idx_t task = L4_INVALID_CAP,
 unsigned unmap_flags = L4_FP_ALL_SPACES) throw () [inline]
```

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 <code>l4_unmap_flags_t</code> . |

## Precondition

The capability has been allocated. Calling free twice results in undefined behaviour.

**Returns**

True, if the capability was managed by this allocator.

**Examples:**

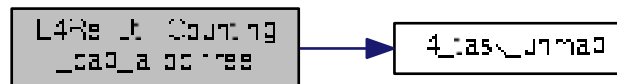
[examples/libs/l4re/c++/mem\\_alloc/ma+rm.cc](#), [examples/libs/l4re/c++/shared\\_ds/ds\\_clnt.cc](#), [examples/libs/l4re/c++/shared\\_ds/ds\\_srv.cc](#), and [examples/libs/l4re/streammap/client.cc](#).

Definition at line 172 of file [counting\\_cap\\_alloc](#).

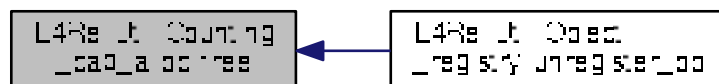
References [l4\\_assert](#), [L4\\_CAP\\_SHIFT](#), [L4\\_INVALID\\_CAP](#), and [l4\\_task\\_unmap\(\)](#).

Referenced by [L4Re::Util::Object\\_registry::unregister\\_obj\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**14.245.3.4 release()**

```

template<typename COUNTERTYPE = L4Re::Util::Counter<unsigned char>>
bool L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::release (
 L4::Cap< void > cap,
 l4_cap_idx_t task = L4_INVALID_CAP,
 unsigned unmap_flags = L4_FP_ALL_SPACES) throw () [inline]

```

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 <code>l4_unmap_flags_t</code> . |

## Precondition

The capability has been allocated. Calling release on a free capability results in undefined behaviour.

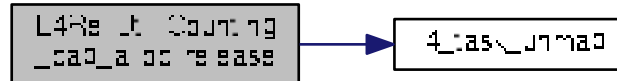
## Returns

True, if the capability was freed as a result of this operation. If false is returned the capability is either still in use or was already free or invalid.

Definition at line 211 of file `counting_cap_alloc`.

References `L4::dec`, `l4_assert`, `L4_CAP_SHIFT`, `L4_INVALID_CAP`, and `l4_task_unmap()`.

Here is the call graph for this function:



## 14.245.3.5 setup()

```

template<typename COUNTERTYPE = L4Re::Util::Counter<unsigned char>>
void L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::setup (
 void * m,
 long capacity,
 long bias) throw () [inline], [protected]

```

Set up the backing memory for the allocator.

## 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. |

Definition at line 96 of file [counting\\_cap\\_alloc](#).

#### 14.245.3.6 take()

```
template<typename COUNTERTYPE = L4Re::Util::Counter<unsigned char>>
void L4Re::Util::Counting_cap_alloc< COUNTERTYPE >::take (
 L4::Cap< void > cap) throw () [inline]
```

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 146 of file [counting\\_cap\\_alloc](#).

References [L4\\_CAP\\_SHIFT](#).

The documentation for this class was generated from the following file:

- [l4/re/util/counting\\_cap\\_alloc](#)

## 14.246 L4Re::Util::Dataspace\_svr Class Reference

[Dataspace](#) server class.

Collaboration diagram for L4Re::Util::Dataspace\_svr:

| L4Re::Util::Dataspace_svr                                                                                                                                                                                                                                                                                                     |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre># _ds_start # _ds_size # _map_flags # _cache_flags # _rv_flags  + L4_RPC_LEGACY_DISPATCH() + Dataspace_svr() + ~Dataspace_svr() + map() + map_hook() + phys() + take() + release() + copy() + clear() and 11 more # size() # map_flags() # rv_flags() # s_wmap() # page_size() # round_size() # cache_m() # size()</pre> |

## Public Member Functions

- `int map (l4_addr_t offset, l4_addr_t local_addr, unsigned long flags, l4_addr_t min_addr, l4_addr_t max_addr, L4::lpc::Snd_fpage &memory)`  
*Map a region of the dataspace.*
- `virtual int map_hook (l4_addr_t offs, unsigned long flags, l4_addr_t min, l4_addr_t max)`  
*A hook that is called as the first operation in each map request.*
- `virtual int phys (l4_addr_t offset, l4_addr_t &phys_addr, l4_size_t &phys_size) throw ()`  
*Return physical address for a virtual address.*
- `virtual void take () throw ()`  
*Take a reference to this dataspace.*
- `virtual unsigned long release () throw ()`  
*Release a reference to this dataspace.*
- `virtual unsigned long copy (l4_addr_t dst_offs, l4_umword_t src_id, l4_addr_t src_offs, unsigned long size) throw ()`  
*Copy from src dataspace to this destination dataspace.*
- `virtual long clear (unsigned long offs, unsigned long size) const throw ()`  
*Clear a region in the dataspace.*
- `virtual long allocate (l4_addr_t offset, l4_size_t size, unsigned access) throw ()`

*Allocate a region within a dataspace.*

- virtual unsigned long [page\\_shift](#) () const throw ()

*Define the size of the flexpage to map.*

- virtual bool [is\\_static](#) () const throw ()

*Return whether the dataspace is static.*

### 14.246.1 Detailed Description

[Dataspace](#) server class.

The default implementation of the interface provides a continuously mapped dataspace.

Definition at line 40 of file [dataspace\\_svr](#).

### 14.246.2 Member Function Documentation

#### 14.246.2.1 [allocate\(\)](#)

```
virtual long L4Re::Util::Dataspace_svr::allocate (
 l4_addr_t offset,
 l4_size_t size,
 unsigned access) throw () [inline], [virtual]
```

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

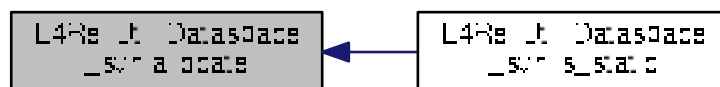
|    |         |
|----|---------|
| 0  | Success |
| <0 | Error   |

Definition at line 163 of file [dataspace\\_svr](#).

References [L4\\_ENODEV](#).

Referenced by [is\\_static\(\)](#).

Here is the caller graph for this function:



#### 14.246.2.2 clear()

```
virtual long L4Re::Util::Dataspace_svr::clear (
 unsigned long offs,
 unsigned long size) const throw () [virtual]
```

Clear a region in the dataspace.

##### Parameters

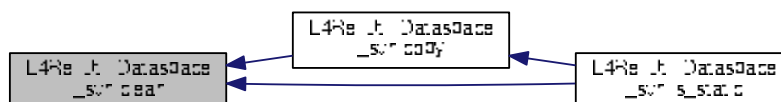
|             |                     |
|-------------|---------------------|
| <i>offs</i> | Start of the region |
| <i>size</i> | Size of the region  |

##### Return values

|    |         |
|----|---------|
| 0  | Success |
| <0 | Error   |

Referenced by [copy\(\)](#), and [is\\_static\(\)](#).

Here is the caller graph for this function:



### 14.246.2.3 copy()

```
virtual unsigned long L4Re::Util::Dataspace_svr::copy (
 l4_addr_t dst_offs,
 l4_umword_t src_id,
 l4_addr_t src_offs,
 unsigned long size) throw () [inline], [virtual]
```

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               |

#### Returns

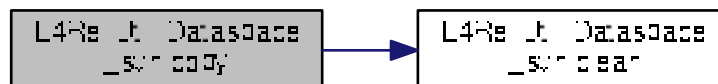
Number of bytes copied

Definition at line 137 of file [dataspace\\_svr](#).

References [clear\(\)](#), and [L4\\_ENODEV](#).

Referenced by [is\\_static\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



## 14.246.2.4 is\_static()

```
virtual bool L4Re::Util::Dataspace_svr::is_static () const throw () [inline], [virtual]
```

Return whether the dataspace is static.

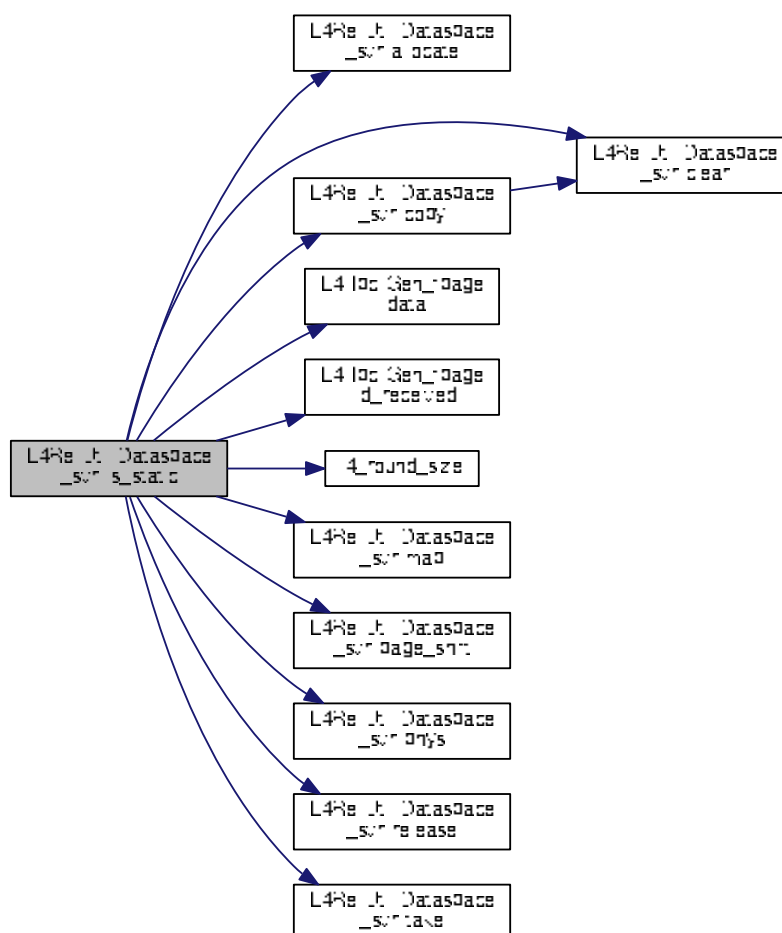
## Returns

True if dataspace is static

Definition at line 179 of file [dataspace\\_svr](#).

References [allocate\(\)](#), [clear\(\)](#), [copy\(\)](#), [L4::ipc::Gen\\_fpage< T >::data\(\)](#), [L4Re::Dataspace::Stats::flags](#), [L4::ipc::Gen\\_fpage< T >::id\\_received\(\)](#), [L4\\_CAP\\_FPAGE\\_W](#), [L4\\_EACCESS](#), [L4\\_EINVAL](#), [L4\\_EOK](#), [L4\\_EPERM](#), [L4::round\\_size\(\)](#), [map\(\)](#), [page\\_shift\(\)](#), [phys\(\)](#), [release\(\)](#), [L4Re::Dataspace::Stats::size](#), and [take\(\)](#).

Here is the call graph for this function:



## 14.246.2.5 map()

```
int L4Re::Util::Dataspace_svr::map (
 l4_addr_t offset,
 l4_addr_t local_addr,
 unsigned long flags,
 l4_addr_t min_addr,
 l4_addr_t max_addr,
 L4::Ipc::Snd_fpage & memory)
```

Map a region of the dataspace.

## Parameters

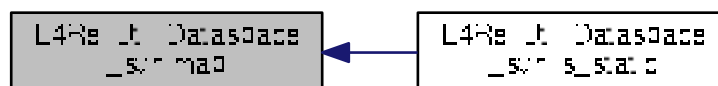
|     |                   |                                                             |
|-----|-------------------|-------------------------------------------------------------|
|     | <i>offset</i>     | Offset to start within data space                           |
|     | <i>local_addr</i> | Local address to map to.                                    |
|     | <i>flags</i>      | Map flags, see <a href="#">L4Re::Dataspace::Map_flags</a> . |
|     | <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   |

Referenced by [is\\_static\(\)](#).

Here is the caller graph for this function:



## 14.246.2.6 map\_hook()

```
virtual int L4Re::Util::Dataspace_svr::map_hook (
 l4_addr_t offs,
 unsigned long flags,
 l4_addr_t min,
 l4_addr_t max) [inline], [virtual]
```

A hook that is called as the first operation in each map request.



## Parameters

|              |                    |
|--------------|--------------------|
| <i>offs</i>  | Offs param to map  |
| <i>flags</i> | Flags param to map |
| <i>min</i>   | Min param to map   |
| <i>max</i>   | Max param to map   |

## Return values

|          |                                                            |
|----------|------------------------------------------------------------|
| $< 0$    | Error and the map request will be aborted with that error. |
| $\geq 0$ | Success                                                    |

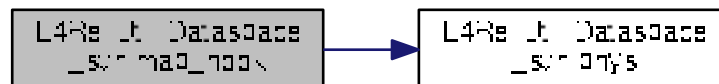
## See also

[map](#)

Definition at line 90 of file [dataspace\\_svr](#).

References [phys\(\)](#).

Here is the call graph for this function:

14.246.2.7 `page_shift()`

```
virtual unsigned long L4Re::Util::Dataspace_svr::page_shift () const throw () [inline],
[virtual]
```

Define the size of the flexpage to map.

**Returns**

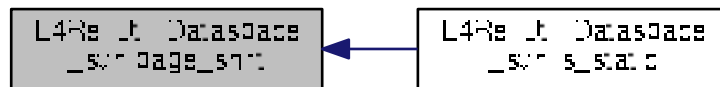
flexpage size

Definition at line 171 of file [dataspace\\_svr](#).

References [L4\\_LOG2\\_PAGESIZE](#).

Referenced by [is\\_static\(\)](#).

Here is the caller graph for this function:

**14.246.2.8 phys()**

```
virtual int L4Re::Util::Dataspace_svr::phys (
 l4_addr_t offset,
 l4_addr_t & phys_addr,
 l4_size_t & phys_size) throw () [virtual]
```

Return physical address for a virtual address.

**Parameters**

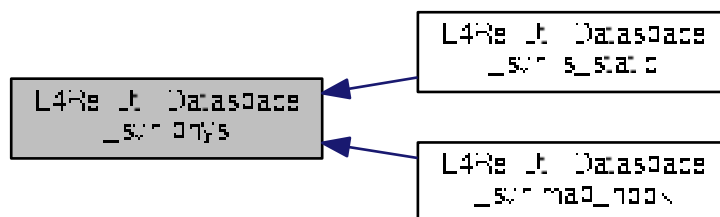
|     |                  |                                    |
|-----|------------------|------------------------------------|
|     | <i>offset</i>    | Offset into the dataspace          |
| out | <i>phys_addr</i> | Physical address                   |
| out | <i>phys_size</i> | Size of continious physical region |

**Return values**

|    |         |
|----|---------|
| 0  | Success |
| <0 | Error   |

Referenced by [is\\_static\(\)](#), and [map\\_hook\(\)](#).

Here is the caller graph for this function:



#### 14.246.2.9 release()

```
virtual unsigned long L4Re::Util::Dataspace_svr::release () throw () [inline], [virtual]
```

Release a reference to this dataspace.

##### Returns

Number of references to the dataspace

Default does nothing and returns always zero.

Definition at line 124 of file [dataspace\\_svr](#).

Referenced by [is\\_static\(\)](#).

Here is the caller graph for this function:



## 14.246.2.10 take()

```
virtual void L4Re::Util::Dataspace_svr::take () throw () [inline], [virtual]
```

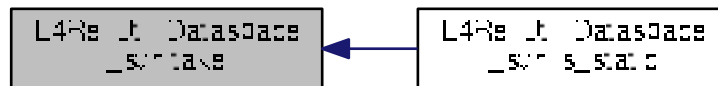
Take a reference to this dataspace.

Default does nothing.

Definition at line 114 of file [dataspace\\_svr](#).

Referenced by [is\\_static\(\)](#).

Here is the caller graph for this function:



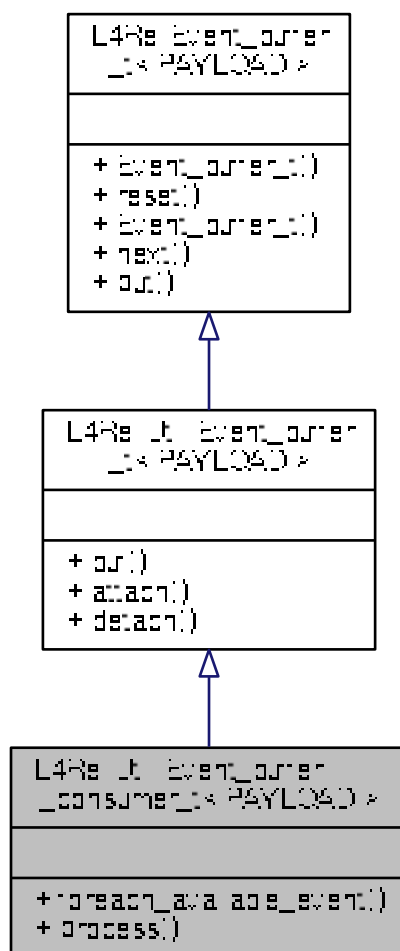
The documentation for this class was generated from the following file:

- [l4/re/util/dataspace\\_svr](#)

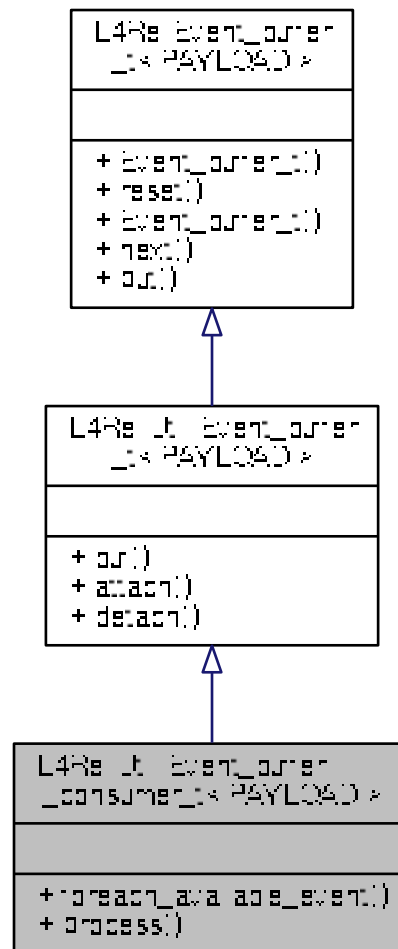
## 14.247 L4Re::Util::Event\_buffer\_consumer\_t< PAYLOAD > Class Template Reference

An event buffer consumer.

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.*

### 14.247.1 Detailed Description

```
template<typename PAYLOAD>
class L4Re::Util::Event_buffer_consumer_t< PAYLOAD >
```

An event buffer consumer.

Definition at line 92 of file [event\\_buffer](#).

## 14.247.2 Member Function Documentation

### 14.247.2.1 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 103 of file [event\\_buffer](#).

References [L4Re::Event\\_buffer\\_t< PAYLOAD >::Event::free\(\)](#).

Here is the call graph for this function:



### 14.247.2.2 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>    | <a href="#">Event</a> 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.   |

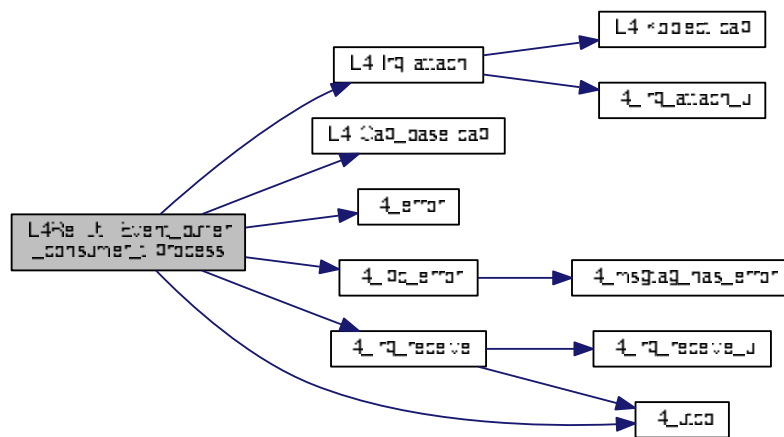
## Note

This function never returns.

Definition at line 124 of file [event\\_buffer](#).

References [L4::Irq::attach\(\)](#), [L4::Cap\\_base::cap\(\)](#), [l4\\_error\(\)](#), [l4\\_ipc\\_error\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_irq\\_receive\(\)](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

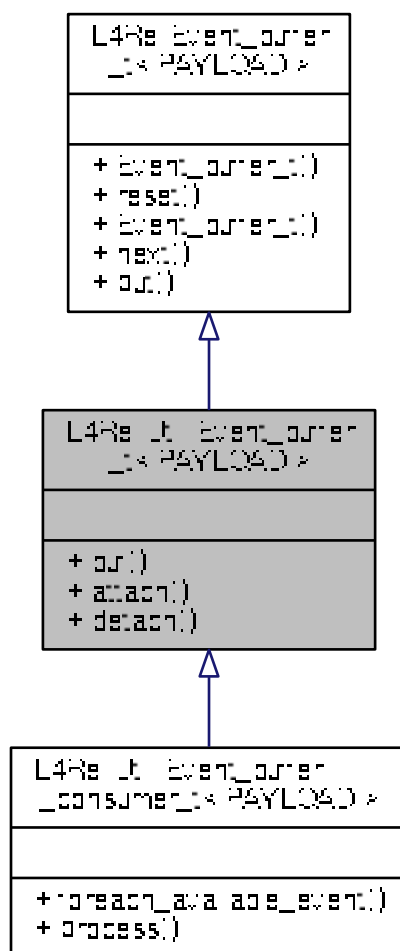
- `l4/re/util/event_buffer`

## 14.248 L4Re::Util::Event\_buffer\_t< PAYLOAD > Class Template Reference

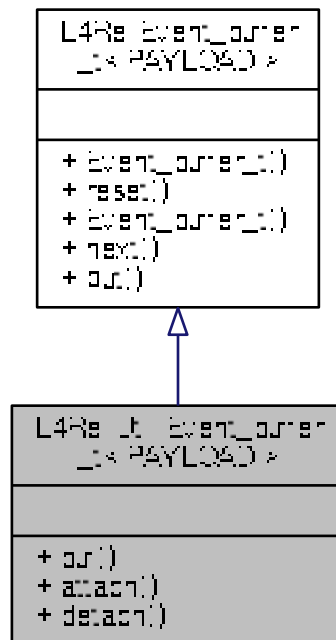
Event\_buffer utility class.



Inheritance diagram for L4Re::Util::Event\_buffer\_t< PAYLOAD >:



Collaboration diagram for L4Re::Util::Event\_buffer\_t< PAYLOAD >:



## Public Member Functions

- `void * buf () const throw ()`  
Return the buffer.
- `long attach (L4::Cap< L4Re::Dataspace > ds, L4::Cap< L4Re::Rm > rm) throw ()`  
Attach event buffer from address space.
- `long detach (L4::Cap< L4Re::Rm > rm) throw ()`  
Detach event buffer from address space.

### 14.248.1 Detailed Description

```
template<typename PAYLOAD>
class L4Re::Util::Event_buffer_t< PAYLOAD >
```

Event\_buffer utility class.

Definition at line 36 of file [event\\_buffer](#).

### 14.248.2 Member Function Documentation

## 14.248.2.1 attach()

```
template<typename PAYLOAD >
long L4Re::Util::Event_buffer_t< PAYLOAD >::attach (
 L4::Cap< L4Re::Dataspace > ds,
 L4::Cap< L4Re::Rm > rm) throw () [inline]
```

Attach event buffer from address space.

## Parameters

|           |                                                |
|-----------|------------------------------------------------|
| <i>ds</i> | <a href="#">Dataspace</a> of the event buffer. |
| <i>rm</i> | Region manager to attach buffer to.            |

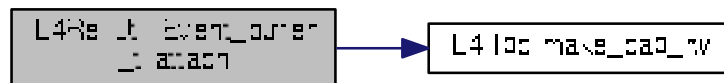
## Returns

0 on success, negative error code otherwise.

Definition at line 56 of file [event\\_buffer](#).

References [L4::ipc::make\\_cap\\_rw\(\)](#), and [L4Re::Rm::Search\\_addr](#).

Here is the call graph for this function:



## 14.248.2.2 buf()

```
template<typename PAYLOAD >
void* L4Re::Util::Event_buffer_t< PAYLOAD >::buf () const throw () [inline]
```

Return the buffer.

## Returns

Pointer to the event buffer.

Definition at line 46 of file [event\\_buffer](#).

## 14.248.2.3 detach()

```
template<typename PAYLOAD >
long L4Re::Util::Event_buffer_t< PAYLOAD >::detach (
 L4::Cap< L4Re::Rm > rm) throw () [inline]
```

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 77 of file [event\\_buffer](#).

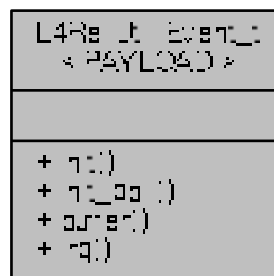
The documentation for this class was generated from the following file:

- [l4/re/util/event\\_buffer](#)

## 14.249 L4Re::Util::Event\_t< PAYLOAD > Class Template Reference

Convenience wrapper for getting access to an event object.

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::Cap_alloc::get_cap_alloc(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::Cap_alloc::get_cap_alloc(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.*

### 14.249.1 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 [41](#) of file [event](#).

### 14.249.2 Member Enumeration Documentation

#### 14.249.2.1 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 [47](#) of file [event](#).

### 14.249.3 Member Function Documentation

#### 14.249.3.1 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 [156](#) of file [event](#).

## 14.249.3.2 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::Cap_alloc::get_cap_alloc(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 <a href="#">L4::Triggerable</a> . |
|-----------------|---------------------------------------------------------------------------------------------------------------------------|

## Parameters

|              |                                  |
|--------------|----------------------------------|
| <i>event</i> | Capability to event.             |
| <i>env</i>   | Pointer to L4Re-Environment      |
| <i>ca</i>    | Pointer to capability allocator. |

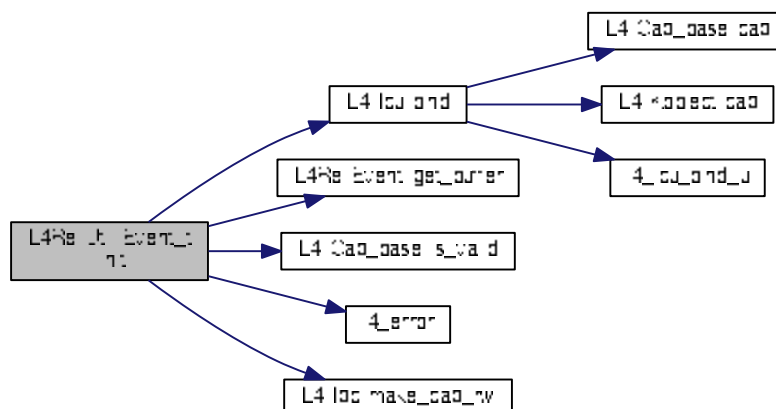
## Return values

|            |                                              |
|------------|----------------------------------------------|
| 0          | Success                                      |
| -L4_ENOMEM | No memory to allocate required capabilities. |
| <0         | Other IPC errors.                            |

Definition at line 68 of file [event](#).

References [L4::lcu::bind\(\)](#), [L4Re::Event::get\\_buffer\(\)](#), [L4::Cap\\_base::is\\_valid\(\)](#), [L4\\_ENOMEM](#), [l4\\_error\(\)](#), [L4::ipc::make\\_cap\\_rw\(\)](#), and [L4Re::Rm::Search\\_addr](#).

Here is the call graph for this function:



## 14.249.3.3 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::Cap_alloc::get_cap_alloc(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. |

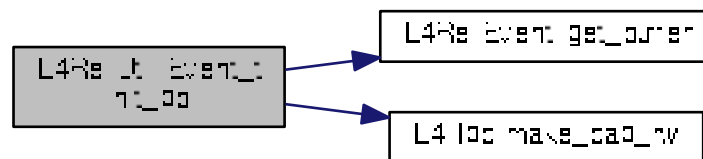
## Return values

|            |                                              |
|------------|----------------------------------------------|
| 0          | Success                                      |
| -L4_ENOMEM | No memory to allocate required capabilities. |
| <0         | Other IPC errors.                            |

Definition at line 121 of file [event](#).

References [L4Re::Event::get\\_buffer\(\)](#), [L4\\_ENOMEM](#), [L4::ipc::make\\_cap\\_rw\(\)](#), and [L4Re::Rm::Search\\_addr](#).

Here is the call graph for this function:



## 14.249.3.4 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 [163](#) of file [event](#).

The documentation for this class was generated from the following file:

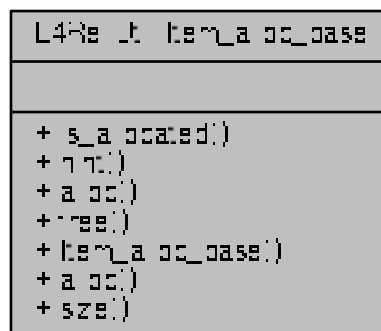
- [l4/re/util/event](#)

## 14.250 L4Re::Util::Item\_alloc\_base Class Reference

Item allocator.

Inherited by [L4Re::Util::Item\\_alloc< Bits >](#).

Collaboration diagram for [L4Re::Util::Item\\_alloc\\_base](#):



### 14.250.1 Detailed Description

Item allocator.

Definition at line [38](#) of file [item\\_alloc](#).

The documentation for this class was generated from the following file:

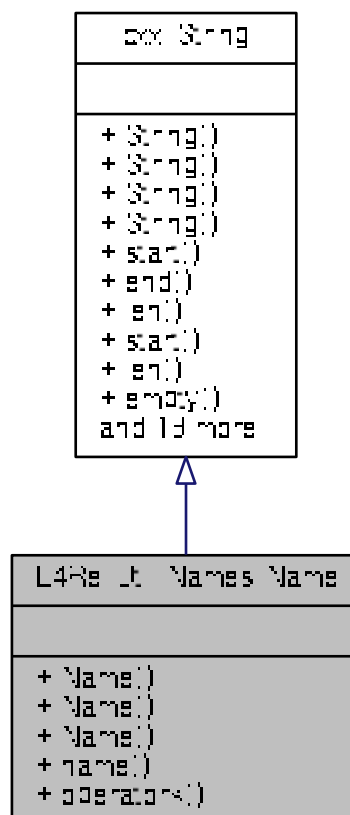
- [l4/re/util/item\\_alloc](#)



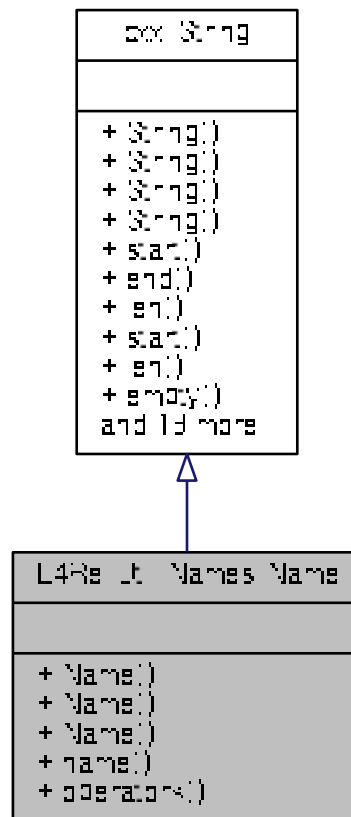
## 14.251 L4Re::Util::Names::Name Class Reference

[Name](#) class.

Inheritance diagram for L4Re::Util::Names::Name:



Collaboration diagram for L4Re::Util::Names::Name:



### 14.251.1 Detailed Description

[Name](#) class.

Definition at line 42 of file [name\\_space\\_svr](#).

The documentation for this class was generated from the following file:

- [l4/re/util/name\\_space\\_svr](#)

## 14.252 L4Re::Util::Names::Name\_space Class Reference

Abstract server-side implementation of the L4::Namespace interface.

Collaboration diagram for L4Re::Util::Names::Name\_space:

| L4Re::Util::Names::Name_space                                                                                                                                                                                                                                                                 |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>#_dog #_err</pre>                                                                                                                                                                                                                                                                        |
| <pre>+begin() +end() +Name_space() +~Name_space() +find() +remove() +find_iter() +insert() +dump() +cc_query() +cc_register_cc() +cc_unregister_cc() #alloc_dynamic_entry() #free_dynamic_entry() #get_epiface() #copy_receive_cap() #free_capability() #free_epiface() #insert_entry()</pre> |

### 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 int [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 int [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](#)().*

#### 14.252.1 Detailed Description

Abstract server-side implementation of the L4::Namespace interface.

Definition at line 184 of file [name\\_space\\_svr](#).

## 14.252.2 Member Function Documentation

### 14.252.2.1 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>     | <a href="#">Name</a> of the entry, must be copied. |
| <i>flags</i> | Entry flags, see <a href="#">Obj::Flags</a> .      |

#### Returns

A pointer to the newly allocated entry or NULL on error.

### 14.252.2.2 copy\_receive\_cap()

```
virtual int 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 <a href="#">free_capability()</a> is called. |
|-----|------------|--------------------------------------------------------------------------------------------------------------------------------|

### 14.252.2.3 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. |
|----|------------|---------------------|

## 14.252.2.4 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. |
|----------|----------------|

## 14.252.2.5 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.

## 14.252.2.6 get\_epiface()

```
virtual int 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 truly local capability was supplied.                             |
| out | <i>lo</i>       | Pointer to epiface responsible for the capability.                          |

## Returns

L4\_OK if a valid interface could be found or an error message otherwise.

The caller 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.

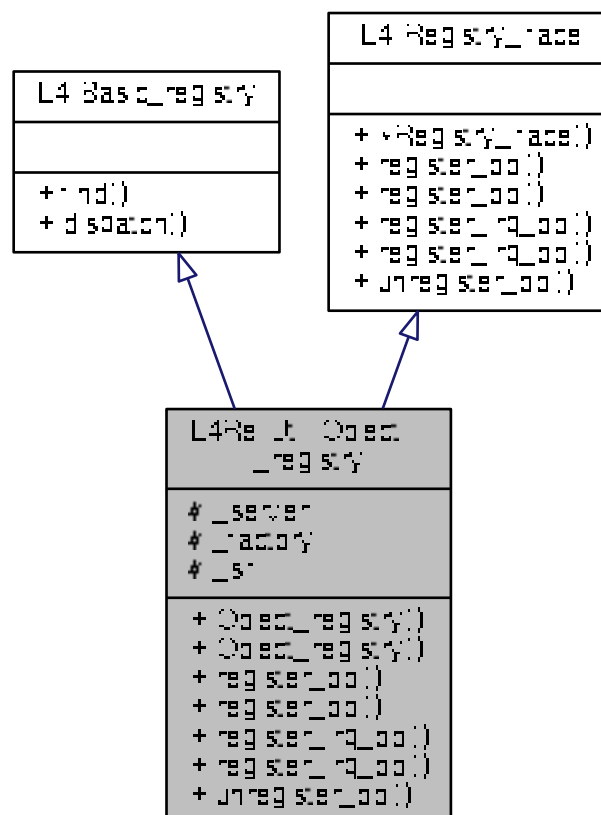
The documentation for this class was generated from the following file:

- `l4/re/util/name_space_svr`

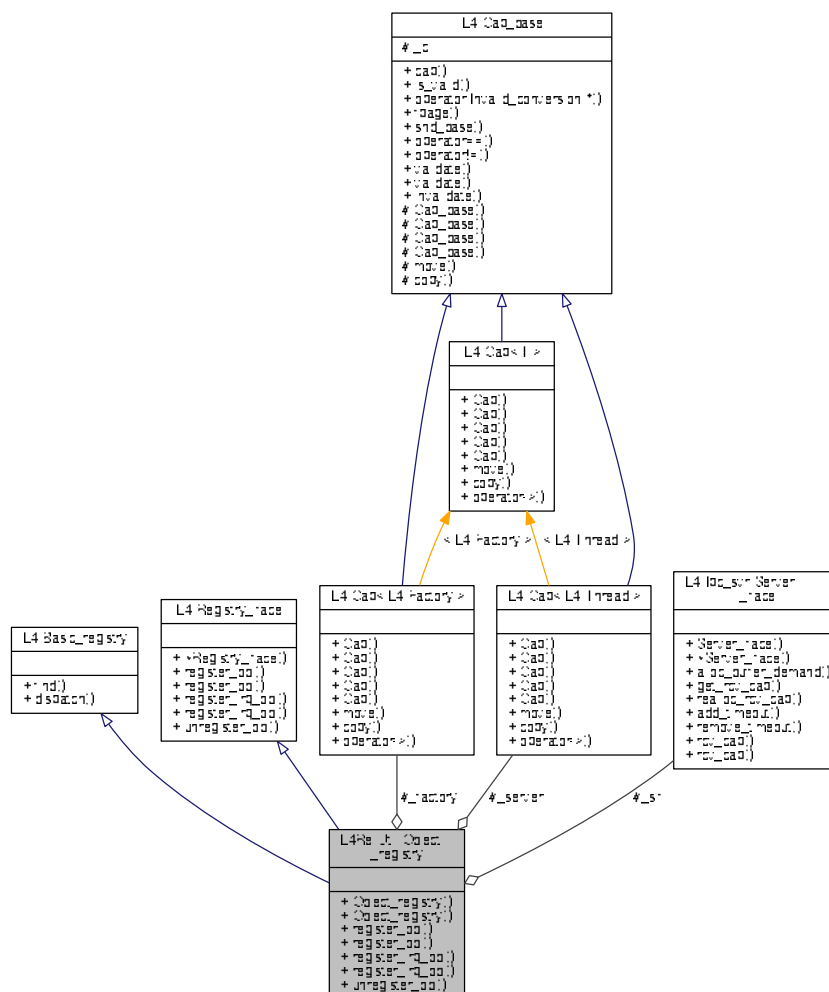
## 14.253 L4Re::Util::Object\_registry Class Reference

A registry that manages server objects and their attached IPC gates for a single server loop for a specific thread.

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 IPC gate.*
- `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::Irq > register_irq_obj (L4::Epiface *o, L4::Cap< L4::Irq > const &irq)` 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

### 14.253.1 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 51 of file [object\\_registry](#).

### 14.253.2 Constructor & Destructor Documentation

#### 14.253.2.1 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 77 of file [object\\_registry](#).

#### 14.253.2.2 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

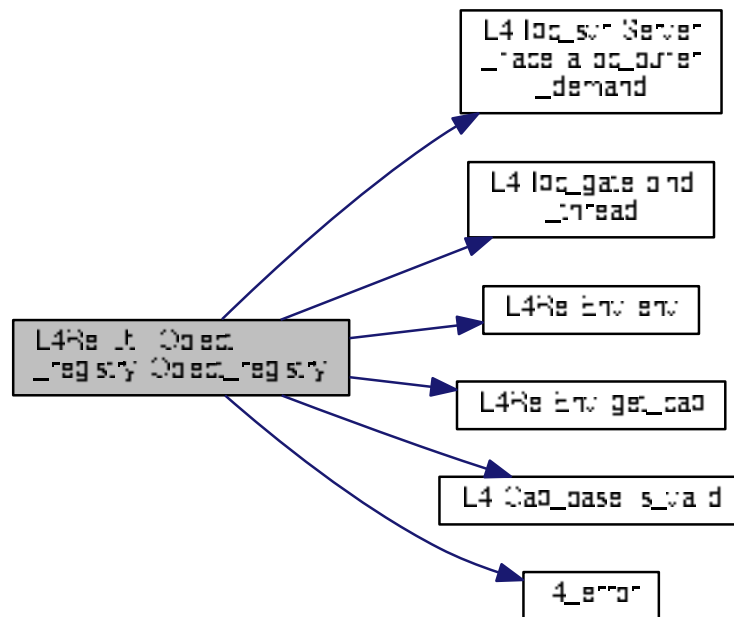
|                |                                                                   |
|----------------|-------------------------------------------------------------------|
| <i>sif</i>     | Server loop interface.                                            |
| <i>server</i>  | Capability to the thread that executes the server objects.        |
| <i>factory</i> | Capability to a factory object capable of creating new IPC gates. |



Definition at line 91 of file [object\\_registry](#).

References [L4::lpc\\_svr::Server\\_iface::alloc\\_buffer\\_demand\(\)](#), [L4::lpc\\_gate::bind\\_thread\(\)](#), [L4Re::Env::env\(\)](#), [L4Re::Env::get\\_cap\(\)](#), [L4::Cap\\_base::is\\_valid\(\)](#), and [l4\\_error\(\)](#).

Here is the call graph for this function:



### 14.253.3 Member Function Documentation

#### 14.253.3.1 register\_irq\_obj() [1/2]

```
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. |
|----------|----------------------------------|

##### Returns

The capability the server object was registered with.

The IRQ will be newly allocated using the registry's factory object.

Implements [L4::Registry\\_iface](#).

Definition at line 249 of file [object\\_registry](#).

#### 14.253.3.2 `register_irq_obj()` [2/2]

```
L4::Cap<L4::Irq> L4Re::Util::Object_registry::register_irq_obj (
 L4::Epiface * o,
 L4::Cap< L4::Irq > const & irq) [inline], [override], [virtual]
```

Register a handler for an already existing interrupt.

##### Parameters

|            |                                                                       |
|------------|-----------------------------------------------------------------------|
| <i>o</i>   | Server object that handles the interrupts.                            |
| <i>irq</i> | Capability to an IRQ object, may be a hardware or software interrupt. |

##### Returns

The capability the server object was registered with.

Implements [L4::Registry\\_iface](#).

Definition at line 265 of file [object\\_registry](#).

#### 14.253.3.3 `register_obj()` [1/2]

```
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 IPC gate.

##### Parameters

|                |                                          |
|----------------|------------------------------------------|
| <i>o</i>       | Server object that handles IPC requests. |
| <i>service</i> | Name of a pre-allocated IPC gate.        |

##### Returns

The capability the server object was registered with.

Implements [L4::Registry\\_iface](#).

Definition at line 221 of file [object\\_registry](#).

#### 14.253.3.4 register\_obj() [2/2]

```
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. |
|----------|------------------------------------------|

##### Returns

The capability the server object was registered with.

The IPC gate will be allocated using the registry's factory.

Implements [L4::Registry\\_iface](#).

Definition at line 235 of file [object\\_registry](#).

#### 14.253.3.5 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 282 of file [object\\_registry](#).

References [L4::Thread::Modify\\_senders::add\(\)](#), [L4Re::Util::cap\\_alloc](#), [L4Re::Util::Counting\\_cap\\_alloc< COUNTERTYPE >::free\(\)](#), [L4\\_FP\\_ALL\\_SPACES](#), and [L4::Thread::modify\\_senders\(\)](#).



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 231 of file [cap\\_alloc](#).

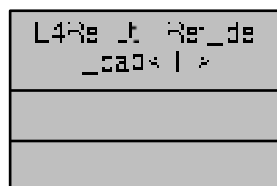
The documentation for this struct was generated from the following file:

- [l4/re/util/cap\\_alloc](#)

## 14.255 L4Re::Util::Ref\_del\_cap< T > Struct Template Reference

Automatic capability that implements automatic free and unmap+delete of the capability selector.

Collaboration diagram for L4Re::Util::Ref\_del\_cap< T >:



### 14.255.1 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 272 of file [cap\\_alloc](#).

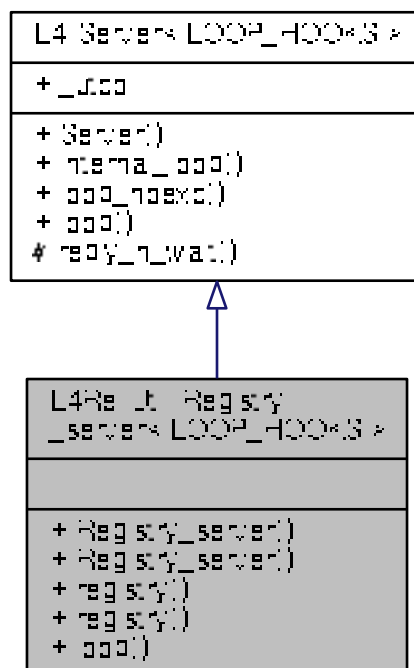
The documentation for this struct was generated from the following file:

- [l4/re/util/cap\\_alloc](#)

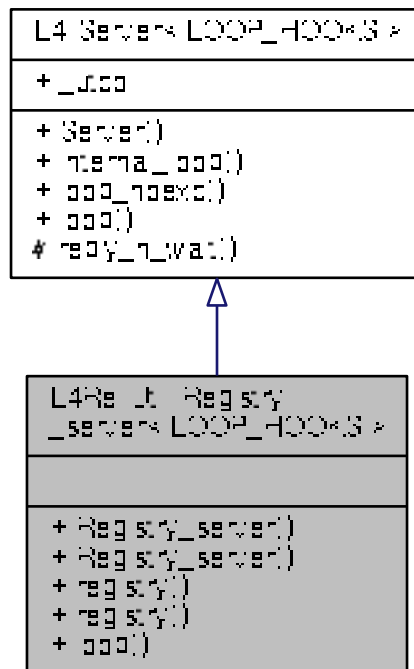
## 14.256 L4Re::Util::Registry\_server< LOOP\_HOOKS > Class Template Reference

A server loop object which has a [Object\\_registry](#) included.

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 \*utcb, 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 ()  
Start the server loop.

## Additional Inherited Members

### 14.256.1 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/server.cc](#), [examples/libs/l4re/c++/shared\\_ds/ds\\_srv.cc](#), and [examples/libs/l4re/streammap/server.cc](#).

Definition at line 312 of file [object\\_registry](#).

## 14.256.2 Constructor & Destructor Documentation

### 14.256.2.1 Registry\_server() [1/2]

```
template<typename LOOP_HOOKS = L4::Ipc_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 324 of file [object\\_registry](#).

### 14.256.2.2 Registry\_server() [2/2]

```
template<typename LOOP_HOOKS = L4::Ipc_svr::Default_loop_hooks>
L4Re::Util::Registry_server< LOOP_HOOKS >::Registry_server (
 l4_utcb_t * utcb,
 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>utcb</i>    | The UTCB of the thread running the server loop.            |
| <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 334 of file [object\\_registry](#).

## 14.256.3 Member Function Documentation

**14.256.3.1 registry()** [1/2]

```
template<typename LOOP_HOOKS = L4::Ipc_svr::Default_loop_hooks>
Object_registry const* L4Re::Util::Registry_server< LOOP_HOOKS >::registry () const [inline]
```

Return registry of this server loop.

Definition at line 340 of file [object\\_registry](#).

**14.256.3.2 registry()** [2/2]

```
template<typename LOOP_HOOKS = L4::Ipc_svr::Default_loop_hooks>
Object_registry* L4Re::Util::Registry_server< LOOP_HOOKS >::registry () [inline]
```

Return registry of this server loop.

Definition at line 342 of file [object\\_registry](#).

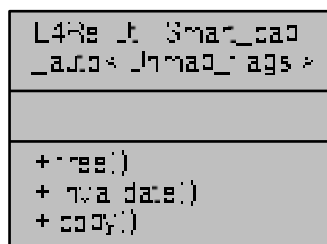
The documentation for this class was generated from the following file:

- [l4/re/util/object\\_registry](#)

**14.257 L4Re::Util::Smart\_cap\_auto< Unmap\_flags > Class Template Reference**

Helper for [Auto\\_cap](#) and [Auto\\_del\\_cap](#).

Collaboration diagram for L4Re::Util::Smart\_cap\_auto< Unmap\_flags >:

**Static Public Member Functions**

- static void [free](#) (L4::Cap\_base &c)  
*Free operation for L4::Smart\_cap.*
- static void [invalidate](#) (L4::Cap\_base &c)  
*Invalidate operation for L4::Smart\_cap.*
- static L4::Cap\_base [copy](#) (L4::Cap\_base const &src)  
*Copy operation for L4::Smart\_cap.*

### 14.257.1 Detailed Description

```
template<unsigned long Unmap_flags = L4_FP_ALL_SPACES>
class L4Re::Util::Smart_cap_auto< Unmap_flags >
```

Helper for [Auto\\_cap](#) and [Auto\\_del\\_cap](#).

Definition at line 59 of file [cap\\_alloc](#).

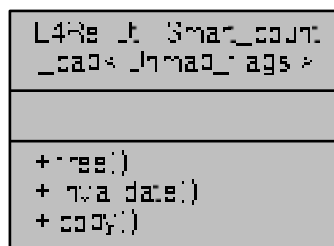
The documentation for this class was generated from the following file:

- [l4/re/util/cap\\_alloc](#)

## 14.258 L4Re::Util::Smart\_count\_cap< Unmap\_flags > Class Template Reference

Helper for [Ref\\_cap](#) and [Ref\\_del\\_cap](#).

Collaboration diagram for L4Re::Util::Smart\_count\_cap< Unmap\_flags >:



### Static Public Member Functions

- static void [free](#) ([L4::Cap\\_base](#) &c) throw ()  
*Free operation for [L4::Smart\\_cap](#) (decrement ref count and delete if 0).*
- static void [invalidate](#) ([L4::Cap\\_base](#) &c) throw ()  
*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).*

### 14.258.1 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 99 of file [cap\\_alloc](#).

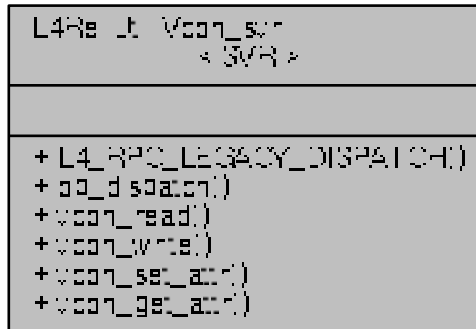
The documentation for this class was generated from the following file:

- [l4/re/util/cap\\_alloc](#)

## 14.259 L4Re::Util::Vcon\_svr< SVR > Class Template Reference

[Console](#) server template class.

Collaboration diagram for L4Re::Util::Vcon\_svr< SVR >:



### 14.259.1 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.

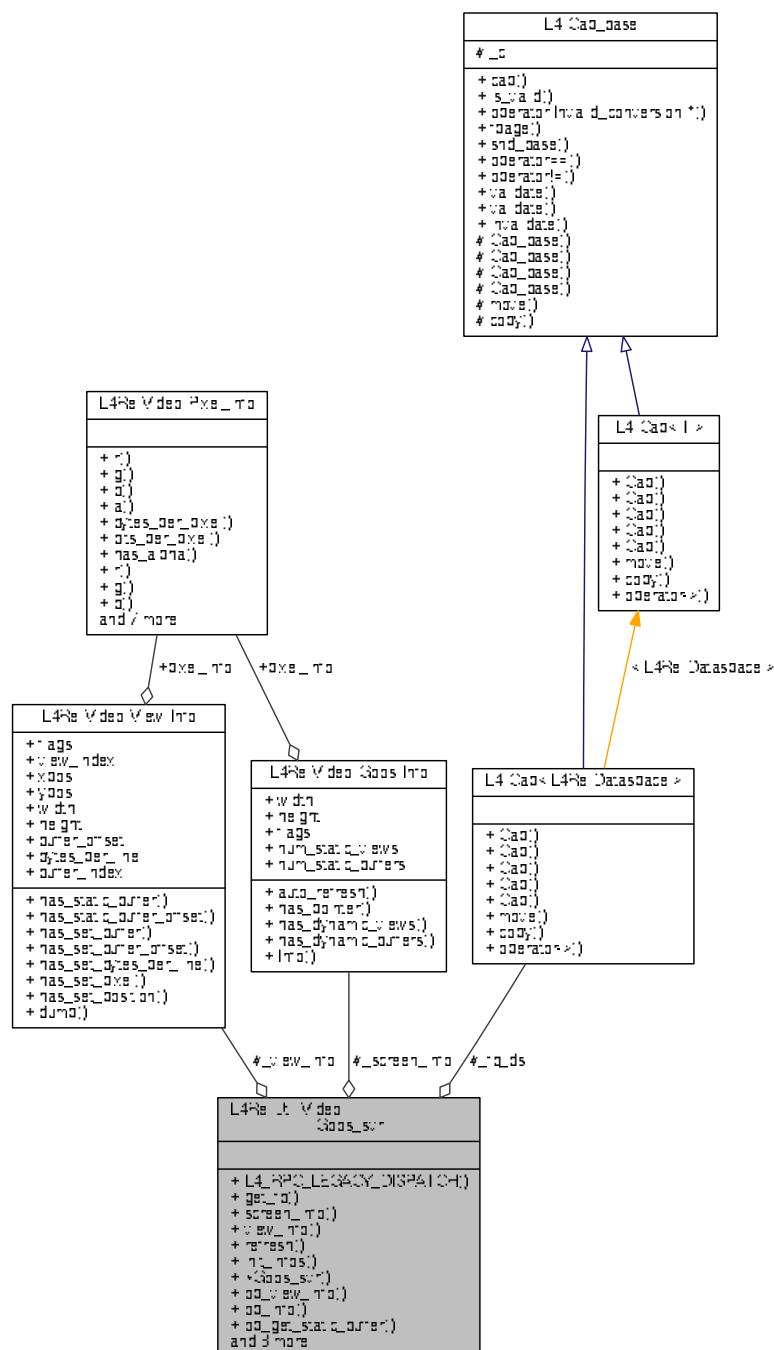
Definition at line 46 of file [vcon\\_svr](#).

The documentation for this class was generated from the following file:

- `l4/re/util/vcon_svr`

Goos server class.

Collaboration diagram for L4Re::Util::Video::Goos svr:



- `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 [~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.*

### 14.260.1 Detailed Description

Goos server class.

Definition at line 36 of file [goos\\_svr](#).

### 14.260.2 Member Function Documentation

#### 14.260.2.1 [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 53 of file [goos\\_svr](#).

References [\\_fb\\_ds](#).

## 14.260.2.2 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 89 of file [goos\\_svr](#).

References [L4Re::Video::View::Info::buffer\\_index](#), [L4Re::Video::View::Info::flags](#), [L4Re::Video::View::Info::height](#), [L4Re::Video::Goos::Info::height](#), [L4Re::Video::View::Info::pixel\\_info](#), [L4Re::Video::Goos::Info::pixel\\_info](#), [L4Re::Video::View::Info::view\\_index](#), [L4Re::Video::View::Info::width](#), [L4Re::Video::Goos::Info::width](#), [L4Re::Video::View::Info::xpos](#), and [L4Re::Video::View::Info::ypos](#).

## 14.260.2.3 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

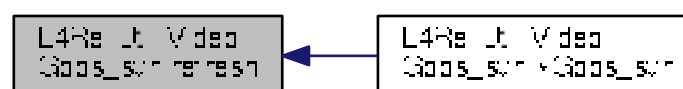
0 on success, negative error code otherwise

Definition at line 77 of file [goos\\_svr](#).

References [L4\\_ENOSYS](#).

Referenced by [~Goos\\_svr\(\)](#).

Here is the caller graph for this function:



#### 14.260.2.4 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 59 of file [goos\\_svr](#).

References [\\_screen\\_info](#).

#### 14.260.2.5 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 65 of file [goos\\_svr](#).

References [\\_view\\_info](#).

The documentation for this class was generated from the following file:

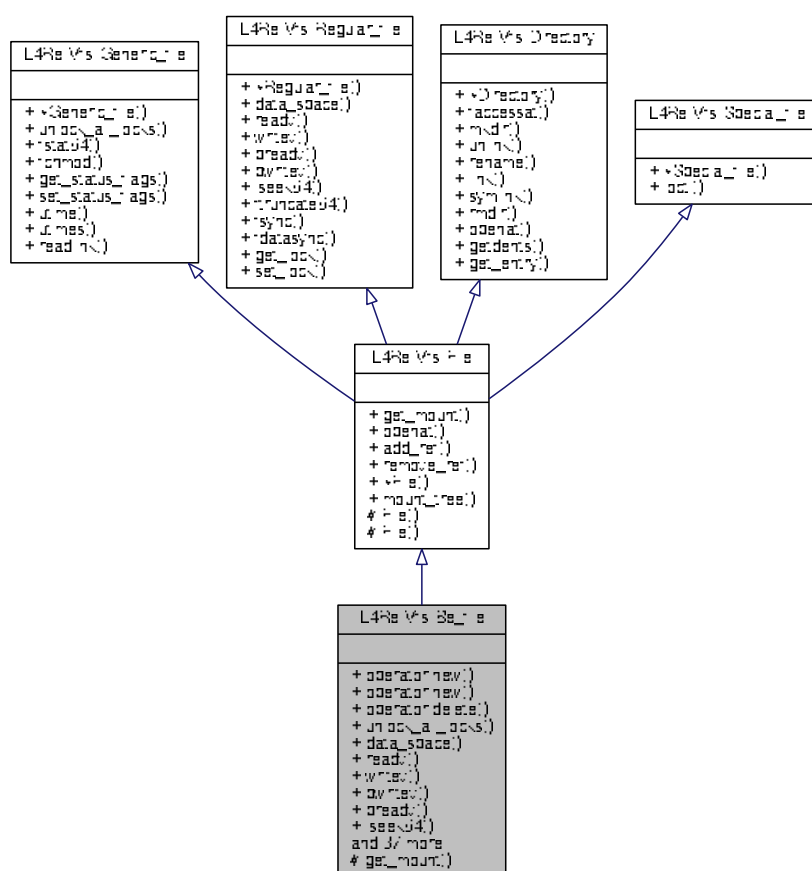
- [l4/re/util/video/goos\\_svr](#)



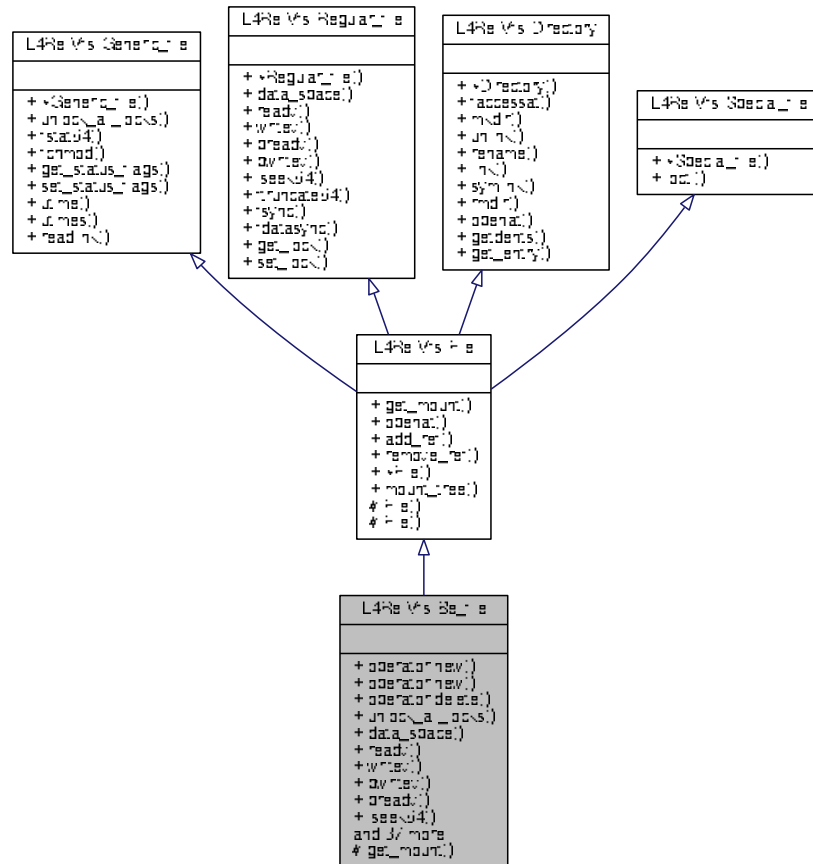
## 14.261 L4Re::Vfs::Be\_file Class Reference

Boiler plate class for implementing an open file for [L4Re::Vfs](#).

Inheritance diagram for L4Re::Vfs::Be\_file:



Collaboration diagram for L4Re::Vfs::Be\_file:



## Public Member Functions

- `int unlock_all_locks ()` throw ()  
Unlock all locks on the file.
- `L4::Cap< L4Re::Dataspace > data_space ()` const throw ()  
Get an `L4Re::Dataspace` object for the file.
- `ssize_t read (const struct iovec *, int)` throw ()  
Default backend for POSIX read and readv functions.
- `ssize_t write (const struct iovec *, int)` throw ()  
Default backend for POSIX write and writev functions.
- `ssize_t pwrite (const struct iovec *, int, off64_t)` throw ()  
Default backend for POSIX pwrite and pwritev functions.
- `ssize_t preadv (const struct iovec *, int, off64_t)` throw ()  
Default backend for POSIX pread and preadv functions.
- `off64_t lseek64 (off64_t, int)` throw ()  
Default backend for POSIX seek and lseek functions.
- `int ftruncate64 (off64_t)` throw ()  
Default backend for the POSIX truncate, ftruncate and similar functions.
- `int fsync ()` const throw ()

- *Default backend for POSIX fsync.*
- `int fdatsync () const throw ()`
- *Default backend for POSIX fdatsync.*
- `int ioctl (unsigned long, va_list) throw ()`
- *Default backend for POSIX ioctl.*
- `int fstat64 (struct stat64 *) const throw ()`
- *Get status information for the file.*
- `int fchmod (mode_t) throw ()`
- *Default backend for POSIX chmod and fchmod.*
- `int get_status_flags () const throw ()`
- *Default backend for POSIX fcntl subfunctions.*
- `int set_status_flags (long) throw ()`
- *Default backend for POSIX fcntl subfunctions.*
- `int get_lock (struct flock64 *) throw ()`
- *Default backend for POSIX fcntl subfunctions.*
- `int set_lock (struct flock64 *, bool) throw ()`
- *Default backend for POSIX fcntl subfunctions.*
- `int faccessat (const char *, int, int) throw ()`
- *Default backend for POSIX access and faccessat functions.*
- `int utime (const struct utimbuf *) throw ()`
- *Default backend for POSIX utime.*
- `int utimes (const struct timeval [2]) throw ()`
- *Default backend for POSIX utimes.*
- `int mkdir (const char *, mode_t) throw ()`
- *Default backend for POSIX mkdir and mkdirat.*
- `int unlink (const char *) throw ()`
- *Default backend for POSIX unlink, unlinkat.*
- `int rename (const char *, const char *) throw ()`
- *Default backend for POSIX rename, renameat.*
- `int link (const char *, const char *) throw ()`
- *Default backend for POSIX link, linkat.*
- `int symlink (const char *, const char *) throw ()`
- *Default backend for POSIX symlink, symlinkat.*
- `int rmdir (const char *) throw ()`
- *Default backend for POSIX rmdir, rmdirat.*
- `ssize_t readlink (char *, size_t)`
- *Default backend for POSIX readlink, readlinkat.*

### 14.261.1 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 39 of file [backend](#).

## 14.261.2 Member Function Documentation

### 14.261.2.1 data\_space()

```
L4::Cap<L4Re::Dataspace> L4Re::Vfs::Be_file::data_space () const throw () [inline], [virtual]
```

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 functions returns an invalid capability.

#### Returns

A capability to an [L4Re::Dataspace](#), that represents the files contents in an [L4Re](#) way.

Implements [L4Re::Vfs::Regular\\_file](#).

Definition at line 56 of file [backend](#).

### 14.261.2.2 fstat64()

```
int L4Re::Vfs::Be_file::fstat64 (
 struct stat64 * buf) const throw () [inline], [virtual]
```

Get status information for the file.

This is the backend for POSIX fstat, stat, fstat64 and friends.

#### Parameters

|     |            |                                                    |
|-----|------------|----------------------------------------------------|
| out | <i>buf</i> | This buffer is filled with the status information. |
|-----|------------|----------------------------------------------------|

#### Returns

0 on success, or <0 on error.

Implements [L4Re::Vfs::Generic\\_file](#).

Definition at line 95 of file [backend](#).

## 14.261.2.3 unlock\_all\_locks()

```
int L4Re::Vfs::Be_file::unlock_all_locks () throw () [inline], [virtual]
```

Unlock all locks on the file.

## Note

All locks means all locks independent by which file the locks were taken.

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 52 of file [backend](#).

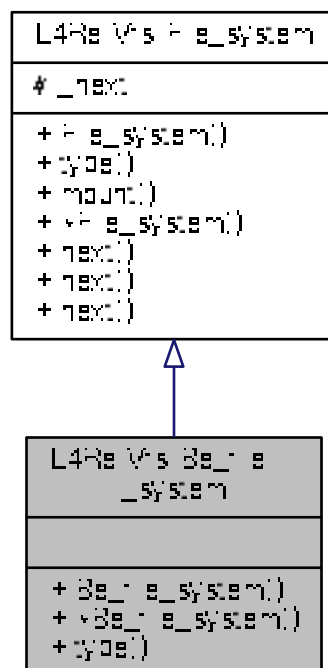
The documentation for this class was generated from the following file:

- l4/l4re\_vfs/backend

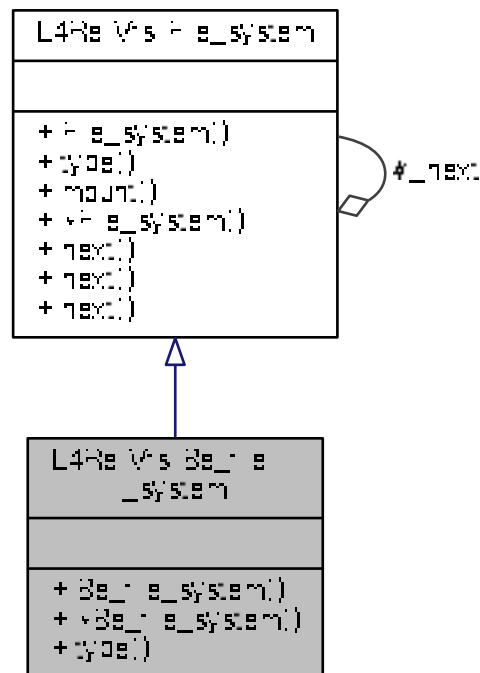
## 14.262 L4Re::Vfs::Be\_file\_system Class Reference

Boilerplate class for implementing a [L4Re::Vfs::File\\_system](#).

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) throw ()  
*Create a file-system object for the given fstype.*
- [~Be\\_file\\_system](#) () throw ()  
*Destroy a file-system object.*
- char const \* [type](#) () const throw ()  
*Return the file-system type.*

### 14.262.1 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 299 of file [backend](#).

## 14.262.2 Constructor & Destructor Documentation

### 14.262.2.1 Be\_file\_system()

```
L4Re::Vfs::Be_file_system::Be_file_system (
 char const * fstype) throw () [inline], [explicit]
```

Create a file-system object for the given *fstype*.

#### Parameters

|               |                                                    |
|---------------|----------------------------------------------------|
| <i>fstype</i> | The type that <a href="#">type()</a> shall return. |
|---------------|----------------------------------------------------|

This constructor takes care of registering the file system in the registry of L4Re::Vfs::vfs\_ops.

Definition at line 313 of file [backend](#).

### 14.262.2.2 ~Be\_file\_system()

```
L4Re::Vfs::Be_file_system::~~Be_file_system () throw () [inline]
```

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 325 of file [backend](#).

## 14.262.3 Member Function Documentation

### 14.262.3.1 type()

```
char const* L4Re::Vfs::Be_file_system::type () const throw () [inline], [virtual]
```

Return the file-system type.

Returns the file-system type given as *fstype* in the constructor.

Implements [L4Re::Vfs::File\\_system](#).

Definition at line 335 of file [backend](#).

The documentation for this class was generated from the following file:

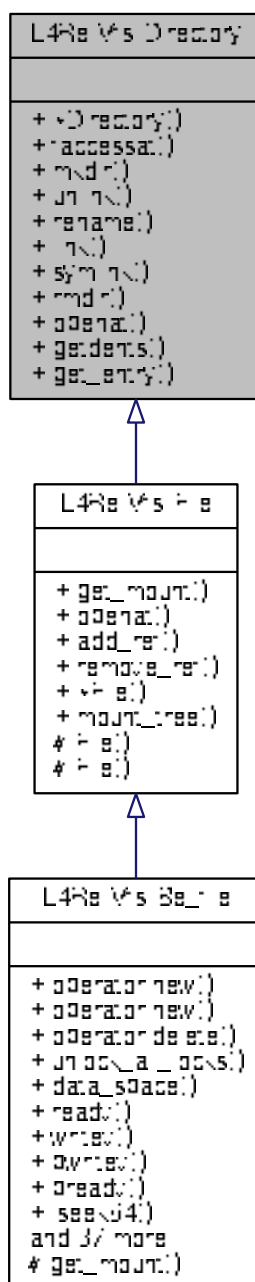
- [l4/l4re\\_vfs/backend](#)

## 14.263 L4Re::Vfs::Directory Class Reference

Interface for a POSIX file that is a directory.

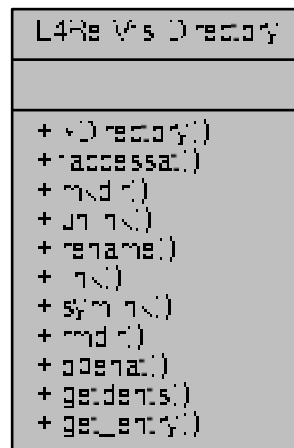
```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::Directory:





Collaboration diagram for L4Re::Vfs::Directory:



## Public Member Functions

- virtual int [faccessat](#) (const char \*path, int mode, int flags)=0 throw ()  
*Check access permissions on the given file.*
- virtual int [mkdir](#) (const char \*path, mode\_t mode)=0 throw ()  
*Create a new subdirectory.*
- virtual int [unlink](#) (const char \*path)=0 throw ()  
*Unlink the given file from that directory.*
- virtual int [rename](#) (const char \*src\_path, const char \*dst\_path)=0 throw ()  
*Rename the given file.*
- virtual int [link](#) (const char \*src\_path, const char \*dst\_path)=0 throw ()  
*Create a hard link (second name) for the given file.*
- virtual int [symlink](#) (const char \*src\_path, const char \*dst\_path)=0 throw ()  
*Create a symbolic link for the given file.*
- virtual int [rmdir](#) (const char \*path)=0 throw ()  
*Delete an empty directory.*

### 14.263.1 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 use always the combined [L4Re::Vfs::File](#) interface.

Definition at line 141 of file [vfs.h](#).

## 14.263.2 Member Function Documentation

### 14.263.2.1 faccessat()

```
virtual int L4Re::Vfs::Directory::faccessat (
 const char * path,
 int mode,
 int flags) throw () [pure virtual]
```

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](#).

### 14.263.2.2 link()

```
virtual int L4Re::Vfs::Directory::link (
 const char * src_path,
 const char * dst_path) throw () [pure virtual]
```

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 to 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](#).

### 14.263.2.3 mkdir()

```
virtual int L4Re::Vfs::Directory::mkdir (
 const char * path,
 mode_t mode) throw () [pure virtual]
```

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 is not a directory.

Implemented in [L4Re::Vfs::Be\\_file](#).

### 14.263.2.4 rename()

```
virtual int L4Re::Vfs::Directory::rename (
 const char * src_path,
 const char * dst_path) throw () [pure virtual]
```

Rename the given file.

Backend for the POSIX rename, renameat functions.

#### Parameters

|                 |                                                                                                                         |
|-----------------|-------------------------------------------------------------------------------------------------------------------------|
| <i>src_path</i> | The old name to 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](#).

#### 14.263.2.5 rmdir()

```
virtual int L4Re::Vfs::Directory::rmdir (
 const char * path) throw () [pure virtual]
```

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](#).

#### 14.263.2.6 symlink()

```
virtual int L4Re::Vfs::Directory::symlink (
 const char * src_path,
 const char * dst_path) throw () [pure virtual]
```

Create a symbolic link for the given file.

Backend for the POSIX symlink and symlinkat functions.

##### Parameters

|                 |                                                                                                           |
|-----------------|-----------------------------------------------------------------------------------------------------------|
| <i>src_path</i> | The old name to 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](#).

#### 14.263.2.7 unlink()

```
virtual int L4Re::Vfs::Directory::unlink (
 const char * path) throw () [pure virtual]
```

Unlink the given file from that directory.

Backend for the POSIX unlink and unlinkat functions.

## Parameters

|             |                                                                                                                 |
|-------------|-----------------------------------------------------------------------------------------------------------------|
| <i>path</i> | The name to 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:

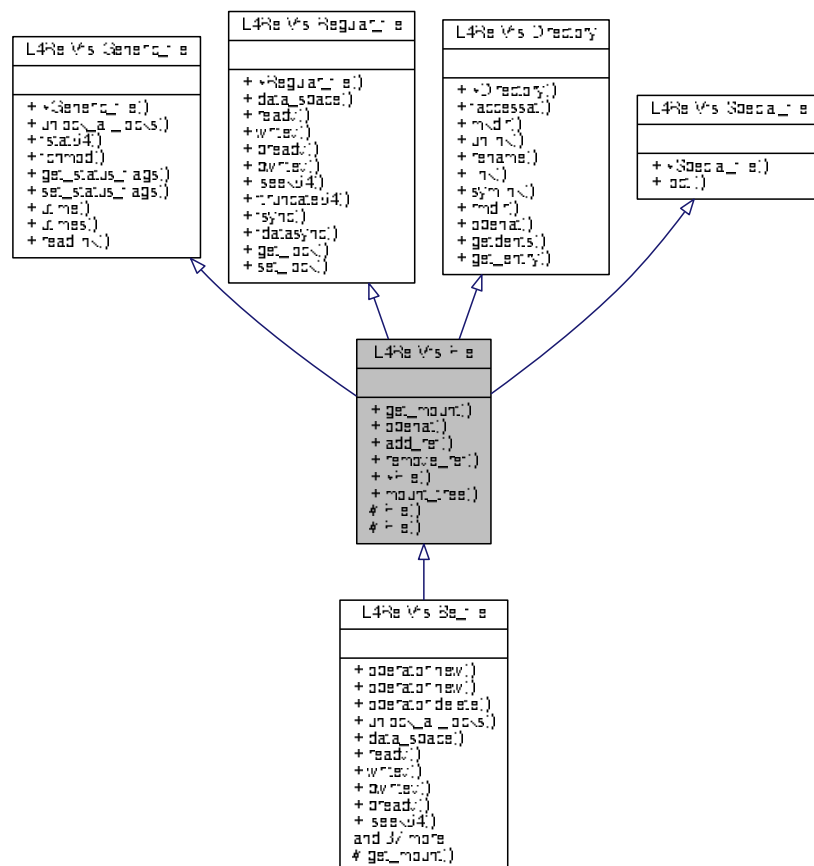
- `l4/l4re_vfs/vfs.h`

## 14.264 L4Re::Vfs::File Class Reference

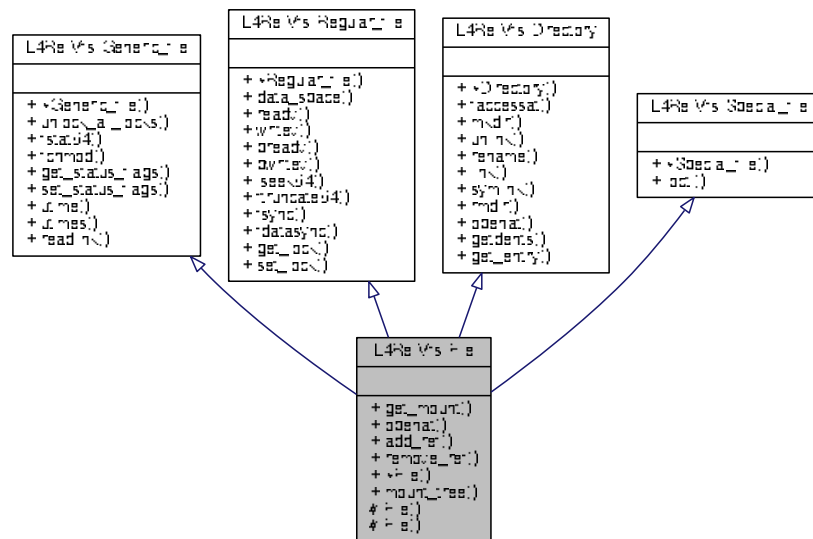
The basic interface for an open POSIX file.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::File:



Collaboration diagram for L4Re::Vfs::File:



## Additional Inherited Members

### 14.264.1 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 a 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](#) you may use [L4Re::Vfs::Be\\_file](#) as a base class.

Definition at line 430 of file [vfs.h](#).

The documentation for this class was generated from the following file:

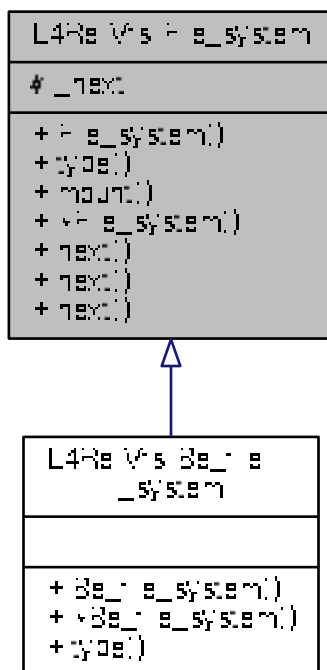
- `l4/l4re_vfs/vfs.h`

## 14.265 L4Re::Vfs::File\_system Class Reference

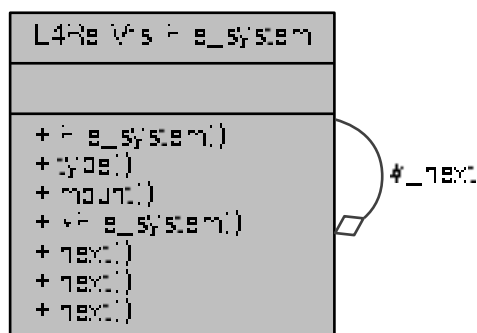
Basic interface for an [L4Re::Vfs](#) file system.

```
#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 =0 throw ()  
*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)=0 throw ()  
*Create a directory object dir representing source mounted with this file system.*

### 14.265.1 Detailed Description

Basic interface for an [L4Re::Vfs](#) file system.

#### Note

For implementing a special file system you may use [L4Re::Vfs::Be\\_file\\_system](#) as a base class.

The may purpose of this interface is that there is a single object for each supported file-system type (e.g., ext2, vfat) exists in your application and is registered at the [L4Re::Vfs::Fs](#) singleton available in via [L4Re::Vfs::vfs\\_ops](#). At the end the POSIX mount function call the [File\\_system::mount](#) method for the given file-system type given in mount.

Definition at line [867](#) of file [vfs.h](#).

### 14.265.2 Member Function Documentation

#### 14.265.2.1 mount()

```
virtual int L4Re::Vfs::File_system::mount (
 char const * source,
 unsigned long mountflags,
 void const * data,
 cxx::Ref_ptr< File > * dir) throw () [pure virtual]
```

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.               |

#### Return values

|            |                                                                     |
|------------|---------------------------------------------------------------------|
| <i>dir</i> | A new directory object representing the file-system root directory. |
|------------|---------------------------------------------------------------------|

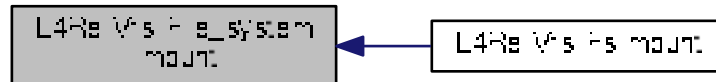


**Returns**

0 on success, and <0 on error (e.g. -EINVAL).

Referenced by [L4Re::Vfs::Fs::mount\(\)](#).

Here is the caller graph for this function:

**14.265.2.2 type()**

```
virtual char const* L4Re::Vfs::File_system::type () const throw () [pure virtual]
```

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:

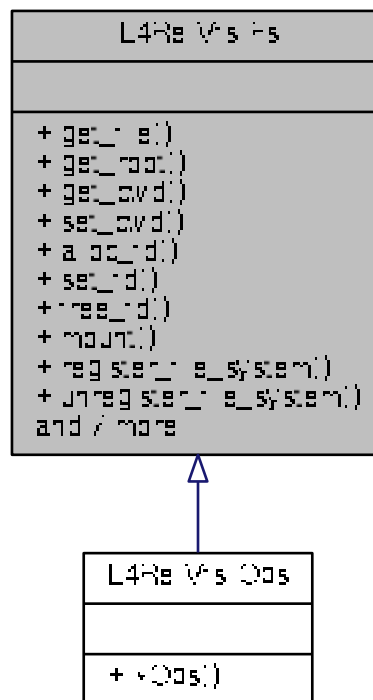
- l4/l4re\_vfs/vfs.h

**14.266 L4Re::Vfs::Fs Class Reference**

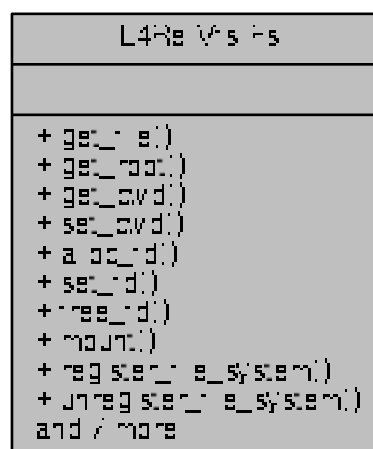
POSIX File-system related functionality.

```
#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)=0 throw ()  
*Get the `L4Re::Vfs::File` for the file descriptor fd.*
- virtual `cxx::Ref_ptr< File > get_root` ()=0 throw ()  
*Get the directory object for the applications root directory.*
- virtual `cxx::Ref_ptr< File > get_cwd` () throw ()  
*Get the directory object for the applications current working directory.*
- virtual void `set_cwd` (cxx::Ref\_ptr< File > const &) throw ()  
*Set the current working directory for the application.*
- virtual int `alloc_fd` (cxx::Ref\_ptr< File > const &f=cxx::Ref\_ptr<>::Nil)=0 throw ()  
*Allocate the next free file descriptor.*
- virtual `cxx::Ref_ptr< File > set_fd` (int fd, cxx::Ref\_ptr< File > const &f=cxx::Ref\_ptr<>::Nil)=0 throw ()  
*Set the file object referenced by the file descriptor fd.*
- virtual `cxx::Ref_ptr< File > free_fd` (int fd)=0 throw ()  
*Free the file descriptor fd.*
- int `mount` (char const \*path, cxx::Ref\_ptr< File > const &dir) throw ()  
*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) throw ()  
*Backend for the POSIX mount call.*

### 14.266.1 Detailed Description

POSIX File-system related functionality.

#### Note

This class usually exists as a singleton as a superclass of `L4Re::Vfs::Ops` (

#### See also

`L4Re::Vfs::vfs_ops`).

Definition at line 919 of file `vfs.h`.

### 14.266.2 Member Function Documentation

#### 14.266.2.1 alloc\_fd()

```
virtual int L4Re::Vfs::Fs::alloc_fd (
 cxx::Ref_ptr< File > const & f = cxx::Ref_ptr<>::Nil) throw () [pure virtual]
```

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.

**14.266.2.2 free\_fd()**

```
virtual cxx::Ref_ptr<File> L4Re::Vfs::Fs::free_fd (
 int fd) throw () [pure virtual]
```

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.

**14.266.2.3 get\_file()**

```
virtual cxx::Ref_ptr<File> L4Re::Vfs::Fs::get_file (
 int fd) throw () [pure virtual]
```

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.

**14.266.2.4 mount()**

```
int L4Re::Vfs::Fs::mount (
 char const * path,
 cxx::Ref_ptr< File > const & dir) throw () [inline]
```

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.

Definition at line 1013 of file [vfs.h](#).

#### 14.266.2.5 set\_fd()

```
virtual cxx::Ref_ptr<File> L4Re::Vfs::Fs::set_fd (
 int fd,
 cxx::Ref_ptr< File > const & f = cxx::Ref_ptr<>::Nil) throw () [pure virtual]
```

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 pointer to the file object that was previously assigned to *fd*.

The documentation for this class was generated from the following file:

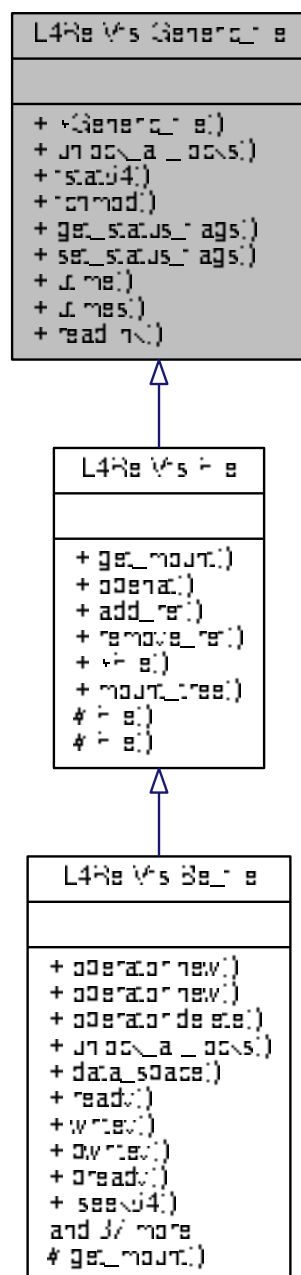
- l4/l4re\_vfs/vfs.h

## 14.267 L4Re::Vfs::Generic\_file Class Reference

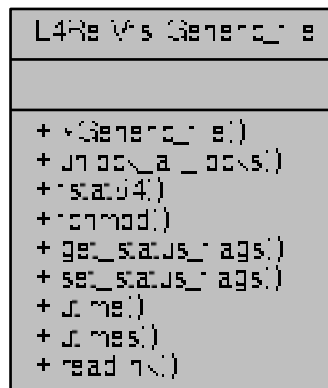
The common interface for an open POSIX file.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::Generic\_file:



Collaboration diagram for L4Re::Vfs::Generic\_file:



## Public Member Functions

- virtual int [unlock\\_all\\_locks](#) ()=0 throw ()  
*Unlock all locks on the file.*
- virtual int [fstat64](#) (struct stat64 \*buf) const =0 throw ()  
*Get status information for the file.*
- virtual int [fchmod](#) (mode\_t)=0 throw ()  
*Change POSIX access rights on that file.*
- virtual int [get\\_status\\_flags](#) () const =0 throw ()  
*Get file status flags (fcntl F\_GETFL).*
- virtual int [set\\_status\\_flags](#) (long flags)=0 throw ()  
*Set file status flags (fcntl F\_SETFL).*

### 14.267.1 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 mor information.

Definition at line 63 of file [vfs.h](#).

### 14.267.2 Member Function Documentation

**14.267.2.1 fchmod()**

```
virtual int L4Re::Vfs::Generic_file::fchmod (
 mode_t) throw) [pure virtual]
```

Change POSIX access rights on that file.

Backend for POSIX chmod and fchmod.

Implemented in [L4Re::Vfs::Be\\_file](#).

**14.267.2.2 fstat64()**

```
virtual int L4Re::Vfs::Generic_file::fstat64 (
 struct stat64 * buf) const throw) [pure virtual]
```

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.

Implemented in [L4Re::Vfs::Be\\_file](#).

**14.267.2.3 get\_status\_flags()**

```
virtual int L4Re::Vfs::Generic_file::get_status_flags () const throw) [pure virtual]
```

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](#).

**14.267.2.4 set\_status\_flags()**

```
virtual int L4Re::Vfs::Generic_file::set_status_flags (
 long flags) throw) [pure virtual]
```

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](#).

**14.267.2.5 unlock\_all\_locks()**

```
virtual int L4Re::Vfs::Generic_file::unlock_all_locks () throw () [pure virtual]
```

Unlock all locks on the file.

**Note**

All locks means all locks independent by which file the locks were taken.

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](#).

The documentation for this class was generated from the following file:

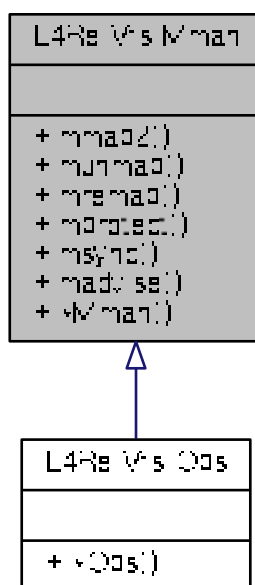
- l4/l4re\_vfs/vfs.h

## 14.268 L4Re::Vfs::Mman Class Reference

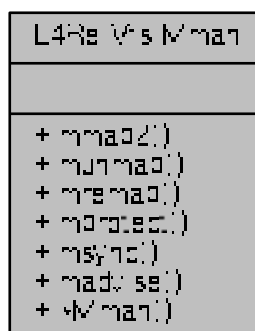
Interface for the POSIX memory management.

```
#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)=0 throw ()  
*Backend for the mmap2 system call.*
- virtual int [munmap](#) (void \*start, size\_t len)=0 throw ()  
*Backend for the munmap system call.*
- virtual int [mremap](#) (void \*old, size\_t old\_sz, size\_t new\_sz, int flags, void \*\*new\_addr)=0 throw ()  
*Backend for the mremap system call.*
- virtual int [mprotect](#) (const void \*a, size\_t sz, int prot)=0 throw ()  
*Backend for the mprotect system call.*
- virtual int [msync](#) (void \*addr, size\_t len, int flags)=0 throw ()  
*Backend for the msync system call.*
- virtual int [madvice](#) (void \*addr, size\_t len, int advice)=0 throw ()  
*Backend for the madvice system call.*

### 14.268.1 Detailed Description

Interface for the POSIX memory management.

#### Note

This interface exists usually as a singleton as 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 [789](#) of file [vfs.h](#).

The documentation for this class was generated from the following file:

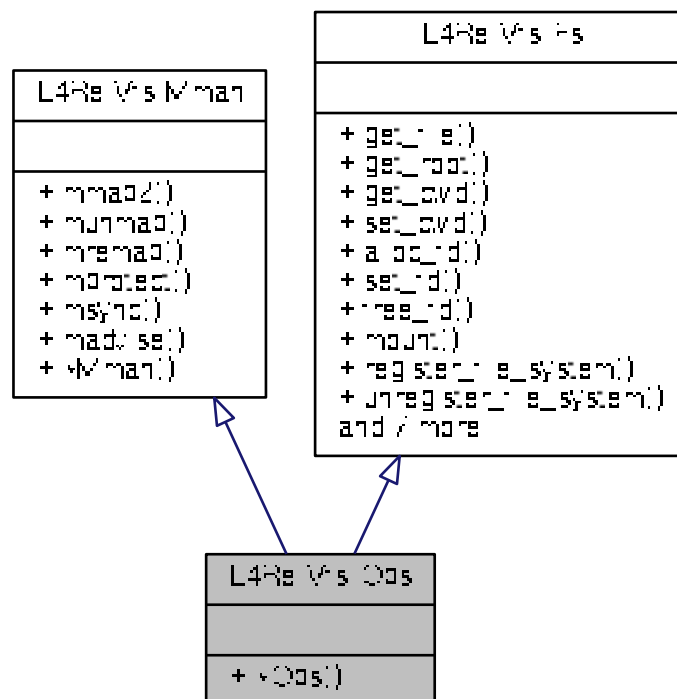
- [l4/l4re\\_vfs/vfs.h](#)

## 14.269 L4Re::Vfs::Ops Class Reference

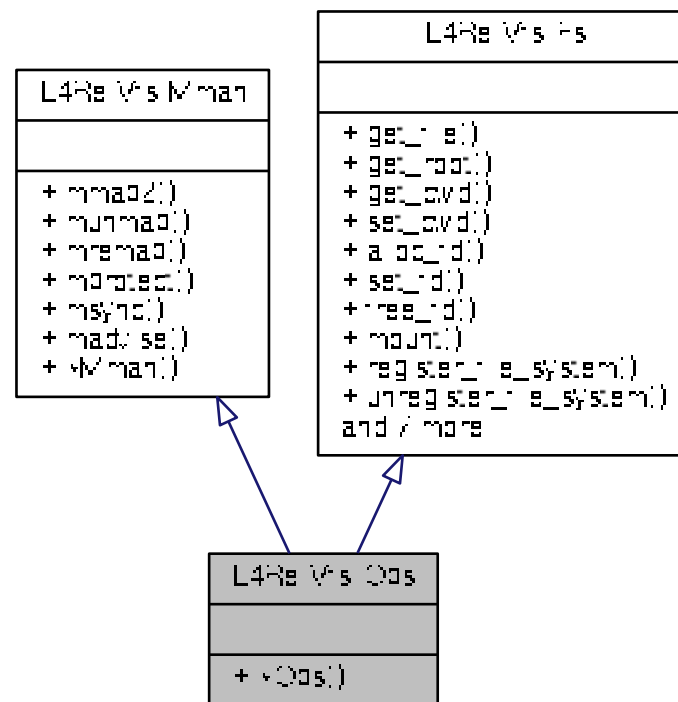
Interface for the POSIX backends for an application.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::Ops:



Collaboration diagram for L4Re::Vfs::Ops:



## Additional Inherited Members

### 14.269.1 Detailed Description

Interface for the POSIX backends for 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 1063 of file [vfs.h](#).

The documentation for this class was generated from the following file:

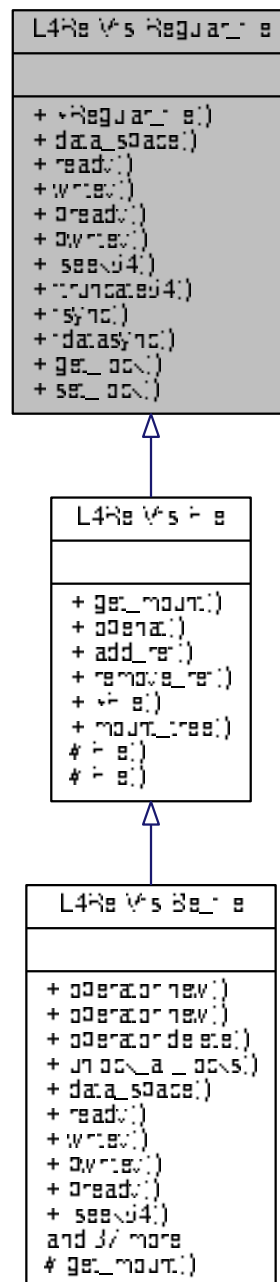
- `l4/l4re_vfs/vfs.h`

## 14.270 L4Re::Vfs::Regular\_file Class Reference

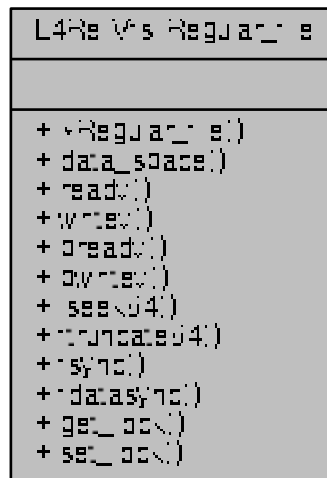
Interface for a POSIX file that provides regular file semantics.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::Regular\_file:



Collaboration diagram for L4Re::Vfs::Regular\_file:



## Public Member Functions

- virtual [L4::Cap< L4Re::Dataspace > data\\_space](#) () const =0 throw ()  
*Get an [L4Re::Dataspace](#) object for the file.*
- virtual ssize\_t [read](#) (const struct iovec \*, int iovcnt)=0 throw ()  
*Read one or more blocks of data from the file.*
- virtual ssize\_t [write](#) (const struct iovec \*, int iovcnt)=0 throw ()  
*Write one or more blocks of data to the file.*
- virtual off64\_t [lseek64](#) (off64\_t, int)=0 throw ()  
*Change the file pointer.*
- virtual int [ftruncate64](#) (off64\_t pos)=0 throw ()  
*Truncate the file at the given position.*
- virtual int [fsync](#) () const =0 throw ()  
*Sync the data and meta data to persistent storage.*
- virtual int [datasync](#) () const =0 throw ()  
*Sync the data to persistent storage.*
- virtual int [get\\_lock](#) (struct flock64 \*lock)=0 throw ()  
*Test if the given lock can be placed in the file.*
- virtual int [set\\_lock](#) (struct flock64 \*lock, bool wait)=0 throw ()  
*Acquire or release the given lock on the file.*

### 14.270.1 Detailed Description

Interface for a POSIX file that provides regular file semantics.

Real objects use always the combined [L4Re::Vfs::File](#) interface.

Definition at line 262 of file [vfs.h](#).

## 14.270.2 Member Function Documentation

### 14.270.2.1 data\_space()

```
virtual L4::Cap<L4Re::Dataspace> L4Re::Vfs::Regular_file::data_space () const throw () [pure virtual]
```

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 functions returns an invalid capability.

#### Returns

A capability to an [L4Re::Dataspace](#), that represents the files contents in an [L4Re](#) way.

Implemented in [L4Re::Vfs::Be\\_file](#).

### 14.270.2.2 fdatsync()

```
virtual int L4Re::Vfs::Regular_file::fdatsync () const throw () [pure virtual]
```

Sync the data to persistent storage.

This is the backend for POSIX fdatsync.

Implemented in [L4Re::Vfs::Be\\_file](#).

### 14.270.2.3 fsync()

```
virtual int L4Re::Vfs::Regular_file::fsync () const throw () [pure virtual]
```

Sync the data and meta data to persistent storage.

This is the backend for POSIX fsync.

Implemented in [L4Re::Vfs::Be\\_file](#).

### 14.270.2.4 ftruncate64()

```
virtual int L4Re::Vfs::Regular_file::ftruncate64 (
 off64_t pos) throw () [pure virtual]
```

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](#).

**14.270.2.5 get\_lock()**

```
virtual int L4Re::Vfs::Regular_file::get_lock (
 struct flock64 * lock) throw () [pure virtual]
```

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 F_UNLCK if the lock could be placed. |
|-------------|-----------------------------------------------------------------------------------------------------------------------|

**Returns**

0 on success, <0 on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

**14.270.2.6 lseek64()**

```
virtual off64_t L4Re::Vfs::Regular_file::lseek64 (
 off64_t ,
 int) throw () [pure virtual]
```

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](#).

#### 14.270.2.7 readv()

```
virtual ssize_t L4Re::Vfs::Regular_file::readv (
 const struct iovec * ,
 int iovcnt) throw () [pure virtual]
```

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 for the `f_pos` pointer of that open file. The file pointer is advanced according to the number of read bytes.

##### Returns

The number of bytes read from the file. or <0 on error-

Implemented in [L4Re::Vfs::Be\\_file](#).

#### 14.270.2.8 set\_lock()

```
virtual int L4Re::Vfs::Regular_file::set_lock (
 struct flock64 * lock,
 bool wait) throw () [pure virtual]
```

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](#).

#### 14.270.2.9 writev()

```
virtual ssize_t L4Re::Vfs::Regular_file::writev (
 const struct iovec * ,
 int iovcnt) throw () [pure virtual]
```

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](#).

The documentation for this class was generated from the following file:

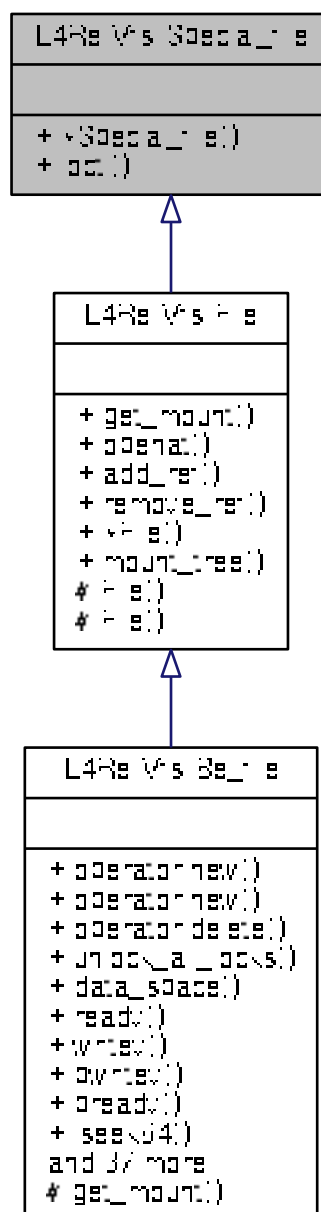
- l4/l4re\_vfs/vfs.h

## 14.271 L4Re::Vfs::Special\_file Class Reference

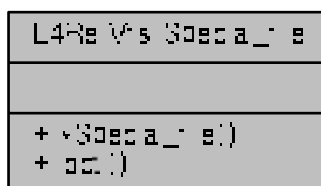
Interface for a POSIX file that provides special file semantics.

```
#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)=0 throw ()  
*The famous IO control.*

### 14.271.1 Detailed Description

Interface for a POSIX file that provides special file semantics.

Real objects use always the combined [L4Re::Vfs::File](#) interface.

Definition at line [395](#) of file [vfs.h](#).

### 14.271.2 Member Function Documentation

#### 14.271.2.1 ioctl()

```
virtual int L4Re::Vfs::Special_file::ioctl (
 unsigned long cmd,
 va_list args) throw () [pure virtual]
```

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**

$\geq 0$  on success, or  $< 0$  on error.

Implemented in [L4Re::Vfs::Be\\_file](#).

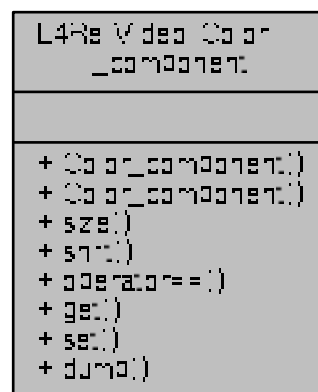
The documentation for this class was generated from the following file:

- [l4/l4re\\_vfs/vfs.h](#)

## 14.272 L4Re::Video::Color\_component Class Reference

A color component.

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.*

### 14.272.1 Detailed Description

A color component.

Definition at line 31 of file [colors](#).

### 14.272.2 Constructor & Destructor Documentation

#### 14.272.2.1 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 46 of file [colors](#).

### 14.272.3 Member Function Documentation

#### 14.272.3.1 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 |
|----------|--------|

##### Returns

The stream

Definition at line 93 of file [colors](#).

### 14.272.3.2 get()

```
int L4Re::Video::Color_component::get (
 unsigned long v) const [inline]
```

Get component from value (normalized to 16bits).

#### Parameters

|   |       |
|---|-------|
| v | Value |
|---|-------|

#### Returns

Converted value

Definition at line 73 of file [colors](#).

### 14.272.3.3 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 65 of file [colors](#).

### 14.272.3.4 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 84 of file [colors](#).



## 14.272.3.5 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 59 of file [colors](#).

## 14.272.3.6 size()

```
unsigned char L4Re::Video::Color_component::size () const [inline]
```

Return the number of bits used by the component.

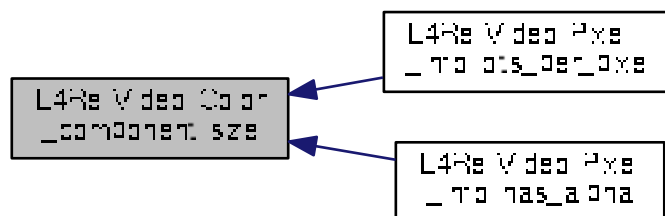
**Returns**

Number of bits used by the component

Definition at line 53 of file [colors](#).

Referenced by [L4Re::Video::Pixel\\_info::bits\\_per\\_pixel\(\)](#), and [L4Re::Video::Pixel\\_info::has\\_alpha\(\)](#).

Here is the caller graph for this function:



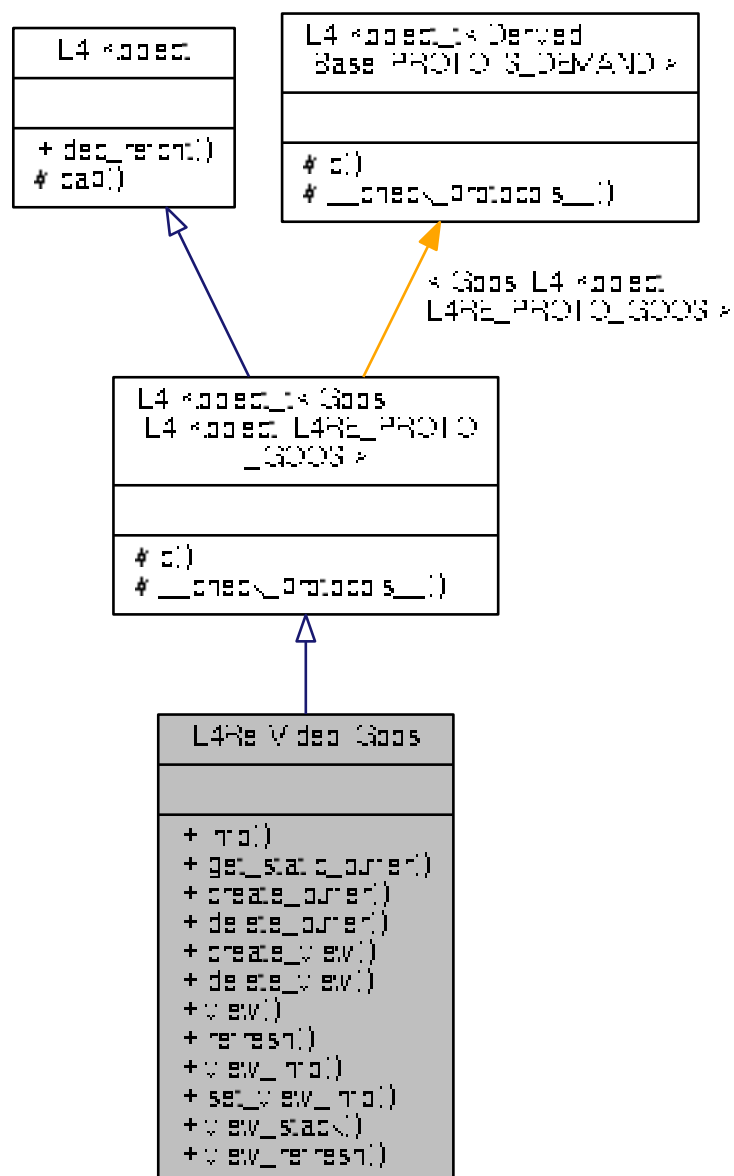
The documentation for this class was generated from the following file:

- [l4/re/video/colors](#)

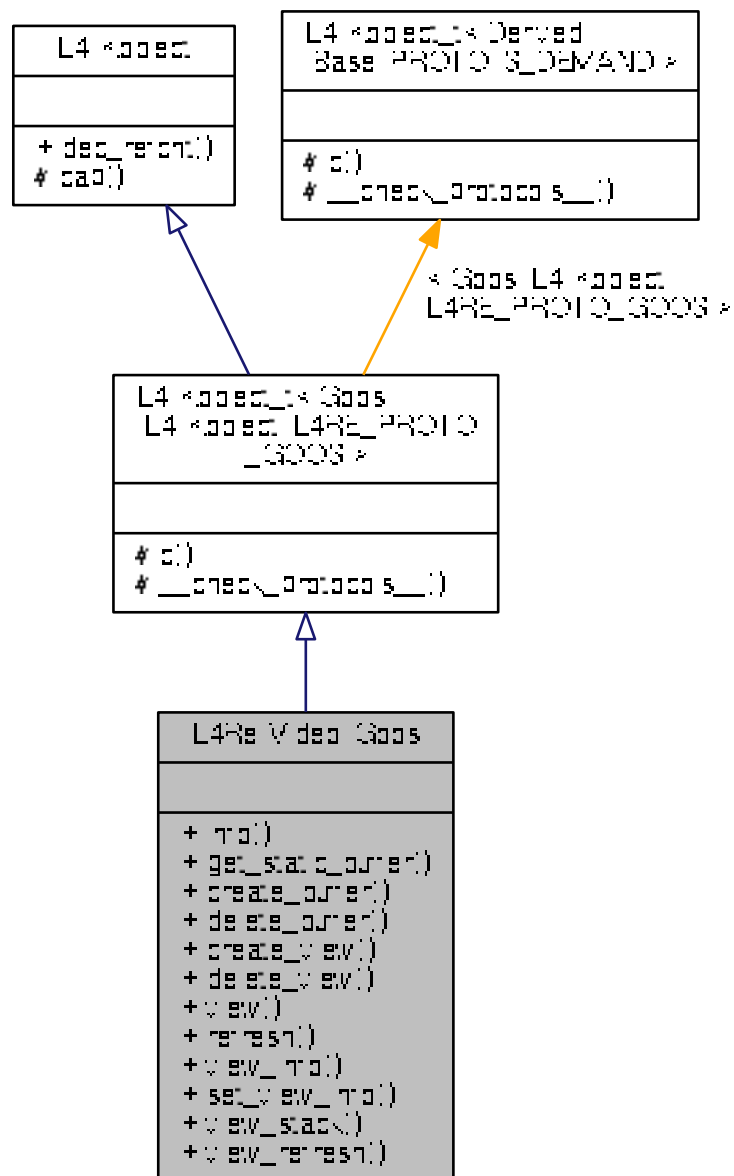
## 14.273 L4Re::Video::Goos Class Reference

A 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

- long [info](#) ([Info](#) \*info)  
*Return the goos information of the goos.*
- long [get\\_static\\_buffer](#) (unsigned idx, [L4::lpc::Out](#)< [L4::Cap](#)< [L4Re::Dataspace](#) > > rbuf)  
*Return a static buffer of a goos.*
- long [create\\_buffer](#) (unsigned long size, [L4::lpc::Out](#)< [L4::Cap](#)< [L4Re::Dataspace](#) > > rbuf)  
*Create a buffer.*
- long [delete\\_buffer](#) (unsigned idx)  
*Delete a buffer.*
- int [create\\_view](#) ([View](#) \*view, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) const throw ()  
*Create a view.*
- int [delete\\_view](#) ([View](#) const &v, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb](#)()) const throw ()  
*Delete a view.*
- [View](#) [view](#) (unsigned index) const throw ()  
*Return a view.*
- long [refresh](#) (int x, int y, int w, int h)  
*Trigger refreshing of the given area on the virtual screen.*

## Additional Inherited Members

### 14.273.1 Detailed Description

A goos.

Definition at line 207 of file [goos](#).

### 14.273.2 Member Enumeration Documentation

#### 14.273.2.1 Flags

```
enum L4Re::Video::Goos::Flags
```

Flags for a goos.

#### Enumerator

|                                   |                                                  |
|-----------------------------------|--------------------------------------------------|
| <a href="#">F_auto_refresh</a>    | The graphics display is automatically refreshed. |
| <a href="#">F_pointer</a>         | We have a mouse pointer.                         |
| <a href="#">F_dynamic_views</a>   | Supports dynamically allocated views.            |
| <a href="#">F_dynamic_buffers</a> | Supports dynamically allocated buffers.          |

Definition at line 212 of file [goos](#).

### 14.273.3 Member Function Documentation

#### 14.273.3.1 create\_buffer()

```
long 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

|          |                                                  |
|----------|--------------------------------------------------|
| $\geq 0$ | Success, the value returned is the buffer index. |
| $< 0$    | Error                                            |

#### 14.273.3.2 create\_view()

```
int L4Re::Video::Goos::create_view (
 View * view,
 l4_utcb_t * utcb = l4_utcb()) const throw () [inline]
```

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

|          |                                                |
|----------|------------------------------------------------|
| $\geq 0$ | Success, the value returned is the view index. |
| $< 0$    | Error                                          |

Definition at line 296 of file [goos](#).

References [L4\\_INLINE\\_RPC\\_NF](#).

### 14.273.3.3 delete\_buffer()

```
long L4Re::Video::Goos::delete_buffer (
 unsigned idx)
```

Delete a buffer.

#### Parameters

|            |                   |
|------------|-------------------|
| <i>idx</i> | Buffer to delete. |
|------------|-------------------|

#### Return values

|    |         |
|----|---------|
| 0  | Success |
| <0 | Error   |

### 14.273.3.4 delete\_view()

```
int L4Re::Video::Goos::delete_view (
 View const & v,
 l4_utcb_t * utcb = l4_utcb()) const throw () [inline]
```

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 316 of file [goos](#).

References [L4\\_INLINE\\_RPC](#).

### 14.273.3.5 get\_static\_buffer()

```
long 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

|              |         |
|--------------|---------|
| <i>0</i>     | Success |
| <i>&lt;0</i> | Error   |

## 14.273.3.6 info()

```
long L4Re::Video::Goos::info (
 Info * info)
```

Return the goos information of the goos.

## Parameters

|     |      |                                     |
|-----|------|-------------------------------------|
| out | info | Goos information structure pointer. |
|-----|------|-------------------------------------|

## Return values

|              |         |
|--------------|---------|
| <i>0</i>     | Success |
| <i>&lt;0</i> | Error   |

## 14.273.3.7 view()

```
View L4Re::Video::Goos::view (
 unsigned index) const throw () [inline]
```

Return a view.

## Parameters

|              |                              |
|--------------|------------------------------|
| <i>index</i> | Index of the view to return. |
|--------------|------------------------------|

## Returns

The view.

Definition at line 347 of file [goos](#).

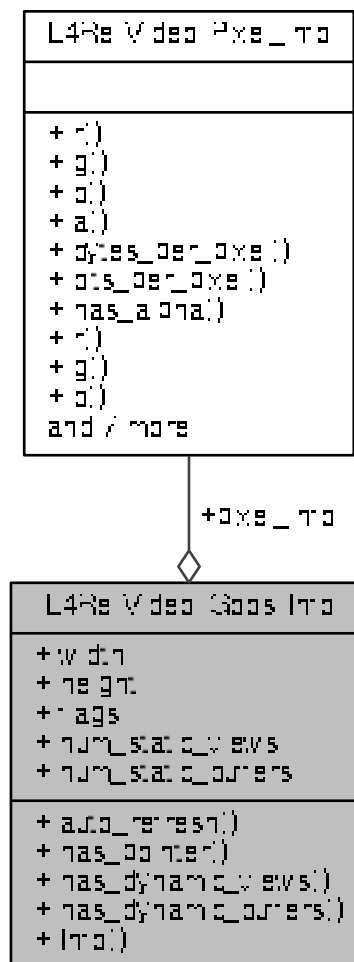
The documentation for this class was generated from the following file:

- [l4/re/video/goos](#)

## 14.274 L4Re::Video::Goos::Info Struct Reference

Information structure of a 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.*

### 14.274.1 Detailed Description

Information structure of a goos.

Definition at line [221](#) of file [goos](#).

### 14.274.2 Member Function Documentation

#### 14.274.2.1 [auto\\_refresh\(\)](#)

```
bool L4Re::Video::Goos::Info::auto_refresh () const [inline]
```

Return whether this goos does auto refreshing or the view refresh functions must be used to make changes visible.

Definition at line [232](#) of file [goos](#).

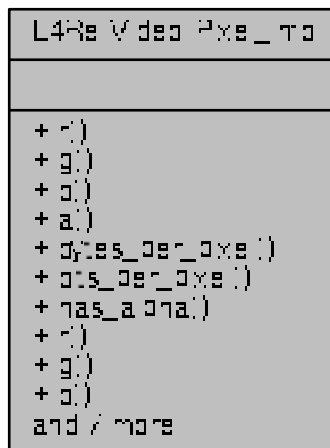
The documentation for this struct was generated from the following file:

- [l4/re/video/goos](#)

## 14.275 L4Re::Video::Pixel\_info Class Reference

Pixel information.

Collaboration diagram for L4Re::Video::Pixel\_info:



### 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.*
- 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.*

### 14.275.1 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 [106](#) of file [colors](#).

### 14.275.2 Constructor & Destructor Documentation

#### 14.275.2.1 [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 203 of file [colors](#).

#### 14.275.2.2 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 215 of file [colors](#).

### 14.275.3 Member Function Documentation

#### 14.275.3.1 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 135 of file [colors](#).

#### 14.275.3.2 a() [2/2]

```
void L4Re::Video::Pixel_info::a (
 Color_component const & c) [inline]
```

Set the alpha color component of the pixel.

##### Parameters

|          |                        |
|----------|------------------------|
| <i>c</i> | Alpha color component. |
|----------|------------------------|

Definition at line 178 of file [colors](#).

#### 14.275.3.3 `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 129 of file [colors](#).

#### 14.275.3.4 `b()` [2/2]

```
void L4Re::Video::Pixel_info::b (
 Color_component const & c) [inline]
```

Set the blue color component of the pixel.

##### Parameters

|                |                       |
|----------------|-----------------------|
| <code>c</code> | Blue color component. |
|----------------|-----------------------|

Definition at line 172 of file [colors](#).

#### 14.275.3.5 `bits_per_pixel()`

```
unsigned char L4Re::Video::Pixel_info::bits_per_pixel () const [inline]
```

Number of bits of the pixel.

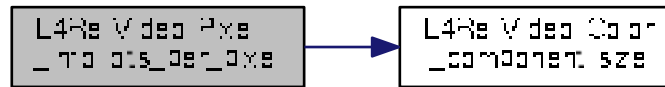
##### Returns

Number of bits used by the pixel.

Definition at line 147 of file [colors](#).

References [L4Re::Video::Color\\_component::size\(\)](#).

Here is the call graph for this function:



#### 14.275.3.6 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 [141](#) of file [colors](#).

#### 14.275.3.7 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 [184](#) of file [colors](#).

#### 14.275.3.8 dump()

```
template<typename OUT >
void L4Re::Video::Pixel_info::dump (
 OUT & s) const [inline]
```

Dump information on the pixel to a stream.

## Parameters

|   |        |
|---|--------|
| s | Stream |
|---|--------|

## Returns

The stream

Definition at line 238 of file [colors](#).

Referenced by [L4Re::Video::View::Info::dump\(\)](#).

Here is the caller graph for this function:



## 14.275.3.9 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 123 of file [colors](#).

## 14.275.3.10 g() [2/2]

```
void L4Re::Video::Pixel_info::g (
 Color_component const & c) [inline]
```

Set the green color component of the pixel.

## Parameters

|                |                        |
|----------------|------------------------|
| <code>c</code> | Green color component. |
|----------------|------------------------|

Definition at line 166 of file [colors](#).

14.275.3.11 `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 154 of file [colors](#).

References [L4Re::Video::Color\\_component::size\(\)](#).

Here is the call graph for this function:

14.275.3.12 `operator==()`

```
bool L4Re::Video::Pixel_info::operator== (
 Pixel_info const & o) const [inline]
```

Compare for complete equality of the color space.

## Parameters

|                |                                             |
|----------------|---------------------------------------------|
| <code>o</code> | A <a href="#">Pixel_info</a> to compare to. |
|----------------|---------------------------------------------|



**Returns**

true if the both [Pixel\\_info](#)'s are equal, false if not.

Definition at line [227](#) of file [colors](#).

**14.275.3.13** [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 [117](#) of file [colors](#).

**14.275.3.14** [r\(\)](#) [2/2]

```
void L4Re::Video::Pixel_info::r (
 Color_component const & c) [inline]
```

Set the red color component of the pixel.

**Parameters**

|          |                      |
|----------|----------------------|
| <b>c</b> | Red color component. |
|----------|----------------------|

Definition at line [160](#) of file [colors](#).

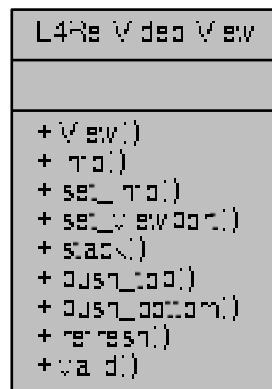
The documentation for this class was generated from the following file:

- [l4/re/video/colors](#)

## 14.276 L4Re::Video::View Class Reference

[View](#).

Collaboration diagram for L4Re::Video::View:



## 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 throw ()  
*Return the view information of the view.*
- int [set\\_info](#) ([Info](#) const &info) const throw ()  
*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 throw ()  
*Set the position of the view in the goos.*
- int [stack](#) ([View](#) const &pivot, bool behind=true) const throw ()  
*Move this view in the view stack.*
- int [push\\_top](#) () const throw ()  
*Make this view the top-most view.*
- int [push\\_bottom](#) () const throw ()  
*Push this view the back.*
- int [refresh](#) (int x, int y, int w, int h) const throw ()  
*Refresh/Redraw the view.*
- bool [valid](#) () const  
*Return whether this view is valid.*

### 14.276.1 Detailed Description

[View](#).

Definition at line 34 of file [goos](#).

### 14.276.2 Member Enumeration Documentation

#### 14.276.2.1 Flags

enum [L4Re::Video::View::Flags](#)

Flags on a view.

Enumerator

|                      |                                                                    |
|----------------------|--------------------------------------------------------------------|
| F_none               | everything for this view is static (the VESA-FB case)              |
| F_set_buffer         | buffer object for this view can be changed                         |
| F_set_buffer_offset  | buffer offset can be set                                           |
| F_set_bytes_per_line | 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      | <a href="#">View</a> is dynamically allocated.                     |
| F_set_background     | Set view as background for session.                                |
| F_set_flags          | Set view flags (.<br><br>See also<br><br><a href="#">V_flags</a> ) |
| F_fully_dynamic      | Flags for a fully dynamic view.                                    |

Definition at line 54 of file [goos](#).

#### 14.276.2.2 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 77 of file [goos](#).

### 14.276.3 Member Function Documentation

#### 14.276.3.1 info()

```
int L4Re::Video::View::info (
 Info * info) const throw) [inline]
```

Return the view information of the view.

##### Parameters

|     |             |                                |
|-----|-------------|--------------------------------|
| out | <i>info</i> | Information structure pointer. |
|-----|-------------|--------------------------------|

##### Return values

|    |         |
|----|---------|
| 0  | Success |
| <0 | Error   |

Definition at line 351 of file [goos](#).

#### 14.276.3.2 refresh()

```
int L4Re::Video::View::refresh (
 int x,
 int y,
 int w,
 int h) const throw) [inline]
```

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 363 of file [goos](#).

### 14.276.3.3 set\_info()

```
int L4Re::Video::View::set_info (
 Info const & info) const throw () [inline]
```

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 355 of file [goos](#).

### 14.276.3.4 set\_viewport()

```
int L4Re::Video::View::set_viewport (
 int scr_x,
 int scr_y,
 int w,
 int h,
 unsigned long buf_offset) const throw () [inline]
```

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 367 of file [goos](#).

References [L4Re::Video::View::Info::buffer\\_offset](#), [L4Re::Video::View::Info::flags](#), [L4Re::Video::View::Info::height](#), [L4Re::Video::View::Info::width](#), [L4Re::Video::View::Info::xpos](#), and [L4Re::Video::View::Info::ypos](#).

#### 14.276.3.5 stack()

```
int L4Re::Video::View::stack (
 View const & pivot,
 bool behind = true) const throw () [inline]
```

Move this view in the view stack.

##### Parameters

|               |                                                                                              |
|---------------|----------------------------------------------------------------------------------------------|
| <i>pivot</i>  | <a href="#">View</a> 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 359 of file [goos](#).

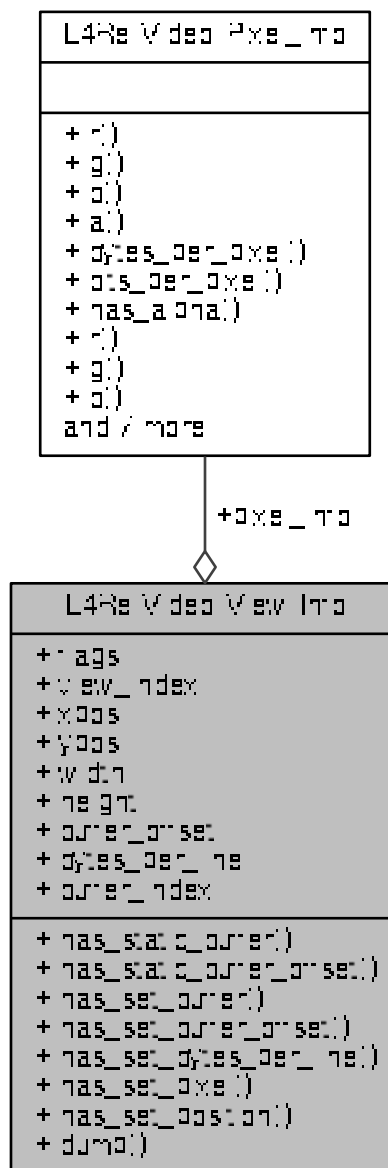
The documentation for this class was generated from the following file:

- [l4/re/video/goos](#)

## 14.277 L4Re::Video::View::Info Struct Reference

Information structure of a view.

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](#)  
*Flags, see [Flags](#) and [V\\_flags](#).*
- unsigned [view\\_index](#)  
*Index of the view.*
- unsigned long [xpos](#)  
*X position in pixels of the view in the goos.*
- unsigned long [ypos](#)  
*Y position in pixels of the view in the goos.*
- unsigned long [width](#)  
*Width of the view in pixels.*
- unsigned long [height](#)  
*Height of the view in pixels.*
- unsigned long [buffer\\_offset](#)  
*Offset in the memory buffer in bytes.*
- unsigned long [bytes\\_per\\_line](#)  
*Bytes per line.*
- [Pixel\\_info](#) [pixel\\_info](#)  
*Pixel information.*
- unsigned [buffer\\_index](#)  
*Number of the buffer used for this view.*

### 14.277.1 Detailed Description

Information structure of a view.

Definition at line 86 of file [goos](#).

The documentation for this struct was generated from the following file:

- [l4/re/video/goos](#)

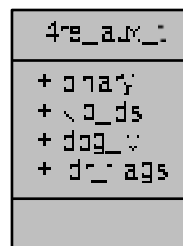


## 14.278 l4re\_aux\_t Struct Reference

Auxiliary descriptor.

```
#include <l4aux.h>
```

Collaboration diagram for l4re\_aux\_t:



### 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.*

### 14.278.1 Detailed Description

Auxiliary descriptor.

Definition at line 51 of file [l4aux.h](#).

The documentation for this struct was generated from the following file:

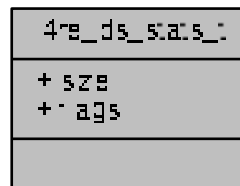
- [l4re/l4aux.h](#)

## 14.279 l4re\_ds\_stats\_t Struct Reference

Information about the data space.

```
#include <dataspace.h>
```

Collaboration diagram for l4re\_ds\_stats\_t:



### Data Fields

- unsigned long [size](#)  
*size*
- unsigned long [flags](#)  
*flags*

### 14.279.1 Detailed Description

Information about the data space.

Definition at line 45 of file [dataspace.h](#).

The documentation for this struct was generated from the following file:

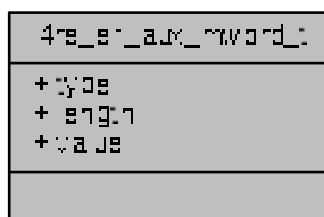
- [l4re/c/dataspace.h](#)

## 14.280 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:



### 14.280.1 Detailed Description

Auxiliary vector element for a single unsigned data word.

Definition at line 124 of file [elf\\_aux.h](#).

The documentation for this struct was generated from the following file:

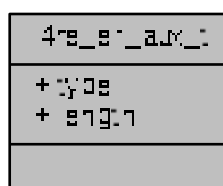
- [l4re/elf\\_aux.h](#)

## 14.281 l4re\_elf\_aux\_t Struct Reference

Generic header for each auxiliary vector element.

```
#include <elf_aux.h>
```

Collaboration diagram for l4re\_elf\_aux\_t:



### 14.281.1 Detailed Description

Generic header for each auxiliary vector element.

Definition at line 104 of file [elf\\_aux.h](#).

The documentation for this struct was generated from the following file:

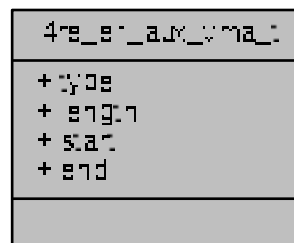
- [l4/re/elf\\_aux.h](#)

## 14.282 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:



### 14.282.1 Detailed Description

Auxiliary vector element for a reserved virtual memory area.

Definition at line 113 of file [elf\\_aux.h](#).

The documentation for this struct was generated from the following file:

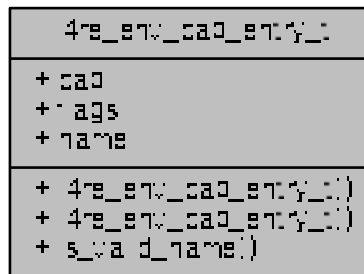
- [l4/re/elf\\_aux.h](#)

## 14.283 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\(\)](#)  
*Create an invalid entry.*
- [l4re\\_env\\_cap\\_entry\\_t](#) (char const \*n, [l4\\_cap\\_idx\\_t](#) c, [l4\\_umword\\_t](#) f=0)  
*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  
*Some flags for the object.*
- char [name](#) [16]  
*The name of the object.*

### 14.283.1 Detailed Description

Entry in the [L4Re](#) environment array for the named initial objects.

Definition at line 46 of file [env.h](#).

### 14.283.2 Constructor & Destructor Documentation

### 14.283.2.1 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*.

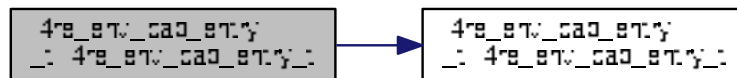
#### Parameters

|          |                                                            |
|----------|------------------------------------------------------------|
| <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 77 of file [env.h](#).

References [l4re\\_env\\_cap\\_entry\\_t\(\)](#), and [name](#).

Here is the call graph for this function:



## 14.283.3 Field Documentation

### 14.283.3.1 flags

```
l4_umword_t l4re_env_cap_entry_t::flags
```

Some flags for the object.

#### Note

Currently unused.

Definition at line 57 of file [env.h](#).

Referenced by [l4re\\_env\\_get\\_cap\\_l\(\)](#).

The documentation for this struct was generated from the following file:

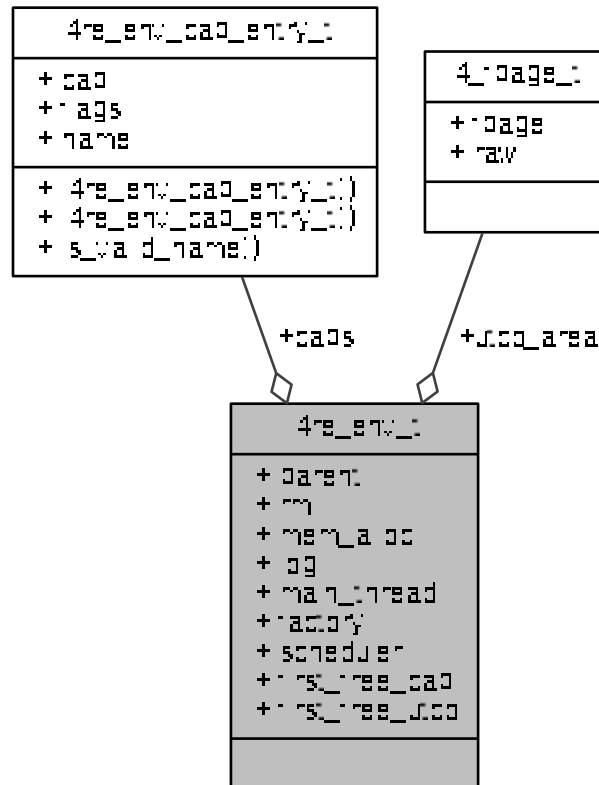
- [l4/re/env.h](#)

## 14.284 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 first\\_free\\_cap](#)

*First 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.*

### 14.284.1 Detailed Description

Initial environment data structure.

See also

[Initial environment](#)

Definition at line 105 of file [env.h](#).

The documentation for this struct was generated from the following file:

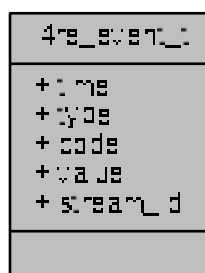
- [l4/re/env.h](#)

### 14.285 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.*

### 14.285.1 Detailed Description

Event structure used in buffer.

Definition at line 40 of file [event.h](#).

The documentation for this struct was generated from the following file:

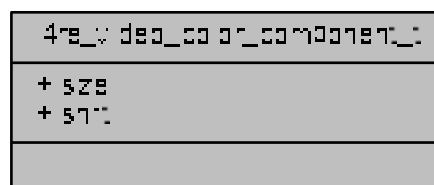
- [l4/re/c/event.h](#)

## 14.286 l4re\_video\_color\_component\_t Struct Reference

Color component structure.

```
#include <colors.h>
```

Collaboration diagram for l4re\_video\_color\_component\_t:



## Data Fields

- unsigned char [size](#)  
*Size in bits.*
- unsigned char [shift](#)  
*offset in pixel*

### 14.286.1 Detailed Description

Color component structure.

Definition at line 31 of file [colors.h](#).

The documentation for this struct was generated from the following file:

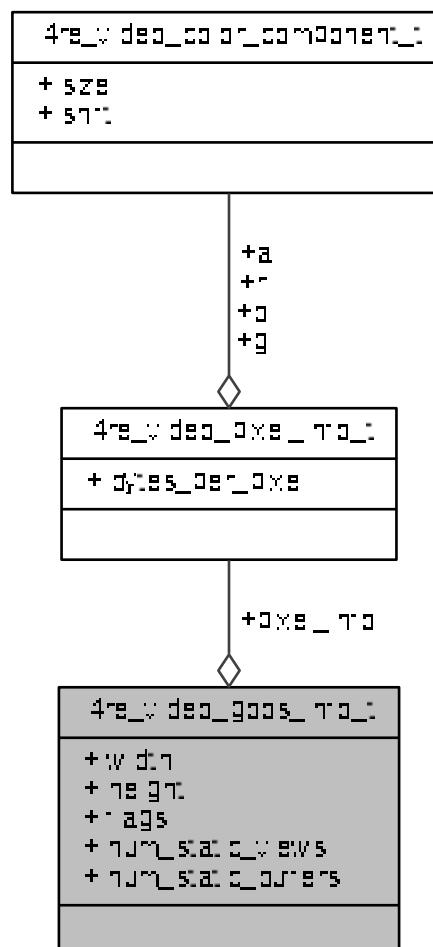
- [l4re/c/video/colors.h](#)

### 14.287 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.*

### 14.287.1 Detailed Description

Goos information structure.

Definition at line 51 of file [goos.h](#).

The documentation for this struct was generated from the following file:

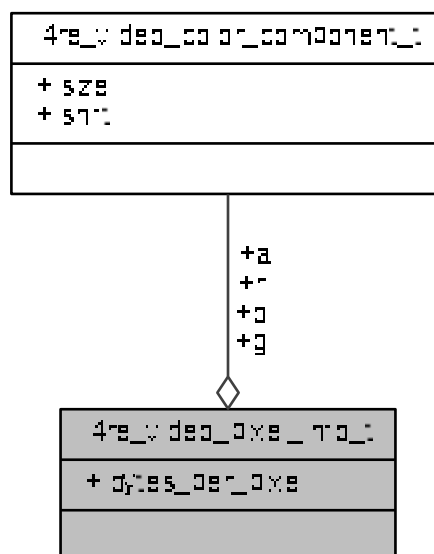
- [l4re/c/video/goos.h](#)

## 14.288 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.*

### 14.288.1 Detailed Description

Pixel\_info structure.

Definition at line [41](#) of file [colors.h](#).

The documentation for this struct was generated from the following file:

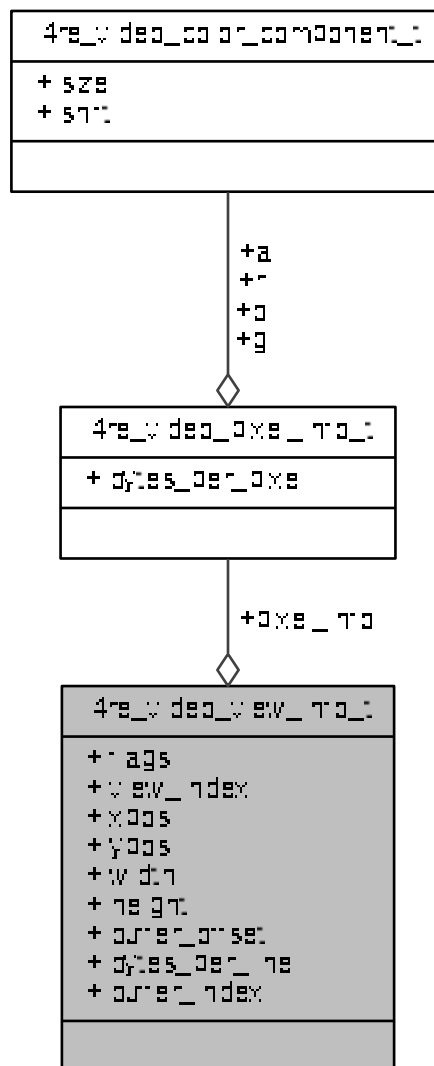
- [l4re/c/video/colors.h](#)

## 14.289 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.*

### 14.289.1 Detailed Description

View information structure.

Definition at line 59 of file [view.h](#).

The documentation for this struct was generated from the following file:

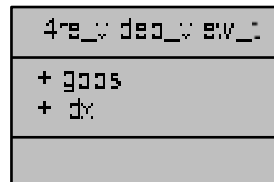
- [l4re/c/video/view.h](#)

## 14.290 l4re\_video\_view\_t Struct Reference

C representation of a goos view.

```
#include <view.h>
```

Collaboration diagram for `l4re_video_view_t`:



### 14.290.1 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 78 of file [view.h](#).

The documentation for this struct was generated from the following file:

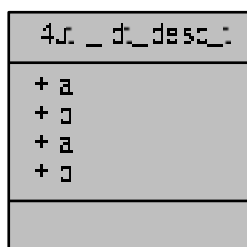
- [l4re/c/video/view.h](#)

## 14.291 l4util\_idt\_desc\_t Struct Reference

IDT entry.

```
#include <idt.h>
```

Collaboration diagram for l4util\_idt\_desc\_t:



### Data Fields

- [l4\\_uint64\\_t b](#)  
*see Intel doc*
- [l4\\_uint32\\_t b](#)  
*see Intel doc*

### 14.291.1 Detailed Description

IDT entry.

Definition at line 33 of file [idt.h](#).

The documentation for this struct was generated from the following file:

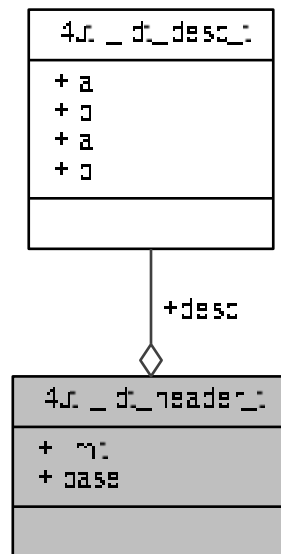
- amd64/l4/util/[idt.h](#)

## 14.292 l4util\_idt\_header\_t Struct Reference

Header of an IDT table.

```
#include <idt.h>
```

Collaboration diagram for l4util\_idt\_header\_t:



### Data Fields

- [l4\\_uint16\\_t limit](#)  
*limit field (see Intel doc)*
- void \* [base](#)  
*idt base (see Intel doc)*

### 14.292.1 Detailed Description

Header of an IDT table.

Definition at line 40 of file [idt.h](#).

The documentation for this struct was generated from the following file:

- amd64/l4/util/[idt.h](#)

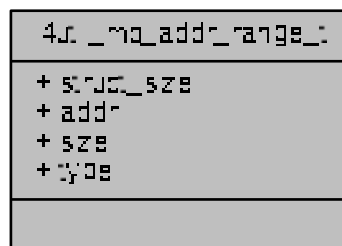


## 14.293 l4util\_mb\_addr\_range\_t Struct Reference

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:



### Data Fields

- [l4\\_uint64\\_t addr](#)  
*<Size of structure*
- [l4\\_uint64\\_t size](#)  
*<Start address*
- [l4\\_uint32\\_t type](#)  
*<Size of memory range*

### 14.293.1 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 47 of file [mb\\_info.h](#).

The documentation for this struct was generated from the following file:

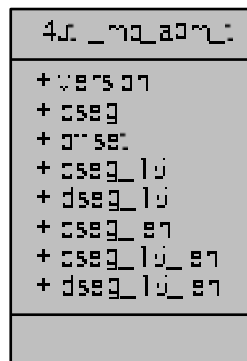
- [l4/util/mb\\_info.h](#)

## 14.294 l4util\_mb\_apm\_t Struct Reference

APM BIOS info.

```
#include <mb_info.h>
```

Collaboration diagram for l4util\_mb\_apm\_t:



### 14.294.1 Detailed Description

APM BIOS info.

Definition at line 95 of file [mb\\_info.h](#).

The documentation for this struct was generated from the following file:

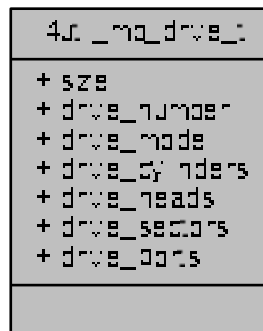
- [l4/util/mb\\_info.h](#)

## 14.295 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

## 14.295.1 Detailed Description

Drive Info structure.

Definition at line 78 of file [mb\\_info.h](#).

## 14.295.2 Field Documentation

### 14.295.2.1 drive\_cylinders

[l4\\_uint16\\_t](#) l4util\_mb\_drive\_t::drive\_cylinders

<The access mode (see below).

Definition at line 83 of file [mb\\_info.h](#).

## 14.295.2.2 drive\_mode

```
l4_uint8_t l4util_mb_drive_t::drive_mode
```

<The BIOS drive number.

Definition at line 82 of file [mb\\_info.h](#).

## 14.295.2.3 drive\_number

```
l4_uint8_t l4util_mb_drive_t::drive_number
```

<The size of this structure.

Definition at line 81 of file [mb\\_info.h](#).

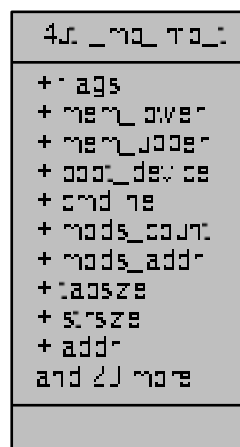
The documentation for this struct was generated from the following file:

- [l4/util/mb\\_info.h](#)

## 14.296 l4util\_mb\_info\_t Struct Reference

```
#include <mb_info.h>
```

Collaboration diagram for l4util\_mb\_info\_t:



## 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.*
- [l4\\_uint32\\_t tabsize](#)  
*(a.out) Kernel symbol table info*
- [l4\\_uint32\\_t num](#)  
*(ELF) Kernel section header table*

### 14.296.1 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 207 of file [mb\\_info.h](#).

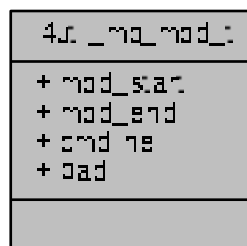
The documentation for this struct was generated from the following file:

- [l4/util/mb\\_info.h](#)

### 14.297 l4util\_mb\_mod\_t Struct Reference

```
#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*

### 14.297.1 Detailed Description

The structure type "mod\_list" is used by the [multiboot\\_info](#) structure.

Definition at line 31 of file [mb\\_info.h](#).

## 14.297.2 Field Documentation

### 14.297.2.1 mod\_end

`l4_uint32_t l4util_mb_mod_t::mod_end`

End address of module in memory.

Definition at line 34 of file [mb\\_info.h](#).

### 14.297.2.2 mod\_start

`l4_uint32_t l4util_mb_mod_t::mod_start`

Starting address of module in memory.

Definition at line 33 of file [mb\\_info.h](#).

The documentation for this struct was generated from the following file:

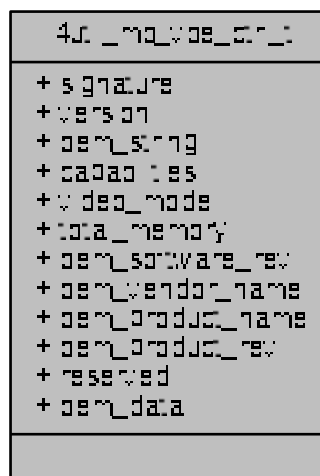
- [l4/util/mb\\_info.h](#)

## 14.298 l4util\_mb\_vbe\_ctrl\_t Struct Reference

VBE controller information.

```
#include <mb_info.h>
```

Collaboration diagram for `l4util_mb_vbe_ctrl_t`:



### 14.298.1 Detailed Description

VBE controller information.

Definition at line [109](#) of file [mb\\_info.h](#).

The documentation for this struct was generated from the following file:

- [l4/util/mb\\_info.h](#)

## 14.299 l4util\_mb\_vbe\_mode\_t Struct Reference

VBE mode information.

```
#include <mb_info.h>
```



[illegible]

**all VESA versions**

- `l4_uint16_t` `mode_attributes`
- `l4_uint8_t` `win_a_attributes`
- `l4_uint8_t` `win_b_attributes`
- `l4_uint16_t` `win_granularity`

- [l4\\_uint16\\_t](#) win\_size
- [l4\\_uint16\\_t](#) win\_a\_segment
- [l4\\_uint16\\_t](#) win\_b\_segment
- [l4\\_uint32\\_t](#) win\_func
- [l4\\_uint16\\_t](#) bytes\_per\_scanline

#### >= VESA version 1.2

- [l4\\_uint16\\_t](#) x\_resolution
- [l4\\_uint16\\_t](#) y\_resolution
- [l4\\_uint8\\_t](#) x\_char\_size
- [l4\\_uint8\\_t](#) y\_char\_size
- [l4\\_uint8\\_t](#) number\_of\_planes
- [l4\\_uint8\\_t](#) bits\_per\_pixel
- [l4\\_uint8\\_t](#) number\_of\_banks
- [l4\\_uint8\\_t](#) memory\_model
- [l4\\_uint8\\_t](#) bank\_size
- [l4\\_uint8\\_t](#) number\_of\_image\_pages
- [l4\\_uint8\\_t](#) reserved0

#### direct color

- [l4\\_uint8\\_t](#) red\_mask\_size
- [l4\\_uint8\\_t](#) red\_field\_position
- [l4\\_uint8\\_t](#) green\_mask\_size
- [l4\\_uint8\\_t](#) green\_field\_position
- [l4\\_uint8\\_t](#) blue\_mask\_size
- [l4\\_uint8\\_t](#) blue\_field\_position
- [l4\\_uint8\\_t](#) reserved\_mask\_size
- [l4\\_uint8\\_t](#) reserved\_field\_position
- [l4\\_uint8\\_t](#) direct\_color\_mode\_info

#### >= VESA version 2.0

- [l4\\_uint32\\_t](#) phys\_base
- [l4\\_uint32\\_t](#) reserved1
- [l4\\_uint16\\_t](#) reversed2

#### >= VESA version 3.0

- [l4\\_uint16\\_t](#) linear\_bytes\_per\_scanline
- [l4\\_uint8\\_t](#) banked\_number\_of\_image\_pages
- [l4\\_uint8\\_t](#) linear\_number\_of\_image\_pages
- [l4\\_uint8\\_t](#) linear\_red\_mask\_size
- [l4\\_uint8\\_t](#) linear\_red\_field\_position
- [l4\\_uint8\\_t](#) linear\_green\_mask\_size
- [l4\\_uint8\\_t](#) linear\_green\_field\_position
- [l4\\_uint8\\_t](#) linear\_blue\_mask\_size
- [l4\\_uint8\\_t](#) linear\_blue\_field\_position
- [l4\\_uint8\\_t](#) linear\_reserved\_mask\_size
- [l4\\_uint8\\_t](#) linear\_reserved\_field\_position
- [l4\\_uint32\\_t](#) max\_pixel\_clock
- [l4\\_uint8\\_t](#) reserved3 [189+1]

### 14.299.1 Detailed Description

VBE mode information.

Definition at line 127 of file [mb\\_info.h](#).

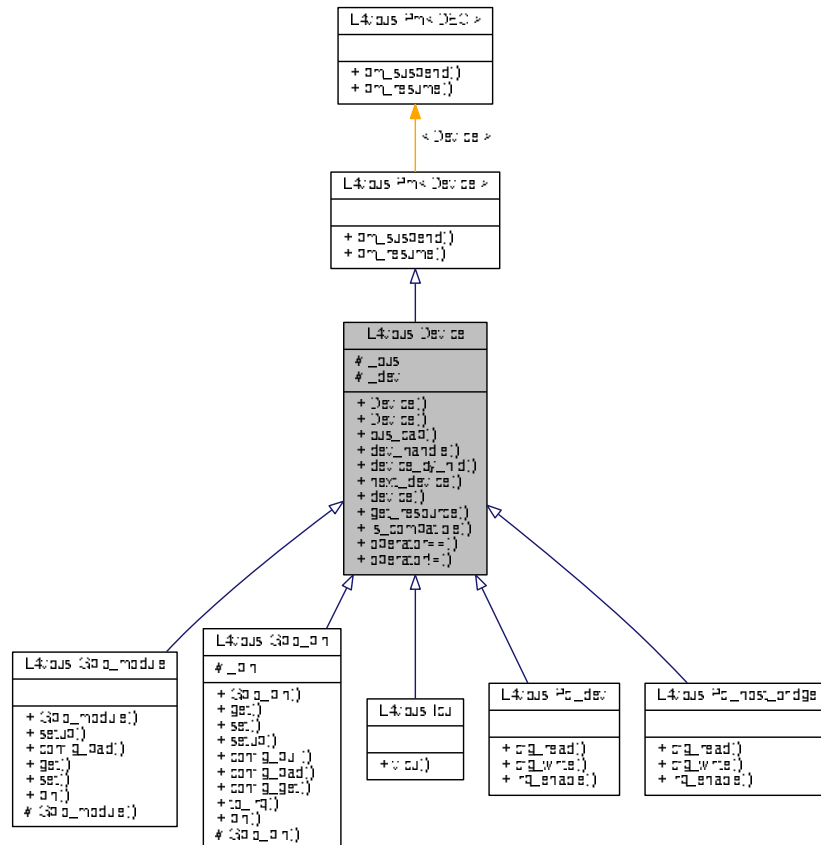
The documentation for this struct was generated from the following file:

- [l4/util/mb\\_info.h](#)

## 14.300 L4vbus::Device Class Reference

[Device](#) on a virtual bus (V-BUS)

Inheritance diagram for L4vbus::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 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](#) (int 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 V-BUS device.*
- bool [operator!=](#) ([Device](#) const &o) const  
*Test if two devices are not the same.*

## Protected Attributes

- [L4::Cap< Vbus > \\_bus](#)
- [l4vbus\\_device\\_handle\\_t \\_dev](#)  
*The device handle for this device.*

### 14.300.1 Detailed Description

[Device](#) on a virtual bus (V-BUS)

Definition at line 66 of file [vbus](#).

### 14.300.2 Member Function Documentation

#### 14.300.2.1 [bus\\_cap\(\)](#)

```
L4::Cap<Vbus> L4vbus::Device::bus_cap () const [inline]
```

Access the V-BUS capability of the underlying virtual bus.

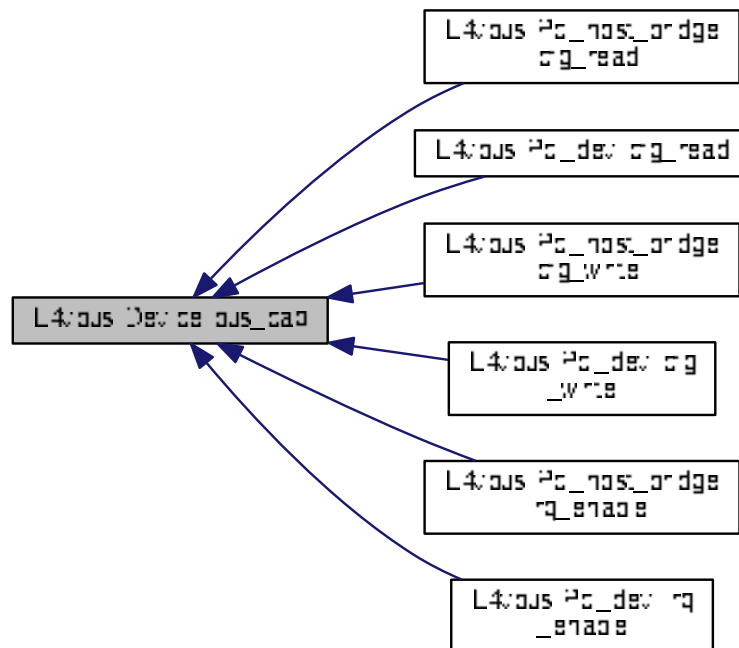
**Returns**

the capability to the underlying V-BUS.

Definition at line 78 of file [vbus](#).

Referenced by [L4vbus::Pci\\_host\\_bridge::cfg\\_read\(\)](#), [L4vbus::Pci\\_dev::cfg\\_read\(\)](#), [L4vbus::Pci\\_host\\_bridge::cfg\\_write\(\)](#), [L4vbus::Pci\\_dev::cfg\\_write\(\)](#), [L4vbus::Pci\\_host\\_bridge::irq\\_enable\(\)](#), and [L4vbus::Pci\\_dev::irq\\_enable\(\)](#).

Here is the caller graph for this function:

**14.300.2.2 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 87 of file [vbus](#).

### 14.300.2.3 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 after a successful call. |
|-----|----------------|-------------------------------------------------------------------------------------------------------------------|

## Return values

|            |                                                                   |
|------------|-------------------------------------------------------------------|
| 0          | Success.                                                          |
| -L4_ENODEV | No device with the given device handle <i>dev</i> could be found. |

Definition at line 159 of file [vbus](#).

References [l4vbus\\_get\\_device\(\)](#).

Here is the call graph for this function:



## 14.300.2.4 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 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 ID.

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 ( <a href="#">L4VBUS_NULL</a> ). 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> | <a href="#">Device</a> information structure (might be NULL)                                                                                                                                                                                                                       |



## Return values

|                         |                                                          |
|-------------------------|----------------------------------------------------------|
| <code>&gt;=</code>      | 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 119 of file [vbus](#).

14.300.2.5 `get_resource()`

```
int L4vbus::Device::get_resource (
 int 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 <a href="#">l4vbus_device_t</a> structure that is returned by <a href="#">L4vbus::Device::device_by_hid()</a> and <a href="#">L4vbus::Device::next_device()</a> . |
| 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

|                         |                                               |
|-------------------------|-----------------------------------------------|
| <code>0</code>          | Success.                                      |
| <code>-L4_ENOENT</code> | Invalid resource index <code>res_idx</code> . |

Definition at line 179 of file [vbus](#).

References [l4vbus\\_get\\_resource\(\)](#).

Here is the call graph for this function:



### 14.300.2.6 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 |
|------------|------------------------------|

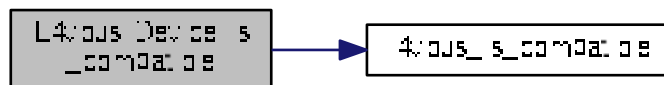
#### Returns

1 when the given ID (*cid*) matches this device, 0 when the given ID does not match, <0 on error.

Definition at line 193 of file [vbus](#).

References [l4vbus\\_is\\_compatible\(\)](#).

Here is the call graph for this function:



### 14.300.2.7 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 found. To start from the beginning <i>child</i> must be initialized using the default ( <a href="#">L4VBUS_NULL</a> ). |
|         | <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 140 of file [vbus](#).

**14.300.2.8 operator!=()**

```
bool L4vbus::Device::operator!= (
 Device const & o) const [inline]
```

Test if two devices are not the same.

**Returns**

true if the two devices are different, false else.

Definition at line 209 of file [vbus](#).

References [\\_bus](#), and [\\_dev](#).

**14.300.2.9 operator==()**

```
bool L4vbus::Device::operator== (
 Device const & o) const [inline]
```

Test if two devices are the same V-BUS device.

**Returns**

true if the two devices are the same, false else.

Definition at line 200 of file [vbus](#).

References [\\_bus](#), and [\\_dev](#).

**14.300.3 Field Documentation**

### 14.300.3.1 `_bus`

`L4::Cap<Vbus> L4vbus::Device::_bus` [protected]

The V-BUS capability (where this device is located on).

Definition at line 215 of file `vbus`.

Referenced by `L4vbus::Gpio_pin::config_get()`, `L4vbus::Gpio_pin::config_pad()`, `L4vbus::Gpio_module::config_pad()`, `L4vbus::Gpio_pin::config_pull()`, `L4vbus::Gpio_pin::get()`, `L4vbus::Gpio_module::get()`, `operator!=()`, `operator==()`, `L4vbus::Gpio_pin::set()`, `L4vbus::Gpio_module::set()`, `L4vbus::Gpio_pin::setup()`, `L4vbus::Gpio_module::setup()`, and `L4vbus::Gpio_pin::to_irq()`.

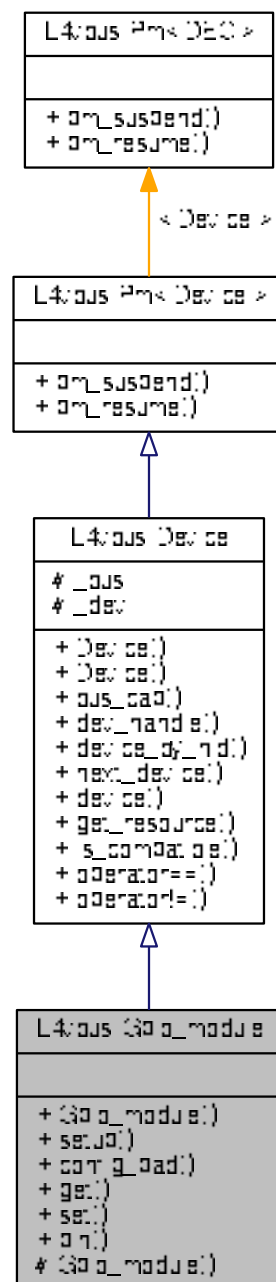
The documentation for this class was generated from the following file:

- `I4/vbus/vbus`

## 14.301 `L4vbus::Gpio_module` Class Reference

A `Gpio_module` groups multiple GPIO pins together.

Inheritance diagram for L4vbus::Gpio\_module:





- 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](#).*

## Additional Inherited Members

### 14.301.1 Detailed Description

A [Gpio\\_module](#) groups multiple GPIO pins together.

Definition at line 129 of file [vbus\\_gpio](#).

### 14.301.2 Member Function Documentation

#### 14.301.2.1 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                                                                                            |

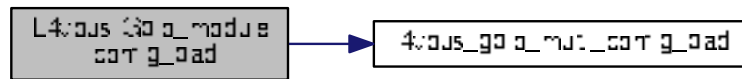
#### Returns

0 if OK, error code otherwise

Definition at line 181 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:



#### 14.301.2.2 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 197 of file [vbus\\_gpio](#).

References [L4vbus::Device::\\_bus](#), [L4vbus::Device::\\_dev](#), and [l4vbus\\_gpio\\_multi\\_get\(\)](#).

Here is the call graph for this function:



#### 14.301.2.3 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 <a href="#">Gpio_pin</a> for. |
|------------|------------------------------------------------------|

## Returns

[Gpio\\_pin](#)

Definition at line 225 of file [vbus\\_gpio](#).

## 14.301.2.4 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 213 of file [vbus\\_gpio](#).

References [L4vbus::Device::\\_bus](#), [L4vbus::Device::\\_dev](#), and [l4vbus\\_gpio\\_multi\\_set\(\)](#).

Here is the call graph for this function:



### 14.301.2.5 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 <a href="#">L4vbus_gpio_generic_func</a> 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                                                                                                               |

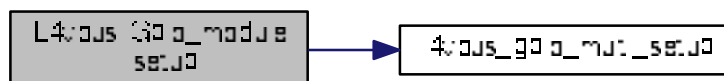
#### Returns

0 if OK, error code otherwise

Definition at line 162 of file [vbus\\_gpio](#).

References [L4vbus::Device::\\_bus](#), [L4vbus::Device::\\_dev](#), and [l4vbus\\_gpio\\_multi\\_setup\(\)](#).

Here is the call graph for this function:



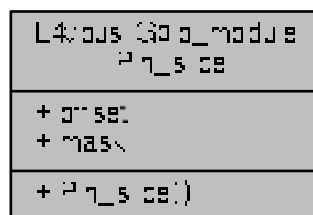
The documentation for this class was generated from the following file:

- `I4/vbus/vbus_gpio`

## 14.302 L4vbus::Gpio\_module::Pin\_slice Struct Reference

A slice of the pins provided by this module.

Collaboration diagram for L4vbus::Gpio\_module::Pin\_slice:



### 14.302.1 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 142 of file [vbus\\_gpio](#).

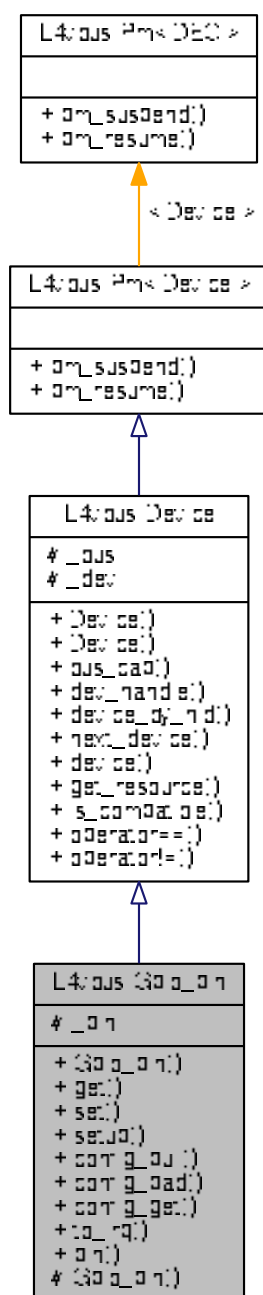
The documentation for this struct was generated from the following file:

- [l4/vbus/vbus\\_gpio](#)

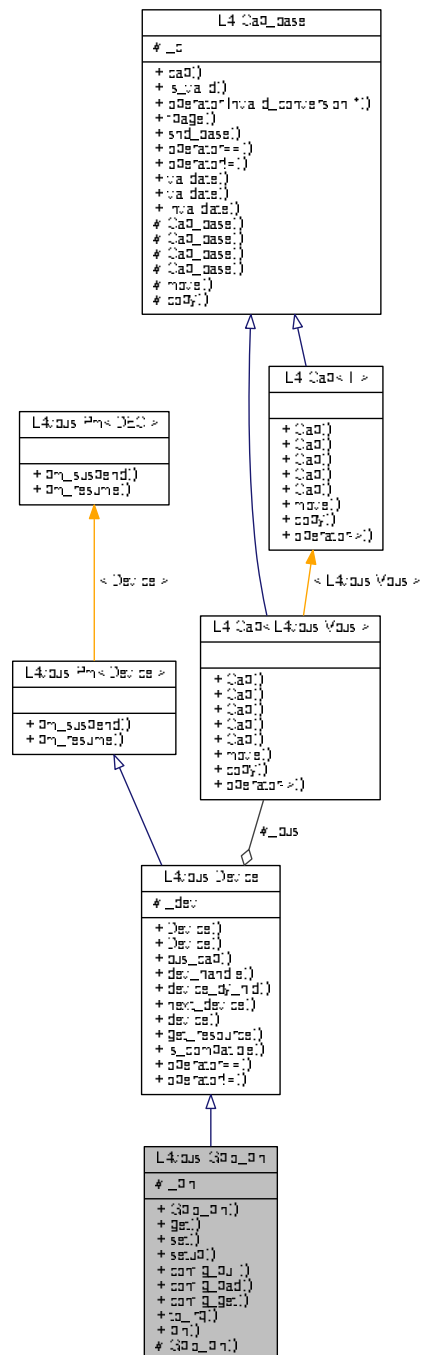
## 14.303 L4vbus::Gpio\_pin Class Reference

A GPIO pin.

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.*

## Additional Inherited Members

### 14.303.1 Detailed Description

A GPIO pin.

Definition at line 22 of file [vbus\\_gpio](#).

### 14.303.2 Member Function Documentation

#### 14.303.2.1 [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 98 of file [vbus\\_gpio](#).

References [L4vbus::Device::\\_bus](#), [L4vbus::Device::\\_dev](#), and [l4vbus\\_gpio\\_config\\_get\(\)](#).

Here is the call graph for this function:



#### 14.303.2.2 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 85 of file [vbus\\_gpio](#).

References [L4vbus::Device::\\_bus](#), [L4vbus::Device::\\_dev](#), and [l4vbus\\_gpio\\_config\\_pad\(\)](#).

Here is the call graph for this function:



#### 14.303.2.3 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 <a href="#">L4vbus_gpio_pull_modes</a> |
|-------------|-----------------------------------------------------------------------------|

## Returns

0 if OK, error code otherwise

Definition at line 71 of file [vbus\\_gpio](#).

References [L4vbus::Device::\\_bus](#), [L4vbus::Device::\\_dev](#), and [l4vbus\\_gpio\\_config\\_pull\(\)](#).

Here is the call graph for this function:

14.303.2.4 `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 34 of file [vbus\\_gpio](#).

References [L4vbus::Device::\\_bus](#), [L4vbus::Device::\\_dev](#), and [l4vbus\\_gpio\\_get\(\)](#).

Here is the call graph for this function:





## 14.303.2.5 pin()

```
unsigned L4vbus::Gpio_pin::pin () const [inline]
```

Get pin number.

## Returns

GPIO pin number

Definition at line 118 of file [vbus\\_gpio](#).

## 14.303.2.6 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 45 of file [vbus\\_gpio](#).

References [L4vbus::Device::\\_bus](#), [L4vbus::Device::\\_dev](#), and [l4vbus\\_gpio\\_set\(\)](#).

Here is the call graph for this function:



## 14.303.2.7 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 <a href="#">L4vbus_gpio_generic_func</a> 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 60 of file [vbus\\_gpio](#).

References [L4vbus::Device::\\_bus](#), [L4vbus::Device::\\_dev](#), and [l4vbus\\_gpio\\_setup\(\)](#).

Here is the call graph for this function:



## 14.303.2.8 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 108 of file [vbus\\_gpio](#).

References [L4vbus::Device::\\_bus](#), [L4vbus::Device::\\_dev](#), and [l4vbus\\_gpio\\_to\\_irq\(\)](#).

Here is the call graph for this function:



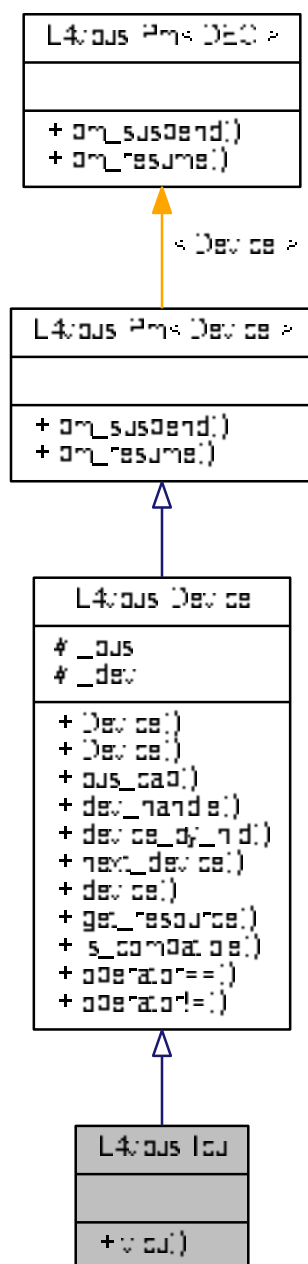
The documentation for this class was generated from the following file:

- [l4/vbus/vbus\\_gpio](#)

## 14.304 L4vbus::lcu Class Reference

V-BUS Interrupt controller API (ICU)

Inheritance diagram for L4vbus::lcu:





- Request the L4Re::lcu capability for this V-BUS ICU.*

## Additional Inherited Members

### 14.304.1 Detailed Description

V-BUS Interrupt controller API (ICU)

Allows to access the underlying L4Re::lcu capability managing IRQs for the V-BUS.

Definition at line 226 of file [vbus](#).

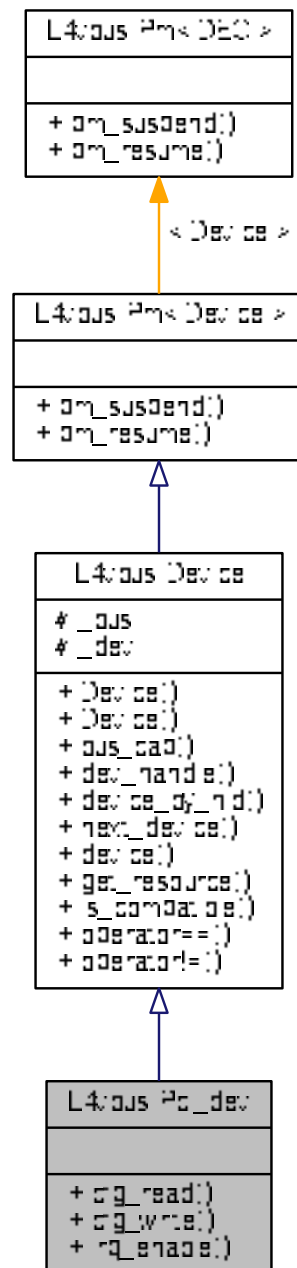
The documentation for this class was generated from the following file:

- l4/vbus/vbus

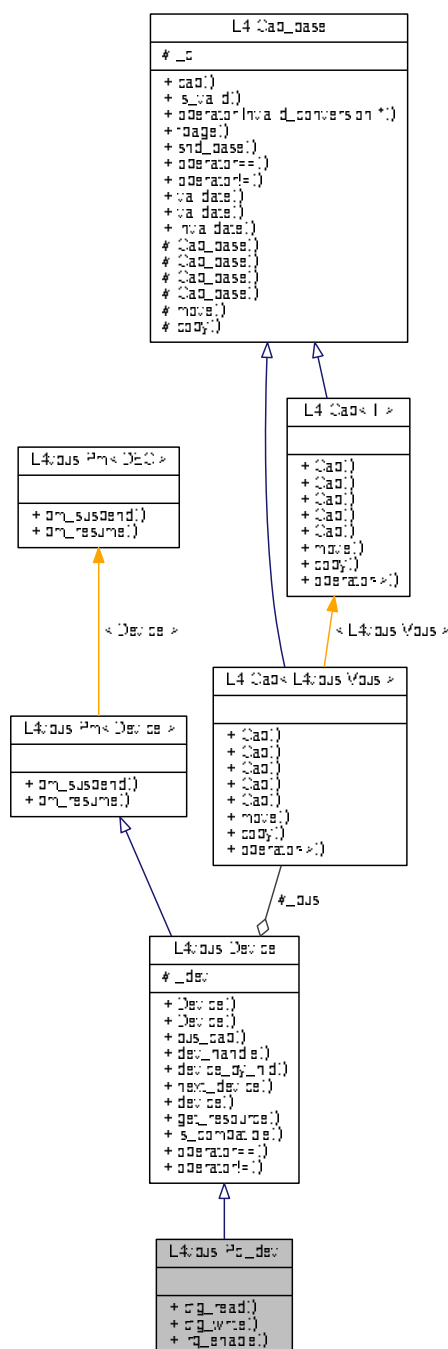
## 14.305 L4vbus::Pci\_dev Class Reference

A PCI device.

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.*

## Additional Inherited Members

### 14.305.1 Detailed Description

A PCI device.

Definition at line 87 of file [vbus\\_pci](#).

### 14.305.2 Member Function Documentation

#### 14.305.2.1 `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)  |

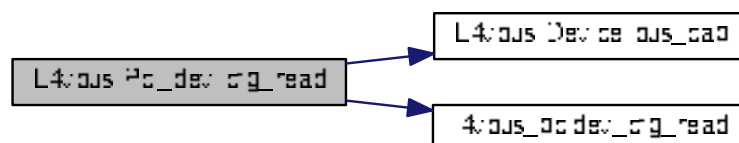
#### Returns

0 on success, else failure

Definition at line 99 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:





14.305.2.2 `cfg_write()`

```
int L4vbus::Pci_dev::cfg_write (
 l4_uint32_t reg,
 l4_uint32_t value,
 l4_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)  |

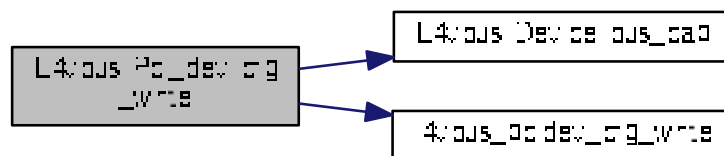
## Returns

0 on success, else failure

Definition at line 115 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:

14.305.2.3 `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  |

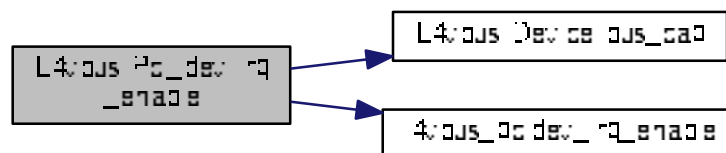
**Returns**

On success: Interrupt line to be used, else failure

Definition at line 131 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:



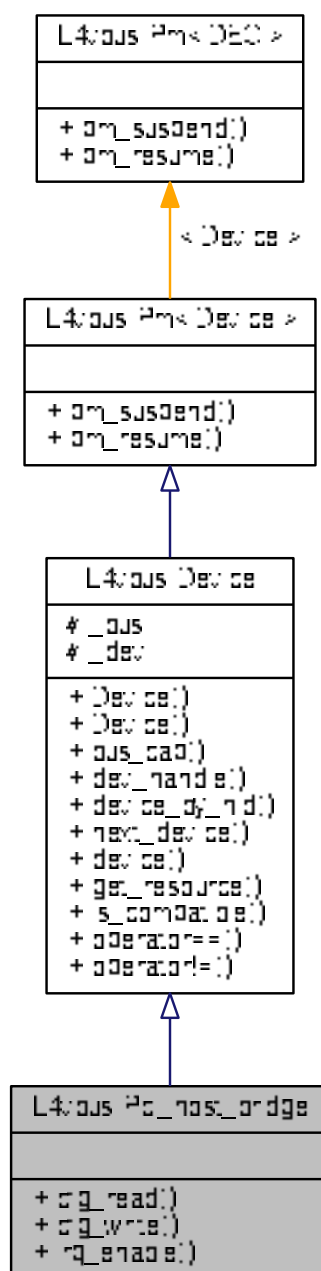
The documentation for this class was generated from the following file:

- `l4/vbus/vbus_pci`

## 14.306 L4vbus::Pci\_host\_bridge Class Reference

A Pci host bridge.

Inheritance diagram for L4vbus::Pci\_host\_bridge:





- Generated for L4Re by Doxygen

- 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.*

## Additional Inherited Members

### 14.306.1 Detailed Description

A Pci host bridge.

Definition at line 20 of file [vbus\\_pci](#).

### 14.306.2 Member Function Documentation

#### 14.306.2.1 [cfg\\_read\(\)](#)

```
int L4vbus::Pci_host_bridge::cfg_read (
 l4_uint32_t bus,
 l4_uint32_t devfn,
 l4_uint32_t reg,
 l4_uint32_t * value,
 l4_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> | <a href="#">Device</a> 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)                             |

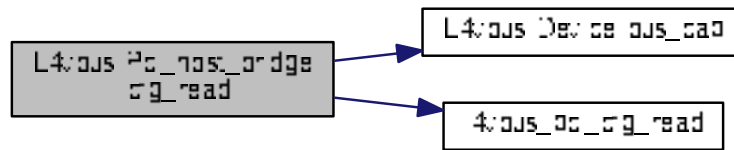
#### Returns

0 on success, else failure

Definition at line 34 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:



#### 14.306.2.2 `cfg_write()`

```

int L4vbus::Pci_host_bridge::cfg_write (
 l4_uint32_t bus,
 l4_uint32_t devfn,
 l4_uint32_t reg,
 l4_uint32_t value,
 l4_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> | <a href="#">Device</a> 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)                            |

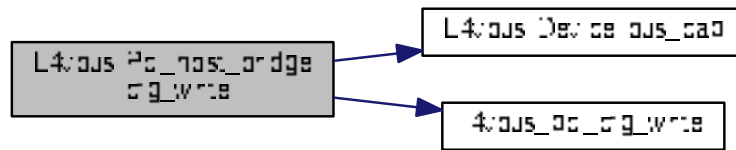
##### Returns

0 on success, else failure

Definition at line 53 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:



### 14.306.2.3 irq\_enable()

```

int L4vbus::Pci_host_bridge::irq_enable (
 l4_uint32_t bus,
 l4_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>    | <a href="#">Device</a> 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                                |

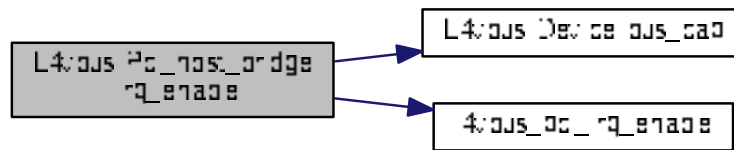
#### Returns

On success: Interrupt line to be used, else failure

Definition at line 74 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:



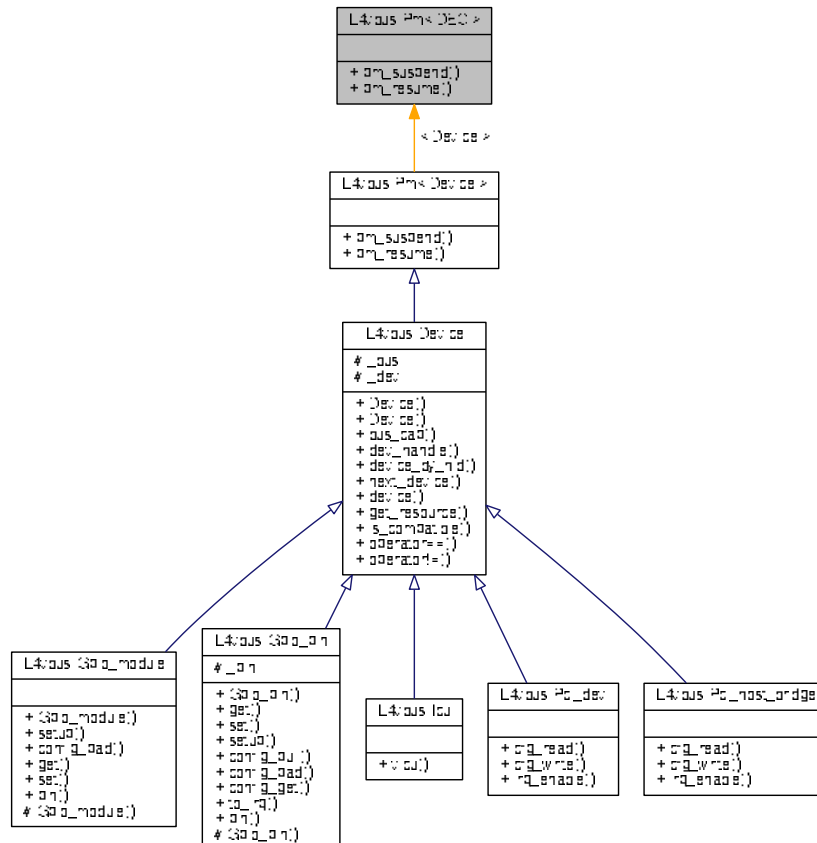
The documentation for this class was generated from the following file:

- l4/vbus/vbus\_pci

## 14.307 L4vbus::Pm< DEC > Class Template Reference

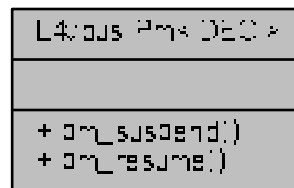
Power-management API mixin.

Inheritance diagram for L4vbus::Pm< DEC >:





Collaboration diagram for L4vbus::Pm< DEC >:



## Public Member Functions

- `int pm_suspend () const`  
*Suspend the module.*
- `int pm_resume () const`  
*Resume the module.*

### 14.307.1 Detailed Description

```
template<typename DEC>
class L4vbus::Pm< DEC >
```

Power-management API mixin.

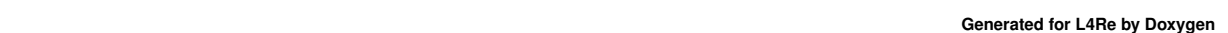
Definition at line 43 of file [vbus](#).

The documentation for this class was generated from the following file:

- `I4/vbus/vbus`

## 14.308 L4vbus::Vbus Class Reference

The virtual BUS.





- Generated for L4Re by Doxygen

## Additional Inherited Members

### 14.308.1 Detailed Description

The virtual BUS.

Definition at line 241 of file [vbus](#).

### 14.308.2 Member Function Documentation

#### 14.308.2.1 `release_resource()`

```
int L4vbus::Vbus::release_resource (
 l4vbus_resource_t * res) const [inline]
```

Release the given resource from the bus.

#### Parameters

|            |                                                    |
|------------|----------------------------------------------------|
| <i>res</i> | The resource that shall be requested from the bus. |
|------------|----------------------------------------------------|

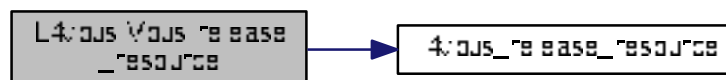
#### Returns

$\geq 0$  on success,  $< 0$  on error.

Definition at line 261 of file [vbus](#).

References [l4vbus\\_release\\_resource\(\)](#).

Here is the call graph for this function:



#### 14.308.2.2 `request_resource()`

```
int L4vbus::Vbus::request_resource (
 l4vbus_resource_t * res,
 int flags = 0) const [inline]
```

Request the given resource from the bus.

## Parameters

|              |                                                    |
|--------------|----------------------------------------------------|
| <i>res</i>   | The resource that shall be requested from the bus. |
| <i>flags</i> | The flags for the request.                         |

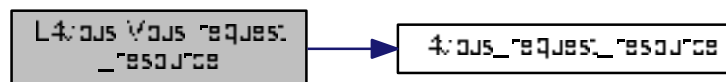
## Returns

>=0 on success, <0 on error.

Definition at line 251 of file [vbus](#).

References [l4vbus\\_request\\_resource\(\)](#).

Here is the call graph for this function:



## 14.308.2.3 root()

```
Device L4vbus::Vbus::root () const [inline]
```

Get the root device of the device tree of this bus.

## Returns

A V-BUS device representing the root of the device tree.

Definition at line 270 of file [vbus](#).

References [L4VBUS\\_ROOT\\_BUS](#).

The documentation for this class was generated from the following file:

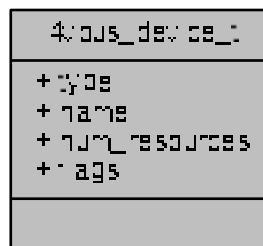
- `l4/vbus/vbus`

## 14.309 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](#).*

### 14.309.1 Detailed Description

Detailed information about a vbus device.

Definition at line 56 of file [vbus\\_types.h](#).

The documentation for this struct was generated from the following file:

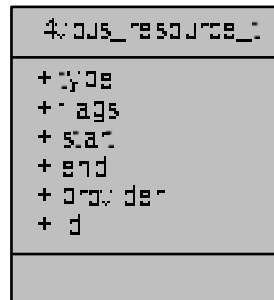
- `l4/vbus/vbus_types.h`

## 14.310 l4vbus\_resource\_t Struct Reference

Description of a single vbus resource.

```
#include <vbus_types.h>
```

Collaboration diagram for l4vbus\_resource\_t:



### 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.*

### 14.310.1 Detailed Description

Description of a single vbus resource.

Definition at line 23 of file [vbus\\_types.h](#).

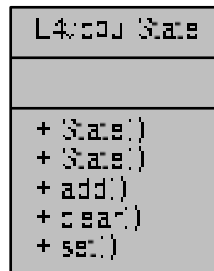
The documentation for this struct was generated from the following file:

- l4/vbus/[vbus\\_types.h](#)

## 14.311 L4vcpu::State Class Reference

C++ implementation of state word in the vCPU area.

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.*

### 14.311.1 Detailed Description

C++ implementation of state word in the vCPU area.

Definition at line 31 of file [vcpu](#).

### 14.311.2 Constructor & Destructor Documentation

#### 14.311.2.1 State()

```
L4vcpu::State::State (
 unsigned v) [inline], [explicit]
```

Initialize state.



## Parameters

|          |                |
|----------|----------------|
| <i>v</i> | Initial state. |
|----------|----------------|

Definition at line 41 of file [vcpu](#).

### 14.311.3 Member Function Documentation

#### 14.311.3.1 add()

```
void L4vcpu::State::add (
 unsigned bits) throw () [inline]
```

Add flags.

## Parameters

|             |                          |
|-------------|--------------------------|
| <i>bits</i> | Bits to add to the word. |
|-------------|--------------------------|

Definition at line 48 of file [vcpu](#).

#### 14.311.3.2 clear()

```
void L4vcpu::State::clear (
 unsigned bits) throw () [inline]
```

Clear flags.

## Parameters

|             |                            |
|-------------|----------------------------|
| <i>bits</i> | Bits to clear in the word. |
|-------------|----------------------------|

Definition at line 55 of file [vcpu](#).

#### 14.311.3.3 set()

```
void L4vcpu::State::set (
 unsigned v) throw () [inline]
```

Set flags.

**Parameters**

|                |                                 |
|----------------|---------------------------------|
| <code>v</code> | Set the word to the value of v. |
|----------------|---------------------------------|

Definition at line [62](#) of file [vcpu](#).

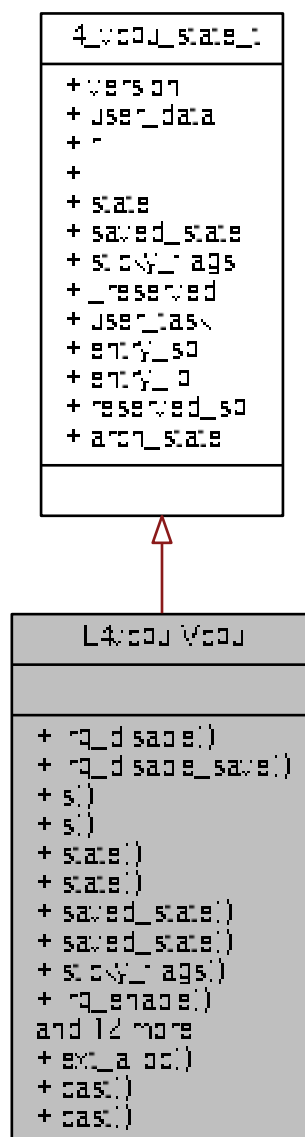
The documentation for this class was generated from the following file:

- `l4/vcpu/vcpu`

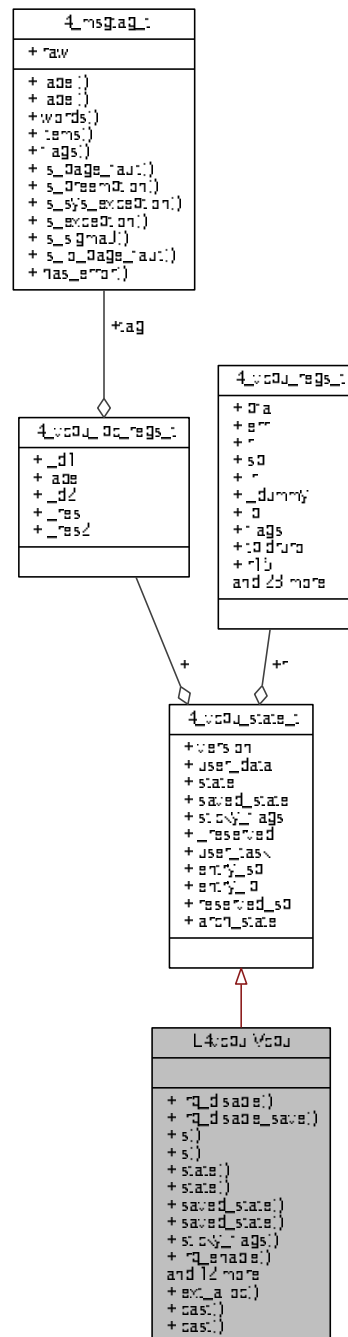
## 14.312 L4vcpu::Vcpu Class Reference

C++ implementation of the vCPU save state area.

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.*

### 14.312.1 Detailed Description

C++ implementation of the vCPU save state area.

Definition at line 72 of file [vcpu](#).

### 14.312.2 Member Function Documentation

#### 14.312.2.1 `cast()` [1/2]

```
static Vcpu* L4vcpu::Vcpu::cast (
 void * x) throw) [inline], [static]
```

Cast a void pointer to a class pointer.

##### Parameters

|                |          |
|----------------|----------|
| <code>x</code> | Pointer. |
|----------------|----------|

##### Returns

Pointer to [Vcpu](#) class.

Definition at line 265 of file [vcpu](#).

#### 14.312.2.2 `cast()` [2/2]

```
static Vcpu* L4vcpu::Vcpu::cast (
 l4_addr_t x) throw) [inline], [static]
```

Cast an address to a class pointer.

##### Parameters

|                |          |
|----------------|----------|
| <code>x</code> | Pointer. |
|----------------|----------|

##### Returns

Pointer to [Vcpu](#) class.

Definition at line 275 of file [vcpu](#).

## 14.312.2.3 entry\_ip()

```
void L4vcpu::Vcpu::entry_ip (
 l4_umword_t ip) [inline]
```

Set vCPU entry instruction pointer.

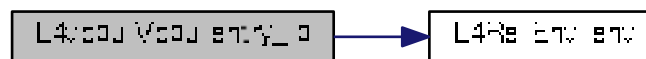
## Parameters

|           |                                     |
|-----------|-------------------------------------|
| <i>ip</i> | Instruction pointer address to set. |
|-----------|-------------------------------------|

Definition at line 239 of file [vcpu](#).

References [l4\\_vcpu\\_state\\_t::entry\\_ip](#), [L4Re::Env::env\(\)](#), and [L4\\_CV](#).

Here is the call graph for this function:



## 14.312.2.4 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 232 of file [vcpu](#).

References [l4\\_vcpu\\_state\\_t::entry\\_sp](#).

14.312.2.5 `ext_alloc()`

```
static 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.

## Return values

|                        |                                     |
|------------------------|-------------------------------------|
| <code>vcpu</code>      | Allocated vcpu-state area.          |
| <code>ext_state</code> | Allocated extended vcpu-state area. |

## Parameters

|                   |                                                                           |
|-------------------|---------------------------------------------------------------------------|
| <code>task</code> | Task to use for allocation, defaults to own task.                         |
| <code>rm</code>   | Region manager to use for allocation defaults to standard region manager. |

## Returns

0 for success, error code otherwise

14.312.2.6 `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 216 of file `vcpu`.

References `l4_vcpu_state_t::i`.

14.312.2.7 `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 223 of file `vcpu`.

References `l4_vcpu_state_t::i`.



## 14.312.2.8 irq\_disable\_save()

```
unsigned L4vcpu::Vcpu::irq_disable_save () throw () [inline]
```

Disable the vCPU for event delivery and return previous state.

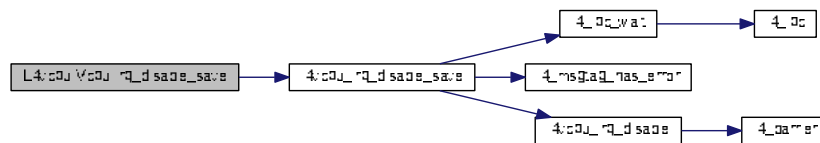
## Returns

IRQ state before disabling IRQs.

Definition at line 85 of file [vcpu](#).

References [l4vcpu\\_irq\\_disable\\_save\(\)](#).

Here is the call graph for this function:



## 14.312.2.9 irq\_enable()

```
void L4vcpu::Vcpu::irq_enable (
 l4_utcb_t * utcb,
 l4vcpu_event_hdl_t do_event_work_cb,
 l4vcpu_setup_ipc_t setup_ipc) throw () [inline]
```

Enable the vCPU for event delivery.

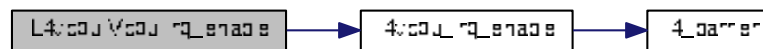
## Parameters

|                         |                                                                                                            |
|-------------------------|------------------------------------------------------------------------------------------------------------|
| <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 142 of file [vcpu](#).

References [l4vcpu\\_irq\\_enable\(\)](#).

Here is the call graph for this function:



#### 14.312.2.10 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.

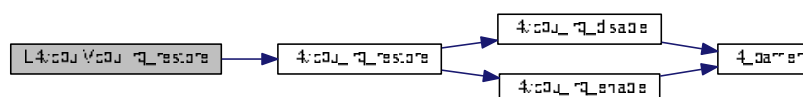
##### Parameters

|                         |                                                                                                            |
|-------------------------|------------------------------------------------------------------------------------------------------------|
| <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 157 of file [vcpu](#).

References [l4vcpu\\_irq\\_restore\(\)](#).

Here is the call graph for this function:



#### 14.312.2.11 is\_irq\_entry()

```

int L4vcpu::Vcpu::is_irq_entry () const [inline]

```

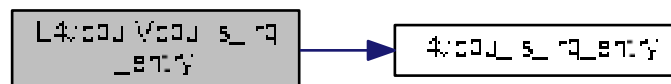
Return whether the entry reason was an IRQ/IPC message.

return 0 if not, !=0 otherwise.

Definition at line 195 of file [vcpu](#).

References [l4vcpu\\_is\\_irq\\_entry\(\)](#).

Here is the call graph for this function:



#### 14.312.2.12 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 188 of file [vcpu](#).

References [l4vcpu\\_is\\_page\\_fault\\_entry\(\)](#).

Here is the call graph for this function:



**14.312.2.13** `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 202 of file `vcpu`.

References `l4_vcpu_state_t::r`.

**14.312.2.14** `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 209 of file `vcpu`.

References `l4_vcpu_state_t::r`.

**14.312.2.15** `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 113 of file `vcpu`.

**14.312.2.16 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 123 of file [vcpu](#).

References [l4\\_vcpu\\_state\\_t::saved\\_state](#).

**14.312.2.17 state()** [1/2]

```
State* L4vcpu::Vcpu::state () throw () [inline]
```

Get state word.

**Returns**

Pointer to state word in the vCPU

Definition at line 95 of file [vcpu](#).

**14.312.2.18 state()** [2/2]

```
State L4vcpu::Vcpu::state () const throw () [inline]
```

Get state word.

**Returns**

Pointer to state word in the vCPU

Definition at line 106 of file [vcpu](#).

References [l4\\_vcpu\\_state\\_t::state](#).

**14.312.2.19 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 181 of file [vcpu](#).

## 14.312.2.20 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

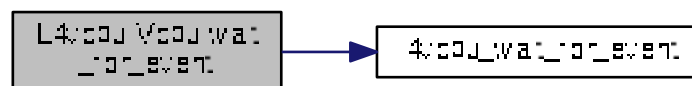
|                         |                                                                                       |
|-------------------------|---------------------------------------------------------------------------------------|
| <i>utcb</i>             | The UTCB to use.                                                                      |
| <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 173 of file [vcpu](#).

References [l4vcpu\\_wait\\_for\\_event\(\)](#).

Here is the call graph for this function:



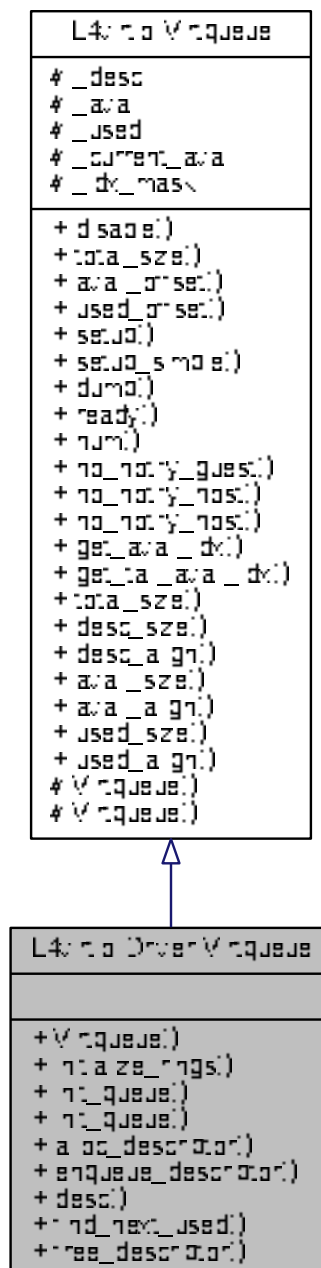
The documentation for this class was generated from the following file:

- `l4/vcpu/vcpu`

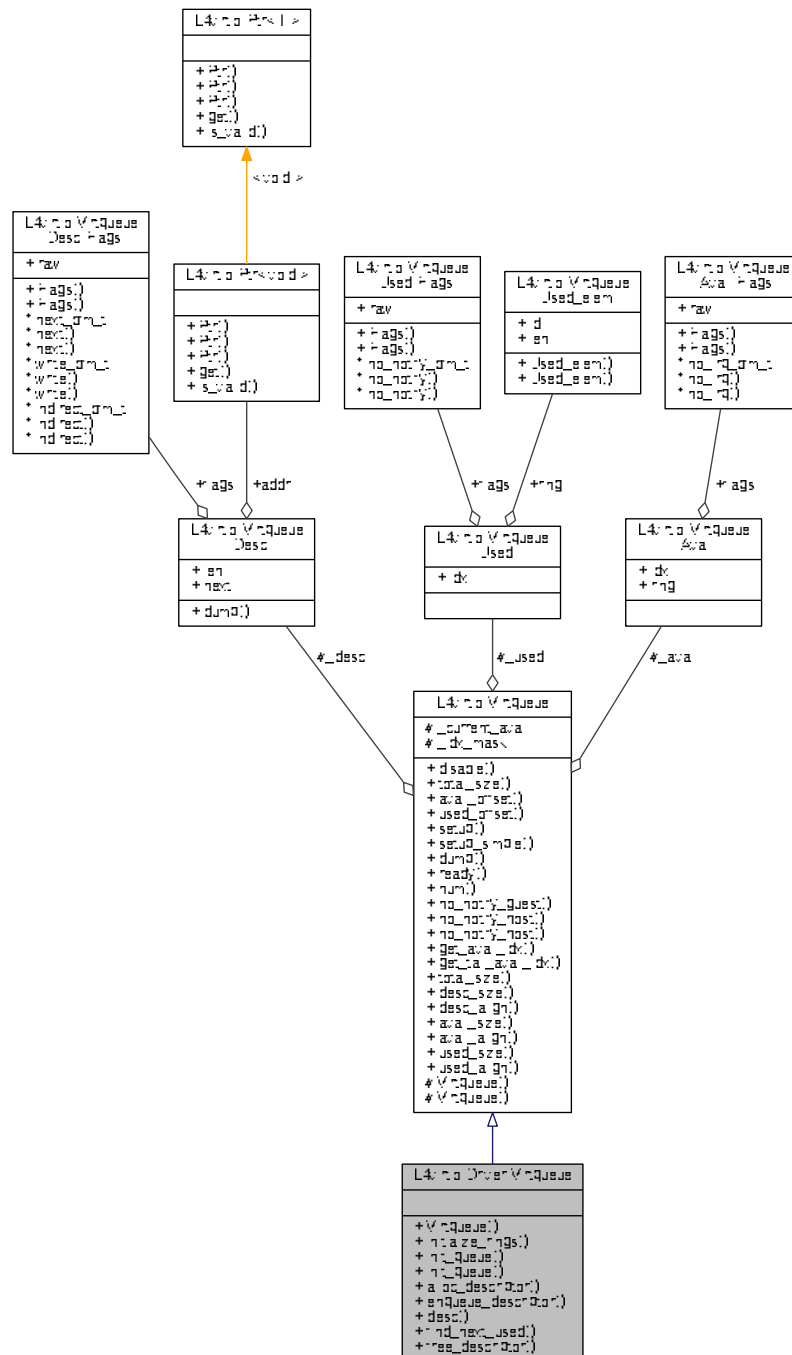
## 14.313 L4virtio::Driver::Virtqueue Class Reference

Driver-side implementation of a [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.*

## Additional Inherited Members

### 14.313.1 Detailed Description

Driver-side implementation of a [Virtqueue](#).

Adds function for managing the descriptor list, enqueueing new and dequeueing finished requests.

Definition at line 471 of file [virtqueue](#).

### 14.313.2 Member Function Documentation

#### 14.313.2.1 [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 559 of file [virtqueue](#).

#### 14.313.2.2 [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 593 of file [virtqueue](#).

#### 14.313.2.3 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 576 of file [virtqueue](#).

#### 14.313.2.4 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::Eoq if no used element is currently available.

Definition at line 612 of file [virtqueue](#).

#### 14.313.2.5 free\_descriptor()

```
void L4virtio::Driver::Virtqueue::free_descriptor (
 14_uint16_t head,
 14_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 635 of file [virtqueue](#).

14.313.2.6 `init_queue()` [1/2]

```
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 523 of file [virtqueue](#).

14.313.2.7 `init_queue()` [2/2]

```
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 538 of file [virtqueue](#).

#### 14.313.2.8 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 495 of file [virtqueue](#).

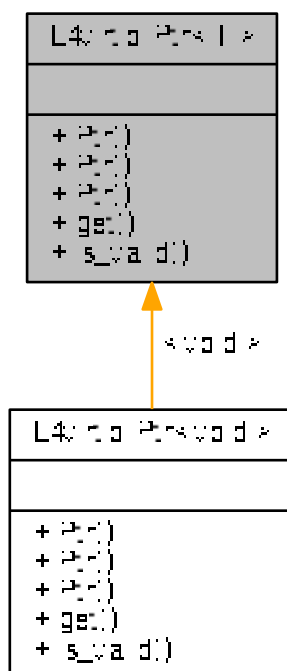
The documentation for this class was generated from the following file:

- l4/l4virtio/virtqueue

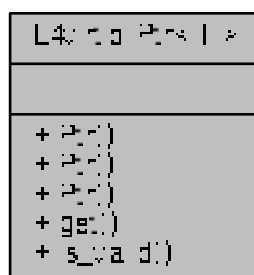
## 14.314 L4virtio::Ptr< T > Class Template Reference

Pointer used in virtio descriptors.

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

### 14.314.1 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 [56](#) of file [virtqueue](#).

### 14.314.2 Member Enumeration Documentation

#### 14.314.2.1 Invalid\_type

```
template<typename T>
enum L4virtio::Ptr::Invalid_type
```

Type for making an invalid (NULL) [Ptr](#).

Enumerator

|         |                                                    |
|---------|----------------------------------------------------|
| Invalid | Use to set a <a href="#">Ptr</a> to invalid (NULL) |
|---------|----------------------------------------------------|

Definition at line [60](#) of file [virtqueue](#).

### 14.314.3 Member Function Documentation

#### 14.314.3.1 get()

```
template<typename T>
l4_uint64_t L4virtio::Ptr< T >::get () const [inline]
```

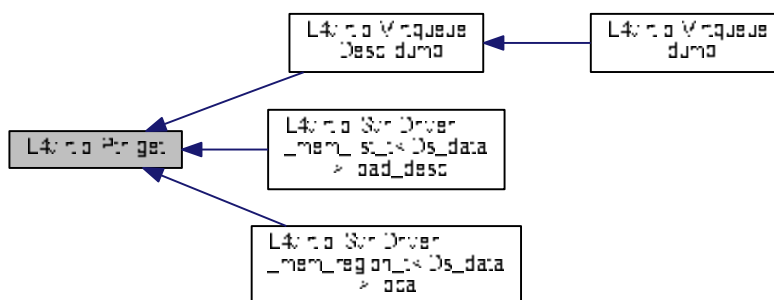
**Returns**

The raw 64bit address of the pointer.

Definition at line 71 of file [virtqueue](#).

Referenced by [L4virtio::Virtqueue::Desc::dump\(\)](#), [L4virtio::Svr::Driver\\_mem\\_list\\_t< Ds\\_data >::load\\_desc\(\)](#), and [L4virtio::Svr::Driver\\_mem\\_region\\_t< Ds\\_data >::local\(\)](#).

Here is the caller graph for this function:

**14.314.3.2 is\_valid()**

```
template<typename T>
bool L4virtio::Ptr< T >::is_valid () const [inline]
```

**Returns**

true if the pointer is invalid (NULL).

Definition at line 74 of file [virtqueue](#).

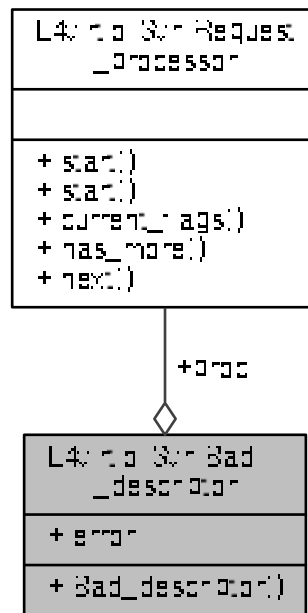
The documentation for this class was generated from the following file:

- `I4/I4virtio/virtqueue`

## 14.315 L4virtio::Svr::Bad\_descriptor Struct Reference

Exception used by Queue to indicate descriptor errors.

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.*

### Data Fields

- [Request\\_processor](#) const \* [proc](#)

*The processor that triggered the exception.*



### 14.315.1 Detailed Description

Exception used by Queue to indicate descriptor errors.

Definition at line 329 of file [virtio](#).

### 14.315.2 Member Enumeration Documentation

#### 14.315.2.1 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 332 of file [virtio](#).

### 14.315.3 Constructor & Destructor Documentation

#### 14.315.3.1 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 352 of file [virtio](#).

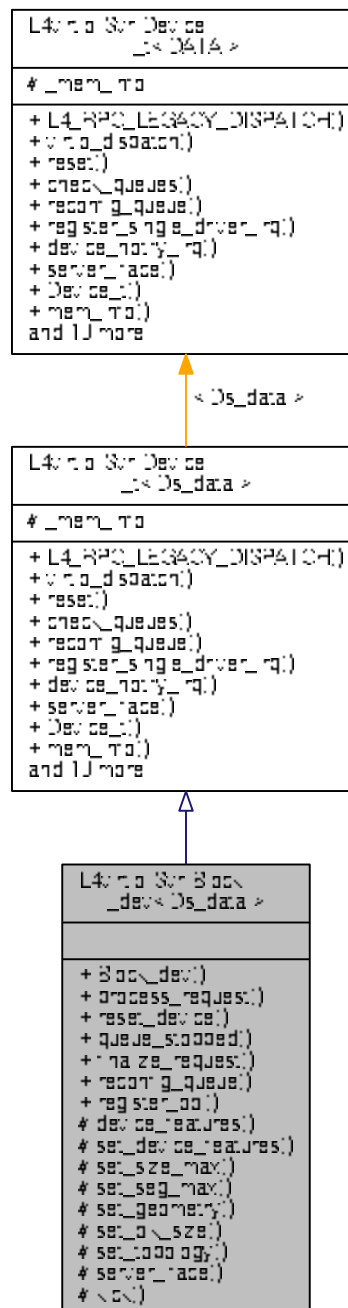
The documentation for this struct was generated from the following file:

- l4/l4virtio/server/virtio

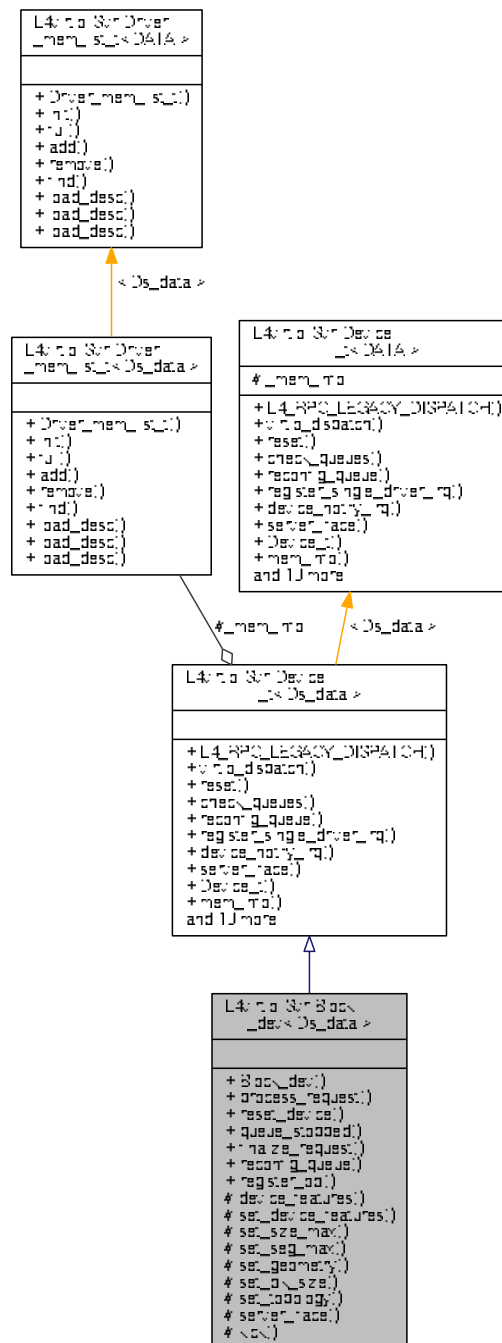
## 14.316 L4virtio::Svr::Block\_dev< Ds\_data > Class Template Reference

Base class for virtio block devices.

Inheritance diagram for L4virtio::Svr::Block\_dev< Ds\_data >:



Collaboration diagram for L4virtio::Svr::Block\_dev< Ds\_data >:



## Public Member Functions

- **Block\_dev** (`l4_uint32_t` vendor, unsigned `queue_size`, `l4_uint64_t` capacity, bool `read_only`)  
Create a new virtio block device.
- virtual bool **process\_request** (`cx::unique_ptr< Request >` &&req)=0  
Implements the actual processing of data in the 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)  
*callback for client queue-config request*
- `L4::Cap`< void > `register_obj` (`L4::Registry_iface` \*registry, char const \*service=0)  
*Attach device to an object registry.*

## 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.*

## Additional Inherited Members

### 14.316.1 Detailed Description

```
template<typename Ds_data>
class L4virtio::Svr::Block_dev< Ds_data >
```

Base class for virtio block devices.

Use this class as a base to implement your own specific block device.

Definition at line 22 of file `virtio-block`.

### 14.316.2 Constructor & Destructor Documentation

#### 14.316.2.1 Block\_dev()

```
template<typename Ds_data>
L4virtio::Svr::Block_dev< Ds_data >::Block_dev (
 l4_uint32_t vendor,
 unsigned queue_size,
 l4_uint64_t capacity,
 bool read_only) [inline]
```

Create a new virtio block device.

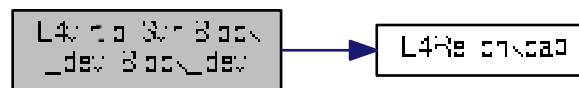
## Parameters

|                   |                                                       |
|-------------------|-------------------------------------------------------|
| <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 377 of file [virtio-block](#).

References [L4Re::Util::cap\\_alloc](#), and [L4Re::chkcap\(\)](#).

Here is the call graph for this function:



### 14.316.3 Member Function Documentation

#### 14.316.3.1 finalize\_request()

```

template<typename Ds_data>
void L4virtio::Svr::Block_dev< 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 <a href="#">L4virtio_block_status</a> ). |

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 433 of file [virtio-block](#).

### 14.316.3.2 process\_request()

```
template<typename Ds_data>
virtual bool L4virtio::Svr::Block_dev< Ds_data >::process_request (
 cxx::unique_ptr< Request > && req) [pure virtual]
```

Implements the actual processing of data in the device.

#### Parameters

|            |                              |
|------------|------------------------------|
| <i>req</i> | The request to be processed. |
|------------|------------------------------|

#### Returns

If false, no further requests will be scheduled.

Synchronous and asynchronous processing of the data is supported. For asynchronous mode, the function should set up the worker and then return false. In synchronous mode, the function should return true, once processing is complete. If there is an error and processing is aborted, the status flag of needs to be set accordingly and the request immediately finished with finish\_request() if the client is to be answered.

### 14.316.3.3 register\_obj()

```
template<typename Ds_data>
L4::Cap<void> L4virtio::Svr::Block_dev< Ds_data >::register_obj (
 L4::Registry_iface * registry,
 char const * service = 0) [inline]
```

Attach device to an object registry.

#### Parameters

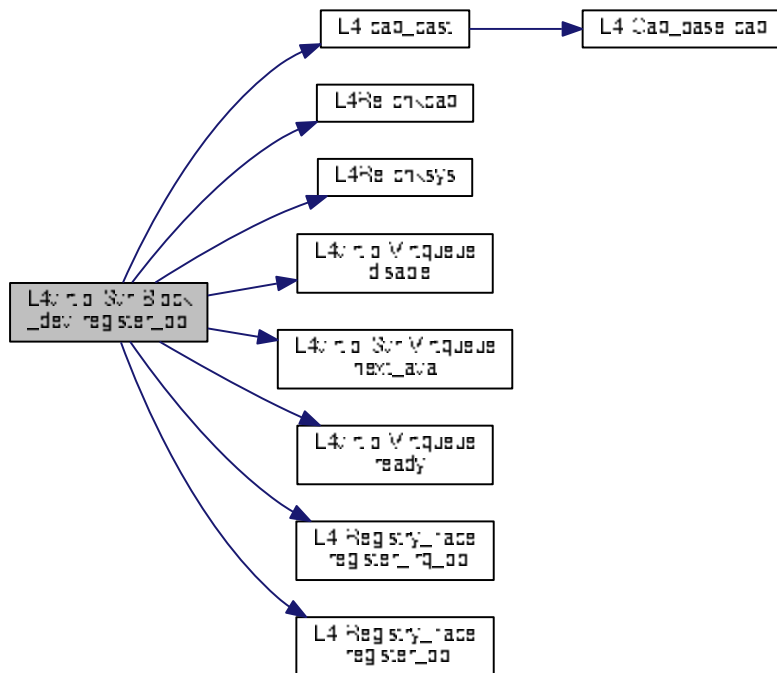
|                 |                                                                    |
|-----------------|--------------------------------------------------------------------|
| <i>registry</i> | Object registry that will be responsible for dispatching requests. |
| <i>service</i>  | Name of an existing capability the device should use.              |

This functions registers the general virtio interface as well as the interrupt handler which is used for receiving client notifications.

Definition at line 467 of file [virtio-block](#).

References [L4::cap\\_cast\(\)](#), [L4Re::chkcapi\(\)](#), [L4Re::chksys\(\)](#), [L4virtio::Virtqueue::disable\(\)](#), [L4virtio::Svr::Virtqueue::next\\_avail\(\)](#), [L4virtio::Virtqueue::ready\(\)](#), [L4::Registry\\_iface::register\\_irq\\_obj\(\)](#), and [L4::Registry\\_iface::register\\_obj\(\)](#).

Here is the call graph for this function:



#### 14.316.3.4 set\_blk\_size()

```
template<typename Ds_data>
void L4virtio::Svr::Block_dev< 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 336 of file [virtio-block](#).

#### 14.316.3.5 set\_size\_max()

```
template<typename Ds_data>
void L4virtio::Svr::Block_dev< Ds_data >::set_size_max (
 l4_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 294 of file [virtio-block](#).

### 14.316.3.6 set\_topology()

```
template<typename Ds_data>
void L4virtio::Svr::Block_dev< 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 352 of file [virtio-block](#).

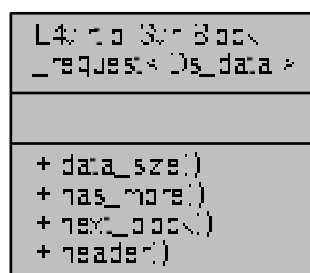
The documentation for this class was generated from the following file:

- l4/l4virtio/server/virtio-block

## 14.317 L4virtio::Svr::Block\_request< Ds\_data > Class Template Reference

A request to read or write data.

Collaboration diagram for L4virtio::Svr::Block\_request< Ds\_data >:





## 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.*

### 14.317.1 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](#).

### 14.317.2 Member Function Documentation

#### 14.317.2.1 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

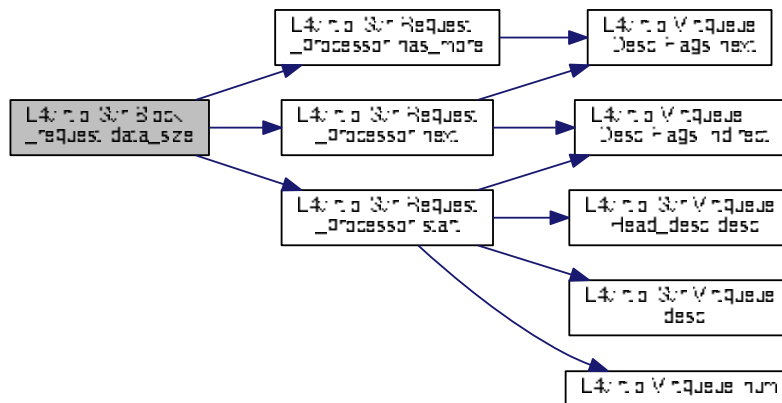
|                                            |                           |
|--------------------------------------------|---------------------------|
| <a href="#">L4::Runtime_error(-L4_EIO)</a> | Request has a bad format. |
|--------------------------------------------|---------------------------|

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:



#### 14.317.2.2 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

|                                          |                                  |
|------------------------------------------|----------------------------------|
| <a href="#"><i>L4::Runtime_error</i></a> | No more data block is available. |
| <a href="#"><i>Bad_descriptor</i></a>    | Virtio request is corrupted.     |

Definition at line 113 of file [virtio-block](#).

References [L4virtio::Svr::Bad\\_descriptor::Bad\\_size](#), and [L4\\_EEXIST](#).

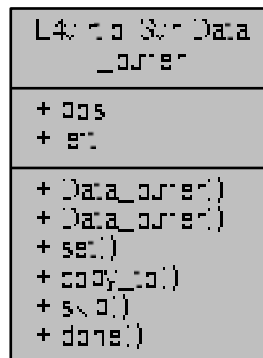
The documentation for this class was generated from the following file:

- [l4/l4virtio/server/virtio-block](#)

## 14.318 L4virtio::Svr::Data\_buffer Struct Reference

Abstract 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)  
*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.*

### 14.318.1 Detailed Description

Abstract data buffer.

Definition at line 245 of file [virtio](#).

## 14.318.2 Constructor & Destructor Documentation

### 14.318.2.1 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 [261](#) of file [virtio](#).

## 14.318.3 Member Function Documentation

### 14.318.3.1 copy\_to()

```
l4_uint32_t L4virtio::Svr::Data_buffer::copy_to (
 Data_buffer * dst) [inline]
```

Copy contents from this buffer to the destination buffer.

#### Parameters

|            |                     |
|------------|---------------------|
| <i>dst</i> | Destination buffer. |
|------------|---------------------|

#### Returns

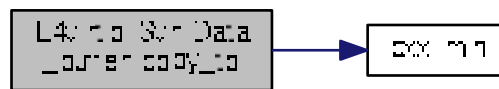
the number of bytes copied.

This function copies the maximum number of bytes from this to *dst*.

Definition at line [288](#) of file [virtio](#).

References [left](#), [cxx::min\(\)](#), and [pos](#).

Here is the call graph for this function:



#### 14.318.3.2 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 [320](#) of file [virtio](#).

#### 14.318.3.3 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 [274](#) of file [virtio](#).

#### 14.318.3.4 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

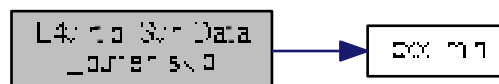
The number of bytes skipped.

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 308 of file [virtio](#).

References [cxx::min\(\)](#).

Here is the call graph for this function:



The documentation for this struct was generated from the following file:

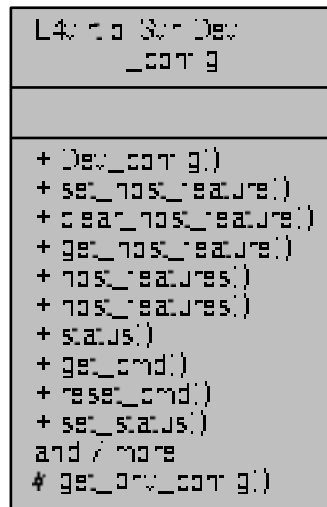
- `I4/I4virtio/server/virtio`

## 14.319 L4virtio::Svr::Dev\_config Class Reference

Abstraction for L4-Virtio device config memory.

Inherited by `L4virtio::Svr::Dev_config_t< l4virtio_block_config_t >`, and `L4virtio::Svr::Dev_config_t< PRIV_CONFIG >`.

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)  
*Setup/Create a L4-Virtio config data space.*
- [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 [set\\_failed](#) ()  
*Set device status failed bit.*
- 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) 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.*

### 14.319.1 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 55 of file [l4virtio](#).

### 14.319.2 Constructor & Destructor Documentation

#### 14.319.2.1 Dev\_config()

```
L4virtio::Svr::Dev_config::Dev_config (
 l4_uint32_t vendor,
 l4_uint32_t device,
 unsigned cfg_size,
 l4_uint32_t num_queues = 0) [inline]
```

Setup/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.

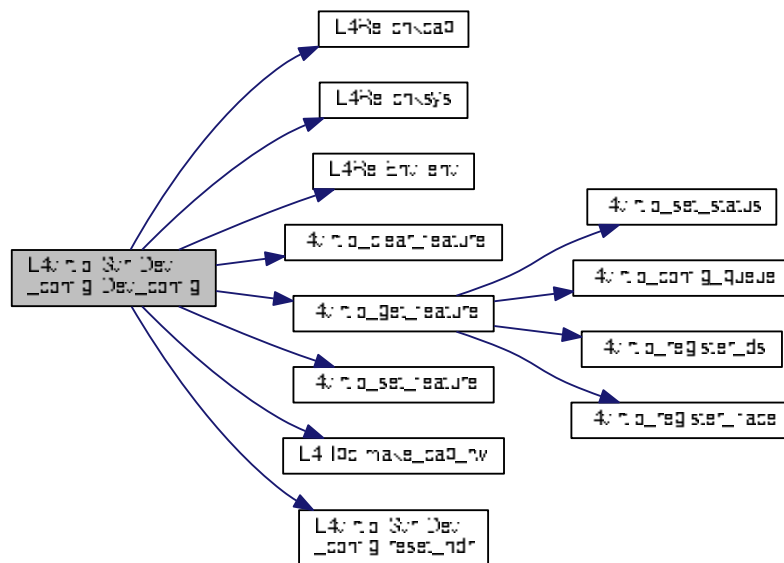
Definition at line 95 of file [l4virtio](#).

References [L4Re::Util::cap\\_alloc](#), [L4Re::chkcap\(\)](#), [L4Re::chksys\(\)](#), [L4Re::Env::env\(\)](#), [L4\\_PAGESIZE](#), [l4virtio\\_↵clear\\_feature\(\)](#), [l4virtio\\_get\\_feature\(\)](#), [l4virtio\\_set\\_feature\(\)](#), [L4::lpc::make\\_cap\\_rw\(\)](#), [reset\\_hdr\(\)](#), and [L4Re::Rm↵::Search\\_addr](#).

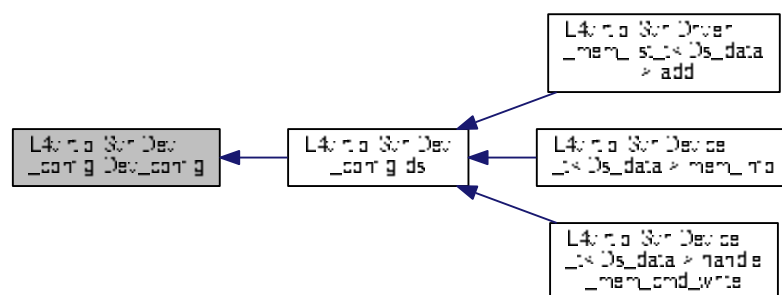
Referenced by [ds\(\)](#).



Here is the call graph for this function:



Here is the caller graph for this function:



### 14.319.3 Member Function Documentation

#### 14.319.3.1 change\_queue\_config()

```
bool L4virtio::Svr::Dev_config::change_queue_config (
 uint32_t num_queues) [inline]
```

Setup new queue configuration.

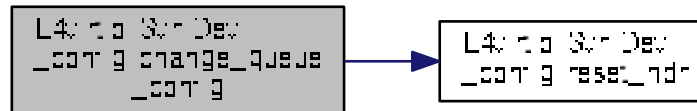
## Parameters

|                         |                                              |
|-------------------------|----------------------------------------------|
| <code>num_queues</code> | The number of queues provided by the device. |
|-------------------------|----------------------------------------------|

Definition at line 203 of file [l4virtio](#).

References [L4\\_PAGESIZE](#), and [reset\\_hdr\(\)](#).

Here is the call graph for this function:



## 14.319.3.2 ds()

```
L4::Cap<L4Re::Dataspace> L4virtio::Svr::Dev_config::ds () const [inline]
```

Get data-space capability for the shared config data space.

## Returns

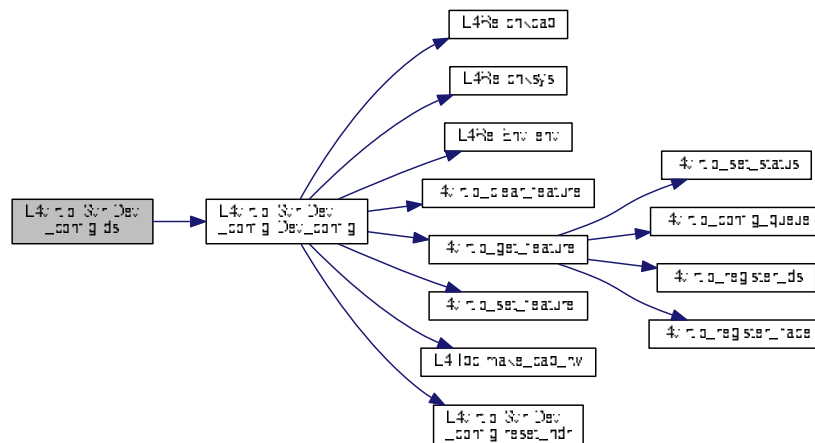
Capability for the shared config data space.

Definition at line 292 of file [l4virtio](#).

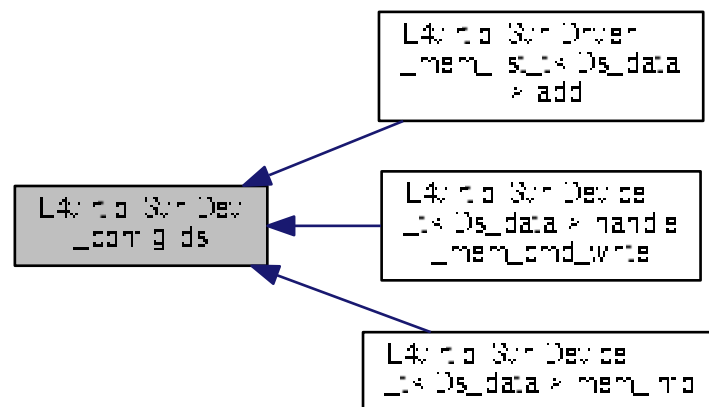
References [Dev\\_config\(\)](#).

Referenced by [L4virtio::Svr::Driver\\_mem\\_list\\_t<Ds\\_data>::add\(\)](#), [L4virtio::Svr::Device\\_t<Ds\\_data>::handle\\_cmd\\_write\(\)](#), and [L4virtio::Svr::Device\\_t<Ds\\_data>::mem\\_info\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 14.319.3.3 get\_cmd()

```
14_uint32_t L4virtio::Svr::Dev_config::get_cmd () const [inline]
```

Get the value from the `cmd` register.

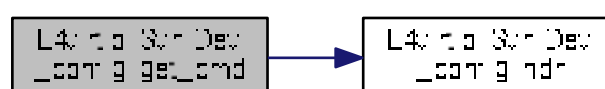
Note, the most significant eight bits are the command (0 is nothing to do). The upper eight bit are reset to zero after the command was handled.

Definition at line 158 of file [l4virtio](#).

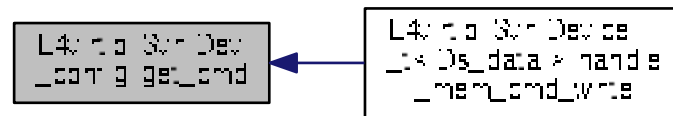
References [l4virtio\\_config\\_hdr\\_t::cmd](#), and [hdr\(\)](#).

Referenced by [L4virtio::Svr::Device\\_t<Ds\\_data>::handle\\_mem\\_cmd\\_write\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 14.319.3.4 `hdr()`

```
l4virtio_config_hdr_t const volatile* L4virtio::Svr::Dev_config::hdr () const [inline]
```

Get a read-only pointer to the config header.

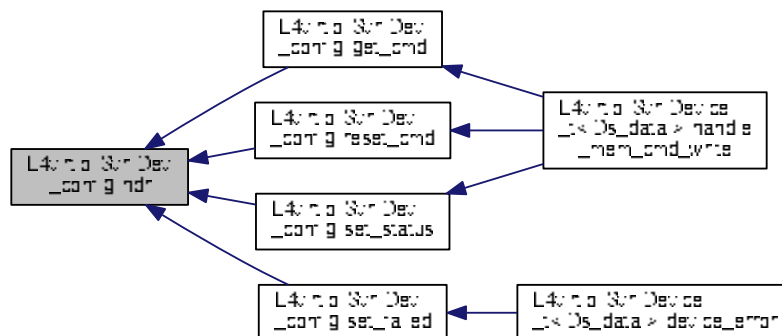
##### Returns

Read-only pointer to the shared config header.

Definition at line 285 of file `l4virtio`.

Referenced by `get_cmd()`, `reset_cmd()`, `set_failed()`, and `set_status()`.

Here is the caller graph for this function:



#### 14.319.3.5 `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

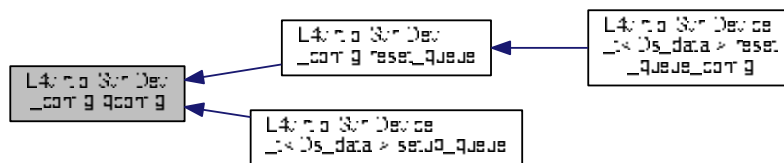
Read-only pointer to the config of the queue with the given *index*, or NULL if *index* is out of range.

Definition at line 220 of file [l4virtio](#).

References [L4\\_UNLIKELY](#).

Referenced by [reset\\_queue\(\)](#), and [L4virtio::Svr::Device\\_t<Ds\\_data>::setup\\_queue\(\)](#).

Here is the caller graph for this function:



## 14.319.3.6 reset\_cmd()

```
void L4virtio::Svr::Dev_config::reset_cmd () [inline]
```

Reset the `cmd` register after execution of a command.

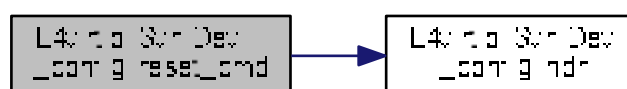
This function resets the `cmd` register in order for the client to detect that the command was executed by the device.

Definition at line 169 of file [l4virtio](#).

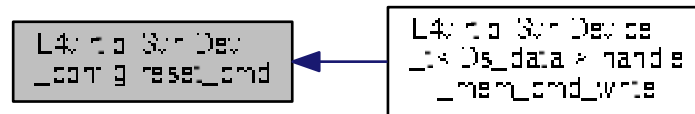
References [l4virtio\\_config\\_hdr\\_t::cmd](#), and [hdr\(\)](#).

Referenced by [L4virtio::Svr::Device\\_t<Ds\\_data>::handle\\_mem\\_cmd\\_write\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 14.319.3.7 reset\_queue()

```
bool L4virtio::Svr::Dev_config::reset_queue (
 unsigned index,
 unsigned num_max,
 bool inc_generation = false) 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 descriptor supported by this queue.    |
| <i>inc_generation</i> | The config generation will be incremented when this is true. |

##### Returns

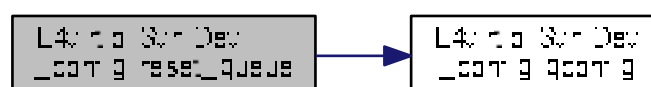
true on success, or false when *index* is out of range.

Definition at line 262 of file [l4virtio](#).

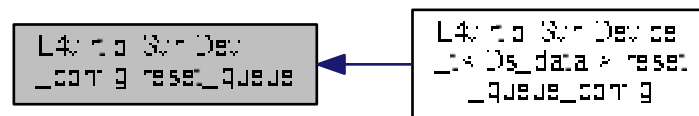
References [L4\\_UNLIKELY](#), [l4virtio\\_config\\_queue\\_t::num](#), [l4virtio\\_config\\_queue\\_t::num\\_max](#), [qconfig\(\)](#), and [l4virtio\\_config\\_queue\\_t::ready](#).

Referenced by [L4virtio::Svr::Device\\_t<Ds\\_data>::reset\\_queue\\_config\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 14.319.3.8 set\_failed()

```
void L4virtio::Svr::Dev_config::set_failed () [inline]
```

Set device status failed bit.

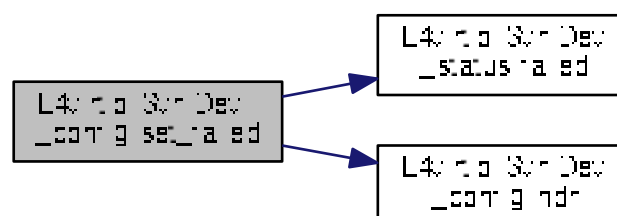
This function sets the internal status register and also the status register in the shared memory to *status*.

Definition at line 193 of file [l4virtio](#).

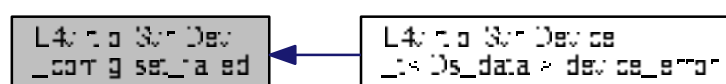
References [L4virtio::Svr::Dev\\_status::failed\(\)](#), [hdr\(\)](#), [L4virtio::Svr::Dev\\_status::raw](#), and [l4virtio\\_config\\_hdr\\_t::status](#).

Referenced by [L4virtio::Svr::Device\\_t<Ds\\_data>::device\\_error\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



14.319.3.9 `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. |
|---------------|-----------------------------------------------|

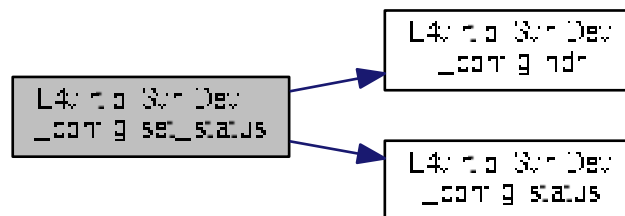
This function sets the internal status register and also the status register in the shared memory to *status*.

Definition at line 181 of file `l4virtio`.

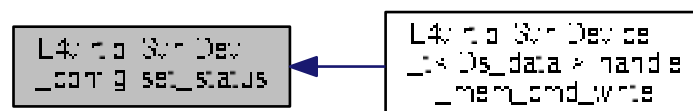
References `hdr()`, `L4virtio::Svr::Dev_status::raw`, `status()`, and `l4virtio_config_hdr_t::status`.

Referenced by `L4virtio::Svr::Device_t<Ds_data>::handle_mem_cmd_write()`.

Here is the call graph for this function:



Here is the caller graph for this function:





## 14.319.3.10 status()

```
Status L4virtio::Svr::Dev_config::status () const [inline]
```

Get current device status (trusted).

## Returns

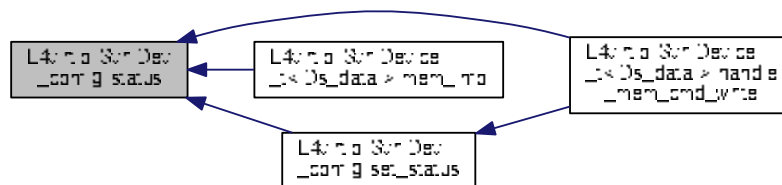
Current device status register (trusted).

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 150 of file [l4virtio](#).

Referenced by [L4virtio::Svr::Device\\_t< Ds\\_data >::handle\\_mem\\_cmd\\_write\(\)](#), [L4virtio::Svr::Device\\_t< Ds\\_data >::mem\\_info\(\)](#), and [set\\_status\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

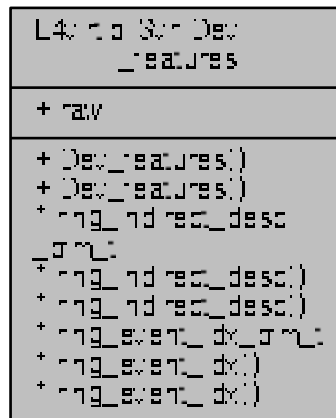
- `l4/l4virtio/server/l4virtio`

## 14.320 L4virtio::Svr::Dev\_features Struct Reference

Type for device feature bitmap.

Inherited by `L4virtio::Svr::Block_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 .*
- [ring\\_indirect\\_desc\\_bfm\\_t::Val](#) [ring\\_indirect\\_desc](#) () const  
*Get the ring\_indirect\_desc bits ( 28 to 28 ) of raw .*
- [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 .*
- [ring\\_event\\_idx\\_bfm\\_t::Val](#) [ring\\_event\\_idx](#) () const  
*Get the ring\_event\_idx bits ( 29 to 29 ) of raw .*
- [ring\\_event\\_idx\\_bfm\\_t::Ref](#) [ring\\_event\\_idx](#) ()  
*Get a reference to the ring\_event\_idx bits ( 29 to 29 ) of raw .*

## 14.320.1 Detailed Description

Type for device feature bitmap.

Definition at line 74 of file [virtio](#).

## 14.320.2 Member Typedef Documentation

### 14.320.2.1 ring\_event\_idx\_bfm\_t

```
typedef cxx::Bitfield<decltype(raw), 29 , 29 > L4virtio::Svr::Dev_features::ring_event_idx↵_bfm_t
```

Type to access the *ring\_event\_idx* bits ( 29 to 29 ) of *raw* .

Definition at line 82 of file [virtio](#).

### 14.320.2.2 ring\_indirect\_desc\_bfm\_t

```
typedef cxx::Bitfield<decltype(raw), 28 , 28 > L4virtio::Svr::Dev_features::ring_indirect_↵desc_bfm_t
```

Type to access the *ring\_indirect\_desc* bits ( 28 to 28 ) of *raw* .

Definition at line 82 of file [virtio](#).

## 14.320.3 Member Function Documentation

### 14.320.3.1 ring\_event\_idx() [1/2]

```
ring_event_idx_bfm_t::Val L4virtio::Svr::Dev_features::ring_event_idx () const [inline]
```

Get the *ring\_event\_idx* bits ( 29 to 29 ) of *raw* .

Definition at line 83 of file [virtio](#).

### 14.320.3.2 ring\_event\_idx() [2/2]

```
ring_event_idx_bfm_t::Ref L4virtio::Svr::Dev_features::ring_event_idx () [inline]
```

Get a reference to the *ring\_event\_idx* bits ( 29 to 29 ) of *raw* .

Definition at line 83 of file [virtio](#).



## 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 .*
- [acked\\_bfm\\_t::Val](#) [acked](#) () const  
*Get the acked bits ( 0 to 0 ) of raw .*
- [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 .*
- [driver\\_bfm\\_t::Val](#) [driver](#) () const  
*Get the driver bits ( 1 to 1 ) of raw .*
- [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 .*
- [driver\\_ok\\_bfm\\_t::Val](#) [driver\\_ok](#) () const  
*Get the driver\_ok bits ( 2 to 2 ) of raw .*
- [driver\\_ok\\_bfm\\_t::Ref](#) [driver\\_ok](#) ()  
*Get a reference to the driver\_ok bits ( 2 to 2 ) of raw .*
- typedef [cxx::Bitfield](#)< [decltype\(raw\)](#), 4, 4 > [feature\\_ok\\_bfm\\_t](#)  
*Type to access the feature\_ok bits ( 4 to 4 ) of raw .*
- [feature\\_ok\\_bfm\\_t::Val](#) [feature\\_ok](#) () const  
*Get the feature\_ok bits ( 4 to 4 ) of raw .*
- [feature\\_ok\\_bfm\\_t::Ref](#) [feature\\_ok](#) ()  
*Get a reference to the feature\_ok bits ( 4 to 4 ) of raw .*
- typedef [cxx::Bitfield](#)< [decltype\(raw\)](#), 7, 7 > [failed\\_bfm\\_t](#)  
*Type to access the failed bits ( 7 to 7 ) of raw .*
- [failed\\_bfm\\_t::Val](#) [failed](#) () const  
*Get the failed bits ( 7 to 7 ) of raw .*
- [failed\\_bfm\\_t::Ref](#) [failed](#) ()  
*Get a reference to the failed bits ( 7 to 7 ) of raw .*

### 14.321.1 Detailed Description

Type of the device status register.

Definition at line 43 of file [virtio](#).

### 14.321.2 Member Typedef Documentation

#### 14.321.2.1 `acked_bfm_t`

```
typedef cxx::Bitfield<decltype(raw), 0 , 0 > L4virtio::Svr::Dev_status::acked_bfm_t
```

Type to access the *acked* bits ( 0 to 0 ) of *raw* .

Definition at line 51 of file [virtio](#).

#### 14.321.2.2 `driver_bfm_t`

```
typedef cxx::Bitfield<decltype(raw), 1 , 1 > L4virtio::Svr::Dev_status::driver_bfm_t
```

Type to access the *driver* bits ( 1 to 1 ) of *raw* .

Definition at line 51 of file [virtio](#).

#### 14.321.2.3 `driver_ok_bfm_t`

```
typedef cxx::Bitfield<decltype(raw), 2 , 2 > L4virtio::Svr::Dev_status::driver_ok_bfm_t
```

Type to access the *driver\_ok* bits ( 2 to 2 ) of *raw* .

Definition at line 52 of file [virtio](#).

#### 14.321.2.4 `failed_bfm_t`

```
typedef cxx::Bitfield<decltype(raw), 7 , 7 > L4virtio::Svr::Dev_status::failed_bfm_t
```

Type to access the *failed* bits ( 7 to 7 ) of *raw* .

Definition at line 54 of file [virtio](#).

## 14.321.2.5 feature\_ok\_bfm\_t

```
typedef cxx::Bitfield<decltype(raw), 4 , 4 > L4virtio::Svr::Dev_status::feature_ok_bfm_t
```

Type to access the *feature\_ok* bits ( 4 to 4 ) of *raw* .

Definition at line 53 of file [virtio](#).

## 14.321.3 Member Function Documentation

14.321.3.1 `acked()` [1/2]

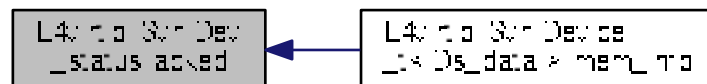
```
acked_bfm_t::Val L4virtio::Svr::Dev_status::acked () const [inline]
```

Get the *acked* bits ( 0 to 0 ) of *raw* .

Definition at line 51 of file [virtio](#).

Referenced by [L4virtio::Svr::Device\\_t<Ds\\_data>::mem\\_info\(\)](#).

Here is the caller graph for this function:

14.321.3.2 `acked()` [2/2]

```
acked_bfm_t::Ref L4virtio::Svr::Dev_status::acked () [inline]
```

Get a reference to the *acked* bits ( 0 to 0 ) of *raw* .

Definition at line 51 of file [virtio](#).

**14.321.3.3 driver()** [1/2]

```
driver_bfm_t::Ref L4virtio::Svr::Dev_status::driver () [inline]
```

Get a reference to the *driver* bits ( 1 to 1 ) of *raw* .

Definition at line 52 of file [virtio](#).

**14.321.3.4 driver()** [2/2]

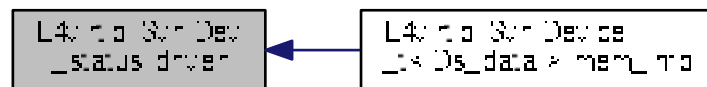
```
driver_bfm_t::Val L4virtio::Svr::Dev_status::driver () const [inline]
```

Get the *driver* bits ( 1 to 1 ) of *raw* .

Definition at line 52 of file [virtio](#).

Referenced by [L4virtio::Svr::Device\\_t<Ds\\_data>::mem\\_info\(\)](#).

Here is the caller graph for this function:

**14.321.3.5 driver\_ok()** [1/2]

```
driver_ok_bfm_t::Val L4virtio::Svr::Dev_status::driver_ok () const [inline]
```

Get the *driver\_ok* bits ( 2 to 2 ) of *raw* .

Definition at line 53 of file [virtio](#).

**14.321.3.6 driver\_ok()** [2/2]

```
driver_ok_bfm_t::Ref L4virtio::Svr::Dev_status::driver_ok () [inline]
```

Get a reference to the *driver\_ok* bits ( 2 to 2 ) of *raw* .

Definition at line 53 of file [virtio](#).



## 14.321.3.7 failed() [1/2]

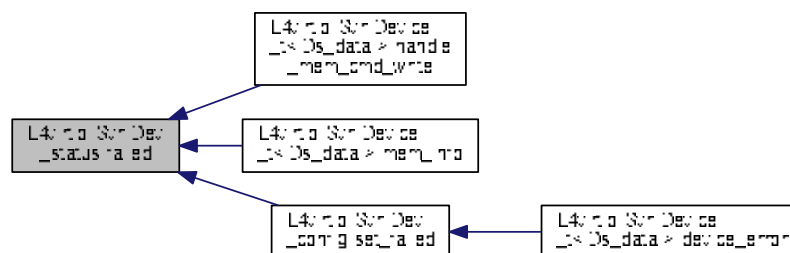
```
failed_bfm_t::Val L4virtio::Svr::Dev_status::failed () const [inline]
```

Get the *failed* bits ( 7 to 7 ) of *raw* .

Definition at line 55 of file [virtio](#).

Referenced by [L4virtio::Svr::Device\\_t< Ds\\_data >::handle\\_mem\\_cmd\\_write\(\)](#), [L4virtio::Svr::Device\\_t< Ds\\_data >::mem\\_info\(\)](#), and [L4virtio::Svr::Dev\\_config::set\\_failed\(\)](#).

Here is the caller graph for this function:



## 14.321.3.8 failed() [2/2]

```
failed_bfm_t::Ref L4virtio::Svr::Dev_status::failed () [inline]
```

Get a reference to the *failed* bits ( 7 to 7 ) of *raw* .

Definition at line 55 of file [virtio](#).

## 14.321.3.9 feature\_ok() [1/2]

```
feature_ok_bfm_t::Ref L4virtio::Svr::Dev_status::feature_ok () [inline]
```

Get a reference to the *feature\_ok* bits ( 4 to 4 ) of *raw* .

Definition at line 54 of file [virtio](#).

14.321.3.10 `feature_ok()` [2/2]

```
feature_ok_bfm_t::Val L4virtio::Svr::Dev_status::feature_ok () const [inline]
```

Get the *feature\_ok* bits ( 4 to 4 ) of *raw* .

Definition at line 54 of file [virtio](#).

14.321.3.11 `running()`

```
bool L4virtio::Svr::Dev_status::running () const [inline]
```

Check if the device is in running state.

## Returns

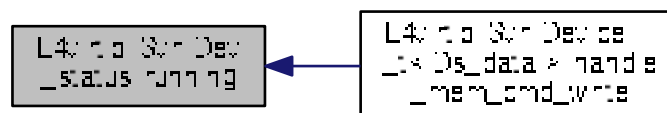
true if the device is in running state.

The device is in running state when [acked\(\)](#), [driver\(\)](#), [feature\\_ok](#), and [driver\\_ok\(\)](#) return true, and [failed\(\)](#) returns false.

Definition at line 65 of file [virtio](#).

Referenced by [L4virtio::Svr::Device\\_t< Ds\\_data >::handle\\_mem\\_cmd\\_write\(\)](#).

Here is the caller graph for this function:



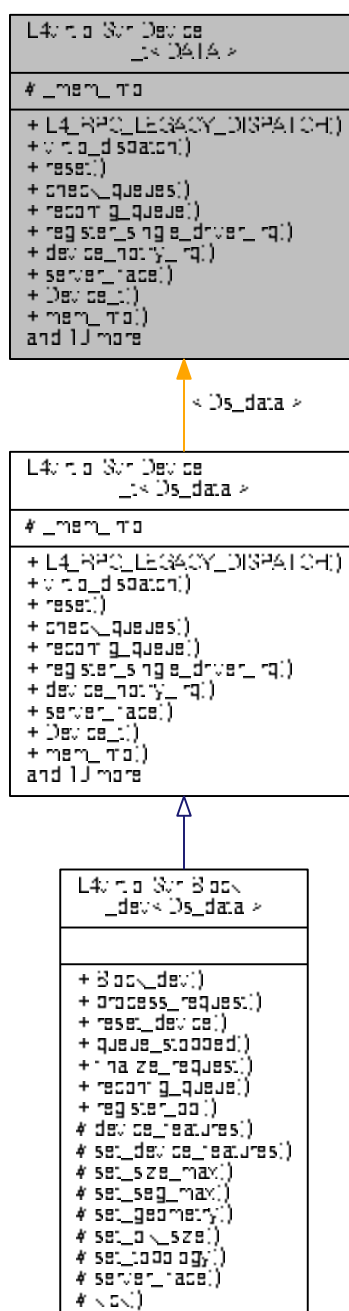
The documentation for this struct was generated from the following file:

- [l4/l4virtio/server/virtio](#)

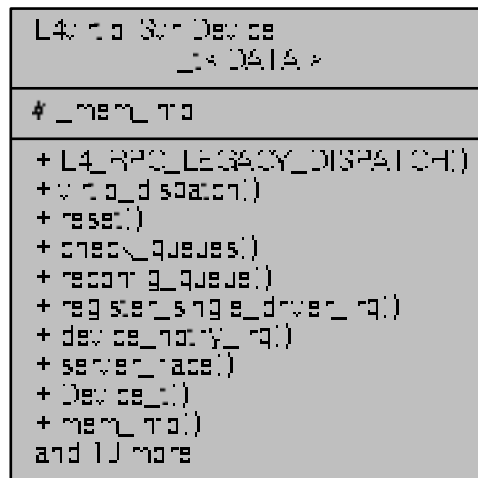
## 14.322 L4virtio::Svr::Device\_t&lt; DATA &gt; Class Template Reference

Server-side L4-VIRTIO device stub.

Inheritance diagram for L4virtio::Svr::Device\_t< DATA >:



Collaboration diagram for L4virtio::Svr::Device\_t< DATA >:



## Public Member Functions

- virtual void [reset](#) ()=0  
*reset callback, called for doing a device reset*
- 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 [register\\_single\\_driver\\_irq](#) ()=0  
*callback for registering a single guest IRQ for all queues*
- virtual [L4::Cap](#)< [L4::Irq](#) > [device\\_notify\\_irq](#) () const =0  
*callback to gather the device notification IRQ*
- [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)  
*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 failed 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.*

## Protected Attributes

- [Mem\\_list\\_mem\\_info](#)  
*Memory region list.*

### 14.322.1 Detailed Description

```
template<typename DATA>
class L4virtio::Svr::Device_t< DATA >
```

Server-side L4-VIRTIO device stub.

Definition at line 652 of file [l4virtio](#).

### 14.322.2 Member Function Documentation

#### 14.322.2.1 device\_error()

```
template<typename DATA>
void L4virtio::Svr::Device_t< DATA >::device_error () [inline]
```

Transition device into failed state.

#### Note

Callers should trigger a guest config IRQ after calling this function.

This function does a full reset, (calls [reset\(\)](#)) and sets the failed bit in the device status register.

Definition at line 777 of file [l4virtio](#).

#### 14.322.2.2 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 897 of file [l4virtio](#).

#### 14.322.2.3 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 764 of file [l4virtio](#).

14.322.2.4 `reset_queue_config()`

```
template<typename DATA>
void L4virtio::Svr::Device_t< DATA >::reset_queue_config (
 unsigned idx,
 unsigned num_max,
 bool inc_generation = false) [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. |

This function resets the driver-readable configuration space for the queue with the given index. The queue configuration is reset to all 0, name *num\_max* to the given value.

Definition at line 754 of file [l4virtio](#).

14.322.2.5 `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 796 of file [l4virtio](#).

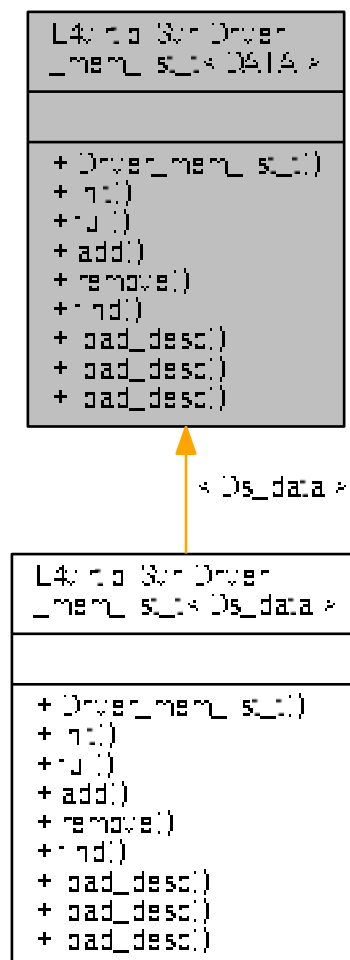
The documentation for this class was generated from the following file:

- [l4/l4virtio/server/l4virtio](#)

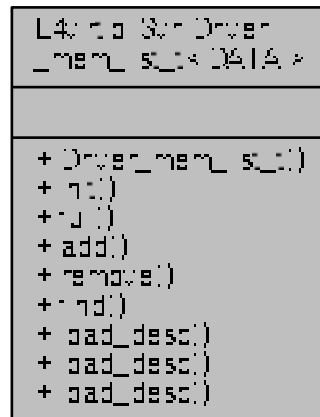
## 14.323 L4virtio::Svr::Driver\_mem\_list\_t< DATA > Class Template Reference

List of driver memory regions assigned to a single L4-VIRTIO transport instance.

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::Auto\\_cap< L4Re::Dataspace >::Cap](#) [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.



### 14.323.1 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 499 of file [l4virtio](#).

### 14.323.2 Member Function Documentation

#### 14.323.2.1 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.                      |

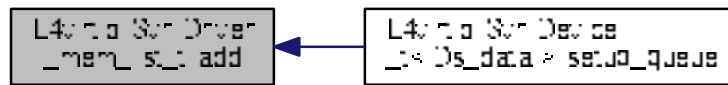
#### Returns

A pointer to the new region.

Definition at line 539 of file [l4virtio](#).

Referenced by [L4virtio::Svr::Device\\_t< Ds\\_data >::setup\\_queue\(\)](#).

Here is the caller graph for this function:



#### 14.323.2.2 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.  |

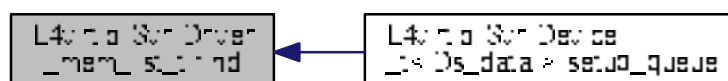
##### Returns

Pointer to the region containing the given region, NULL if none is found.

Definition at line 573 of file [l4virtio](#).

Referenced by [L4virtio::Svr::Device\\_t<Ds\\_data>::setup\\_queue\(\)](#).

Here is the caller graph for this function:



## 14.323.2.3 full()

```
template<typename DATA>
bool L4virtio::Svr::Driver_mem_list_t< DATA >::full () const [inline]
```

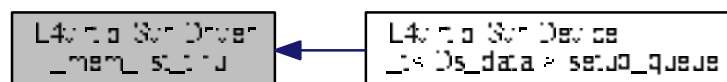
## Returns

True if the remaining capacity is 0.

Definition at line 528 of file [l4virtio](#).

Referenced by [L4virtio::Svr::Device\\_t< Ds\\_data >::setup\\_queue\(\)](#).

Here is the caller graph for this function:



## 14.323.2.4 init()

```
template<typename DATA>
void L4virtio::Svr::Driver_mem_list_t< DATA >::init (
 unsigned max) [inline]
```

Make a fresh list with capacity *max*.

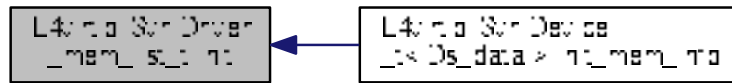
## Parameters

|            |                              |
|------------|------------------------------|
| <i>max</i> | The capacity of this vector. |
|------------|------------------------------|

Definition at line 520 of file [l4virtio](#).

Referenced by [L4virtio::Svr::Device\\_t< Ds\\_data >::init\\_mem\\_info\(\)](#).

Here is the caller graph for this function:



#### 14.323.2.5 load\_desc() [1/3]

```

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 |

Definition at line 584 of file [l4virtio](#).

#### 14.323.2.6 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 Driver_mem_region that covers the descriptor. |

Definition at line 601 of file [l4virtio](#).

#### 14.323.2.7 load\_desc() [3/3]

```
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 <a href="#">Request_processor::start()</a> and <a href="#">Request_processor::next()</a> 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 <a href="#">Virtqueue::Desc</a> descriptor, and (3) a pointer to the calling <a href="#">Request_processor</a> . |
|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

#### Parameters

|     |             |                                        |
|-----|-------------|----------------------------------------|
|     | <i>desc</i> | The descriptor to load                 |
|     | <i>p</i>    | The request processor calling us       |
| out | <i>data</i> | Shall be assigned to ARG(mem, desc, p) |

Definition at line 624 of file [l4virtio](#).

#### 14.323.2.8 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 <a href="#">add()</a> , or <a href="#">find()</a> ). |
|----------|----------------------------------------------------------------------------------------|

Definition at line 553 of file [l4virtio](#).

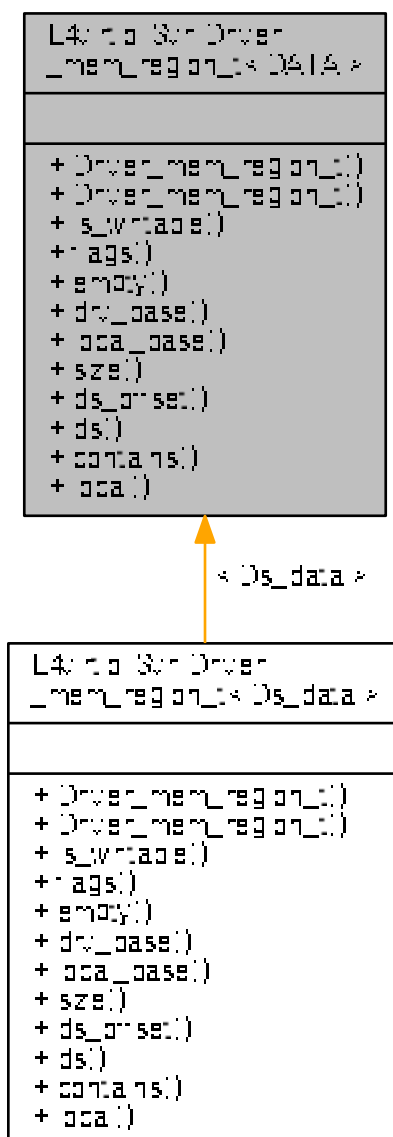
The documentation for this class was generated from the following file:

- `l4/l4virtio/server/l4virtio`

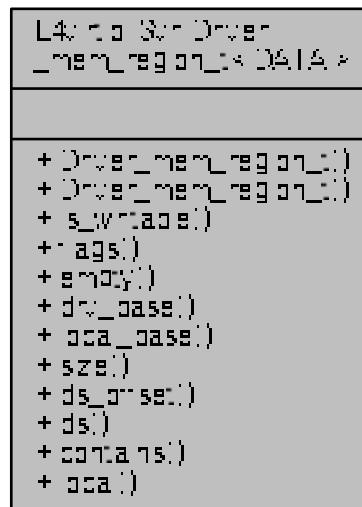
### 14.324 L4virtio::Svr::Driver\_mem\_region\_t< DATA > Class Template Reference

Region of driver memory, that shall be managed locally.

Inheritance diagram for L4virtio::Svr::Driver\_mem\_region\_t< DATA >:



Collaboration diagram for L4virtio::Svr::Driver\_mem\_region\_t< DATA >:



## 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
- void \* [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.*

### 14.324.1 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 342 of file [l4virtio](#).

## 14.324.2 Constructor & Destructor Documentation

### 14.324.2.1 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 387 of file [l4virtio](#).

## 14.324.3 Member Function Documentation

### 14.324.3.1 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 462 of file [l4virtio](#).

#### 14.324.3.2 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 441 of file [l4virtio](#).

#### 14.324.3.3 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 453 of file [l4virtio](#).

#### 14.324.3.4 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 450 of file [l4virtio](#).

#### 14.324.3.5 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 else.

Definition at line 437 of file [l4virtio](#).

#### 14.324.3.6 flags()

```
template<typename DATA>
Flags L4virtio::Svr::Driver_mem_region_t< DATA >::flags () const [inline]
```

##### Returns

The flags for this region.

Definition at line 434 of file [l4virtio](#).

#### 14.324.3.7 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 else.

Definition at line 431 of file [l4virtio](#).

#### 14.324.3.8 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.

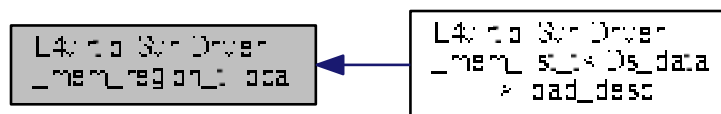
## Returns

Local address for the given driver address *p*.

Definition at line 486 of file [l4virtio](#).

Referenced by [L4virtio::Svr::Driver\\_mem\\_list\\_t< Ds\\_data >::load\\_desc\(\)](#).

Here is the caller graph for this function:



## 14.324.3.9 local\_base()

```
template<typename DATA>
void* L4virtio::Svr::Driver_mem_region_t< DATA >::local_base () const [inline]
```

## Returns

The local base address.

Definition at line 444 of file [l4virtio](#).

### 14.324.3.10 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 447 of file [l4virtio](#).

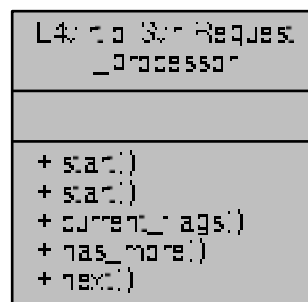
The documentation for this class was generated from the following file:

- l4/l4virtio/server/l4virtio

## 14.325 L4virtio::Svr::Request\_processor Class Reference

Encapsulate the state for processing a VIRTIO request.

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.*

### 14.325.1 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 381 of file [virtio](#).

### 14.325.2 Member Function Documentation

#### 14.325.2.1 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 445 of file [virtio](#).

References [L4virtio::Virtqueue::Desc::flags](#).

### 14.325.2.2 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 452 of file [virtio](#).

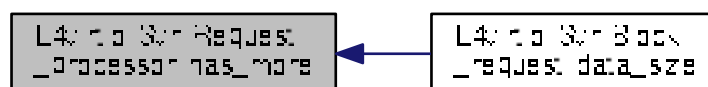
References [L4virtio::Virtqueue::Desc::flags](#), and [L4virtio::Virtqueue::Desc::Flags::next\(\)](#).

Referenced by [L4virtio::Svr::Block\\_request< Ds\\_data >::data\\_size\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 14.325.2.3 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>DESCM_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()                                 |

## Returns

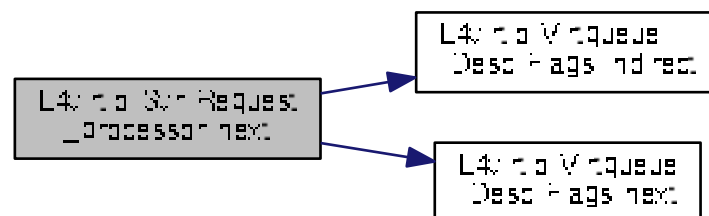
true if a descriptor is available, false if not.

Definition at line 463 of file [virtio](#).

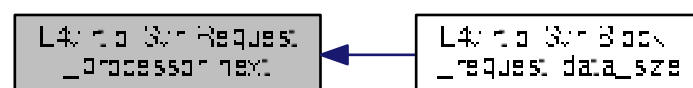
References [L4virtio::Svr::Bad\\_descriptor::Bad\\_flags](#), [L4virtio::Svr::Bad\\_descriptor::Bad\\_next](#), [L4virtio::Virtqueue::Desc::flags](#), [L4virtio::Virtqueue::Desc::Flags::indirect\(\)](#), [L4\\_UNLIKELY](#), [L4virtio::Virtqueue::Desc::Flags::next\(\)](#), and [L4virtio::Virtqueue::Desc::next](#).

Referenced by [L4virtio::Svr::Block\\_request<Ds\\_data>::data\\_size\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



14.325.2.4 `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>DESCM_MAN</code> | Type of descriptor manager (implicit). |
|------------------------|----------------------------------------|

## Parameters

|                      |                                                                           |
|----------------------|---------------------------------------------------------------------------|
| <code>dm</code>      | Descriptor manager that is used to translate VIRTIO descriptor addresses. |
| <code>ring</code>    | VIRTIO ring of the request.                                               |
| <code>request</code> | VIRTIO request from <a href="#">Virtqueue::next_avail()</a>               |
| <code>args</code>    | Extra arguments passed to <code>dm-&gt;load_desc()</code>                 |

## Precondition

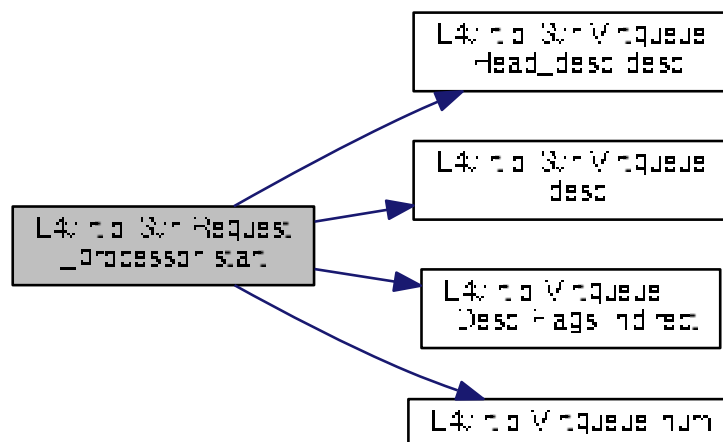
The given request must be valid.

Definition at line 404 of file [virtio](#).

References [L4virtio::Svr::Bad\\_descriptor::Bad\\_size](#), [L4virtio::Svr::Virtqueue::Head\\_desc::desc\(\)](#), [L4virtio::Svr::Virtqueue::desc\(\)](#), [L4virtio::Virtqueue::Desc::flags](#), [L4virtio::Virtqueue::Desc::Flags::indirect\(\)](#), [L4\\_UNLIKELY](#), [L4virtio::Virtqueue::Desc::len](#), and [L4virtio::Virtqueue::num\(\)](#).

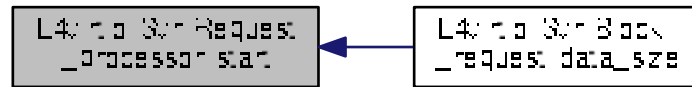
Referenced by [L4virtio::Svr::Block\\_request<Ds\\_data>::data\\_size\(\)](#).

Here is the call graph for this function:





Here is the caller graph for this function:



#### 14.325.2.5 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

|                  |                                        |
|------------------|----------------------------------------|
| <i>DESCM_MAN</i> | Type of descriptor manager (implicit). |
|------------------|----------------------------------------|

##### Parameters

|                |                                                                           |
|----------------|---------------------------------------------------------------------------|
| <i>dm</i>      | Descriptor manager that is used to translate VIRTIO descriptor addresses. |
| <i>request</i> | VIRTIO request from <a href="#">Virtqueue::next_avail()</a>               |
| <i>args</i>    | Extra arguments passed to dm->load_desc()                                 |

##### Precondition

The given request must be valid.

Definition at line 435 of file [virtio](#).

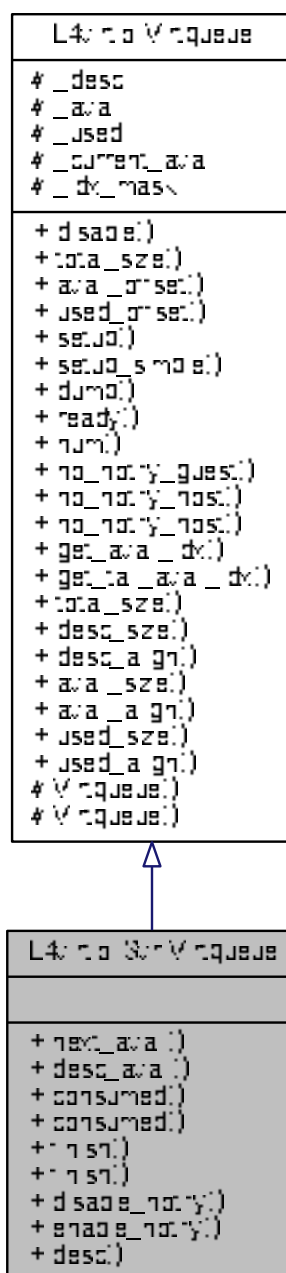
The documentation for this class was generated from the following file:

- I4/I4virtio/server/virtio

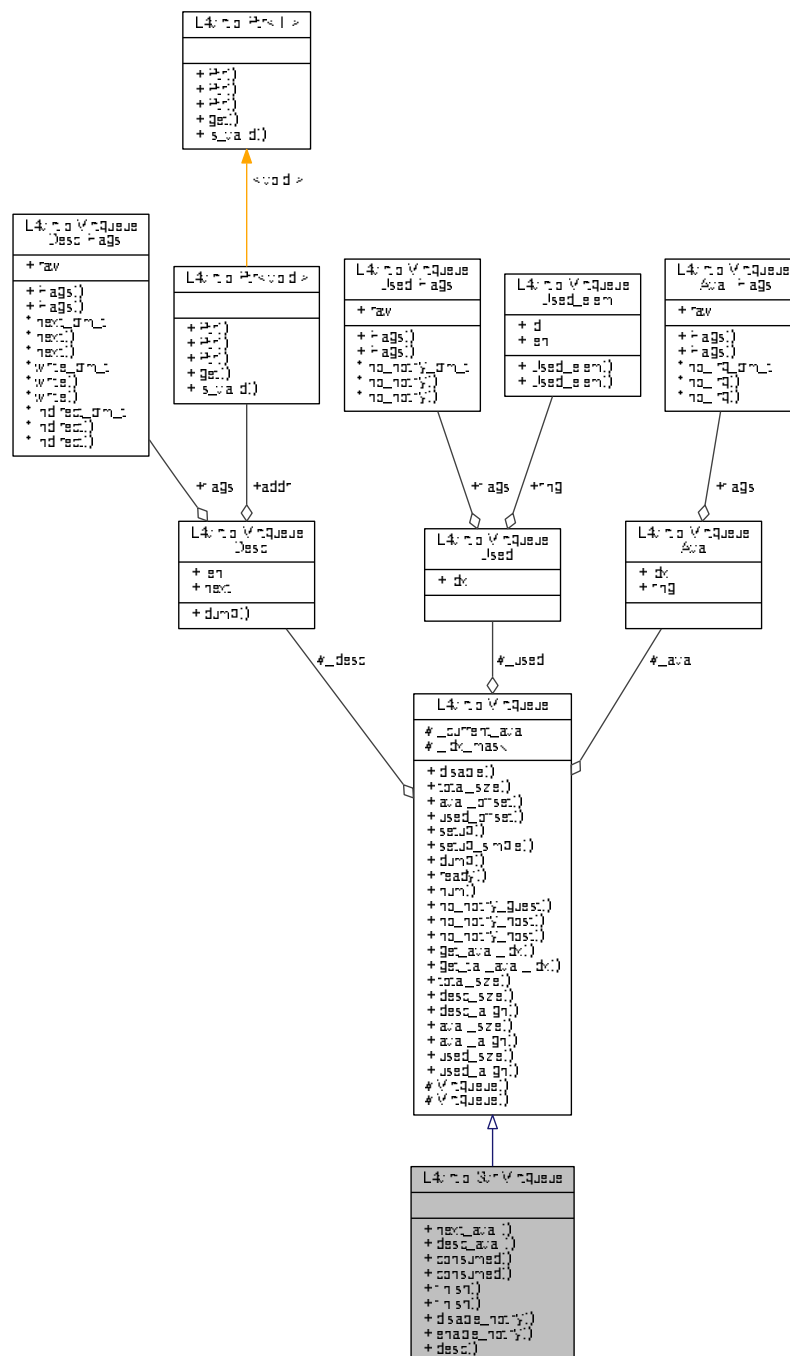
## 14.326 L4virtio::Svr::Virtqueue Class Reference

[Virtqueue](#) implementation for the device.

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.*
- 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.*
- 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.*

## Additional Inherited Members

### 14.326.1 Detailed Description

[Virtqueue](#) implementation for the device.

This class represents a single virtqueue, with a local running available index.

Definition at line 93 of file [virtio](#).

### 14.326.2 Member Function Documentation

#### 14.326.2.1 consumed()

```
void L4virtio::Svr::Virtqueue::consumed (
 Head_desc const & r,
 l4_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.        |

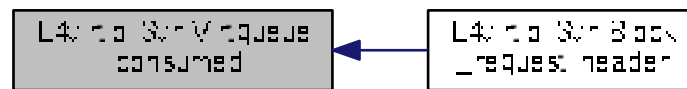
#### Precondition

queue must be in working state.  
*r* must be a valid request from this queue.

Definition at line 172 of file [virtio](#).

Referenced by [L4virtio::Svr::Block\\_request<Ds\\_data>::header\(\)](#).

Here is the caller graph for this function:



#### 14.326.2.2 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. |
|------------|------------------------------|

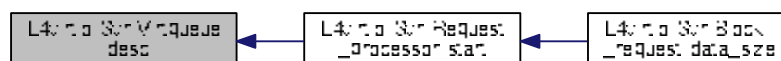
##### Precondition

$idx < num$   
queue must be in working state

Definition at line 237 of file [virtio](#).

Referenced by [L4virtio::Svr::Request\\_processor::start\(\)](#).

Here is the caller graph for this function:



#### 14.326.2.3 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 160 of file [virtio](#).

#### 14.326.2.4 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 214 of file [virtio](#).

References [L4\\_LIKELY](#).

#### 14.326.2.5 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 225 of file [virtio](#).

References [L4\\_LIKELY](#).

## 14.326.2.6 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**

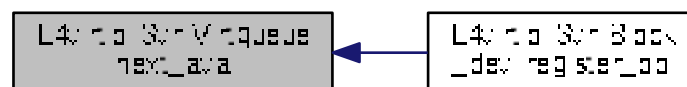
The return value must be checked even when a previous [desc\\_avail\(\)](#) returned true.

Definition at line 143 of file [virtio](#).

References [L4\\_LIKELY](#).

Referenced by [L4virtio::Svr::Block\\_dev<Ds\\_data>::register\\_obj\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

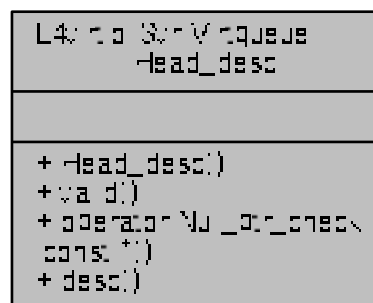
- `I4/I4virtio/server/virtio`

## 14.327 L4virtio::Svr::Virtqueue::Head\_desc Class Reference

VIRTIO request, essentially a descriptor from the available ring.

Inherited by L4virtio::Svr::Virtqueue::Request.

Collaboration diagram for L4virtio::Svr::Virtqueue::Head\_desc:



### Public Member Functions

- [Head\\_desc](#) ()  
*Make invalid (NULL) request.*
- bool [valid](#) () const
- [operator Null\\_ptr\\_check const \\*](#) () const
- [Desc](#) const \* [desc](#) () const

### 14.327.1 Detailed Description

VIRTIO request, essentially a descriptor from the available ring.

Definition at line 99 of file [virtio](#).

### 14.327.2 Member Function Documentation



## 14.327.2.1 desc()

```
Desc const* L4virtio::Svr::Virtqueue::Head_desc::desc () const [inline]
```

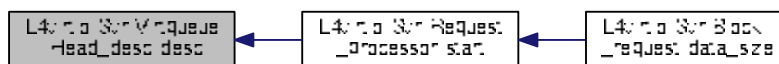
## Returns

Pointer to the head descriptor of the request.

Definition at line 120 of file [virtio](#).

Referenced by [L4virtio::Svr::Request\\_processor::start\(\)](#).

Here is the caller graph for this function:



## 14.327.2.2 operator Null\_ptr\_check const \*()

```
L4virtio::Svr::Virtqueue::Head_desc::operator Null_ptr_check const * () const [inline]
```

## Returns

True if the request is valid (not NULL).

Definition at line 116 of file [virtio](#).

## 14.327.2.3 valid()

```
bool L4virtio::Svr::Virtqueue::Head_desc::valid () const [inline]
```

## Returns

True if the request is valid (not NULL).

Definition at line 113 of file [virtio](#).

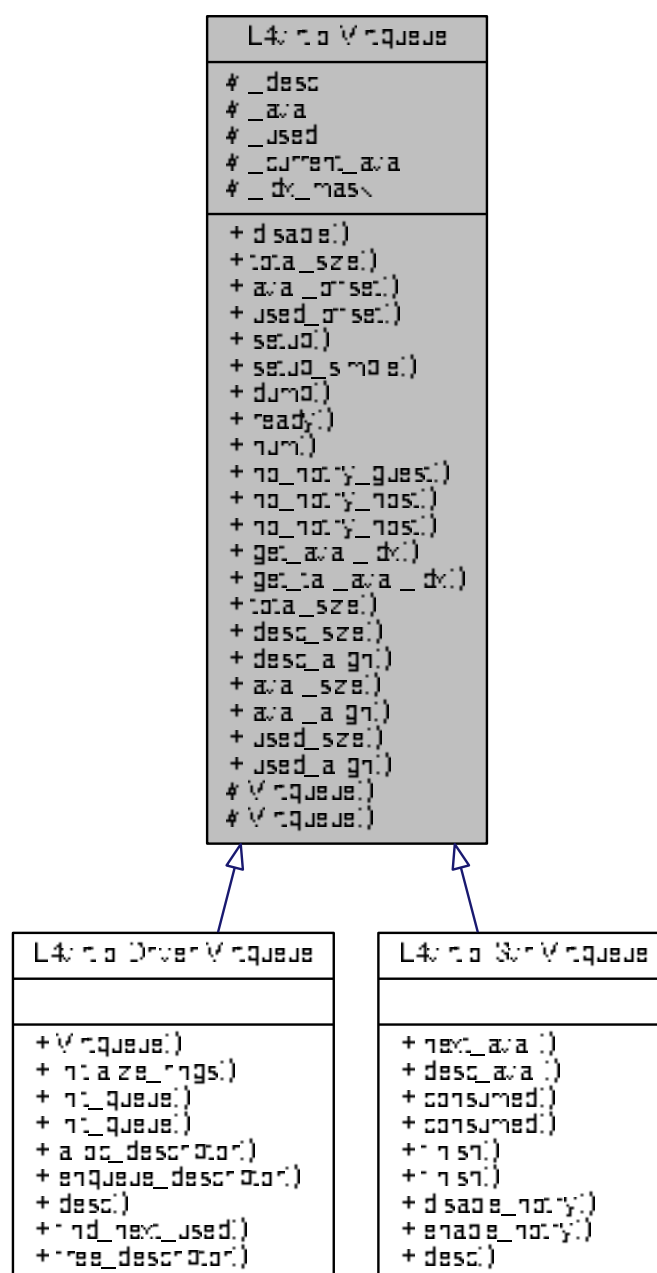
The documentation for this class was generated from the following file:

- `I4/I4virtio/server/virtio`

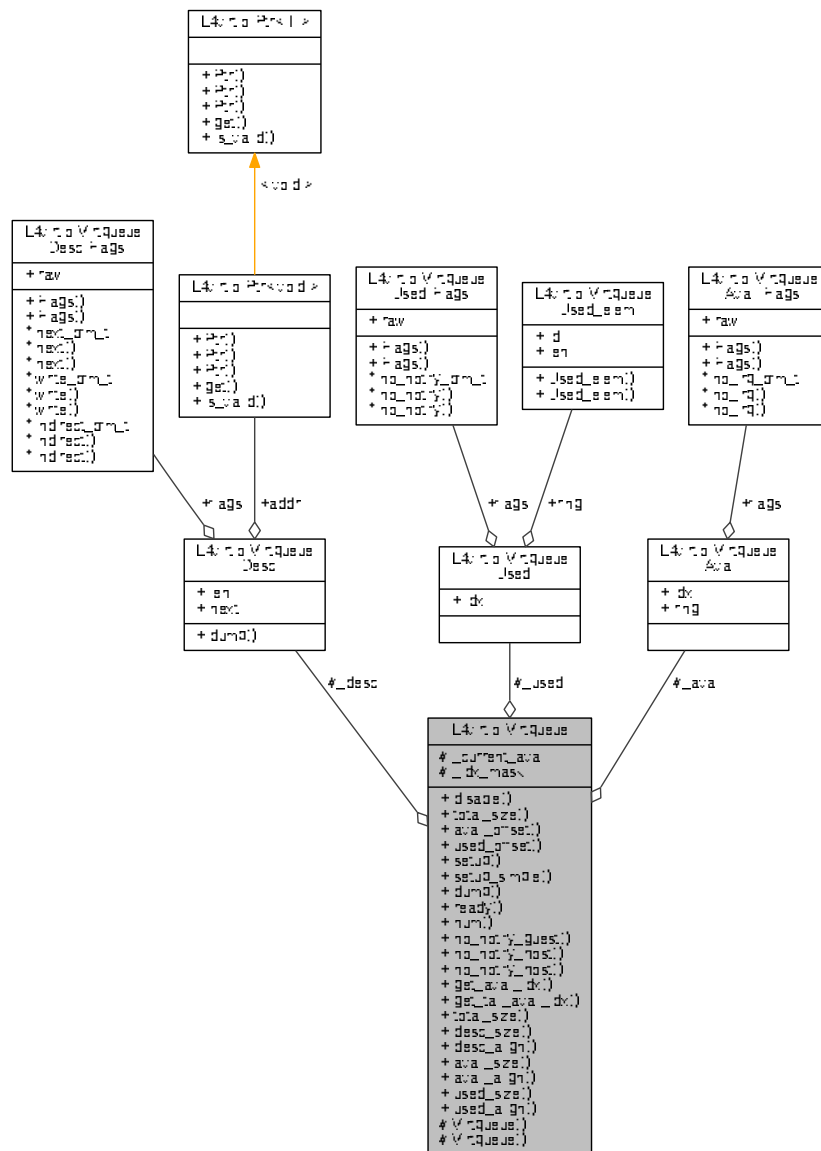
## 14.328 L4virtio::Virtqueue Class Reference

Low-level [Virtqueue](#).

Inheritance diagram for L4virtio::Virtqueue:



Collaboration diagram for L4virtio::Virtqueue:



## Data Structures

- class [Avail](#)  
Type of available ring, this is read-only for the host.
- class [Desc](#)  
Descriptor in the descriptor table.
- class [Used](#)  
Used ring.
- struct [Used\\_elem](#)  
Type of an element of the 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 \(\)](#)  
*Create a disabled virtqueue.*

## Protected Attributes

- [Desc \\* \\_desc](#)  
*pointer to descriptor table, NULL if queue is off.*
- [Avail \\* \\_avail](#)  
*pointer to available ring.*
- [Used \\* \\_used](#)  
*pointer to used ring.*
- [l4\\_uint16\\_t \\_current\\_avail](#)  
*The life counter for the queue.*
- [l4\\_uint16\\_t \\_idx\\_mask](#)  
*mask used for indexing into the descriptor table and the rings.*

### 14.328.1 Detailed Description

Low-level [Virtqueue](#).

This class represents a single virtqueue, with a local running available index.

Definition at line 87 of file [virtqueue](#).

### 14.328.2 Member Function Documentation

#### 14.328.2.1 [avail\\_align\(\)](#)

```
static 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 291 of file [virtqueue](#).

#### 14.328.2.2 [avail\\_size\(\)](#)

```
static unsigned long L4virtio::Virtqueue::avail_size (
 unsigned num) [inline], [static]
```

Calculate the size of the available ring for `num` entries.

## Parameters

|            |                                              |
|------------|----------------------------------------------|
| <i>num</i> | The number of entries in the available ring. |
|------------|----------------------------------------------|

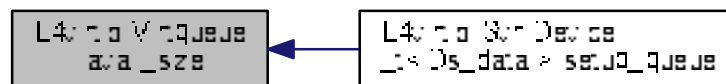
## Returns

The size in bytes needed for an available ring with *num* entries.

Definition at line 283 of file [virtqueue](#).

Referenced by [L4virtio::Svr::Device\\_t< Ds\\_data >::setup\\_queue\(\)](#).

Here is the caller graph for this function:



## 14.328.2.3 desc\_align()

```
static 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 273 of file [virtqueue](#).

## 14.328.2.4 desc\_size()

```
static 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. |
|------------|------------------------------------------------|

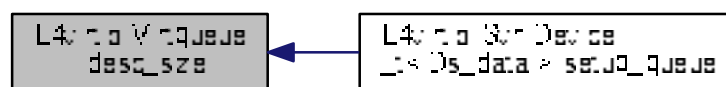
## Returns

The size in bytes needed for a descriptor table with *num* entries.

Definition at line 265 of file [virtqueue](#).

Referenced by [L4virtio::Svr::Device\\_t< Ds\\_data >::setup\\_queue\(\)](#).

Here is the caller graph for this function:

14.328.2.5 `disable()`

```
void L4virtio::Virtqueue::disable () [inline]
```

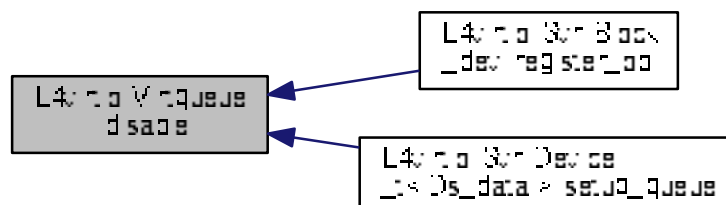
Completely disable the queue.

[setup\(\)](#) must be used to enable the queue again.

Definition at line 228 of file [virtqueue](#).

Referenced by [L4virtio::Svr::Block\\_dev< Ds\\_data >::register\\_obj\(\)](#), and [L4virtio::Svr::Device\\_t< Ds\\_data >::setup\\_queue\(\)](#).

Here is the caller graph for this function:



#### 14.328.2.6 dump()

```
void L4virtio::Virtqueue::dump (
 Desc const * d) const [inline]
```

Dump descriptors for this queue.

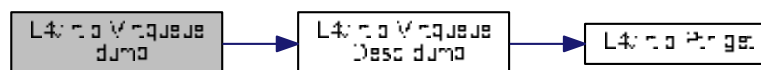
##### Precondition

the queue must be in working state.

Definition at line 395 of file [virtqueue](#).

References [L4virtio::Virtqueue::Desc::dump\(\)](#).

Here is the call graph for this function:



#### 14.328.2.7 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 452 of file [virtqueue](#).

References [L4virtio::Virtqueue::Avail::idx](#).



## 14.328.2.8 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 459 of file [virtqueue](#).

## 14.328.2.9 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 417 of file [virtqueue](#).

References [L4virtio::Virtqueue::Avail::flags](#), and [L4virtio::Virtqueue::Avail::Flags::no\\_irq\(\)](#).

Here is the call graph for this function:



**14.328.2.10 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.

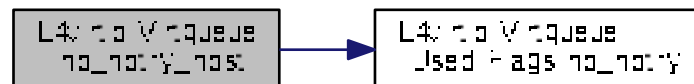
**Returns**

true if the host does not want to get IRQs (currently).

Definition at line 429 of file [virtqueue](#).

References [L4virtio::Virtqueue::Used::flags](#), and [L4virtio::Virtqueue::Used::Flags::no\\_notify\(\)](#).

Here is the call graph for this function:

**14.328.2.11 no\_notify\_host()** [2/2]

```
void L4virtio::Virtqueue::no_notify_host (
 bool value) [inline]
```

Set the no-notify flag for this queue.

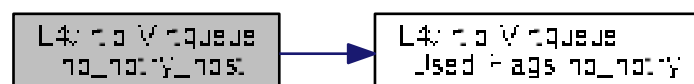
**Precondition**

Queue must be in a working state.

Definition at line 439 of file [virtqueue](#).

References [L4virtio::Virtqueue::Used::flags](#), and [L4virtio::Virtqueue::Used::Flags::no\\_notify\(\)](#).

Here is the call graph for this function:



## 14.328.2.12 num()

```
unsigned L4virtio::Virtqueue::num () const [inline]
```

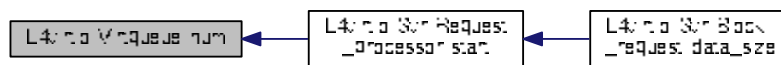
## Returns

The number of entries in the ring.

Definition at line 407 of file [virtqueue](#).

Referenced by [L4virtio::Svr::Request\\_processor::start\(\)](#).

Here is the caller graph for this function:



## 14.328.2.13 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 403 of file [virtqueue](#).

References [L4\\_LIKELY](#).

Referenced by [L4virtio::Svr::Block\\_dev<Ds\\_data>::register\\_obj\(\)](#).

Here is the caller graph for this function:



**14.328.2.14 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 Desc_align aligned and at least desc_size(num) bytes in size.) |
| <i>avail</i> | The address of the available ring. (Must be Avail_align aligned and at least avail_size(num) bytes in size.) |
| <i>used</i>  | The address of the used ring. (Must be Used_align aligned and at least used_size(num) bytes in size.)        |

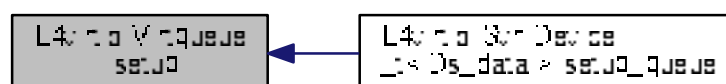
Due to the data type of the descriptors, the queue can have a maximum size of  $2^{16}$ .

Definition at line 353 of file [virtqueue](#).

References [L4\\_EINVAL](#).

Referenced by [L4virtio::Svr::Device\\_t<Ds\\_data>::setup\\_queue\(\)](#).

Here is the caller graph for this function:

**14.328.2.15 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 382 of file [virtqueue](#).

References [l4\\_round\\_size\(\)](#).

Here is the call graph for this function:

14.328.2.16 `total_size()` [1/2]

```
static 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 249 of file [virtqueue](#).

14.328.2.17 `total_size()` [2/2]

```
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 318 of file [virtqueue](#).

**14.328.2.18 used\_align()**

```
static 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 310 of file [virtqueue](#).

**14.328.2.19 used\_size()**

```
static 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. |
|------------|-----------------------------------------|

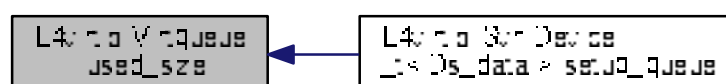
**Returns**

The size in bytes needed for an used ring with `num` entries.

Definition at line 302 of file [virtqueue](#).

Referenced by [L4virtio::Svr::Device\\_t<Ds\\_data>::setup\\_queue\(\)](#).

Here is the caller graph for this function:



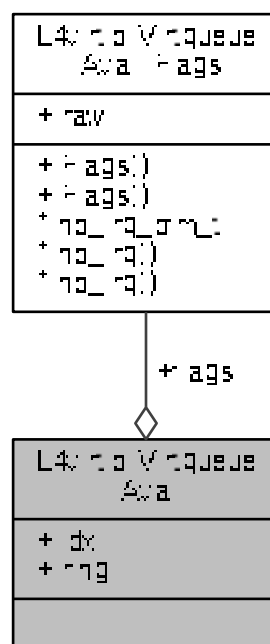
The documentation for this class was generated from the following file:

- l4/l4virtio/virtqueue

## 14.329 L4virtio::Virtqueue::Avail Class Reference

Type of available ring, this is read-only for the host.

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.*

### 14.329.1 Detailed Description

Type of available ring, this is read-only for the host.

Definition at line 134 of file [virtqueue](#).

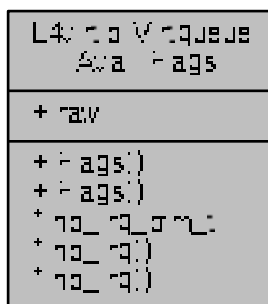
The documentation for this class was generated from the following file:

- l4/l4virtio/virtqueue

## 14.330 L4virtio::Virtqueue::Avail::Flags Struct Reference

[Flags](#) of the available ring.

Collaboration diagram for L4virtio::Virtqueue::Avail::Flags:



### 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.*
- [no\\_irq\\_bfm\\_t::Val](#) no\_irq () const  
*Get the no\_irq bits ( 0 to 0 ) of raw .*
- [no\\_irq\\_bfm\\_t::Ref](#) no\_irq ()  
*Get a reference to the no\_irq bits ( 0 to 0 ) of raw .*



### 14.330.1 Detailed Description

Flags of the available ring.

Definition at line 140 of file [virtqueue](#).

### 14.330.2 Member Typedef Documentation

#### 14.330.2.1 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](#).

### 14.330.3 Member Function Documentation

#### 14.330.3.1 no\_irq() [1/2]

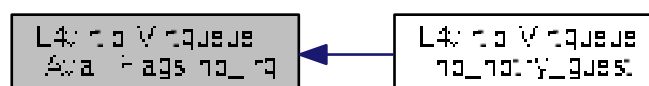
```
no_irq_bfm_t::Val L4virtio::Virtqueue::Avail::Flags::no_irq () const [inline]
```

Get the *no\_irq* bits ( 0 to 0 ) of *raw* .

Definition at line 149 of file [virtqueue](#).

Referenced by [L4virtio::Virtqueue::no\\_notify\\_guest\(\)](#).

Here is the caller graph for this function:



### 14.330.3.2 no\_irq() [2/2]

```
no_irq_bfm_t::Ref L4virtio::Virtqueue::Avail::Flags::no_irq () [inline]
```

Get a reference to the *no\_irq* bits ( 0 to 0 ) of *raw* .

Definition at line 149 of file [virtqueue](#).

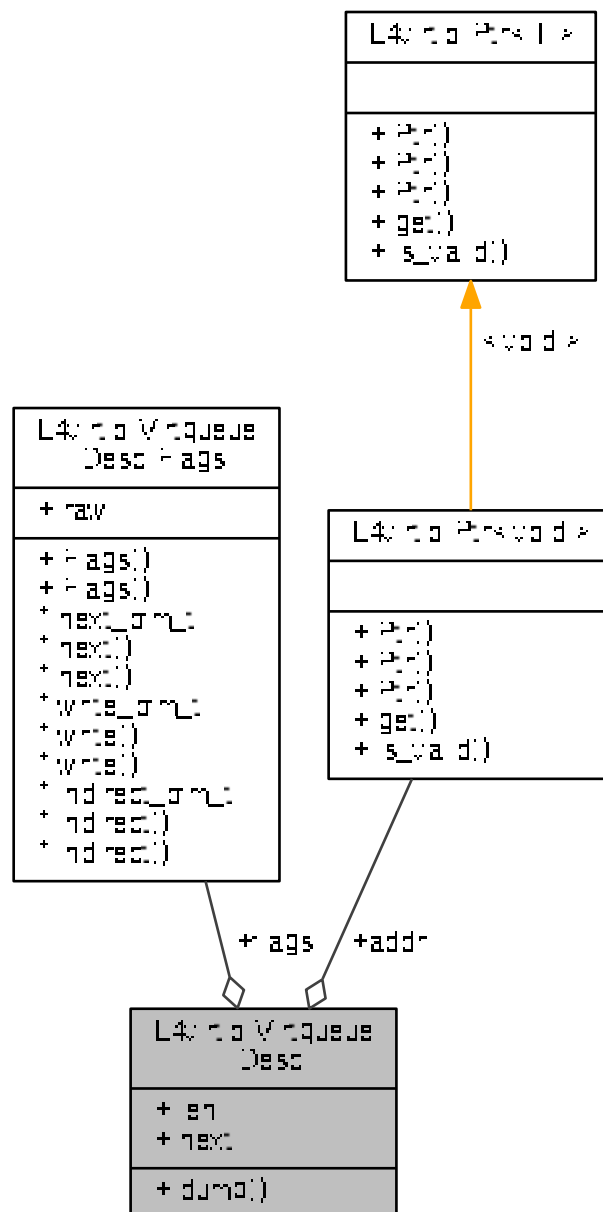
The documentation for this struct was generated from the following file:

- l4/l4virtio/virtqueue

## 14.331 L4virtio::Virtqueue::Desc Class Reference

Descriptor in the descriptor table.

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.*

### 14.331.1 Detailed Description

Descriptor in the descriptor table.

Definition at line 93 of file [virtqueue](#).

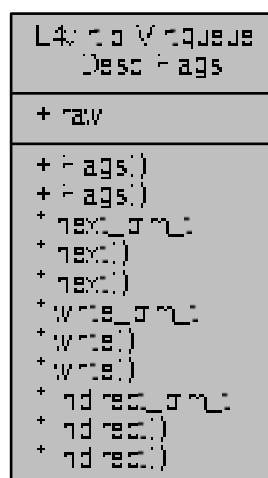
The documentation for this class was generated from the following file:

- [l4/l4virtio/virtqueue](#)

### 14.332 L4virtio::Virtqueue::Desc::Flags Struct Reference

Type for descriptor flags.

Collaboration diagram for L4virtio::Virtqueue::Desc::Flags:



## 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.*
- [next\\_bfm\\_t::Val](#) next () const  
*Get the next bits ( 0 to 0 ) of raw .*
- [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.*
- [write\\_bfm\\_t::Val](#) write () const  
*Get the write bits ( 1 to 1 ) of raw .*
- [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.*
- [indirect\\_bfm\\_t::Val](#) indirect () const  
*Get the indirect bits ( 2 to 2 ) of raw .*
- [indirect\\_bfm\\_t::Ref](#) indirect ()  
*Get a reference to the indirect bits ( 2 to 2 ) of raw .*

### 14.332.1 Detailed Description

Type for descriptor flags.

Definition at line 99 of file [virtqueue](#).

### 14.332.2 Member Typedef Documentation

## 14.332.2.1 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 110 of file [virtqueue](#).

## 14.332.2.2 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 108 of file [virtqueue](#).

## 14.332.2.3 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 108 of file [virtqueue](#).

## 14.332.3 Member Function Documentation

## 14.332.3.1 indirect() [1/2]

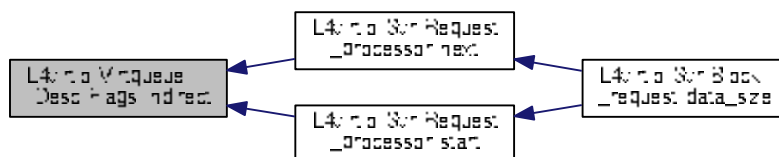
```
indirect_bfm_t::Val L4virtio::Virtqueue::Desc::Flags::indirect () const [inline]
```

Get the *indirect* bits ( 2 to 2 ) of *raw* .

Definition at line 112 of file [virtqueue](#).

Referenced by [L4virtio::Svr::Request\\_processor::next\(\)](#), and [L4virtio::Svr::Request\\_processor::start\(\)](#).

Here is the caller graph for this function:



## 14.332.3.2 indirect() [2/2]

```
indirect_bfm_t::Ref L4virtio::Virtqueue::Desc::Flags::indirect () [inline]
```

Get a reference to the *indirect* bits ( 2 to 2 ) of *raw* .

Definition at line 112 of file [virtqueue](#).

## 14.332.3.3 next() [1/2]

```
next_bfm_t::Ref L4virtio::Virtqueue::Desc::Flags::next () [inline]
```

Get a reference to the *next* bits ( 0 to 0 ) of *raw* .

Definition at line 108 of file [virtqueue](#).

## 14.332.3.4 next() [2/2]

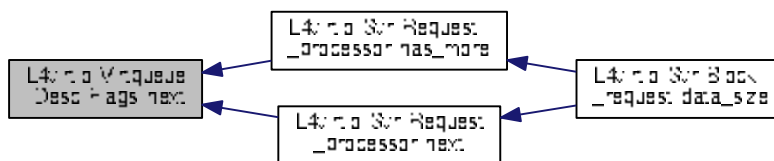
```
next_bfm_t::Val L4virtio::Virtqueue::Desc::Flags::next () const [inline]
```

Get the *next* bits ( 0 to 0 ) of *raw* .

Definition at line 108 of file [virtqueue](#).

Referenced by [L4virtio::Svr::Request\\_processor::has\\_more\(\)](#), and [L4virtio::Svr::Request\\_processor::next\(\)](#).

Here is the caller graph for this function:



## 14.332.3.5 write() [1/2]

```
write_bfm_t::Val L4virtio::Virtqueue::Desc::Flags::write () const [inline]
```

Get the *write* bits ( 1 to 1 ) of *raw* .

Definition at line 110 of file [virtqueue](#).

## 14.332.3.6 write() [2/2]

```
write_bfm_t::Ref L4virtio::Virtqueue::Desc::Flags::write () [inline]
```

Get a reference to the *write* bits ( 1 to 1 ) of *raw* .

Definition at line 110 of file [virtqueue](#).

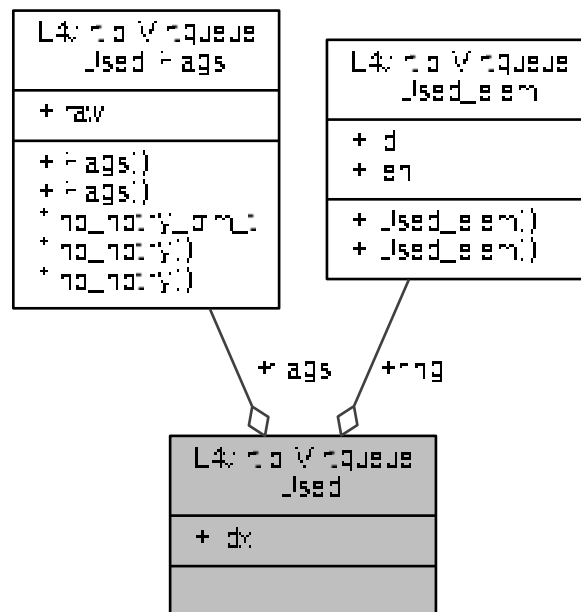
The documentation for this struct was generated from the following file:

- [l4/l4virtio/virtqueue](#)

## 14.333 L4virtio::Virtqueue::Used Class Reference

[Used](#) ring.

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.*

14.333.1 Detailed Description

[Used](#) ring.

Definition at line 179 of file [virtqueue](#).

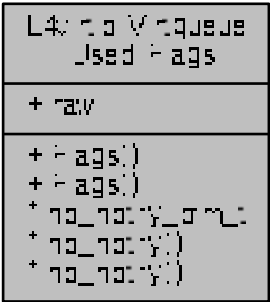
The documentation for this class was generated from the following file:

- l4/l4virtio/virtqueue

14.334 L4virtio::Virtqueue::Used::Flags Struct Reference

flags for the used ring.

Collaboration diagram for L4virtio::Virtqueue::Used::Flags:



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.*
- [no\\_notify\\_bfm\\_t::Val](#) `no_notify` () const  
*Get the no\_notify bits ( 0 to 0 ) of raw .*
- [no\\_notify\\_bfm\\_t::Ref](#) `no_notify` ()  
*Get a reference to the no\_notify bits ( 0 to 0 ) of raw .*

### 14.334.1 Detailed Description

flags for the used ring.

Definition at line [185](#) of file [virtqueue](#).

### 14.334.2 Member Typedef Documentation

#### 14.334.2.1 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](#).

### 14.334.3 Member Function Documentation

## 14.334.3.1 no\_notify() [1/2]

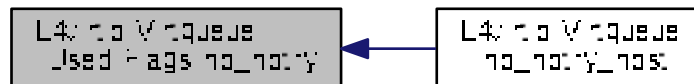
```
no_notify_bfm_t::Val L4virtio::Virtqueue::Used::Flags::no_notify () const [inline]
```

Get the *no\_notify* bits ( 0 to 0 ) of *raw* .

Definition at line 194 of file [virtqueue](#).

Referenced by [L4virtio::Virtqueue::no\\_notify\\_host\(\)](#).

Here is the caller graph for this function:



## 14.334.3.2 no\_notify() [2/2]

```
no_notify_bfm_t::Ref L4virtio::Virtqueue::Used::Flags::no_notify () [inline]
```

Get a reference to the *no\_notify* bits ( 0 to 0 ) of *raw* .

Definition at line 194 of file [virtqueue](#).

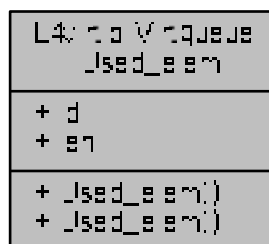
The documentation for this struct was generated from the following file:

- [l4/l4virtio/virtqueue](#)

## 14.335 L4virtio::Virtqueue::Used\_elem Struct Reference

Type of an element of the used ring.

Collaboration diagram for L4virtio::Virtqueue::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*

### 14.335.1 Detailed Description

Type of an element of the used ring.

Definition at line [160](#) of file [virtqueue](#).

### 14.335.2 Constructor & Destructor Documentation

#### 14.335.2.1 Used\_elem()

```
L4virtio::Virtqueue::Used_elem::Used_elem (
 l4_uint16_t id,
 l4_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](#).

The documentation for this struct was generated from the following file:

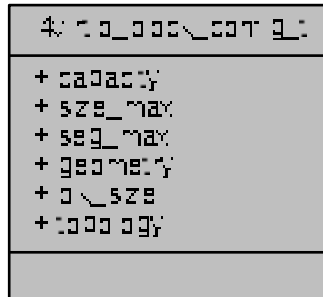
- [l4/l4virtio/virtqueue](#)

## 14.336 l4virtio\_block\_config\_t Struct Reference

Device configuration for block devices.

```
#include <virtio_block.h>
```

Collaboration diagram for l4virtio\_block\_config\_t:



## 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.*

## 14.336.1 Detailed Description

Device configuration for block devices.

Definition at line 63 of file [virtio\\_block.h](#).

## 14.336.2 Field Documentation

### 14.336.2.1 blk\_size

```
l4_uint32_t l4virtio_block_config_t::blk_size
```

Block size of underlying disk.

Definition at line 74 of file [virtio\\_block.h](#).

The documentation for this struct was generated from the following file:

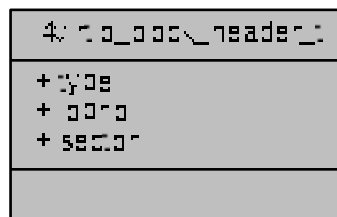
- l4/l4virtio/virtio\_block.h

## 14.337 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:



### 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.*

### 14.337.1 Detailed Description

Header structure of a request for a block device.

Definition at line 52 of file [virtio\\_block.h](#).

The documentation for this struct was generated from the following file:

- [l4/l4virtio/virtio\\_block.h](#)

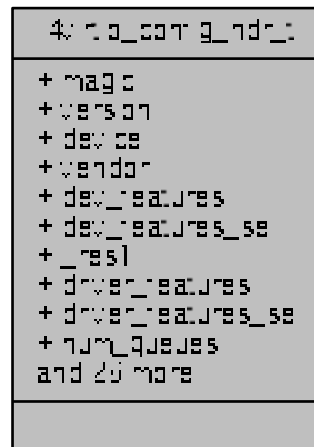
## 14.338 l4virtio\_config\_hdr\_t Struct Reference

L4-VIRTIO config header, provided in shared data space.

```
#include <virtio.h>
```

Inherited by L4virtio::Device::Config\_hdr.

Collaboration diagram for l4virtio\_config\_hdr\_t:



### 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*
- [l4\\_uint32\\_t status](#)  
*Device status register (read-only).*
- [l4\\_uint32\\_t cfg\\_driver\\_notify\\_index](#)  
*W: Event index to be used for config notifications (device to driver)*
- [l4\\_uint32\\_t cfg\\_device\\_notify\\_index](#)  
*R: Event index to be used for config notifications (driver to device)*
- [l4\\_uint32\\_t cmd](#)  
*L4 specific command register polled by the driver iff supported.*

### 14.338.1 Detailed Description

L4-VIRTIO config header, provided in shared data space.

Definition at line 127 of file [virtio.h](#).

### 14.338.2 Field Documentation

#### 14.338.2.1 magic

```
l4_uint32_t l4virtio_config_hdr_t::magic
```

magic value (must be 'virt').

Definition at line 129 of file [virtio.h](#).

#### 14.338.2.2 status

```
l4_uint32_t l4virtio_config_hdr_t::status
```

Device status register (read-only).

The register must be written using [l4virtio\\_set\\_status\(\)](#).

must be at offset 0x70 (virtio-mmio)

Definition at line 166 of file [virtio.h](#).

Referenced by [l4virtio\\_get\\_feature\(\)](#), [L4virtio::Svr::Dev\\_config::set\\_failed\(\)](#), and [L4virtio::Svr::Dev\\_config::set\\_status\(\)](#).

The documentation for this struct was generated from the following file:

- l4/l4virtio/virtio.h

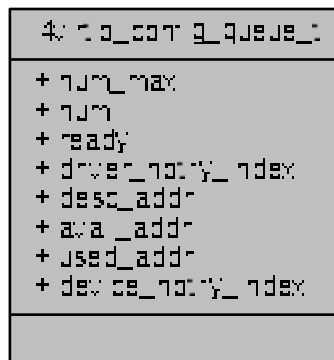


## 14.339 l4virtio\_config\_queue\_t Struct Reference

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)*

### 14.339.1 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 208 of file [virtio.h](#).

The documentation for this struct was generated from the following file:

- `l4/l4virtio/virtio.h`

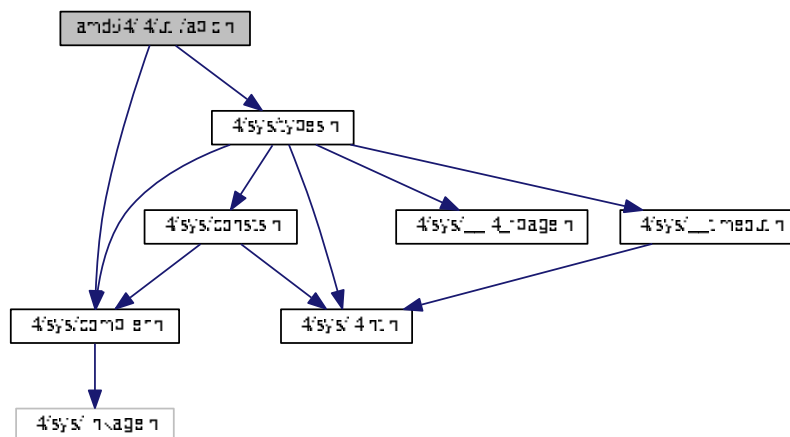
## Chapter 15

# File Documentation

### 15.1 amd64/l4/util/apic.h File Reference

APIC for AMD64.

```
#include <l4/sys/compiler.h>
#include <l4/sys/types.h>
Include dependency graph for apic.h:
```



#### 15.1.1 Detailed Description

APIC for AMD64.

Definition in file [apic.h](#).

## 15.2 apic.h

```

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 * 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 #ifndef __L4_UTIL_APIC_H
00014 #define __L4_UTIL_APIC_H
00015
00016 /*
00017 * Local APIC programming library
00018 *
00019 * For documentation, see
00020 *
00021 * "Intel Architecture Software Developer's Manual", Volume 3, chapter 7.5:
00022 * "Advanced Programmable Interrupt Controller (APIC)"
00023 *
00024 * Local APIC is present since
00025 * - INTEL P6 (PPro)
00026 * - AMD K7 (Athlon), Model 2
00027 *
00028 * In non-SMP-boards, local APIC is disabled, but
00029 * can be activated by writing to a MSR register.
00030 * For using APIC see packets cpufreq and l4rtl.
00031 *
00032 * See linux/include/asm-i386/i82489.h for further details.
00033 */
00034
00035 #define APIC_PHYS_BASE 0xFEE00000
00036 #define APIC_MAP_BASE 0xA0200000
00037 #define APIC_BASE_MSR 0x1b
00038
00039 #define APIC_ID 0x20
00040 #define GET_APIC_ID(x) (((x)>>24)&0x0F)
00041 #define APIC_LVR 0x30
00042 #define GET_APIC_VERSION(x) ((x)&0xFF)
00043 #define APIC_TASKPRI 0x80
00044 #define APIC_TPRI_MASK 0xFF
00045 #define APIC_EOI 0xB0
00046 #define APIC_LDR 0xD0
00047 #define APIC_LDR_MASK (0xFF<<24)
00048 #define APIC_DFR 0xE0
00049 #define SET_APIC_DFR(x) ((x)<<28)
00050 #define APIC_SPIV 0xF0
00051 #define APIC_LVTT 0x320
00052 #define APIC_LVTPC 0x340
00053 #define APIC_LVT0 0x350
00054 #define SET_APIC_TIMER_BASE(x) (((x)<<18))
00055 #define APIC_TIMER_BASE_DIV 0x2
00056 #define APIC_LVT1 0x360
00057 #define APIC_LVTERR 0x370
00058 #define APIC_TMICT 0x380
00059 #define APIC_TMCCT 0x390
00060 #define APIC_TDCR 0x3E0
00061
00062 #define APIC_LVT_MASKED (1<<16)
00063 #define APIC_LVT_TIMER_PERIODIC (1<<17)
00064 #define APIC_TDR_DIV_1 0xB
00065 #define APIC_TDR_DIV_2 0x0
00066 #define APIC_TDR_DIV_4 0x1
00067 #define APIC_TDR_DIV_8 0x2
00068 #define APIC_TDR_DIV_16 0x3
00069 #define APIC_TDR_DIV_32 0x8
00070 #define APIC_TDR_DIV_64 0x9
00071 #define APIC_TDR_DIV_128 0xA
00072
00073 #include <l4/sys/compiler.h>
00074 #include <l4/sys/types.h>
00075
00076 EXTERN_C_BEGIN
00077
00078 /* prototypes */
00079 extern unsigned long apic_map_base;
00080 extern unsigned long apic_timer_divisor;
00081
00082 extern unsigned long l4_scaler_apic_to_ms;
00083
00084 L4_CV void apic_show_registers(void);
00085 L4_CV int apic_check_working(void);
00086 L4_CV void apic_activate_by_io(void);
00087 L4_CV void apic_timer_set_divisor(int divisor);

```

```

00088
00089 L4_CV unsigned long l4_calibrate_apic(void);
00090
00091 EXTERN_C_END
00092
00093 L4_INLINE void apic_write(unsigned long reg, unsigned long v);
00094 L4_INLINE unsigned long apic_read(unsigned long reg);
00095 L4_INLINE void apic_activate_by_msr(void);
00096 L4_INLINE void apic_deactivate_by_msr(void);
00097 L4_INLINE unsigned long apic_read_phys_address(void);
00098 L4_INLINE int apic_test_present(void);
00099 L4_INLINE void apic_soft_enable(void);
00100 L4_INLINE void apic_init(unsigned long map_addr);
00101 L4_INLINE void apic_done(void);
00102 L4_INLINE void apic_irq_ack(void);
00103
00104 L4_INLINE void apic_lvt0_disable_irq(void);
00105 L4_INLINE void apic_lvt0_enable_irq(void);
00106 L4_INLINE void apic_lvt1_disable_irq(void);
00107 L4_INLINE void apic_lvt1_enable_irq(void);
00108
00109 L4_INLINE void apic_timer_write(unsigned long value);
00110 L4_INLINE unsigned long apic_timer_read(void);
00111 L4_INLINE void apic_timer_disable_irq(void);
00112 L4_INLINE void apic_timer_enable_irq(void);
00113 L4_INLINE void apic_timer_assign_irq(unsigned long vector);
00114 L4_INLINE void apic_timer_set_periodic(void);
00115 L4_INLINE void apic_timer_set_one_shot(void);
00116
00117 L4_INLINE void apic_perf_disable_irq(void);
00118 L4_INLINE void apic_perf_enable_irq(void);
00119 L4_INLINE void apic_perf_assign_irq(unsigned long vector);
00120
00121
00122 /* write APIC register */
00123 L4_INLINE void
00124 apic_write(unsigned long reg, unsigned long v)
00125 {
00126 *((volatile unsigned long *) (apic_map_base+reg))=v;
00127 }
00128
00129
00130 /* read APIC register */
00131 L4_INLINE unsigned long
00132 apic_read(unsigned long reg)
00133 {
00134 return *((volatile unsigned long *) (apic_map_base+reg));
00135 }
00136
00137
00138 /* disable LINT0 */
00139 L4_INLINE void
00140 apic_lvt0_disable_irq(void)
00141 {
00142 unsigned long tmp_val;
00143 tmp_val = apic_read(APIC_LVT0);
00144 tmp_val |= APIC_LVT_MASKED;
00145 apic_write(APIC_LVT0, tmp_val);
00146 }
00147
00148
00149 /* enable LINT0 */
00150 L4_INLINE void
00151 apic_lvt0_enable_irq(void)
00152 {
00153 unsigned long tmp_val;
00154 tmp_val = apic_read(APIC_LVT0);
00155 tmp_val &= ~(APIC_LVT_MASKED);
00156 apic_write(APIC_LVT0, tmp_val);
00157 }
00158
00159
00160 /* disable LINT1 */
00161 L4_INLINE void
00162 apic_lvt1_disable_irq(void)
00163 {
00164 unsigned long tmp_val;
00165 tmp_val = apic_read(APIC_LVT1);
00166 tmp_val |= APIC_LVT_MASKED;
00167 apic_write(APIC_LVT1, tmp_val);
00168 }
00169
00170
00171 /* enable LINT1 */
00172 L4_INLINE void
00173 apic_lvt1_enable_irq(void)
00174 {

```

```
00175 unsigned long tmp_val;
00176 tmp_val = apic_read(APIC_LVT1);
00177 tmp_val &= ~(APIC_LVT_MASKED);
00178 apic_write(APIC_LVT1, tmp_val);
00179 }
00180
00181
00182 /* write APIC timer register */
00183 L4_INLINE void
00184 apic_timer_write(unsigned long value)
00185 {
00186 apic_read(APIC_TMICT);
00187 apic_write(APIC_TMICT, value);
00188 }
00189
00190
00191 /* read APIC timer register */
00192 L4_INLINE unsigned long
00193 apic_timer_read(void)
00194 {
00195 return apic_read(APIC_TMCCT);
00196 }
00197
00198
00199 /* disable IRQ when APIC timer passes 0 */
00200 L4_INLINE void
00201 apic_timer_disable_irq(void)
00202 {
00203 unsigned long tmp_val;
00204 tmp_val = apic_read(APIC_LVTT);
00205 tmp_val |= APIC_LVT_MASKED;
00206 apic_write(APIC_LVTT, tmp_val);
00207 }
00208
00209
00210 /* enable IRQ when APIC timer passes 0 */
00211 L4_INLINE void
00212 apic_timer_enable_irq(void)
00213 {
00214 unsigned long tmp_val;
00215 tmp_val = apic_read(APIC_LVTT);
00216 tmp_val &= ~(APIC_LVT_MASKED);
00217 apic_write(APIC_LVTT, tmp_val);
00218 }
00219
00220
00221 L4_INLINE void
00222 apic_timer_set_periodic(void)
00223 {
00224 unsigned long tmp_val;
00225 tmp_val = apic_read(APIC_LVTT);
00226 tmp_val |= APIC_LVT_TIMER_PERIODIC;
00227 tmp_val |= APIC_LVT_MASKED;
00228 apic_write(APIC_LVTT, tmp_val);
00229 }
00230
00231
00232 L4_INLINE void
00233 apic_timer_set_one_shot(void)
00234 {
00235 unsigned long tmp_val;
00236 tmp_val = apic_read(APIC_LVTT);
00237 tmp_val &= ~APIC_LVT_TIMER_PERIODIC;
00238 tmp_val |= APIC_LVT_MASKED;
00239 apic_write(APIC_LVTT, tmp_val);
00240 }
00241
00242
00243 /* set vector of APIC timer irq */
00244 L4_INLINE void
00245 apic_timer_assign_irq(unsigned long vector)
00246 {
00247 unsigned long tmp_val;
00248 tmp_val = apic_read(APIC_LVTT);
00249 tmp_val &= 0xfffffff0;
00250 tmp_val |= vector;
00251 tmp_val |= APIC_LVT_MASKED;
00252 apic_write(APIC_LVTT, tmp_val);
00253 }
00254
00255
00256 /* disable IRQ when performance counter passes 0 */
00257 L4_INLINE void
00258 apic_perf_disable_irq(void)
00259 {
00260 unsigned long tmp_val;
00261 tmp_val = apic_read(APIC_LVTPC);
```

```

00262 tmp_val |= APIC_LVT_MASKED;
00263 apic_write(APIC_LVTPC, tmp_val);
00264 }
00265
00266
00267 /* enable IRQ when performance counter passes 0 */
00268 L4_INLINE void
00269 apic_perf_enable_irq(void)
00270 {
00271 unsigned long tmp_val;
00272 tmp_val = apic_read(APIC_LVTPC);
00273 tmp_val &= ~(APIC_LVT_MASKED);
00274 apic_write(APIC_LVTPC, tmp_val);
00275 }
00276
00277
00278 /* set vector of performance counter irq */
00279 L4_INLINE void
00280 apic_perf_assign_irq(unsigned long vector)
00281 {
00282 unsigned long tmp_val;
00283 tmp_val = apic_read(APIC_LVTPC);
00284 tmp_val &= 0xffffffff00;
00285 tmp_val |= vector;
00286 tmp_val |= APIC_LVT_MASKED;
00287 apic_write(APIC_LVTPC, tmp_val);
00288 }
00289
00290
00291 /* activate APIC by writing to appropriate MSR */
00292 L4_INLINE void
00293 apic_activate_by_msr(void)
00294 {
00295 unsigned long low;
00296 unsigned long high;
00297
00298 /* rdmsr */
00299 asm volatile(".byte 0xf; .byte 0x32\n"
00300 : "=a" (low),
00301 " =d" (high)
00302 : "c" (APIC_BASE_MSR)
00303);
00304
00305 low |= 0x800; /* activate APIC */
00306 low &= 0x00000fff;
00307 low |= (APIC_PHYS_BASE & 0xfffff000); /* set address */
00308
00309 /* wrmsr */
00310 asm volatile(".byte 0xf; .byte 0x30\n"
00311 :
00312 : "c" (APIC_BASE_MSR),
00313 "a" (low),
00314 "d" (high)
00315);
00316 }
00317
00318
00319 /* deactivate APIC by writing to appropriate MSR */
00320 L4_INLINE void
00321 apic_deactivate_by_msr(void)
00322 {
00323 unsigned long low;
00324 unsigned long high;
00325
00326 /* rdmsr */
00327 asm volatile(".byte 0xf; .byte 0x32\n"
00328 : "=a" (low),
00329 " =d" (high)
00330 : "c" (APIC_BASE_MSR)
00331);
00332
00333 low &= 0xfffff7ff; /* deactivate APIC */
00334
00335 /* wrmsr */
00336 asm volatile(".byte 0xf; .byte 0x30\n"
00337 :
00338 : "c" (APIC_BASE_MSR),
00339 "a" (low),
00340 "d" (high)
00341);
00342 }
00343
00344
00345 /* read memory mapped address of apic */
00346 L4_INLINE unsigned long
00347 apic_read_phys_address(void)
00348 {

```

```

00349 unsigned long low;
00350 unsigned long high;
00351
00352 /* rdmsr */
00353 asm volatile(".byte 0xf; .byte 0x32\n"
00354 : "=a" (low),
00355 : "d" (high)
00356 : "c" (APIC_BASE_MSR)
00357);
00358
00359 return (low &= 0xfffff000);
00360 }
00361
00362
00363 /* test if APIC present */
00364 L4_INLINE int
00365 apic_test_present(void)
00366 {
00367 unsigned int dummy;
00368 unsigned int capability;
00369
00370 asm volatile("pushl %%ebx ; cpuid ; popl %%ebx"
00371 : "=a" (dummy),
00372 : "c" (dummy),
00373 : "d" (capability)
00374 : "a" (0x00000001)
00375 : "cc");
00376
00377 return ((capability & 1<<9) !=0);
00378 }
00379
00380
00381 L4_INLINE void
00382 apic_soft_enable(void)
00383 {
00384 unsigned long tmp_val;
00385 tmp_val = apic_read(APIC_SPIV);
00386 tmp_val |= (1<<8); /* enable APIC */
00387 tmp_val &= ~(1<<9); /* enable Focus Processor Checking */
00388 tmp_val |= 0xff; /* Set spurious IRQ vector to 0xff */
00389 apic_write(APIC_SPIV, tmp_val);
00390 }
00391
00392
00393 L4_INLINE void
00394 apic_init(unsigned long base_addr)
00395 {
00396 apic_map_base = base_addr;
00397 }
00398
00399
00400 L4_INLINE void
00401 apic_done(void)
00402 {
00403 apic_map_base = 0;
00404 }
00405
00406
00407 L4_INLINE void
00408 apic_irq_ack(void)
00409 {
00410 apic_read(APIC_SPIV);
00411 apic_write(APIC_EOI, 0);
00412 }
00413
00414
00415 #endif /* __L4_UTIL_APIC_H */

```

## 15.3 x86/I4/util/apic.h File Reference

APIC for X86.

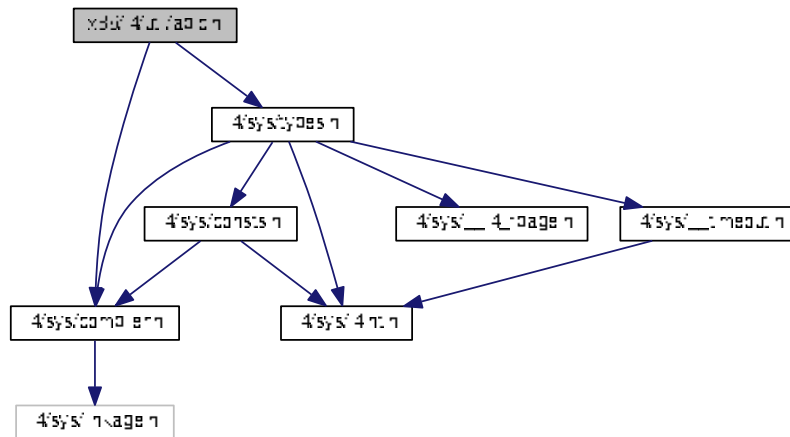
```

#include <l4/sys/compiler.h>
#include <l4/sys/types.h>

```



Include dependency graph for apic.h:



### 15.3.1 Detailed Description

APIC for X86.

Definition in file [apic.h](#).

## 15.4 apic.h

```

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 * 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 #ifndef __L4_UTIL_APIC_H
00014 #define __L4_UTIL_APIC_H
00015
00016 /*
00017 * Local APIC programming library
00018 *
00019 * For documentation, see
00020 *
00021 * "Intel Architecture Software Developer's Manual", Volume 3, chapter 7.5:
00022 * "Advanced Programmable Interrupt Controller (APIC)"
00023 *
00024 * Local APIC is present since
00025 * - INTEL P6 (PPro)
00026 * - AMD K7 (Athlon), Model 2
00027 *
00028 * In non-SMP-boards, local APIC is disabled, but
00029 * can be activated by writing to a MSR register.
00030 * For using APIC see packets cpufreq and l4rtl.
00031 *
00032 * See linux/include/asm-i386/i82489.h for further details.
00033 */
00034
00035 #define APIC_PHYS_BASE 0xFEE00000
00036 #define APIC_MAP_BASE 0xA0200000
00037 #define APIC_BASE_MSR 0x1b
00038
00039 #define APIC_ID 0x20

```

```

00040 #define GET_APIC_ID(x) ((x)>>24)&0x0F)
00041 #define APIC_LVR 0x30
00042 #define GET_APIC_VERSION(x) ((x)&0xFF)
00043 #define APIC_TASKPRI 0x80
00044 #define APIC_TPRI_MASK 0xFF
00045 #define APIC_EOI 0xB0
00046 #define APIC_LDR 0xD0
00047 #define APIC_LDR_MASK (0xFF<<24)
00048 #define APIC_DFR 0xE0
00049 #define SET_APIC_DFR(x) ((x)<<28)
00050 #define APIC_SPIV 0xF0
00051 #define APIC_LVTT 0x320
00052 #define APIC_LVTPC 0x340
00053 #define APIC_LVT0 0x350
00054 #define SET_APIC_TIMER_BASE(x) ((x)<<18)
00055 #define APIC_TIMER_BASE_DIV 0x2
00056 #define APIC_LVT1 0x360
00057 #define APIC_LVTERR 0x370
00058 #define APIC_TMICT 0x380
00059 #define APIC_TMCCT 0x390
00060 #define APIC_TDCR 0x3E0
00061
00062 #define APIC_LVT_MASKED (1<<16)
00063 #define APIC_LVT_TIMER_PERIODIC (1<<17)
00064 #define APIC_TDR_DIV_1 0xB
00065 #define APIC_TDR_DIV_2 0x0
00066 #define APIC_TDR_DIV_4 0x1
00067 #define APIC_TDR_DIV_8 0x2
00068 #define APIC_TDR_DIV_16 0x3
00069 #define APIC_TDR_DIV_32 0x8
00070 #define APIC_TDR_DIV_64 0x9
00071 #define APIC_TDR_DIV_128 0xA
00072
00073 #include <l4/sys/compiler.h>
00074 #include <l4/sys/types.h>
00075
00076 EXTERN_C_BEGIN
00077
00078 /* prototypes */
00079 extern unsigned long apic_map_base;
00080 extern unsigned long apic_timer_divisor;
00081
00082 extern unsigned long l4_scaler_apic_to_ms;
00083
00084 L4_CV void apic_show_registers(void);
00085 L4_CV int apic_check_working(void);
00086 L4_CV void apic_activate_by_io(void);
00087 L4_CV void apic_timer_set_divisor(int divisor);
00088
00089 L4_CV unsigned long l4_calibrate_apic(void);
00090
00091 EXTERN_C_END
00092
00093 L4_INLINE void apic_write(unsigned long reg, unsigned long v);
00094 L4_INLINE unsigned long apic_read(unsigned long reg);
00095 L4_INLINE void apic_activate_by_msr(void);
00096 L4_INLINE void apic_deactivate_by_msr(void);
00097 L4_INLINE unsigned long apic_read_phys_address(void);
00098 L4_INLINE int apic_test_present(void);
00099 L4_INLINE void apic_soft_enable(void);
00100 L4_INLINE void apic_init(unsigned long map_addr);
00101 L4_INLINE void apic_done(void);
00102 L4_INLINE void apic_irq_ack(void);
00103
00104 L4_INLINE void apic_lvt0_disable_irq(void);
00105 L4_INLINE void apic_lvt0_enable_irq(void);
00106 L4_INLINE void apic_lvt1_disable_irq(void);
00107 L4_INLINE void apic_lvt1_enable_irq(void);
00108
00109 L4_INLINE void apic_timer_write(unsigned long value);
00110 L4_INLINE unsigned long apic_timer_read(void);
00111 L4_INLINE void apic_timer_disable_irq(void);
00112 L4_INLINE void apic_timer_enable_irq(void);
00113 L4_INLINE void apic_timer_assign_irq(unsigned long vector);
00114 L4_INLINE void apic_timer_set_periodic(void);
00115 L4_INLINE void apic_timer_set_one_shot(void);
00116
00117 L4_INLINE void apic_perf_disable_irq(void);
00118 L4_INLINE void apic_perf_enable_irq(void);
00119 L4_INLINE void apic_perf_assign_irq(unsigned long vector);
00120
00121
00122 /* write APIC register */
00123 L4_INLINE void
00124 apic_write(unsigned long reg, unsigned long v)
00125 {
00126 *((volatile unsigned long *) (apic_map_base+reg))=v;

```

```

00127 }
00128
00129
00130 /* read APIC register */
00131 L4_INLINE unsigned long
00132 apic_read(unsigned long reg)
00133 {
00134 return *((volatile unsigned long *) (apic_map_base+reg));
00135 }
00136
00137
00138 /* disable LINT0 */
00139 L4_INLINE void
00140 apic_lvt0_disable_irq(void)
00141 {
00142 unsigned long tmp_val;
00143 tmp_val = apic_read(APIC_LVT0);
00144 tmp_val |= APIC_LVT_MASKED;
00145 apic_write(APIC_LVT0, tmp_val);
00146 }
00147
00148
00149 /* enable LINT0 */
00150 L4_INLINE void
00151 apic_lvt0_enable_irq(void)
00152 {
00153 unsigned long tmp_val;
00154 tmp_val = apic_read(APIC_LVT0);
00155 tmp_val &= ~(APIC_LVT_MASKED);
00156 apic_write(APIC_LVT0, tmp_val);
00157 }
00158
00159
00160 /* disable LINT1 */
00161 L4_INLINE void
00162 apic_lvt1_disable_irq(void)
00163 {
00164 unsigned long tmp_val;
00165 tmp_val = apic_read(APIC_LVT1);
00166 tmp_val |= APIC_LVT_MASKED;
00167 apic_write(APIC_LVT1, tmp_val);
00168 }
00169
00170
00171 /* enable LINT1 */
00172 L4_INLINE void
00173 apic_lvt1_enable_irq(void)
00174 {
00175 unsigned long tmp_val;
00176 tmp_val = apic_read(APIC_LVT1);
00177 tmp_val &= ~(APIC_LVT_MASKED);
00178 apic_write(APIC_LVT1, tmp_val);
00179 }
00180
00181
00182 /* write APIC timer register */
00183 L4_INLINE void
00184 apic_timer_write(unsigned long value)
00185 {
00186 apic_read(APIC_TMICT);
00187 apic_write(APIC_TMICT, value);
00188 }
00189
00190
00191 /* read APIC timer register */
00192 L4_INLINE unsigned long
00193 apic_timer_read(void)
00194 {
00195 return apic_read(APIC_TMCCT);
00196 }
00197
00198
00199 /* disable IRQ when APIC timer passes 0 */
00200 L4_INLINE void
00201 apic_timer_disable_irq(void)
00202 {
00203 unsigned long tmp_val;
00204 tmp_val = apic_read(APIC_LVTT);
00205 tmp_val |= APIC_LVT_MASKED;
00206 apic_write(APIC_LVTT, tmp_val);
00207 }
00208
00209
00210 /* enable IRQ when APIC timer passes 0 */
00211 L4_INLINE void
00212 apic_timer_enable_irq(void)
00213 {

```

```

00214 unsigned long tmp_val;
00215 tmp_val = apic_read(APIC_LVTT);
00216 tmp_val &= ~(APIC_LVT_MASKED);
00217 apic_write(APIC_LVTT, tmp_val);
00218 }
00219
00220
00221 L4_INLINE void
00222 apic_timer_set_periodic(void)
00223 {
00224 unsigned long tmp_val;
00225 tmp_val = apic_read(APIC_LVTT);
00226 tmp_val |= APIC_LVT_TIMER_PERIODIC;
00227 tmp_val |= APIC_LVT_MASKED;
00228 apic_write(APIC_LVTT, tmp_val);
00229 }
00230
00231
00232 L4_INLINE void
00233 apic_timer_set_one_shot(void)
00234 {
00235 unsigned long tmp_val;
00236 tmp_val = apic_read(APIC_LVTT);
00237 tmp_val &= ~APIC_LVT_TIMER_PERIODIC;
00238 tmp_val |= APIC_LVT_MASKED;
00239 apic_write(APIC_LVTT, tmp_val);
00240 }
00241
00242
00243 /* set vector of APIC timer irq */
00244 L4_INLINE void
00245 apic_timer_assign_irq(unsigned long vector)
00246 {
00247 unsigned long tmp_val;
00248 tmp_val = apic_read(APIC_LVTT);
00249 tmp_val &= 0xffffffff00;
00250 tmp_val |= vector;
00251 tmp_val |= APIC_LVT_MASKED;
00252 apic_write(APIC_LVTT, tmp_val);
00253 }
00254
00255
00256 /* disable IRQ when performance counter passes 0 */
00257 L4_INLINE void
00258 apic_perf_disable_irq(void)
00259 {
00260 unsigned long tmp_val;
00261 tmp_val = apic_read(APIC_LVTPC);
00262 tmp_val |= APIC_LVT_MASKED;
00263 apic_write(APIC_LVTPC, tmp_val);
00264 }
00265
00266
00267 /* enable IRQ when performance counter passes 0 */
00268 L4_INLINE void
00269 apic_perf_enable_irq(void)
00270 {
00271 unsigned long tmp_val;
00272 tmp_val = apic_read(APIC_LVTPC);
00273 tmp_val &= ~(APIC_LVT_MASKED);
00274 apic_write(APIC_LVTPC, tmp_val);
00275 }
00276
00277
00278 /* set vector of performance counter irq */
00279 L4_INLINE void
00280 apic_perf_assign_irq(unsigned long vector)
00281 {
00282 unsigned long tmp_val;
00283 tmp_val = apic_read(APIC_LVTPC);
00284 tmp_val &= 0xffffffff00;
00285 tmp_val |= vector;
00286 tmp_val |= APIC_LVT_MASKED;
00287 apic_write(APIC_LVTPC, tmp_val);
00288 }
00289
00290
00291 /* activate APIC by writing to appropriate MSR */
00292 L4_INLINE void
00293 apic_activate_by_msr(void)
00294 {
00295 unsigned long low;
00296 unsigned long high;
00297
00298 /* rdmsr */
00299 asm volatile(".byte 0xf; .byte 0x32\n"
00300 : "=a" (low),

```

```

00301 "d" (high)
00302 : "c" (APIC_BASE_MSR)
00303);
00304
00305 low |= 0x800; /* activate APIC */
00306 low &= 0x00000fff;
00307 low |= (APIC_PHYS_BASE & 0xfffff000); /* set address */
00308
00309 /* wrmsr */
00310 asm volatile(".byte 0xf; .byte 0x30\n"
00311 :
00312 : "c" (APIC_BASE_MSR),
00313 "a" (low),
00314 "d" (high)
00315);
00316 }
00317
00318
00319 /* deactivate APIC by writing to appropriate MSR */
00320 L4_INLINE void
00321 apic_deactivate_by_msr(void)
00322 {
00323 unsigned long low;
00324 unsigned long high;
00325
00326 /* rdmsr */
00327 asm volatile(".byte 0xf; .byte 0x32\n"
00328 : "=a" (low),
00329 "d" (high)
00330 : "c" (APIC_BASE_MSR)
00331);
00332
00333 low &= 0xfffff7ff; /* deactivate APIC */
00334
00335 /* wrmsr */
00336 asm volatile(".byte 0xf; .byte 0x30\n"
00337 :
00338 : "c" (APIC_BASE_MSR),
00339 "a" (low),
00340 "d" (high)
00341);
00342 }
00343
00344
00345 /* read memory mapped address of apic */
00346 L4_INLINE unsigned long
00347 apic_read_phys_address(void)
00348 {
00349 unsigned long low;
00350 unsigned long high;
00351
00352 /* rdmsr */
00353 asm volatile(".byte 0xf; .byte 0x32\n"
00354 : "=a" (low),
00355 "d" (high)
00356 : "c" (APIC_BASE_MSR)
00357);
00358
00359 return (low &= 0xfffff000);
00360 }
00361
00362
00363 /* test if APIC present */
00364 L4_INLINE int
00365 apic_test_present(void)
00366 {
00367 unsigned int dummy;
00368 unsigned int capability;
00369
00370 asm volatile("pushl %%ebx; cpuid; popl %%ebx"
00371 : "=a" (dummy),
00372 "c" (dummy),
00373 "d" (capability)
00374 : "a" (0x00000001)
00375 : "cc");
00376
00377 return ((capability & 1<<9) != 0);
00378 }
00379
00380
00381 L4_INLINE void
00382 apic_soft_enable(void)
00383 {
00384 unsigned long tmp_val;
00385 tmp_val = apic_read(APIC_SPIV);
00386 tmp_val |= (1<<8); /* enable APIC */
00387 tmp_val &= ~(1<<9); /* enable Focus Processor Checking */

```

```

00388 tmp_val |= 0xff; /* Set spurious IRQ vector to 0xff */
00389 apic_write(APIC_SPIV, tmp_val);
00390 }
00391
00392
00393 L4_INLINE void
00394 apic_init(unsigned long base_addr)
00395 {
00396 apic_map_base = base_addr;
00397 }
00398
00399
00400 L4_INLINE void
00401 apic_done(void)
00402 {
00403 apic_map_base = 0;
00404 }
00405
00406
00407 L4_INLINE void
00408 apic_irq_ack(void)
00409 {
00410 apic_read(APIC_SPIV);
00411 apic_write(APIC_EOI, 0);
00412 }
00413
00414
00415 #endif /* __L4_UTIL_APIC_H */

```

## 15.5 amd64/l4/util/idt.h File Reference

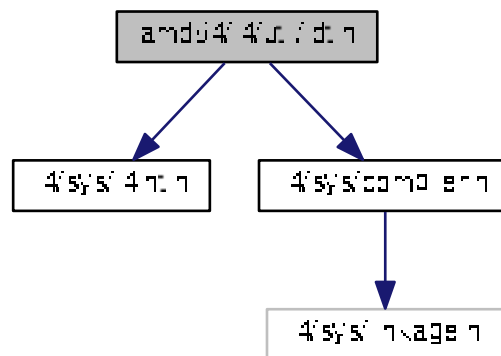
IDT related functions.

```

#include <l4/sys/l4int.h>
#include <l4/sys/compiler.h>

```

Include dependency graph for idt.h:



## Data Structures

- struct [l4util\\_idt\\_desc\\_t](#)  
*IDT entry.*
- struct [l4util\\_idt\\_header\\_t](#)  
*Header of an IDT table.*

### 15.5.1 Detailed Description

IDT related functions.

#### Date

2003

#### Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [idt.h](#).

## 15.6 idt.h

```

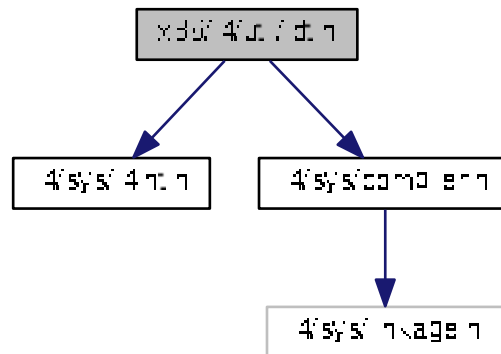
00001
00009 /*
00010 * (c) 2003-2009 Author(s)
00011 * economic rights: Technische Universität Dresden (Germany)
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU Lesser General Public License 2.1.
00014 * Please see the COPYING-LGPL-2.1 file for details.
00015 */
00016
00017 #ifndef __L4UTIL_IDT_H
00018 #define __L4UTIL_IDT_H
00019
00020 #include <l4/sys/l4int.h>
00021 #include <l4/sys/compiler.h>
00022
00023 EXTERN_C_BEGIN
00024
00030
00033 typedef struct
00034 {
00035 l4_uint64_t a, b;
00036 } __attribute__((packed)) l4util_idt_desc_t;
00037
00040 typedef struct
00041 {
00042 l4_uint16_t limit;
00043 void *base;
00044 l4util_idt_desc_t desc[0];
00045 } __attribute__((packed)) l4util_idt_header_t;
00046
00052 static inline void
00053 l4util_idt_entry(l4util_idt_header_t *idt, int nr, void(*handler)(void))
00054 {
00055 idt->desc[nr].a = (l4_uint64_t)handler & 0x0000ffff;
00056 idt->desc[nr].b = 0x0000ef00 | ((l4_uint64_t)handler & 0xffff0000);
00057 }
00058
00063 static inline void
00064 l4util_idt_init(l4util_idt_header_t *idt, int entries)
00065 {
00066 int i;
00067 idt->limit = entries*8 - 1;
00068 idt->base = &idt->desc;
00069
00070 for (i=0; i<entries; i++)
00071 l4util_idt_entry(idt, i, 0);
00072 }
00073
00077 static inline void
00078 l4util_idt_load(l4util_idt_header_t *idt)
00079 {
00080 asm volatile ("lidt (%rax) \n\t" : : "a" (idt));
00081 }
00083 EXTERN_C_END
00084
00085 #endif
00086

```

## 15.7 x86/l4/util/idt.h File Reference

IDT related functions.

```
#include <l4/sys/l4int.h>
#include <l4/sys/compiler.h>
Include dependency graph for idt.h:
```



### Data Structures

- struct [l4util\\_idt\\_desc\\_t](#)  
*IDT entry.*
- struct [l4util\\_idt\\_header\\_t](#)  
*Header of an IDT table.*

### 15.7.1 Detailed Description

IDT related functions.

Date

2003

Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [idt.h](#).



## 15.8 idt.h

```

00001
00009 /*
00010 * (c) 2003-2009 Author(s)
00011 * economic rights: Technische Universität Dresden (Germany)
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU Lesser General Public License 2.1.
00014 * Please see the COPYING-LGPL-2.1 file for details.
00015 */
00016
00017 #ifndef __L4UTIL_IDT_H
00018 #define __L4UTIL_IDT_H
00019
00020 #include <l4/sys/l4int.h>
00021 #include <l4/sys/compiler.h>
00022
00023 EXTERN_C_BEGIN
00024
00030
00033 typedef struct
00034 {
00035 l4_uint32_t a, b;
00036 } __attribute__((packed)) l4util_idt_desc_t;
00037
00040 typedef struct
00041 {
00042 l4_uint16_t limit;
00043 void *base;
00044 l4util_idt_desc_t desc[0];
00045 } __attribute__((packed)) l4util_idt_header_t;
00046
00052 static inline void
00053 l4util_idt_entry(l4util_idt_header_t *idt, int nr, void(*handler)(void))
00054 {
00055 idt->desc[nr].a = (l4_uint32_t)handler & 0x0000ffff;
00056 idt->desc[nr].b = 0x0000ef00 | ((l4_uint32_t)handler & 0xffff0000);
00057 }
00058
00063 static inline void
00064 l4util_idt_init(l4util_idt_header_t *idt, int entries)
00065 {
00066 int i;
00067 idt->limit = entries*8 - 1;
00068 idt->base = &idt->desc;
00069
00070 for (i=0; i<entries; i++)
00071 l4util_idt_entry(idt, i, 0);
00072 }
00073
00077 static inline void
00078 l4util_idt_load(l4util_idt_header_t *idt)
00079 {
00080 asm volatile ("lidt (%eax) \n\t" : : "a" (idt));
00081 }
00082
00084 EXTERN_C_END
00085
00086 #endif
00087

```

## 15.9 amd64/l4/util/perform.h File Reference

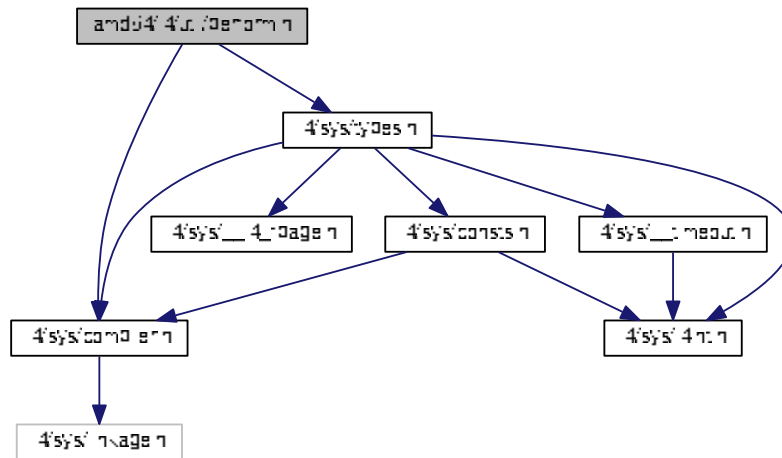
Performance Monitoring using P5/P6 Measurement Counters.

```

#include <l4/sys/types.h>
#include <l4/sys/compiler.h>

```

Include dependency graph for `perform.h`:



### 15.9.1 Detailed Description

Performance Monitoring using P5/P6 Measurement Counters.

Define either `CPU_PENTIUM` or `CPU_P6`

Definition in file [perform.h](#).

## 15.10 `perform.h`

```

00001
00007 /*
00008 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009 * Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010 * economic rights: Technische Universität Dresden (Germany)
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU Lesser General Public License 2.1.
00013 * Please see the COPYING-LGPL-2.1 file for details.
00014 */
00015 #ifndef __L4UTIL_PERFORM_H
00016 #define __L4UTIL_PERFORM_H
00017
00018 #include <14/sys/types.h>
00019 #include <14/sys/compiler.h>
00020
00021 EXTERN_C_BEGIN
00022
00023 extern const char*strp6pmc_event(14_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 %%rax, %%rax\n"
00081 "xor %%rdx, %%rdx\n"
00082 "mov $0x12, %%rcx\n"
00083 ".byte 0x0f, 0x30\n"
00084 "movl $0x13, %%rcx\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 /*
00095 *movl $0, %%eax\n"
00096 *movl $0x11, %%ecx\n"
00097 *.byte 0x0f, 0x30\n" */ /* stop event counting */
00098 "mov $0x12, %%rcx\n"
00099 ".byte 0x0f, 0x32\n"
00100 "mov %%rax, (%%%rbx)\n"
00101 "mov %%rdx, 4(%%rbx)\n"
00102 "mov $0x13, %%ecx\n"
00103 ".byte 0x0f, 0x32\n"
00104 "mov %%rax, (%%rsi)\n"
00105 "mov %%rdx, 4(%%rsi)\n"
00106 : /* no output */
00107 : "b" (counter0), "S" (counter1)
00108 : "ax", "cx", "dx"
00109);
00110 }
00111
00112 static inline void
00113 l4_i586_read_event_counter(int *counter0, int *counter1)
00114 {
00115 asm volatile("push %%rdx\n"
00116 ".byte 0x0f, 0x30\n"
00117 "mov $0x12, %%rcx\n"
00118 ".byte 0x0f, 0x32\n"
00119 "mov %%rax, %%rbx\n"
00120 "movl $0x13, %%rcx\n"
00121 ".byte 0x0f, 0x32\n"
00122 "popl %%edx\n"
00123 : "=b" (*counter0), "=a" (*counter1)
00124 : "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 /* ($) 12 ifetches */
00246 #define P6_L2_LD 0x29 /* ($) 12 data loads */
00247 #define P6_L2_ST 0x2a /* ($) 12 data stores */
00248 #define P6_L2_LINES_IN 0x24 /* lines allocated in L2 */
00249 #define P6_L2_LINES_OUT 0x26 /* lines removed from L2 */
00250 #define P6_L2_M_LINES_INM 0x25 /* modified lines allocated in L2 */
00251 #define P6_L2_M_LINES_OUTM 0x27 /* modified lines removed from L2 */
00252 #define P6_L2_RQSTS 0x2e /* ($) number of 12 requests */
00253 #define P6_L2_ADS 0x21 /* number of 12 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 12->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 */
00295 #define P6_INST_DECODER 0xd0 /* number of instructions decoded */
00296
00297 /* Interrupts */

```



```

00385
00386 static inline l4_uint32_t l4_i686_rdpmc_32(int cntnr){
00387 l4_uint32_t x;
00388
00389 __asm__ __volatile__(
00390 ".byte 0xf; .byte 0x33 # RDPMC instruction"
00391 : "=a" (x)
00392 : "c" (cntnr)
00393 : "rcx", "rax", "rdx");
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_EVTSEL0+counter, &val);
00400 }
00401
00402 static inline void l4_i686_select_perfctr0_event(long long *val){
00403 __asm volatile(
00404 "mov $MSR_P6_EVTSEL0, %%rcx\n"
00405 "mov (%%rbx), %%rax\n"
00406 "mov 4(%%rbx), %%rdx\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_EVTSEL0 0xC0010000
00430 #define MSR_K7_EVTSEL1 0xC0010001
00431 #define MSR_K7_EVTSEL2 0xC0010002
00432 #define MSR_K7_EVTSEL3 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 EXTERN_C_END
00449
00450 #endif

```

## 15.11 x86/I4/util/perform.h File Reference

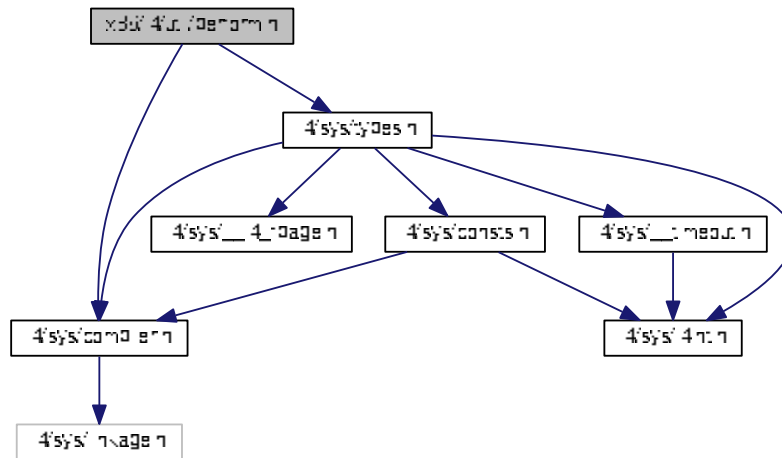
Performance Monitoring using P5/P6 Measurement Counters.

```

#include <l4/sys/types.h>
#include <l4/sys/compiler.h>

```

Include dependency graph for `perform.h`:



### 15.11.1 Detailed Description

Performance Monitoring using P5/P6 Measurement Counters.

Define either `CPU_PENTIUM` or `CPU_P6`

Definition in file [perform.h](#).

## 15.12 `perform.h`

```

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 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU Lesser General Public License 2.1.
00014 * Please see the COPYING-LGPL-2.1 file for details.
00015 */
00016
00017 #ifndef __L4UTIL_PERFORM_H
00018 #define __L4UTIL_PERFORM_H
00019
00020 #include <l4/sys/types.h>
00021 #include <l4/sys/compiler.h>
00022
00023 EXTERN_C_BEGIN
00024
00025 extern const char*strp6pmc_event(l4_uint32_t event);
00026
00027 #ifndef CONFIG_PERFORM_ONLY_PROTOTYPES
00028
00029 #if ! (defined CPU_PENTIUM ^ defined CPU_P6 ^ defined CPU_K7)
00030
00031 #error You must define your target architecture.
00032 #error Define EITHER CPU_PENTIUM for Intel Pentium or CPU_P6 for Intel PPro/PII/PIII.
00033
00034 #else
00035
00036 /* P5/P6/K7 section */

```



```

00037
00038 /* Makros for access to model specific registers (MSR) */
00039
00040 /* Write the 64-Bit Model Specific Register. First argument is the register,
00041 second the 64-Bit value. This can only be called at privilege level 0.
00042 With L4, the kernel emulates the WRMSR when calling in PL 3.
00043 */
00044 static inline void l4_i586_wrmsr(unsigned reg,unsigned long long*val){
00045 unsigned long dummyeax, dummyecx, dummyedx;
00046
00047 asm volatile(
00048 ".byte 0xf; .byte 0x30\n" /* wrmsr */
00049 : "=a" (dummyeax), "=d" (dummyedx), "=c" (dummyecx)
00050 : "2" (reg), "0" (*(unsigned *)val), "1" (*(unsigned *)val+1))
00051);
00052 }
00053
00054 /* Read the 64-Bit Model Specific Register. First argument is the register,
00055 second the address to a 64-Bit value. This can only be called at
00056 privilege level 0. With L4, the kernel emulates the RDMSR when calling
00057 in PL 3.
00058 */
00059 static inline void l4_i586_rdmsr(unsigned reg,unsigned long long*val){
00060 unsigned dummy;
00061
00062 asm volatile(
00063 ".byte 0xf; .byte 0x32\n" /* rdmsr */
00064 : "=a" (*(unsigned *)val), "=d" (*(unsigned *)val+1), "=c" (dummy)
00065 : "2" (reg)
00066);
00067 }
00068
00069
00070 #ifdef CPU_PENTIUM
00071 /* Pentium section */
00072
00073 /* functions and events defined here are only usable at Pentium
00074 Processors. P6 architecture does NOT support this kind of measuring and
00075 these events. P6 architecture has its own counters and its own events.
00076 See P6-section for details. */
00077
00078 /* from l4linux/arch/l4-i386/include/perform.h */
00079
00080 static inline void
00081 l4_i586_reset_event_counter(void){
00082 asm volatile("xor %%eax, %%eax\n"
00083 "xor %%edx, %%edx\n"
00084 "movl $0x12, %%ecx\n"
00085 ".byte 0x0f, 0x30\n"
00086 "movl $0x13, %%ecx\n"
00087 ".byte 0x0f, 0x30\n"
00088 : : : "cx", "ax", "dx"
00089);
00090 };
00091
00092 static inline void
00093 l4_i586_read_event_counter_long(long long *counter0, long long *counter1)
00094 {
00095 asm volatile(
00096 /* "movl $0, %%eax\n"
00097 "movl $0x11, %%ecx\n"
00098 ".byte 0x0f, 0x30\n" *//* stop event counting */
00099 "movl $0x12, %%ecx\n"
00100 ".byte 0x0f, 0x32\n"
00101 "movl %%eax, (%ebx)\n"
00102 "movl %%edx, 4(%ebx)\n"
00103 "movl $0x13, %%ecx\n"
00104 ".byte 0x0f, 0x32\n"
00105 "movl %%eax, (%esi)\n"
00106 "movl %%edx, 4(%esi)\n"
00107 : /* no output */
00108 : "b" (counter0), "S" (counter1)
00109 : "ax", "cx", "dx"
00110);
00111 }
00112
00113 static inline void
00114 l4_i586_read_event_counter(int *counter0, int *counter1)
00115 {
00116 asm volatile("pushl %%edx\n"
00117 ".byte 0x0f, 0x30\n"
00118 "movl $0x12, %%ecx\n"
00119 ".byte 0x0f, 0x32\n"
00120 "movl %%eax, %%ebx\n"
00121 "movl $0x13, %%ecx\n"
00122 ".byte 0x0f, 0x32\n"
00123 "popl %%edx\n"

```

```

00124 : "b" (*counter0), "a" (*counter1)
00125 : "1" (0), "c" (0x11)
00126);
00127 }
00128
00129 static inline void
00130 l4_i586_select_event(int event0, int event1)
00131 {
00132 asm volatile(".byte 0x0f, 0x30\n"
00133 :
00134 :
00135 "a" (event0 + (event1 << 16)),
00136 "d" (0),
00137 "c" (0x11)
00138);
00139 };
00140
00141 #define P5_RD_MISS 0x003 /* 000011B */
00142 #define P5_WR_MISS 0x008 /* 000100B */
00143 #define P5_RW_MISS 0x029 /* 101001B */
00144 #define P5_EX_MISS 0x00e /* 001110B */
00145
00146 #define P5_D_WBACK 0x006 /* 000110B */
00147
00148 #define P5_RW_TLB 0x002 /* 00010B */
00149 #define P5_EX_TLB 0x00d /* 01101B */
00150
00151 #define P5_A_STALL 0x01f /* 11111B */
00152 #define P5_W_STALL 0x019 /* 11001B */
00153 #define P5_R_STALL 0x01a /* 11010B */
00154 #define P5_X_STALL 0x01b /* 11011B */
00155
00156 #define P5_AGI_STALL 0x01f /* 11111B */
00157
00158 #define P5_PIPELINE_FLUSH 0x015 /* 10101B */
00159
00160 #define P5_NON_CACHE_RD 0x01e /* 11110B */
00161 #define P5_NCACHE_REFS 0x01e /* 11110B */
00162 #define P5_LOCKED_BUS 0x01c /* 11100B */
00163
00164 #define P5_MEM2PIPE 0x009 /* 01001B */
00165 #define P5_BANK_CONF 0x00a /* 01010B */
00166
00167
00168 #define P5_INSTRS_EX 0x016 /* 10110B */
00169 #define P5_INSTRS_EX_V 0x017 /* 10111B */
00170
00171
00172 #define P5_CNT_NOTHING (0x00 << 6) /* 00B << 6 */
00173 #define P5_CNT_EVENT_PL0 (0x01 << 6) /* 01B << 6 */
00174 #define P5_CNT_EVENT_PL3 (0x02 << 6) /* 10B << 6 */
00175 #define P5_CNT_EVENT (0x03 << 6) /* 11B << 6 */
00176 #define P5_CNT_CLOCKS_PL0 (0x05 << 6) /* 101B << 6 */
00177 #define P5_CNT_CLOCKS_PL3 (0x06 << 6) /* 110B << 6 */
00178 #define P5_CNT_CLOCKS (0x07 << 6) /* 111B << 6 */
00179
00180
00181 #else
00182 #if defined CPU_P6
00183 /* PPro/PII/PIII section */
00184
00185 /*-
00186 * Copyright (c) 1997 The President and Fellows of Harvard College.
00187 * All rights reserved.
00188 * Copyright (c) 1997 Aaron B. Brown.
00189 *
00190 * Redistribution and use in source and binary forms, with or without
00191 * modification, are permitted provided that the following conditions
00192 * are met:
00193 * 1. Redistributions of source code must retain the above copyright
00194 * notice, this list of conditions and the following disclaimer.
00195 * 2. Redistributions in binary form must reproduce the above copyright
00196 * notice, this list of conditions and the following disclaimer in the
00197 * documentation and/or other materials provided with the distribution.
00198 * 3. All advertising materials mentioning features or use of this software
00199 * must display the following acknowledgement:
00200 * This product includes software developed by Harvard University
00201 * and its contributors.
00202 * 4. Neither the name of the University nor the names of its contributors
00203 * may be used to endorse or promote products derived from this software
00204 * without specific prior written permission.
00205 *
00206 * THIS SOFTWARE IS PROVIDED BY HARVARD AND CONTRIBUTORS ``AS IS'' AND
00207 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
00208 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
00209 * ARE DISCLAIMED. IN NO EVENT SHALL HARVARD UNIVERSITY OR CONTRIBUTORS BE
00210 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR

```

```

00211 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
00212 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
00213 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
00214 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
00215 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
00216 * POSSIBILITY OF SUCH DAMAGE.
00217 */
00218
00219 /*****
00220 ** Symbolic names for counter numbers (used in select_p6counter()) **
00221 *****/
00222 *
00223 * These correspond in order to the Pentium Pro counters. Add new counters at
00224 * the end. These agree with the mnemonics in the Pentium Pro Family
00225 * Developer's Manual, vol 3.
00226 *
00227 * Those events marked with a $ require a MESI unit field; those marked with
00228 * a @ require a self/any unit field. Those marked with a 0 are only supported
00229 * in counter 0; those marked with 1 are only supported in counter 1.
00230 */
00231
00232 /* Data cache unit */
00233 #define P6_DATA_MEM_REFS 0x43 /* total memory refs */
00234 #define P6_DCU_LINES_IN 0x45 /* all lines allocated in cache unit */
00235 #define P6_DCU_M_LINES_IN 0x46 /* M lines allocated in cache unit */
00236 #define P6_DCU_M_LINES_OUT 0x47 /* M lines evicted from cache */
00237 #define P6_DCU_MISS_OUTSTANDING 0x48 /* #cycles a miss is outstanding */
00238
00239 /* Instruction fetch unit */
00240 #define P6_IFU_IFETCH 0x80 /* instruction fetches */
00241 #define P6_IFU_IFETCH_MISS 0x81 /* instruction fetch misses */
00242 #define P6_ITLB_MISS 0x85 /* ITLB misses */
00243 #define P6_IFU_MEM_STALL 0x86 /* number of cycles IFU is stalled */
00244 #define P6_ILD_STALL 0x87 /* #stalls in instr length decode */
00245
00246 /* L2 Cache */
00247 #define P6_L2_IFETCH 0x28 /* ($) L2 ifetches */
00248 #define P6_L2_LD 0x29 /* ($) L2 data loads */
00249 #define P6_L2_ST 0x2a /* ($) L2 data stores */
00250 #define P6_L2_LINES_IN 0x24 /* lines allocated in L2 */
00251 #define P6_L2_LINES_OUT 0x26 /* lines removed from L2 */
00252 #define P6_L2_M_LINES_INM 0x25 /* modified lines allocated in L2 */
00253 #define P6_L2_M_LINES_OUTM 0x27 /* modified lines removed from L2 */
00254 #define P6_L2_RQSTS 0x2e /* ($) number of L2 requests */
00255 #define P6_L2_ADS 0x21 /* number of L2 addr strobes */
00256 #define P6_L2_DBUS_BUSY 0x22 /* number of data bus busy cycles */
00257 #define P6_L2_DBUS_BUSY_RD 0x23 /* #bus cycles xferring L2->CPU */
00258
00259 /* External bus logic */
00260 #define P6_BUS_DRDY_CLOCKS 0x62 /* (@) #clocks DRDY is asserted */
00261 #define P6_BUS_LOCK_CLOCKS 0x63 /* (@) #clocks LOCK is asserted */
00262 #define P6_BUS_REQ_OUTSTANDING 0x60 /* #bus requests outstanding */
00263 #define P6_BUS_TRAN_BRD 0x65 /* (@) bus burst read txns */
00264 #define P6_BUS_TRAN_RFO 0x66 /* (@) bus read for ownership txns */
00265 #define P6_BUS_TRAN_WB 0x67 /* (@) bus writeback txns */
00266 #define P6_BUS_TRAN_IFETCH 0x68 /* (@) bus instr fetch txns */
00267 #define P6_BUS_TRAN_INVALID 0x69 /* (@) bus invalidate txns */
00268 #define P6_BUS_TRAN_PWR 0x6a /* (@) bus partial write txns */
00269 #define P6_BUS_TRANS_P 0x6b /* (@) bus partial txns */
00270 #define P6_BUS_TRANS_IO 0x6c /* (@) bus I/O txns */
00271 #define P6_BUS_TRAN_DEF 0x6d /* (@) bus deferred txns */
00272 #define P6_BUS_TRAN_BURST 0x6e /* (@) bus burst txns */
00273 #define P6_BUS_TRAN_ANY 0x70 /* (@) total bus txns */
00274 #define P6_BUS_TRAN_MEM 0x6f /* (@) total memory txns */
00275 #define P6_BUS_DATA_RCV 0x64 /* #busclocks CPU is receiving data */
00276 #define P6_BUS_BNR_DRV 0x61 /* #busclocks CPU is driving BNR pin */
00277 #define P6_BUS_HIT_DRV 0x7a /* #busclocks CPU is driving HIT pin */
00278 #define P6_BUS_HITM_DRV 0x7b /* #busclocks CPU is driving HITM pin */
00279 #define P6_BUS_SNOOP_STALL 0x7e /* #clkcycles bus is snoop-stalled */
00280
00281 /* FPU */
00282 #define P6_FLOPS 0xc1 /* (0) number of FP ops retired */
00283 #define P6_FP_COMP_OPS 0x10 /* (0) computational FPOPS exec'd */
00284 #define P6_FP_ASSIST 0x11 /* (1) FP excep's handled in ucode */
00285 #define P6_MUL 0x12 /* (1) number of FP multiplies */
00286 #define P6_DIV 0x13 /* (1) number of FP divides */
00287 #define P6_CYCLES_DIV_BUSY 0x14 /* (0) number of cycles divider busy */
00288
00289 /* Memory ordering */
00290 #define P6_LD_BLOCKS 0x03 /* number of store buffer blocks */
00291 #define P6_SB_DRAINS 0x04 /* # of store buffer drain cycles */
00292 #define P6_MISALING_MEM_REF 0x05 /* # misaligned data memory refs */
00293
00294 /* Instruction decoding and retirement */
00295 #define P6_INST_RETIRED 0xc0 /* number of instrs retired */
00296 #define P6_UOPS_RETIRED 0xc2 /* number of micro-ops retired */
00297 #define P6_INST_DECODER 0xd0 /* number of instructions decoded */

```

```

00298
00299 /* Interrupts */
00300 #define P6_HW_INT_RX 0xc8 /* number of hardware interrupts */
00301 #define P6_CYCLES_INT_MASKED 0xc6 /* number of cycles hardints masked */
00302 #define P6_CYCLES_INT_PENDING_AND_MASKED 0xc7 /* #cycles masked but pending */
00303
00304 /* Branches */
00305 #define P6_BR_INST_RETIRED 0xc4 /* number of branch instrs retired */
00306 #define P6_BR_MISS_PRED_RETIRED 0xc5 /* number of mispred'd brs retired */
00307 #define P6_BR_TAKEN_RETIRED 0xc9 /* number of taken branches retired */
00308 #define P6_BR_MISS_PRED_TAKEN_RET 0xca /* #taken mispredictions br's retired*/
00309 #define P6_BR_INST_DECODED 0xe0 /* number of branch instrs decoded */
00310 #define P6_BTBMISSES 0xe2 /* # of branches that missed in BTB */
00311 #define P6_BR_BOGUS 0xe4 /* number of bogus branches */
00312 #define P6_BACLEAR 0xe6 /* # times BACLEAR is asserted */
00313
00314 /* Stalls */
00315 #define P6_RESOURCE_STALLS 0xa2 /* # resource-related stall cycles */
00316 #define P6_PARTIAL_RAT_STALLS 0xd2 /* # cycles/events for partial stalls*/
00317
00318 /* Segment register loads */
00319 #define P6_SEGMENT_REG_LOADS 0x06 /* number of segment register loads */
00320
00321 /* Clocks */
00322 #define P6_CPU_CLK_UNHALTED 0x79 /* #clocks CPU is not halted */
00323
00324 /* Unit field tags */
00325 #define P6_UNIT_M 0x0800
00326 #define P6_UNIT_E 0x0400
00327 #define P6_UNIT_S 0x0200
00328 #define P6_UNIT_I 0x0100
00329 #define P6_UNIT_MESI 0x0f00
00330
00331 #define P6_UNIT_SELF 0x0000
00332 #define P6_UNIT_ANY 0x2000
00333
00334 /*****
00335 ** Flag bit definitions (used for the 'flag' field in select_p6counter()) **
00336 *****/
00337 *
00338 * The driver accepts fully-formed counter specifications from user-level.
00339 * The following flags are mnemonics for the bits that get set in the
00340 * PerfEvtSel0 and PerfEvtSel1 MSR's
00341 *
00342 */
00343 #define P6CNT_U 0x010000 /* Monitor user-level events */
00344 #define P6CNT_K 0x020000 /* Monitor kernel-level events */
00345 #define P6CNT_E 0x040000 /* Edge detect: count state transitions */
00346 #define P6CNT_PC 0x080000 /* Pin control: ?? */
00347 #define P6CNT_IE 0x100000 /* Int enable: enable interrupt on overflow */
00348 #define P6CNT_F 0x200000 /* Freeze counter (handled in software) */
00349 #define P6CNT_EN 0x400000 /* enable counters (in PerfEvtSel0) */
00350 #define P6CNT_IV 0x800000 /* Invert counter mask comparison result */
00351
00352 /*****
00353 ** Miscellaneous constants **
00354 *****/
00355 *
00356 * Number of Pentium Pro programable hardware counters.
00357 */
00358 #define NUM_P6HWC 2
00359
00360 /*****
00361 *
00362 * End of Copyright by Harvard College
00363 *
00364 *****/
00365
00366
00367 #define MSR_P6_EVTSEL0 0x186
00368 #define MSR_P6_EVTSEL1 0x187
00369 #define MSR_P6_PERFCTR0 0xc1
00370 #define MSR_P6_PERFCTR1 0xc2
00371
00372 /* P6-specific Makros to manipulate and read counters */
00373
00374 /* Read the 40 bit performance monitoring counter. This requires
00375 the PCE-flag in CR4 to be set. Otherwise GP0 is raised. Works only
00376 at P6.
00377 */
00378 #define l4_i686_rdpmc(cnr, res_p) \
00379 __asm __volatile(\
00380 "movl %2, %%ecx # put counter number in \n\
00381 .byte 0xf; .byte 0x33 # RDPMC instruction \n\
00382 movl %%edx, %1 # High order 32 bits \n\
00383 movl %%eax, %0 # Low order 32 bits" \
00384 : "=g" (*(int *) (res_p)), "=g" (((int *) res_p)+1)) \

```

```

00385 : "g" (cnt) \
00386 : "ecx", "eax", "edx")
00387
00388 static inline l4_uint32_t l4_i686_rdtsc_32(int cnt){
00389 l4_uint32_t x;
00390
00391 __asm__ __volatile__(
00392 ".byte 0xf; .byte 0x33 # RDTSC instruction"
00393 : "=a" (x)
00394 : "c" (cnt)
00395 : "ecx", "eax", "edx");
00396 return x;
00397 }
00398
00399 static inline void l4_i686_select_perfctr_event(int counter,
00400 unsigned long long val){
00401 l4_i586_wrmsr(MSR_P6_EVNTSEL0+counter, &val);
00402 }
00403
00404 static inline void l4_i686_select_perfctr0_event(long long *val){
00405 __asm__ volatile(
00406 "movl $MSR_P6_EVNTSEL0, %%ecx\n"
00407 "movl (%ebx), %%eax\n"
00408 "movl 4(%%ebx), %%edx\n"
00409 /* ".byte 0xcc, 0xeb, 0x01, 0x21\n"
00410 ".byte 0x0f, 0x30\n" // wrmsr
00411 /* ".byte 0xcc, 0xeb, 0x01, 0x21\n"
00412 : /* no output */
00413 : "b" (val)
00414 : "ax", "cx", "dx", "bx"
00415);
00416
00417 }
00418
00419 /* end of P6 section */
00420 #else
00421
00422 #define K7CNT_U 0x010000 /* Monitor user-level events */
00423 #define K7CNT_K 0x020000 /* Monitor kernel-level events */
00424 #define K7CNT_E 0x040000 /* Edge detect: count state transitions */
00425 #define K7CNT_PC 0x080000 /* Pin control: ?? */
00426 #define K7CNT_IE 0x100000 /* Int enable: enable interrupt on overflow */
00427 #define K7CNT_F 0x200000 /* Freeze counter (handled in software) */
00428 #define K7CNT_EN 0x400000 /* enable counters (in PerfEvtSel0) */
00429 #define K7CNT_IV 0x800000 /* Invert counter mask comparison result */
00430
00431 #define MSR_K7_EVNTSEL0 0xC0010000
00432 #define MSR_K7_EVNTSEL1 0xC0010001
00433 #define MSR_K7_EVNTSEL2 0xC0010002
00434 #define MSR_K7_EVNTSEL3 0xC0010003
00435 #define MSR_K7_PERFCTR0 0xC0010004
00436 #define MSR_K7_PERFCTR1 0xC0010005
00437 #define MSR_K7_PERFCTR2 0xC0010006
00438 #define MSR_K7_PERFCTR3 0xC0010007
00439
00440 #endif
00441
00442 #endif
00443
00444 /* end of P5/P6/K7 section */
00445 #endif
00446
00447 /* end of not only lib-prototypes section */
00448 #endif
00449
00450 EXTERN_C_END
00451
00452 #endif

```

## 15.13 amd64/l4/util/rdtsc.h File Reference

time stamp counter related functions

```

#include <l4/sys/compiler.h>
#include <l4/sys/l4int.h>

```



- 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 \*kip)  
*Calibrate scalers for time stamp calculations.*
- [l4\\_uint32\\_t l4\\_tsc\\_init](#) (int constraint, l4\_kernel\_info\_t \*kip)  
*Initialitze scaler for TSC calicaltions.*
- [l4\\_uint32\\_t l4\\_get\\_hz](#) (void)  
*Get CPU frequency in Hz.*

### 15.13.1 Detailed Description

time stamp counter related functions

#### Date

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [rdtsc.h](#).

## 15.14 rdtsc.h

```

00001
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 * This file is part of TUD:OS and distributed under the terms of the
00015 * GNU Lesser General Public License 2.1.
00016 * Please see the COPYING-LGPL-2.1 file for details.
00017 */
00018
00019 #ifndef __l4_rdtsc_h
00020 #define __l4_rdtsc_h
00021
00027 #include <l4/sys/compiler.h>
00028 #include <l4/sys/l4int.h>
00029 #include <l4/sys/kip.h>
00030
00031 EXTERN_C_BEGIN
00032
00033 /* interface */
00038
00039 #define L4_TSC_INIT_AUTO 0
00040 #define L4_TSC_INIT_KERNEL 1
00041 #define L4_TSC_INIT_CALIBRATE 2
00042
00043 extern l4_uint32_t l4_scaler_tsc_to_ns;
00044 extern l4_uint32_t l4_scaler_tsc_to_us;
00045 extern l4_uint32_t l4_scaler_ns_to_tsc;
00046 extern l4_uint32_t l4_scaler_tsc_linux;
00047
00052 L4_INLINE l4_cpu_time_t
00053 l4_rdtsc (void);
00054
00060 L4_INLINE
00061 l4_uint32_t l4_rdtsc_32 (void);
00062
00067 L4_INLINE l4_cpu_time_t
00068 l4_rdpmc (int nr);
00069
00075 L4_INLINE
00076 l4_uint32_t l4_rdpmc_32 (int nr);
00077

```

```

00082 L4_INLINE l4_uint64_t
00083 l4_tsc_to_ns (l4_cpu_time_t tsc);
00084
00089 L4_INLINE l4_uint64_t
00090 l4_tsc_to_us (l4_cpu_time_t tsc);
00091
00097 L4_INLINE void
00098 l4_tsc_to_s_and_ns (l4_cpu_time_t tsc,
00099 l4_uint32_t *s, l4_uint32_t *ns);
00099
00105 L4_INLINE l4_cpu_time_t
00106 l4_ns_to_tsc (l4_uint64_t ns);
00107
00113 L4_INLINE void
00114 l4_busy_wait_ns (l4_uint64_t ns);
00115
00121 L4_INLINE void
00122 l4_busy_wait_us (l4_uint64_t us);
00123
00124 EXTERN_C_BEGIN
00125
00134 L4_INLINE l4_uint32_t
00135 l4_calibrate_tsc (l4_kernel_info_t *kip);
00136
00162 L4_CV l4_uint32_t
00163 l4_tsc_init (int constraint, l4_kernel_info_t *kip);
00164
00169 L4_CV l4_uint32_t
00170 l4_get_hz (void);
00171
00174 EXTERN_C_END
00175
00176 /* implementaion */
00177
00178 L4_INLINE l4_uint32_t
00179 l4_calibrate_tsc (l4_kernel_info_t *kip)
00180 {
00181 return l4_tsc_init(L4_TSC_INIT_AUTO, kip);
00182 }
00183
00184 L4_INLINE l4_cpu_time_t
00185 l4_rdtsc (void)
00186 {
00187 l4_cpu_time_t v;
00188
00189 __asm__ __volatile__ (
00190 ("byte 0x0f, 0x31\n\t"
00191 "mov $0xffffffff, %%rcx\n\t"
00192 "and %%rcx, %%rax\n\t"
00193 "shlq $32, %%rdx\n\t"
00194 "orq %%rdx, %%rax\n\t"
00195 :
00196 "=a" (v)
00197 : /* no inputs */
00198 : "rdx", "rcx"
00199);
00200
00201 return v;
00202 }
00203
00204 L4_INLINE l4_cpu_time_t
00205 l4_rdpmc (int nr)
00206 {
00207 l4_cpu_time_t v;
00208 l4_uint64_t dummy;
00209
00210 __asm__ __volatile__ (
00211 "rdpmc\n\t"
00212 "mov $0xffffffff, %%rcx\n\t"
00213 "and %%rcx, %%rax\n\t"
00214 "shlq $32, %%rdx\n\t"
00215 "orq %%rdx, %%rax\n\t"
00216 :
00217 "=a" (v), "=c" (dummy)
00218 : "c" (nr)
00219 : "rdx"
00220);
00221
00222 return v;
00223 }
00224
00225 /* the same, but only 32 bit. Useful for smaller differences */
00226 L4_INLINE
00227 l4_uint32_t l4_rdpmc_32 (int nr)
00228 {
00229 l4_uint32_t x;
00230 l4_uint64_t dummy;

```



```

00231
00232 __asm__ __volatile__ (
00233 "rdpmc \n\t"
00234 "mov $0xffffffff, %%rcx \n\t" /* clears the upper 32 bits! */
00235 "and %%rcx, %%rax \n\t"
00236 : "=a" (x), "=c" (dummy)
00237 : "c" (nr)
00238 : "rdx");
00239
00240 return x;
00241 }
00242
00243 /* the same, but only 32 bit. Useful for smaller differences,
00244 needs less cycles. */
00245 L4_INLINE
00246 l4_uint32_t l4_rdtsc_32(void)
00247 {
00248 l4_uint32_t x;
00249
00250 __asm__ __volatile__ (
00251 ".byte 0x0f, 0x31\n\t" // rdtsc
00252 : "=a" (x)
00253 :
00254 : "rdx");
00255
00256 return x;
00257 }
00258
00259 L4_INLINE l4_uint64_t
00260 l4_tsc_to_ns (l4_cpu_time_t tsc)
00261 {
00262 l4_uint64_t ns, dummy;
00263
00264 __asm__
00265 (" \n\t"
00266 "mulq %3 \n\t"
00267 "shrd $27, %%rdx, %%rax \n\t"
00268 : "=a" (ns), "=d" (dummy)
00269 : "a" (tsc), "r" ((l4_uint64_t)l4_scaler_tsc_to_ns)
00270);
00271 return ns;
00272 }
00273 L4_INLINE l4_uint64_t
00274 l4_tsc_to_us (l4_cpu_time_t tsc)
00275 {
00276 l4_uint64_t ns, dummy;
00277
00278 __asm__
00279 (" \n\t"
00280 "mulq %3 \n\t"
00281 "shrd $32, %%rdx, %%rax \n\t"
00282 : "=a" (ns), "=d" (dummy)
00283 : "a" (tsc), "r" ((l4_uint64_t)l4_scaler_tsc_to_us)
00284);
00285 return ns;
00286 }
00287 L4_INLINE void
00288 l4_tsc_to_s_and_ns (l4_cpu_time_t tsc,
00289 l4_uint32_t *s, l4_uint32_t *ns)
00290 {
00291 __asm__
00292 (" \n\t"
00293 "mulq %3 \n\t"
00294 "shrd $27, %%rdx, %%rax \n\t"
00295 "xorq %%rdx, %%rdx \n\t"
00296 "divq %4 \n\t"
00297 : "=a" (*s), "=d" (*ns)
00298 : "a" (tsc), "r" ((l4_uint64_t)l4_scaler_tsc_to_ns),
00299 "rm" (1000000000ULL)
00300);
00301 }
00302 L4_INLINE l4_cpu_time_t
00303 l4_ns_to_tsc (l4_uint64_t ns)
00304 {
00305 l4_uint64_t tsc, dummy;
00306
00307 __asm__
00308 (" \n\t"
00309 "mulq %3 \n\t"
00310 "shrd $27, %%rdx, %%rax \n\t"
00311 : "=a" (tsc), "=d" (dummy)
00312 : "a" (ns), "r" ((l4_uint64_t)l4_scaler_ns_to_tsc)
00313);
00314 return tsc;
00315 }
00316 L4_INLINE void

```

```

00317 l4_busy_wait_ns (l4_uint64_t ns)
00318 {
00319 l4_cpu_time_t stop = l4_rdtsc();
00320 stop += l4_ns_to_tsc(ns);
00321
00322 while (l4_rdtsc() < stop)
00323 ;
00324 }
00325
00326 L4_INLINE void
00327 l4_busy_wait_us (l4_uint64_t us)
00328 {
00329 l4_cpu_time_t stop = l4_rdtsc ();
00330 stop += l4_ns_to_tsc(us*1000ULL);
00331
00332 while (l4_rdtsc() < stop)
00333 ;
00334 }
00335
00336 EXTERN_C_END
00337
00338 #endif /* __l4_rdtsc_h */
00339

```

## 15.15 x86/l4/util/rdtsc.h File Reference

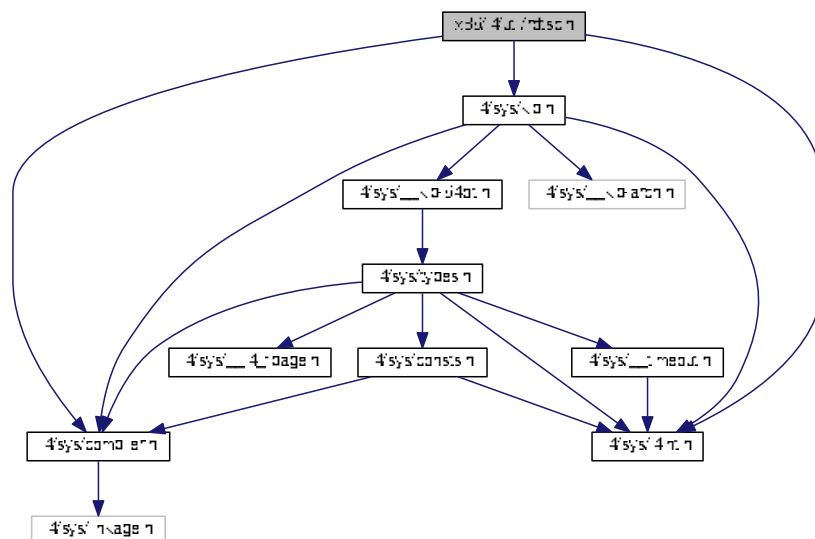
time stamp counter related functions

```

#include <l4/sys/compiler.h>
#include <l4/sys/l4int.h>
#include <l4/sys/kip.h>

```

Include dependency graph for rdtsc.h:



### Macros

- `#define L4_TSC_INIT_AUTO 0`  
*Automatic init.*
- `#define L4_TSC_INIT_KERNEL 1`  
*Initialized by kernel.*
- `#define L4_TSC_INIT_CALIBRATE 2`  
*Initialized by user-level.*

## Functions

- [l4\\_cpu\\_time\\_t](#) [l4\\_rdtsc](#) (void)  
*Read current value of CPU-internal time stamp counter.*
- [l4\\_uint32\\_t](#) [l4\\_rdtsc\\_32](#) (void)  
*Read the least significant 32 bit of the TSC.*
- [l4\\_cpu\\_time\\_t](#) [l4\\_rdpmc](#) (int nr)  
*Return current value of CPU-internal performance measurement counter.*
- [l4\\_uint32\\_t](#) [l4\\_rdpmc\\_32](#) (int nr)  
*Return the least significant 32 bit of a performance counter.*
- [l4\\_uint64\\_t](#) [l4\\_tsc\\_to\\_ns](#) ([l4\\_cpu\\_time\\_t](#) tsc)  
*Convert time stamp to ns value.*
- [l4\\_uint64\\_t](#) [l4\\_tsc\\_to\\_us](#) ([l4\\_cpu\\_time\\_t](#) tsc)  
*Convert time stamp 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 time stamp 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](#) \*kip)  
*Calibrate scalers for time stamp calculations.*
- [l4\\_uint32\\_t](#) [l4\\_tsc\\_init](#) (int constraint, [l4\\_kernel\\_info\\_t](#) \*kip)  
*Initialitze scaler for TSC calications.*
- [l4\\_uint32\\_t](#) [l4\\_get\\_hz](#) (void)  
*Get CPU frequency in Hz.*

### 15.15.1 Detailed Description

time stamp counter related functions

#### Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [rdtsc.h](#).

## 15.16 rdtsc.h

```

00001
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 * This file is part of TUD:OS and distributed under the terms of the
00017 * GNU Lesser General Public License 2.1.
00018 * Please see the COPYING-LGPL-2.1 file for details.
00019 */

```

```

00020
00021 #ifndef __l4_rdtsc_h
00022 #define __l4_rdtsc_h
00023
00029 #include <l4/sys/compiler.h>
00030 #include <l4/sys/l4int.h>
00031 #include <l4/sys/kip.h>
00032
00033 EXTERN_C_BEGIN
00034
00035 /* interface */
00040
00041 #define L4_TSC_INIT_AUTO 0
00042 #define L4_TSC_INIT_KERNEL 1
00043 #define L4_TSC_INIT_CALIBRATE 2
00044
00045 extern l4_uint32_t l4_scaler_tsc_to_ns;
00046 extern l4_uint32_t l4_scaler_tsc_to_us;
00047 extern l4_uint32_t l4_scaler_ns_to_tsc;
00048 extern l4_uint32_t l4_scaler_tsc_linux;
00049
00054 L4_INLINE l4_cpu_time_t
00055 l4_rdtsc (void);
00056
00062 L4_INLINE
00063 l4_uint32_t l4_rdtsc_32(void);
00064
00069 L4_INLINE l4_cpu_time_t
00070 l4_rdpmc (int nr);
00071
00077 L4_INLINE
00078 l4_uint32_t l4_rdpmc_32(int nr);
00079
00084 L4_INLINE l4_uint64_t
00085 l4_tsc_to_ns (l4_cpu_time_t tsc);
00086
00091 L4_INLINE l4_uint64_t
00092 l4_tsc_to_us (l4_cpu_time_t tsc);
00093
00099 L4_INLINE void
00100 l4_tsc_to_s_and_ns (l4_cpu_time_t tsc,
00101 l4_uint32_t *s, l4_uint32_t *ns);
00101
00107 L4_INLINE l4_cpu_time_t
00108 l4_ns_to_tsc (l4_uint64_t ns);
00109
00115 L4_INLINE void
00116 l4_busy_wait_ns (l4_uint64_t ns);
00117
00123 L4_INLINE void
00124 l4_busy_wait_us (l4_uint64_t us);
00125
00126 EXTERN_C_BEGIN
00127
00136 L4_INLINE l4_uint32_t
00137 l4_calibrate_tsc (l4_kernel_info_t *kip);
00138
00164 L4_CV l4_uint32_t
00165 l4_tsc_init (int constraint, l4_kernel_info_t *kip);
00166
00171 L4_CV l4_uint32_t
00172 l4_get_hz (void);
00173
00176 EXTERN_C_END
00177
00178 /* implementaion */
00179
00180 L4_INLINE l4_uint32_t
00181 l4_calibrate_tsc (l4_kernel_info_t *kip)
00182 {
00183 return l4_tsc_init(L4_TSC_INIT_AUTO, kip);
00184 }
00185
00186 L4_INLINE l4_cpu_time_t
00187 l4_rdtsc (void)
00188 {
00189 l4_cpu_time_t v;
00190
00191 __asm__ __volatile__
00192 (
00193 " .byte 0x0f, 0x31 \n\t"
00194 /*"rdtsc\n\t"*/
00195 :
00196 "=A" (v)
00197 : /* no inputs */
00198);
00199

```

```

00200 return v;
00201 }
00202
00203 /* the same, but only 32 bit. Useful for smaller differences,
00204 needs less cycles. */
00205 L4_INLINE
00206 l4_uint32_t l4_rdtsc_32(void)
00207 {
00208 l4_uint32_t x;
00209
00210 __asm__ __volatile__ (
00211 ".byte 0x0f, 0x31\n\t" // rdtsc
00212 : "=a" (x)
00213 :
00214 : "edx");
00215
00216 return x;
00217 }
00218
00219 L4_INLINE l4_cpu_time_t
00220 l4_rdpmc (int nr)
00221 {
00222 l4_cpu_time_t v;
00223
00224 __asm__ __volatile__ (
00225 "rdpmc\n\t"
00226 :
00227 : "=A" (v)
00228 : "c" (nr)
00229);
00230
00231 return v;
00232 }
00233
00234 /* the same, but only 32 bit. Useful for smaller differences,
00235 needs less cycles. */
00236 L4_INLINE
00237 l4_uint32_t l4_rdpmc_32(int nr)
00238 {
00239 l4_uint32_t x;
00240
00241 __asm__ __volatile__ (
00242 "rdpmc\n\t"
00243 : "=a" (x)
00244 : "c" (nr)
00245 : "edx");
00246
00247 return x;
00248 }
00249
00250 L4_INLINE l4_uint64_t
00251 l4_tsc_to_ns (l4_cpu_time_t tsc)
00252 {
00253 l4_uint32_t dummy;
00254 l4_uint64_t ns;
00255 __asm__
00256 (
00257 "\n\t"
00258 "movl %%edx, %%ecx\n\t"
00259 "mull %3\n\t"
00260 "movl %%ecx, %%eax\n\t"
00261 "movl %%edx, %%ecx\n\t"
00262 "mull %3\n\t"
00263 "addl %%ecx, %%eax\n\t"
00264 "adcl $0, %%edx\n\t"
00265 "shld $5, %%eax, %%edx\n\t"
00266 "shll $5, %%eax\n\t"
00267 : "=A" (ns),
00268 "=c" (dummy)
00269 : "0" (tsc),
00270 "g" (l4_scaler_tsc_to_ns)
00271);
00272 return ns;
00273 }
00274
00275 L4_INLINE l4_uint64_t
00276 l4_tsc_to_us (l4_cpu_time_t tsc)
00277 {
00278 l4_uint32_t dummy;
00279 l4_uint64_t us;
00280 __asm__
00281 (
00282 "\n\t"
00283 "movl %%edx, %%ecx\n\t"
00284 "mull %3\n\t"
00285 "movl %%ecx, %%eax\n\t"
00286 "movl %%edx, %%ecx\n\t"
00287 "mull %3\n\t"
00288 "addl %%ecx, %%eax\n\t"

```

```

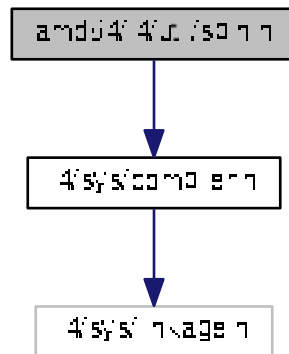
00287 "adcl $0, %%edx \n\t"
00288 : "=A" (us),
00289 "=&c" (dummy)
00290 : "0" (tsc),
00291 "g" (l4_scaler_tsc_to_us)
00292);
00293 return us;
00294 }
00295
00296 L4_INLINE void
00297 l4_tsc_to_s_and_ns (l4_cpu_time_t tsc,
00298 l4_uint32_t *s, l4_uint32_t *ns)
00299 {
00300 l4_uint32_t dummy;
00301 __asm__
00302 (
00303 "movl %%edx, %%ecx \n\t"
00304 "mull %4 \n\t"
00305 "movl %%ecx, %%eax \n\t"
00306 "movl %%edx, %%ecx \n\t"
00307 "mull %4 \n\t"
00308 "addl %%ecx, %%eax \n\t"
00309 "adcl $0, %%edx \n\t"
00310 "movl $1000000000, %%ecx \n\t"
00311 "shld $5, %%eax, %%edx \n\t"
00312 "shll $5, %%eax \n\t"
00313 "divl %%ecx \n\t"
00314 : "=a" (*s), "=d" (*ns), "=&c" (dummy)
00315 : "A" (tsc), "g" (l4_scaler_tsc_to_ns)
00316);
00317 }
00318 L4_INLINE l4_cpu_time_t
00319 l4_ns_to_tsc (l4_uint64_t ns)
00320 {
00321 l4_uint32_t dummy;
00322 l4_cpu_time_t tsc;
00323 __asm__
00324 (
00325 "movl %%edx, %%ecx \n\t"
00326 "mull %3 \n\t"
00327 "movl %%ecx, %%eax \n\t"
00328 "movl %%edx, %%ecx \n\t"
00329 "mull %3 \n\t"
00330 "addl %%ecx, %%eax \n\t"
00331 "adcl $0, %%edx \n\t"
00332 "shld $5, %%eax, %%edx \n\t"
00333 "shll $5, %%eax \n\t"
00334 : "=A" (tsc),
00335 "=&c" (dummy)
00336 : "0" (ns),
00337 "g" (l4_scaler_ns_to_tsc)
00338);
00339 return tsc;
00340 }
00341
00342 L4_INLINE void
00343 l4_busy_wait_ns (l4_uint64_t ns)
00344 {
00345 l4_cpu_time_t stop = l4_rdtsc();
00346 stop += l4_ns_to_tsc(ns);
00347
00348 while (l4_rdtsc() < stop)
00349 ;
00350 }
00351
00352 L4_INLINE void
00353 l4_busy_wait_us (l4_uint64_t us)
00354 {
00355 l4_cpu_time_t stop = l4_rdtsc ();
00356 stop += l4_ns_to_tsc(us*1000ULL);
00357
00358 while (l4_rdtsc() < stop)
00359 ;
00360 }
00361
00362 EXTERN_C_END
00363
00364 #endif /* __l4_rdtsc_h */
00365

```

## 15.17 amd64/l4/util/spin.h File Reference

Spinning for amd64.

```
#include <l4/sys/compiler.h>
Include dependency graph for spin.h:
```



### 15.17.1 Detailed Description

Spinning for amd64.

Definition in file [spin.h](#).

## 15.18 spin.h

```

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 * 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 #ifndef __l4util_spin_h
00014 #define __l4util_spin_h
00015
00016 #include <l4/sys/compiler.h>
00017
00018 EXTERN_C_BEGIN
00019
00020 L4_CV void l4_spin(int x,int y);
00021 L4_CV void l4_spin_vga(int x,int y);
00022 L4_CV void l4_spin_n_text(int x, int y, int len, const char*s);
00023 L4_CV void l4_spin_n_text_vga(int x, int y, int len, const char*s);
00024
00025 /*****
00026 *
00027 * spin_text() - spinning wheel at the hercules screen. The given text
00028 * must be a text constant, no variables or arrays. Its
00029 * size is determined with the sizeof operator, it's much
00030 * faster than the strlen function.
00031 *
00032 *****/

```

```

00031 * spin_text_vga() - same for vga.
00032 *
00033 *****/
00034 #define l4_spin_text(x, y, text) \
00035 l4_spin_n_text((x), (y), sizeof(text)-1, "" text)
00036 #define l4_spin_text_vga(x, y, text) \
00037 l4_spin_n_text_vga((x), (y), sizeof(text)-1, "" text)
00038
00039 EXTERN_C_END
00040
00041 #endif

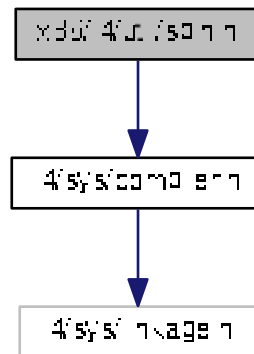
```

## 15.19 x86/l4/util/spin.h File Reference

Spinning for x86.

```
#include <l4/sys/compiler.h>
```

Include dependency graph for spin.h:



### 15.19.1 Detailed Description

Spinning for x86.

Definition in file [spin.h](#).

## 15.20 spin.h

```

00001
00002 /*
00003 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004 * Frank Mehnert <fm3@os.inf.tu-dresden.de>
00005 * economic rights: Technische Universität Dresden (Germany)
00006 * This file is part of TUD:OS and distributed under the terms of the
00007 * GNU Lesser General Public License 2.1.
00008 * Please see the COPYING-LGPL-2.1 file for details.
00009 */
00010 #ifndef __l4util_spin_h
00011 #define __l4util_spin_h
00012

```



```

00016 #include <l4/sys/compiler.h>
00017
00018 EXTERN_C_BEGIN
00019
00020 L4_CV void l4_spin(int x,int y);
00021 L4_CV void l4_spin_vga(int x,int y);
00022 L4_CV void l4_spin_n_text(int x, int y, int len, const char*s);
00023 L4_CV void l4_spin_n_text_vga(int x, int y, int len, const char*s);
00024
00025 /*****
00026 *
00027 * spin_text() - spinning wheel at the hercules screen. The given text
00028 * must be a text constant, no variables or arrays. Its
00029 * size is determined with the sizeof operator, it's much
00030 * faster than the strlen function.
00031 * spin_text_vga() - same for vga.
00032 *
00033 *****/
00034 #define l4_spin_text(x, y, text) \
00035 l4_spin_n_text((x), (y), sizeof(text)-1, "" text)
00036 #define l4_spin_text_vga(x, y, text) \
00037 l4_spin_n_text_vga((x), (y), sizeof(text)-1, "" text)
00038
00039 EXTERN_C_END
00040
00041 #endif

```

## 15.21 amd64/l4/util/util.h File Reference

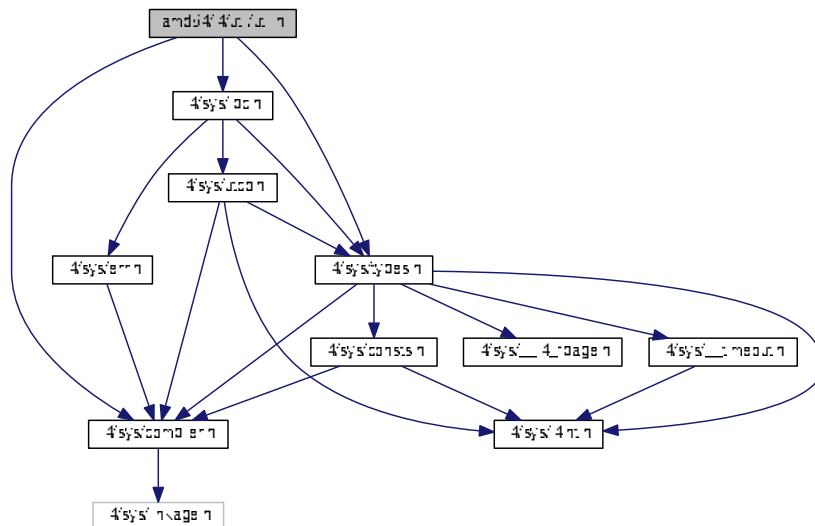
Utilities, amd64 version.

```

#include <l4/sys/types.h>
#include <l4/sys/compiler.h>
#include <l4/sys/ipc.h>

```

Include dependency graph for util.h:



## Functions

- [l4\\_timeout\\_s l4util\\_micros2l4to](#) (unsigned int mus) [L4\\_NOTHROW](#)  
Calculate l4 timeouts.

- void [l4\\_sleep](#) (int ms) [L4\\_NOTHROW](#)  
*Suspend thread for a period of ms milliseconds.*
- void [l4\\_sleep\\_forever](#) (void) [L4\\_NOTHROW](#)  
*Go sleep and never wake up.*

### 15.21.1 Detailed Description

Utilities, amd64 version.

Definition in file [util.h](#).

### 15.21.2 Function Documentation

#### 15.21.2.1 l4\_sleep()

```
void l4_sleep (
 int ms)
```

Suspend thread for a period of *ms* milliseconds.

#### Parameters

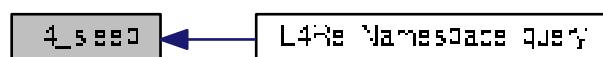
|           |                      |
|-----------|----------------------|
| <i>ms</i> | Time in milliseconds |
|-----------|----------------------|

#### 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](#).

Referenced by [L4Re::Namespace::query\(\)](#).

Here is the caller graph for this function:



## 15.21.2.2 l4util\_micros2l4to()

```
l4_timeout_s l4util_micros2l4to (
 unsigned int mus)
```

Calculate l4 timeouts.

## Parameters

|            |                                                                                                                                               |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| <i>mus</i> | time in microseconds. Special cases: <ul style="list-style-type: none"> <li>• 0 -&gt; timeout 0</li> <li>• ~0U -&gt; timeout NEVER</li> </ul> |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------|

## Returns

the corresponding l4\_timeout value

## 15.22 util.h

```
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 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU Lesser General Public License 2.1.
00012 * Please see the COPYING-LGPL-2.1 file for details.
00013 */
00014 #ifndef __UTIL_H
00015 #define __UTIL_H
00016
00017 #include <l4/sys/types.h>
00018 #include <l4/sys/compiler.h>
00019 #include <l4/sys/ipc.h>
00020
00021 EXTERN_C_BEGIN
00022
00029 L4_CV l4_timeout_s l4util_micros2l4to(unsigned int mus)
 L4_NOTHROW;
00030
00034 L4_CV void l4_sleep(int ms) L4_NOTHROW;
00035
00036 /* Suspend thread for a period of \a us microseconds.
00037 * \param us Time in microseconds
00038 * WARNING: This function is mostly bogus since the timer resolution of
00039 * current L4 implementations is about 1ms! */
00040 L4_CV void l4_usleep(int us) L4_NOTHROW;
00041
00047 L4_INLINE void l4_sleep_forever(void) L4_NOTHROW __attribute__((noreturn));
00048
00049 L4_INLINE void
00050 l4_sleep_forever(void) L4_NOTHROW
00051 {
00052 for (;;)
00053 l4_ipc_sleep(L4_IPC_NEVER);
00054 }
00055
00057 static inline void
00058 l4_touch_ro(const void*addr, unsigned size) L4_NOTHROW
00059 {
00060 const char *bptr, *eptr;
00061
00062 bptr = (const char*)((l4_addr_t)addr) & L4_PAGEMASK;
00063 eptr = (const char*)((l4_addr_t)addr+size-1) & L4_PAGEMASK;
00064 for(;bptr<=eptr;bptr+=L4_PAGE_SIZE){
00065 asm volatile("or %0,%rax \n"
```



- void [l4\\_sleep](#) (int *ms*) [L4\\_NOTHROW](#)  
*Suspend thread for a period of *ms* milliseconds.*
- void [l4\\_sleep\\_forever](#) (void) [L4\\_NOTHROW](#)  
*Go sleep and never wake up.*

### 15.23.1 Detailed Description

Utilities for x86.

Definition in file [util.h](#).

### 15.23.2 Function Documentation

#### 15.23.2.1 l4\_sleep()

```
void l4_sleep (
 int ms)
```

Suspend thread for a period of *ms* milliseconds.

##### Parameters

|           |                      |
|-----------|----------------------|
| <i>ms</i> | Time in milliseconds |
|-----------|----------------------|

#### 15.23.2.2 l4util\_micros2l4to()

```
l4_timeout_s l4util_micros2l4to (
 unsigned int mus)
```

Calculate I4 timeouts.

##### Parameters

|            |                                                                                                                                            |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| <i>mus</i> | time in microseconds. Special cases: <ul style="list-style-type: none"><li>• 0 -&gt; timeout 0</li><li>• ~0U -&gt; timeout NEVER</li></ul> |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------|

##### Returns

the corresponding [l4\\_timeout](#) value

## 15.24 util.h

```

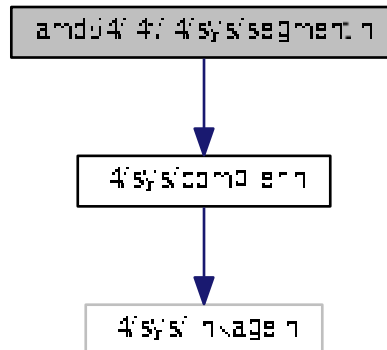
00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Alexander Warg <warg@os.inf.tu-dresden.de>,
00008 * Frank Mehnert <fm3@os.inf.tu-dresden.de>,
00009 * Jork Löser <jork@os.inf.tu-dresden.de>
00010 * economic rights: Technische Universität Dresden (Germany)
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU Lesser General Public License 2.1.
00013 * Please see the COPYING-LGPL-2.1 file for details.
00014 */
00015 #ifndef __UTIL_H
00016 #define __UTIL_H
00017
00018 #include <l4/sys/types.h>
00019 #include <l4/sys/compiler.h>
00020 #include <l4/sys/ipc.h>
00021
00022 EXTERN_C_BEGIN
00023
00030 L4_CV l4_timeout_s l4util_micros2l4to(unsigned int mus)
00031 L4_NOTHROW;
00032
00033 L4_CV void l4_sleep(int ms) L4_NOTHROW;
00034
00035 /* Suspend thread for a period of \a us microseconds.
00036 * \param us Time in microseconds
00037 * WARNING: This function is mostly bogus since the timer resolution of
00038 * current L4 implementations is about 1ms! */
00039 L4_CV void l4_usleep(int us) L4_NOTHROW;
00040
00041 L4_INLINE void l4_sleep_forever(void) L4_NOTHROW __attribute__((noreturn));
00042
00043 L4_INLINE void
00044 l4_sleep_forever(void) L4_NOTHROW
00045 {
00046 for (;;)
00047 l4_ipc_sleep(L4_IPC_NEVER);
00048 }
00049
00050 static inline void
00051 l4_touch_ro(const void*addr, unsigned size) L4_NOTHROW
00052 {
00053 const char *bptr, *eptr;
00054
00055 bptr = (const char*)((l4_addr_t)addr) & L4_PAGEMASK;
00056 eptr = (const char*)((l4_addr_t)addr+size-1) & L4_PAGEMASK;
00057 for(;bptr<=eptr;bptr+=L4_PAGESIZE){
00058 asm volatile("or %0,%0\n"
00059 :
00060 : "m" (*(const unsigned*)bptr)
00061 : "eax");
00062 }
00063 }
00064
00065 static inline void
00066 l4_touch_rw(const void*addr, unsigned size) L4_NOTHROW
00067 {
00068 const char *bptr, *eptr;
00069
00070 bptr = (const char*)((l4_addr_t)addr) & L4_PAGEMASK;
00071 eptr = (const char*)((l4_addr_t)addr+size-1) & L4_PAGEMASK;
00072 for(;bptr<=eptr;bptr+=L4_PAGESIZE){
00073 asm volatile("orb $0,%0\n"
00074 :
00075 : "m" (*(const unsigned*)bptr)
00076 :);
00077 }
00078 }
00079
00080 EXTERN_C_END
00081
00082 #endif
00083

```

## 15.25 amd64/l4f/l4/sys/segment.h File Reference

l4f specific fs/gs manipulation

#include <l4/sys/compiler.h>  
 Include dependency graph for segment.h:



## Functions

- long [fiasco\\_amd64\\_set\\_fs](#) ([l4\\_cap\\_idx\\_t](#) thread, [l4\\_umword\\_t](#) base, [l4\\_utcb\\_t](#) \*utcb)  
*Set the FS register.*
- 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 FS register.*
- 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.*

### 15.25.1 Detailed Description

l4f specific fs/gs manipulation

Definition in file [segment.h](#).

### 15.25.2 Function Documentation

#### 15.25.2.1 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 FS register.

**Parameters**

|               |                          |
|---------------|--------------------------|
| <i>thread</i> | Thread to get info from. |
| <i>base</i>   | Base address.            |
| <i>utcb</i>   | UTCB of the caller.      |

**Returns**

System call error

Definition at line 35 of file [segment.h](#).

**15.25.2.2 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 FS register.

**Parameters**

|               |                                                  |
|---------------|--------------------------------------------------|
| <i>thread</i> | Thread to get info from.                         |
| <i>segr</i>   | Segment register to set (one of L4_sys_segment). |
| <i>base</i>   | Base address.                                    |
| <i>utcb</i>   | UTCB of the caller.                              |

**Returns**

System call error

Definition at line 43 of file [segment.h](#).

**15.26 segment.h**

```
00001 #include_next <l4/sys/segment.h>
00002
00008 /*
00009 * (c) 2011 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00010 * economic rights: Technische Universität Dresden (Germany)
00011 *
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
```



```

00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025 #ifndef __L4_SYS__ARCH_AMD64__L4API_L4F__SEGMENT_H__
00026 #define __L4_SYS__ARCH_AMD64__L4API_L4F__SEGMENT_H__
00027
00028 #include <l4/sys/compiler.h>
00029
00030 /***** Implementation *****/
00031
00032
00033
00034 L4_INLINE long
00035 fiasco_amd64_set_fs(l4_cap_idx_t thread,
00036 l4_umword_t base, l4_utcb_t *utcb)
00037 {
00038 l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_AMD64_SET_SEGMENT_BASE_OP | ((
00039 l4_umword_t)L4_AMD64_SEGMENT_FS << 16);
00040 l4_utcb_mr_u(utcb)->mr[1] = base;
00041 return l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(
00042 L4_PROTO_THREAD, 2, 0, 0), L4_IPC_NEVER), utcb);
00043 }
00044
00045 L4_INLINE long
00046 fiasco_amd64_set_segment_base(l4_cap_idx_t thread, enum
00047 L4_sys_segment sgr,
00048 l4_umword_t base, l4_utcb_t *utcb)
00049 {
00050 l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_AMD64_SET_SEGMENT_BASE_OP | ((
00051 l4_umword_t)sgr << 16);
00052 l4_utcb_mr_u(utcb)->mr[1] = base;
00053 return l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(
00054 L4_PROTO_THREAD, 2, 0, 0), L4_IPC_NEVER), utcb);
00055 }
00056
00057 L4_INLINE long
00058 fiasco_gdt_set(l4_cap_idx_t thread, void *desc, unsigned int size,
00059 unsigned int entry_number_start, l4_utcb_t *utcb)
00060 {
00061 l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_X86_GDT_OP;
00062 l4_utcb_mr_u(utcb)->mr[1] = entry_number_start;
00063 __builtin_memcpy(&l4_utcb_mr_u(utcb)->mr[2], desc, size);
00064 return l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(
00065 L4_PROTO_THREAD, 2 + (size / 8), 0, 0), L4_IPC_NEVER), utcb);
00066 }
00067
00068 #endif /* ! __L4_SYS__ARCH_X86__L4API_L4F__SEGMENT_H__ */

```

## 15.27 amd64/l4/sys/segment.h File Reference

Segment handling.

```

#include <l4/sys/ipc.h>
#include <l4/sys/task.h>
#include <l4/sys/thread.h>

```



## 15.27.1 Detailed Description

Segment handling.

Definition in file [segment.h](#).

## 15.27.2 Enumeration Type Documentation

### 15.27.2.1 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. |
| 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 76 of file [segment.h](#).

## 15.27.3 Function Documentation

### 15.27.3.1 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. |
| in  | <i>utcb</i>      | UTCB of the caller.         |

**Returns**

System call error

Definition at line 158 of file [segment.h](#).

**15.27.3.2 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 FS register.

**Parameters**

|               |                          |
|---------------|--------------------------|
| <i>thread</i> | Thread to get info from. |
| <i>base</i>   | Base address.            |
| <i>utcb</i>   | UTCB of the caller.      |

**Returns**

System call error

Definition at line 35 of file [segment.h](#).

**15.27.3.3 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 FS register.

**Parameters**

|               |                                                  |
|---------------|--------------------------------------------------|
| <i>thread</i> | Thread to get info from.                         |
| <i>segr</i>   | Segment register to set (one of L4_sys_segment). |
| <i>base</i>   | Base address.                                    |
| <i>utcb</i>   | UTCB of the caller.                              |

## Returns

System call error

Definition at line 43 of file [segment.h](#).

## 15.28 segment.h

```

00001
00006 /*
00007 * (c) 2008-2009 Adam Lackorzynski <adam@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 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
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_SYS_ARCH_X86__SEGMENT_H__
00025 #define __L4_SYS_ARCH_X86__SEGMENT_H__
00026
00027 #ifndef L4API_l4f
00028 #error This header file can only be used with a L4API version!
00029 #endif
00030
00031 #include <l4/sys/ipc.h>
00032
00043 L4_INLINE long
00044 fiasco_ldt_set(l4_cap_idx_t task, void *ldt, unsigned int num_desc,
00045 unsigned int entry_number_start, l4_utcb_t *utcb);
00046
00060 L4_INLINE long
00061 fiasco_gdt_set(l4_cap_idx_t thread, void *desc, unsigned int size,
00062 unsigned int entry_number_start, l4_utcb_t *utcb);
00063
00070 L4_INLINE unsigned
00071 fiasco_gdt_get_entry_offset(l4_cap_idx_t thread,
00072 l4_utcb_t *utcb);
00072
00076 enum L4_task_ldt_x86_consts
00077 {
00079 L4_TASK_LDT_X86_ENTRY_SIZE = 8,
00081 L4_TASK_LDT_X86_MAX_ENTRIES
00082 = (L4_UTCB_GENERIC_DATA_SIZE - 2)
00083 / (L4_TASK_LDT_X86_ENTRY_SIZE / (L4_MWORD_BITS / 8)),
00084 };
00085
00093 L4_INLINE long
00094 fiasco_amd64_set_fs(l4_cap_idx_t thread,
00095 l4_umword_t base, l4_utcb_t *utcb);
00095
00096 enum L4_sys_segment
00097 {
00098 L4_AMD64_SEGMENT_FS = 0,
00099 L4_AMD64_SEGMENT_GS = 1
00100 };
00101
00110 L4_INLINE long
00111 fiasco_amd64_set_segment_base(l4_cap_idx_t thread, enum
00112 L4_sys_segment segr,
00113 l4_umword_t base, l4_utcb_t *utcb);
00113
00123 L4_INLINE long
00124 fiasco_amd64_segment_info(l4_cap_idx_t thread, unsigned *user_ds,
00125 unsigned *user_cs, unsigned *user32_cs,
00126 l4_utcb_t *utcb);
00126
00127
00128 /*****
00129 *** Implementation
00130 *****/
00131

```

```

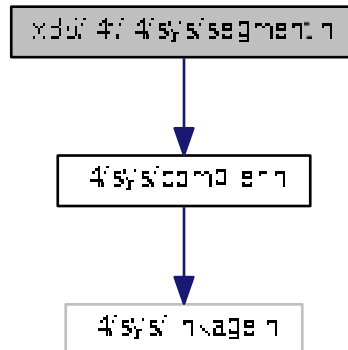
00132 #include <l4/sys/task.h>
00133 #include <l4/sys/thread.h>
00134
00135 L4_INLINE long
00136 fiasco_ldt_set(l4_cap_idx_t task, void *ldt, unsigned int num_desc,
00137 unsigned int entry_number_start, l4_utcb_t *utcb)
00138 {
00139 if (num_desc > L4_TASK_LDT_X86_MAX_ENTRIES)
00140 return -L4_EINVAL;
00141 l4_utcb_mr_u(utcb)->mr[0] = L4_TASK_LDT_SET_X86_OP;
00142 l4_utcb_mr_u(utcb)->mr[1] = entry_number_start;
00143 __builtin_memcpy(&l4_utcb_mr_u(utcb)->mr[2], ldt,
00144 num_desc * L4_TASK_LDT_X86_ENTRY_SIZE);
00145 return l4_error_u(l4_ipc_call(task, utcb, l4_msgtag(
00146 L4_PROTO_TASK, 2 + num_desc * 2, 0, 0), L4_IPC_NEVER), utcb);
00147 }
00148 L4_INLINE unsigned
00149 fiasco_gdt_get_entry_offset(l4_cap_idx_t thread,
00150 l4_utcb_t *utcb)
00151 {
00152 l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_X86_GDT_OP;
00153 if (l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(
00154 L4_PROTO_THREAD, 1, 0, 0), L4_IPC_NEVER), utcb))
00155 return -1;
00156 return l4_utcb_mr_u(utcb)->mr[0];
00157 }
00158 L4_INLINE long
00159 fiasco_amd64_segment_info(l4_cap_idx_t thread, unsigned *user_ds,
00160 unsigned *user_cs, unsigned *user32_cs,
00161 l4_utcb_t *utcb)
00162 {
00163 l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00164 int r;
00165 m->mr[0] = L4_THREAD_AMD64_GET_SEGMENT_INFO_OP;
00166 r = l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(
00167 L4_PROTO_THREAD, 1, 0, 0), L4_IPC_NEVER), utcb);
00168 if (r < 0)
00169 return r;
00170 *user_ds = m->mr[0];
00171 *user_cs = m->mr[1];
00172 *user32_cs = m->mr[2];
00173 return 0;
00174 }
00175 #endif /* ! __L4_SYS_ARCH_X86_SEGMENT_H__ */

```

## 15.29 x86/i4f/l4/sys/segment.h File Reference

i4f specific segment manipulation

```
#include <l4/sys/compiler.h>
Include dependency graph for segment.h:
```



## Functions

- 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.

### 15.29.1 Detailed Description

l4f specific segment manipulation

Definition in file [segment.h](#).

## 15.30 segment.h

```
00001 #include_next <l4/sys/segment.h>
00002
00008 /*
00009 * (c) 2008–2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00010 * economic rights: Technische Universität Dresden (Germany)
00011 *
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025 #ifndef __L4_SYS__ARCH_X86__L4API_L4F__SEGMENT_H__
00026 #define __L4_SYS__ARCH_X86__L4API_L4F__SEGMENT_H__
00027
00028 #include <l4/sys/compiler.h>
```

```

00029
00030 /*****
00031 *** Implementation
00032 *****/
00033
00034 L4_INLINE long
00035 fiasco_gdt_set(l4_cap_idx_t thread, void *desc, unsigned int size,
00036 unsigned int entry_number_start, l4_utcb_t *utcb)
00037 {
00038 l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_X86_GDT_OP;
00039 l4_utcb_mr_u(utcb)->mr[1] = entry_number_start;
00040 __builtin_memcpy(&l4_utcb_mr_u(utcb)->mr[2], desc, size);
00041 return l4_error_u(l4_ipc_call(thread, utcb, l4_msgtag(
00042 L4_PROTO_THREAD, 2 + (size >> 2), 0, 0), L4_IPC_NEVER), utcb);
00043 }
00044 #endif /* ! __L4_SYS__ARCH_X86__L4API_L4F__SEGMENT_H__ */

```

## 15.31 x86/l4/sys/segment.h File Reference

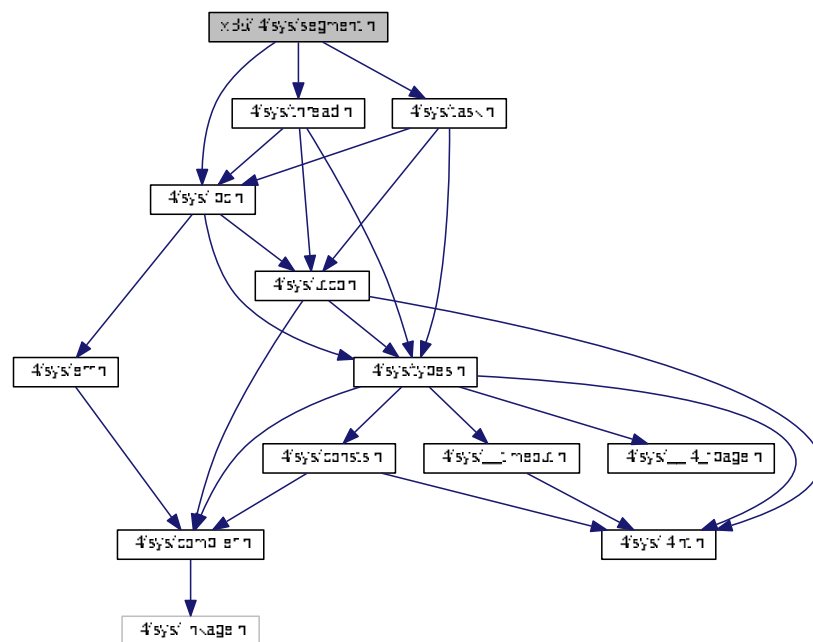
Segment handling.

```

#include <l4/sys/ipc.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`, `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.

### 15.31.1 Detailed Description

Segment handling.

Definition in file [segment.h](#).

### 15.31.2 Enumeration Type Documentation

#### 15.31.2.1 L4\_task\_ldt\_x86\_consts

```
enum L4_task_ldt_x86_consts
```

Contents 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. |
| <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 76 of file [segment.h](#).

## 15.32 segment.h

```
00001
00006 /*
00007 * (c) 2008-2009 Adam Lackorzynski <adam@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 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
```

```

00018 * file does not by itself cause the resulting executable to be covered by
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_SYS__ARCH_X86__SEGMENT_H__
00025 #define __L4_SYS__ARCH_X86__SEGMENT_H__
00026
00027 #ifndef L4API_L4f
00028 #error This header file can only be used with a L4API version!
00029 #endif
00030
00031 #include <l4/sys/ipc.h>
00032
00043 L4_INLINE long
00044 fiasco_ldt_set(l4_cap_idx_t task, void *ldt, unsigned int size,
00045 unsigned int entry_number_start, l4_utcb_t *utcb);
00046
00060 L4_INLINE long
00061 fiasco_gdt_set(l4_cap_idx_t thread, void *desc, unsigned int size,
00062 unsigned int entry_number_start, l4_utcb_t *utcb);
00063
00070 L4_INLINE unsigned
00071 fiasco_gdt_get_entry_offset(l4_cap_idx_t thread,
00072 l4_utcb_t *utcb);
00072
00076 enum L4_task_ldt_x86_consts
00077 {
00079 L4_TASK_LDT_X86_ENTRY_SIZE = 8,
00081 L4_TASK_LDT_X86_MAX_ENTRIES
00082 = (L4_UTCB_GENERIC_DATA_SIZE - 2)
00083 / (L4_TASK_LDT_X86_ENTRY_SIZE / (L4_MWORD_BITS / 8)),
00084 };
00085
00086 /*****
00087 *** Implementation
00088 *****/
00089
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(
00104 L4_PROTO_TASK, 2 + num_desc * 2, 0, 0), L4_IPC_NEVER), utcb);
00104 }
00105
00106 L4_INLINE unsigned
00107 fiasco_gdt_get_entry_offset(l4_cap_idx_t thread,
00108 l4_utcb_t *utcb)
00109 {
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(
00111 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 #endif /* ! __L4_SYS__ARCH_X86__SEGMENT_H__ */

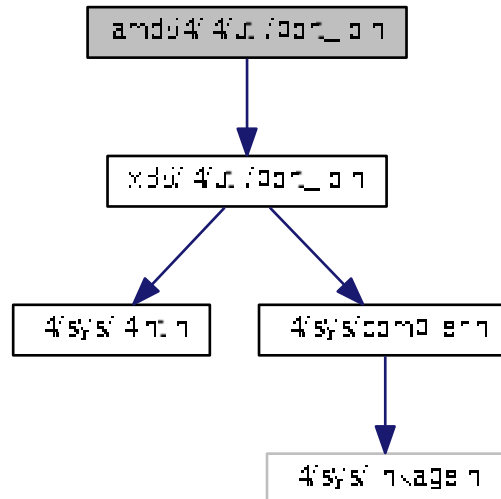
```

## 15.33 amd64/l4/l4/util/port\_io.h File Reference

Port I/O functions.



```
#include <x86/l4/util/port_io.h>
Include dependency graph for port_io.h:
```



### 15.35.1 Detailed Description

Port I/O functions.

Definition in file [port\\_io.h](#).

## 15.36 port\_io.h

```

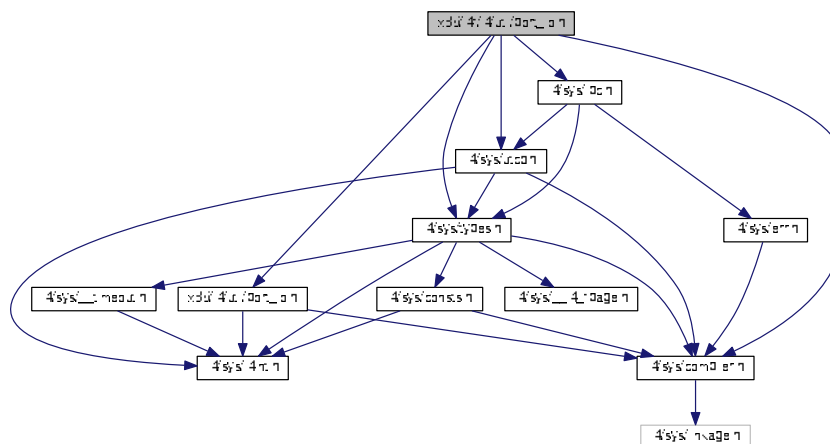
00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007 * economic rights: Technische Universität Dresden (Germany)
00008 * This file is part of TUD:OS and distributed under the terms of the
00009 * GNU Lesser General Public License 2.1.
00010 * Please see the COPYING-LGPL-2.1 file for details.
00011 */
00012 #include <x86/l4/util/port_io.h>
```

## 15.37 x86/l4/l4/util/port\_io.h File Reference

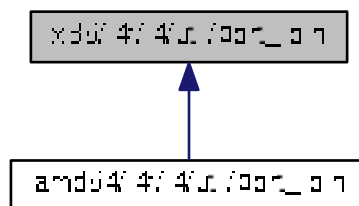
port I/O functions

```
#include <l4/sys/compiler.h>
#include <l4/sys/types.h>
#include <x86/l4/util/port_io.h>
#include <l4/sys/utcb.h>
```

```
#include <linux/sys/ipc.h>
```



This graph shows which files directly or indirectly include this file:



## Functions

- `int l4util_ioport_map (l4_cap_idx_t sigma0id, unsigned port_start, unsigned log2size)`  
*Map a range of I/O ports.*

### 15.37.1 Detailed Description

## port I/O functions

Date \_\_\_\_\_

06/2003

### Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [port\\_io.h](#).

## 15.37.2 Function Documentation

### 15.37.2.1 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. |

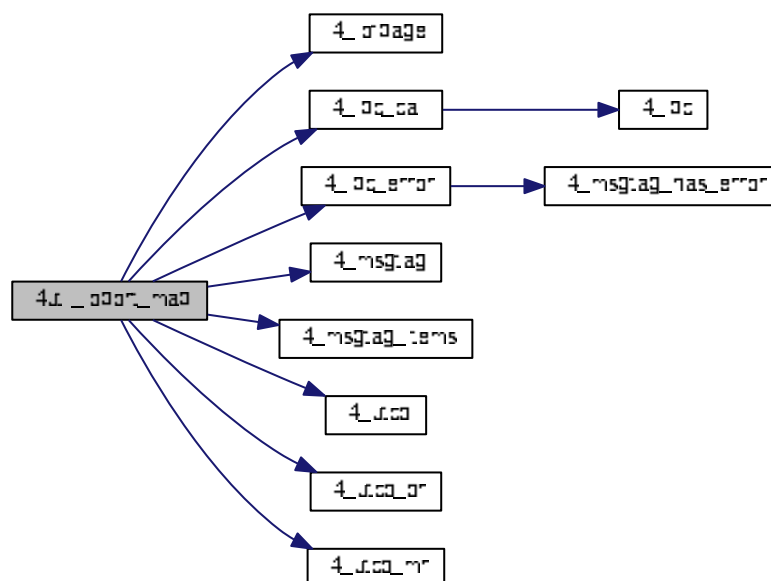
#### Returns

IPC result: 0 if the range could be successfully mapped on error: IPC failure, or -L4\_ENOENT if nothing mapped

Definition at line 56 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:



## 15.38 port\_io.h

```

00001 /*****/
00009 /*****/
00010
00011 /*
00012 * (c) 2003-2009 Author(s)
00013 * economic rights: Technische Universität Dresden (Germany)
00014 * This file is part of TUD:OS and distributed under the terms of the
00015 * GNU Lesser General Public License 2.1.
00016 * Please see the COPYING-LGPL-2.1 file for details.
00017 */
00018
00019 #ifndef _L4UTIL_PORT_IO_API_H
00020 #define _L4UTIL_PORT_IO_API_H
00021
00022 #include <l4/sys/compiler.h>
00023 #include <l4/sys/types.h>
00024
00025 #include <x86/l4/util/port_io.h>
00026
00027 EXTERN_C_BEGIN
00028
00029 L4_INLINE int
00041 l4util_ioport_map(l4_cap_idx_t sigma0id,
00042 unsigned port_start, unsigned log2size);
00043
00044 EXTERN_C_END
00045
00046
00047 /*****/
00048 *** Implementation
00049 *****/
00050
00051 #include <l4/sys/utcb.h>
00052 #include <l4/sys/ipc.h>
00053
00054
00055 L4_INLINE int
00056 l4util_ioport_map(l4_cap_idx_t sigma0id,
00057 unsigned port_start, unsigned log2size)
00058 {
00059 l4_fpage_t iofp;
00060 l4_msgtag_t tag;
00061 long err;
00062
00063 iofp = l4_iofpage(port_start, log2size);
00064 l4_utcb_mr()->mr[0] = iofp.raw;
00065 l4_utcb_br()->bdr = 0;
00066 l4_utcb_br()->br[0] = L4_ITEM_MAP;
00067 l4_utcb_br()->br[1] = iofp.raw;
00068 tag = l4_ipc_call(sigma0id, l4_utcb(),
00069 l4_msgtag(L4_PROTO_IO_PAGE_FAULT, 1, 0, 0),
00070 L4_IPC_NEVER);
00071
00072 if ((err = l4_ipc_error(tag, l4_utcb())))
00073 return err;
00074
00075 return l4_msgtag_items(tag) > 0 ? 0 : -L4_ENOENT;
00076 }
00077
00078 #endif
00079

```

## 15.39 x86/l4/util/port\_io.h File Reference

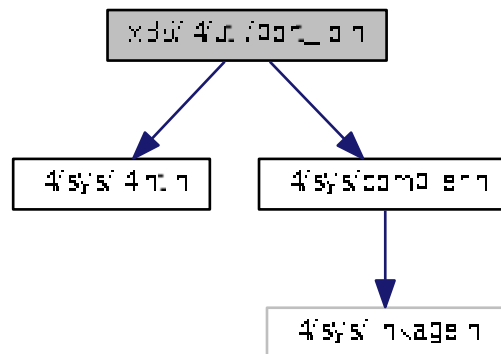
x86 port I/O

```

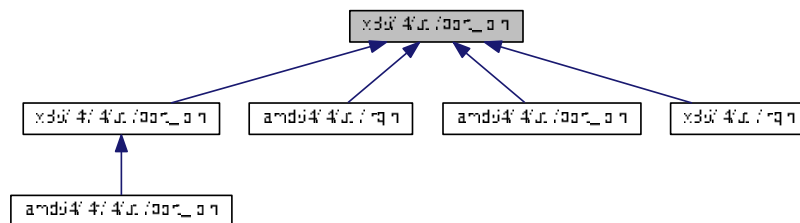
#include <l4/sys/l4int.h>
#include <l4/sys/compiler.h>

```

Include dependency graph for port\_io.h:



This graph shows which files directly or indirectly include this file:



## 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*

### 15.39.1 Detailed Description

x86 port I/O

Date

06/2003

Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [port\\_io.h](#).

## 15.40 port\_io.h

```
00001 /*****
00009 /*****
00010
00011 */
00012 * (c) 2003-2009 Author(s)
00013 * economic rights; Technische Universität Dresden (Germany)
00014 * This file is part of TUD:OS and distributed under the terms of the
00015 * GNU Lesser General Public License 2.1.
00016 * Please see the COPYING-LGPL-2.1 file for details.
00017 */
00018
00019 #ifndef _L4UTIL_PORT_IO_H
00020 #define _L4UTIL_PORT_IO_H
00021
00022 /* L4 includes */
00023 #include <l4/sys/l4int.h>
00024 #include <l4/sys/compiler.h>
00025
00026 /*****
00027 *** Prototypes
00028 *****/
00029
00030 EXTERN_C_BEGIN
00031 L4_INLINE l4_uint8_t
00032 l4util_in8(l4_uint16_t port);
00033
00034 L4_INLINE l4_uint16_t
00035 l4util_in16(l4_uint16_t port);
00036
00037 L4_INLINE l4_uint32_t
00038 l4util_in32(l4_uint16_t port);
00039
00040 L4_INLINE void
00041 l4util_ins8(l4_uint16_t port, l4_umword_t addr,
```

```

 l4_umword_t count);
00076
00084 L4_INLINE void
00085 l4util_ins16(l4_uint16_t port, l4_umword_t addr,
 l4_umword_t count);
00086
00094 L4_INLINE void
00095 l4util_ins32(l4_uint16_t port, l4_umword_t addr,
 l4_umword_t count);
00096
00103 L4_INLINE void
00104 l4util_out8(l4_uint8_t value, l4_uint16_t port);
00105
00113 L4_INLINE void
00114 l4util_out16(l4_uint16_t value, l4_uint16_t port);
00115
00122 L4_INLINE void
00123 l4util_out32(l4_uint32_t value, l4_uint16_t port);
00124
00132 L4_INLINE void
00133 l4util_outs8(l4_uint16_t port, l4_umword_t addr,
 l4_umword_t count);
00134
00143 L4_INLINE void
00144 l4util_outs16(l4_uint16_t port, l4_umword_t addr,
 l4_umword_t count);
00145
00153 L4_INLINE void
00154 l4util_outs32(l4_uint16_t port, l4_umword_t addr,
 l4_umword_t count);
00155
00159 L4_INLINE void
00160 l4util_iodelay(void);
00161
00164 EXTERN_C_END
00165
00166
00167 /*****
00168 *** Implementation
00169 *****/
00170
00171 L4_INLINE l4_uint8_t
00172 l4util_in8(l4_uint16_t port)
00173 {
00174 l4_uint8_t value;
00175 asm volatile ("inb %w1, %b0" : "=a" (value) : "Nd" (port));
00176 return value;
00177 }
00178
00179 L4_INLINE l4_uint16_t
00180 l4util_in16(l4_uint16_t port)
00181 {
00182 l4_uint16_t value;
00183 asm volatile ("inw %w1, %w0" : "=a" (value) : "Nd" (port));
00184 return value;
00185 }
00186
00187 L4_INLINE l4_uint32_t
00188 l4util_in32(l4_uint16_t port)
00189 {
00190 l4_uint32_t value;
00191 asm volatile ("inl %w1, %0" : "=a" (value) : "Nd" (port));
00192 return value;
00193 }
00194
00195 L4_INLINE void
00196 l4util_ins8(l4_uint16_t port, l4_umword_t addr,
 l4_umword_t count)
00197 {
00198 l4_umword_t dummy1, dummy2;
00199 asm volatile ("rep insb" : "=D" (dummy1), "=c" (dummy2)
00200 : "d" (port), "D" (addr), "c" (count)
00201 : "memory");
00202 }
00203
00204 L4_INLINE void
00205 l4util_ins16(l4_uint16_t port, l4_umword_t addr,
 l4_umword_t count)
00206 {
00207 l4_umword_t dummy1, dummy2;
00208 asm volatile ("rep insw" : "=D" (dummy1), "=c" (dummy2)
00209 : "d" (port), "D" (addr), "c" (count)
00210 : "memory");
00211 }
00212
00213 L4_INLINE void
00214 l4util_ins32(l4_uint16_t port, l4_umword_t addr,

```

```

 l4_umword_t count)
00215 {
00216 l4_umword_t dummy1, dummy2;
00217 asm volatile ("rep insl" : "=D"(dummy1), "=c"(dummy2)
00218 : "d" (port), "D" (addr), "c"(count)
00219 : "memory");
00220 }
00221
00222 L4_INLINE void
00223 l4util_out8(l4_uint8_t value, l4_uint16_t port)
00224 {
00225 asm volatile ("outb %b0, %w1" : : "a" (value), "Nd" (port));
00226 }
00227
00228 L4_INLINE void
00229 l4util_out16(l4_uint16_t value, l4_uint16_t port)
00230 {
00231 asm volatile ("outw %w0, %w1" : : "a" (value), "Nd" (port));
00232 }
00233
00234 L4_INLINE void
00235 l4util_out32(l4_uint32_t value, l4_uint16_t port)
00236 {
00237 asm volatile ("outl %0, %w1" : : "a" (value), "Nd" (port));
00238 }
00239
00240 L4_INLINE void
00241 l4util_outs8(l4_uint16_t port, l4_umword_t addr,
00242 l4_umword_t count)
00243 {
00244 l4_umword_t dummy1, dummy2;
00245 asm volatile ("rep outsb" : "=S"(dummy1), "=c"(dummy2)
00246 : "d" (port), "S" (addr), "c"(count)
00247 : "memory");
00248 }
00249
00250 L4_INLINE void
00251 l4util_outs16(l4_uint16_t port, l4_umword_t addr,
00252 l4_umword_t count)
00253 {
00254 l4_umword_t dummy1, dummy2;
00255 asm volatile ("rep outsw" : "=S"(dummy1), "=c"(dummy2)
00256 : "d" (port), "S" (addr), "c"(count)
00257 : "memory");
00258 }
00259
00260 L4_INLINE void
00261 l4util_outs32(l4_uint16_t port, l4_umword_t addr,
00262 l4_umword_t count)
00263 {
00264 l4_umword_t dummy1, dummy2;
00265 asm volatile ("rep outsl" : "=S"(dummy1), "=c"(dummy2)
00266 : "d" (port), "S" (addr), "c"(count)
00267 : "memory");
00268 }
00269
00270 L4_INLINE void
00271 l4util_iodelay(void)
00272 {
00273 asm volatile ("outb %a1, $0x80");
00274 }
00275
00276 #endif

```

## 15.41 amd64/l4f/l4/util/setjmp.h File Reference

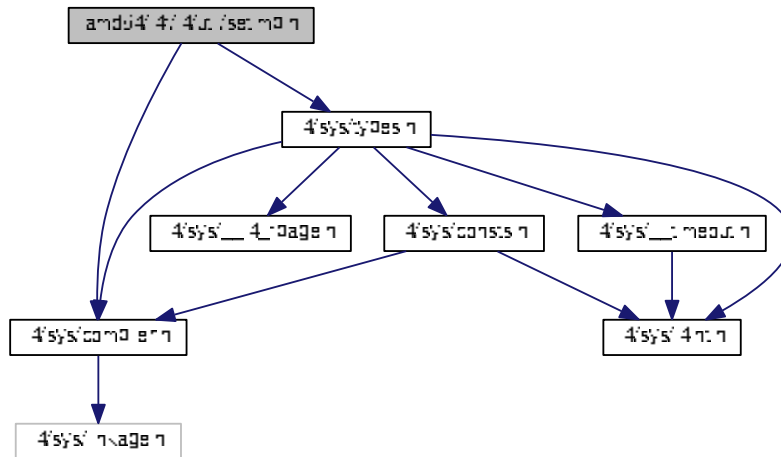
Inter-thread setjmp/longjmp.

```

#include <l4/sys/types.h>
#include <l4/sys/compiler.h>

```

Include dependency graph for `setjmp.h`:



## Functions

- `int l4_thread_setjmp (l4_thread_jmp_buf env)`  
*inter-thread setjmp*
- `void l4_thread_longjmp (l4_threadid_t thread, l4_thread_jmp_buf env, int val)`  
*inter-thread longjmp*

### 15.41.1 Detailed Description

Inter-thread `setjmp/longjmp`.

Date

12/21/2005

Author

Jork Loeser [jork.loeser@inf.tu-dresden.de](mailto:jork.loeser@inf.tu-dresden.de)

Definition in file [setjmp.h](#).

### 15.41.2 Function Documentation

#### 15.41.2.1 l4\_thread\_longjmp()

```
void l4_thread_longjmp (
 l4_threadid_t thread,
 l4_thread_jmp_buf env,
 int val)
```

inter-thread `longjmp`

This function sets `thread` to the location obtained by its former `l4_thread_setjump` on `env`.

## Parameters

|               |                                |
|---------------|--------------------------------|
| <i>thread</i> | thread to apply the longjmp to |
| <i>env</i>    | jump buffer                    |
| <i>val</i>    | 0: setjmp returns with 1       |
| <i>val</i>    | !0: return value of setjmp     |

## See also

longjmp(3)

## Note

In contrast to longjmp(3), this function returns.

## 15.41.2.2 l4\_thread\_setjmp()

```
int l4_thread_setjmp (
 l4_thread_jump_buf env)
```

## inter-thread setjmp

Use this function to prepare a longjmp from another thread for this thread.

## Parameters

|            |             |
|------------|-------------|
| <i>env</i> | jump buffer |
|------------|-------------|

## Return values

|    |                       |
|----|-----------------------|
| 0  | returned directly     |
| !0 | returned from longjmp |

## See also

setjmp(3)

## 15.42 setjmp.h

```
00001
00009 /*
00010 * (c) 2004-2009 Author(s)
00011 * economic rights: Technische Universität Dresden (Germany)
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU Lesser General Public License 2.1.
00014 * Please see the COPYING-LGPL-2.1 file for details.
00015 */
00016 #ifndef __UTIL_INCLUDE_ARCH_AMD64_L4API_L4F_SETJMP_H_
```

```

00017 #define __UTIL_INCLUDE_ARCH_Amd64_L4API_L4F_SETJMP_H_
00018 #include <l4/sys/types.h>
00019 #include <l4/sys/compiler.h>
00020
00021 EXTERN_C_BEGIN
00022
00023 typedef struct{
00024 l4_umword_t r8; /* 0x00 */
00025 l4_umword_t r9; /* 0x08 */
00026 l4_umword_t r10; /* 0x10 */
00027 l4_umword_t r11; /* 0x18 */
00028 l4_umword_t r12; /* 0x20 */
00029 l4_umword_t r13; /* 0x28 */
00030 l4_umword_t r14; /* 0x30 */
00031 l4_umword_t r15; /* 0x38 */
00032 l4_umword_t rbx; /* 0x40 */
00033 l4_umword_t rsi; /* 0x48 */
00034 l4_umword_t rbp; /* 0x50 */
00035 l4_umword_t rsp; /* 0x58 */
00036 l4_umword_t rip; /* 0x60 */
00037 l4_umword_t rip_caller; /* 0x68 */
00038 l4_umword_t rflags; /* 0x70 */
00039 l4_umword_t stack[40];
00040 } l4_thread_jump_buf_s;
00041 typedef int l4_thread_jump_buf[sizeof(l4_thread_jump_buf_s)/sizeof(l4_umword_t)];
00042
00043 typedef union{
00044 l4_thread_jump_buf_s s;
00045 l4_thread_jump_buf raw;
00046 } l4_thread_jump_buf_u;
00047
00059 L4_CV int l4_thread_setjmp(l4_thread_jump_buf env);
00060
00075 L4_CV void l4_thread_longjmp(l4_threadid_t thread, l4_thread_jump_buf env, int val);
00076
00077 EXTERN_C_END
00078
00079 #endif

```

## 15.43 x86/i4f/i4/util/setjmp.h File Reference

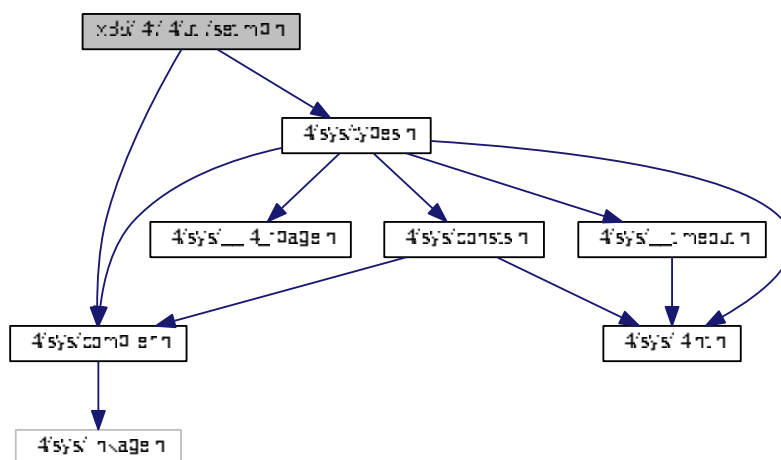
Inter-thread setjmp/longjmp.

```

#include <l4/sys/types.h>
#include <l4/sys/compiler.h>

```

Include dependency graph for setjmp.h:



## Functions

- int [l4\\_thread\\_setjmp](#) (l4\_thread\_jmp\_buf env)  
*inter-thread setjmp*
- void [l4\\_thread\\_longjmp](#) (l4\_threadid\_t thread, l4\_thread\_jmp\_buf env, int val)  
*inter-thread longjmp*

### 15.43.1 Detailed Description

Inter-thread setjmp/longjmp.

#### Date

11/26/2004

#### Author

Jork Loeser [jork.loeser@inf.tu-dresden.de](mailto:jork.loeser@inf.tu-dresden.de)

Definition in file [setjmp.h](#).

### 15.43.2 Function Documentation

#### 15.43.2.1 l4\_thread\_longjmp()

```
void l4_thread_longjmp (
 l4_threadid_t thread,
 l4_thread_jmp_buf env,
 int val)
```

inter-thread longjmp

This function sets `thread` to the location obtained by its former `l4_thread_setjump` on `env`.

#### Parameters

|               |                                |
|---------------|--------------------------------|
| <i>thread</i> | thread to apply the longjmp to |
| <i>env</i>    | jump buffer                    |
| <i>val</i>    | 0: setjmp returns with 1       |
| <i>val</i>    | !0: return value of setjmp     |

#### See also

[longjmp\(3\)](#)

**Note**

In contrast to `longjmp(3)`, this function returns.

**15.43.2.2 l4\_thread\_setjmp()**

```
int l4_thread_setjmp (
 l4_thread_jump_buf env)
```

**inter-thread setjmp**

Use this function to prepare a `longjmp` from another thread for this thread.

**Parameters**

|            |             |
|------------|-------------|
| <i>env</i> | jump buffer |
|------------|-------------|

**Return values**

|    |                                    |
|----|------------------------------------|
| 0  | returned directly                  |
| !0 | returned from <code>longjmp</code> |

**See also**

`setjmp(3)`

**15.44 setjmp.h**

```
00001
00009 /*
00010 * (c) 2004-2009 Author(s)
00011 * economic rights: Technische Universität Dresden (Germany)
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU Lesser General Public License 2.1.
00014 * Please see the COPYING-LGPL-2.1 file for details.
00015 */
00016 #ifndef __UTIL_INCLUDE_ARCH_X86_L4API_L4F_SETJMP_H_
00017 #define __UTIL_INCLUDE_ARCH_X86_L4API_L4F_SETJMP_H_
00018 #include <l4/sys/types.h>
00019 #include <l4/sys/compiler.h>
00020
00021 EXTERN_C_BEGIN
00022
00023 typedef struct{
00024 l4_umword_t ebx; /* 0 */
00025 l4_umword_t esi; /* 4 */
00026 l4_umword_t edi; /* 8 */
00027 l4_umword_t ebp; /* 12 */
00028 l4_umword_t esp; /* 16 */
00029 l4_umword_t eip; /* 20 */
00030 l4_umword_t eip_caller; /* 24 */
00031 l4_umword_t eflags; /* 28 */
00032 l4_umword_t stack[40];
00033 } l4_thread_jump_buf_s;
00034 typedef int l4_thread_jump_buf[sizeof(l4_thread_jump_buf_s)/sizeof(l4_umword_t)];
00035
00036 typedef union{
00037 l4_thread_jump_buf_s s;
00038 l4_thread_jump_buf raw;
```



```

00039 } l4_thread_jump_buf_u;
00040
00052 extern int l4_thread_setjmp(l4_thread_jump_buf env);
00053
00068 void l4_thread_longjmp(l4_threadid_t thread, l4_thread_jump_buf env, int val);
00069
00070 EXTERN_C_END
00071
00072 #endif

```

## 15.45 arm/l4/sys/linkage.h File Reference

Linkage.

### Macros

- `#define L4_CV`  
Define calling convention.

### 15.45.1 Detailed Description

Linkage.

Definition in file [linkage.h](#).

## 15.46 linkage.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 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024 #ifndef __L4_SYS_ARCH_ARM_LINKAGE_H__
00025 #define __L4_SYS_ARCH_ARM_LINKAGE_H__
00026
00027 #ifdef __ASSEMBLY__
00028 #ifndef ENTRY
00029 #define ENTRY(name) \
00030 .globl name; \
00031 .p2align(2); \
00032 name:
00033 #endif
00034 #endif
00035
00036 #define L4_FASTCALL(x) x
00037 #define l4_fastcall
00038
00044 #define L4_CV
00045
00046 #ifdef __PIC__
00047 # define L4_LONG_CALL
00048 #else
00049 # define L4_LONG_CALL __attribute__((long_call))
00050 #endif
00051
00052 #endif /* ! __L4_SYS_ARCH_ARM_LINKAGE_H__ */

```

## 15.47 amd64/l4/sys/linkage.h File Reference

Linkage.

### Macros

- `#define L4_CV`  
*Define calling convention.*

### 15.47.1 Detailed Description

Linkage.

Definition in file [linkage.h](#).

## 15.48 linkage.h

```

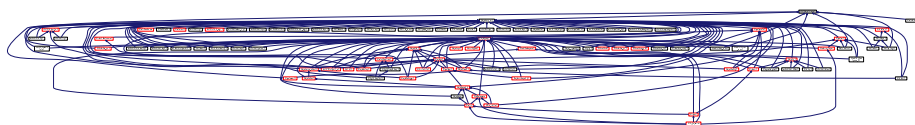
00001
00006 /*
00007 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008 * Alexander Warg <warg@os.inf.tu-dresden.de>,
00009 * Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00010 * economic rights: Technische Universität Dresden (Germany)
00011 *
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025 #ifndef __L4__SYS__ARCH_AMD64__LINKAGE_H__
00026 #define __L4__SYS__ARCH_AMD64__LINKAGE_H__
00027
00028 #ifdef __ASSEMBLY__
00029
00030 #ifndef ENTRY
00031 #define ENTRY(name) \
00032 .globl name; \
00033 .p2align(2); \
00034 name:
00035
00036 #endif /* __ASSEMBLY__ */
00037 #endif /* ! ENTRY */
00038
00039 #define L4_FASTCALL(x) x
00040 #define l4_fastcall
00041
00047 #define L4_CV
00048
00049 #endif /* ! __L4__SYS__ARCH_AMD64__LINKAGE_H__ */

```

## 15.49 x86/l4/sys/linkage.h File Reference

Linkage.

This graph shows which files directly or indirectly include this file:



## Macros

- `#define L4_CV __attribute__((regparm(0)))`  
*Define calling convention.*

### 15.49.1 Detailed Description

Linkage.

Definition in file [linkage.h](#).

## 15.50 linkage.h

```

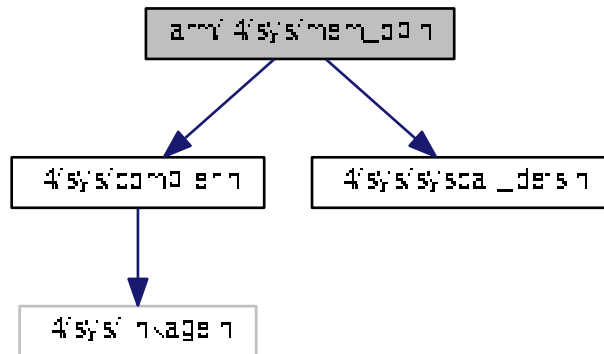
00001
00006 /*
00007 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008 * Alexander Warg <warg@os.inf.tu-dresden.de>,
00009 * Frank Mehnert <fm3@os.inf.tu-dresden.de>
00010 * economic rights: Technische Universität Dresden (Germany)
00011 *
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025 #ifndef __L4__SYS__ARCH_X86__LINKAGE_H__
00026 #define __L4__SYS__ARCH_X86__LINKAGE_H__
00027
00028 #ifdef __ASSEMBLY__
00029
00030 #ifndef ENTRY
00031 #define ENTRY(name) \
00032 .globl name; \
00033 .p2align(2); \
00034 name:
00035
00036 #endif /* ! ENTRY */
00037 #endif /* __ASSEMBLY__ */
00038
00039 #define L4_FASTCALL(x) x __attribute__((regparm(3)))
00040 #define l4_fastcall __attribute__((regparm(3)))
00041
00047 #define L4_CV __attribute__((regparm(0)))
00048
00049 #endif /* ! __L4__SYS__ARCH_X86__LINKAGE_H__ */

```

## 15.51 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 memory from kernel privilege level.*
- void [l4\\_mem\\_write](#) (unsigned long virtaddress, unsigned width, unsigned long value)  
*Write 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.*

### 15.51.1 Detailed Description

Memory access functions (ARM specific)

Date

2010-10

Author

Adam Lackorzynski [adam@os.inf.tu-dresden.de](mailto:adam@os.inf.tu-dresden.de)

Definition in file [mem\\_op.h](#).

## 15.52 mem\_op.h

```

00001
00009 /*
00010 * (c) 2010 Author(s)
00011 * economic rights: Technische Universität Dresden (Germany)
00012 *
00013 * This file is part of TUD:OS and distributed under the terms of the
00014 * GNU General Public License 2.
00015 * Please see the COPYING-GPL-2 file for details.
00016 *
00017 * As a special exception, you may use this file as part of a free software
00018 * library without restriction. Specifically, if other files instantiate
00019 * templates or use macros or inline functions from this file, or you compile
00020 * this file and link it with other files to produce an executable, this
00021 * file does not by itself cause the resulting executable to be covered by
00022 * the GNU General Public License. This exception does not however
00023 * invalidate any other reasons why the executable file might be covered by
00024 * the GNU General Public License.
00025 */
00026 #ifndef __L4SYS__INCLUDE__ARCH_ARM__MEM_OP_H__
00027 #define __L4SYS__INCLUDE__ARCH_ARM__MEM_OP_H__
00028
00029 #include <l4/sys/compiler.h>
00030 #include <l4/sys/syscall_defs.h>
00031
00032 EXTERN_C_BEGIN
00033
00051 enum L4_mem_op_widths
00052 {
00053 L4_MEM_WIDTH_1BYTE = 0,
00054 L4_MEM_WIDTH_2BYTE = 1,
00055 L4_MEM_WIDTH_4BYTE = 2,
00056 };
00057
00070 L4_INLINE unsigned long
00071 l4_mem_read(unsigned long virtaddress, unsigned width);
00072
00085 L4_INLINE void
00086 l4_mem_write(unsigned long virtaddress, unsigned width,
00087 unsigned long value);
00088
00089 enum L4_mem_ops
00090 {
00091 L4_MEM_OP_MEM_READ = 0x10,
00092 L4_MEM_OP_MEM_WRITE = 0x11,
00093 };
00094
00098 L4_INLINE unsigned long
00099 l4_mem_arm_op_call(unsigned long op,
00100 unsigned long va,
00101 unsigned long width,
00102 unsigned long value);
00103
00106 L4_INLINE unsigned long
00107 l4_mem_arm_op_call(unsigned long op,
00108 unsigned long va,
00109 unsigned long width,
00110 unsigned long value)
00111 {
00112 register unsigned long _op __asm__ ("r0") = op;
00113 register unsigned long _va __asm__ ("r1") = va;
00114 register unsigned long _width __asm__ ("r2") = width;
00115 register unsigned long _value __asm__ ("r3") = value;
00116
00117 __asm__ __volatile__
00118 ("@ l4_cache_op_arm_call(start) \n\t"
00119 "mov lr, pc \n\t"
00120 "mov pc, %[sc] \n\t"
00121 "@ l4_cache_op_arm_call(end) \n\t"
00122 :
00123 "=r" (_op),
00124 "=r" (_va),
00125 "=r" (_width),
00126 "=r" (_value)
00127 :
00128 [sc] "i" (L4_SYSCALL_MEM_OP),
00129 "0" (_op),
00130 "1" (_va),
00131 "2" (_width),
00132 "3" (_value)
00133 :
00134 "cc", "memory", "lr"
00135);
00136
00137 return _value;

```

```

00138 }
00139
00140 L4_INLINE unsigned long
00141 l4_mem_read(unsigned long virtaddress, unsigned width)
00142 {
00143 return l4_mem_arm_op_call(L4_MEM_OP_MEM_READ, virtaddress, width, 0);
00144 }
00145
00146 L4_INLINE void
00147 l4_mem_write(unsigned long virtaddress, unsigned width,
00148 unsigned long value)
00149 {
00150 l4_mem_arm_op_call(L4_MEM_OP_MEM_WRITE, virtaddress, width, value);
00151 }
00152
00153 EXTERN_C_END
00154
00155 #endif /* ! __L4SYS__INCLUDE__ARCH_ARM__MEM_OP_H__ */

```

## 15.53 arm/l4/sys/vm.h File Reference

ARM virtualization interface.

### Data Structures

- struct [l4\\_vm\\_tz\\_state](#)  
*state structure for TrustZone VMs*

### 15.53.1 Detailed Description

ARM virtualization interface.

Definition in file [vm.h](#).

## 15.54 vm.h

```

00001
00002 /*
00003 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004 * Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00005 *
00006 * economic rights: Technische Universität Dresden (Germany)
00007 *
00008 * This file is part of TUD:OS and distributed under the terms of the
00009 * GNU General Public License 2.
00010 * Please see the COPYING-GPL-2 file for details.
00011 *
00012 * As a special exception, you may use this file as part of a free software
00013 * library without restriction. Specifically, if other files instantiate
00014 * templates or use macros or inline functions from this file, or you compile
00015 * this file and link it with other files to produce an executable, this
00016 * file does not by itself cause the resulting executable to be covered by
00017 * the GNU General Public License. This exception does not however
00018 * invalidate any other reasons why the executable file might be covered by
00019 * the GNU General Public License.
00020 */
00021 #pragma once
00022
00023 struct l4_vm_tz_state_mode
00024 {
00025 l4_umword_t sp;
00026 l4_umword_t lr;
00027 l4_umword_t spsr;
00028 };
00029
00030 struct l4_vm_tz_state_irq_inject

```

```

00043 {
00044 l4_uint32_t group;
00045 l4_uint32_t irqs[8];
00046 };
00047
00052 struct l4_vm_tz_state
00053 {
00054 l4_umword_t r[13]; // r0 - r12
00055
00056 l4_umword_t sp_usr;
00057 l4_umword_t lr_usr;
00058
00059 struct l4_vm_tz_state_mode irq;
00060
00061 l4_umword_t r_fiq[5]; // r8 - r12
00062 struct l4_vm_tz_state_mode fiq;
00063 struct l4_vm_tz_state_mode abt;
00064 struct l4_vm_tz_state_mode und;
00065 struct l4_vm_tz_state_mode svc;
00066
00067 l4_umword_t pc;
00068 l4_umword_t cpsr;
00069
00070 l4_umword_t pending_events;
00071 l4_uint32_t cpacr;
00072 l4_umword_t cpl0_fpexc;
00073
00074 l4_umword_t pfs;
00075 l4_umword_t pfa;
00076 l4_umword_t exit_reason;
00077
00078 struct l4_vm_tz_state_irq_inject irq_inject;
00079 };
00080
00081 enum L4_vm_exit_reason
00082 {
00083 L4_vm_exit_reason_vmm_call = 1,
00084 L4_vm_exit_reason_inst_abort = 2,
00085 L4_vm_exit_reason_data_abort = 3,
00086 L4_vm_exit_reason_irq = 4,
00087 L4_vm_exit_reason_fiq = 5,
00088 L4_vm_exit_reason_undef = 6,
00089 };
00090
00091 L4_INLINE int
00092 l4_vm_tz_irq_inject(struct l4_vm_tz_state *state, unsigned irq);
00093
00094 L4_INLINE int
00095 l4_vm_tz_irq_inject(struct l4_vm_tz_state *state, unsigned irq)
00096 {
00097 if (irq > sizeof(state->irq_inject.irqs) * 8)
00098 return -L4_EINVAL;
00099
00100 unsigned g = irq / 32;
00101 state->irq_inject.group |= 1 << g;
00102 state->irq_inject.irqs[g] |= 1 << (irq & 31);
00103
00104 return 0;
00105 }

```

## 15.55 arm/l4/util/bitops\_arch.h File Reference

ARM specific implementation of bitops functions.

### 15.55.1 Detailed Description

ARM specific implementation of bitops functions.

Definition in file [bitops\\_arch.h](#).

## 15.56 bitops\_arch.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007 * economic rights: Technische Universität Dresden (Germany)
00008 * This file is part of TUD:OS and distributed under the terms of the
00009 * GNU Lesser General Public License 2.1.
00010 * Please see the COPYING-LGPL-2.1 file for details.
00011 */
00012 #ifndef __L4UTIL__ARCH_ARM__BITOPS_ARCH_H__
00013 #define __L4UTIL__ARCH_ARM__BITOPS_ARCH_H__
00014
00015
00016 #endif /* ! __L4UTIL__ARCH_ARM__BITOPS_ARCH_H__ */

```

## 15.57 amd64/l4/util/bitops\_arch.h File Reference

amd64 bit manipulation functions

### 15.57.1 Detailed Description

amd64 bit manipulation functions

Date

07/03/2001

Author

Lars Reuther [reuther@os.inf.tu-dresden.de](mailto:reuther@os.inf.tu-dresden.de) Torsten Frenzel [frenzel@os.inf.tu-dresden.de](mailto:frenzel@os.inf.tu-dresden.de)  
 de Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [bitops\\_arch.h](#).

## 15.58 bitops\_arch.h

```

00001 /*****
00011 */
00012 * (c) 2000-2009 Author(s)
00013 * economic rights: Technische Universität Dresden (Germany)
00014 * This file is part of TUD:OS and distributed under the terms of the
00015 * GNU Lesser General Public License 2.1.
00016 * Please see the COPYING-LGPL-2.1 file for details.
00017 */
00018
00019 /*****
00020 #ifndef __L4UTIL__INCLUDE__ARCH_AMD64__BITOPS_ARCH_H__
00021 #define __L4UTIL__INCLUDE__ARCH_AMD64__BITOPS_ARCH_H__
00022
00023 EXTERN_C_BEGIN
00024
00025 /*****
00026 *** Implementation
00027 *****/
00028
00029 #define __L4UTIL__BITOPS_HAVE_ARCH_SET_BIT
00030 L4_INLINE void
00031 l4util_set_bit(int b, volatile l4_umword_t * dest)
00032 {
00033 __asm__ __volatile__

```



```

00034 (
00035 "lock; bts %1,%0 \n\t"
00036 :
00037 :
00038 "m" (*dest), /* 0 mem, destination operand */
00039 "Ir" (b) /* 1, bit number */
00040 :
00041 "memory", "cc"
00042);
00043 }
00044
00045 /* clear bit */
00046 #define __L4UTIL_BITOPS_HAVE_ARCH_CLEAR_BIT
00047 L4_INLINE void
00048 l4util_clear_bit(int b, volatile l4_umword_t * dest)
00049 {
00050 __asm__ __volatile__
00051 (
00052 "lock; btr %1,%0 \n\t"
00053 :
00054 :
00055 "m" (*dest), /* 0 mem, destination operand */
00056 "Ir" (b) /* 1, bit number */
00057 :
00058 "memory", "cc"
00059);
00060 }
00061
00062 /* change bit */
00063 #define __L4UTIL_BITOPS_HAVE_ARCH_COMPLEMENT_BIT
00064 L4_INLINE void
00065 l4util_complement_bit(int b, volatile l4_umword_t * dest)
00066 {
00067 __asm__ __volatile__
00068 (
00069 "lock; btc %1,%0 \n\t"
00070 :
00071 :
00072 "m" (*dest), /* 0 mem, destination operand */
00073 "Ir" (b) /* 1, bit number */
00074 :
00075 "memory", "cc"
00076);
00077 }
00078
00079 /* test bit */
00080 #define __L4UTIL_BITOPS_HAVE_ARCH_TEST_BIT
00081 L4_INLINE int
00082 l4util_test_bit(int b, const volatile l4_umword_t * dest)
00083 {
00084 l4_int8_t bit;
00085
00086 __asm__ __volatile__
00087 (
00088 "bt %2,%1 \n\t"
00089 "setc %0 \n\t"
00090 :
00091 "=r" (bit) /* 0, old bit value */
00092 :
00093 "m" (*dest), /* 1 mem, destination operand */
00094 "Ir" (b) /* 2, bit number */
00095 :
00096 "memory", "cc"
00097);
00098
00099 return (int)bit;
00100 }
00101
00102
00103 /* bit test and set */
00104 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_SET
00105 L4_INLINE int
00106 l4util_bts(int b, volatile l4_umword_t * dest)
00107 {
00108 l4_int8_t bit;
00109
00110 __asm__ __volatile__
00111 (
00112 "lock; bts %2,%1 \n\t"
00113 "setc %0 \n\t"
00114 :
00115 "=r" (bit) /* 0, old bit value */
00116 :
00117 "m" (*dest), /* 1 mem, destination operand */
00118 "Ir" (b) /* 2, bit number */
00119 :
00120 "memory", "cc"

```

```

00121);
00122
00123 return (int)bit;
00124 }
00125
00126 /* bit test and reset */
00127 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_RESET
00128 L4_INLINE int
00129 l4util_btr(int b, volatile l4_umword_t * dest)
00130 {
00131 l4_int8_t bit;
00132
00133 __asm__ __volatile__
00134 (
00135 "lock; btr %2,%1 \n\t"
00136 "setc %0 \n\t"
00137 :
00138 "=r" (bit) /* 0, old bit value */
00139 :
00140 "m" (*dest), /* 1 mem, destination operand */
00141 "Ir" (b) /* 2, bit number */
00142 :
00143 "memory", "cc"
00144);
00145
00146 return (int)bit;
00147 }
00148
00149 /* bit test and complement */
00150 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_COMPLEMENT
00151 L4_INLINE int
00152 l4util_btc(int b, volatile l4_umword_t * dest)
00153 {
00154 l4_int8_t bit;
00155
00156 __asm__ __volatile__
00157 (
00158 "lock; btc %2,%1 \n\t"
00159 "setc %0 \n\t"
00160 :
00161 "=r" (bit) /* 0, old bit value */
00162 :
00163 "m" (*dest), /* 1 mem, destination operand */
00164 "Ir" ((l4_umword_t)b) /* 2, bit number */
00165 :
00166 "memory", "cc"
00167);
00168
00169 return (int)bit;
00170 }
00171
00172 /* bit scan reverse */
00173 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_SCAN_REVERSE
00174 L4_INLINE int
00175 l4util_bsr(l4_umword_t word)
00176 {
00177 l4_umword_t tmp;
00178
00179 if (L4_UNLIKELY(word == 0))
00180 return -1;
00181
00182 __asm__ __volatile__
00183 (
00184 "bsr %1,%0 \n\t"
00185 :
00186 "=r" (tmp) /* 0, index of most significant set bit */
00187 :
00188 "r" (word) /* 1, argument */
00189);
00190
00191 return tmp;
00192 }
00193
00194 /* bit scan forward */
00195 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_SCAN_FORWARD
00196 L4_INLINE int
00197 l4util_bsf(l4_umword_t word)
00198 {
00199 l4_umword_t tmp;
00200
00201 if (L4_UNLIKELY(word == 0))
00202 return -1;
00203
00204 __asm__ __volatile__
00205 (
00206 "bsf %1,%0 \n\t"
00207 :

```

```

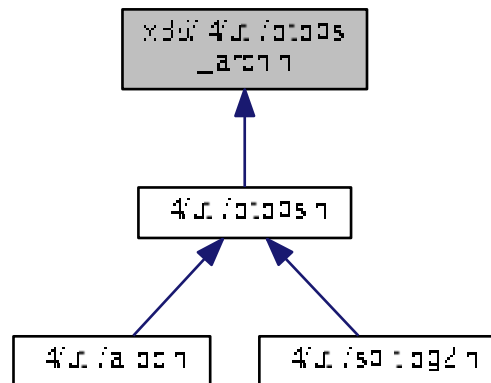
00208 "r" (tmp) /* 0, index of least significant set bit */
00209 :
00210 "r" (word) /* 1, argument */
00211);
00212
00213 return tmp;
00214 }
00215
00216 #define __L4UTIL_BITOPS_HAVE_ARCH_FIND_FIRST_SET_BIT
00217 L4_INLINE int
00218 l4util_find_first_set_bit(const void * dest, l4_size_t size)
00219 {
00220 l4_mword_t dummy0, dummy1, res;
00221
00222 __asm__ __volatile__
00223 (
00224 "xor %%rax,%%rax \n\t"
00225 "repe; scasl \n\t"
00226 "jz 1f \n\t"
00227 "lea -4(%%rdi),%%rdi \n\t"
00228 "bsfq (%%rdi),%%rax \n\t"
00229 "1: \n\t"
00230 "sub %%rbx,%%rdi \n\t"
00231 "shl $3,%%rdi \n\t"
00232 "add %%rdi,%%rax \n\t"
00233 :
00234 "a" (res), "=&c" (dummy0), "=&D" (dummy1)
00235 :
00236 "1" ((size + 31) >> 5), "2" (dest), "b" (dest)
00237 :
00238 "cc", "memory");
00239
00240 return res;
00241 }
00242
00243 #define __L4UTIL_BITOPS_HAVE_ARCH_FIND_FIRST_ZERO_BIT
00244 L4_INLINE int
00245 l4util_find_first_zero_bit(const void * dest,
00246 l4_size_t size)
00247 {
00248 l4_mword_t dummy0, dummy1, dummy2, res;
00249
00250 if (!size)
00251 return 0;
00252
00253 __asm__ __volatile__
00254 (
00255 "mov $-1,%%rax \n\t"
00256 "xor %%rdx,%%rdx \n\t"
00257 "repe; scasl \n\t"
00258 "je 1f \n\t"
00259 "xor -4(%%rdi),%%rax \n\t"
00260 "sub $4,%%rdi \n\t"
00261 "bsf %%rax,%%rdx \n\t"
00262 "1: \n\t"
00263 "sub %[dest],%%rdi \n\t"
00264 "shl $3,%%rdi \n\t"
00265 "add %%rdi,%%rdx \n\t"
00266 :
00267 "d" (res), "=&c" (dummy0), "=&D" (dummy1), "=&a" (dummy2)
00268 :
00269 "1" ((size + 31) >> 5), "2" (dest), [dest] "S" (dest)
00270 :
00271 "cc", "memory");
00272
00273 return res;
00274 }
00275 EXTERN_C_END
00276
00277 #endif /* ! __L4UTIL__INCLUDE__ARCH_AMD64__BITOPS_ARCH_H__ */

```

## 15.59 x86/I4/util/bitops\_arch.h File Reference

x86 bit manipulation functions

This graph shows which files directly or indirectly include this file:



### 15.59.1 Detailed Description

x86 bit manipulation functions

Date

07/03/2001

Author

Lars Reuther [reuther@os.inf.tu-dresden.de](mailto:reuther@os.inf.tu-dresden.de)

Definition in file [bitops\\_arch.h](#).

## 15.60 bitops\_arch.h

```

00001 /*****
00009 */
00010 * (c) 2000-2009 Author(s)
00011 * economic rights: Technische Universität Dresden (Germany)
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU Lesser General Public License 2.1.
00014 * Please see the COPYING-LGPL-2.1 file for details.
00015 */
00016
00017 /*****
00018 #ifndef __L4UTIL__INCLUDE__ARCH_X86__BITOPS_ARCH_H__
00019 #define __L4UTIL__INCLUDE__ARCH_X86__BITOPS_ARCH_H__
00020
00021 /*****
00022 *** Implementation
00023 *****/
00024
00025 EXTERN_C_BEGIN
00026
00027 /* set bit */
00028 #define __L4UTIL_BITOPS_HAVE_ARCH_SET_BIT

```

```

00029 L4_INLINE void
00030 l4util_set_bit(int b, volatile l4_umword_t * dest)
00031 {
00032 __asm__ __volatile__
00033 (
00034 "lock; btsl %1,%0 \n\t"
00035 :
00036 :
00037 "m" (*dest), /* 0 mem, destination operand */
00038 "Ir" (b) /* 1, bit number */
00039 :
00040 "memory", "cc"
00041);
00042 }
00043
00044 /* clear bit */
00045 #define __L4UTIL_BITOPS_HAVE_ARCH_CLEAR_BIT
00046 L4_INLINE void
00047 l4util_clear_bit(int b, volatile l4_umword_t * dest)
00048 {
00049 __asm__ __volatile__
00050 (
00051 "lock; btrl %1,%0 \n\t"
00052 :
00053 :
00054 "m" (*dest), /* 0 mem, destination operand */
00055 "Ir" (b) /* 1, bit number */
00056 :
00057 "memory", "cc"
00058);
00059 }
00060
00061 /* change bit */
00062 #define __L4UTIL_BITOPS_HAVE_ARCH_COMPLEMENT_BIT
00063 L4_INLINE void
00064 l4util_complement_bit(int b, volatile l4_umword_t * dest)
00065 {
00066 __asm__ __volatile__
00067 (
00068 "lock; btc1 %1,%0 \n\t"
00069 :
00070 :
00071 "m" (*dest), /* 0 mem, destination operand */
00072 "Ir" (b) /* 1, bit number */
00073 :
00074 "memory", "cc"
00075);
00076 }
00077
00078 /* test bit */
00079 #define __L4UTIL_BITOPS_HAVE_ARCH_TEST_BIT
00080 L4_INLINE int
00081 l4util_test_bit(int b, const volatile l4_umword_t * dest)
00082 {
00083 l4_int8_t bit;
00084
00085 __asm__ __volatile__
00086 (
00087 "btl %2,%1 \n\t"
00088 "setc %0 \n\t"
00089 :
00090 "=q" (bit) /* 0, old bit value */
00091 :
00092 "m" (*dest), /* 1 mem, destination operand */
00093 "Ir" (b) /* 2, bit number */
00094 :
00095 "memory", "cc"
00096);
00097
00098 return (int)bit;
00099 }
00100
00101 /* bit test and set */
00102 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_SET
00103 L4_INLINE int
00104 l4util_bts(int b, volatile l4_umword_t * dest)
00105 {
00106 l4_int8_t bit;
00107
00108 __asm__ __volatile__
00109 (
00110 "lock; btsl %2,%1 \n\t"
00111 "setc %0 \n\t"
00112 :
00113 "=q" (bit) /* 0, old bit value */
00114 :
00115 "m" (*dest), /* 1 mem, destination operand */

```

```

00116 "Ir" (b) /* 2, bit number */
00117 :
00118 "memory", "cc"
00119);
00120
00121 return (int)bit;
00122 }
00123
00124 /* bit test and reset */
00125 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_RESET
00126 L4_INLINE int
00127 l4util_btr(int b, volatile l4_umword_t * dest)
00128 {
00129 l4_int8_t bit;
00130
00131 __asm__ __volatile__
00132 (
00133 "lock; btrl %2,%1 \n\t"
00134 "setc %0 \n\t"
00135 :
00136 "=q" (bit) /* 0, old bit value */
00137 :
00138 "m" (*dest), /* 1 mem, destination operand */
00139 "Ir" (b) /* 2, bit number */
00140 :
00141 "memory", "cc"
00142);
00143
00144 return (int)bit;
00145 }
00146
00147 /* bit test and complement */
00148 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_TEST_AND_COMPLEMENT
00149 L4_INLINE int
00150 l4util_btc(int b, volatile l4_umword_t * dest)
00151 {
00152 l4_int8_t bit;
00153
00154 __asm__ __volatile__
00155 (
00156 "lock; btcl %2,%1 \n\t"
00157 "setc %0 \n\t"
00158 :
00159 "=q" (bit) /* 0, old bit value */
00160 :
00161 "m" (*dest), /* 1 mem, destination operand */
00162 "Ir" (b) /* 2, bit number */
00163 :
00164 "memory", "cc"
00165);
00166
00167 return (int)bit;
00168 }
00169
00170 /* bit scan reverse */
00171 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_SCAN_REVERSE
00172 L4_INLINE int
00173 l4util_bsr(l4_umword_t word)
00174 {
00175 int tmp;
00176
00177 if (L4_UNLIKELY(word == 0))
00178 return -1;
00179
00180 __asm__ __volatile__
00181 (
00182 "bsrl %1,%0 \n\t"
00183 :
00184 "=r" (tmp) /* 0, index of most significant set bit */
00185 :
00186 "r" (word) /* 1, argument */
00187);
00188
00189 return tmp;
00190 }
00191
00192 /* bit scan forward */
00193 #define __L4UTIL_BITOPS_HAVE_ARCH_BIT_SCAN_FORWARD
00194 L4_INLINE int
00195 l4util_bsfl(l4_umword_t word)
00196 {
00197 int tmp;
00198
00199 if (L4_UNLIKELY(word == 0))
00200 return -1;
00201
00202 __asm__ __volatile__

```

```

00203 (
00204 "bsfl %1,%0 \n\t"
00205 :
00206 "=r" (tmp) /* 0, index of least significant set bit */
00207 :
00208 "r" (word) /* 1, argument */
00209);
00210
00211 return tmp;
00212 }
00213
00214 #define __L4UTIL_BITOPS_HAVE_ARCH_FIND_FIRST_SET_BIT
00215 L4_INLINE int
00216 l4util_find_first_set_bit(const void * dest, l4_size_t size)
00217 {
00218 l4_mword_t dummy0, dummy1, res;
00219
00220 __asm__ __volatile__
00221 (
00222 "repe; scasl \n\t"
00223 "jz 1f \n\t"
00224 "leal -4(%edi),%edi \n\t"
00225 "bsfl (%edi),%eax \n\t"
00226 "1: \n\t"
00227 "subl %%esi,%edi \n\t"
00228 "shll $3,%edi \n\t"
00229 "addl %edi,%eax \n\t"
00230 :
00231 "=a" (res), "=c" (dummy0), "=D" (dummy1)
00232 :
00233 "a" (0), "c" ((size+31) >> 5), "D" (dest), "S" (dest)
00234 :
00235 "cc", "memory");
00236
00237 return res;
00238 }
00239
00240 #define __L4UTIL_BITOPS_HAVE_ARCH_FIND_FIRST_ZERO_BIT
00241 L4_INLINE int
00242 l4util_find_first_zero_bit(const void * dest,
00243 l4_size_t size)
00244 {
00245 l4_mword_t dummy0, dummy1, dummy2, res;
00246
00247 if (!size)
00248 return 0;
00249
00250 __asm__ __volatile__
00251 (
00252 "repe; scasl \n\t"
00253 "je 1f \n\t"
00254 "xorl -4(%edi),%eax \n\t"
00255 "subl $4,%edi \n\t"
00256 "bsfl %%eax,%%edx \n\t"
00257 "1: \n\t"
00258 "subl %%esi,%edi \n\t"
00259 "shll $3,%edi \n\t"
00260 "addl %edi,%%edx \n\t"
00261 :
00262 "=d" (res), "=c" (dummy0), "=D" (dummy1), "=a" (dummy2)
00263 :
00264 "a" (~0), "c" ((size+31) >> 5), "d" (0), "D" (dest), "S" (dest)
00265 :
00266 "cc", "memory");
00267
00268 return res;
00269 }
00270 EXTERN_C_END
00271
00272 #endif /* ! __L4UTIL__INCLUDE__ARCH_X86__BITOPS_ARCH_H__ */

```

## 15.61 arm/l4/util/cpu.h File Reference

CPU related functions.

### 15.61.1 Detailed Description

CPU related functions.

**Author**

Adam Lackorzynski [adam@os.inf.tu-dresden.de](mailto:adam@os.inf.tu-dresden.de)

Definition in file [cpu.h](#).

**15.62 cpu.h**

```

00001
00008 /*
00009 * (c) 2004-2009 Author(s)
00010 * economic rights: Technische Universität Dresden (Germany)
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU Lesser General Public License 2.1.
00013 * Please see the COPYING-LGPL-2.1 file for details.
00014 */
00015
00016 #ifndef __L4_UTIL__ARCH_ARM__CPU_H__
00017 #define __L4_UTIL__ARCH_ARM__CPU_H__
00018
00019 /* Nothing yet */
00020
00021 #endif /* __L4_UTIL__ARCH_ARM__CPU_H__ */

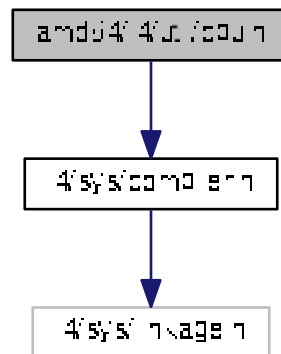
```

**15.63 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.*



### 15.63.1 Detailed Description

CPU related functions.

#### Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [cpu.h](#).

## 15.64 cpu.h

```

00001
00007 /*
00008 * (c) 2004-2009 Author(s)
00009 * economic rights: Technische Universität Dresden (Germany)
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU Lesser General Public License 2.1.
00012 * Please see the COPYING-LGPL-2.1 file for details.
00013 */
00014
00015 #ifndef __L4_UTIL_CPU_H
00016 #define __L4_UTIL_CPU_H
00017
00018 #include <14/sys/compiler.h>
00019
00020 EXTERN_C_BEGIN
00021
00027
00033 L4_INLINE int l4util_cpu_has_cpuid(void);
00034
00041 L4_INLINE unsigned int l4util_cpu_capabilities(void);
00042
00048 L4_INLINE unsigned int l4util_cpu_capabilities_nocheck(void);
00049
00053 L4_INLINE void
00054 l4util_cpu_cpuid(unsigned long mode,
00055 unsigned long *eax, unsigned long *ebx,
00056 unsigned long *ecx, unsigned long *edx);
00057
00059 static inline void
00060 l4util_cpu_pause(void)
00061 {
00062 __asm__ __volatile__ ("rep; nop");
00063 }
00064
00065 L4_INLINE int
00066 l4util_cpu_has_cpuid(void)
00067 {
00068 return 1;
00069 }
00070
00071 L4_INLINE void
00072 l4util_cpu_cpuid(unsigned long mode,
00073 unsigned long *eax, unsigned long *ebx,
00074 unsigned long *ecx, unsigned long *edx)
00075 {
00076 asm volatile("cpuid"
00077 : "=a" (*eax),
00078 "=b" (*ebx),
00079 "=c" (*ecx),
00080 "=d" (*edx)
00081 : "a" (mode)
00082);
00083 }
00084
00085 L4_INLINE unsigned int
00086 l4util_cpu_capabilities_nocheck(void)
00087 {
00088 unsigned long dummy, capability;
00089
00090 /* get CPU capabilities */
00091 l4util_cpu_cpuid(1, &dummy, &dummy, &dummy, &capability);
00092
00093 return capability;
00094 }

```

```

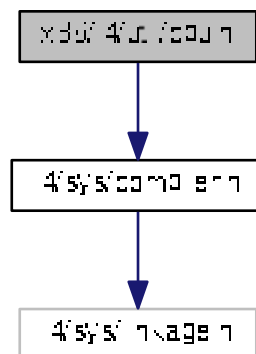
00095
00096 L4_INLINE unsigned int
00097 l4util_cpu_capabilities(void)
00098 {
00099 if (!l4util_cpu_has_cpuid())
00100 return 0; /* CPU has not cpuid instruction */
00101 return l4util_cpu_capabilities_nocheck();
00102 }
00103
00104 EXTERN_C_END
00105
00106 #endif
00107
00108

```

## 15.65 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\\_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.*

## 15.65.1 Detailed Description

CPU related functions.

### Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [cpu.h](#).

## 15.66 cpu.h

```

00001
00007 /*
00008 * (c) 2004-2009 Author(s)
00009 * economic rights: Technische Universität Dresden (Germany)
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU Lesser General Public License 2.1.
00012 * Please see the COPYING-LGPL-2.1 file for details.
00013 */
00014
00015 #ifndef __L4_UTIL_CPU_H
00016 #define __L4_UTIL_CPU_H
00017
00018 #include <l4/sys/compiler.h>
00019
00020 EXTERN_C_BEGIN
00021
00027
00033 L4_INLINE int l4util_cpu_has_cpuid(void);
00034
00041 L4_INLINE unsigned int l4util_cpu_capabilities(void);
00042
00048 L4_INLINE unsigned int l4util_cpu_capabilities_nocheck(void);
00049
00053 L4_INLINE void
00054 l4util_cpu_cpuid(unsigned long mode,
00055 unsigned long *eax, unsigned long *ebx,
00056 unsigned long *ecx, unsigned long *edx);
00057
00059 static inline void
00060 l4util_cpu_pause(void)
00061 {
00062 __asm__ __volatile__ ("rep; nop");
00063 }
00064
00065 L4_INLINE int
00066 l4util_cpu_has_cpuid(void)
00067 {
00068 unsigned long eax;
00069
00070 asm volatile("pushl %%ebx \t\n"
00071 "pushfl \t\n"
00072 "popl %%eax \t\n" /* get eflags */
00073 "movl %%eax, %%ebx \t\n" /* save it */
00074 "xorl $0x200000, %%eax \t\n" /* toggle ID bit */
00075 "pushl %%eax \t\n"
00076 "popfl \t\n" /* set again */
00077 "pushfl \t\n"
00078 "popl %%eax \t\n" /* get it again */
00079 "xorl %%ebx, %%eax \t\n"
00080 "pushl %%ebx \t\n"
00081 "popfl \t\n" /* restore saved flags */
00082 "popl %%ebx \t\n"
00083 : "=a" (eax)
00084 : /* no input */
00085);
00086
00087 return eax & 0x200000;
00088 }
00089
00090 L4_INLINE void
00091 l4util_cpu_cpuid(unsigned long mode,
00092 unsigned long *eax, unsigned long *ebx,
00093 unsigned long *ecx, unsigned long *edx)
00094 {

```

```

00095 asm volatile("pushl %%ebx \t\n"
00096 "cpuid \t\n"
00097 "mov %%ebx, %%esi \t\n"
00098 "popl %%ebx \t\n"
00099 : "=a" (*eax),
00100 "=S" (*ebx),
00101 "=c" (*ecx),
00102 "=d" (*edx)
00103 : "a" (mode));
00104 }
00105
00106 L4_INLINE unsigned int
00107 l4util_cpu_capabilities_nocheck(void)
00108 {
00109 unsigned long dummy, capability;
00110
00111 /* get CPU capabilities */
00112 l4util_cpu_cpuid(1, &dummy, &dummy, &dummy, &capability);
00113
00114 return capability;
00115 }
00116
00117 L4_INLINE unsigned int
00118 l4util_cpu_capabilities(void)
00119 {
00120 if (!l4util_cpu_has_cpuid())
00121 return 0; /* CPU has not cpuid instruction */
00122
00123 return l4util_cpu_capabilities_nocheck();
00124 }
00125
00126 EXTERN_C_END
00127
00128 #endif
00129

```

## 15.67 arm/l4/util/l4\_macros.h File Reference

Main function.

### 15.67.1 Detailed Description

Main function.

#### Date

08/29/2000

#### Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [l4\\_macros.h](#).

## 15.68 l4\_macros.h

```

00001
00008 /*
00009 * (c) 2006-2009 Author(s)
00010 * economic rights: Technische Universität Dresden (Germany)
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU Lesser General Public License 2.1.
00013 * Please see the COPYING-LGPL-2.1 file for details.
00014 */

```

```

00015
00016 #ifndef _L4UTIL__ARCH_ARM__L4_MACROS_H
00017 #define _L4UTIL__ARCH_ARM__L4_MACROS_H
00018
00019 #include_next <l4/util/l4_macros.h>
00020
00021 #ifndef l4_addr_fmt
00022 # define l4_addr_fmt "%08lx"
00023 #endif
00024
00025 #endif /* !_L4UTIL__ARCH_ARM__L4_MACROS_H */

```

## 15.69 amd64/l4/util/l4\_macros.h File Reference

Main function.

### 15.69.1 Detailed Description

Main function.

Date

08/29/2000

Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [l4\\_macros.h](#).

## 15.70 l4\_macros.h

```

00001
00008 /*
00009 * (c) 2006-2009 Author(s)
00010 * economic rights: Technische Universität Dresden (Germany)
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU Lesser General Public License 2.1.
00013 * Please see the COPYING-LGPL-2.1 file for details.
00014 */
00015
00016 #ifndef _L4UTIL__ARCH_AMD64__L4_MACROS_H
00017 #define _L4UTIL__ARCH_AMD64__L4_MACROS_H
00018
00019 #include_next <l4/util/l4_macros.h>
00020
00021 #ifndef l4_addr_fmt
00022 # define l4_addr_fmt "%016lx"
00023 #endif
00024
00025 #endif /* !_L4UTIL__ARCH_AMD64__L4_MACROS_H */

```

## 15.71 l4/util/l4\_macros.h File Reference

Some useful generic macros, L4f version.

### 15.71.1 Detailed Description

Some useful generic macros, L4f version.

#### Date

11/12/2002

#### Author

Lars Reuther [reuther@os.inf.tu-dresden.de](mailto:reuther@os.inf.tu-dresden.de)

Definition in file [l4\\_macros.h](#).

## 15.72 l4\_macros.h

```

00001 /*****
00002 */
00003 * (c) 2000-2009 Author(s)
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 /*****
00011 */
00012 #pragma once
00013
00014 /*****
00015 */
00016
00017 *** generic macros
00018
00019
00020
00021
00022
00023
00024 /* generate L4 thread id printf string */
00025 #ifndef l4util_idstr
00026 # define l4util_idfmt "%lx"
00027 # define l4util_idfmt_adjust "%04lx"
00028 # define l4util_idstr(tid) (tid >> L4_CAP_SHIFT)
00029 #endif
00030

```

## 15.73 x86/l4/uti/l4\_macros.h File Reference

Main function.

### 15.73.1 Detailed Description

Main function.

#### Date

08/29/2000

#### Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [l4\\_macros.h](#).

## 15.74 l4\_macros.h

```

00001
00008 /*
00009 * (c) 2006-2009 Author(s)
00010 * economic rights: Technische Universität Dresden (Germany)
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU Lesser General Public License 2.1.
00013 * Please see the COPYING-LGPL-2.1 file for details.
00014 */
00015
00016
00017 #ifndef _L4UTIL__ARCH_X86__L4_MACROS_H
00018 #define _L4UTIL__ARCH_X86__L4_MACROS_H
00019
00020 #include_next <l4/util/l4_macros.h>
00021
00022 #ifndef l4_addr_fmt
00023 # define l4_addr_fmt "%08lx"
00024 #endif
00025
00026 #endif /* !_L4UTIL__ARCH_X86__L4_MACROS_H */

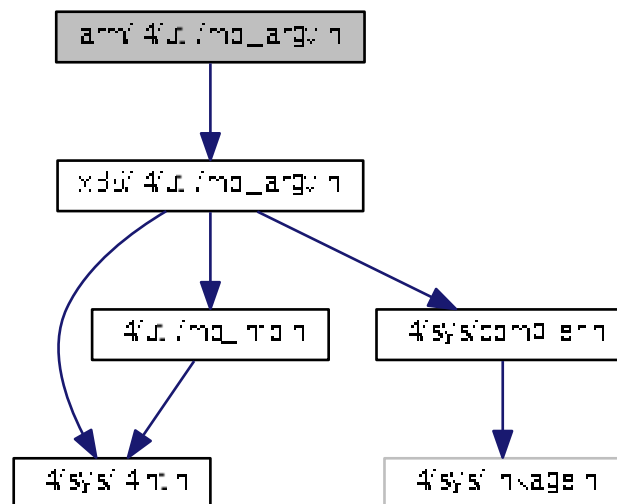
```

## 15.75 arm/l4/util/mbi\_argv.h File Reference

Multiboot.

```
#include <x86/l4/util/mbi_argv.h>
```

Include dependency graph for mbi\_argv.h:



### 15.75.1 Detailed Description

Multiboot.

Definition in file [mbi\\_argv.h](#).

## 15.76 mbi\_argv.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007 * economic rights: Technische Universität Dresden (Germany)
00008 * This file is part of TUD:OS and distributed under the terms of the
00009 * GNU Lesser General Public License 2.1.
00010 * Please see the COPYING-LGPL-2.1 file for details.
00011 */
00012 /* If this persists, move it to a generic place */
00013 #include <x86/l4/util/mbi_argv.h>

```

## 15.77 amd64/l4/util/mbi\_argv.h File Reference

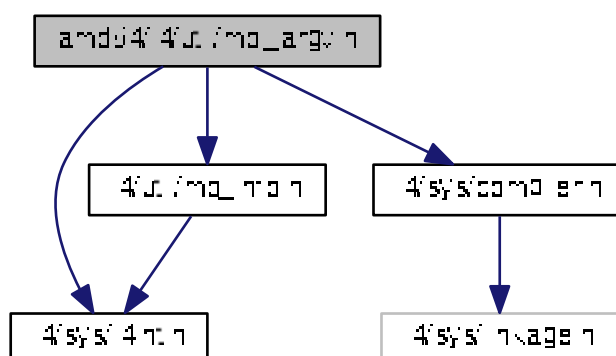
command line handling

```

#include <l4/sys/l4int.h>
#include <l4/util/mb_info.h>
#include <l4/sys/compiler.h>

```

Include dependency graph for mbi\_argv.h:



### 15.77.1 Detailed Description

command line handling

Date

2003

Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [mbi\\_argv.h](#).



## 15.78 mbi\_argv.h

```

00001
00008 /*
00009 * (c) 2003-2009 Author(s)
00010 * economic rights: Technische Universität Dresden (Germany)
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU Lesser General Public License 2.1.
00013 * Please see the COPYING-LGPL-2.1 file for details.
00014 */
00015
00016 #ifndef L4UTIL_MBI_ARGV
00017 #define L4UTIL_MBI_ARGV
00018
00019 #include <l4/sys/l4int.h>
00020 #include <l4/util/mb_info.h>
00021 #include <l4/sys/compiler.h>
00022
00023 EXTERN_C_BEGIN
00024
00025 L4_CV void l4util_mbi_to_argv(l4_mword_t flag, l4util_mb_info_t *mbi);
00026
00027 extern int l4util_argc;
00028 extern char *l4util_argv[];
00029
00030 EXTERN_C_END
00031
00032 #endif
00033

```

## 15.79 x86/l4/util/mbi\_argv.h File Reference

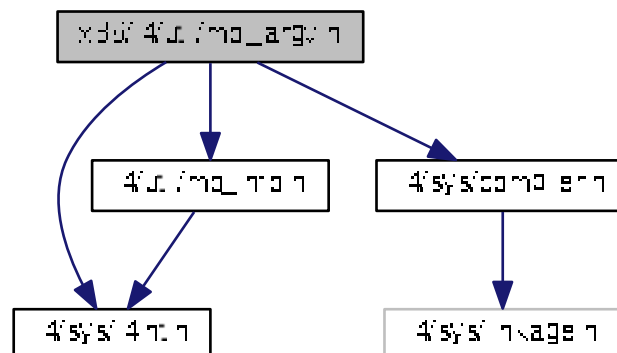
command line handling

```

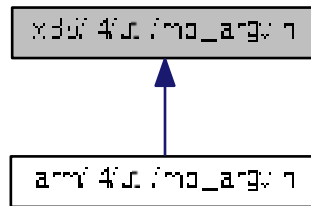
#include <l4/sys/l4int.h>
#include <l4/util/mb_info.h>
#include <l4/sys/compiler.h>

```

Include dependency graph for mbi\_argv.h:



This graph shows which files directly or indirectly include this file:



### 15.79.1 Detailed Description

command line handling

#### Date

2003

#### Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [mbi\\_argv.h](#).

## 15.80 mbi\_argv.h

```

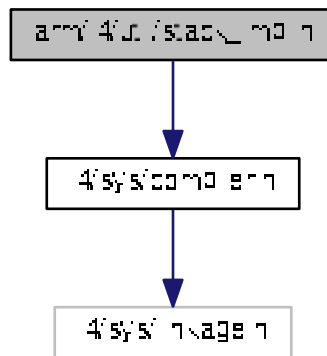
00001
00008 /*
00009 * (c) 2003-2009 Author(s)
00010 * economic rights: Technische Universität Dresden (Germany)
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU Lesser General Public License 2.1.
00013 * Please see the COPYING-LGPL-2.1 file for details.
00014 */
00015
00016 #ifndef L4UTIL_MBI_ARGV
00017 #define L4UTIL_MBI_ARGV
00018
00019 #include <l4/sys/l4int.h>
00020 #include <l4/util/mb_info.h>
00021 #include <l4/sys/compiler.h>
00022
00023 EXTERN_C_BEGIN
00024
00025 void l4util_mbi_to_argv(l4_mword_t flag, l4util_mb_info_t *mbi);
00026
00027 extern int l4util_argc;
00028 extern char *l4util_argv[];
00029
00030 EXTERN_C_END
00031
00032 #endif
00033

```

## 15.81 arm/l4/util/stack\_impl.h File Reference

Stack utilities, arm version.

```
#include <l4/sys/compiler.h>
Include dependency graph for stack_impl.h:
```



### Functions

- [l4\\_addr\\_t l4util\\_stack\\_get\\_sp](#) (void)  
*Get current stack pointer.*

#### 15.81.1 Detailed Description

Stack utilities, arm version.

Definition in file [stack\\_impl.h](#).

#### 15.81.2 Function Documentation

##### 15.81.2.1 l4util\_stack\_get\_sp()

```
l4_addr_t l4util_stack_get_sp (
 void) [inline]
```

Get current stack pointer.

##### Returns

stack pointer.

Definition at line 23 of file [stack\\_impl.h](#).

## 15.82 stack\_impl.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007 * economic rights; Technische Universität Dresden (Germany)
00008 * This file is part of TUD:OS and distributed under the terms of the
00009 * GNU Lesser General Public License 2.1.
00010 * Please see the COPYING-LGPL-2.1 file for details.
00011 */
00012 #ifndef __L4UTIL__INCLUDE__ARCH_ARM__STACK_IMPL_H__
00013 #define __L4UTIL__INCLUDE__ARCH_ARM__STACK_IMPL_H__
00014
00015 #include <l4/sys/compiler.h>
00016
00017 EXTERN_C_BEGIN
00018
00019 #ifndef _L4UTIL_STACK_H
00020 #error Do not include stack_impl.h directly, use stack.h instead
00021 #endif
00022
00023 L4_INLINE l4_addr_t l4util_stack_get_sp(void)
00024 {
00025 register l4_addr_t sp asm ("sp");
00026 return sp;
00027 }
00028
00029 EXTERN_C_END
00030
00031 #endif /* ! __L4UTIL__INCLUDE__ARCH_ARM__STACK_IMPL_H__ */

```

## 15.83 amd64/l4/util/stack\_impl.h File Reference

Stack utilities for amd64.

### Functions

- [l4\\_addr\\_t l4util\\_stack\\_get\\_sp](#) (void)  
*Get current stack pointer.*

### 15.83.1 Detailed Description

Stack utilities for amd64.

Definition in file [stack\\_impl.h](#).

### 15.83.2 Function Documentation

#### 15.83.2.1 l4util\_stack\_get\_sp()

```

l4_addr_t l4util_stack_get_sp (
 void) [inline]

```

Get current stack pointer.

#### Returns

stack pointer.

Definition at line 22 of file [stack\\_impl.h](#).

References [EXTERN\\_C\\_END](#).

## 15.84 stack\_impl.h

```

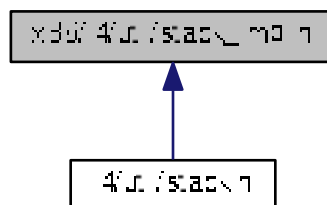
00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 * This file is part of TUD:OS and distributed under the terms of the
00010 * GNU Lesser General Public License 2.1.
00011 * Please see the COPYING-LGPL-2.1 file for details.
00012 */
00013 #ifndef __L4UTIL__INCLUDE__ARCH_AMD64__STACK_IMPL_H__
00014 #define __L4UTIL__INCLUDE__ARCH_AMD64__STACK_IMPL_H__
00015
00016 EXTERN_C_BEGIN
00017
00018 #ifndef __L4UTIL__STACK_H
00019 #error Do not include stack_impl.h directly, use stack.h instead
00020 #endif
00021
00022 L4_INLINE l4_addr_t l4util_stack_get_sp(void)
00023 {
00024 l4_addr_t rsp;
00025
00026 asm("movq %%rsp, %0\n\t" : "=r" (rsp) :);
00027 return rsp;
00028 }
00029
00030 EXTERN_C_END
00031
00032 #endif /* ! __L4UTIL__INCLUDE__ARCH_AMD64__STACK_IMPL_H__ */

```

## 15.85 x86/l4/util/stack\_impl.h File Reference

Stack utilities for x86.

This graph shows which files directly or indirectly include this file:



### Functions

- `l4_addr_t l4util_stack_get_sp (void)`  
Get current stack pointer.

### 15.85.1 Detailed Description

Stack utilities for x86.

Definition in file [stack\\_impl.h](#).

## 15.85.2 Function Documentation

### 15.85.2.1 l4util\_stack\_get\_sp()

```
l4_addr_t l4util_stack_get_sp (
 void) [inline]
```

Get current stack pointer.

#### Returns

stack pointer.

Definition at line 21 of file [stack\\_impl.h](#).

References [EXTERN\\_C\\_END](#).

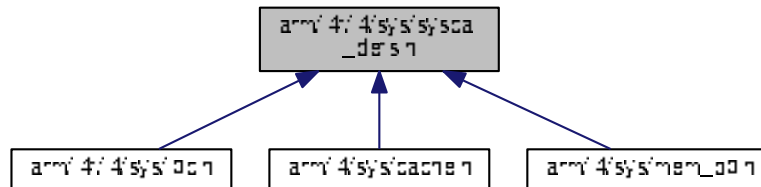
## 15.86 stack\_impl.h

```
00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007 * economic rights: Technische Universität Dresden (Germany)
00008 * This file is part of TUD:OS and distributed under the terms of the
00009 * GNU Lesser General Public License 2.1.
00010 * Please see the COPYING-LGPL-2.1 file for details.
00011 */
00012 #ifndef __L4UTIL__INCLUDE__ARCH_X86__STACK_IMPL_H__
00013 #define __L4UTIL__INCLUDE__ARCH_X86__STACK_IMPL_H__
00014
00015 #ifndef _L4UTIL_STACK_H
00016 #error Do not include stack_impl.h directly, use stack.h instead
00017 #endif
00018
00019 EXTERN_C_BEGIN
00020
00021 L4_INLINE l4_addr_t l4util_stack_get_sp(void)
00022 {
00023 l4_addr_t esp;
00024
00025 asm("movl %%esp, %0\n\t" : "=r" (esp) :);
00026 return esp;
00027 }
00028
00029 EXTERN_C_END
00030
00031 #endif /* ! __L4UTIL__INCLUDE__ARCH_ARM__STACK_IMPL_H__ */
```

## 15.87 arm/l4f/l4/sys/syscall\_defs.h File Reference

Syscall entry definitions.

This graph shows which files directly or indirectly include this file:



### 15.87.1 Detailed Description

Syscall entry definitions.

Definition in file [syscall\\_defs.h](#).

## 15.88 syscall\_defs.h

```

00001
00005 /*
00006 * (c) 2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007 * economic rights: Technische Universität Dresden (Germany)
00008 *
00009 * This file is part of TUD:OS and distributed under the terms of the
00010 * GNU General Public License 2.
00011 * Please see the COPYING-GPL-2 file for details.
00012 *
00013 * As a special exception, you may use this file as part of a free software
00014 * library without restriction. Specifically, if other files instantiate
00015 * templates or use macros or inline functions from this file, or you compile
00016 * this file and link it with other files to produce an executable, this
00017 * file does not by itself cause the resulting executable to be covered by
00018 * the GNU General Public License. This exception does not however
00019 * invalidate any other reasons why the executable file might be covered by
00020 * the GNU General Public License.
00021 */
00022 #ifndef __L4SYS__ARCH_ARM__L4API_L4F__SYSCALL_DEFS_H__
00023 #define __L4SYS__ARCH_ARM__L4API_L4F__SYSCALL_DEFS_H__
00024
00025 #ifndef L4_SYSCALL_MAGIC_OFFSET
00026 # define L4_SYSCALL_MAGIC_OFFSET 8
00027 #endif
00028 #define L4_SYSCALL_INVOKE (-0x00000004-L4_SYSCALL_MAGIC_OFFSET)
00029 #define L4_SYSCALL_MEM_OP (-0x00000008-L4_SYSCALL_MAGIC_OFFSET)
00030 #define L4_SYSCALL_DEBUGGER (-0x0000000C-L4_SYSCALL_MAGIC_OFFSET)
00031
00032 #endif /* __L4SYS__ARCH_ARM__L4API_L4F__SYSCALL_DEFS_H__ */

```

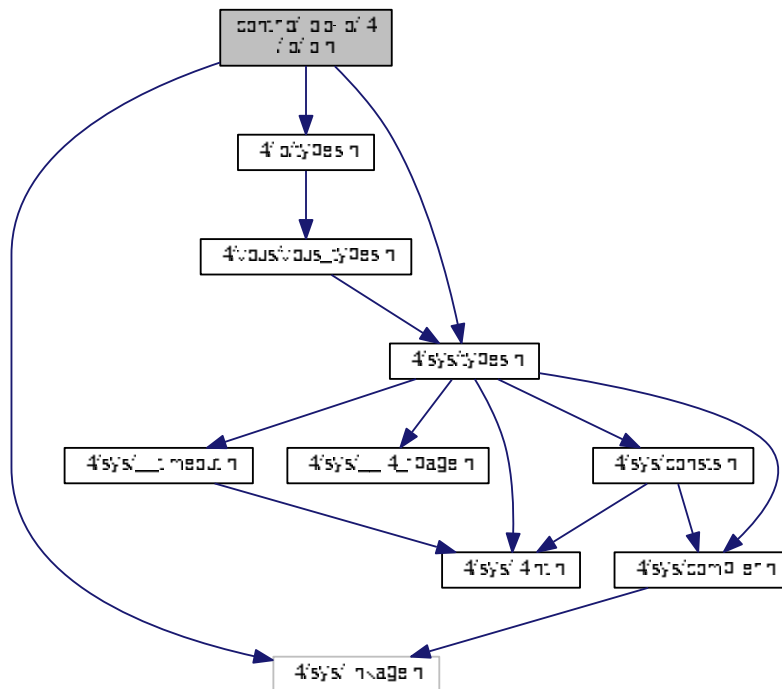
## 15.89 contrib/libio-io/l4/io/io.h File Reference

```

#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_search_iomem_region` (`l4_addr_t` phys, `l4_addr_t` size, `l4_addr_t` \*rstart, `l4_addr_t` \*rsize)  
*Search for a 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.*

## 15.89.1 Function Documentation

### 15.89.1.1 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 268 of file [io.h](#).

References [EXTERN\\_C\\_END](#).

### 15.89.1.2 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

|                           |                                     |
|---------------------------|-------------------------------------|
| <a href="#">devhandle</a> | Device handle to start iterating at |
|---------------------------|-------------------------------------|

#### Return values

|                           |                    |
|---------------------------|--------------------|
| <a href="#">devhandle</a> | Next device handle |
|---------------------------|--------------------|

## Return values

|                  |                                 |
|------------------|---------------------------------|
| <i>dev</i>       | Device information, may be NULL |
| <i>reshandle</i> | Resource handle.                |

## Returns

0 on success, error code otherwise

## 15.89.1.3 l4io\_request\_all\_ioports()

```
void l4io_request_all_ioports (
 void(*) (l4vbus_resource_t const *res) res_cb)
```

Request all available IO-port resources.

## Parameters

|               |                                                                                    |
|---------------|------------------------------------------------------------------------------------|
| <i>res_cb</i> | Callback function called for every port resource found, give NULL for no callback. |
|---------------|------------------------------------------------------------------------------------|

## 15.89.1.4 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.

## 15.90 io.h

```
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 * This file is part of TUD:OS and distributed under the terms of the
00009 * GNU Lesser General Public License 2.1.
00010 * Please see the COPYING-LGPL-2.1 file for details.
00011 */
00012
00013
00014 #pragma once
00015
00016 #include <l4/sys/types.h>
00017 #include <l4/sys/linkage.h>
00018 #include <l4/io/types.h>
00019
```

```

00020 EXTERN_C_BEGIN
00021
00038 L4_CV long L4_EXPORT
00039 l4io_request_irq(int irqnum, l4_cap_idx_t irqcap);
00040
00047 L4_CV l4_cap_idx_t L4_EXPORT
00048 l4io_request_icu(void);
00049
00061 L4_CV long L4_EXPORT
00062 l4io_release_irq(int irqnum, l4_cap_idx_t irq_cap);
00063
00088 L4_CV long L4_EXPORT
00089 l4io_request_iomem(l4_addr_t phys, unsigned long size, int flags,
00090 l4_addr_t *virt);
00091
00113 L4_CV long L4_EXPORT
00114 l4io_request_iomem_region(l4_addr_t phys,
00115 l4_addr_t virt,
00116 unsigned long size, int flags);
00116
00124 L4_CV long L4_EXPORT
00125 l4io_release_iomem(l4_addr_t virt, unsigned long size);
00126
00136 L4_CV long L4_EXPORT
00137 l4io_search_iomem_region(l4_addr_t phys,
00138 l4_addr_t size,
00139 l4_addr_t *rstart, l4_addr_t *rsize);
00139
00149 L4_CV long L4_EXPORT
00150 l4io_request_ioport(unsigned portnum, unsigned len);
00151
00161 L4_CV long L4_EXPORT
00162 l4io_release_ioport(unsigned portnum, unsigned len);
00163
00164
00165 /* ----- Device handling ----- */
00166
00172 L4_INLINE
00173 l4io_device_handle_t l4io_get_root_device(void);
00174
00184 L4_CV int L4_EXPORT
00185 l4io_iterate_devices(l4io_device_handle_t *devhandle,
00186 l4io_device_t *dev, l4io_resource_handle_t *reshandle);
00187
00199 L4_CV int L4_EXPORT
00200 l4io_lookup_device(const char *devname,
00201 l4io_device_handle_t *dev_handle,
00202 l4io_device_t *dev, l4io_resource_handle_t *res_handle);
00203
00218 L4_CV int L4_EXPORT
00219 l4io_lookup_resource(l4io_device_handle_t devhandle,
00220 enum l4io_resource_types_t type,
00221 l4io_resource_handle_t *reshandle,
00222 l4io_resource_t *res);
00223
00224
00225 /* ----- Convenience functions ----- */
00226
00239 L4_CV l4_addr_t L4_EXPORT
00240 l4io_request_resource_iomem(l4io_device_handle_t devhandle,
00241 l4io_resource_handle_t *reshandle);
00242
00249 L4_CV void L4_EXPORT
00250 l4io_request_all_ioports(void (*res_cb)(
00251 l4vbus_resource_t const *res));
00251
00260 L4_CV int L4_EXPORT
00261 l4io_has_resource(enum l4io_resource_types_t type,
00262 l4vbus_paddr_t start, l4vbus_paddr_t end);
00263
00264 /* ----- Implementations ----- */
00265 /* Implementations */
00266
00267 L4_INLINE
00268 l4io_device_handle_t l4io_get_root_device(void)
00269 { return 0; }
00270
00271 EXTERN_C_END

```

## 15.91 l4/sys/types.h File Reference

Common L4 ABI Data Types.



```

L4_PROTO_SIGMA0 = -6L, L4_PROTO_IO_PAGE_FAULT = -8L, 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_IRQ_MUX = -19L,
L4_PROTO_SEMAPHORE = -20L, L4_PROTO_META = -21L, L4_PROTO_IOMMU = -22L, L4_PROTO_DEBUGGER = -23L }

```

*Message tag for IPC operations.*

- enum `l4_msgtag_flags` {  
`L4_MSGTAG_ERROR`, `L4_MSGTAG_TRANSFER_FPU`, `L4_MSGTAG_SCHEDULE`, `L4_MSGTAG_PRIVILEGE`,  
`L4_MSGTAG_FLAGS` }

*Flags for message tags.*

## Functions

- `l4_msgtag_t l4_msgtag` (long label, unsigned words, unsigned items, unsigned flags) `L4_NOTHROW`  
*Create a message tag from the specified values.*
- 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_preemption` (`l4_msgtag_t` t) `L4_NOTHROW`  
*Test for preemption protocol.*
- unsigned `l4_msgtag_is_sys_exception` (`l4_msgtag_t` t) `L4_NOTHROW`  
*Test for system-exception 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.*
- 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 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.*

### 15.91.1 Detailed Description

Common [L4](#) ABI Data Types.

Definition in file [types.h](#).

### 15.91.2 Function Documentation

#### 15.91.2.1 l4\_capability\_next()

```
l4_cap_idx_t l4_capability_next (
 l4_cap_idx_t c) [inline]
```

Get the next capability selector after `c`.

#### Parameters

|                |                                                                        |
|----------------|------------------------------------------------------------------------|
| <code>c</code> | The capability selector for which the next selector shall be computed. |
|----------------|------------------------------------------------------------------------|

#### Returns

The next capability selector after `c`.

Definition at line [458](#) of file [types.h](#).

## 15.92 types.h

```
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 * This file is part of TUD:OS and distributed under the terms of the
00010 * GNU General Public License 2.
00011 * Please see the COPYING-GPL-2 file for details.
00012 *
00013 * As a special exception, you may use this file as part of a free software
00014 * library without restriction. Specifically, if other files instantiate
00015 * templates or use macros or inline functions from this file, or you compile
00016 * this file and link it with other files to produce an executable, this
00017 * file does not by itself cause the resulting executable to be covered by
00018 * the GNU General Public License. This exception does not however
00019 * invalidate any other reasons why the executable file might be covered by
00020 * the GNU General Public License.
00021 */
00022 /*****/
00023 #pragma once
00024
00025 #include <l4/sys/l4int.h>
00026 #include <l4/sys/compiler.h>
00027 #include <l4/sys/consts.h>
00028
00029 enum l4_msgtag_protocol
```

```

00050 {
00051 L4_PROTO_NONE = 0,
00052 L4_PROTO_ALLOW_SYSCALL = 1,
00053 L4_PROTO_PF_EXCEPTION = 1,
00054
00055 L4_PROTO_IRQ = -1L,
00056 L4_PROTO_PAGE_FAULT = -2L,
00057 L4_PROTO_PREEMPTION = -3L,
00058 L4_PROTO_SYS_EXCEPTION = -4L,
00059 L4_PROTO_EXCEPTION = -5L,
00060 L4_PROTO_SIGMA0 = -6L,
00061 L4_PROTO_IO_PAGE_FAULT = -8L,
00062 L4_PROTO_KOBJECT = -10L,
00063 L4_PROTO_TASK = -11L,
00064 L4_PROTO_THREAD = -12L,
00065 L4_PROTO_LOG = -13L,
00066 L4_PROTO_SCHEDULER = -14L,
00067 L4_PROTO_FACTORY = -15L,
00068 L4_PROTO_VM = -16L,
00069 L4_PROTO_DMA_SPACE = -17L,
00070 L4_PROTO_IRQ_SENDER = -18L,
00071 L4_PROTO_IRQ_MUX = -19L,
00072 L4_PROTO_SEMAPHORE = -20L,
00073 L4_PROTO_META = -21L,
00074 L4_PROTO_IOMMU = -22L,
00075 L4_PROTO_DEBUGGER = -23L,
00076 };
00077
00078 enum L4_varg_type
00079 {
00080 L4_VARG_TYPE_NIL = 0x00,
00081 L4_VARG_TYPE_UMWORD = 0x01,
00082 L4_VARG_TYPE_MWORD = 0x81,
00083 L4_VARG_TYPE_STRING = 0x02,
00084 L4_VARG_TYPE_FPAGE = 0x03,
00085
00086 L4_VARG_TYPE_SIGN = 0x80,
00087 };
00088
00089
00094 enum l4_msgtag_flags
00095 {
00096 // flags for received IPC
00101 L4_MSGTAG_ERROR = 0x8000,
00102
00103 // flags for sending IPC
00113 L4_MSGTAG_TRANSFER_FPU = 0x1000,
00122 L4_MSGTAG_SCHEDULE = 0x2000,
00132 L4_MSGTAG_PROPAGATE = 0x4000,
00133
00138 L4_MSGTAG_FLAGS = 0xf000,
00139 };
00140
00141
00158 typedef struct l4_msgtag_t
00159 {
00160 l4_mword_t raw;
00161 #ifdef __cplusplus
00162 long label() const throw() { return raw >> 16; }
00165 void label(long v) throw() { raw = (raw & 0xffff) | (v << 16); }
00167 unsigned words() const throw() { return raw & 0x3f; }
00169 unsigned items() const throw() { return (raw >> 6) & 0x3f; }
00176 unsigned flags() const throw() { return raw & 0xf000; }
00178 bool is_page_fault() const throw() { return label() ==
00180 L4_PROTO_PAGE_FAULT; }
00182 bool is_preemption() const throw() { return label() ==
00184 L4_PROTO_PREEMPTION; }
00186 bool is_sys_exception() const throw() { return label() ==
00188 L4_PROTO_SYS_EXCEPTION; }
00190 bool is_exception() const throw() { return label() ==
00192 L4_PROTO_EXCEPTION; }
00194 bool is_sigma0() const throw() { return label() ==
00196 L4_PROTO_SIGMA0; }
00198 bool is_io_page_fault() const throw() { return label() ==
00200 L4_PROTO_IO_PAGE_FAULT; }
00202 unsigned has_error() const throw() { return raw & L4_MSGTAG_ERROR; }
00203 #endif
00204 } l4_msgtag_t;
00205
00206
00207 L4_INLINE l4_msgtag_t l4_msgtag(long label, unsigned
00208 words, unsigned items,
00209 unsigned flags) L4_NOTHROW;
00210
00211 L4_INLINE long l4_msgtag_label(l4_msgtag_t t)
00212 L4_NOTHROW;

```

```

00219
00228 L4_INLINE unsigned l4_msgtag_words(l4_msgtag_t t)
 L4_NOTHROW;
00229
00238 L4_INLINE unsigned l4_msgtag_items(l4_msgtag_t t)
 L4_NOTHROW;
00239
00250 L4_INLINE unsigned l4_msgtag_flags(l4_msgtag_t t)
 L4_NOTHROW;
00251
00264 L4_INLINE unsigned l4_msgtag_has_error(l4_msgtag_t t)
 L4_NOTHROW;
00265
00274 L4_INLINE unsigned l4_msgtag_is_page_fault(l4_msgtag_t t)
 L4_NOTHROW;
00275
00283 L4_INLINE unsigned l4_msgtag_is_preemption(l4_msgtag_t t)
 L4_NOTHROW;
00284
00293 L4_INLINE unsigned l4_msgtag_is_sys_exception(
 l4_msgtag_t t) L4_NOTHROW;
00294
00303 L4_INLINE unsigned l4_msgtag_is_exception(l4_msgtag_t t)
 L4_NOTHROW;
00304
00313 L4_INLINE unsigned l4_msgtag_is_sigma0(l4_msgtag_t t)
 L4_NOTHROW;
00314
00323 L4_INLINE unsigned l4_msgtag_is_io_page_fault(
 l4_msgtag_t t) L4_NOTHROW;
00324
00341 typedef unsigned long l4_cap_idx_t;
00342
00352 L4_INLINE unsigned l4_is_invalid_cap(l4_cap_idx_t c) L4_NOTHROW;
00353
00363 L4_INLINE unsigned l4_is_valid_cap(l4_cap_idx_t c) L4_NOTHROW;
00364
00375 L4_INLINE unsigned l4_capability_equal(l4_cap_idx_t c1, l4_cap_idx_t c2)
 L4_NOTHROW;
00376
00385 L4_INLINE l4_cap_idx_t l4_capability_next(l4_cap_idx_t c)
 L4_NOTHROW;
00386
00387 /* ***** */
00388 /* Implementation */
00389
00390 L4_INLINE unsigned
00391 l4_is_invalid_cap(l4_cap_idx_t c) L4_NOTHROW
00392 { return c & L4_INVALID_CAP_BIT; }
00393
00394 L4_INLINE unsigned
00395 l4_is_valid_cap(l4_cap_idx_t c) L4_NOTHROW
00396 { return !(c & L4_INVALID_CAP_BIT); }
00397
00398 L4_INLINE unsigned
00399 l4_capability_equal(l4_cap_idx_t c1, l4_cap_idx_t c2) L4_NOTHROW
00400 { return (c1 >> L4_CAP_SHIFT) == (c2 >> L4_CAP_SHIFT); }
00401
00402
00406 L4_INLINE
00407 l4_msgtag_t l4_msgtag(long label, unsigned words, unsigned items,
 unsigned flags) L4_NOTHROW
00408 {
00409 return (l4_msgtag_t){(label << 16) | (l4_mword_t)(words & 0x3f)
 | (l4_mword_t)((items & 0x3f) << 6)
 | (l4_mword_t)(flags & 0xf000)};
00410 }
00411
00412
00413 }
00414
00415
00416
00417 L4_INLINE
00418 long l4_msgtag_label(l4_msgtag_t t) L4_NOTHROW
00419 { return t.raw >> 16; }
00420
00421 L4_INLINE
00422 unsigned l4_msgtag_words(l4_msgtag_t t) L4_NOTHROW
00423 { return t.raw & 0x3f; }
00424
00425 L4_INLINE
00426 unsigned l4_msgtag_items(l4_msgtag_t t) L4_NOTHROW
00427 { return (t.raw >> 6) & 0x3f; }
00428
00429 L4_INLINE
00430 unsigned l4_msgtag_flags(l4_msgtag_t t) L4_NOTHROW
00431 { return t.raw & 0xf000; }
00432
00433

```



```

00434 L4_INLINE
00435 unsigned l4_msgtag_has_error(l4_msgtag_t t) L4_NOTHROW
00436 { return t.raw & L4_MSGTAG_ERROR; }
00437
00438
00439
00440 L4_INLINE unsigned l4_msgtag_is_page_fault(l4_msgtag_t t) L4_NOTHROW
00441 { return l4_msgtag_label(t) == L4_PROTO_PAGE_FAULT; }
00442
00443 L4_INLINE unsigned l4_msgtag_is_preemption(l4_msgtag_t t) L4_NOTHROW
00444 { return l4_msgtag_label(t) == L4_PROTO_PREEMPTION; }
00445
00446 L4_INLINE unsigned l4_msgtag_is_sys_exception(
00447 l4_msgtag_t t) L4_NOTHROW
00448 { return l4_msgtag_label(t) == L4_PROTO_SYS_EXCEPTION; }
00449
00449 L4_INLINE unsigned l4_msgtag_is_exception(l4_msgtag_t t) L4_NOTHROW
00450 { return l4_msgtag_label(t) == L4_PROTO_EXCEPTION; }
00451
00452 L4_INLINE unsigned l4_msgtag_is_sigma0(l4_msgtag_t t) L4_NOTHROW
00453 { return l4_msgtag_label(t) == L4_PROTO_SIGMA0; }
00454
00455 L4_INLINE unsigned l4_msgtag_is_io_page_fault(
00456 l4_msgtag_t t) L4_NOTHROW
00457 { return l4_msgtag_label(t) == L4_PROTO_IO_PAGE_FAULT; }
00458
00458 L4_INLINE l4_cap_idx_t l4_capability_next(l4_cap_idx_t c) L4_NOTHROW
00459 { return c + L4_CAP_OFFSET; }
00460
00461 #include <l4/sys/__l4_fpage.h>
00462 #include <l4/sys/__timeout.h>

```

## 15.93 l4/cxx/avl\_map File Reference

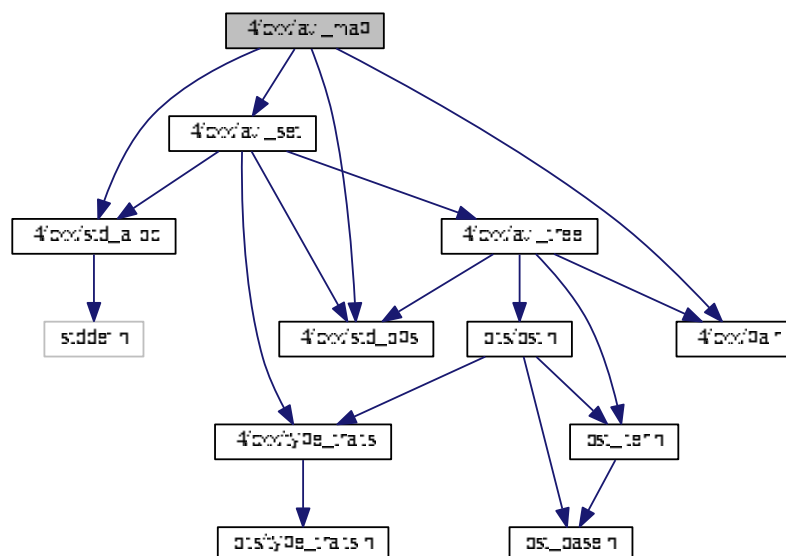
AVL map.

```

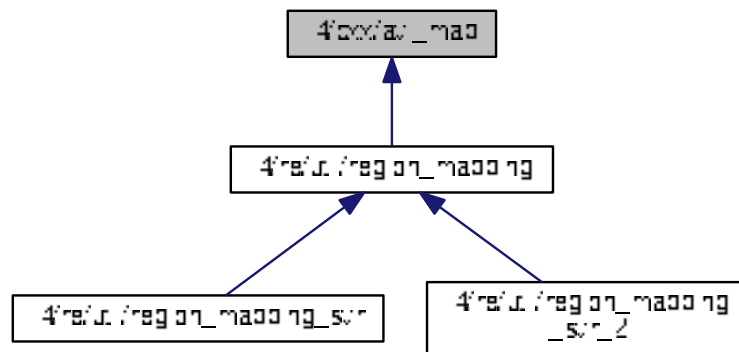
#include <l4/cxx/std_alloc>
#include <l4/cxx/std_ops>
#include <l4/cxx/pair>
#include <l4/cxx/avl_set>

```

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

- [cxx](#)  
*Our C++ library.*
- [cxx::Bits](#)  
*Internal helpers for the [cxx](#) package.*

### 15.93.1 Detailed Description

AVL map.

Definition in file [avl\\_map](#).

## 15.94 avl\_map

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007 * (c) 2008-2009 Alexander Warg <warg@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 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate

```

```

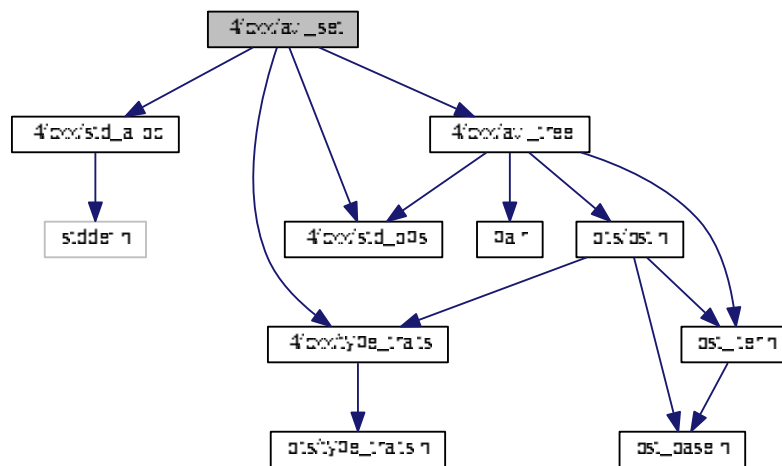
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
00019 * the GNU General Public License. This exception does not however
00020 * invalidate any other reasons why the executable file might be covered by
00021 * the GNU General Public License.
00022 */
00023
00024 #pragma once
00025
00026 #include <l4/cxx/std_alloc>
00027 #include <l4/cxx/std_ops>
00028 #include <l4/cxx/pair>
00029 #include <l4/cxx/avl_set>
00030
00031 namespace cxx {
00032 namespace Bits {
00033
00035 template<typename KEY_TYPE>
00036 struct Avl_map_get_key
00037 {
00038 typedef KEY_TYPE Key_type;
00039 template<typename NODE>
00040 static Key_type const &key_of(NODE const *n)
00041 { return n->item.first; }
00042 };
00043 }
00044
00053 template< typename KEY_TYPE, typename DATA_TYPE,
00054 template<typename A> class COMPARE = Lt_functor,
00055 template<typename B> class ALLOC = New_allocator >
00056 class Avl_map :
00057 public Bits::Base_avl_set<Pair<KEY_TYPE, DATA_TYPE>,
00058 COMPARE<KEY_TYPE>, ALLOC,
00059 Bits::Avl_map_get_key<KEY_TYPE> >
00060 {
00061 private:
00062 typedef Pair<KEY_TYPE, DATA_TYPE> Local_item_type;
00063 typedef Bits::Base_avl_set<Local_item_type, COMPARE<KEY_TYPE>
00064 , ALLOC,
00065 Bits::Avl_map_get_key<KEY_TYPE> >
00066 Base_type;
00067
00068 public:
00069 typedef COMPARE<KEY_TYPE> Key_compare;
00070 typedef KEY_TYPE Key_type;
00071 typedef DATA_TYPE Data_type;
00072 typedef typename Base_type::Node Node;
00073 typedef typename Base_type::Node_allocator
00074 Node_allocator;
00075
00076 typedef typename Base_type::Iterator Iterator;
00077 typedef typename Base_type::Iterator iterator;
00078 typedef typename Base_type::Const_iterator Const_iterator;
00079 typedef typename Base_type::Const_iterator const_iterator;
00080 typedef typename Base_type::Rev_iterator Rev_iterator;
00081 typedef typename Base_type::Rev_iterator reverse_iterator;
00082 typedef typename Base_type::Const_rev_iterator Const_rev_iterator;
00083 typedef typename Base_type::Const_rev_iterator const_reverse_iterator;
00084
00085 Avl_map(Node_allocator const &alloc = Node_allocator())
00086 : Base_type(alloc)
00087 {}
00088
00089 cxx::Pair<Iterator, int> insert(Key_type const &key, Data_type const &data)
00090 { return Base_type::insert(Pair<Key_type, Data_type>(key, data)); }
00091
00092 Data_type const &operator [] (Key_type const &key) const
00093 { return this->find_node(key)->second; }
00094
00095 Data_type &operator [] (Key_type const &key)
00096 {
00097 Node n = this->find_node(key);
00098 if (n)
00099 return const_cast<Data_type&>(n->second);
00100 else
00101 return insert(key, Data_type()).first->second;
00102 }
00103 };
00104
00105 }
00106
00107 }

```

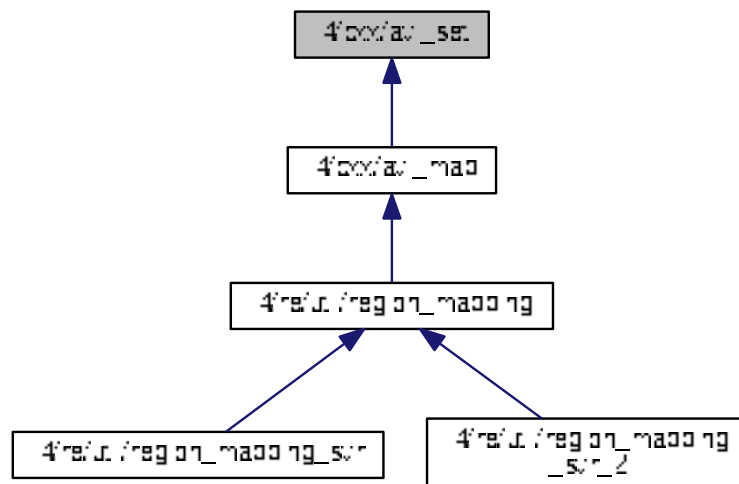
## 15.95 l4/cxx/avl\_set File Reference

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

- `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 >`  
*Internal: 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 compareable items.*

## Namespaces

- `cxx`  
*Our C++ library.*
- `cxx::Bits`  
*Internal helpers for the `cxx` package.*

### 15.95.1 Detailed Description

AVL set.

Definition in file `avl_set`.

## 15.96 avl\_set

```

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 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024
00025 #pragma once
00026
00027 #include <l4/cxx/std_alloc>
00028 #include <l4/cxx/std_ops>
00029 #include <l4/cxx/type_traits>
00030 #include <l4/cxx/avl_tree>
00031
00032 namespace cxx {
00033 namespace Bits {
00042 template< typename Node, typename Key, typename Node_op >
00043 class Avl_set_iter : public __Bst_iter_b<Node, Node_op>
00044 {
00045 private:
00047 typedef __Bst_iter_b<Node, Node_op> Base;
00048
00049 using Base::_n;
00050 using Base::_r;

```

```

00051 using Base::inc;
00052
00053 public:
00054 Avl_set_iter() {}
00055
00062 Avl_set_iter(Node const *t) : Base(t) {}
00063
00064 Avl_set_iter(Base const &o) : Base(o) {}
00065
00066 Avl_set_iter(Avl_set_iter<Node, typename Type_traits<Key>::Non_const_type, Node_op> const &o)
00067 : Base(o) {}
00068
00073 Key &operator * () const { return const_cast<Node*>(_n)->item; }
00078 Key *operator -> () const { return &const_cast<Node*>(_n)->item; }
00082 Avl_set_iter &operator ++ () { inc(); return *this; }
00086 Avl_set_iter &operator ++ (int)
00087 { Avl_set_iter tmp = *this; inc(); return tmp; }
00088
00089 };
00090
00092 template<typename KEY_TYPE>
00093 struct Avl_set_get_key
00094 {
00095 typedef KEY_TYPE Key_type;
00096 template<typename NODE>
00097 static Key_type const &key_of(NODE const *n)
00098 { return n->item; }
00099 };
00100
00101
00114 template< typename ITEM_TYPE, class COMPARE,
00115 template<typename A> class ALLOC,
00116 typename GET_KEY>
00117 class Base_avl_set
00118 {
00119 public:
00120 enum
00121 {
00122 E_noent = 2,
00123 E_exist = 17,
00124 E_nomem = 12,
00125 E_inval = 22
00126 };
00127 typedef ITEM_TYPE Item_type;
00128 typedef GET_KEY Get_key;
00132 typedef typename GET_KEY::Key_type Key_type;
00134 typedef typename Type_traits<Item_type>::Const_type Const_item_type;
00136 typedef COMPARE Item_compare;
00137
00138 private:
00140 class _Node : public Avl_tree_node
00141 {
00142 public:
00144 Item_type item;
00145
00146 _Node() : Avl_tree_node(), item() {}
00147
00148 _Node(Item_type const &item) : Avl_tree_node(), item(item) {}
00149 };
00150
00151 public:
00155 class Node
00156 {
00157 private:
00158 struct No_type;
00159 friend class Base_avl_set<ITEM_TYPE, COMPARE, ALLOC, GET_KEY>;
00160 _Node const *_n;
00161 explicit Node(_Node const *n) : _n(n) {}
00162
00163 public:
00165 Node() : _n(0) {}
00166
00168 Node &operator = (Node const &o) { _n = o._n; return *this; }
00169
00175 Item_type const &operator * () { return _n->item; }
00181 Item_type const *operator -> () { return &_n->item; }
00182
00187 bool valid() const { return _n; }
00188
00190 operator Item_type const * () { if (_n) return &_n->item; else return 0; }
00191 };
00192
00194 typedef ALLOC<_Node> Node_allocator;
00195
00196 private:
00197 typedef Avl_tree<_Node, GET_KEY, COMPARE>
 Tree;

```

```

00198 Tree _tree;
00200 Node_allocator _alloc;
00201
00202 Base_avl_set &operator = (Base_avl_set const &) = delete;
00203
00204 typedef typename Tree::Fwd_iter_ops Fwd;
00205 typedef typename Tree::Rev_iter_ops Rev;
00206
00207 public:
00208 typedef typename Type_traits<Item_type>::Param_type Item_param_type;
00209
00211 typedef Avl_set_iter<Node, Item_type, Fwd> Iterator;
00212 typedef Iterator iterator;
00214 typedef Avl_set_iter<Node, Const_item_type, Fwd> Const_iterator;
00215 typedef Const_iterator const_iterator;
00217 typedef Avl_set_iter<Node, Item_type, Rev> Rev_iterator;
00218 typedef Rev_iterator reverse_iterator;
00220 typedef Avl_set_iter<Node, Const_item_type, Rev> Const_rev_iterator;
00221 typedef Const_rev_iterator const_reverse_iterator;
00222
00229 explicit Base_avl_set(Node_allocator const &alloc = Node_allocator())
00230 : _tree(), _alloc(alloc)
00231 {}
00232
00233 ~Base_avl_set()
00234 {
00235 _tree.remove_all([this](_Node *n)
00236 {
00237 n->~_Node();
00238 _alloc.free(n);
00239 });
00240 }
00241
00249 inline Base_avl_set(Base_avl_set const &o);
00250
00267 cxx::Pair<Iterator, int> insert(Item_type const &item);
00268
00278 int remove(Key_type const &item)
00279 {
00280 _Node *n = _tree.remove(item);
00281 if ((long)n == -E_inval)
00282 return -E_inval;
00283
00284 if (n)
00285 {
00286 n->~_Node();
00287 _alloc.free(n);
00288 return 0;
00289 }
00290
00291 return -E_noent;
00292 }
00293
00298 int erase(Key_type const &item)
00299 { return remove(item); }
00300
00309 Node find_node(Key_type const &item) const
00310 { return Node(_tree.find_node(item)); }
00311
00320 Node lower_bound_node(Key_type const &key) const
00321 { return Node(_tree.lower_bound_node(key)); }
00322
00323 Node lower_bound_node(Key_type &&key) const
00324 { return Node(_tree.lower_bound_node(key)); }
00325
00330 Const_iterator begin() const { return _tree.begin(); }
00335 Const_iterator end() const { return _tree.end(); }
00336
00341 Iterator begin() { return _tree.begin(); }
00346 Iterator end() { return _tree.end(); }
00347
00352 Const_rev_iterator rbegin() const { return _tree.rbegin(); }
00357 Const_rev_iterator rend() const { return _tree.rend(); }
00358
00363 Rev_iterator rbegin() { return _tree.rbegin(); }
00368 Rev_iterator rend() { return _tree.rend(); }
00369
00370 Const_iterator find(Key_type const &item) const
00371 { return _tree.find(item); }
00372 };
00373
00374
00375 //-----
00376 /* Implementation of AVL Tree */
00377
00378 /* Create a copy */
00379 template< typename Item, class Compare, template<typename A> class Alloc, typename KEY_TYPE>

```

```

00380 Base_avl_set<Item,Compare,Alloc,KEY_TYPE>::Base_avl_set
 (Base_avl_set const &o)
00381 : _tree(), _alloc(o._alloc)
00382 {
00383 for (Const_iterator i = o.begin(); i != o.end(); ++i)
00384 insert(*i);
00385 }
00386
00387 /* Insert new _Node. */
00388 template< typename Item, class Compare, template< typename A > class Alloc, typename KEY_TYPE>
00389 Pair<typename Base_avl_set<Item,Compare,Alloc,KEY_TYPE>::Iterator
 , int>
00390 Base_avl_set<Item,Compare,Alloc,KEY_TYPE>::insert(Item
 const &item)
00391 {
00392 _Node *n = _alloc.alloc();
00393 if (!n)
00394 return cxx::pair(end(), -E_nomem);
00395
00396 new (n, Nothrow()) _Node(item);
00397 Pair<_Node *, bool> err = _tree.insert(n);
00398 if (!err.second)
00399 _alloc.free(n);
00400
00401 return cxx::pair(Iterator(typename Tree::Iterator(err.first, err.
 first)), err.second ? 0 : -E_exist);
00402 }
00403
00404 } // namespace Bits
00405
00417 template< typename ITEM_TYPE, class COMPARE = Lt_functor<ITEM_TYPE>,
00418 template<typename A> class ALLOC = New_allocator>
00419 class Avl_set :
00420 public Bits::Base_avl_set<ITEM_TYPE, COMPARE, ALLOC,
00421 Bits::Avl_set_get_key<ITEM_TYPE> >
00422 {
00423 private:
00424 typedef Bits::Base_avl_set<ITEM_TYPE, COMPARE, ALLOC,
00425 Bits::Avl_set_get_key<ITEM_TYPE> >
00426 Base;
00427 public:
00428 typedef typename Base::Node_allocator Node_allocator;
00429 Avl_set() = default;
00430 Avl_set(Node_allocator const &alloc)
00431 : Base(alloc)
00432 {}
00433 };
00434
00435 } // namespace cxx

```

## 15.97 l4/cxx/avl\_tree File Reference

AVL tree.

```

#include "std_ops"
#include "pair"
#include "bits/bst.h"
#include "bits/bst_iter.h"

```





*Node of an AVL tree.*

- [class cxx::Avl\\_tree< Node, Get\\_key, Compare >](#)

*A generic AVL tree.*

## Namespaces

- [CXX](#)

*Our C++ library.*

## 15.97.1 Detailed Description

AVL tree.

Definition in file [avl\\_tree](#).

## 15.98 avl\_tree

```

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 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024
00025 #pragma once
00026
00027 #include "std_ops"
00028 #include "pair"
00029
00030 #include "bits/bst.h"
00031 #include "bits/bst_iter.h"
00032
00033 namespace cxx {
00034
00038 class Avl_tree_node : public Bits::Bst_node
00039 {
00040 private:
00041 template< typename Node, typename Get_key, typename Compare >
00042 friend class Avl_tree;
00043
00045 typedef Bits::Direction Bal;
00047 typedef Bits::Direction Dir;
00048
00049 // We are a final BST node, hide interior.
00051 using Bits::Bst_node::next;
00052 using Bits::Bst_node::next_p;
00053 using Bits::Bst_node::rotate;
00056 Bal _balance;
00058
00059 protected:
00061 Avl_tree_node() {}
00062
00063 private:
00065 Avl_tree_node(Avl_tree_node const &o);
00066
00068 void operator = (Avl_tree_node const &o)

```

```

00069 {
00070 Bits::Bst_node::operator = (o);
00071 _balance = o._balance;
00072 }
00073
00075 explicit Avl_tree_node(bool) : Bits::Bst_node(true), _balance(
Dir::N) {}
00076
00078 static Bits::Bst_node *rotate2(Bst_node **t, Bal idir, Bal pre);
00079
00081 bool balanced() const { return _balance == Bal::N; }
00082
00084 Bal balance() const { return _balance; }
00085
00087 void balance(Bal b) { _balance = b; }
00088 };
00089
00090
00107 template< typename Node, typename Get_key,
00108 typename Compare = Lt_functor<typename Get_key::Key_type> >
00109 class Avl_tree : public Bits::Bst<Node, Get_key, Compare>
00110 {
00111 private:
00112 typedef Bits::Bst<Node, Get_key, Compare> Bst;
00113
00115 using Bst::_head;
00116
00118 using Bst::k;
00119
00121 typedef typename Avl_tree_node::Bal Bal;
00123 typedef typename Avl_tree_node::Bal Dir;
00124
00126 Avl_tree(Avl_tree const &o);
00127
00129 void operator = (Avl_tree const &o);
00130
00131 public:
00133 typedef typename Bst::Key_type Key_type;
00134 typedef typename Bst::Key_param_type Key_param_type;
00136
00137 // Grab iterator types from Bst
00139 typedef typename Bst::Iterator Iterator;
00142 typedef typename Bst::Const_iterator Const_iterator;
00144 typedef typename Bst::Rev_iterator Rev_iterator;
00146 typedef typename Bst::Const_rev_iterator
Const_rev_iterator;
00148
00156 Pair<Node *, bool> insert(Node *new_node);
00157
00164 Node *remove(Key_param_type key);
00168 Node *erase(Key_param_type key) { return remove(key); }
00169
00171 Avl_tree() : Bst() {}
00173 ~Avl_tree() noexcept
00174 {
00175 this->remove_all([](Node *){});
00176 }
00177
00178 #ifdef __DEBUG_L4_AVL
00179 bool rec_dump(Avl_tree_node *n, int depth, int *dp, bool print, char pfx);
00180 bool rec_dump(bool print)
00181 {
00182 int dp=0;
00183 return rec_dump(static_cast<Avl_tree_node *>(_head), 0, &dp, print, '+');
00184 }
00185 #endif
00186 };
00187
00188
00189 //-----
00190 /* IMPLEMENTATION: Bits::__Bst_iter_b */
00191
00192
00193 inline
00194 Bits::Bst_node *
00195 Avl_tree_node::rotate2(Bst_node **t, Bal idir, Bal pre)
00196 {
00197 typedef Bits::Bst_node N;
00198 typedef Avl_tree_node A;
00199 N *tmp[2] = { *t, N::next(*t, idir) };
00200 *t = N::next(tmp[1], !idir);
00201 A *n = static_cast<A*>(*t);
00202
00203 N::next(tmp[0], idir, N::next(n, !idir));
00204 N::next(tmp[1], !idir, N::next(n, idir));
00205 N::next(n, !idir, tmp[0]);
00206 N::next(n, idir, tmp[1]);

```

```

00207
00208 n->balance(Bal::N);
00209
00210 if (pre == Bal::N)
00211 {
00212 static_cast<A*>(tmp[0])->balance(Bal::N);
00213 static_cast<A*>(tmp[1])->balance(Bal::N);
00214 return 0;
00215 }
00216
00217 static_cast<A*>(tmp[pre != idir])->balance(!pre);
00218 static_cast<A*>(tmp[pre == idir])->balance(Bal::N);
00219
00220 return N::next(tmp[pre == idir], !pre);
00221 }
00222
00223 //-----
00224 /* Implementation of AVL Tree */
00225
00226 /* Insert new _Node. */
00227 template< typename Node, typename Get_key, class Compare>
00228 Pair<Node *, bool>
00229 Avl_tree<Node, Get_key, Compare>::insert(Node *new_node)
00230 {
00231 typedef Avl_tree_node A;
00232 typedef Bits::Bst_node N;
00233 N **t = &_head; /* search variable */
00234 N **s = &_head; /* node where rebalancing may occur */
00235 Key_param_type new_key = Get_key::key_of(new_node);
00236
00237 // search insertion point
00238 for (N *p; (p = *t);)
00239 {
00240 Dir b = this->dir(new_key, p);
00241 if (b == Dir::N)
00242 return pair(static_cast<Node*>(p), false);
00243
00244 if (!static_cast<A const *>(p)->balanced())
00245 s = t;
00246
00247 t = N::next_p(p, b);
00248 }
00249
00250 *static_cast<A*>(new_node) = A(true);
00251 *t = new_node;
00252
00253 N *n = *s;
00254 A *a = static_cast<A*>(n);
00255 if (!a->balanced())
00256 {
00257 A::Bal b(this->greater(new_key, n));
00258 if (a->balance() != b)
00259 {
00260 // ok we got in balance the shorter subtree go higher
00261 a->balance(Bal::N);
00262 // propagate the new balance down to the new node
00263 n = N::next(n, b);
00264 }
00265 else if (b == Bal(this->greater(new_key, N::next(n, b))))
00266 {
00267 // left-left or right-right case -> single rotation
00268 A::rotate(s, b);
00269 a->balance(Bal::N);
00270 static_cast<A*>(*s)->balance(Bal::N);
00271 n = N::next(*s, b);
00272 }
00273 else
00274 {
00275 // need a double rotation
00276 n = N::next(N::next(n, b), !b);
00277 n = A::rotate2(s, b, n == new_node ? Bal::N : Bal(this->greater(new_key, n)));
00278 }
00279 }
00280
00281 for (A::Bal b; n && n != new_node; static_cast<A*>(n)->balance(b), n = N::next(n, b))
00282 b = Bal(this->greater(new_key, n));
00283
00284 return pair(new_node, true);
00285 }
00286
00287 /* remove an element */
00288 template< typename Node, typename Get_key, class Compare>
00289 inline
00290 Node *Avl_tree<Node, Get_key, Compare>::remove(Key_param_type key)
00291 {
00292 typedef Avl_tree_node A;

```

```

00294 typedef Bits::Bst_node N;
00295 N **q = &_head; /* search variable */
00296 N **s = &_head; /* last ('deepest') node on the search path to q
00297 * with balance 0, at this place the rebalancing
00298 * stops in any case */
00299 N **t = 0;
00300 Dir dir;
00301
00302 // find target node and rebalancing entry
00303 for (N *n; (n = *q); q = N::next_p(n, dir))
00304 {
00305 dir = Dir(this->greater(key, n));
00306 if (dir == Dir::L && !this->greater(k(n), key))
00307 /* found node */
00308 t = q;
00309
00310 if (!N::next(n, dir))
00311 break;
00312
00313 A const *a = static_cast<A const *>(n);
00314 if (a->balanced() || (a->balance() == !dir && N::next<A>(n, !dir)->balanced()))
00315 s = q;
00316 }
00317
00318 // nothing found
00319 if (!t)
00320 return 0;
00321
00322 A *i = static_cast<A*>(*t);
00323
00324 for (N *n; (n = *s); s = N::next_p(n, dir))
00325 {
00326 dir = Dir(this->greater(key, n));
00327
00328 if (!N::next(n, dir))
00329 break;
00330
00331 A *a = static_cast<A*>(n);
00332 // got one out of balance
00333 if (a->balanced())
00334 a->balance(!dir);
00335 else if (a->balance() == dir)
00336 a->balance(Bal::N);
00337 else
00338 {
00339 // we need rotations to get in balance
00340 Bal b = N::next<A>(n, !dir)->balance();
00341 if (b == dir)
00342 A::rotate2(s, !dir, N::next<A>(N::next(n, !dir), dir)->balance());
00343 else
00344 {
00345 A::rotate(s, !dir);
00346 if (b != Bal::N)
00347 {
00348 a->balance(Bal::N);
00349 static_cast<A*>(*s)->balance(Bal::N);
00350 }
00351 else
00352 {
00353 a->balance(!dir);
00354 static_cast<A*>(*s)->balance(dir);
00355 }
00356 }
00357 if (n == i)
00358 t = N::next_p(*s, dir);
00359 }
00360 }
00361
00362 A *n = static_cast<A*>(*q);
00363 *t = n;
00364 *q = N::next(n, !dir);
00365 *n = *l;
00366
00367 return static_cast<Node*>(i);
00368 }
00369
00370 #ifdef __DEBUG_L4_AVL
00371 template< typename Node, typename Get_key, class Compare>
00372 bool Avl_tree<Node, Get_key, Compare>::rec_dump(
00373 Avl_tree_node *n, int depth, int *dp, bool print, char pfx)
00374 {
00375 typedef Avl_tree_node A;
00376
00377 if (!n)
00378 return true;
00379
00380 int dpx[2] = {depth, depth};

```



## Namespaces

- [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.*

### 15.99.1 Detailed Description

Basic IO stream.

Definition in file [basic\\_ostream](#).

## 15.100 basic\_ostream

```

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 * 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 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
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 #pragma once
00024
00025 namespace L4 {
00026
00033 class IOModifier
00034 {
00035 public:
00036 IOModifier(int x) : mod(x) {}
00037 bool operator == (IOModifier o) { return mod == o.mod; }
00038 bool operator != (IOModifier o) { return mod != o.mod; }
00039 int mod;
00040 };
00041
00046 class IOBackend
00047 {
00048 public:
00049 typedef int Mode;
00050
00051 protected:
00052 friend class BasicOStream;
00053
00054 IOBackend()
00055 : int_mode(10)
00056 {}
00057
00058 virtual ~IOBackend() {}
00059
00060 virtual void write(char const *str, unsigned len) = 0;
00061
00062 private:

```

```

00063 void write(IOModifier m);
00064 void write(long long int c, int len);
00065 void write(long long unsigned c, int len);
00066 void write(long long unsigned c, unsigned char base = 10,
00067 unsigned char len = 0, char pad = ' ');
00068
00069 Mode mode() const
00070 { return int_mode; }
00071
00072 void mode(Mode m)
00073 { int_mode = m; }
00074
00075 int int_mode;
00076 };
00077
00082 class BasicOStream
00083 {
00084 public:
00085 BasicOStream(IOBackend *b)
00086 : iob(b)
00087 {}
00088
00089 void write(char const *str, unsigned len)
00090 { if(iob) iob->write(str, len); }
00091
00092 void write(long long int c, int len)
00093 { if(iob) iob->write(c, len); }
00094
00095 void write(long long unsigned c, unsigned char base = 10,
00096 unsigned char len = 0, char pad = ' ')
00097 { if(iob) iob->write(c, base, len, pad); }
00098
00099 void write(long long unsigned c, int len)
00100 { if(iob) iob->write(c, len); }
00101
00102 void write(IOModifier m)
00103 { if(iob) iob->write(m); }
00104
00105 IOBackend::Mode be_mode() const
00106 { if(iob) return iob->mode(); return 0; }
00107
00108 void be_mode(IOBackend::Mode m)
00109 { if(iob) iob->mode(m); }
00110
00111 private:
00112 IOBackend *iob;
00113 };
00114
00120 class IONumFmt
00121 {
00122 public:
00123 IONumFmt(unsigned long long n, unsigned char base = 10,
00124 unsigned char len = 0, char pad = ' ')
00125 : n(n), base(base), len(len), pad(pad)
00126 {}
00127
00128 BasicOStream &print(BasicOStream &o) const;
00129
00130 private:
00131 unsigned long long n;
00132 unsigned char base, len;
00133 char pad;
00134 };
00135
00136 inline IONumFmt n_hex(unsigned long long n) { return IONumFmt(n, 16); }
00137
00141 extern IOModifier const hex;
00142
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 }
00156
00157 // implementation
00158
00160 inline
00161 L4::BasicOStream &
00162 operator << (L4::BasicOStream &s, char const * const str)
00163 {

```



```

00164 if (!str)
00165 {
00166 s.write("(NULL)", 6);
00167 return s;
00168 }
00169
00170 unsigned l=0;
00171 for(; str[l]!=0; l++)
00172 ;
00173 s.write(str,l);
00174 return s;
00175 }
00176
00177 inline
00178 L4::BasicOStream &
00179 operator << (L4::BasicOStream &s, signed short u)
00180 {
00181 s.write((long long signed)u,-1);
00182 return s;
00183 }
00184
00185 inline
00186 L4::BasicOStream &
00187 operator << (L4::BasicOStream &s, signed u)
00188 {
00189 s.write((long long signed)u,-1);
00190 return s;
00191 }
00192
00193 inline
00194 L4::BasicOStream &
00195 operator << (L4::BasicOStream &s, signed long u)
00196 {
00197 s.write((long long signed)u,-1);
00198 return s;
00199 }
00200
00201 inline
00202 L4::BasicOStream &
00203 operator << (L4::BasicOStream &s, signed long long u)
00204 {
00205 s.write(u,-1);
00206 return s;
00207 }
00208
00209 inline
00210 L4::BasicOStream &
00211 operator << (L4::BasicOStream &s, unsigned short u)
00212 {
00213 s.write((long long unsigned)u,-1);
00214 return s;
00215 }
00216
00217 inline
00218 L4::BasicOStream &
00219 operator << (L4::BasicOStream &s, unsigned u)
00220 {
00221 s.write((long long unsigned)u,-1);
00222 return s;
00223 }
00224
00225 inline
00226 L4::BasicOStream &
00227 operator << (L4::BasicOStream &s, unsigned long u)
00228 {
00229 s.write((long long unsigned)u,-1);
00230 return s;
00231 }
00232
00233 inline
00234 L4::BasicOStream &
00235 operator << (L4::BasicOStream &s, unsigned long long u)
00236 {
00237 s.write(u,-1);
00238 return s;
00239 }
00240
00241 inline
00242 L4::BasicOStream &
00243 operator << (L4::BasicOStream &s, void const *u)
00244 {
00245 long unsigned x = (long unsigned)u;
00246 L4::IOBackend::Mode mode = s.be_mode();
00247 s.write(L4::hex);
00248 s.write((long long unsigned)x,-1);
00249 s.be_mode(mode);
00250 return s;

```

```

00251 }
00252
00253 inline
00254 L4::BasicOStream &
00255 operator << (L4::BasicOStream &s, L4::IOModifier m)
00256 {
00257 s.write(m);
00258 return s;
00259 }
00260
00261 inline
00262 L4::BasicOStream &
00263 operator << (L4::BasicOStream &s, char c)
00264 {
00265 s.write(&c, 1);
00266 return s;
00267 }
00268
00269
00270
00271 inline
00272 L4::BasicOStream &
00273 operator << (L4::BasicOStream &o, L4::IONumFmt const &n)
00274 { return n.print(o); }
00275

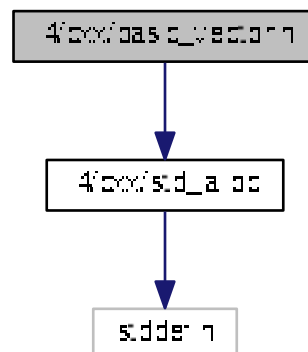
```

## 15.101 l4/cxx/basic\_vector.h File Reference

Basic vector.

```
#include <l4/cxx/std_alloc>
```

Include dependency graph for basic\_vector.h:



### Namespaces

- [cxx](#)

*Our C++ library.*

### 15.101.1 Detailed Description

Basic vector.

Definition in file [basic\\_vector.h](#).

## 15.102 basic\_vector.h

```

00001
00005 /*
00006 * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00007 * economic rights: Technische Universität Dresden (Germany)
00008 *
00009 * This file is part of TUD:OS and distributed under the terms of the
00010 * GNU General Public License 2.
00011 * Please see the COPYING-GPL-2 file for details.
00012 *
00013 * As a special exception, you may use this file as part of a free software
00014 * library without restriction. Specifically, if other files instantiate
00015 * templates or use macros or inline functions from this file, or you compile
00016 * this file and link it with other files to produce an executable, this
00017 * file does not by itself cause the resulting executable to be covered by
00018 * the GNU General Public License. This exception does not however
00019 * invalidate any other reasons why the executable file might be covered by
00020 * the GNU General Public License.
00021 */
00022 #pragma once
00023
00024 #include <l4/cxx/std_alloc>
00025
00026 namespace cxx {
00027
00028 template< typename T >
00029 class Basic_vector
00030 {
00031 public:
00032 Basic_vector(T *array, unsigned long capacity)
00033 : _array(array), _capacity(capacity)
00034 {
00035 for (unsigned long i = 0; i < capacity; ++i)
00036 new (&_array[i]) T();
00037 }
00038
00039 private:
00040 T *_array;
00041 unsigned long _capacity;
00042 };
00043
00044 };

```

## 15.103 l4/cxx/bits/bst.h File Reference

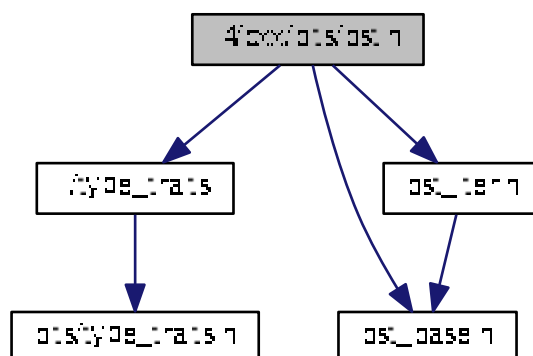
AVL tree.

```

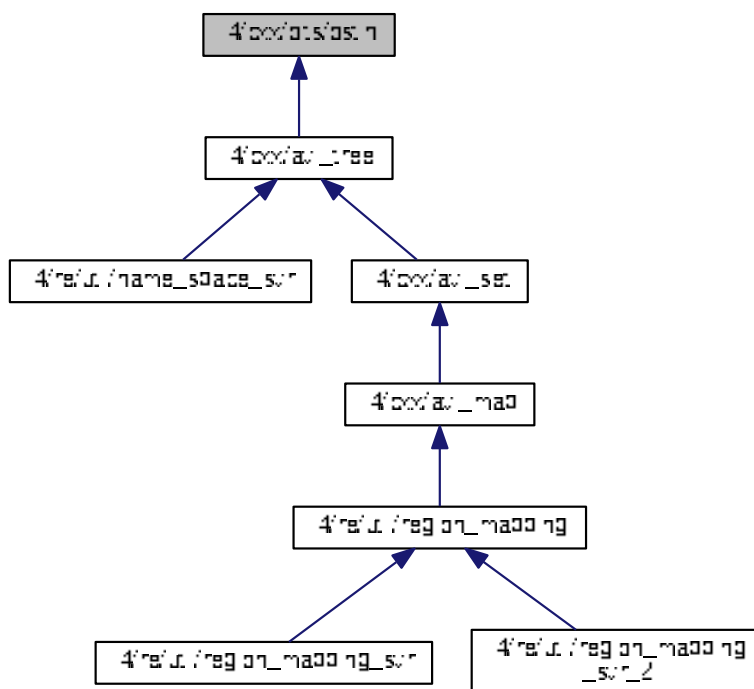
#include "../type_traits"
#include "bst_base.h"
#include "bst_iter.h"

```

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

- [cxx](#)  
*Our C++ library.*
- [cxx::Bits](#)  
*Internal helpers for the cxx package.*

### 15.103.1 Detailed Description

AVL tree.

Definition in file [bst.h](#).

## 15.104 bst.h

```
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 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024 #pragma once
00025
00026 #include "../type_traits"
00027 #include "bst_base.h"
00028 #include "bst_iter.h"
00029
00030 namespace cxx { namespace Bits {
00031
00039 template< typename Node, typename Get_key, typename Compare >
00040 class Bst
00041 {
00042 private:
00043 typedef Direction Dir;
00044 struct Fwd
00045 {
00046 static Node *child(Node const *n, Direction d)
00047 { return Bst_node::next<Node>(n, d); }
00048
00049 static bool cmp(Node const *l, Node const *r)
00050 { return Compare()(Get_key::key_of(l), Get_key::key_of(r)); }
00051 };
00052
00053 struct Rev
00054 {
00055 static Node *child(Node const *n, Direction d)
00056 { return Bst_node::next<Node>(n, !d); }
00057
00058 static bool cmp(Node const *l, Node const *r)
00059 { return Compare()(Get_key::key_of(r), Get_key::key_of(l)); }
00060 };
00061
00062 public:
00066 typedef typename Get_key::Key_type Key_type;
00068 typedef typename Type_traits<Key_type>::Param_type Key_param_type;
00069
00071 typedef Fwd Fwd_iter_ops;
00073 typedef Rev Rev_iter_ops;
00074
00076 }
```

```

00077 typedef __Bst_iter<Node, Node, Fwd> Iterator;
00080 typedef __Bst_iter<Node, Node const, Fwd> Const_iterator;
00082 typedef __Bst_iter<Node, Node, Rev> Rev_iterator;
00084 typedef __Bst_iter<Node, Node const, Rev> Const_rev_iterator;
00086
00087 protected:
00096
00098 Bst_node *_head;
00099
00101 Bst() : _head(0) {}
00102
00104 Node *head() const { return static_cast<Node*>(_head); }
00105
00107 static Key_type k(Bst_node const *n)
00108 { return Get_key::key_of(static_cast<Node const *>(n)); }
00109
00118 static Dir dir(Key_param_type l, Key_param_type r)
00119 {
00120 Compare cmp;
00121 Dir d(cmp(r, l));
00122 if (d == Direction::L && !cmp(l, r))
00123 return Direction::N;
00124 return d;
00125 }
00126
00135 static Dir dir(Key_param_type l, Bst_node const *r)
00136 { return dir(l, k(r)); }
00137
00139 static bool greater(Key_param_type l, Key_param_type r)
00140 { return Compare()(r, l); }
00141
00143 static bool greater(Key_param_type l, Bst_node const *r)
00144 { return greater(l, k(r)); }
00145
00147 static bool greater(Bst_node const *l, Bst_node const *r)
00148 { return greater(k(l), k(r)); }
00157 template<typename FUNC>
00158 static void remove_tree(Bst_node *head, FUNC &&callback)
00159 {
00160 if (Bst_node *n = Bst_node::next(head, Dir::L))
00161 remove_tree(n, callback);
00162
00163 if (Bst_node *n = Bst_node::next(head, Dir::R))
00164 remove_tree(n, callback);
00165
00166 callback(static_cast<Node *>(head));
00167 }
00168
00169 public:
00170
00179 Const_iterator begin() const { return Const_iterator(head()); }
00184 Const_iterator end() const { return Const_iterator(); }
00185
00190 Iterator begin() { return Iterator(head()); }
00195 Iterator end() { return Iterator(); }
00196
00201 Const_rev_iterator rbegin() const { return Const_rev_iterator(
head()); }
00206 Const_rev_iterator rend() const { return Const_rev_iterator(); }
00207
00212 Rev_iterator rbegin() { return Rev_iterator(head()); }
00217 Rev_iterator rend() { return Rev_iterator(); }
00231 Node *find_node(Key_param_type key) const;
00232
00239 Node *lower_bound_node(Key_param_type key) const;
00240
00247 Const_iterator find(Key_param_type key) const;
00248
00257 template<typename FUNC>
00258 void remove_all(FUNC &&callback)
00259 {
00260 if (!_head)
00261 return;
00262
00263 Bst_node *head = _head;
00264 _head = 0;
00265 remove_tree(head, cxx::forward<FUNC>(callback));
00266 }
00267
00268
00270 };
00271
00272 /* find an element */
00273 template< typename Node, typename Get_key, class Compare>
00274 inline
00275 Node *
00276 Bst<Node, Get_key, Compare>::find_node(

```

```

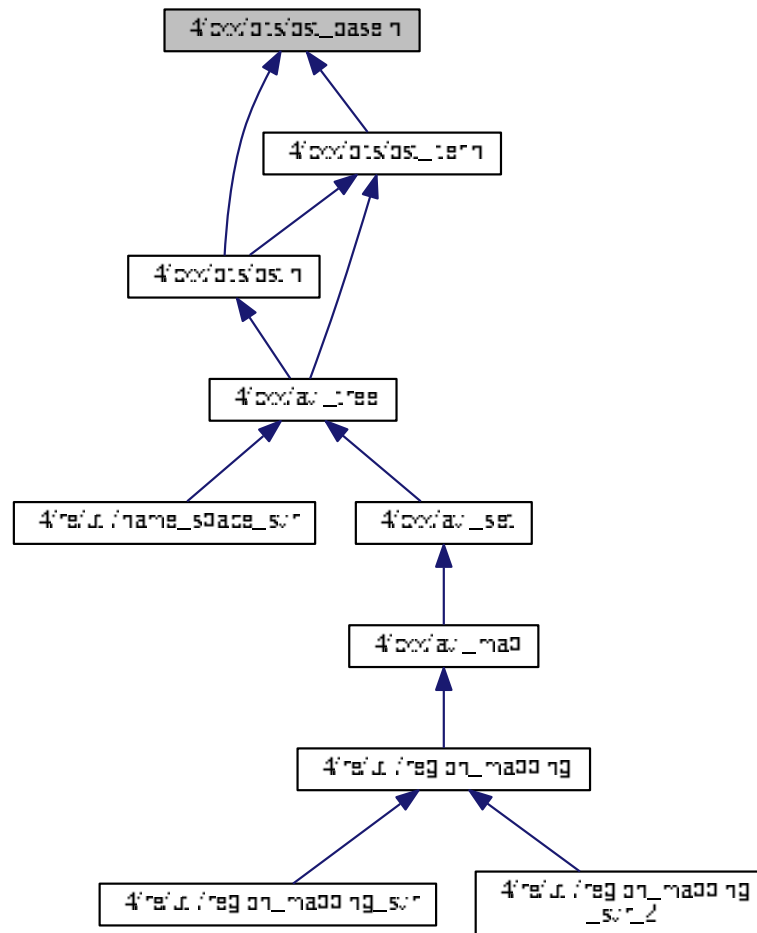
 Key_param_type key) const
00277 {
00278 Dir d;
00279
00280 for (Bst_node *q = _head; q; q = Bst_node::next(q, d))
00281 {
00282 d = dir(key, q);
00283 if (d == Dir::N)
00284 return static_cast<Node*>(q);
00285 }
00286 return 0;
00287 }
00288
00289 template< typename Node, typename Get_key, class Compare>
00290 inline
00291 Node *
00292 Bst<Node, Get_key, Compare>::lower_bound_node(
 Key_param_type key) const
00293 {
00294 Dir d;
00295 Bst_node *r = 0;
00296
00297 for (Bst_node *q = _head; q; q = Bst_node::next(q, d))
00298 {
00299 d = dir(key, q);
00300 if (d == Dir::L)
00301 r = q; // found a node greater than key
00302 else if (d == Dir::N)
00303 return static_cast<Node*>(q);
00304 }
00305 return static_cast<Node*>(r);
00306 }
00307
00308 /* find an element */
00309 template< typename Node, typename Get_key, class Compare>
00310 inline
00311 typename Bst<Node, Get_key, Compare>::Const_iterator
00312 Bst<Node, Get_key, Compare>::find(
 Key_param_type key) const
00313 {
00314 Bst_node *q = _head;
00315 Bst_node *r = q;
00316
00317 for (Dir d; q; q = Bst_node::next(q, d))
00318 {
00319 d = dir(key, q);
00320 if (d == Dir::N)
00321 return Iterator(static_cast<Node*>(q), static_cast<Node *>(r));
00322
00323 if (d != Dir::L && q == r)
00324 r = Bst_node::next(q, d);
00325 }
00326 return Iterator();
00327 }
00328
00329 }}

```

## 15.105 l4/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

- [cxx](#)  
*Our C++ library.*
- [cxx::Bits](#)  
*Internal helpers for the cxx package.*



### 15.105.1 Detailed Description

AVL tree.

Definition in file [bst\\_base.h](#).

## 15.106 bst\_base.h

```

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 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024
00025 #pragma once
00026
00027 /*
00028 * This file contains very basic bits for implementing binary search trees
00029 */
00030 namespace cxx {
00034 namespace Bits {
00035
00039 struct Direction
00040 {
00042 enum Direction_e
00043 {
00044 L = 0,
00045 R = 1,
00046 N = 2
00047 };
00048 unsigned char d;
00049
00051 Direction() {}
00052
00054 Direction(Direction_e d) : d(d) {}
00055
00057 explicit Direction(bool b) : d(Direction_e(b)) /*d(b ? R : L)*/ {}
00058
00063 Direction operator ! () const { return Direction(!d); }
00064
00066
00067 bool operator == (Direction_e o) const { return d == o; }
00068 bool operator != (Direction_e o) const { return d != o; }
00069 bool operator == (Direction o) const { return d == o.d; }
00070 bool operator != (Direction o) const { return d != o.d; }
00072 };
00073
00077 class Bst_node
00078 {
00079 // all BSTs are friends
00080 template< typename Node, typename Get_key, typename Compare >
00081 friend class Bst;
00082
00083 protected:
00092
00094 static Bst_node *next(Bst_node const *p, Direction d)
00095 { return p->_c[d.d]; }
00096
00098 static void next(Bst_node *p, Direction d, Bst_node *n)
00099 { p->_c[d.d] = n; }
00100
00102 static Bst_node **next_p(Bst_node *p, Direction d)
00103 { return &p->_c[d.d]; }
00104
00106 template< typename Node > static

```

```

00107 Node *next(Bst_node const *p, Direction d)
00108 { return static_cast<Node *>(p->_c[d.d]); }
00109
00111 static void rotate(Bst_node **t, Direction idir);
00114 private:
00115 Bst_node *_c[2];
00116
00117 protected:
00119 Bst_node() {}
00120
00122 explicit Bst_node(bool) { _c[0] = _c[1] = 0; }
00123 };
00124
00125 inline
00126 void
00127 Bst_node::rotate(Bst_node **t, Direction idir)
00128 {
00129 Bst_node *tmp = *t;
00130 *t = next(tmp, idir);
00131 next(tmp, idir, next(*t, !idir));
00132 next(*t, !idir, tmp);
00133 }
00134
00135 }}

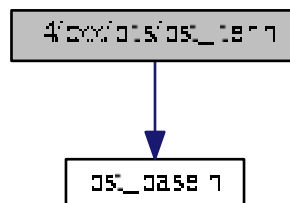
```

## 15.107 l4/cxx/bits/bst\_iter.h File Reference

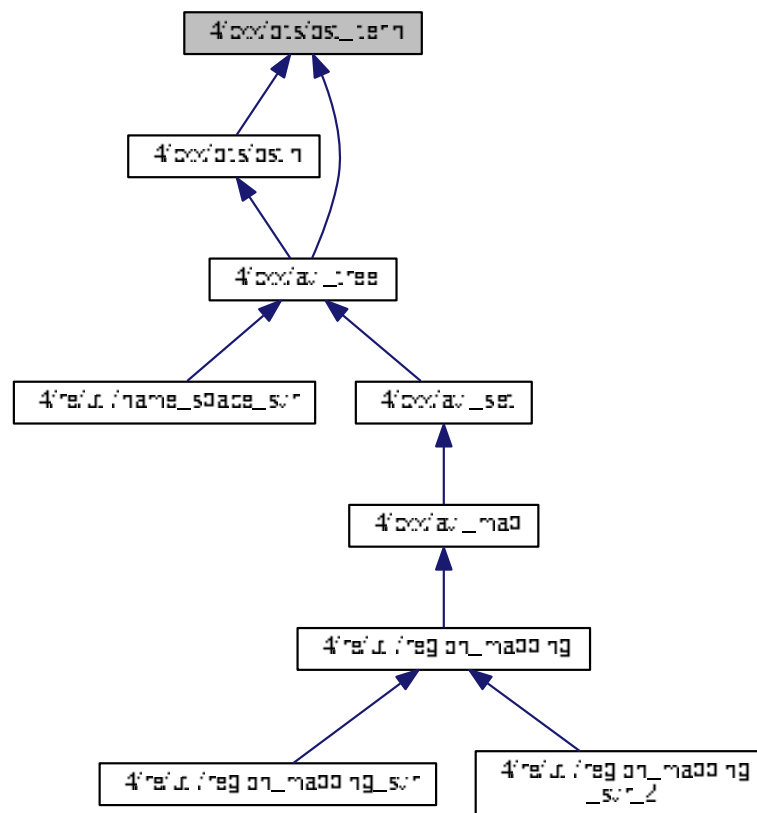
AVL tree.

```
#include "bst_base.h"
```

Include dependency graph for bst\_iter.h:



This graph shows which files directly or indirectly include this file:



## Namespaces

- [cxx](#)  
*Our C++ library.*
- [cxx::Bits](#)  
*Internal helpers for the cxx package.*

## 15.107.1 Detailed Description

AVL tree.

Definition in file [bst\\_iter.h](#).

## 15.108 bst\_iter.h

```

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 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024
00025 #pragma once
00026
00027 #include "bst_base.h"
00028
00029 namespace cxx { namespace Bits {
00030
00031 template< typename Node, typename Node_op >
00032 class __Bst_iter_b
00033 {
00034 protected:
00035 typedef Direction Dir;
00036 Node const *_n;
00037 Node const *_r;
00038
00039 __Bst_iter_b() : _n(0), _r(0) {}
00040
00041 __Bst_iter_b(Node const *t)
00042 : _n(t), _r(_n)
00043 { _downmost(); }
00044
00045 __Bst_iter_b(Node const *t, Node const *r)
00046 : _n(t), _r(r)
00047 {}
00048
00049 inline void _downmost();
00050
00051 inline void inc();
00052
00053 public:
00054 bool operator == (__Bst_iter_b const &o) const { return _n == o._n; }
00055 bool operator != (__Bst_iter_b const &o) const { return _n != o._n; }
00056 };
00057
00058 template< typename Node, typename Node_type, typename Node_op >
00059 class __Bst_iter : public __Bst_iter_b<Node, Node_op>
00060 {
00061 private:
00062 typedef __Bst_iter_b<Node, Node_op> Base;
00063
00064 using Base::_n;
00065 using Base::_r;
00066 using Base::inc;
00067
00068 public:
00069 __Bst_iter() {}
00070
00071 __Bst_iter(Node const *t) : Base(t) {}
00072 __Bst_iter(Node const *t, Node const *r) : Base(t, r) {}
00073
00074 // template<typename Key2>
00075 __Bst_iter(Base const &o) : Base(o) {}
00076
00077 Node_type &operator * () const { return *const_cast<Node *>(_n); }
00078 Node_type *operator -> () const { return const_cast<Node *>(_n); }
00079 __Bst_iter &operator ++ () { inc(); return *this; }
00080 __Bst_iter &operator ++ (int)
00081 { __Bst_iter tmp = *this; inc(); return tmp; }
00082 };
00083
00084 //-----
00085 /* IMPLEMENTATION: __Bst_iter_b */
00086
00087 template< typename Node, typename Node_op>

```

```

00136 inline
00137 void __Bst_iter_b<Node, Node_op>::_downmost()
00138 {
00139 while (_n)
00140 {
00141 Node *n = Node_op::child(_n, Dir::L);
00142 if (n)
00143 _n = n;
00144 else
00145 return;
00146 }
00147 }
00148
00149 template< typename Node, typename Node_op>
00150 void __Bst_iter_b<Node, Node_op>::inc()
00151 {
00152 if (!_n)
00153 return;
00154
00155 if (_n == _r)
00156 {
00157 _r = _n = Node_op::child(_r, Dir::R);
00158 _downmost();
00159 return;
00160 }
00161
00162 if (Node_op::child(_n, Dir::R))
00163 {
00164 _n = Node_op::child(_n, Dir::R);
00165 _downmost();
00166 return;
00167 }
00168
00169 Node const *q = _r;
00170 Node const *p = _r;
00171 while (1)
00172 {
00173 if (Node_op::cmp(_n, q))
00174 {
00175 p = q;
00176 q = Node_op::child(q, Dir::L);
00177 }
00178 else if (_n == q || Node_op::child(q, Dir::R) == _n)
00179 {
00180 _n = p;
00181 return;
00182 }
00183 else
00184 q = Node_op::child(q, Dir::R);
00185 }
00186 }
00187
00188 }}

```

## 15.109 l4/cxx/exceptions File Reference

Base exceptions.

```

#include <l4/cxx/l4types.h>
#include <l4/cxx/basic_ostream>
#include <l4/sys/err.h>
#include <l4/sys/capability>
#include <l4/util/backtrace.h>

```



- 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

- [L4](#)  
*L4 low-level kernel interface.*

## Macros

- `#define L4\_CXX\_EXCEPTION\_BACKTRACE 20`  
*Number of instruction pointers in backtrace.*

### 15.109.1 Detailed Description

Base exceptions.

Definition in file [exceptions](#).

## 15.110 exceptions

```

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 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025
00026 #pragma once
00027
00028 #include <l4/cxx/l4types.h>
00029 #include <l4/cxx/basic_ostream>
00030 #include <l4/sys/err.h>
00031 #include <l4/sys/capability>
00032

```

```

00033
00039
00040 #ifndef L4_CXX_NO_EXCEPTION_BACKTRACE
00041 # define L4_CXX_EXCEPTION_BACKTRACE 20
00042 #endif
00043
00044 #if defined(L4_CXX_EXCEPTION_BACKTRACE)
00045 #include <l4/util/backtrace.h>
00046 #endif
00047
00049 namespace L4
00050 {
00063 class Exception_tracer
00064 {
00065 #if defined(L4_CXX_EXCEPTION_BACKTRACE)
00066 private:
00067 void *_pc_array[L4_CXX_EXCEPTION_BACKTRACE];
00068 int _frame_cnt;
00069
00070 protected:
00074 #if defined(__PIC__)
00075 Exception_tracer() throw() : _frame_cnt(0) {}
00076 #else
00077 Exception_tracer() throw() : _frame_cnt(l4util_backtrace(_pc_array,
00078 L4_CXX_EXCEPTION_BACKTRACE)) {}
00078 #endif
00079
00080 public:
00084 void const *const *pc_array() const throw() { return _pc_array; }
00088 int frame_count() const throw() { return _frame_cnt; }
00089 #else
00090 protected:
00094 Exception_tracer() throw() {}
00095
00096 public:
00100 void const *const *pc_array() const throw() { return 0; }
00104 int frame_count() const throw() { return 0; }
00105 #endif
00106 };
00107
00116 class Base_exception : public Exception_tracer
00117 {
00118 protected:
00120 Base_exception() throw() {}
00121
00122 public:
00126 virtual char const *str() const throw () = 0;
00127
00129 virtual ~Base_exception() throw () {}
00130 };
00131
00139 class Runtime_error : public Base_exception
00140 {
00141 private:
00142 long _errno;
00143 char _extra[80];
00144
00145 public:
00152 explicit Runtime_error(long err_no, char const *extra = 0) throw ()
00153 : _errno(err_no)
00154 {
00155 if (!extra)
00156 _extra[0] = 0;
00157 else
00158 {
00159 unsigned i = 0;
00160 for (; i < sizeof(_extra) && extra[i]; ++i)
00161 _extra[i] = extra[i];
00162 _extra[i < sizeof(_extra) ? i : sizeof(_extra) - 1] = 0;
00163 }
00164 }
00165 char const *str() const throw ()
00166 { return l4sys_errtostr(_errno); }
00167
00173 char const *extra_str() const { return _extra; }
00174 ~Runtime_error() throw () {}
00175
00181 long err_no() const throw() { return _errno; }
00182 };
00183
00188 class Out_of_memory : public Runtime_error
00189 {
00190 public:
00192 explicit Out_of_memory(char const *extra = "") throw()
00193 : Runtime_error(-L4_ENOMEM, extra) {}
00195 ~Out_of_memory() throw() {}
00196 };

```



```

00197
00198
00203 class Element_already_exists : public Runtime_error
00204 {
00205 public:
00206 explicit Element_already_exists(char const *e = "") throw()
00207 : Runtime_error(-L4_EEXIST, e) {}
00208 ~Element_already_exists() throw() {}
00209 };
00210
00219 class Unknown_error : public Base_exception
00220 {
00221 public:
00222 Unknown_error() throw() {}
00223 char const *str() const throw() { return "unknown error"; }
00224 ~Unknown_error() throw() {}
00225 };
00226
00227
00232 class Element_not_found : public Runtime_error
00233 {
00234 public:
00235 explicit Element_not_found(char const *e = "") throw()
00236 : Runtime_error(-L4_ENOENT, e) {}
00237 };
00238
00246 class Invalid_capability : public Base_exception
00247 {
00248 private:
00249 Cap<void> const _o;
00250
00251 public:
00256 explicit Invalid_capability(Cap<void> const &o) throw() : _o(o) {}
00257 template< typename T>
00258 explicit Invalid_capability(Cap<T> const &o) throw() : _o(o.
00259 cap()) {}
00259 char const *str() const throw() { return "invalid object"; }
00260
00265 Cap<void> const &cap() const throw() { return _o; }
00266 ~Invalid_capability() throw() {}
00267 };
00268
00275 class Com_error : public Runtime_error
00276 {
00277 public:
00282 explicit Com_error(long err) throw() : Runtime_error(err) {}
00283
00284 ~Com_error() throw() {}
00285 };
00286
00290 class Bounds_error : public Runtime_error
00291 {
00292 public:
00293 explicit Bounds_error(char const *e = "") throw()
00294 : Runtime_error(-L4_ERANGE, e) {}
00295 ~Bounds_error() throw() {}
00296 };
00298 };
00299
00300 inline
00301 L4::BasicOStream &
00302 operator << (L4::BasicOStream &o, L4::Base_exception const &e)
00303 {
00304 o << "Exception: " << e.str() << ", backtrack ...\\n";
00305 for (int i = 0; i < e.frame_count(); ++i)
00306 o << L4::n_hex(l4_addr_t(e.pc_array()[i])) << '\\n';
00307
00308 return o;
00309 }
00310
00311 inline
00312 L4::BasicOStream &
00313 operator << (L4::BasicOStream &o, L4::Runtime_error const &e)
00314 {
00315 o << "Exception: " << e.str() << ": ";
00316 if (e.extra_str())
00317 o << e.extra_str() << ": ";
00318 o << "backtrace ...\\n";
00319 for (int i = 0; i < e.frame_count(); ++i)
00320 o << L4::n_hex(l4_addr_t(e.pc_array()[i])) << '\\n';
00321
00322 return o;
00323 }
00324

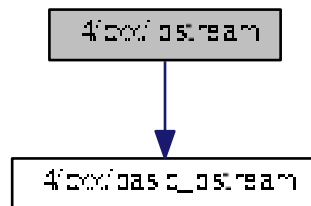
```

## 15.111 l4/cxx/iostream File Reference

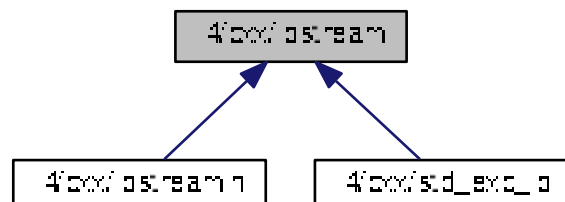
IO Stream.

```
#include <l4/cxx/basic_ostream>
```

Include dependency graph for iostream:



This graph shows which files directly or indirectly include this file:



### Namespaces

- [L4](#)  
*L4 low-level kernel interface.*

### Variables

- BasicOStream [L4::cout](#)  
*Standard output stream.*
- BasicOStream [L4::cerr](#)  
*Standard error stream.*

### 15.111.1 Detailed Description

IO Stream.

Definition in file [iostream](#).

## 15.112 iostream

```

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 * 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 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
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_Iostream_H__
00025 #define L4_Iostream_H__
00026
00027 #include <l4/cxx/basic_ostream>
00028
00029 namespace L4 {
00030
00034 extern BasicOStream cout;
00035
00039 extern BasicOStream cerr;
00040
00041
00042 extern void iostream_init();
00043
00044 static void __attribute__((used, constructor)) __iostream_init()
00045 { iostream_init(); }
00046
00047 };
00048
00049 #endif /* L4_Iostream_H__ */

```

## 15.113 l4/cxx/ipc\_helper File Reference

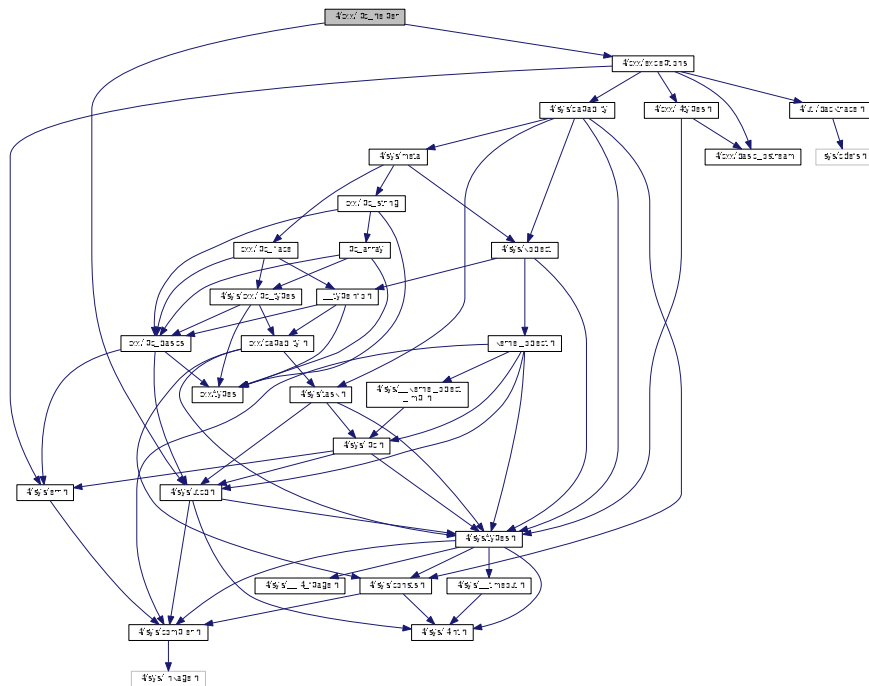
IPC helper.

```

#include <l4/cxx/exceptions>
#include <l4/sys/utcb.h>

```

Include dependency graph for `ipc_helper`:



## Namespaces

- [L4](#)

[L4](#) low-level kernel interface.

## 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.

### 15.113.1 Detailed Description

IPC helper.

Definition in file [ipc\\_helper](#).

## 15.114 ipc\_helper

```

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 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024
00025 #pragma once
00026
00027 #include <l4/cxx/exceptions>
00028 #include <l4/sys/utcb.h>
00029
00030 namespace L4
00031 {
00032
00033 #ifdef __EXCEPTIONS
00034 inline void
00044 throw_ipc_exception(L4::Cap<void> const &o,
00045 l4_msgtag_t const &err,
00046 l4_utcb_t *utcb)
00047 {
00048 (void)o;
00049 if (err.has_error())
00050 throw (L4::Com_error(l4_error_u(err, utcb)));
00051 }
00060 inline void
00061 throw_ipc_exception(void const *o, l4_msgtag_t const &err,
00062 l4_utcb_t *utcb)
00063 { throw_ipc_exception(L4::Cap<void>(o), err, utcb); }
00064 #endif
00065
00066 };

```

## 15.115 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

- [L4](#)  
*L4 low-level kernel interface.*
- [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.*

### 15.115.1 Detailed Description

IPC stream.

Definition in file [ipc\\_stream](#).

### 15.115.2 Function Documentation

#### 15.115.2.1 `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

|   |                                     |
|---|-------------------------------------|
| s | The stream to insert the element v. |
| v | The element to insert.              |

#### Returns

The stream s.

Definition at line 1193 of file [ipc\\_stream](#).

References [L4::Ipc::Ostream::put\(\)](#).

Referenced by [operator>>\(\)](#).

Here is the call graph for this function:





Here is the caller graph for this function:



### 15.115.2.2 operator<<() [2/4]

```

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 <a href="#">L4</a> 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 [1228](#) of file [ipc\\_stream](#).

References [L4::Ipc::Ostream::tag\(\)](#).

Here is the call graph for this function:



**15.115.2.3** `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**

|   |                                                            |
|---|------------------------------------------------------------|
| s | The stream to insert the array v.                          |
| v | The array to insert (see <code>lpc::Buf_cp_out()</code> ). |

**Returns**

the stream s.

Definition at line 1244 of file `ipc_stream`.

References `L4::lpc::Ostream::put()`.

Here is the call graph for this function:

**15.115.2.4** `operator<<()` [4/4]

```
L4::Ipc::Ostream& operator<< (
 L4::Ipc::Ostream & s,
 char const * v) [inline]
```

Insert a zero terminated character string into the stream s.

**Parameters**

|   |                                    |
|---|------------------------------------|
| s | The stream to insert the string v. |
| v | The string to insert.              |

**Returns**

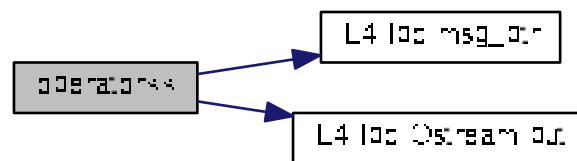
The stream `s`.

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 1265 of file `ipc_stream`.

References [L4\\_DEPRECATED](#), [L4::Ipc::msg\\_ptr\(\)](#), and [L4::Ipc::Ostream::put\(\)](#).

Here is the call graph for this function:

**15.115.2.5 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**

|                  |                |                             |
|------------------|----------------|-----------------------------|
|                  | <code>s</code> | The stream to extract from. |
| <code>out</code> | <code>v</code> | Extracted value.            |

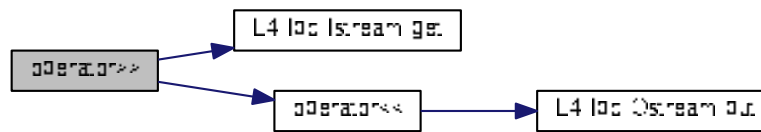
**Returns**

The stream `s`.

Definition at line 1040 of file `ipc_stream`.

References [L4::Ipc::Istream::get\(\)](#), and [operator<<\(\)](#).

Here is the call graph for this function:



#### 15.115.2.6 operator>>() [2/6]

```

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 [1074](#) of file [ipc\\_stream](#).

References [L4::ipc::Istream::tag\(\)](#).

Here is the call graph for this function:



## 15.115.2.7 operator&gt;&gt;() [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

|     |   |                                                |
|-----|---|------------------------------------------------|
|     | s | The stream to extract from.                    |
| out | v | Pointer to the extracted array (ipc_buf_in()). |

## 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

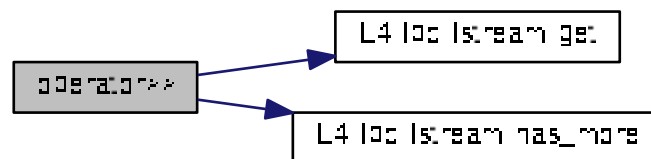
If array does not fit into transmitted words size will be set to zero. Client has to implement check against zero.

See `Ipc::Buf_in`, `Ipc::Buf_cp_in`, and `Ipc::Buf_cp_out`.

Definition at line 1099 of file `ipc_stream`.

References `L4::Ipc::Istream::get()`, and `L4::Ipc::Istream::has_more()`.

Here is the call graph for this function:



## 15.115.2.8 operator&gt;&gt;() [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**

|     |          |                                   |
|-----|----------|-----------------------------------|
|     | <i>s</i> | The stream to extract from.       |
| out | <i>v</i> | 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 1126 of file `ipc_stream`.

References `L4::Ipc::Istream::get()`.

Here is the call graph for this function:

**15.115.2.9 operator>>() [5/6]**

```

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**

|     |          |                                                                             |
|-----|----------|-----------------------------------------------------------------------------|
|     | <i>s</i> | The stream to extract from.                                                 |
| out | <i>v</i> | Buffer description to copy the array to ( <code>Ipc::Buf_cp_out()</code> ). |

**Returns**

The stream *s*.

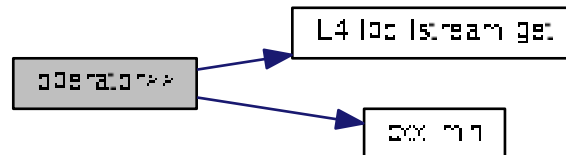
This operator does a copy out of the data into the given buffer.

See `lpc::Buf_in`, `lpc::Buf_cp_in`, and `lpc::Buf_cp_out`.

Definition at line 1147 of file `ipc_stream`.

References `L4::lpc::Istream::get()`, and `cxx::min()`.

Here is the call graph for this function:



#### 15.115.2.10 `operator>>()` [6/6]

```

template<typename T >
L4::lpc::Istream& operator>> (
 L4::lpc::Istream & s,
 L4::lpc::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 ( <code>lpc::Str_cp_out()</code> ). |

##### Returns

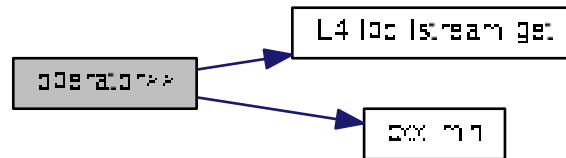
the stream *s*.

This operator does a copy out of the data into the given buffer.

Definition at line 1168 of file `ipc_stream`.

References `L4::lpc::Istream::get()`, and `cxx::min()`.

Here is the call graph for this function:



## 15.116 ipc\_stream

```

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 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025
00026 #pragma once
00027
00028 #include <l4/sys/ipc.h>
00029 #include <l4/sys/capability>
00030 #include <l4/sys/cxx/ipc_types>
00031 #include <l4/sys/cxx/ipc_varg>
00032 #include <l4/cxx/type_traits>
00033 #include <l4/cxx/minmax>
00034
00035 #define L4_CXX_IPC_BACKWARD_COMPAT
00036
00037 namespace L4 {
00038 namespace Ipc {
00039
00040 class Ostream;
00041 class Istream;
00042
00043 namespace Internal {
00044 template< typename T >
00045 class Buf_cp_out
00046 {
00047 public:
00048 Buf_cp_out(T const *v, unsigned long size) : _v(v), _s(size) {}
00049
00050 unsigned long size() const { return _s; }
00051
00052 T const *buf() const { return _v; }
00053
00054 private:
00055 friend class Ostream;
00056 T const *_v;
00057 unsigned long _s;
00058 };
00059
00060 template< typename T >

```



```

00114 Internal::Buf_cp_out<T> buf_cp_out(T const *v, unsigned long size)
00115 { return Internal::Buf_cp_out<T>(v, size); }
00116
00117
00118 namespace Internal {
00131 template< typename T >
00132 class Buf_cp_in
00133 {
00134 public:
00143 Buf_cp_in(T *v, unsigned long &size) : _v(v), _s(&size) {}
00144
00145 unsigned long &size() const { return *_s; }
00146 T *buf() const { return _v; }
00147 private:
00148 friend class Istream;
00149 T *_v;
00150 unsigned long *_s;
00151 };
00152 }
00153
00171 template< typename T >
00172 Internal::Buf_cp_in<T> buf_cp_in(T *v, unsigned long &size)
00173 { return Internal::Buf_cp_in<T>(v, size); }
00174
00190 template< typename T >
00191 class Str_cp_in
00192 {
00193 public:
00202 Str_cp_in(T *v, unsigned long &size) : _v(v), _s(&size) {}
00203
00204 unsigned long &size() const { return *_s; }
00205 T *buf() const { return _v; }
00206 private:
00207 friend class Istream;
00208 T *_v;
00209 unsigned long *_s;
00210 };
00211
00224 template< typename T >
00225 Str_cp_in<T> str_cp_in(T *v, unsigned long &size)
00226 { return Str_cp_in<T>(v, size); }
00227
00240 template< typename T >
00241 class Msg_ptr
00242 {
00243 private:
00244 T **_p;
00245 public:
00252 explicit Msg_ptr(T *p) : _p(&p) {}
00253 void set(T *p) const { *_p = p; }
00254 };
00255
00263 template< typename T >
00264 Msg_ptr<T> msg_ptr(T *p)
00265 { return Msg_ptr<T>(p); }
00266
00267
00268 namespace Internal {
00282 template< typename T >
00283 class Buf_in
00284 {
00285 public:
00292 Buf_in(T *v, unsigned long &size) : _v(&v), _s(&size) {}
00293
00294 void set_size(unsigned long s) const { *_s = s; }
00295 T *buf() const { return _v; }
00296 private:
00297 friend class Istream;
00298 T **_v;
00299 unsigned long *_s;
00300 };
00301 }
00302
00320 template< typename T >
00321 Internal::Buf_in<T> buf_in(T *v, unsigned long &size)
00322 { return Internal::Buf_in<T>(v, size); }
00323
00324 namespace Utc_stream_check
00325 {
00326 static bool check_utcb_data_offset(unsigned sz)
00327 { return sz > sizeof(l4_umword_t) * L4_UTCB_GENERIC_DATA_SIZE; }
00328 }
00329
00330
00345 class Istream
00346 {
00347 public:

```

```

00359 Istream(l4_utcb_t *utcb)
00360 : _tag(), _utcb(utcb),
00361 _current_msg(reinterpret_cast<char*>(l4_utcb_mr_u(utcb)->mr)),
00362 _pos(0), _current_buf(0)
00363 {}
00364
00369 void reset()
00370 {
00371 _pos = 0;
00372 _current_buf = 0;
00373 _current_msg = reinterpret_cast<char*>(l4_utcb_mr_u(_utcb)->mr);
00374 }
00375
00379 template< typename T >
00380 bool has_more(unsigned long count = 1)
00381 {
00382 auto const max_bytes = L4_UTCB_GENERIC_DATA_SIZE * sizeof(
l4_umword_t);
00383 unsigned apos = cxx::Type_traits<T>::align(_pos);
00384 return (count <= max_bytes / sizeof(T))
00385 && (apos + (sizeof(T) * count)
00386 <= _tag.words() * sizeof(l4_umword_t));
00387 }
00388
00393
00402 template< typename T >
00403 unsigned long get(T *buf, unsigned long elems)
00404 {
00405 if (L4_UNLIKELY(!has_more<T>(elems)))
00406 return 0;
00407
00408 unsigned long size = elems * sizeof(T);
00409 _pos = cxx::Type_traits<T>::align(_pos);
00410
00411 __builtin_memcpy(buf, _current_msg + _pos, size);
00412 _pos += size;
00413 return elems;
00414 }
00415
00416
00421 template< typename T >
00422 void skip(unsigned long elems)
00423 {
00424 if (L4_UNLIKELY(!has_more<T>(elems)))
00425 return;
00426
00427 unsigned long size = elems * sizeof(T);
00428 _pos = cxx::Type_traits<T>::align(_pos);
00429 _pos += size;
00430 }
00431
00444 template< typename T >
00445 unsigned long get(Msg_ptr<T> const &buf, unsigned long elems = 1)
00446 {
00447 if (L4_UNLIKELY(!has_more<T>(elems)))
00448 return 0;
00449
00450 unsigned long size = elems * sizeof(T);
00451 _pos = cxx::Type_traits<T>::align(_pos);
00452
00453 buf.set(reinterpret_cast<T*>(_current_msg + _pos));
00454 _pos += size;
00455 return elems;
00456 }
00457
00458
00466 template< typename T >
00467 bool get(T &v)
00468 {
00469 if (L4_UNLIKELY(!has_more<T>()))
00470 {
00471 v = T();
00472 return false;
00473 }
00474
00475 _pos = cxx::Type_traits<T>::align(_pos);
00476 v = *(reinterpret_cast<T*>(_current_msg + _pos));
00477 _pos += sizeof(T);
00478 return true;
00479 }
00480
00481
00482 bool get(Ipc::Varg *va)
00483 {
00484 Ipc::Varg::Tag t;
00485 if (!has_more<Ipc::Varg::Tag>())
00486 {

```

```

00487 va->tag(0);
00488 return 0;
00489 }
00490 get(t);
00491 va->tag(t);
00492 char const *d;
00493 get(msg_ptr(d), va->length());
00494 va->data(d);
00495
00496 return 1;
00497 }
00498
00508 l4_msgtag_t tag() const { return _tag; }
00509
00510
00520 l4_msgtag_t &tag() { return _tag; }
00521
00523
00528 inline bool put(Buf_item const &);
00529
00534 inline bool put(Small_buf const &);
00535
00536
00541
00551 inline l4_msgtag_t wait(l4_umword_t *src)
00552 { return wait(src, L4_IPC_NEVER); }
00553
00564 inline l4_msgtag_t wait(l4_umword_t *src, l4_timeout_t timeout);
00565
00575 inline l4_msgtag_t receive(l4_cap_idx_t src)
00576 { return receive(src, L4_IPC_NEVER); }
00577 inline l4_msgtag_t receive(l4_cap_idx_t src,
00578 l4_timeout_t timeout);
00579
00580
00584 inline l4_utcb_t *utcb() const { return _utcb; }
00585
00586 protected:
00587 l4_msgtag_t _tag;
00588 l4_utcb_t *_utcb;
00589 char *_current_msg;
00590 unsigned _pos;
00591 unsigned char _current_buf;
00592 };
00593
00594 class Istream_copy : public Istream
00595 {
00596 private:
00597 l4_msg_regs_t _mrs;
00598
00599 public:
00600 Istream_copy(Istream const &o) : Istream(o), _mrs(*l4_utcb_mr_u(o.
00601 utcb()))
00602 {
00603 // do some reverse mr to utcb trickery
00604 _utcb = (l4_utcb_t*)((l4_addr_t)&mrs - (l4_addr_t)l4_utcb_mr_u((
00605 l4_utcb_t*)0));
00606 _current_msg = reinterpret_cast<char*>(l4_utcb_mr_u(_utcb)->mr);
00607 }
00608 };
00609
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 *(reinterpret_cast<T*>(_current_msg + _pos)) = v;
00679 _pos += sizeof(T);
00680 return true;
00681 }
00682
00683 int put(Varg const &va)
00684 {
00685 put(va.tag());
00686 put(va.data(), va.length());
00687 return 0;
00688 }
00689
00690 template< typename T >
00691 int put(Varg_t<T> const &va)
00692 { return put(static_cast<Varg const &>(va)); }
00693
00694 l4_msgtag_t tag() const { return _tag; }
00695
00696 l4_msgtag_t &tag() { return _tag; }
00697
00698 inline bool put_snd_item(Snd_item const &);
00699
00700 inline l4_msgtag_t send(l4_cap_idx_t dst, long proto = 0, unsigned flags = 0);
00701
00702 inline l4_utcb_t *utcb() const { return _utcb; }
00703 #if 0
00704 unsigned long tell() const
00705 {
00706 unsigned w = (_pos + sizeof(l4_umword_t)-1) / sizeof(l4_umword_t);
00707 w -= _current_item * 2;
00708 _tag = l4_msgtag(0, w, _current_item, 0);
00709 }
00710 #endif
00711 public:
00712 l4_msgtag_t prepare_ipc(long proto = 0, unsigned flags = 0)
00713 {
00714 unsigned w = (_pos + sizeof(l4_umword_t)-1) / sizeof(l4_umword_t);
00715 w -= _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

```

```

00817 void reset()
00818 {
00819 Istream::reset();
00820 Ostream::reset();
00821 }
00822
00823
00831
00832 using Istream::get;
00833 using Istream::put;
00834 using Ostream::put;
00835
00837
00842
00858 inline l4_msgtag_t call(l4_cap_idx_t dst, l4_timeout_t timeout, long
proto = 0);
00859 inline l4_msgtag_t call(l4_cap_idx_t dst, long proto = 0);
00860
00876 inline l4_msgtag_t reply_and_wait(l4_umword_t *src_dst, long proto =
0)
00877 { return reply_and_wait(src_dst, L4_IPC_SEND_TIMEOUT_0, proto); }
00878
00879 inline l4_msgtag_t send_and_wait(l4_cap_idx_t dest,
l4_umword_t *src,
00880 long proto = 0)
00881 { return send_and_wait(dest, src, L4_IPC_SEND_TIMEOUT_0, proto); }
00882
00899 inline l4_msgtag_t reply_and_wait(l4_umword_t *src_dst,
l4_timeout_t timeout, long proto = 0);
00900
00901 inline l4_msgtag_t send_and_wait(l4_cap_idx_t dest,
l4_umword_t *src,
00902 l4_timeout_t timeout, long proto = 0);
00903 inline l4_msgtag_t reply(l4_timeout_t timeout, long proto = 0);
00904 inline l4_msgtag_t reply(long proto = 0)
00905 { return reply(L4_IPC_SEND_TIMEOUT_0, proto); }
00906
00908 };
00909
00910
00911 inline bool
00912 Ostream::put_snd_item(Snd_item const &v)
00913 {
00914 typedef Snd_item T;
00915 _pos = cxx::Type_traits<Snd_item>::align(_pos);
00916 if (Utcb_stream_check::check_utcb_data_offset(_pos + sizeof(T)))
00917 return false;
00918
00919 *(reinterpret_cast<T*>(_current_msg + _pos)) = v;
00920 _pos += sizeof(T);
00921 ++_current_item;
00922 return true;
00923 }
00924
00925
00926 inline bool
00927 Istream::put(Buf_item const &item)
00928 {
00929 if (_current_buf >= L4_UTCB_GENERIC_BUFFERS_SIZE - 3)
00930 return false;
00931
00932 l4_utcb_br_u(_utcb)->bdr &= ~L4_BDR_OFFSET_MASK;
00933
00934 reinterpret_cast<Buf_item&>(l4_utcb_br_u(_utcb)->br[_current_buf]) = item;
00935 _current_buf += 2;
00936 return true;
00937 }
00938
00939
00940 inline bool
00941 Istream::put(Small_buf const &item)
00942 {
00943 if (_current_buf >= L4_UTCB_GENERIC_BUFFERS_SIZE - 2)
00944 return false;
00945
00946 l4_utcb_br_u(_utcb)->bdr &= ~L4_BDR_OFFSET_MASK;
00947
00948 reinterpret_cast<Small_buf&>(l4_utcb_br_u(_utcb)->br[_current_buf]) = item;
00949 _current_buf += 1;
00950 return true;
00951 }
00952
00953
00954 inline l4_msgtag_t
00955 Ostream::send(l4_cap_idx_t dst, long proto, unsigned flags)
00956 {
00957 l4_msgtag_t tag = prepare_ipc(proto, L4_MSGTAG_FLAGS & flags);
00958 return l4_ipc_send(dst, _utcb, tag, L4_IPC_NEVER);

```

```

00959 }
00960
00961 inline l4_msgtag_t
00962 Iostream::call(l4_cap_idx_t dst, l4_timeout_t timeout, long label)
00963 {
00964 l4_msgtag_t tag = prepare_ipc(label);
00965 tag = l4_ipc_call(dst, Ostream::_utcb, tag, timeout);
00966 Istream::tag() = tag;
00967 Istream::_pos = 0;
00968 return tag;
00969 }
00970
00971 inline l4_msgtag_t
00972 Iostream::call(l4_cap_idx_t dst, long label)
00973 { return call(dst, L4_IPC_NEVER, label); }
00974
00975
00976 inline l4_msgtag_t
00977 Iostream::reply_and_wait(l4_umword_t *src_dst,
00978 l4_timeout_t timeout, long proto)
00979 {
00978 l4_msgtag_t tag = prepare_ipc(proto);
00979 tag = l4_ipc_reply_and_wait(Ostream::_utcb, tag, src_dst, timeout);
00980 Istream::tag() = tag;
00981 Istream::_pos = 0;
00982 return tag;
00983 }
00984
00985
00986 inline l4_msgtag_t
00987 Iostream::send_and_wait(l4_cap_idx_t dest, l4_umword_t *src,
00988 l4_timeout_t timeout, long proto)
00989 {
00990 l4_msgtag_t tag = prepare_ipc(proto);
00991 tag = l4_ipc_send_and_wait(dest, Ostream::_utcb, tag, src, timeout);
00992 Istream::tag() = tag;
00993 Istream::_pos = 0;
00994 return tag;
00995 }
00996
00997
00998 inline l4_msgtag_t
00999 Iostream::reply(l4_timeout_t timeout, long proto)
01000 {
01001 l4_msgtag_t tag = prepare_ipc(proto);
01002 tag = l4_ipc_send(L4_INVALID_CAP | L4_SYSF_REPLY, Ostream::_utcb,
01003 tag, timeout);
01004 Istream::tag() = tag;
01005 Istream::_pos = 0;
01006 return tag;
01007 }
01008
01009 inline l4_msgtag_t
01010 Istream::wait(l4_umword_t *src, l4_timeout_t timeout)
01011 {
01012 l4_msgtag_t res;
01013 res = l4_ipc_wait(_utcb, src, timeout);
01014 tag() = res;
01015 _pos = 0;
01016 return res;
01017 }
01018
01019 inline l4_msgtag_t
01020 Istream::receive(l4_cap_idx_t src, l4_timeout_t timeout)
01021 {
01022 l4_msgtag_t res;
01023 res = l4_ipc_receive(src, _utcb, timeout);
01024 tag() = res;
01025 _pos = 0;
01026 return res;
01027 }
01028
01029 } // namespace Ipc
01030 } // namespace L4
01031
01040 inline L4::Ipc::Istream &operator >> (
01041 L4::Ipc::Istream &s, bool &v) { s.get(v); return s; }
01042 inline L4::Ipc::Istream &operator >> (
01043 L4::Ipc::Istream &s, int &v) { s.get(v); return s; }
01044 inline L4::Ipc::Istream &operator >> (
01045 L4::Ipc::Istream &s, long int &v) { s.get(v); return s; }
01046 inline L4::Ipc::Istream &operator >> (
01047 L4::Ipc::Istream &s, long long int &v) { s.get(v); return s; }
01048 inline L4::Ipc::Istream &operator >> (
01049 L4::Ipc::Istream &s, unsigned int &v) { s.get(v); return s; }
01050 inline L4::Ipc::Istream &operator >> (
01051 L4::Ipc::Istream &s, unsigned long int &v) { s.get(v); return s; }

```

```

01046 inline L4::Ipc::Istream &operator >> (
 L4::Ipc::Istream &s, unsigned long long int &v) { s.get(v); return s; }
01047 inline L4::Ipc::Istream &operator >> (
 L4::Ipc::Istream &s, short int &v) { s.get(v); return s; }
01048 inline L4::Ipc::Istream &operator >> (
 L4::Ipc::Istream &s, unsigned short int &v) { s.get(v); return s; }
01049 inline L4::Ipc::Istream &operator >> (
 L4::Ipc::Istream &s, char &v) { s.get(v); return s; }
01050 inline L4::Ipc::Istream &operator >> (
 L4::Ipc::Istream &s, unsigned char &v) { s.get(v); return s; }
01051 inline L4::Ipc::Istream &operator >> (
 L4::Ipc::Istream &s, signed char &v) { s.get(v); return s; }
01052 inline L4::Ipc::Istream &operator << (
 L4::Ipc::Istream &s, L4::Ipc::Buf_item const &v) { s.put(v); return s; }
01053 inline L4::Ipc::Istream &operator << (
 L4::Ipc::Istream &s, L4::Ipc::Small_buf const &v) { s.put(v); return s; }
01054 inline L4::Ipc::Istream &operator >> (
 L4::Ipc::Istream &s, L4::Ipc::Snd_item &v)
01055 {
01056 L4_umword_t b, d;
01057 s >> b >> d;
01058 v = L4::Ipc::Snd_item(b, d);
01059 return s;
01060 }
01061 inline L4::Ipc::Istream &operator >> (
 L4::Ipc::Istream &s, L4::Ipc::Varg &v)
01062 { s.get(&v); return s; }
01063
01064
01073 inline
01074 L4::Ipc::Istream &operator >> (L4::Ipc::Istream &s,
 L4_msgtag_t &v)
01075 {
01076 v = s.tag();
01077 return s;
01078 }
01079
01097 template< typename T >
01098 inline
01099 L4::Ipc::Istream &operator >> (L4::Ipc::Istream &s,
 L4::Ipc::Internal::Buf_in<T> const &v)
01100 {
01101 unsigned long si;
01102 if (s.get(si) && s.has_more<T>(si))
01103 v.set_size(s.get(L4::Ipc::Msg_ptr<T>(v.buf()), si));
01104 else
01105 v.set_size(0);
01106 return s;
01107 }
01108
01109
01124 template< typename T >
01125 inline
01126 L4::Ipc::Istream &operator >> (L4::Ipc::Istream &s,
 L4::Ipc::Msg_ptr<T> const &v)
01127 {
01128 s.get(v);
01129 return s;
01130 }
01131
01132
01145 template< typename T >
01146 inline
01147 L4::Ipc::Istream &operator >> (L4::Ipc::Istream &s,
 L4::Ipc::Internal::Buf_cp_in<T> const &v)
01148 {
01149 unsigned long sz;
01150 s.get(sz);
01151 v.size() = s.get(v.buf(), cxx::min(v.size(), sz));
01152 return s;
01153 }
01154
01155
01166 template< typename T >
01167 inline
01168 L4::Ipc::Istream &operator >> (L4::Ipc::Istream &s,
 L4::Ipc::Str_cp_in<T> const &v)
01169 {
01170 unsigned long sz;
01171 s.get(sz);
01172 unsigned long rsz = s.get(v.buf(), cxx::min(v.size(), sz));
01173 if (rsz < v.size() && v.buf()[rsz - 1])
01174 ++rsz; // add the zero termination behind the received data
01175 if (rsz != 0)
01176 v.buf()[rsz - 1] = 0;
01177 v.size() = rsz;
01178 return s;
01179 }
01180
01181
01182 }

```

```

01183
01184
01193 inline L4::Ipc::Ostream &operator << (
 L4::Ipc::Ostream &s, bool v) { s.put(v); return s; }
01194 inline L4::Ipc::Ostream &operator << (
 L4::Ipc::Ostream &s, int v) { s.put(v); return s; }
01195 inline L4::Ipc::Ostream &operator << (
 L4::Ipc::Ostream &s, long int v) { s.put(v); return s; }
01196 inline L4::Ipc::Ostream &operator << (
 L4::Ipc::Ostream &s, long long int v) { s.put(v); return s; }
01197 inline L4::Ipc::Ostream &operator << (
 L4::Ipc::Ostream &s, unsigned int v) { s.put(v); return s; }
01198 inline L4::Ipc::Ostream &operator << (
 L4::Ipc::Ostream &s, unsigned long int v) { s.put(v); return s; }
01199 inline L4::Ipc::Ostream &operator << (
 L4::Ipc::Ostream &s, unsigned long long int v) { s.put(v); return s; }
01200 inline L4::Ipc::Ostream &operator << (
 L4::Ipc::Ostream &s, short int v) { s.put(v); return s; }
01201 inline L4::Ipc::Ostream &operator << (
 L4::Ipc::Ostream &s, unsigned short int v) { s.put(v); return s; }
01202 inline L4::Ipc::Ostream &operator << (
 L4::Ipc::Ostream &s, char v) { s.put(v); return s; }
01203 inline L4::Ipc::Ostream &operator << (
 L4::Ipc::Ostream &s, unsigned char v) { s.put(v); return s; }
01204 inline L4::Ipc::Ostream &operator << (
 L4::Ipc::Ostream &s, signed char v) { s.put(v); return s; }
01205 inline L4::Ipc::Ostream &operator << (
 L4::Ipc::Ostream &s, L4::Ipc::Snd_item const &v) { s.put_snd_item(v);
 return s; }
01206 template< typename T >
01207 inline L4::Ipc::Ostream &operator << (L4::Ipc::Ostream &s, L4::Cap<T> const &v)
01208 { s << L4::Ipc::Snd_fpage(v.fpage()); return s; }
01209
01210 inline L4::Ipc::Ostream &operator << (
 L4::Ipc::Ostream &s, L4::Ipc::Varg const &v)
01211 { s.put(v); return s; }
01212 template< typename T >
01213 inline L4::Ipc::Ostream &operator << (L4::Ipc::Ostream &s, L4::Ipc::Varg_t<T> const &v)
01214 { s.put(v); return s; }
01215
01227 inline
01228 L4::Ipc::Ostream &operator << (L4::Ipc::Ostream &s,
 L4_msgtag_t const &v)
01229 {
01230 s.tag() = v;
01231 return s;
01232 }
01233
01242 template< typename T >
01243 inline
01244 L4::Ipc::Ostream &operator << (L4::Ipc::Ostream &s,
 L4::Ipc::Internal::Buf_cp_out<T> const &v)
01245 {
01246 s.put(v.size());
01247 s.put(v.buf(), v.size());
01248 return s;
01249 }
01250
01251
01264 inline
01265 L4::Ipc::Ostream &operator << (L4::Ipc::Ostream &s, char const
 *v)
01266 {
01267 unsigned long l = __builtin_strlen(v) + 1;
01268 s.put(l);
01269 s.put(v, l);
01270 return s;
01271 }
01272
01273
01274 #ifdef L4_CXX_IPC_BACKWARD_COMPAT
01275 namespace L4 {
01276
01277 #if 0
01278 template< typename T > class Ipc_buf_cp_out : public Ipc::Buf_cp_out<T> {};
01279 template< typename T > class Ipc_buf_cp_in : public Ipc::Buf_cp_in<T> {};
01280 template< typename T > class Ipc_buf_in : public Ipc::Buf_in<T> {};
01281 template< typename T > class Msg_ptr : public Ipc::Msg_ptr<T> {};
01282 #endif
01283
01284 template< typename T >
01285 Ipc::Internal::Buf_cp_out<T> ipc_buf_cp_out(T *v, unsigned long size)
01286 L4_DEPRECATED("Use L4::Ipc::buf_cp_out() now");
01287
01288 template< typename T >
01289 Ipc::Internal::Buf_cp_out<T> ipc_buf_cp_out(T *v, unsigned long size)
01290 { return Ipc::Internal::Buf_cp_out<T>(v, size); }
01291

```



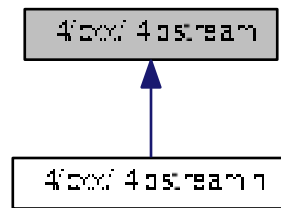
```

01292
01293 template< typename T >
01294 Ipc::Internal::Buf_cp_in<T> ipc_buf_cp_in(T *v, unsigned long &size)
01295 L4_DEPRECATED("Use L4::Ipc::buf_cp_in() now");
01296
01297 template< typename T >
01298 Ipc::Internal::Buf_cp_in<T> ipc_buf_cp_in(T *v, unsigned long &size)
01299 { return Ipc::Internal::Buf_cp_in<T>(v, size); }
01300
01301
01302 template< typename T >
01303 Ipc::Internal::Buf_in<T> ipc_buf_in(T *v, unsigned long &size)
01304 L4_DEPRECATED("Use L4::Ipc::buf_in() now");
01305
01306 template< typename T >
01307 Ipc::Internal::Buf_in<T> ipc_buf_in(T *v, unsigned long &size)
01308 { return Ipc::Internal::Buf_in<T>(v, size); }
01309
01310
01311 template< typename T >
01312 Ipc::Msg_ptr<T> msg_ptr(T *p)
01313 L4_DEPRECATED("Use L4::Ipc::msg_ptr() now");
01314
01315 template< typename T >
01316 Ipc::Msg_ptr<T> msg_ptr(T *p)
01317 { return Ipc::Msg_ptr<T>(p); }
01318
01319 typedef Ipc::Istream Ipc_istream L4_DEPRECATED("Use L4::Ipc::Istream now");
01320 typedef Ipc::Ostream Ipc_ostream L4_DEPRECATED("Use L4::Ipc::Ostream now");
01321 typedef Ipc::Iostream Ipc_iostream L4_DEPRECATED("Use L4::Ipc::Iostream now");
01322 typedef Ipc::Snd_fpage Snd_fpage L4_DEPRECATED("Use L4::Ipc::Snd_fpage
01323 now");
01324 typedef Ipc::Rcv_fpage Rcv_fpage L4_DEPRECATED("Use L4::Ipc::Rcv_fpage
01325 now");
01326
01327 namespace Ipc {
01328 template< typename T > class Buf_cp_in : public Internal::Buf_cp_in<T>
01329 {
01330 public:
01331 Buf_cp_in(T *v, unsigned long &size) : Internal::Buf_cp_in<T>(v, size) {}
01332 };
01333
01334 template< typename T >
01335 class Buf_cp_out : public Internal::Buf_cp_out<T>
01336 {
01337 public:
01338 Buf_cp_out(T const *v, unsigned long size) : Internal::Buf_cp_out<T>(v, size) {}
01339 };
01340
01341 template< typename T >
01342 class Buf_in : public Internal::Buf_in<T>
01343 {
01344 public:
01345 Buf_in(T *v, unsigned long &size) : Internal::Buf_in<T>(v, size) {}
01346 };
01347 } // namespace Ipc
01348 } // namespace L4
01349
01350 template< typename T >
01351 inline
01352 L4::Ipc::Istream &operator >> (L4::Ipc::Istream &s,
01353 L4::Ipc::Buf_cp_in<T> const &v)
01354 L4_DEPRECATED("Use L4::Ipc::buf_cp_in() now");
01355
01356 template< typename T >
01357 inline
01358 L4::Ipc::Istream &operator >> (L4::Ipc::Istream &s,
01359 L4::Ipc::Buf_cp_in<T> const &v)
01360 { return operator>>(s, static_cast<L4::Ipc::Internal::Buf_cp_in<T>>(v)); }
01361
01362 template< typename T >
01363 inline
01364 L4::Ipc::Istream &operator >> (L4::Ipc::Istream &s,
01365 L4::Ipc::Buf_in<T> const &v)
01366 L4_DEPRECATED("Use L4::Ipc::buf_in() now");
01367
01368 template< typename T >
01369 inline
01370 L4::Ipc::Istream &operator >> (L4::Ipc::Istream &s,
01371 L4::Ipc::Buf_in<T> const &v)
01372 { return operator>>(s, static_cast<L4::Ipc::Internal::Buf_in<T>>(v)); }
01373
01374 template< typename T >
01375 inline
01376 L4::Ipc::Ostream &operator << (L4::Ipc::Ostream &s, L4::Ipc::Buf_cp_out<T> const &v)

```



This graph shows which files directly or indirectly include this file:



### 15.117.1 Detailed Description

[L4](#) IO stream.

Definition in file [l4iostream](#).

## 15.118 l4iostream

```

00001
00005 /*
00006 * (c) 2004-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00007 * economic rights: Technische Universität Dresden (Germany)
00008 *
00009 * This file is part of TUD:OS and distributed under the terms of the
00010 * GNU General Public License 2.
00011 * Please see the COPYING-GPL-2 file for details.
00012 *
00013 * As a special exception, you may use this file as part of a free software
00014 * library without restriction. Specifically, if other files instantiate
00015 * templates or use macros or inline functions from this file, or you compile
00016 * this file and link it with other files to produce an executable, this
00017 * file does not by itself cause the resulting executable to be covered by
00018 * the GNU General Public License. This exception does not however
00019 * invalidate any other reasons why the executable file might be covered by
00020 * the GNU General Public License.
00021 */
00022
00023 #ifndef L4_L4IOSTREAM_H__
00024 #define L4_L4IOSTREAM_H__
00025
00026 #include <l4/cxx/basic_ostream>
00027 #include <l4/sys/types.h>
00028 #include <l4/sys/capability>
00029
00030 inline
00031 L4::BasicOStream &operator << (L4::BasicOStream &o, l4_msgtag_t const &tag)
00032 {
00033 L4::IOBackend::Mode m = o.be_mode();
00034 o << "[l=" << L4::dec << tag.label() << "; w=" << tag.words() << "; i="
00035 << tag.items() << "];";
00036 o.be_mode(m);
00037 return o;
00038 }
00039
00040 template<typename T>
00041 inline
00042 L4::BasicOStream &operator << (L4::BasicOStream &o, L4::Cap<T> const &cap)
00043 {
00044 o << "[C:" << L4::n_hex(cap.cap()) << "];";
00045 return o;
00046 }
00047
00048
00049 #endif

```



## 15.120 l4types.h

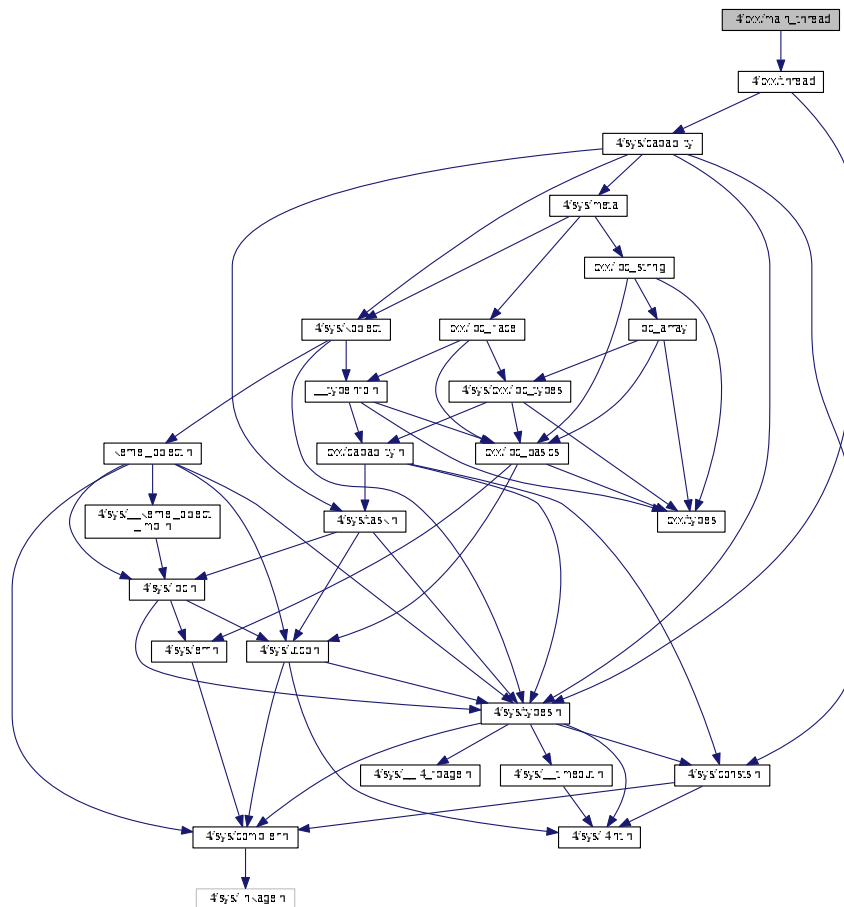
```
00001
00005 /*
00006 * (c) 2004-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00007 * economic rights: Technische Universität Dresden (Germany)
00008 *
00009 * This file is part of TUD:OS and distributed under the terms of the
00010 * GNU General Public License 2.
00011 * Please see the COPYING-GPL-2 file for details.
00012 *
00013 * As a special exception, you may use this file as part of a free software
00014 * library without restriction. Specifically, if other files instantiate
00015 * templates or use macros or inline functions from this file, or you compile
00016 * this file and link it with other files to produce an executable, this
00017 * file does not by itself cause the resulting executable to be covered by
00018 * the GNU General Public License. This exception does not however
00019 * invalidate any other reasons why the executable file might be covered by
00020 * the GNU General Public License.
00021 */
00022
00023 #pragma once
00024
00025 #include <l4/sys/types.h>
00026 #include <l4/cxx/basic_ostream>
00027
```

## 15.121 l4/cxx/main\_thread File Reference

Main thread.

```
#include <lib/cxx/thread>
```

Include dependency graph for `main_thread`:



## Namespaces

- [cxx](#)

*Our C++ library.*

### 15.121.1 Detailed Description

Main thread.

Definition in file [main\\_thread](#).

### 15.122 main\_thread

```
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)
```

```

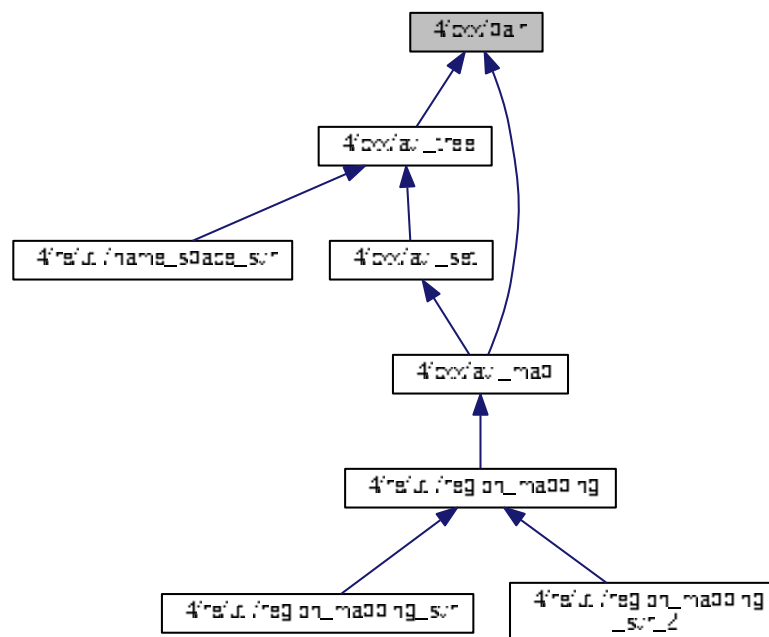
00009 *
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU General Public License 2.
00012 * Please see the COPYING-GPL-2 file for details.
00013 *
00014 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
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__ */

```

## 15.123 l4/cxx/pair File Reference

Pair implementation.

This graph shows which files directly or indirectly include this file:



## Data Structures

- struct `cxx::Pair< First, Second >`  
*Pair of two values.*
- class `cxx::Pair_first_compare< Cmp, Typ >`  
*Comparison functor for [Pair](#).*

## Namespaces

- `CXX`  
*Our C++ library.*

### 15.123.1 Detailed Description

Pair implementation.

Definition in file [pair](#).

## 15.124 pair

```

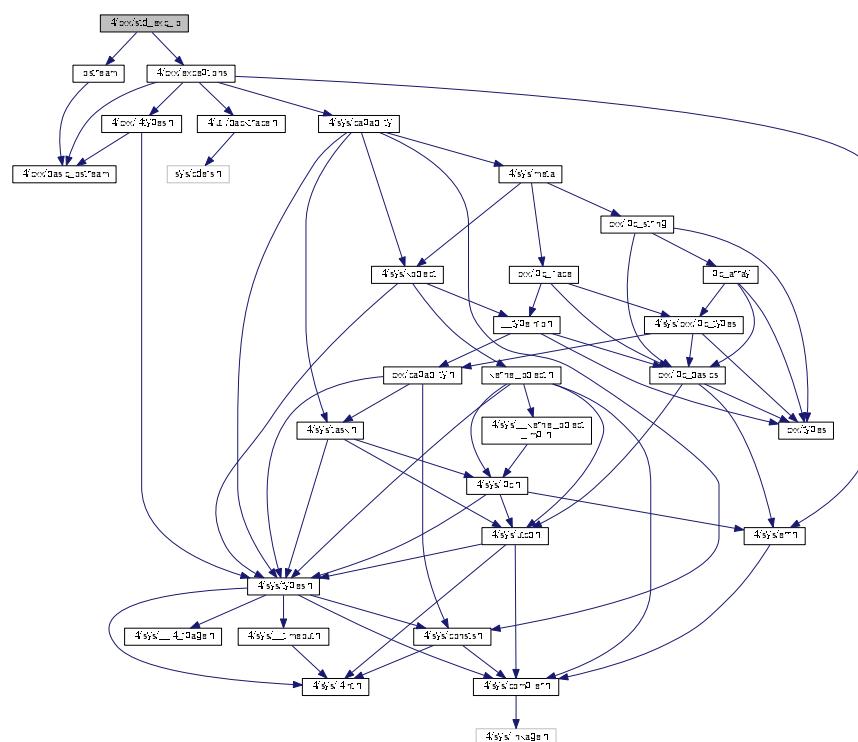
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007 * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU General Public License 2.
00012 * Please see the COPYING-GPL-2 file for details.
00013 *
00014 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
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 #pragma once
00024
00025 namespace cxx {
00026
00035 template< typename First, typename Second >
00036 struct Pair
00037 {
00039 typedef First First_type;
00041 typedef Second Second_type;
00042
00044 First first;
00046 Second second;
00047
00053 template<typename A1, typename A2>
00054 Pair(A1 &&first, A2 &&second)
00055 : first(first), second(second) {}
00056
00058 Pair() {}
00059 };
00060
00061 template< typename F, typename S >
00062 Pair<F,S> pair(F const &f, S const &s)
00063 { return cxx::Pair<F,S>(f,s); }
00064
00065
00074 template< typename Cmp, typename Typ >
00075 class Pair_first_compare
00076 {
00077 private:
00078 Cmp const &_cmp;

```



### 15.125 I4/cxx/std\_exc\_io File Reference

```
#include <l4/cxx/exceptions>
#include <iostream>
Include dependency graph for std_exc_io:
```



Definition in file `std_exc_io`.

## 15.126 std\_exc\_io

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003 * (c) 2011 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 * This file is part of TUD:OS and distributed under the terms of the
00008 * GNU General Public License 2.
00009 * Please see the COPYING-GPL-2 file for details.
00010 *
00011 * As a special exception, you may use this file as part of a free software
00012 * library without restriction. Specifically, if other files instantiate
00013 * templates or use macros or inline functions from this file, or you compile
00014 * this file and link it with other files to produce an executable, this
00015 * file does not by itself cause the resulting executable to be covered by
00016 * the GNU General Public License. This exception does not however
00017 * invalidate any other reasons why the executable file might be covered by
00018 * the GNU General Public License.
00019 */
00020 #pragma once
00021
00022 #include <l4/cxx/exceptions>
00023 #include <iostream>
00024
00025 inline
00026 std::ostream &
00027 operator << (std::ostream &o, L4::Base_exception const &e)
00028 {
00029 o << "Exception: " << e.str() << ", backtrace ...\n";
00030 for (int i = 0; i < e.frame_count(); ++i)
00031 o << (void *) (e.pc_array()[i]) << '\n';
00032 return o;
00033 }
00034
00035 inline
00036 std::ostream &
00037 operator << (std::ostream &o, L4::Runtime_error const &e)
00038 {
00039 o << "Exception: " << e.str() << ": ";
00040 if (e.extra_str())
00041 o << e.extra_str() << ": ";
00042 o << "backtrace ...\n";
00043 for (int i = 0; i < e.frame_count(); ++i)
00044 o << (void *) (e.pc_array()[i]) << '\n';
00045 return o;
00046 }

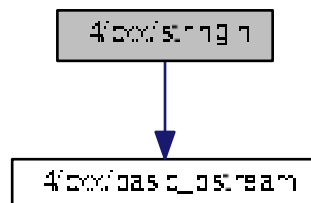
```

## 15.127 l4/cxx/string.h File Reference

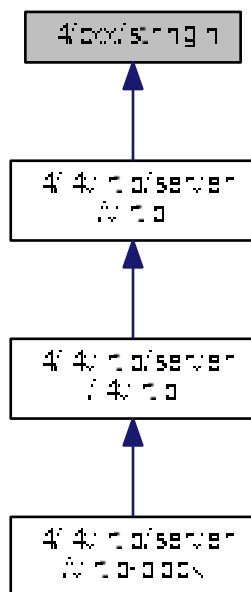
String.

```
#include <l4/cxx/basic_ostream>
```

Include dependency graph for string.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [L4::String](#)

*A null-terminated string container class.*

## Namespaces

- [L4](#)

*[L4](#) low-level kernel interface.*

### 15.127.1 Detailed Description

String.

Definition in file [string.h](#).

## 15.128 string.h

```

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 * 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 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
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_STRING_H__
00025 #define L4_CXX_STRING_H__
00026
00027 #include <l4/cxx/basic_ostream>
00028
00029 namespace L4 {
00030
00035 class String
00036 {
00037 public:
00038 String(char const *str = "") : _str(str)
00039 {
00040
00041 unsigned length() const
00042 {
00043 unsigned l;
00044 for (l = 0; _str[l]; l++)
00045 ;
00046 return l;
00047 }
00048
00049 char const *p_str() const { return _str; }
00050
00051 private:
00052 char const *_str;
00053 };
00054
00055 };
00056
00057 inline
00058 L4::BasicOStream &operator << (L4::BasicOStream &o, L4::String const &s)
00059 {
00060 o << s.p_str();
00061 return o;
00062 }
00063
00064 #endif /* L4_CXX_STRING_H__ */

```

## 15.129 l4/irq/irq.h File Reference

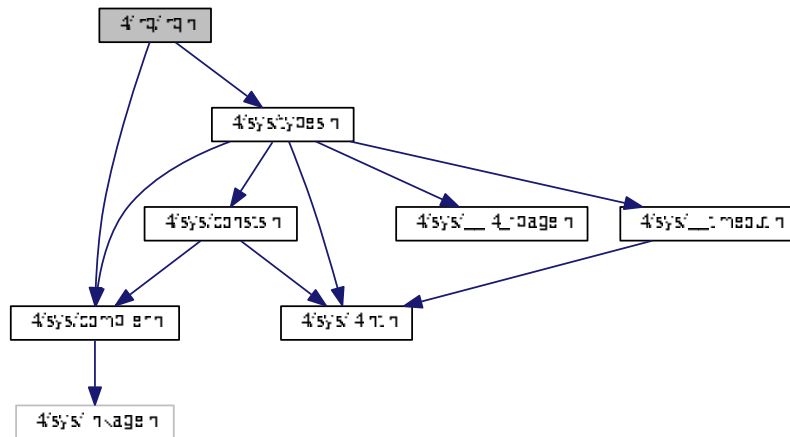
IRQ handling routines.

```

#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.*

### 15.129.1 Detailed Description

IRQ handling routines.

Definition in file [irq.h](#).

## 15.130 irq.h

```

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
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
00104 L4_CV long
00105 l4irq_unmask_and_wait_any(l4irq_t *unmask_irq, l4irq_t **ret_irq);
00106
00114 L4_CV long
00115 l4irq_wait_any(l4irq_t **irq);
00116
00127 L4_CV long
00128 l4irq_unmask(l4irq_t *irq);
00129
00137 L4_CV long
00138 l4irq_detach(l4irq_t *irq);
00139
00140
00141
00142 /*****
00164 L4_CV l4irq_t *
00165 l4irq_request(int irqnum, void (*isr_handler)(void *), void *isr_data,
00166 int irq_thread_prio, unsigned mode);
00167
00175 L4_CV long
00176 l4irq_release(l4irq_t *irq);
00177
00178

```

```

00179
00180 /*****
00181 /*****
00182
00188 L4_CV l4irq_t *
00199 l4irq_attach_cap(l4_cap_idx_t irqcap);
00200
00212 L4_CV l4irq_t *
00213 l4irq_attach_cap_ft(l4_cap_idx_t irqcap, unsigned mode);
00214
00225 L4_CV l4irq_t *
00226 l4irq_attach_thread_cap(l4_cap_idx_t irqcap,
00227 l4_cap_idx_t to_thread);
00227
00239 L4_CV l4irq_t *
00240 l4irq_attach_thread_cap_ft(l4_cap_idx_t irqcap,
00241 l4_cap_idx_t to_thread,
00242 unsigned mode);
00242
00243 /*****
00264 L4_CV l4irq_t *
00265 l4irq_request_cap(l4_cap_idx_t irqcap,
00266 void (*isr_handler)(void *), void *isr_data,
00267 int irq_thread_prio, unsigned mode);
00268
00269 __END_DECLS

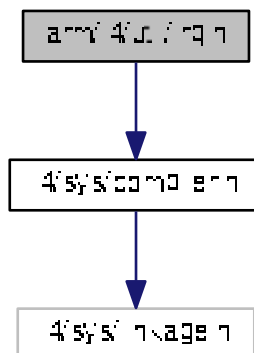
```

## 15.131 arm/l4/util/irq.h File Reference

ARM specific implementation of irq functions.

#include <l4/sys/compiler.h>

Include dependency graph for irq.h:



### 15.131.1 Detailed Description

ARM specific implementation of irq functions.

Do not use.

Definition in file [irq.h](#).

## 15.132 irq.h

```

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 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU Lesser General Public License 2.1.
00014 * Please see the COPYING-LGPL-2.1 file for details.
00015 */
00016 #ifndef __L4UTIL__ARCH_ARCH__IRQ_H__
00017 #define __L4UTIL__ARCH_ARCH__IRQ_H__
00018
00019 #ifdef __GNUC__
00020
00021 #include <l4/sys/compiler.h>
00022
00023 EXTERN_C_BEGIN
00024
00025 L4_INLINE void l4util_cli (void);
00026 L4_INLINE void l4util_sti (void);
00027 L4_INLINE void l4util_flags_save(l4_umword_t *flags);
00028 L4_INLINE void l4util_flags_restore(l4_umword_t *flags);
00029
00030 L4_INLINE
00031 void
00032 l4util_cli(void)
00033 {
00034 extern void __do_not_use_l4util_cli(void);
00035 __do_not_use_l4util_cli();
00036 }
00037
00038
00039 L4_INLINE
00040 void
00041 l4util_sti(void)
00042 {
00043 extern void __do_not_use_l4util_sti(void);
00044 __do_not_use_l4util_sti();
00045 }
00046
00047
00048 L4_INLINE
00049 void
00050 l4util_flags_save(l4_umword_t *flags)
00051 {
00052 (void) flags;
00053 extern void __do_not_use_l4util_flags_save(void);
00054 __do_not_use_l4util_flags_save();
00055 }
00056
00057 L4_INLINE
00058 void
00059 l4util_flags_restore(l4_umword_t *flags)
00060 {
00061 (void) flags;
00062 extern void __do_not_use_l4util_flags_restore(void);
00063 __do_not_use_l4util_flags_restore();
00064 }
00065
00066 EXTERN_C_END
00067
00068 #endif // __GNUC__
00069
00070 #endif /* ! __L4UTIL__ARCH_ARCH__IRQ_H__ */

```

## 15.133 amd64/l4/util/irq.h File Reference

some PIC and hardware interrupt related functions

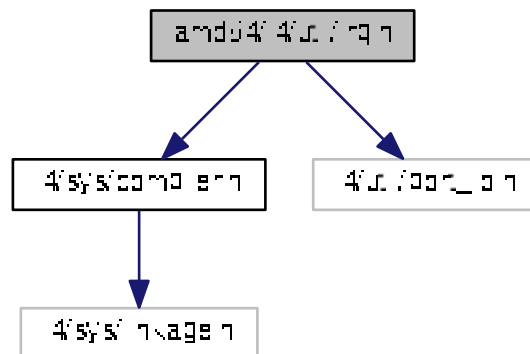
```

#include <l4/sys/compiler.h>
#include <l4/util/port_io.h>

```



Include dependency graph for irq.h:



## Functions

- void [l4util\\_irq\\_acknowledge](#) (unsigned int irq)  
*Acknowledge IRQ at PIC in fully special nested mode.*

### 15.133.1 Detailed Description

some PIC and hardware interrupt related functions

Date

2003

Author

Jork Loeser [jork.loeser@inf.tu-dresden.de](mailto:jork.loeser@inf.tu-dresden.de) Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [irq.h](#).

### 15.133.2 Function Documentation

#### 15.133.2.1 l4util\_irq\_acknowledge()

```
void l4util_irq_acknowledge (
 unsigned int irq) [inline]
```

Acknowledge IRQ at PIC in fully special nested mode.

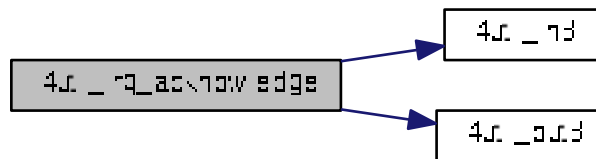
## Parameters

|            |                                    |
|------------|------------------------------------|
| <i>irq</i> | number of interrupt to acknowledge |
|------------|------------------------------------|

Definition at line 66 of file [irq.h](#).

References [EXTERN\\_C\\_END](#), [l4util\\_in8\(\)](#), and [l4util\\_out8\(\)](#).

Here is the call graph for this function:



## 15.134 irq.h

```

00001
00010 /*
00011 * (c) 2003-2009 Author(s)
00012 * economic rights: Technische Universität Dresden (Germany)
00013 * This file is part of TUD:OS and distributed under the terms of the
00014 * GNU Lesser General Public License 2.1.
00015 * Please see the COPYING-LGPL-2.1 file for details.
00016 */
00017
00018 #ifndef __L4_IRQ_H__
00019 #define __L4_IRQ_H__
00020
00021 #include <l4/sys/compiler.h>
00022 #include <l4/util/port_io.h>
00023
00024 EXTERN_C_BEGIN
00025
00029 L4_INLINE void
00030 l4util_irq_acknowledge(unsigned int irq);
00031
00034 static inline void
00035 l4util_cli(void)
00036 {
00037 __asm__ __volatile__ ("cli" : : : "memory");
00038 }
00039
00042 static inline void
00043 l4util_sti(void)
00044 {
00045 __asm__ __volatile__ ("sti" : : : "memory");
00046 }
00047
00051 static inline void
00052 l4util_flags_save(l4_umword_t *flags)
00053 {
00054 __asm__ __volatile__ ("pushf ; popq %0" : "=g" (*flags) : : "memory");
00055 }
00056
00059 static inline void
00060 l4util_flags_restore(l4_umword_t *flags)
00061 {
00062 __asm__ __volatile__ ("pushq %0 ; popf" : : "g" (*flags) : "memory");
00063 }

```

```

00064
00065 L4_INLINE void
00066 l4util_irq_acknowledge(unsigned int irq)
00067 {
00068 if (irq > 7)
00069 {
00070 l4util_out8(0x60+(irq & 7), 0xA0);
00071 l4util_out8(0x0B, 0xA0);
00072 if (l4util_in8(0xA0) == 0)
00073 l4util_out8(0x60 + 2, 0x20);
00074 }
00075 else
00076 l4util_out8(0x60+irq, 0x20); /* acknowledge the irq */
00077 };
00078
00079 EXTERN_C_END
00080
00081 #endif

```

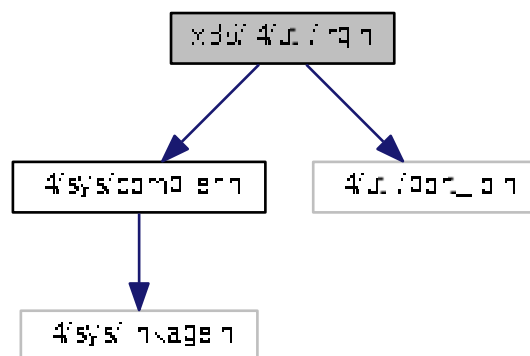
## 15.135 x86/I4/util/irq.h File Reference

some PIC and hardware interrupt related functions

```

#include <l4/sys/compiler.h>
#include <l4/util/port_io.h>
Include dependency graph for irq.h:

```



### Functions

- void `l4util_irq_acknowledge` (unsigned int irq)  
Acknowledge IRQ at PIC in fully special nested mode.

### 15.135.1 Detailed Description

some PIC and hardware interrupt related functions

## Date

2003

## Author

Jork Loeser [jork.loeser@inf.tu-dresden.de](mailto:jork.loeser@inf.tu-dresden.de) Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de)Definition in file [irq.h](#).

## 15.135.2 Function Documentation

## 15.135.2.1 l4util\_irq\_acknowledge()

```
void l4util_irq_acknowledge (
 unsigned int irq) [inline]
```

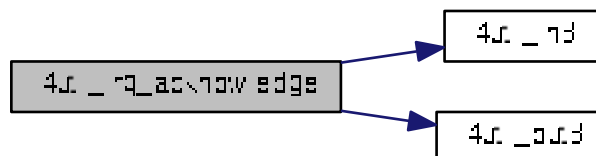
Acknowledge IRQ at PIC in fully special nested mode.

## Parameters

|            |                                    |
|------------|------------------------------------|
| <i>irq</i> | number of interrupt to acknowledge |
|------------|------------------------------------|

Definition at line 66 of file [irq.h](#).References [EXTERN\\_C\\_END](#), [l4util\\_in8\(\)](#), and [l4util\\_out8\(\)](#).

Here is the call graph for this function:



## 15.136 irq.h

```
00001
00010 /*
00011 * (c) 2003-2009 Author(s)
00012 * economic rights: Technische Universität Dresden (Germany)
```

```

00013 * This file is part of TUD:OS and distributed under the terms of the
00014 * GNU Lesser General Public License 2.1.
00015 * Please see the COPYING-LGPL-2.1 file for details.
00016 */
00017
00018 #ifndef __L4_IRQ_H__
00019 #define __L4_IRQ_H__
00020
00021 #include <l4/sys/compiler.h>
00022 #include <l4/util/port_io.h>
00023
00024 EXTERN_C_BEGIN
00025
00026 L4_INLINE void
00027 l4util_irq_acknowledge(unsigned int irq);
00028
00029 static inline void
00030 l4util_cli (void)
00031 {
00032 __asm__ __volatile__ ("cli" : : : "memory");
00033 }
00034
00035 static inline void
00036 l4util_sti (void)
00037 {
00038 __asm__ __volatile__ ("sti" : : : "memory");
00039 }
00040
00041 static inline void
00042 l4util_flags_save (l4_umword_t *flags)
00043 {
00044 __asm__ __volatile__ ("pushfl ; popl %0" : "=g" (*flags) : : "memory");
00045 }
00046
00047 static inline void
00048 l4util_flags_restore (l4_umword_t *flags)
00049 {
00050 __asm__ __volatile__ ("pushl %0 ; popfl" : : "g" (*flags) : "memory");
00051 }
00052
00053 L4_INLINE void
00054 l4util_irq_acknowledge(unsigned int irq)
00055 {
00056 if (irq > 7)
00057 {
00058 l4util_out8(0x60+(irq & 7), 0xA0);
00059 l4util_out8(0x0B, 0xA0);
00060 if (l4util_in8(0xA0) == 0)
00061 l4util_out8(0x60 + 2, 0x20);
00062 }
00063 else
00064 l4util_out8(0x60+irq, 0x20); /* acknowledge the irq */
00065 };
00066
00067 EXTERN_C_END
00068
00069 #endif

```

## 15.137 l4/sys/irq.h File Reference

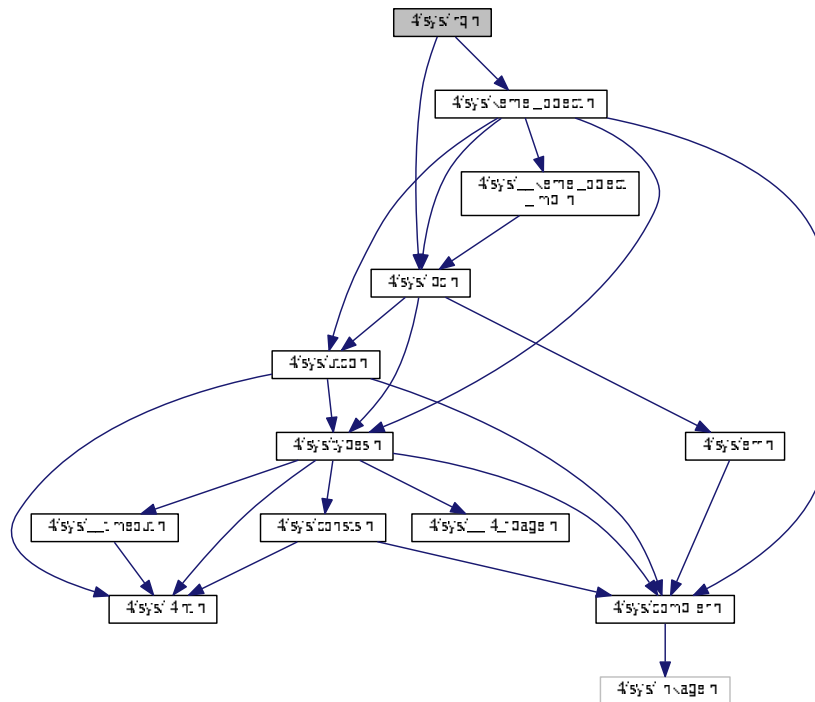
C Irq interface.

```

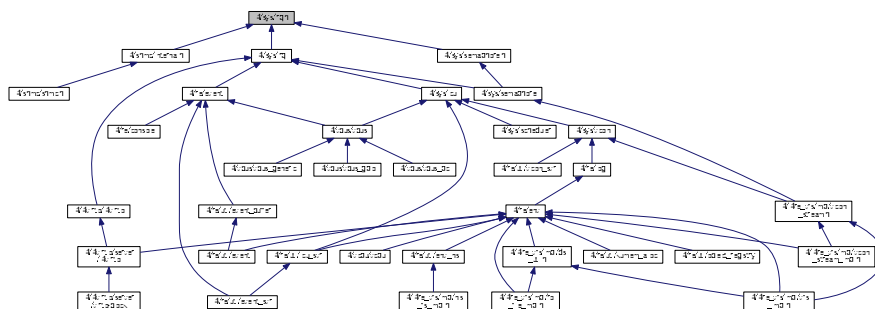
#include <l4/sys/kernel_object.h>
#include <l4/sys/ipc.h>

```

Include dependency graph for irq.h:



This graph shows which files directly or indirectly include this file:



## Functions

- `l4_msgtag_t l4_irq_attach (l4_cap_idx_t irq, l4_umword_t label, l4_cap_idx_t thread) L4_NOTHROW`  
*Attach a thread to an interrupt source.*
- `l4_msgtag_t l4_irq_attach_u (l4_cap_idx_t irq, l4_umword_t label, l4_cap_idx_t thread, l4_utcb_t *utcb) L4_NOTHROW`  
*Attach a thread to this interrupt.*
- `l4_msgtag_t l4_irq_mux_chain (l4_cap_idx_t irq, l4_cap_idx_t slave) L4_NOTHROW`  
*Chain an IRQ to another master IRQ source.*
- `l4_msgtag_t l4_irq_mux_chain_u (l4_cap_idx_t irq, l4_cap_idx_t slave, l4_utcb_t *utcb) L4_NOTHROW`

- Attach an IRQ to this multiplexer.*
- [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\\_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.*
- [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 IRQ.*

### 15.137.1 Detailed Description

C Irq interface.

Definition in file [irq.h](#).

## 15.138 irq.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 * economic rights: Technische Universität Dresden (Germany)
00011 *
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025 #pragma once
00026
00027 #include <l4/sys/kernel_object.h>
00028 #include <l4/sys/ipc.h>
00029
00065 L4_INLINE l4_msgtag_t
00066 l4_irq_attach(l4_cap_idx_t irq, l4_umword_t label,
00067 l4_cap_idx_t thread) L4_NOTHROW;
00068
00076 L4_INLINE l4_msgtag_t

```

```

00077 l4_irq_attach_u(l4_cap_idx_t irq, l4_umword_t label,
00078 l4_cap_idx_t thread, l4_utcb_t *utcb)
00079 L4_NOTHROW;
00099 L4_INLINE l4_msgtag_t
00100 l4_irq_mux_chain(l4_cap_idx_t irq, l4_cap_idx_t slave)
00101 L4_NOTHROW;
00108 L4_INLINE l4_msgtag_t
00109 l4_irq_mux_chain_u(l4_cap_idx_t irq, l4_cap_idx_t slave,
00110 l4_utcb_t *utcb) L4_NOTHROW;
00111
00120 L4_INLINE l4_msgtag_t
00121 l4_irq_detach(l4_cap_idx_t irq) L4_NOTHROW;
00122
00129 L4_INLINE l4_msgtag_t
00130 l4_irq_detach_u(l4_cap_idx_t irq, l4_utcb_t *utcb)
00131 L4_NOTHROW;
00132
00148 L4_INLINE l4_msgtag_t
00149 l4_irq_trigger(l4_cap_idx_t irq) L4_NOTHROW;
00150
00157 L4_INLINE l4_msgtag_t
00158 l4_irq_trigger_u(l4_cap_idx_t irq, l4_utcb_t *utcb)
00159 L4_NOTHROW;
00169 L4_INLINE l4_msgtag_t
00170 l4_irq_receive(l4_cap_idx_t irq, l4_timeout_t to)
00171 L4_NOTHROW;
00178 L4_INLINE l4_msgtag_t
00179 l4_irq_receive_u(l4_cap_idx_t irq, l4_timeout_t timeout,
00180 l4_utcb_t *utcb) L4_NOTHROW;
00180
00191 L4_INLINE l4_msgtag_t
00192 l4_irq_wait(l4_cap_idx_t irq, l4_umword_t *label,
00193 l4_timeout_t to) L4_NOTHROW;
00194
00201 L4_INLINE l4_msgtag_t
00202 l4_irq_wait_u(l4_cap_idx_t irq, l4_umword_t *label,
00203 l4_timeout_t timeout, l4_utcb_t *utcb)
00204 L4_NOTHROW;
00204
00215 L4_INLINE l4_msgtag_t
00216 l4_irq_unmask(l4_cap_idx_t irq) L4_NOTHROW;
00217
00224 L4_INLINE l4_msgtag_t
00225 l4_irq_unmask_u(l4_cap_idx_t irq, l4_utcb_t *utcb)
00226 L4_NOTHROW;
00226
00230 enum L4_irq_sender_op
00231 {
00232 L4_IRQ_SENDER_OP_ATTACH = 0,
00233 L4_IRQ_SENDER_OP_DETACH = 1
00234 };
00235
00239 enum L4_irq_mux_op
00240 {
00241 L4_IRQ_MUX_OP_CHAIN = 0
00242 };
00243
00247 enum L4_irq_op
00248 {
00249 L4_IRQ_OP_TRIGGER = 2,
00250 L4_IRQ_OP_EOI = 4
00251 };
00252
00253 /*****
00254 * Implementations
00255 */
00256
00257 L4_INLINE l4_msgtag_t
00258 l4_irq_attach_u(l4_cap_idx_t irq, l4_umword_t label,
00259 l4_cap_idx_t thread, l4_utcb_t *utcb)
00260 L4_NOTHROW
00260 {
00261 int items = 0;
00262 l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00263 m->mr[0] = L4_IRQ_SENDER_OP_ATTACH;
00264 m->mr[1] = label;
00265
00266 if (!l4_is_invalid_cap(thread))
00267 {
00268 items = 1;
00269 m->mr[2] = l4_map_obj_control(0, 0);
00270 m->mr[3] = l4_obj_fpage(thread, 0, L4_FPAGE_RWX).

```



```

 raw;
00271 }
00272 return l4_ipc_call(irq, utcb, l4_msgtag(
 L4_PROTO_IRQ_SENDER, 2, items, 0),
00273 L4_IPC_NEVER);
00274 }
00275
00276 L4_INLINE l4_msgtag_t
00277 l4_irq_mux_chain_u(l4_cap_idx_t irq, l4_cap_idx_t slave,
00278 l4_utcb_t *utcb) L4_NOTHROW
00279 {
00280 l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00281 m->mr[0] = L4_IRQ_MUX_OP_CHAIN;
00282 m->mr[1] = l4_map_obj_control(0, 0);
00283 m->mr[2] = l4_obj_fpage(slave, 0, L4_FPAGE_RWX).raw;
00284 return l4_ipc_call(irq, utcb, l4_msgtag(L4_PROTO_IRQ_MUX, 1, 1, 0),
00285 L4_IPC_NEVER);
00286 }
00287
00288 L4_INLINE l4_msgtag_t
00289 l4_irq_detach_u(l4_cap_idx_t irq, l4_utcb_t *utcb)
 L4_NOTHROW
00290 {
00291 l4_utcb_mr_u(utcb)->mr[0] = L4_IRQ_SENDER_OP_DETACH;
00292 return l4_ipc_call(irq, utcb, l4_msgtag(
 L4_PROTO_IRQ_SENDER, 1, 0, 0),
00293 L4_IPC_NEVER);
00294 }
00295
00296 L4_INLINE l4_msgtag_t
00297 l4_irq_trigger_u(l4_cap_idx_t irq, l4_utcb_t *utcb)
 L4_NOTHROW
00298 {
00299 return l4_ipc_send(irq, utcb, l4_msgtag(L4_PROTO_IRQ, 0, 0, 0),
00300 L4_IPC_BOTH_TIMEOUT_0);
00301 }
00302
00303 L4_INLINE l4_msgtag_t
00304 l4_irq_receive_u(l4_cap_idx_t irq, l4_timeout_t to,
 l4_utcb_t *utcb) L4_NOTHROW
00305 {
00306 l4_utcb_mr_u(utcb)->mr[0] = L4_IRQ_OP_EOI;
00307 return l4_ipc_call(irq, utcb, l4_msgtag(L4_PROTO_IRQ, 1, 0, 0), to);
00308 }
00309
00310 L4_INLINE l4_msgtag_t
00311 l4_irq_wait_u(l4_cap_idx_t irq, l4_umword_t *label,
 l4_timeout_t to, l4_utcb_t *utcb) L4_NOTHROW
00312 {
00313 l4_utcb_mr_u(utcb)->mr[0] = L4_IRQ_OP_EOI;
00314 return l4_ipc_send_and_wait(irq, utcb, l4_msgtag(
 L4_PROTO_IRQ, 1, 0, 0),
00315 label, to);
00316 }
00317
00318
00319 L4_INLINE l4_msgtag_t
00320 l4_irq_unmask_u(l4_cap_idx_t irq, l4_utcb_t *utcb)
 L4_NOTHROW
00321 {
00322 l4_utcb_mr_u(utcb)->mr[0] = L4_IRQ_OP_EOI;
00323 return l4_ipc_send(irq, utcb, l4_msgtag(L4_PROTO_IRQ, 1, 0, 0),
 L4_IPC_NEVER);
00324 }
00325
00326
00327 L4_INLINE l4_msgtag_t
00328 l4_irq_attach(l4_cap_idx_t irq, l4_umword_t label,
 l4_cap_idx_t thread) L4_NOTHROW
00329 {
00330 return l4_irq_attach_u(irq, label, thread, l4_utcb());
00331 }
00332
00333
00334 L4_INLINE l4_msgtag_t
00335 l4_irq_mux_chain(l4_cap_idx_t irq, l4_cap_idx_t slave)
 L4_NOTHROW
00336 {
00337 return l4_irq_mux_chain_u(irq, slave, l4_utcb());
00338 }
00339
00340 L4_INLINE l4_msgtag_t
00341 l4_irq_detach(l4_cap_idx_t irq) L4_NOTHROW
00342 {
00343 return l4_irq_detach_u(irq, l4_utcb());
00344 }
00345
00346 L4_INLINE l4_msgtag_t
00347 l4_irq_trigger(l4_cap_idx_t irq) L4_NOTHROW

```

```

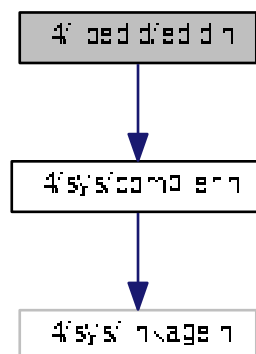
00348 {
00349 return l4_irq_trigger_u(irq, l4_utcb());
00350 }
00351
00352 L4_INLINE l4_msgtag_t
00353 l4_irq_receive(l4_cap_idx_t irq, l4_timeout_t to)
00354 L4_NOTHROW
00355 {
00356 return l4_irq_receive_u(irq, to, l4_utcb());
00357 }
00358 L4_INLINE l4_msgtag_t
00359 l4_irq_wait(l4_cap_idx_t irq, l4_umword_t *label,
00360 l4_timeout_t to) L4_NOTHROW
00361 {
00362 return l4_irq_wait_u(irq, label, to, l4_utcb());
00363 }
00364
00365 L4_INLINE l4_msgtag_t
00366 l4_irq_unmask(l4_cap_idx_t irq) L4_NOTHROW
00367 {
00368 return l4_irq_unmask_u(irq, l4_utcb());
00369 }
00370

```

## 15.139 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.*

## 15.140 edid.h

```

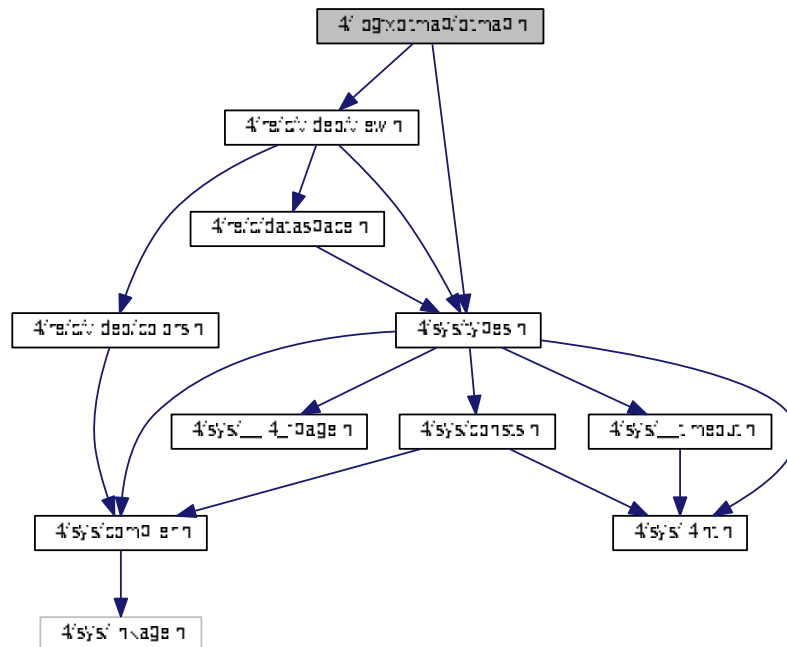
00001
00004 /*
00005 * (c) 2014 Matthias Lange <matthias.lange@kernkonzept.com>
00006 *
00007 * This file is part of TUD:OS and distributed under the terms of the
00008 * GNU Lesser General Public License 2.1.
00009 * Please see the COPYING-LGPL-2.1 file for details.
00010 */
00011 #pragma once
00012
00013 #include <14/sys/compiler.h>
00014
00023 enum Libedid_consts
00024 {
00025 Libedid_block_size = 128,
00026 };
00027
00028 __BEGIN_DECLS
00029
00036 int libedid_check_header(const unsigned char *edid);
00037
00044 int libedid_checksum(const unsigned char *edid);
00045
00052 unsigned libedid_version(const unsigned char *edid);
00053
00060 unsigned libedid_revision(const unsigned char *edid);
00061
00068 void libedid_pnp_id(const unsigned char *edid, unsigned char *id);
00069
00077 void libedid_preferred_resolution(const unsigned char *edid,
00078 unsigned *w, unsigned *h);
00079
00086 unsigned libedid_num_ext_blocks(const unsigned char *edid);
00087
00094 unsigned libedid_dump_standard_timings(const unsigned char *edid);
00095
00101 void libedid_dump(const unsigned char *edid);
00102
00105 __END_DECLS

```

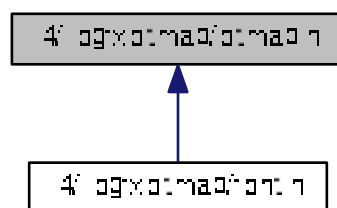
## 15.141 14/libgfxbitmap/bitmap.h File Reference

Bitmap renderer header file.

```
#include <l4/sys/types.h>
#include <l4/re/c/video/view.h>
Include dependency graph for bitmap.h:
```



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct [gfxbitmap\\_offset](#)  
offsets in *pmap[]* and *bmap[]*

## Macros

### Param macros for `bmap_*`

*Bitmap type - start least or start most significant bit*

- `#define PSLIM_BMAP_START_MSB 0x02`  
*'pbm'-style: "The bits are stored eight per byte, high bit first low bit last."*
- `#define PSLIM_BMAP_START_LSB 0x01`

## Typedefs

- typedef unsigned int `gfxbitmap_color_t`  
*Standard color type.*
- typedef unsigned int `gfxbitmap_color_pix_t`  
*Specific color type.*

## Functions

- `gfxbitmap_color_pix_t gfxbitmap_convert_color (l4re_video_view_info_t *vi, gfxbitmap_color_t rgb)`  
*Convert a color.*
- void `gfxbitmap_fill (l4_uint8_t *vfb, l4re_video_view_info_t *vi, int x, int y, int w, int h, gfxbitmap_color_pix_t color)`  
*Fill a rectangular area with a color.*
- void `gfxbitmap_bmap (l4_uint8_t *vfb, l4re_video_view_info_t *vi, l4_int16_t x, l4_int16_t y, l4_uint32_t w, l4_uint32_t h, l4_uint8_t *bmap, gfxbitmap_color_pix_t fg, gfxbitmap_color_pix_t bg, struct gfxbitmap_offset *offset, l4_uint8_t mode)`  
*Fill a rectangular area with a bicolor bitmap pattern.*
- void `gfxbitmap_set (l4_uint8_t *vfb, l4re_video_view_info_t *vi, l4_int16_t x, l4_int16_t y, l4_uint32_t w, l4_uint32_t h, l4_uint32_t xoffs, l4_uint32_t yoffs, l4_uint8_t *pmap, struct gfxbitmap_offset *offset, l4_uint32_t pwidth)`  
*Set area from source area.*
- void `gfxbitmap_copy (l4_uint8_t *dest, l4_uint8_t *src, l4re_video_view_info_t *vi, int x, int y, int w, int h, int dx, int dy)`  
*Copy a rectangular area.*

### 15.141.1 Detailed Description

Bitmap renderer header file.

Definition in file `bitmap.h`.

### 15.141.2 Macro Definition Documentation

## 15.141.2.1 pSLIM\_BMAP\_START\_LSB

```
#define pSLIM_BMAP_START_LSB 0x01
```

the other way round

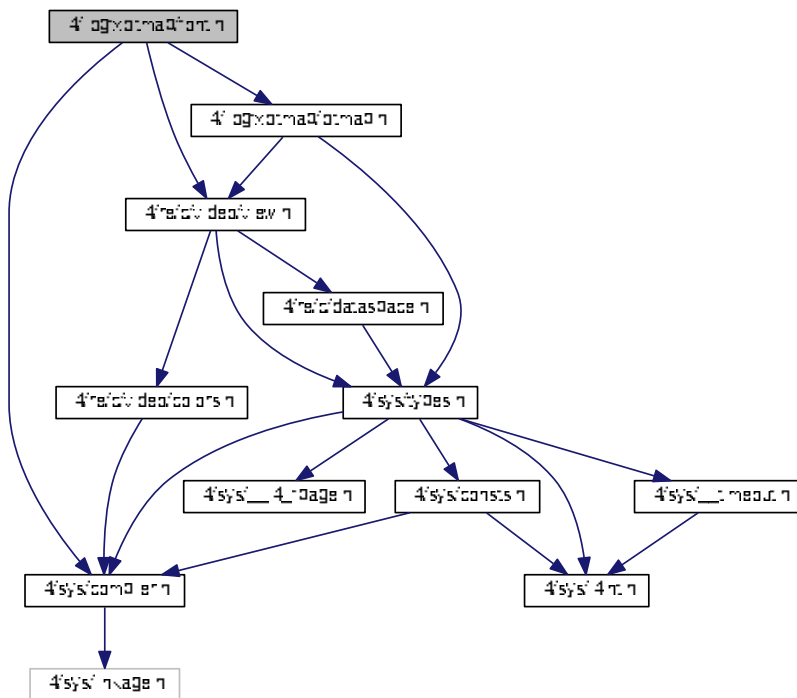
Definition at line 43 of file [bitmap.h](#).

## 15.142 bitmap.h

```
00001
00005 /*
00006 * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007 * economic rights: Technische Universität Dresden (Germany)
00008 * This file is part of TUD:OS and distributed under the terms of the
00009 * GNU Lesser General Public License 2.1.
00010 * Please see the COPYING-LGPL-2.1 file for details.
00011 */
00012 #pragma once
00013
00014 #include <l4/sys/types.h>
00015 #include <l4/re/c/video/view.h>
00016
00032 EXTERN_C_BEGIN
00038 #define pSLIM_BMAP_START_MSB 0x02
00041 #define pSLIM_BMAP_START_LSB 0x01
00043
00044
00048
00055 typedef unsigned int gfxbitmap_color_t;
00056
00064 typedef unsigned int gfxbitmap_color_pix_t;
00065
00067 struct gfxbitmap_offset
00068 {
00069 l4_uint32_t preskip_x;
00070 l4_uint32_t preskip_y;
00071 l4_uint32_t endskip_x;
00072 };
00073
00080 gfxbitmap_color_pix_t
00081 gfxbitmap_convert_color(l4re_video_view_info_t *vi,
00082 gfxbitmap_color_t rgb);
00082
00094 void
00095 gfxbitmap_fill(l4_uint8_t *vfb, l4re_video_view_info_t *vi,
00096 int x, int y, int w, int h, gfxbitmap_color_pix_t color);
00097
00115 void
00116 gfxbitmap_bmap(l4_uint8_t *vfb, l4re_video_view_info_t *vi,
00117 l4_int16_t x, l4_int16_t y, l4_uint32_t w,
00118 l4_uint32_t h, l4_uint8_t *bmap,
00119 gfxbitmap_color_pix_t fg,
00120 gfxbitmap_color_pix_t bg,
00121 struct gfxbitmap_offset *offset, l4_uint8_t mode);
00121
00137 void
00138 gfxbitmap_set(l4_uint8_t *vfb, l4re_video_view_info_t *vi,
00139 l4_int16_t x, l4_int16_t y, l4_uint32_t w,
00140 l4_uint32_t h, l4_uint32_t xoffs,
00141 l4_uint32_t yoffs,
00142 l4_uint8_t *pmap, struct gfxbitmap_offset *offset,
00143 l4_uint32_t pwidth);
00143
00157 void
00158 gfxbitmap_copy(l4_uint8_t *dest, l4_uint8_t *src,
00159 l4re_video_view_info_t *vi,
00160 int x, int y, int w, int h, int dx, int dy);
00162 EXTERN_C_END
```

Bitmap font renderer header file.

```
#include <l4/sys/compiler.h>
#include <l4/re/c/video/view.h>
#include <l4/libgfxbitmap/bitmap.h>
Include dependency graph for font.h:
```



- `#define GFXBITMAP_DEFAULT_FONT (void *)0`  
*Constant to use for the default font.*

- typedef void \* **gfxbitmap\_font\_t**  
*Font.*

- enum  
*Constant for length field.*

## Functions

- `int gfxbitmap_font_init (void)`  
*Initialize the library.*
- `gfxbitmap_font_t gfxbitmap_font_get (const char *name)`  
*Get a font descriptor.*
- `unsigned gfxbitmap_font_width (gfxbitmap_font_t font)`  
*Get the font width.*
- `unsigned gfxbitmap_font_height (gfxbitmap_font_t font)`  
*Get the font height.*
- `void * gfxbitmap_font_data (gfxbitmap_font_t font, unsigned c)`  
*Get bitmap font data for a specific character.*
- `void gfxbitmap_font_text (void *fb, l4re_video_view_info_t *vi, gfxbitmap_font_t font, const char *text, unsigned len, unsigned x, unsigned y, gfxbitmap_color_pix_t fg, gfxbitmap_color_pix_t bg)`  
*Render a string to a framebuffer.*
- `void gfxbitmap_font_text_scale (void *fb, l4re_video_view_info_t *vi, gfxbitmap_font_t font, const char *text, unsigned len, unsigned x, unsigned y, gfxbitmap_color_pix_t fg, gfxbitmap_color_pix_t bg, int scale_x, int scale_y)`  
*Render a string to a framebuffer, including scaling.*

### 15.143.1 Detailed Description

Bitmap font renderer header file.

Definition in file [font.h](#).

## 15.144 font.h

```

00001
00005 /*
00006 * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007 * economic rights: Technische Universität Dresden (Germany)
00008 * This file is part of TUD:OS and distributed under the terms of the
00009 * GNU Lesser General Public License 2.1.
00010 * Please see the COPYING-LGPL-2.1 file for details.
00011 */
00012 #pragma once
00013
00014 #include <l4/sys/compiler.h>
00015 #include <l4/re/c/video/view.h>
00016 #include <l4/libgfxbitmap/bitmap.h>
00017
00027
00031 #define GFXBITMAP_DEFAULT_FONT (void *)0
00032
00038 enum { GFXBITMAP_USE_STRLEN = ~0U };
00039
00040 EXTERN_C_BEGIN
00041
00043 typedef void *gfxbitmap_font_t;
00044
00053 L4_CV int gfxbitmap_font_init(void);
00054
00062 L4_CV gfxbitmap_font_t gfxbitmap_font_get(const char *name);
00063
00070 L4_CV unsigned
00071 gfxbitmap_font_width(gfxbitmap_font_t font);
00072
00079 L4_CV unsigned
00080 gfxbitmap_font_height(gfxbitmap_font_t font);
00081
00089 L4_CV void *
00090 gfxbitmap_font_data(gfxbitmap_font_t font, unsigned c);
00091

```



## 15.145 I4/libgfxbitmap/support File Reference

```
#include <l4/re/video/view>
Include dependency graph for support:
```



Generated for L4Re by Doxygen

## Date

2009

## Author

Adam Lackorzynski [adam@os.inf.tu-dresden.de](mailto:adam@os.inf.tu-dresden.de)Definition in file [support](#).

## 15.146 support

```

00001 /* vim:set ft=cpp: */
00009 /*
00010 * (c) 2009 Author(s)
00011 * economic rights: Technische Universität Dresden (Germany)
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU Lesser General Public License 2.1.
00014 * Please see the COPYING-LGPL-2.1 file for details.
00015 */
00016 #ifndef __LIBTERM_SUPPORT_H__
00017 #define __LIBTERM_SUPPORT_H__
00018
00019 #include <l4/re/video/view>
00020
00021 void
00022 libterm_init_colors(L4Re::Video::View::Info *fbi);
00023
00024 int
00025 libterm_get_color(int mode, int color);
00026
00027 #endif

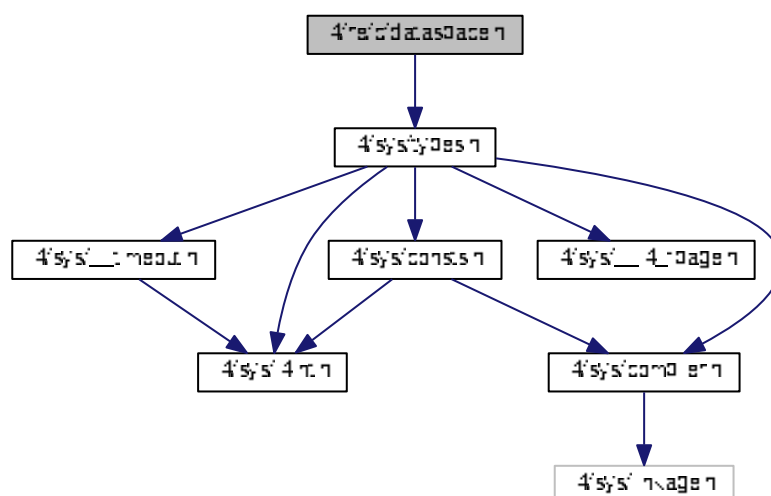
```

## 15.147 l4/re/c/dataspace.h File Reference

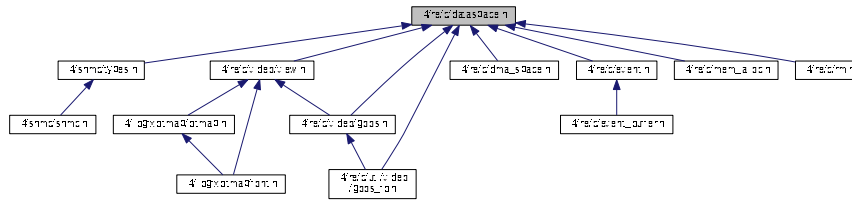
Data space C interface.

#include &lt;l4/sys/types.h&gt;

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.*

## Typedefs

- typedef [l4\\_cap\\_idx\\_t](#) [l4re\\_ds\\_t](#)  
*Dataspace type.*

## Enumerations

- enum [l4re\\_ds\\_map\\_flags](#) { ,  
[L4RE\\_DS\\_MAP\\_NORMAL](#) = 0x00, [L4RE\\_DS\\_MAP\\_CACHEABLE](#) = [L4RE\\_DS\\_MAP\\_NORMAL](#), [L4RE\\_DS\\_MAP\\_BUFFERABLE](#) = 0x10, [L4RE\\_DS\\_MAP\\_UNCACHEABLE](#) = 0x20,  
[L4RE\\_DS\\_MAP\\_CACHING\\_MASK](#) = 0x30, [L4RE\\_DS\\_MAP\\_CACHING\\_SHIFT](#) = 4 }
- Flags to specify the memory mapping type of a request.*

## Functions

- long [l4re\\_ds\\_clear](#) (const [l4re\\_ds\\_t](#) ds, [l4\\_addr\\_t](#) offset, unsigned long size) [L4\\_NOTHROW](#)
- long [l4re\\_ds\\_allocate](#) (const [l4re\\_ds\\_t](#) ds, [l4\\_addr\\_t](#) offset, [l4\\_size\\_t](#) size) [L4\\_NOTHROW](#)
- int [l4re\\_ds\\_copy\\_in](#) (const [l4re\\_ds\\_t](#) ds, [l4\\_addr\\_t](#) dst\_offs, const [l4re\\_ds\\_t](#) src, [l4\\_addr\\_t](#) src\_offs, unsigned long size) [L4\\_NOTHROW](#)
- unsigned long [l4re\\_ds\\_size](#) (const [l4re\\_ds\\_t](#) ds) [L4\\_NOTHROW](#)
- long [l4re\\_ds\\_flags](#) (const [l4re\\_ds\\_t](#) ds) [L4\\_NOTHROW](#)
- int [l4re\\_ds\\_info](#) (const [l4re\\_ds\\_t](#) ds, [l4re\\_ds\\_stats\\_t](#) \*stats) [L4\\_NOTHROW](#)
- int [l4re\\_ds\\_phys](#) (const [l4re\\_ds\\_t](#) ds, [l4\\_addr\\_t](#) offset, [l4\\_addr\\_t](#) \*phys\_addr, [l4\\_size\\_t](#) \*phys\_size) [L4\\_NOTHROW](#)

*Return physical address.*

### 15.147.1 Detailed Description

Data space C interface.

Definition in file [dataspace.h](#).

## 15.148 dataspace.h

```

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 * 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 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
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 #pragma once
00024
00031 #include <l4/sys/types.h>
00032
00033 EXTERN_C_BEGIN
00034
00039 typedef l4_cap_idx_t l4re_ds_t;
00040
00045 typedef struct {
00046 unsigned long size;
00047 unsigned long flags;
00048 } l4re_ds_stats_t;
00049
00054 enum l4re_ds_map_flags {
00055 L4RE_DS_MAP_FLAG_RO = 0,
00056 L4RE_DS_MAP_FLAG_RW = 1,
00057
00058 L4RE_DS_MAP_NORMAL = 0x00,
00059 L4RE_DS_MAP_CACHEABLE = L4RE_DS_MAP_NORMAL,
00060 L4RE_DS_MAP_BUFFERABLE = 0x10,
00061 L4RE_DS_MAP_UNCACHEABLE = 0x20,
00062 L4RE_DS_MAP_CACHING_MASK = 0x30,
00063 L4RE_DS_MAP_CACHING_SHIFT = 4,
00064 };
00065
00071 L4_CV int
00072 l4re_ds_map(const l4re_ds_t ds, l4_addr_t offset, unsigned long flags,
00073 l4_addr_t local_addr, l4_addr_t min_addr,
00074 l4_addr_t max_addr) L4_NOTHROW;
00075
00080 L4_CV int
00081 l4re_ds_map_region(const l4re_ds_t ds, l4_addr_t offset, unsigned long flags,
00082 l4_addr_t min_addr, l4_addr_t max_addr)
00083 L4_NOTHROW;
00084
00090 L4_CV long
00091 l4re_ds_clear(const l4re_ds_t ds, l4_addr_t offset, unsigned long size)
00092 L4_NOTHROW;
00093
00098 L4_CV long
00099 l4re_ds_allocate(const l4re_ds_t ds,
00100 l4_addr_t offset, l4_size_t size) L4_NOTHROW;
00101
00107 L4_CV int
00108 l4re_ds_copy_in(const l4re_ds_t ds, l4_addr_t dst_offs, const
00109 l4re_ds_t src,
00110 l4_addr_t src_offs, unsigned long size) L4_NOTHROW;
00111
00116 L4_CV unsigned long
00117 l4re_ds_size(const l4re_ds_t ds) L4_NOTHROW;
00118
00123 L4_CV long
00124 l4re_ds_flags(const l4re_ds_t ds) L4_NOTHROW;
00125
00130 L4_CV int
00131 l4re_ds_info(const l4re_ds_t ds, l4re_ds_stats_t *stats)
00132 L4_NOTHROW;
00133
00151 L4_CV int
00152 l4re_ds_phys(const l4re_ds_t ds, l4_addr_t offset,
00153 l4_addr_t *phys_addr, l4_size_t *phys_size)
00154 L4_NOTHROW;
00155
00156 EXTERN_C_END

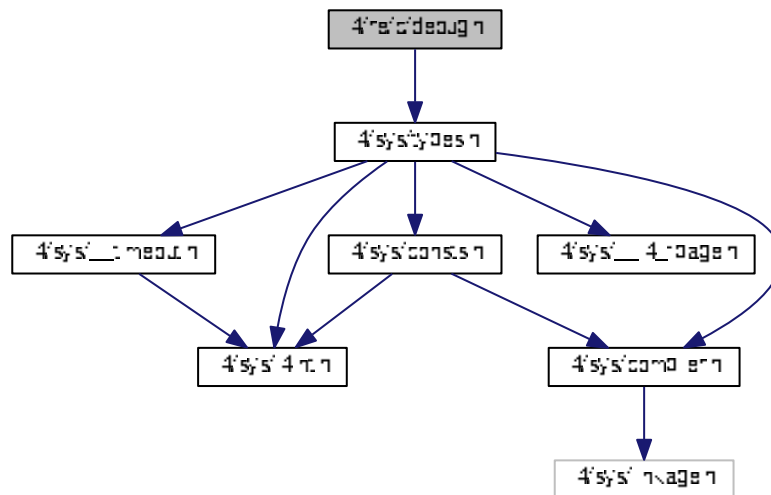
```

## 15.149 l4/re/c/debug.h File Reference

Debug C interface.

```
#include <l4/sys/types.h>
```

Include dependency graph for debug.h:



### Functions

- long [l4re\\_debug\\_obj\\_debug](#) ([l4\\_cap\\_idx\\_t](#) srv, unsigned long function) [L4\\_NOTHROW](#)  
Call debug function of [L4Re](#) service.

### 15.149.1 Detailed Description

Debug C interface.

Definition in file [debug.h](#).

## 15.150 debug.h

```

00001
00002 /*
00003 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00004 * economic rights: Technische Universität Dresden (Germany)
00005 *
00006 * This file is part of TUD:OS and distributed under the terms of the
00007 * GNU General Public License 2.
00008 * Please see the COPYING-GPL-2 file for details.
00009 *
00010 * As a special exception, you may use this file as part of a free software
00011 * library without restriction. Specifically, if other files instantiate
00012 * templates or use macros or inline functions from this file, or you compile
00013 * this file and link it with other files to produce an executable, this
00014 * file does not by itself cause the resulting executable to be covered by
00015 *

```

```

00018 * the GNU General Public License. This exception does not however
00019 * invalidate any other reasons why the executable file might be covered by
00020 * the GNU General Public License.
00021 */
00022 #pragma once
00023
00029 #include <l4/sys/types.h>
00030
00031 EXTERN_C_BEGIN
00032
00040 L4_CV long
00041 l4re_debug_obj_debug(l4_cap_idx_t srv, unsigned long function)
00042 L4_NOTHROW;
00043 EXTERN_C_END

```

## 15.151 l4re/c/dma\_space.h File Reference

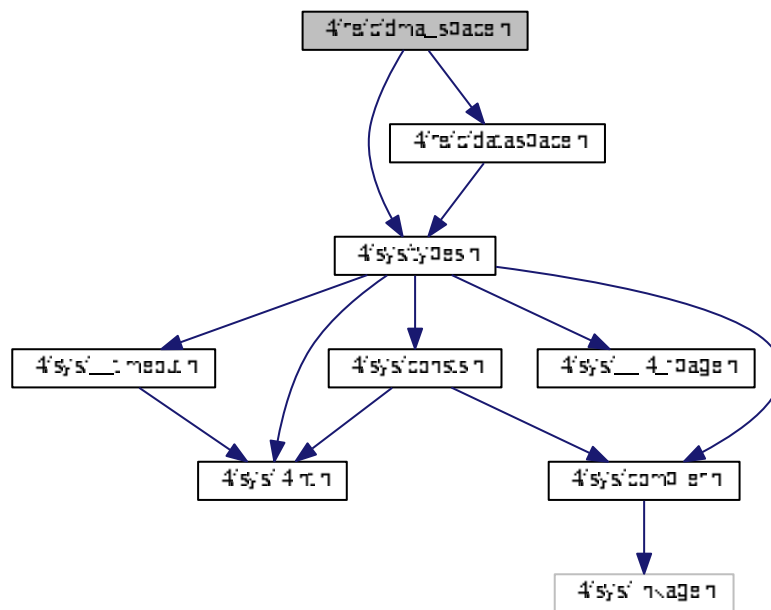
DMA space C interface.

```

#include <l4/sys/types.h>
#include <l4re/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

- long [l4re\\_dma\\_space\\_map](#) ([l4re\\_dma\\_space\\_t](#) dma, [l4re\\_ds\\_t](#) src, [l4\\_addr\\_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.*

- long [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.*

- long [l4re\\_dma\\_space\\_associate](#) ([l4re\\_dma\\_space\\_t](#) dma, [l4\\_cap\\_idx\\_t](#) dma\_task, unsigned long attr) [L4\\_NOTHROW](#)

*Associate a DMA task for a device to this Dma\_space.*

- long [l4re\\_dma\\_space\\_disassociate](#) ([l4re\\_dma\\_space\\_t](#) dma)

*Disassociate the DMA task from this Dma\_space.*

### 15.151.1 Detailed Description

DMA space C interface.

Definition in file [dma\\_space.h](#).

### 15.151.2 Typedef Documentation

#### 15.151.2.1 [l4re\\_dma\\_space\\_dma\\_addr\\_t](#)

```
typedef l4_uint64_t l4re_dma_space_dma_addr_t
```

Data type for DMA addresses.

Definition at line 62 of file [dma\\_space.h](#).

### 15.151.3 Enumeration Type Documentation

#### 15.151.3.1 [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 37 of file [dma\\_space.h](#).

## 15.151.3.2 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   |  |
| L4RE_DMA_SPACE_PHYS_SPACE |  |

Definition at line 48 of file [dma\\_space.h](#).

## 15.152 dma\_space.h

```

00001
00005 /*
00006 * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00007 *
00008 * This file is part of TUD:OS and distributed under the terms of the
00009 * GNU General Public License 2.
00010 * Please see the COPYING-GPL-2 file for details.
00011 *
00012 * As a special exception, you may use this file as part of a free software
00013 * library without restriction. Specifically, if other files instantiate
00014 * templates or use macros or inline functions from this file, or you compile
00015 * this file and link it with other files to produce an executable, this
00016 * file does not by itself cause the resulting executable to be covered by
00017 * the GNU General Public License. This exception does not however
00018 * invalidate any other reasons why the executable file might be covered by
00019 * the GNU General Public License.
00020 */
00021 #pragma once
00022
00029 #include <l4/sys/types.h>
00030 #include <l4/re/c/dataspace.h>
00031
00032 EXTERN_C_BEGIN
00033
00037 enum l4re_dma_space_direction
00038 {
00039 L4RE_DMA_SPACE_BIDIRECTIONAL,
00040 L4RE_DMA_SPACE_TO_DEVICE,
00041 L4RE_DMA_SPACE_FROM_DEVICE,

```



```

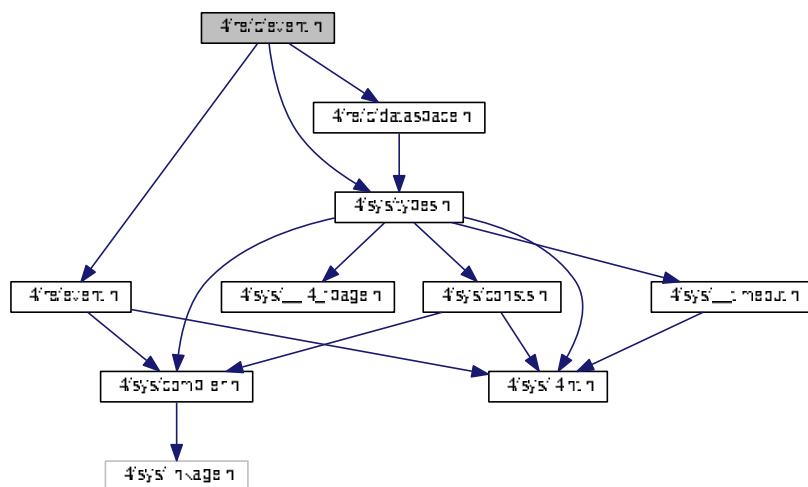
00042 L4RE_DMA_SPACE_NONE
00043 };
00044
00045 enum l4re_dma_space_space_attrbs
00046 {
00047 L4RE_DMA_SPACE_COHERENT = 1 << 0,
00048 L4RE_DMA_SPACE_PHYS_SPACE = 1 << 1,
00049 };
00050
00051 typedef l4_cap_idx_t l4re_dma_space_t;
00052
00053 typedef l4_uint64_t l4re_dma_space_dma_addr_t;
00054
00055 L4_CV long
00056 l4re_dma_space_map(l4re_dma_space_t dma,
00057 l4re_ds_t src, l4_addr_t offset,
00058 l4_size_t * size, unsigned long attrs,
00059 enum l4re_dma_space_direction dir,
00060 l4re_dma_space_dma_addr_t *dma_addr)
00061 L4_NOTHROW;
00062
00063 L4_CV long
00064 l4re_dma_space_unmap(l4re_dma_space_t dma,
00065 l4re_dma_space_dma_addr_t dma_addr,
00066 l4_size_t size, unsigned long attrs,
00067 enum l4re_dma_space_direction dir)
00068 L4_NOTHROW;
00069
00070 L4_CV long
00071 l4re_dma_space_associate(l4re_dma_space_t dma,
00072 l4_cap_idx_t dma_task,
00073 unsigned long attr) L4_NOTHROW;
00074
00075 L4_CV long
00076 l4re_dma_space_disassociate(l4re_dma_space_t dma);
00077
00078 EXTERN_C_END
00079

```

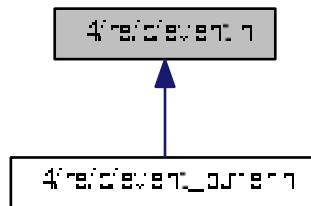
### 15.153 I4/re/c/event.h File Reference

Event C interface.

```
#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

- long [l4re\\_event\\_get\\_buffer](#) (const [l4\\_cap\\_idx\\_t](#) server, const [l4re\\_ds\\_t](#) ds) [L4\\_NOTHROW](#)  
*Get an event signal buffer.*
- long [l4re\\_event\\_get\\_num\\_streams](#) (const [l4\\_cap\\_idx\\_t](#) server) [L4\\_NOTHROW](#)  
*Get number of streams.*
- long [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.*
- long [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.*
- long [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.*

### 15.153.1 Detailed Description

Event C interface.

Definition in file [event.h](#).

## 15.154 event.h

```

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 * 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 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
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 #pragma once
00024
00031 #include <l4/sys/types.h>
00032 #include <l4/re/c/dataspace.h>
00033 #include <l4/re/event.h>
00034
00035 EXTERN_C_BEGIN
00036
00040 typedef struct
00041 {
00042 long long time;
00043 unsigned short type;
00044 unsigned short code;
00045 int value;
00046 l4_umword_t stream_id;
00047 } l4re_event_t;
00048
00060 L4_CV long
00061 l4re_event_get_buffer(const l4_cap_idx_t server,
00062 const l4re_ds_t ds) L4_NOTHROW;
00063
00074 L4_CV long
00075 l4re_event_get_num_streams(const l4_cap_idx_t server)
00076 L4_NOTHROW;
00077
00089 L4_CV long
00090 l4re_event_get_stream_info(const l4_cap_idx_t server,
00091 int idx, l4re_event_stream_info_t *info) L4_NOTHROW;
00092
00105 L4_CV long
00106 l4re_event_get_stream_info_for_id(const
00107 l4_cap_idx_t server,
00108 l4_umword_t stream_id,
00109 l4re_event_stream_info_t *info) L4_NOTHROW;
00110
00125 L4_CV long
00126 l4re_event_get_axis_info(const l4_cap_idx_t server,
00127 l4_umword_t id,
00128 unsigned naxes, unsigned const *axis,
00129 l4re_event_absinfo_t *info) L4_NOTHROW;
00130
00130 EXTERN_C_END

```

## 15.155 l4/re/event.h File Reference

Events.

```

#include <l4/sys/compiler.h>
#include <l4/sys/l4int.h>

```



```

00018 * the GNU General Public License. This exception does not however
00019 * invalidate any other reasons why the executable file might be covered by
00020 * the GNU General Public License.
00021 */
00022 #pragma once
00023
00024 #include <l4/sys/compiler.h>
00025 #include <l4/sys/l4int.h>
00026
00027 typedef struct L4_EXPORT_TYPE l4re_event_stream_id_t
00028 {
00029 l4_uint16_t bustype;
00030 l4_uint16_t vendor;
00031 l4_uint16_t product;
00032 l4_uint16_t version;
00033 } l4re_event_stream_id_t;
00034
00035 typedef struct L4_EXPORT_TYPE l4re_event_absinfo_t
00036 {
00037 l4_int32_t value;
00038 l4_int32_t min;
00039 l4_int32_t max;
00040 l4_int32_t fuzz;
00041 l4_int32_t flat;
00042 l4_int32_t resolution;
00043 } l4re_event_absinfo_t;
00044
00045 enum l4re_event_stream_max_values_t
00046 {
00047 L4RE_EVENT_EV_MAX = 0x1f,
00048 L4RE_EVENT_KEY_MAX = 0x1ff,
00049 L4RE_EVENT_REL_MAX = 0xf,
00050 L4RE_EVENT_ABS_MAX = 0x3f,
00051 L4RE_EVENT_PROP_MAX = 0x1f,
00052 L4RE_EVENT_SW_MAX = 0xf, // should be >= L4RE_SW_MAX
00053 };
00054
00055 enum l4re_event_stream_props_t
00056 {
00057 L4RE_EVENT_STREAM_CALIBRATE = 0x001,
00058 };
00059
00060
00061 #define __UNUM_B(x) ((x+1) + sizeof(unsigned long)*8 - 1) / (sizeof(unsigned long)*8)
00062
00063 typedef struct L4_EXPORT_TYPE l4re_event_stream_info_t
00064 {
00065 l4_umword_t stream_id;
00066 char name[32];
00067 char phys[32];
00068 l4re_event_stream_id_t id;
00069
00070 unsigned long propbits[__UNUM_B(L4RE_EVENT_PROP_MAX)];
00071
00072 unsigned long evbits[__UNUM_B(L4RE_EVENT_EV_MAX)];
00073 unsigned long keybits[__UNUM_B(L4RE_EVENT_KEY_MAX)];
00074 unsigned long relbits[__UNUM_B(L4RE_EVENT_REL_MAX)];
00075 unsigned long absbits[__UNUM_B(L4RE_EVENT_ABS_MAX)];
00076 unsigned long swbits[__UNUM_B(L4RE_EVENT_SW_MAX)];
00077
00078 } l4re_event_stream_info_t;
00079
00080 typedef struct L4_EXPORT_TYPE l4re_event_stream_state_t
00081 {
00082 unsigned long keybits[__UNUM_B(L4RE_EVENT_KEY_MAX)];
00083 unsigned long swbits[__UNUM_B(L4RE_EVENT_SW_MAX)];
00084 } l4re_event_stream_state_t;
00085
00086 #undef __UNUM_B
00087

```

## 15.157 l4/re/c/log.h File Reference

Log C interface.



```

00011 * Please see the COPYING-GPL-2 file for details.
00012 *
00013 * As a special exception, you may use this file as part of a free software
00014 * library without restriction. Specifically, if other files instantiate
00015 * templates or use macros or inline functions from this file, or you compile
00016 * this file and link it with other files to produce an executable, this
00017 * file does not by itself cause the resulting executable to be covered by
00018 * the GNU General Public License. This exception does not however
00019 * invalidate any other reasons why the executable file might be covered by
00020 * the GNU General Public License.
00021 */
00022 #pragma once
00023
00024 #include <l4/re/env.h>
00025
00026 EXTERN_C_BEGIN
00027
00028 L4_CV L4_INLINE void
00029 l4re_log_print(char const *string) L4_NOTHROW;
00030
00031 L4_CV L4_INLINE void
00032 l4re_log_printn(char const *string, int len) L4_NOTHROW;
00033
00034
00035 L4_CV void
00036 l4re_log_print_srv(const l4_cap_idx_t logcap,
00037 char const *string) L4_NOTHROW;
00038
00039 L4_CV void
00040 l4re_log_printn_srv(const l4_cap_idx_t logcap,
00041 char const *string, int len) L4_NOTHROW;
00042
00043 /***** Implementations *****/
00044
00045 L4_CV L4_INLINE void
00046 l4re_log_print(char const *string) L4_NOTHROW
00047 {
00048 l4re_log_print_srv(l4re_global_env->log, string);
00049 }
00050
00051 L4_CV L4_INLINE void
00052 l4re_log_printn(char const *string, int len) L4_NOTHROW
00053 {
00054 l4re_log_printn_srv(l4re_global_env->log, string, len);
00055 }
00056
00057 EXTERN_C_END

```

## 15.159 l4/re/c/mem\_alloc.h File Reference

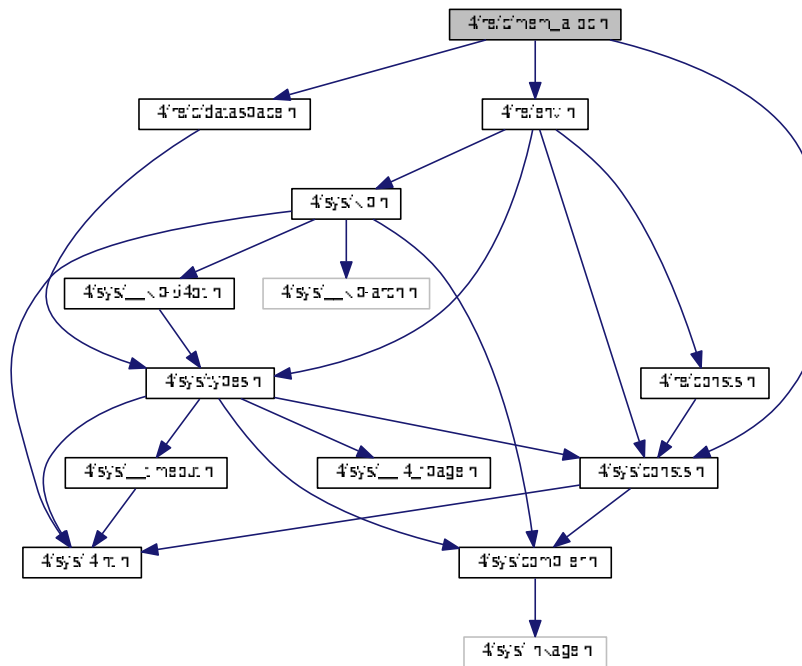
Memory allocator C interface.

```

#include <l4/re/env.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

- long [l4re\\_ma\\_alloc](#) (long size, [l4re\\_ds\\_t](#) const mem, unsigned long flags) [L4\\_NOTHROW](#)  
*Allocate memory.*
- long [l4re\\_ma\\_alloc\\_align](#) (long size, [l4re\\_ds\\_t](#) const mem, unsigned long flags, unsigned long align) [L4\\_NOTHROW](#)  
*Allocate memory.*
- long [l4re\\_ma\\_free](#) ([l4re\\_ds\\_t](#) const mem) [L4\\_NOTHROW](#))  
*Free memory.*
- long [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.*
- long [l4re\\_ma\\_free\\_srv](#) ([l4\\_cap\\_idx\\_t](#) srv, [l4re\\_ds\\_t](#) const mem) [L4\\_NOTHROW](#))  
*Free memory.*

### 15.159.1 Detailed Description

Memory allocator C interface.

Definition in file [mem\\_alloc.h](#).



## 15.160 mem\_alloc.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007 * economic rights: Technische Universität Dresden (Germany)
00008 *
00009 * This file is part of TUD:OS and distributed under the terms of the
00010 * GNU General Public License 2.
00011 * Please see the COPYING-GPL-2 file for details.
00012 *
00013 * As a special exception, you may use this file as part of a free software
00014 * library without restriction. Specifically, if other files instantiate
00015 * templates or use macros or inline functions from this file, or you compile
00016 * this file and link it with other files to produce an executable, this
00017 * file does not by itself cause the resulting executable to be covered by
00018 * the GNU General Public License. This exception does not however
00019 * invalidate any other reasons why the executable file might be covered by
00020 * the GNU General Public License.
00021 */
00022 #pragma once
00023
00024 #include <l4/re/env.h>
00025 #include <l4/sys/consts.h>
00026
00027 #include <l4/re/c/dataspace.h>
00028
00035 EXTERN_C_BEGIN
00036
00042 enum l4re_ma_flags {
00043 L4RE_MA_CONTINUOUS = 0x01,
00044 L4RE_MA_PINNED = 0x02,
00045 L4RE_MA_SUPER_PAGES = 0x04,
00046 };
00047
00048
00066 L4_CV L4_INLINE long
00067 l4re_ma_alloc(long size, l4re_ds_t const mem,
00068 unsigned long flags) L4_NOTHROW;
00069
00091 L4_CV L4_INLINE long
00092 l4re_ma_alloc_align(long size, l4re_ds_t const mem,
00093 unsigned long flags, unsigned long align) L4_NOTHROW;
00094
00106 /* Deprecation message added Q4 2016.
00107 * Remove together with L4Re::Mem_alloc::free() */
00108 L4_CV L4_INLINE long
00109 l4re_ma_free(l4re_ds_t const mem) L4_NOTHROW
00110 L4_DEPRECATED("This function is an empty stub and remains for backward compatibility only. See
00111 documentation for replacement options.");
00112
00113
00114
00133 L4_CV long
00134 l4re_ma_alloc_align_srv(l4_cap_idx_t srv, long size,
00135 l4re_ds_t const mem, unsigned long flags,
00136 unsigned long align) L4_NOTHROW;
00137
00150 /* Deprecation message added Q4 2016.
00151 * Remove together with L4Re::Mem_alloc::free() */
00152 L4_CV long
00153 l4re_ma_free_srv(l4_cap_idx_t srv, l4re_ds_t const mem)
00154 L4_NOTHROW
00155 L4_DEPRECATED("This function is an empty stub and remains for backward compatibility only. See
00156 documentation for replacement options.");
00157
00158
00159
00160 /***** Implementation *****/
00161
00162 /* Just warn actual users, but not for internal implementations */
00163 #pragma GCC diagnostic push
00164 #pragma GCC diagnostic ignored "-Wdeprecated-declarations"
00165
00166 L4_CV L4_INLINE long
00167 l4re_ma_alloc(long size, l4re_ds_t const mem,
00168 unsigned long flags) L4_NOTHROW
00169 {
00170 return l4re_ma_alloc_align_srv(l4re_global_env->
00171 mem_alloc, size, mem,
00172 flags, 0);
00173 }
00174

```



## Functions

- long `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.*
- long `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.*
- long `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.*

### 15.161.1 Detailed Description

Namespace functions, C interface.

Definition in file [namespace.h](#).

## 15.162 namespace.h

```

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 * 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 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
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 #pragma once
00024
00031 #include <l4/re/env.h>
00032
00039 enum l4re_ns_register_flags {
00040 L4RE_NS_REGISTER_RO = L4_FPAGE_RO,
00041 L4RE_NS_REGISTER_DIR = 0x10,
00042 L4RE_NS_REGISTER_RW = L4_FPAGE_RX,
00043 L4RE_NS_REGISTER_RWS = L4_FPAGE_RWX,
00044 L4RE_NS_REGISTER_S = L4_FPAGE_W,
00045 };
00046
00047 EXTERN_C_BEGIN
00048
00053 typedef l4_cap_idx_t l4re_namespace_t;
00054
00055
00056
00065 L4_CV long
00066 l4re_ns_query_to_srv(l4re_namespace_t srv, char const *name,
00067 l4_cap_idx_t const cap, int timeout) L4_NOTHROW;
00068
00085 L4_CV L4_INLINE long
00086 l4re_ns_query_srv(l4re_namespace_t srv, char const *name,
00087 l4_cap_idx_t const cap) L4_NOTHROW;
00088
00096 L4_CV long
00097 l4re_ns_register_obj_srv(l4re_namespace_t srv, char const *name,
00098 l4_cap_idx_t const obj, unsigned flags)
00099 L4_NOTHROW;
00099
00100

```



## Functions

- `int l4re_rm_reserve_area (l4_addr_t *start, unsigned long size, unsigned flags, unsigned char align) L4_NOTHROW`
- `int l4re_rm_free_area (l4_addr_t addr) L4_NOTHROW`
- `int l4re_rm_attach (void **start, unsigned long size, unsigned long flags, l4re_ds_t const mem, l4_addr_t offs, unsigned char align) L4_NOTHROW`
- `int l4re_rm_detach (void *addr) L4_NOTHROW`  
*Detach and unmap in current task.*
- `int l4re_rm_detach_ds (void *addr, l4re_ds_t *ds) L4_NOTHROW`  
*Detach, unmap and return affected dataspace in 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, l4_addr_t *offset, unsigned *flags, l4re_ds_t *m) L4_NOTHROW`
- `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, unsigned 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, unsigned long flags, l4re_ds_t const mem, l4_addr_t offs, unsigned char align) 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, l4_addr_t *offset, unsigned *flags, l4re_ds_t *m) L4_NOTHROW`
- `void l4re_rm_show_lists_srv (l4_cap_idx_t rm) L4_NOTHROW`  
*Dump region map internal data structures.*

### 15.163.1 Detailed Description

Region map interface, C interface.

Definition in file [rm.h](#).

## 15.164 rm.h

```

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 * 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 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
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 #pragma once

```

```

00024
00031 #include <l4/re/env.h>
00032 #include <l4/re/c/dataspace.h>
00033
00034 EXTERN_C_BEGIN
00035
00040 enum l4re_rm_flags_t {
00041 L4RE_RM_READ_ONLY = 0x01,
00042 L4RE_RM_NO_ALIAS = 0x02,
00043 L4RE_RM_PAGER = 0x04,
00044 L4RE_RM_RESERVED = 0x08,
00045 L4RE_RM_REGION_FLAGS = 0x0f,
00047 L4RE_RM_OVERMAP = 0x10,
00048 L4RE_RM_SEARCH_ADDR = 0x20,
00049 L4RE_RM_IN_AREA = 0x40,
00050 L4RE_RM_EAGER_MAP = 0x80,
00051 L4RE_RM_ATTACH_FLAGS = 0xf0,
00052 };
00053
00054
00055
00063 L4_CV L4_INLINE int
00064 l4re_rm_reserve_area(l4_addr_t *start, unsigned long size,
00065 unsigned flags, unsigned char align) L4_NOTHROW;
00066
00074 L4_CV L4_INLINE int
00075 l4re_rm_free_area(l4_addr_t addr) L4_NOTHROW;
00076
00085 L4_CV L4_INLINE int
00086 l4re_rm_attach(void **start, unsigned long size, unsigned long flags,
00087 l4re_ds_t const mem, l4_addr_t offs,
00088 unsigned char align) L4_NOTHROW;
00089
00090
00101 L4_CV L4_INLINE int
00102 l4re_rm_detach(void *addr) L4_NOTHROW;
00103
00116 L4_CV L4_INLINE int
00117 l4re_rm_detach_ds(void *addr, l4re_ds_t *ds)
00118 L4_NOTHROW;
00118
00130 L4_CV L4_INLINE int
00131 l4re_rm_detach_unmap(l4_addr_t addr, l4_cap_idx_t task)
00132 L4_NOTHROW;
00132
00145 L4_CV L4_INLINE int
00146 l4re_rm_detach_ds_unmap(void *addr, l4re_ds_t *ds,
00147 l4_cap_idx_t task) L4_NOTHROW;
00147
00148
00149
00155 L4_CV L4_INLINE int
00156 l4re_rm_find(l4_addr_t *addr, unsigned long *size,
00157 l4_addr_t *offset,
00158 unsigned *flags, l4re_ds_t *m) L4_NOTHROW;
00158
00165 L4_CV L4_INLINE void
00166 l4re_rm_show_lists(void) L4_NOTHROW;
00166
00167
00168
00169 /*
00170 * Variants of functions that also take a capability of the region map
00171 * service.
00172 */
00172
00173
00174
00179 L4_CV int
00180 l4re_rm_reserve_area_srv(l4_cap_idx_t rm,
00181 l4_addr_t *start, unsigned long size,
00182 unsigned flags, unsigned char align) L4_NOTHROW;
00182
00187 L4_CV int
00188 l4re_rm_free_area_srv(l4_cap_idx_t rm,
00189 l4_addr_t addr) L4_NOTHROW;
00189
00194 L4_CV int
00195 l4re_rm_attach_srv(l4_cap_idx_t rm, void **start, unsigned long size,
00196 unsigned long flags, l4re_ds_t const mem, l4_addr_t offs,
00197 unsigned char align) L4_NOTHROW;
00197
00198
00199
00204 L4_CV int
00205 l4re_rm_detach_srv(l4_cap_idx_t rm, l4_addr_t addr,
00206 l4re_ds_t *ds, l4_cap_idx_t task)
00207 L4_NOTHROW;
00207
00208
00213 L4_CV int

```

```

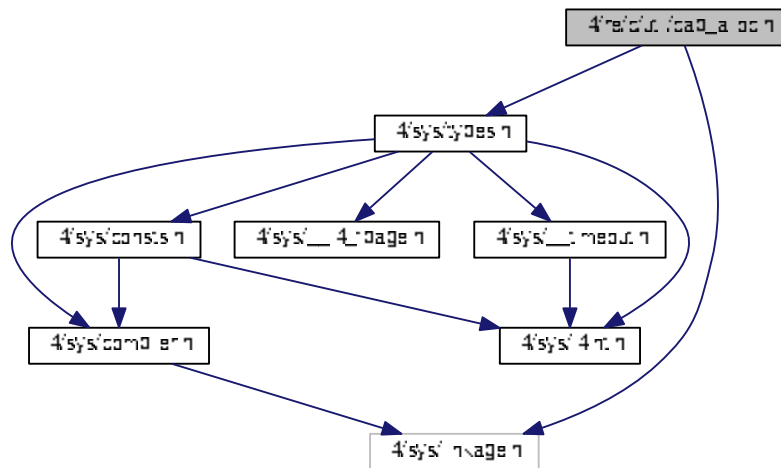
00214 l4re_rm_find_srv(l4_cap_idx_t rm, l4_addr_t *addr,
00215 unsigned long *size, l4_addr_t *offset,
00216 unsigned *flags, l4re_ds_t *m) L4_NOTHROW;
00217
00222 L4_CV void
00223 l4re_rm_show_lists_srv(l4_cap_idx_t rm)
00224 L4_NOTHROW;
00225
00226 /***** Implementations *****/
00227
00228 L4_CV L4_INLINE int
00229 l4re_rm_reserve_area(l4_addr_t *start, unsigned long size,
00230 unsigned flags, unsigned char align) L4_NOTHROW
00231 {
00232 return l4re_rm_reserve_area_srv(l4re_global_env->rm, start, size,
00233 flags, align);
00234 }
00235
00236 L4_CV L4_INLINE int
00237 l4re_rm_free_area(l4_addr_t addr) L4_NOTHROW
00238 {
00239 return l4re_rm_free_area_srv(l4re_global_env->rm, addr);
00240 }
00241
00242 L4_CV L4_INLINE int
00243 l4re_rm_attach(void **start, unsigned long size, unsigned long flags,
00244 l4re_ds_t const mem, l4_addr_t offs,
00245 unsigned char align) L4_NOTHROW
00246 {
00247 return l4re_rm_attach_srv(l4re_global_env->rm, start, size,
00248 flags, mem, offs, align);
00249 }
00250
00251
00252 L4_CV L4_INLINE int
00253 l4re_rm_detach(void *addr) L4_NOTHROW
00254 {
00255 return l4re_rm_detach_srv(l4re_global_env->rm,
00256 (l4_addr_t)addr, 0, L4_BASE_TASK_CAP);
00257 }
00258
00259 L4_CV L4_INLINE int
00260 l4re_rm_detach_unmap(l4_addr_t addr, l4_cap_idx_t task)
00261 L4_NOTHROW
00262 {
00263 return l4re_rm_detach_srv(l4re_global_env->rm, addr, 0, task);
00264 }
00265
00266 L4_CV L4_INLINE int
00267 l4re_rm_detach_ds(void *addr, l4re_ds_t *ds)
00268 L4_NOTHROW
00269 {
00270 return l4re_rm_detach_srv(l4re_global_env->rm, (l4_addr_t)addr,
00271 ds, L4_BASE_TASK_CAP);
00272 }
00273
00274 L4_CV L4_INLINE int
00275 l4re_rm_detach_ds_unmap(void *addr, l4re_ds_t *ds,
00276 l4_cap_idx_t task) L4_NOTHROW
00277 {
00278 return l4re_rm_detach_srv(l4re_global_env->rm, (l4_addr_t)addr,
00279 ds, task);
00280 }
00281
00282 L4_CV L4_INLINE int
00283 l4re_rm_find(l4_addr_t *addr, unsigned long *size,
00284 l4_addr_t *offset,
00285 unsigned *flags, l4re_ds_t *m) L4_NOTHROW
00286 {
00287 return l4re_rm_find_srv(l4re_global_env->rm, addr, size, offset, flags, m);
00288 }
00289
00290 L4_CV L4_INLINE void
00291 l4re_rm_show_lists(void) L4_NOTHROW
00292 {
00293 l4re_rm_show_lists_srv(l4re_global_env->rm);
00294 }
00295
00296 EXTERN_C_END

```

## 15.165 l4re/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\\_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.*

### 15.165.1 Detailed Description

Capability allocator C interface.

Definition in file [cap\\_alloc.h](#).



## 15.166 cap\_alloc.h

```

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 * 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 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
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 #pragma once
00024
00031 #include <l4/sys/types.h>
00032 #include <l4/sys/linkage.h>
00033
00034 EXTERN_C_BEGIN
00035
00040 L4_CV l4_cap_idx_t
00041 l4re_util_cap_alloc(void) L4_NOTHROW;
00042
00047 L4_CV void
00048 l4re_util_cap_free(l4_cap_idx_t cap) L4_NOTHROW;
00049
00055 L4_CV void
00056 l4re_util_cap_free_um(l4_cap_idx_t cap)
 L4_NOTHROW;
00057
00063 L4_CV long
00064 l4re_util_cap_last(void) L4_NOTHROW;
00065
00066 EXTERN_C_END

```

## 15.167 l4re/c/util/kumem\_alloc.h File Reference

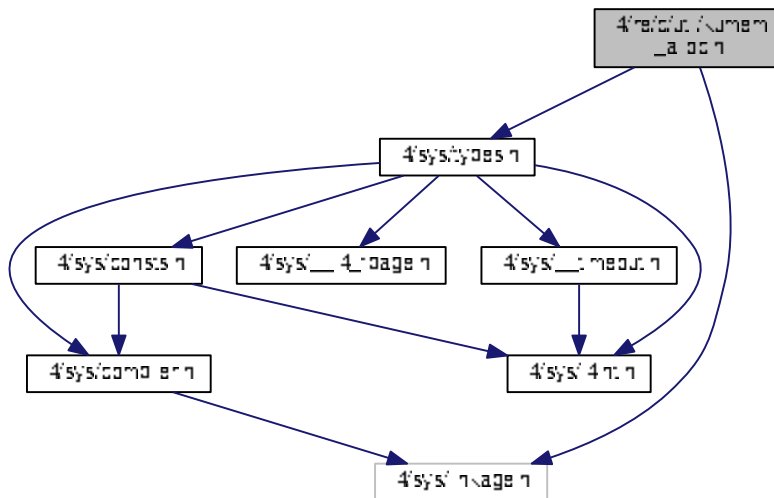
Kumem allocator utility C interface.

```

#include <l4/sys/types.h>
#include <l4/sys/linkage.h>

```

Include dependency graph for `kumem_alloc.h`:



## Functions

- `int l4re_util_kumem_alloc (l4_addr_t *mem, unsigned pages_order, l4_cap_idx_t task, l4_cap_idx_t regmgr)`  
`L4_NOTHROW`  
*Allocate state area.*

### 15.167.1 Detailed Description

Kumem allocator utility C interface.

Definition in file `kumem_alloc.h`.

## 15.168 kumem\_alloc.h

```

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 * 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 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
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 #pragma once
00024

```

```

00031 #include <l4/sys/types.h>
00032 #include <l4/sys/linkage.h>
00033
00034 EXTERN_C_BEGIN
00035
00047 L4_CV int
00048 l4re_util_kumem_alloc(l4_addr_t *mem, unsigned pages_order,
00049 l4_cap_idx_t task, l4_cap_idx_t regmgr)
00049 L4_NOTHROW;
00050
00051 EXTERN_C_END

```

## 15.169 l4/re/c/util/video/goos\_fb.h File Reference

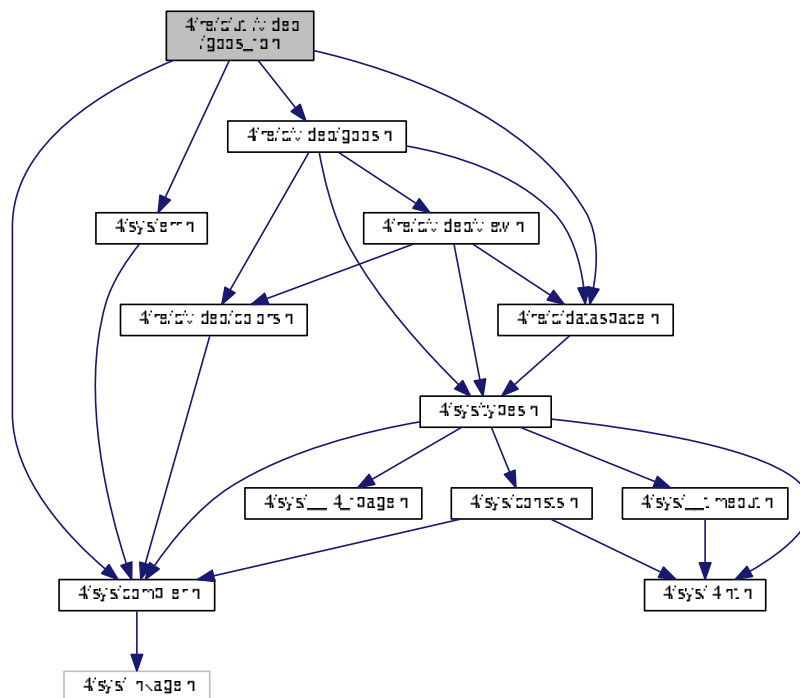
Framebuffer utility functionality.

```

#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:



### 15.169.1 Detailed Description

Framebuffer utility functionality.

Definition in file [goos\\_fb.h](#).

## 15.170 goos\_fb.h

```

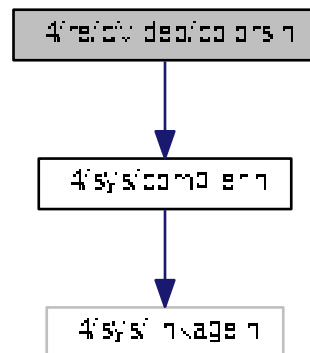
00001
00005 /*
00006 * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007 * economic rights: Technische Universität Dresden (Germany)
00008 *
00009 * This file is part of TUD:OS and distributed under the terms of the
00010 * GNU General Public License 2.
00011 * Please see the COPYING-GPL-2 file for details.
00012 *
00013 * As a special exception, you may use this file as part of a free software
00014 * library without restriction. Specifically, if other files instantiate
00015 * templates or use macros or inline functions from this file, or you compile
00016 * this file and link it with other files to produce an executable, this
00017 * file does not by itself cause the resulting executable to be covered by
00018 * the GNU General Public License. This exception does not however
00019 * invalidate any other reasons why the executable file might be covered by
00020 * the GNU General Public License.
00021 */
00022 #pragma once
00023
00024 #include <l4/sys/compiler.h>
00025 #include <l4/re/c/video/goos.h>
00026 #include <l4/sys/err.h>
00027 #include <l4/re/c/dataspace.h>
00028
00029 EXTERN_C_BEGIN
00030
00031 typedef struct
00032 {
00033 unsigned long _obj_buf[6];
00034 } l4re_util_video_goos_fb_t;
00035
00036 L4_CV int
00037 l4re_util_video_goos_fb_setup_name(l4re_util_video_goos_fb_t *goosfb,
00038 char const *name) L4_NOTHROW;
00039
00040 L4_CV void
00041 l4re_util_video_goos_fb_destroy(l4re_util_video_goos_fb_t *goosfb) L4_NOTHROW;
00042
00043 L4_CV int
00044 l4re_util_video_goos_fb_view_info(l4re_util_video_goos_fb_t *goosfb,
00045 l4re_video_view_info_t *info)
00046 L4_NOTHROW;
00047
00048 L4_CV void *
00049 l4re_util_video_goos_fb_attach_buffer(l4re_util_video_goos_fb_t *goosfb)
00050 L4_NOTHROW;
00051
00052 L4_CV int
00053 l4re_util_video_goos_fb_refresh(l4re_util_video_goos_fb_t *goosfb,
00054 int x, int y, int w, int h) L4_NOTHROW;
00055
00056 L4_CV l4re_ds_t
00057 l4re_util_video_goos_fb_buffer(l4re_util_video_goos_fb_t *goosfb) L4_NOTHROW;
00058
00059 L4_CV l4_cap_idx_t
00060 l4re_util_video_goos_fb_goos(l4re_util_video_goos_fb_t *goosfb) L4_NOTHROW;
00061
00062 EXTERN_C_END

```

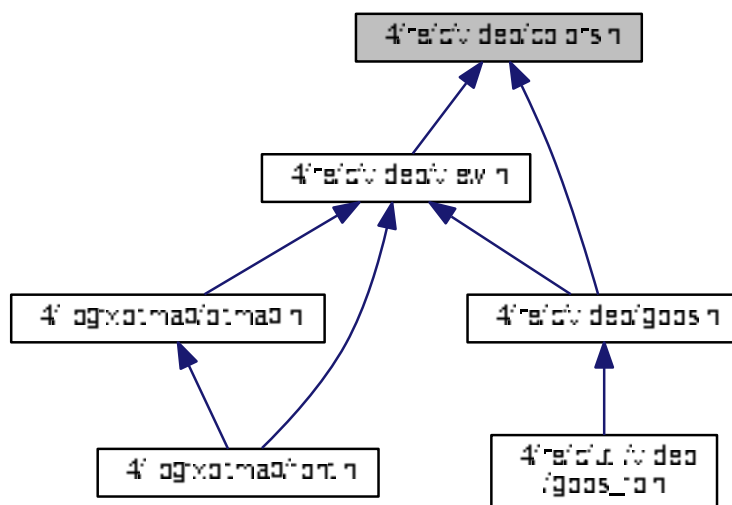
## 15.171 l4/re/c/video/colors.h File Reference

```
#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.*

### 15.171.1 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](#).

### 15.172 colors.h

```

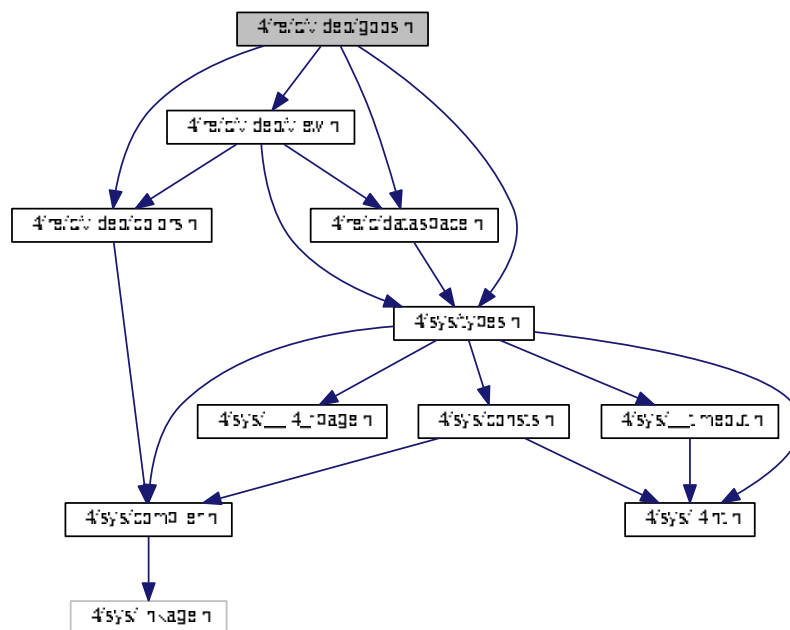
00001
00006 /*
00007 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00008 * economic rights: Technische Universität Dresden (Germany)
00009 *
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU General Public License 2.
00012 * Please see the COPYING-GPL-2 file for details.
00013 *
00014 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
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 #pragma once
00024
00025 #include <l4/sys/compiler.h>
00026
00031 typedef struct l4re_video_color_component_t
00032 {
00033 unsigned char size;
00034 unsigned char shift;
00035 } __attribute__((packed)) l4re_video_color_component_t;
00036
00041 typedef struct l4re_video_pixel_info_t
00042 {
00043 l4re_video_color_component_t r, g, b, a;
00044 unsigned char bytes_per_pixel;
00045 } l4re_video_pixel_info_t;
00046
00047 EXTERN_C_BEGIN
00048
00049 L4_INLINE L4_CV int
00050 l4re_video_bits_per_pixel(l4re_video_pixel_info_t *p)
00051 L4_NOTHROW;
00052
00053 /* ***** */
00054 /* Implementations */
00055
00056 L4_INLINE L4_CV int
00057 l4re_video_bits_per_pixel(l4re_video_pixel_info_t *p) L4_NOTHROW
00058 {
00059 return p->r.size + p->b.size + p->g.size + p->a.size;
00060 }
00061 EXTERN_C_END

```

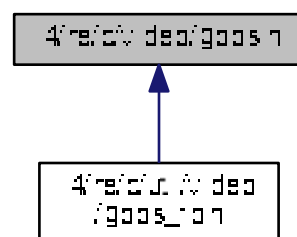
## 15.173 l4/re/c/video/goos.h File Reference

```
#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

- 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.*

### 15.173.1 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](#).



## 15.174 goos.h

```

00001
00006 /*
00007 * (c) 2008-2009 Adam Lackorzynski <adam@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 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
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 #pragma once
00024
00025 #include <l4/sys/types.h>
00026 #include <l4/re/c/dataspace.h>
00027 #include <l4/re/c/video/colors.h>
00028 #include <l4/re/c/video/view.h>
00029
00039 enum l4re_video_goos_info_flags_t
00040 {
00041 F_l4re_video_goos_auto_refresh = 0x01,
00042 F_l4re_video_goos_pointer = 0x02,
00043 F_l4re_video_goos_dynamic_views = 0x04,
00044 F_l4re_video_goos_dynamic_buffers = 0x08,
00045 };
00046
00051 typedef struct
00052 {
00053 unsigned long width;
00054 unsigned long height;
00055 unsigned flags;
00056 unsigned num_static_views;
00057 unsigned num_static_buffers;
00058 l4re_video_pixel_info_t pixel_info;
00059 } l4re_video_goos_info_t;
00060
00065 typedef l4_cap_idx_t l4re_video_goos_t;
00066
00067 EXTERN_C_BEGIN
00068
00080 L4_CV int
00081 l4re_video_goos_info(l4re_video_goos_t goos,
00082 l4re_video_goos_info_t *ginfo)
00083 L4_NOTHROW;
00083
00093 L4_CV int
00094 l4re_video_goos_refresh(l4re_video_goos_t goos, int x, int y, int w
00095 ,
00096 int h) L4_NOTHROW;
00096
00108 L4_CV int
00109 l4re_video_goos_create_buffer(l4re_video_goos_t goos,
00110 unsigned long size,
00111 l4_cap_idx_t buffer) L4_NOTHROW;
00111
00119 L4_CV int
00120 l4re_video_goos_delete_buffer(l4re_video_goos_t goos,
00121 unsigned idx) L4_NOTHROW;
00121
00132 L4_CV int
00133 l4re_video_goos_get_static_buffer(
00134 l4re_video_goos_t goos, unsigned idx,
00135 l4_cap_idx_t buffer) L4_NOTHROW;
00135
00142 L4_CV int
00143 l4re_video_goos_create_view(l4re_video_goos_t goos,
00144 l4re_video_view_t *view) L4_NOTHROW;
00145
00153 L4_CV int
00154 l4re_video_goos_delete_view(l4re_video_goos_t goos,
00155 l4re_video_view_t *view) L4_NOTHROW;
00156
00157
00170 L4_CV int
00171 l4re_video_goos_get_view(l4re_video_goos_t goos, unsigned idx,
00172 l4re_video_view_t *view) L4_NOTHROW;
00172
00173

```

### 15.175 I4/re/c/video/view.h File Reference

Include dependency graph for view.h:



## 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\\_above](#) = 0x01000, [F\\_l4re\\_video\\_view\\_flags\\_mask](#)  
= 0xff000 }  
*Flags of information on a view.*

## Functions

- 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.*

### 15.175.1 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](#).

## 15.176 view.h

```

00001
00006 /*
00007 * (c) 2008-2009 Adam Lackorzynski <adam@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 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
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 #pragma once
00024
00025 #include <l4/sys/types.h>
00026 #include <l4/re/c/dataspace.h>
00027 #include <l4/re/c/video/colors.h>
00028
00033 enum l4re_video_view_info_flags_t
00034 {
00035 F_l4re_video_view_none = 0x00,
00036 F_l4re_video_view_set_buffer = 0x01,
00037 F_l4re_video_view_set_buffer_offset = 0x02,
00038 F_l4re_video_view_set_bytes_per_line = 0x04,
00039 F_l4re_video_view_set_pixel = 0x08,
00040 F_l4re_video_view_set_position = 0x10,
00041 F_l4re_video_view_dyn_allocated = 0x20,
00042 F_l4re_video_view_set_background = 0x40,
00043 F_l4re_video_view_set_flags = 0x80,
00044 F_l4re_video_view_fully_dynamic = F_l4re_video_view_set_buffer
00045 | F_l4re_video_view_set_buffer_offset
00046 | F_l4re_video_view_set_bytes_per_line
00047 | F_l4re_video_view_set_pixel
00048 | F_l4re_video_view_set_position
00049 | F_l4re_video_view_dyn_allocated,
00050
00051 F_l4re_video_view_above = 0x01000,
00052 F_l4re_video_view_flags_mask = 0xff000,
00053 };
00054
00059 typedef struct l4re_video_view_info_t
00060 {
00061 unsigned flags;
00062 unsigned view_index;
00063 unsigned long xpos, ypos, width, height;
00064 unsigned long buffer_offset;
00065 unsigned long bytes_per_line;
00066 l4re_video_pixel_info_t pixel_info;
00067 unsigned buffer_index;
00068 } l4re_video_view_info_t;
00069
00070
00078 typedef struct l4re_video_view_t
00079 {
00080 l4_cap_idx_t goos;
00081 unsigned idx;
00082 } l4re_video_view_t;
00083
00084
00085 EXTERN_C_BEGIN
00086
00096 L4_CV int
00097 l4re_video_view_refresh(l4re_video_view_t *view, int x, int y, int
00098 w,
00099 int h) L4_NOTHROW;
00100
00106 L4_CV int
00107 l4re_video_view_get_info(l4re_video_view_t *view,
00108 l4re_video_view_info_t *info)
00109 L4_NOTHROW;
00110
00120 L4_CV int
00121 l4re_video_view_set_info(l4re_video_view_t *view,
00122 l4re_video_view_info_t *info)
00123 L4_NOTHROW;
00139 L4_CV int
00140 l4re_video_view_set_viewport(l4re_video_view_t *view, int x,

```

```

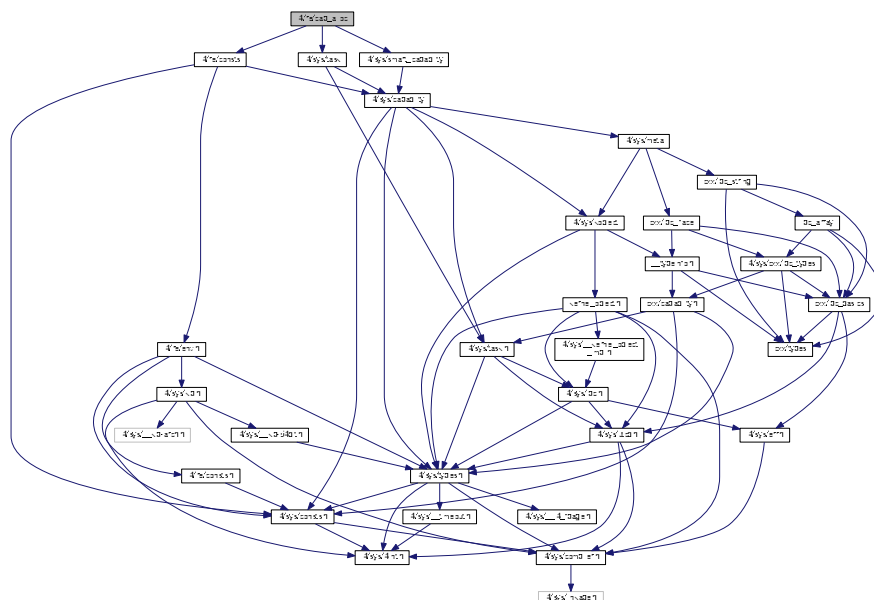
00141 int y, int w,
00142 int h, unsigned long bofs) L4_NOTHROW;
00152 L4_CV int
00153 l4re_video_view_stack(l4re_video_view_t *view,
00154 l4re_video_view_t *pivot,
00155 int behind) L4_NOTHROW;
00156 EXTERN_C_END
00157

```

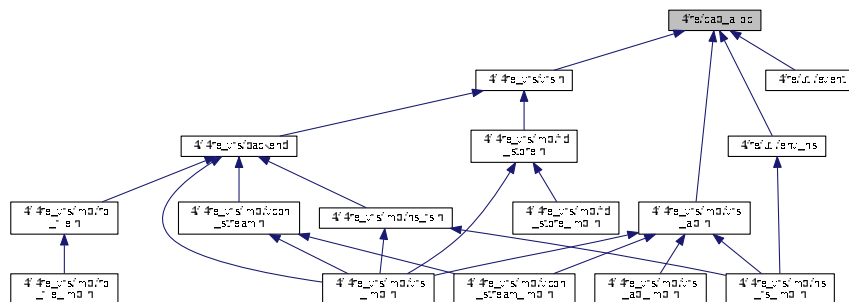
### 15.177 l4/re/cap\_alloc File Reference

### Abstract capability-allocator interface.

```
#include <l4/sys/task>
#include <l4/sys/smart_capability>
#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::Cap\\_alloc](#)  
*Capability allocator interface.*
- class [L4Re::Smart\\_cap\\_auto< Unmap\\_flags >](#)  
*Helper for Auto\_cap and Auto\_del\_cap.*

## Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*

### 15.177.1 Detailed Description

Abstract capability-allocator interface.

Definition in file [cap\\_alloc](#).

## 15.178 cap\_alloc

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007 * (c) 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 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024
00025 #pragma once
00026
00027 #include <l4/sys/task>
00028 #include <l4/sys/smart_capability>
00029 #include <l4/re/consts>
00030
00031 namespace L4Re {
00032
00040 class Cap_alloc
00041 {
00042 private:
00043 void operator = (Cap_alloc const &);
00044
00045 protected:
00046 Cap_alloc(Cap_alloc const &) {}
00047 Cap_alloc() {}
00048
00049 public:
00050
00055 virtual L4::Cap<void> alloc() throw() = 0;
00056
00061 template< typename T >
00062 L4::Cap<T> alloc() throw()
00063 { return L4::cap_cast<T>(alloc()); }
00064
00069 virtual void free(L4::Cap<void> cap) throw() = 0;
00070
00071 //virtual L4::Cap<void> next_allocated(l4_umword_t *refs, L4::Cap<void> pivot = L4::Cap<void>::Invalid)

```

```

 throw() = 0;
00072
00076 virtual ~Cap_alloc() = 0;
00077
00083 template< typename CAP_ALLOC >
00084 static inline L4Re::Cap_alloc *
00085 get_cap_alloc(CAP_ALLOC &ca)
00086 {
00087 struct CA : public L4Re::Cap_alloc
00088 {
00089 CAP_ALLOC &_ca;
00090 L4::Cap<void> alloc() throw() { return _ca.alloc(); }
00091 void free(L4::Cap<void> cap) throw() { _ca.free(cap); }
00092 CA(CAP_ALLOC &ca) : _ca(ca) {}
00093 };
00094
00095 static CA _ca(ca);
00096 return &_ca;
00097 }
00098 };
00099
00100 inline
00101 Cap_alloc::~Cap_alloc()
00102 {}
00103
00104
00109 template< unsigned long Unmap_flags = L4_FP_ALL_SPACES >
00110 class Smart_cap_auto
00111 {
00112 private:
00113 Cap_alloc *_ca;
00114
00115 public:
00116 Smart_cap_auto() : _ca(0) {}
00117 Smart_cap_auto(Cap_alloc *ca) : _ca(ca) {}
00118
00119 void free(L4::Cap_base &c)
00120 {
00121 if (c.is_valid() && _ca)
00122 {
00123 L4::Cap<L4::Task>(This_task)->unmap(c.fpage(), Unmap_flags);
00124 _ca->free(L4::Cap<void>(c.cap()));
00125 }
00126 invalidate(c);
00127 }
00128
00129 static void invalidate(L4::Cap_base &c)
00130 {
00131 if (c.is_valid())
00132 c.invalidate();
00133 }
00134
00135 static L4::Cap_base copy(L4::Cap_base const &src)
00136 {
00137 L4::Cap_base r = src;
00138 invalidate(const_cast<L4::Cap_base &>(src));
00139 return r;
00140 }
00141 };
00142
00168 template< typename T >
00169 struct Auto_cap
00170 {
00171 typedef L4::Smart_cap<T, Smart_cap_auto<> > Cap;
00172 };
00173
00203 template< typename T >
00204 struct Auto_del_cap
00205 {
00206 typedef L4::Smart_cap<T, Smart_cap_auto<L4_FP_DELETE_OBJ>
00207 > Cap;
00208 };
00210 }

```

## 15.179 l4/re/util/cap\_alloc File Reference

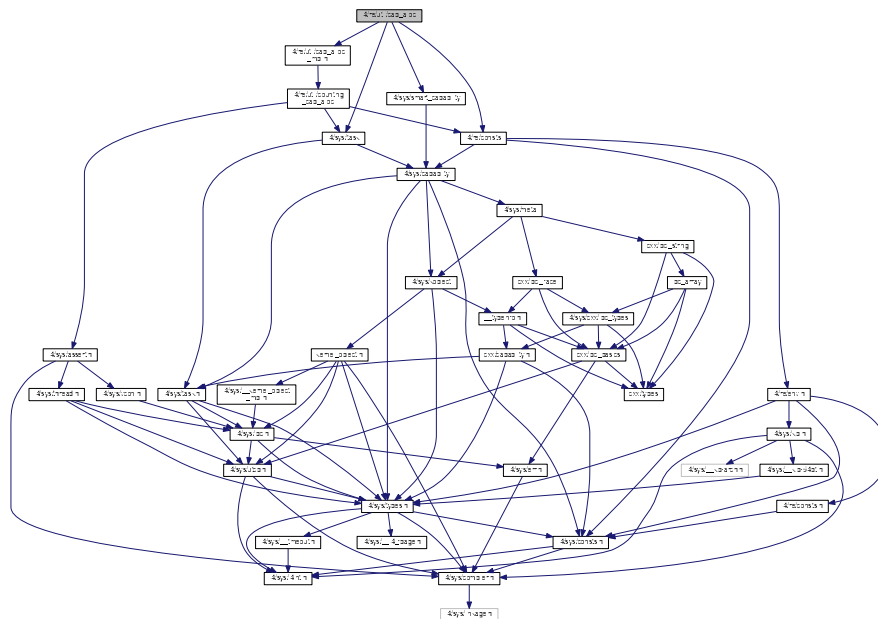
Capability allocator.

```

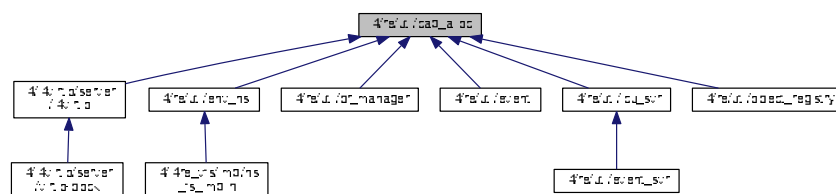
#include <l4/re/util/cap_alloc_impl.h>
#include <l4/sys/smart_capability>

```

```
#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 [Auto\\_cap](#) and [Auto\\_del\\_cap](#).
- class [L4Re::Util::Smart\\_count\\_cap< Unmap\\_flags >](#)  
Helper for [Ref\\_cap](#) and [Ref\\_del\\_cap](#).
- struct [L4Re::Util::Auto\\_cap< T >](#)  
Automatic capability that implements automatic free and unmap of the capability selector.
- struct [L4Re::Util::Auto\\_del\\_cap< T >](#)  
Automatic capability that implements automatic free and unmap+delete of the capability selector.
- 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

- [L4Re](#)  
*L4Re C++ Interfaces.*

## Functions

- `template<typename T>`  
`Auto_cap< T >::Cap L4Re::Util::make\_auto\_cap ()`  
*Allocate a capability slot and wrap it in an [Auto\\_cap](#).*
- `template<typename T>`  
`Auto_del_cap< T >::Cap L4Re::Util::make\_auto\_del\_cap ()`  
*Allocate a capability slot and wrap it in an [Auto\\_del\\_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<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.*

### 15.179.1 Detailed Description

Capability allocator.

Definition in file [cap\\_alloc](#).

## 15.180 cap\_alloc

```
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 * This file is part of TUD:OS and distributed under the terms of the
00009 * GNU General Public License 2.
00010 * Please see the COPYING-GPL-2 file for details.
00011 *
00012 * As a special exception, you may use this file as part of a free software
00013 * library without restriction. Specifically, if other files instantiate
00014 * templates or use macros or inline functions from this file, or you compile
00015 * this file and link it with other files to produce an executable, this
00016 * file does not by itself cause the resulting executable to be covered by
00017 * the GNU General Public License. This exception does not however
00018 * invalidate any other reasons why the executable file might be covered by
00019 * the GNU General Public License.
00020 */
00021 #pragma once
00022 #include <l4/re/util/cap_alloc_impl.h>
00023 #include <l4/sys/smart_capability>
```

```

00030 #include <l4/sys/task>
00031 #include <l4/re/consts>
00032
00033 namespace L4Re { namespace Util {
00034
00053 extern _Cap_alloc &cap_alloc;
00054
00058 template< unsigned long Unmap_flags = L4_FP_ALL_SPACES >
00059 class Smart_cap_auto
00060 {
00061 public:
00065 static void free(L4::Cap_base &c)
00066 {
00067 if (c.is_valid())
00068 {
00069 cap_alloc.free(L4::Cap<void>(c.cap()), This_task, Unmap_flags);
00070 c.invalidate();
00071 }
00072 }
00073
00077 static void invalidate(L4::Cap_base &c)
00078 {
00079 if (c.is_valid())
00080 c.invalidate();
00081 }
00082
00086 static L4::Cap_base copy(L4::Cap_base const &src)
00087 {
00088 L4::Cap_base r = src;
00089 invalidate(const_cast<L4::Cap_base &>(src));
00090 return r;
00091 }
00092 };
00093
00094
00098 template< unsigned long Unmap_flags = L4_FP_ALL_SPACES >
00099 class Smart_count_cap
00100 {
00101 public:
00106 static void free(L4::Cap_base &c) throw()
00107 {
00108 if (c.is_valid())
00109 {
00110 if (cap_alloc.release(L4::Cap<void>(c.cap()), This_task, Unmap_flags))
00111 c.invalidate();
00112 }
00113 }
00114
00118 static void invalidate(L4::Cap_base &c) throw()
00119 {
00120 if (c.is_valid())
00121 c.invalidate();
00122 }
00123
00127 static L4::Cap_base copy(L4::Cap_base const &src)
00128 {
00129 cap_alloc.take(L4::Cap<void>(src.cap()));
00130 return src;
00131 }
00132 };
00133
00134
00160 template< typename T >
00161 struct Auto_cap
00162 {
00163 typedef L4::Smart_cap<T, Smart_cap_auto< L4_FP_ALL_SPACES>
00164 > Cap;
00165 };
00166
00195 template< typename T >
00196 struct Auto_del_cap
00197 {
00198 typedef L4::Smart_cap<T, Smart_cap_auto<L4_FP_DELETE_OBJ>
00199 > Cap;
00200 };
00201
00230 template< typename T >
00231 struct Ref_cap
00232 {
00233 typedef L4::Smart_cap<T, Smart_count_cap<L4_FP_ALL_SPACES>
00234 > Cap;
00235 };
00236
00271 template< typename T >
00272 struct Ref_del_cap
00273 {
00274 typedef L4::Smart_cap<T, Smart_count_cap<L4_FP_DELETE_OBJ>

```

```

> Cap;
00275 };
00276
00282 template< typename T >
00283 typename Auto_cap<T>::Cap
00284 make_auto_cap()
00285 { return typename Auto_cap<T>::Cap(cap_alloc.alloc<T>()); }
00286
00292 template< typename T >
00293 typename Auto_del_cap<T>::Cap
00294 make_auto_del_cap()
00295 { return typename Auto_del_cap<T>::Cap(cap_alloc.alloc<T>()); }
00296
00302 template< typename T >
00303 typename Ref_cap<T>::Cap
00304 make_ref_cap() { return typename Ref_cap<T>::Cap(cap_alloc.alloc<T>()); }
00305
00311 template< typename T >
00312 typename Ref_del_cap<T>::Cap
00313 make_ref_del_cap()
00314 { return typename Ref_del_cap<T>::Cap(cap_alloc.alloc<T>()); }
00315
00318 }}
00319

```

## 15.181 l4/re/consts File Reference

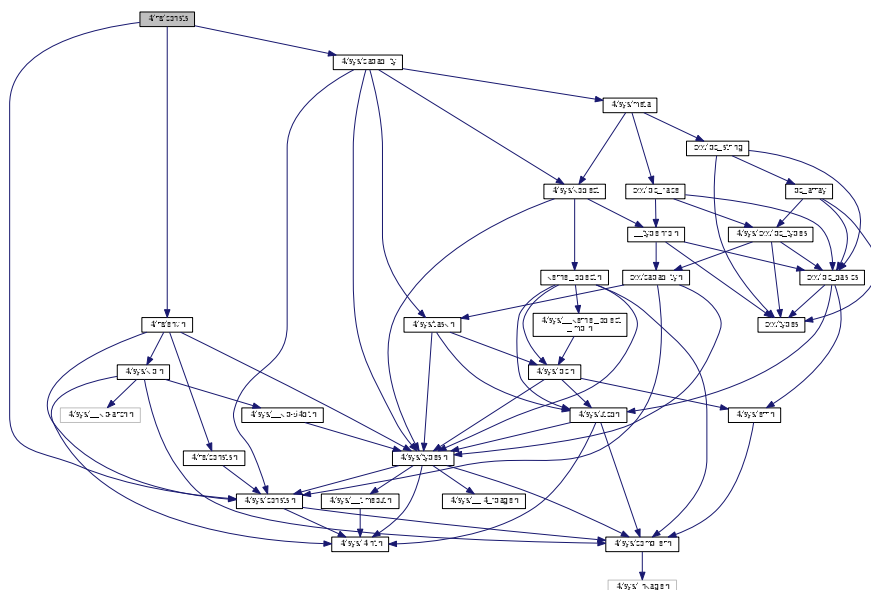
Constants.

```

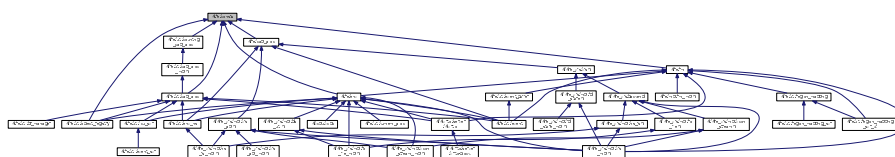
#include <l4/sys/capability>
#include <l4/sys/consts.h>
#include <l4/re/env.h>

```

Include dependency graph for consts:



This graph shows which files directly or indirectly include this file:



## Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*

### 15.181.1 Detailed Description

Constants.

Definition in file [consts](#).

## 15.182 consts

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008 * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00009 * economic rights: Technische Universität Dresden (Germany)
00010 *
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024 #pragma once
00025
00026 #include <l4/sys/capability>
00027 #include <l4/sys/consts.h>
00028 #include <l4/re/env.h>
00029
00030 namespace L4Re {
00031 static L4::Cap<L4::Task>::Cap_type const This_task
00032 = (L4::Cap<L4::Task>::Cap_type) (L4RE_THIS_TASK_CAP);
00033 }

```

## 15.183 l4/re/dataspace File Reference

Dataspace interface.

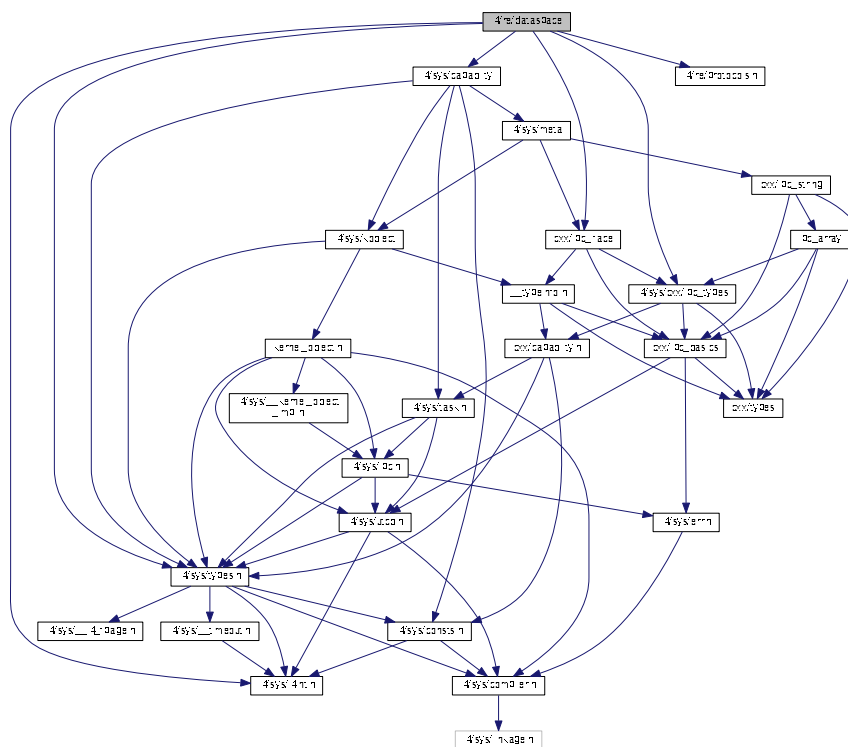
```

#include <l4/sys/types.h>
#include <l4/sys/l4int.h>
#include <l4/sys/capability>
#include <l4/re/protocols.h>
#include <l4/sys/cxx/ipc_types>

```

```
#include <l4/sys/cxx/ipc_iface>
```

Include dependency graph for dataspace:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class `L4Re::Dataspace`  
*Interface for memory-like objects.*
- struct `L4Re::Dataspace::Stats`  
*Information about the dataspace.*

## Namespaces

- L4Re
- L4Re C++ Interfaces.*

### 15.183.1 Detailed Description

Dataspace interface.

Definition in file [dataspace](#).

## 15.184 dataspace

```

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 * Björn Döbel <doebel@os.inf.tu-dresden.de>,
00007 * Torsten Frenzel <frenzel@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 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
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 #pragma once
00024 #include <l4/sys/types.h>
00025 #include <l4/sys/l4int.h>
00026 #include <l4/sys/capability>
00027 #include <l4/re/protocols.h>
00028 #include <l4/sys/cxx/ipc_types>
00029 #include <l4/sys/cxx/ipc_iface>
00030 namespace L4Re
00031 {
00032 // MISSING:
00033 // * size support in map, mapped size in reply
00034
00035 class L4_EXPORT Dataspace :
00036 public L4::Kobject<Dataspace, L4::Kobject, L4RE_PROTO_DATASPACE,
00037 L4::Type_info::Demand_t<1> >
00038 {
00039 public:
00040 enum Map_flags
00041 {
00042 Map_ro = 0,
00043 Map_rw = 1,
00044 Map_normal = 0x00,
00045 Map_cacheable = Map_normal,
00046 Map_bufferable = 0x10,
00047 Map_uncacheable = 0x20,
00048 Map_caching_mask = 0x30,
00049 Map_caching_shift = 4,
00050 };
00051 struct Stats {
00052 unsigned long size;
00053 unsigned long flags;
00054 };
00055
00056 long map(l4_addr_t offset, unsigned long flags, l4_addr_t local_addr,
00057 l4_addr_t min_addr, l4_addr_t max_addr) const throw();
00058
00059 long map_region(l4_addr_t offset, unsigned long flags,
00060 l4_addr_t min_addr, l4_addr_t max_addr) const throw();
00061
00062 L4_RPC(long, clear, (l4_addr_t offset, unsigned long size));
00063
00064 L4_RPC(long, allocate, (l4_addr_t offset, l4_size_t size));
00065
00066 L4_RPC(long, copy_in, (l4_addr_t dst_offs,
00067 L4::Ipc::Cap<Dataspace> src,
00068 l4_addr_t src_offs, unsigned long size));
00069
00070 L4_RPC(long, phys, (l4_addr_t offset, l4_addr_t &phys_addr,
00071 l4_size_t &phys_size));
00072
00073 unsigned long size() const throw();
00074 };
00075 }

```

```

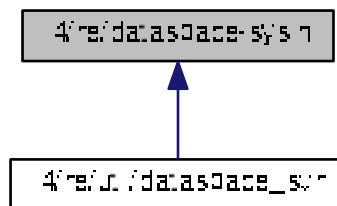
00229 long flags() const throw();
00230
00239 L4_RPC(long, info, (Stats *stats));
00240
00250 L4_RPC(long, take, ());
00251
00261 L4_RPC(long, release, ());
00262
00263 L4_RPC_NF(long, map, (unsigned long offset, l4_addr_t spot,
00264 unsigned long flags, L4::Ipc::Rcv_fpage r,
00265 L4::Ipc::Snd_fpage &fp));
00266 private:
00267
00268 long __map(l4_addr_t offset, unsigned char *size, unsigned long flags,
00269 l4_addr_t local_addr) const throw();
00270
00271 public:
00272 typedef L4::Typeid::Rpc<map_t, clear_t, info_t, copy_in_t, take_t,
00273 release_t, phys_t, allocate_t> Rpc;
00274
00275 };
00276
00277
00278 }
00279

```

## 15.185 l4/re/dataspace-sys.h File Reference

Dataspace protocol definition.

This graph shows which files directly or indirectly include this file:



### Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*

### Enumerations

- enum [L4Re::Dataspace\\_::Opcodes](#)  
*Data-space communication-protocol opcodes.*

#### 15.185.1 Detailed Description

Dataspace protocol definition.

Definition in file [dataspace-sys.h](#).

## 15.186 dataspace-sys.h

```

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 * 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 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
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 #pragma once
00024
00025 namespace L4Re
00026 {
00027 namespace Dataspace_
00028 {
00034 enum Opcodes { Map, Clear, Stats, Copy, Take, Release, Phys, Allocate };
00035 };
00036 };
00037

```

## 15.187 l4/re/debug File Reference

Debug interface.

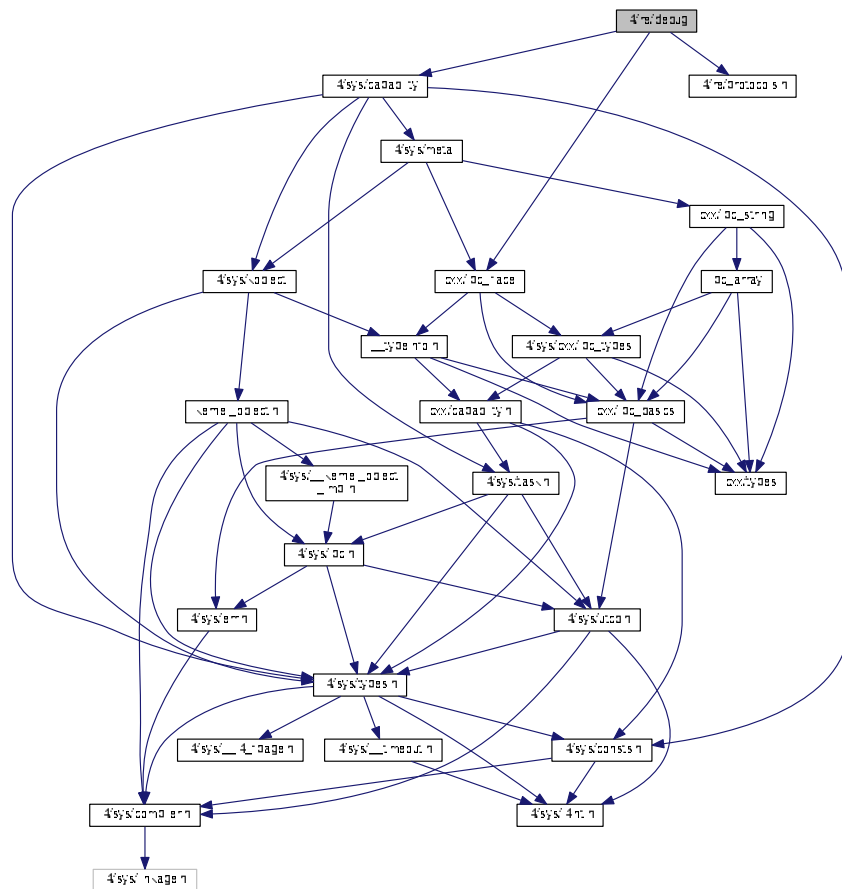
```

#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

- [L4Re](#)  
*L4Re C++ Interfaces.*

## 15.187.1 Detailed Description

Debug interface.

Definition in file [debug](#).

## 15.188 debug

```

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 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024 #pragma once
00025
00026 #include <l4/sys/capability>
00027 #include <l4/re/protocols.h>
00028 #include <l4/sys/cxx/ipc_iface>
00029
00030 namespace L4Re {
00051 class L4_EXPORT Debug_obj :
00052 public L4::Kobject_t<Debug_obj, L4::Kobject, L4RE_PROTO_DEBUG>
00053 {
00054 public:
00055
00063 L4_INLINE_RPC(long, debug, (unsigned long function));
00064 typedef L4::Typeid::Rpc_nocode<debug_t> Rpccs;
00065 };
00066
00067 template<typename BASE>
00068 class Debug_obj_t :
00069 public L4::Kobject_2t<Debug_obj_t<BASE>, BASE, Debug_obj, L4::PROTO_EMPTY>
00070 {
00071 typedef L4::Typeid::Rpccs<> Rpccs;
00072 };
00073 }

```

## 15.189 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>

```



```

00021 */
00022
00023 #pragma once
00024
00025 #include <l4/sys/types.h>
00026 #include <l4/sys/l4int.h>
00027 #include <l4/sys/capability>
00028 #include <l4/re/dataspace>
00029 #include <l4/re/protocols.h>
00030 #include <l4/sys/cxx/types>
00031 #include <l4/sys/cxx/ipc_types>
00032 #include <l4/sys/cxx/ipc_iface>
00033
00034 namespace L4Re
00035 {
00036
00061 class Dma_space :
00062 public L4::Kobject_0t< Dma_space,
00063 L4RE_PROTO_DMA_SPACE,
00064 L4::Type_info::Demand_t<1> >
00065 {
00066 public:
00068 typedef l4_uint64_t Dma_addr;
00069
00073 enum Direction
00074 {
00075 Bidirectional,
00076 To_device,
00077 From_device,
00078 None
00079 };
00080
00085 enum Attribute
00086 {
00098 No_sync
00099 };
00100
00106 typedef L4::Types::Flags<Attribute> Attributes;
00107
00113 enum Space_attrib
00114 {
00121 Coherent,
00122
00127 Phys_space
00128 };
00129
00131 typedef L4::Types::Flags<Space_attrib>
Space_attribs;
00132
00157 L4_INLINE_RPC(
00158 long, map, (L4::Ipc::Cap<L4Re::Dataspace> src,
00159 l4_addr_t offset,
00160 L4::Ipc::In_out<l4_size_t *> size,
00161 Attributes attrs, Direction dir,
00162 Dma_addr *dma_addr));
00171 L4_INLINE_RPC(
00172 long, unmap, (Dma_addr dma_addr,
00173 l4_size_t size, Attributes attrs, Direction dir));
00186 L4_INLINE_RPC(
00187 long, associate, (L4::Ipc::Opt<L4::Ipc::Cap<L4::Task> >
00188 dma_task,
00189 Space_attribs attr),
00189 L4::Ipc::Call_t<L4_CAP_FPAGE_RW>);
00190
00196 L4_INLINE_RPC(
00197 long, disassociate, (),
00198 L4::Ipc::Call_t<L4_CAP_FPAGE_RW>);
00199
00200 typedef L4::Typeid::Rpcs<map_t, unmap_t, associate_t, disassociate_t>
Rpcs;
00201 };
00202
00203 }
00204

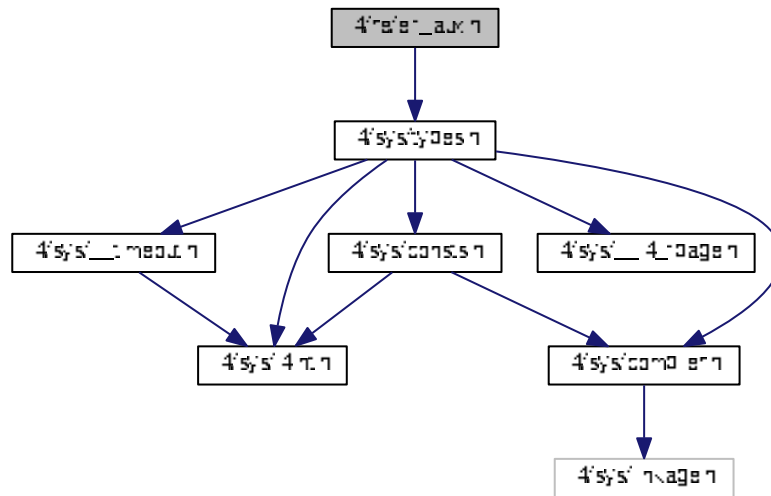
```

## 15.191 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(".ro.l4re\_elf\_aux"), aligned(sizeof([l4re\\_elf\\_aux\\_mword\\_t](#)))))  
Define an auxiliary vector element.
- #define [L4RE\\_ELF\\_AUX\\_ELEM\\_T](#)(type, id, tag, val...) static [L4RE\\_ELF\\_AUX\\_ELEM](#) type id = {tag, sizeof(type), 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](#) }

### 15.191.1 Detailed Description

Auxiliary information for binaries.

Definition in file [elf\\_aux.h](#).

## 15.192 elf\_aux.h

```

00001
00005 /*
00006 * (c) 2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00007 * economic rights: Technische Universität Dresden (Germany)
00008 *
00009 * This file is part of TUD:OS and distributed under the terms of the
00010 * GNU General Public License 2.
00011 * Please see the COPYING-GPL-2 file for details.
00012 *
00013 * As a special exception, you may use this file as part of a free software
00014 * library without restriction. Specifically, if other files instantiate
00015 * templates or use macros or inline functions from this file, or you compile
00016 * this file and link it with other files to produce an executable, this
00017 * file does not by itself cause the resulting executable to be covered by
00018 * the GNU General Public License. This exception does not however
00019 * invalidate any other reasons why the executable file might be covered by
00020 * the GNU General Public License.
00021 */
00022 #pragma once
00023
00024 #include <l4/sys/types.h>
00025
00026
00039
00052 #define L4RE_ELF_AUX_ELEM const __attribute__((used, section(".rol4re_elf_aux"),
 aligned(sizeof(l4_umword_t))))
00053
00067 #define L4RE_ELF_AUX_ELEM_T(type, id, tag, val...) \
00068 static L4RE_ELF_AUX_ELEM type id = {tag, sizeof(type), val}
00069
00070 enum
00071 {
00075 L4RE_ELF_AUX_T_NONE = 0,
00076
00080 L4RE_ELF_AUX_T_VMA,
00081
00086 L4RE_ELF_AUX_T_STACK_SIZE,
00087
00092 L4RE_ELF_AUX_T_STACK_ADDR,
00093
00098 L4RE_ELF_AUX_T_KIP_ADDR,
00099 };
00100
00104 typedef struct l4re_elf_aux_t
00105 {
00106 l4_umword_t type;
00107 l4_umword_t length;
00108 } l4re_elf_aux_t;
00109
00113 typedef struct l4re_elf_aux_vma_t
00114 {
00115 l4_umword_t type;
00116 l4_umword_t length;
00117 l4_umword_t start;
00118 l4_umword_t end;
00119 } l4re_elf_aux_vma_t;
00120
00124 typedef struct l4re_elf_aux_mword_t
00125 {
00126 l4_umword_t type;
00127 l4_umword_t length;
00128 l4_umword_t value;
00129 } l4re_elf_aux_mword_t;
00130

```



### 15.193.1 Detailed Description

Environment interface.

Definition in file [env](#).

## 15.194 env

```

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 * This file is part of TUD:OS and distributed under the terms of the
00014 * GNU General Public License 2.
00015 * Please see the COPYING-GPL-2 file for details.
00016 *
00017 * As a special exception, you may use this file as part of a free software
00018 * library without restriction. Specifically, if other files instantiate
00019 * templates or use macros or inline functions from this file, or you compile
00020 * this file and link it with other files to produce an executable, this
00021 * file does not by itself cause the resulting executable to be covered by
00022 * the GNU General Public License. This exception does not however
00023 * invalidate any other reasons why the executable file might be covered by
00024 * the GNU General Public License.
00025 */
00026 #pragma once
00027
00028 #include <l4/sys/types.h>
00029
00030 #include <l4/re/rm>
00031 #include <l4/re/parent>
00032 #include <l4/re/mem_alloc>
00033 #include <l4/re/log>
00034 #include <l4/re/consts>
00035
00036 #include <l4/re/env.h>
00037
00038 namespace L4 {
00039 class Scheduler;
00040 }
00041
00042 namespace L4Re
00043 {
00082 class L4_EXPORT Env
00083 {
00084 private:
00085 l4re_env_t _env;
00086 public:
00087
00091 typedef l4re_env_cap_entry_t Cap_entry;
00092
00093 static Env const *env() throw()
00100 { return reinterpret_cast<Env*>(l4re_global_env); }
00101
00102 L4::Cap<Parent> parent() const throw()
00107 { return L4::Cap<Parent>(_env.parent); }
00108
00113 L4::Cap<Mem_alloc> mem_alloc() const throw()
00114 { return L4::Cap<Mem_alloc>(_env.mem_alloc); }
00115
00118 L4::Cap<L4::Factory> user_factory() const throw()
00119 { return L4::Cap<L4::Factory>(_env.mem_alloc); }
00120
00124 L4::Cap<Rm> rm() const throw()
00125 { return L4::Cap<Rm>(_env.rm); }
00126
00130 L4::Cap<Log> log() const throw()
00131 { return L4::Cap<Log>(_env.log); }
00132
00136 L4::Cap<L4::Thread> main_thread() const throw()
00137 { return L4::Cap<L4::Thread>(_env.main_thread); }
00138
00142 L4::Cap<L4::Task> task() const throw()
00143 { return L4::Cap<L4::Task>(L4RE_THIS_TASK_CAP); }
00144
00148 L4::Cap<L4::Factory> factory() const throw()
00149 { return L4::Cap<L4::Factory>(_env.factory); }
00150
00156 l4_cap_idx_t first_free_cap() const throw()
00157 { return _env.first_free_cap; }
00158
00162 l4_fpage_t utcb_area() const throw()
00163 { return _env.utcb_area; }
00164
00171 l4_addr_t first_free_utcb() const throw()

```



```

00172 { return _env.first_free_utcb; }
00173
00178 Cap_entry const *initial_caps() const throw()
00179 { return _env.caps; }
00180
00189 Cap_entry const *get(char const *name, unsigned l) const throw()
00190 { return l4re_env_get_cap_l(name, l, &_env); }
00191
00200 template< typename T >
00201 L4::Cap<T> get_cap(char const *name, unsigned l) const throw()
00202 {
00203 if (Cap_entry const *e = get(name, l))
00204 return L4::Cap<T>(e->cap);
00205
00206 return L4::Cap<T>(-L4_ENOENT);
00207 }
00208
00215 template< typename T >
00216 L4::Cap<T> get_cap(char const *name) const throw()
00217 { return get_cap<T>(name, __builtin_strlen(name)); }
00218
00223 void parent(L4::Cap<Parent> const &c) throw()
00224 { _env.parent = c.cap(); }
00229 void mem_alloc(L4::Cap<Mem_alloc> const &c) throw()
00230 { _env.mem_alloc = c.cap(); }
00235 void rm(L4::Cap<Rm> const &c) throw()
00236 { _env.rm = c.cap(); }
00241 void log(L4::Cap<Log> const &c) throw()
00242 { _env.log = c.cap(); }
00247 void main_thread(L4::Cap<L4::Thread> const &c) throw()
00248 { _env.main_thread = c.cap(); }
00253 void factory(L4::Cap<L4::Factory> const &c) throw()
00254 { _env.factory = c.cap(); }
00259 void first_free_cap(l4_cap_idx_t c) throw()
00260 { _env.first_free_cap = c; }
00265 void utcb_area(l4_fpage_t utcbs) throw()
00266 { _env.utcb_area = utcbs; }
00271 void first_free_utcb(l4_addr_t u) throw()
00272 { _env.first_free_utcb = u; }
00273
00279 L4::Cap<L4::Scheduler> scheduler() const throw()
00280 { return L4::Cap<L4::Scheduler>(_env.scheduler); }
00281
00286 void scheduler(L4::Cap<L4::Scheduler> const &c) throw()
00287 { _env.scheduler = c.cap(); }
00288
00293 void initial_caps(Cap_entry *first) throw()
00294 { _env.caps = first; }
00295 };
00296 ;

```

## 15.195 l4/re/env.h File Reference

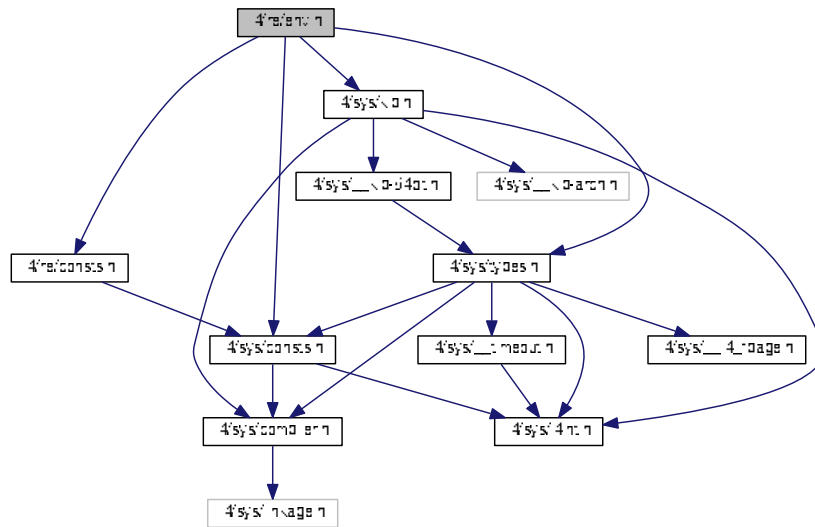
Environment interface.

```

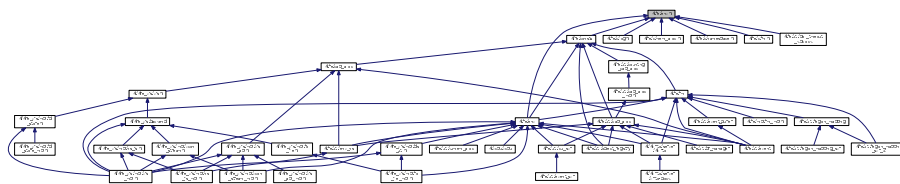
#include <l4/sys/consts.h>
#include <l4/sys/types.h>
#include <l4/sys/kip.h>
#include <l4/re/consts.h>

```

Include dependency graph for env.h:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct `l4re_env_cap_entry_t`  
*Entry in the L4Re environment array for the named initial objects.*
- struct `l4re_env_t`  
*Initial environment data structure.*

## Typedefs

- typedef struct `l4re_env_cap_entry_t` `l4re_env_cap_entry_t`  
*Entry in the `L4Re` environment array for the named initial objects.*
- typedef struct `l4re_env_t` `l4re_env_t`  
*Initial environment data structure.*

## Functions

- `l4re_env_t * l4re_env (void)` `L4_NOTHROW`  
*Get L4Re initial environment.*
- `l4_kernel_info_t * 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.*

### 15.195.1 Detailed Description

Environment interface.

Definition in file [env.h](#).

### 15.195.2 Typedef Documentation

#### 15.195.2.1 l4re\_env\_t

```
typedef struct l4re_env_t l4re_env_t
```

Initial environment data structure.

See also

[Initial environment](#)

## 15.196 env.h

```
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 * 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 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
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 #pragma once
00024
00025 #include <l4/sys/consts.h>
00026 #include <l4/sys/types.h>
00027 #include <l4/sys/kip.h>
00028
00029 #include <l4/re/consts.h>
00030
00046 typedef struct l4re_env_cap_entry_t
00047 {
00051 l4_cap_idx_t cap;
00052
00057 l4_umword_t flags;
00058
00062 char name[16];
00063 #ifdef __cplusplus
00064 l4re_env_cap_entry_t() : cap(L4_INVALID_CAP), flags(~0) {}
00069
00077 l4re_env_cap_entry_t(char const *n, l4_cap_idx_t c,
00078 l4_umword_t f = 0)
00079 : cap(c), flags(f)
00080 {
00081 for (unsigned i = 0; n && i < sizeof(name); ++i, ++n)
00082 name[i] = *n;
00083 if (!*n)
00084 break;
00085 }
00086 }
00087
00088 static bool is_valid_name(char const *n)
00089 {
00090 for (unsigned i = 0; *n; ++i, ++n)
00091 if (i > sizeof(name))
00092 return false;
00093 return true;
00094 }
00095 #endif
00096 } l4re_env_cap_entry_t;
00097
00098
00099
00105 typedef struct l4re_env_t
00106 {
00107 l4_cap_idx_t parent;
00108 l4_cap_idx_t rm;
00109 l4_cap_idx_t mem_alloc;
00110 l4_cap_idx_t log;
00111 l4_cap_idx_t main_thread;
00112 l4_cap_idx_t factory;
00113 l4_cap_idx_t scheduler;
00114 l4_cap_idx_t first_free_cap;
00115 l4_fpage_t utcb_area;
00116 l4_addr_t first_free_utcb;
00117 l4re_env_cap_entry_t *caps;
00118 } l4re_env_t;
00119
00125 extern l4re_env_t *l4re_global_env;
00126
00127
00133 L4_INLINE l4re_env_t *l4re_env(void) L4_NOTHROW;
00134
00135 /*
00136 * FIXME: this seems to be at the wrong place here
00137 */
00143 L4_INLINE l4_kernel_info_t *l4re_kip(void) L4_NOTHROW;
00144
00145
00153 L4_INLINE l4_cap_idx_t
00154 l4re_env_get_cap(char const *name) L4_NOTHROW;
00155
00164 L4_INLINE l4_cap_idx_t
00165 l4re_env_get_cap_e(char const *name, l4re_env_t const *e)
00166 L4_NOTHROW;
00177 L4_INLINE l4re_env_cap_entry_t const *
00178 l4re_env_get_cap_l(char const *name, unsigned l,
00179 l4re_env_t const *e) L4_NOTHROW;
00179
00180 L4_INLINE
00181 l4re_env_t *l4re_env() L4_NOTHROW
00182 { return l4re_global_env; }
00183
00184 L4_INLINE
00185 l4_kernel_info_t *l4re_kip() L4_NOTHROW
00186 {

```

```

00187 extern char __L4_KIP_ADDR__[];
00188 return (l4_kernel_info_t *)__L4_KIP_ADDR__;
00189 }
00190
00191 L4_INLINE l4re_env_cap_entry_t const *
00192 l4re_env_get_cap_l(char const *name, unsigned l,
00193 l4re_env_t const *e) L4_NOTHROW
00194 {
00195 l4re_env_cap_entry_t const *c = e->caps;
00196 for (; c && c->flags != ~0UL; ++c)
00197 {
00198 unsigned i;
00199 for (i = 0;
00200 i < sizeof(c->name) && i < l && c->name[i] && name[i] &&
00201 name[i] == c->name[i];
00202 ++i)
00203 if (i == l && (i == sizeof(c->name) || !c->name[i]))
00204 return c;
00205 }
00206 return NULL;
00207 }
00208
00209 L4_INLINE l4_cap_idx_t
00210 l4re_env_get_cap_e(char const *name, l4re_env_t const *e)
00211 L4_NOTHROW
00212 {
00213 unsigned l;
00214 l4re_env_cap_entry_t const *r;
00215 for (l = 0; name[l]; ++l) ;
00216 r = l4re_env_get_cap_l(name, l, e);
00217 if (r)
00218 return r->cap;
00219 return L4_INVALID_CAP;
00220 }
00221
00222 L4_INLINE l4_cap_idx_t
00223 l4re_env_get_cap(char const *name) L4_NOTHROW
00224 { return l4re_env_get_cap_e(name, l4re_env()); }
00225

```

## 15.197 l4/re/error\_helper File Reference

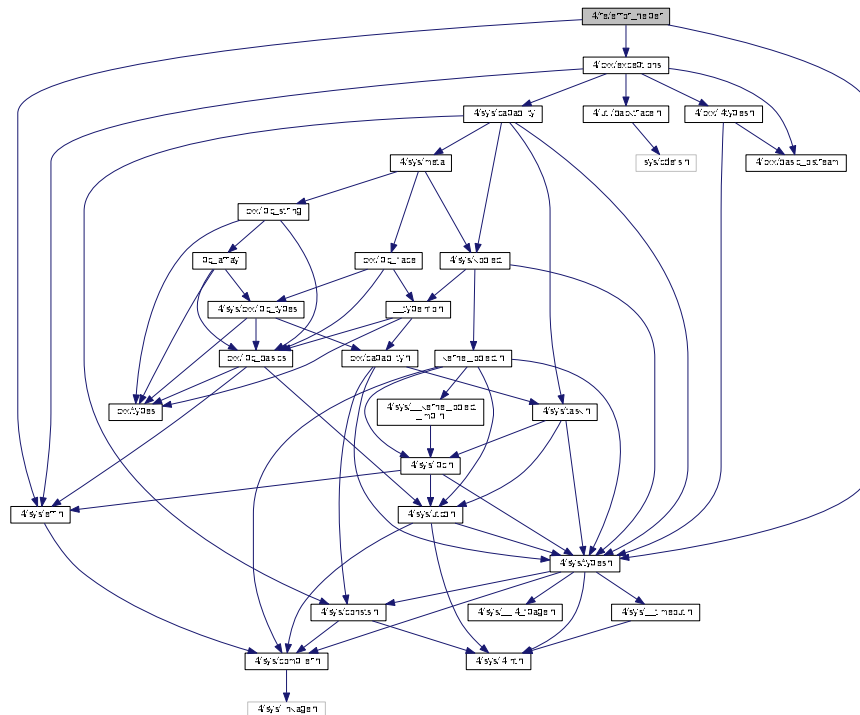
Error helper.

```

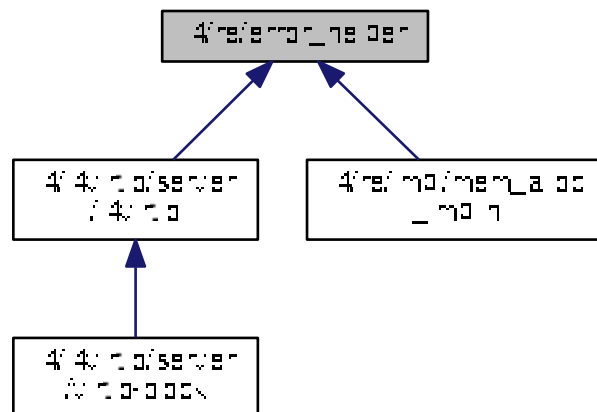
#include <l4/sys/types.h>
#include <l4/cxx/exceptions>
#include <l4/sys/err.h>

```

Include dependency graph for error\_helper:



This graph shows which files directly or indirectly include this file:



## Namespaces

- L4Re
- L4Re C++ Interfaces.*

## Functions

- long `L4Re::chksys` (long err, char const \*extra="", long ret=0)  
*Generate C++ exception on error.*
- long `L4Re::chksys` (`l4_msgtag_t` const &t, char const \*extra="", `l4_utcb_t` \*utcb=`l4_utcb`(), long ret=0)  
*Generate C++ exception on error.*
- long `L4Re::chksys` (`l4_msgtag_t` const &t, `l4_utcb_t` \*utcb, char const \*extra="")  
*Generate C++ exception on error.*
- template<typename T >  
T `L4Re::chkcap` (T &&cap, char const \*extra="", long err=-`L4_ENOMEM`)  
*Check for valid capability or raise C++ exception.*

### 15.197.1 Detailed Description

Error helper.

Definition in file [error\\_helper](#).

## 15.198 error\_helper

```

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 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025 #pragma once
00026
00027 #include <l4/sys/types.h>
00028 #include <l4/cxx/exceptions>
00029 #include <l4/sys/err.h>
00030
00031 namespace L4Re {
00032
00033 #ifdef __EXCEPTIONS
00034 namespace Priv {
00035 inline long __attribute__((__noreturn__)) __runtime_error(long err, char const *extra);
00036
00037 inline long __runtime_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 }
00050
00061 inline
00062 long chksys(long err, char const *extra = "", long ret = 0)
00063 {

```

```

00064 if (L4_UNLIKELY(err < 0))
00065 Priv::__runtime_error(ret ? ret : err, extra);
00066
00067 return err;
00068 }
00069
00082 inline
00083 long chksys(l4_msgtag_t const &t, char const *extra = "",
00084 l4_utcb_t *utcb = l4_utcb(), long ret = 0)
00085 {
00086 if (L4_UNLIKELY(t.has_error()))
00087 Priv::__runtime_error(ret ? ret : l4_error_u(t, utcb), extra);
00088 else if (L4_UNLIKELY(t.label() < 0))
00089 throw L4::Runtime_error(ret ? ret: t.label(), extra);
00090
00091 return t.label();
00092 }
00093
00105 inline
00106 long chksys(l4_msgtag_t const &t, l4_utcb_t *utcb, char const *extra = "")
00107 { return chksys(t, extra, utcb); }
00108
00109 #if 0
00110 inline
00111 long chksys(long ret, long err, char const *extra = "")
00112 {
00113 if (L4_UNLIKELY(ret < 0))
00114 Priv::__runtime_error(err, extra);
00115
00116 return ret;
00117 }
00118 #endif
00119
00136 template<typename T>
00137 inline
00138 #if __cplusplus >= 201103L
00139 T chkcap(T &&cap, char const *extra = "", long err = -L4_ENOMEM)
00140 #else
00141 T chkcap(T cap, char const *extra = "", long err = -L4_ENOMEM)
00142 #endif
00143 {
00144 if (L4_UNLIKELY(!cap.is_valid()))
00145 Priv::__runtime_error(err ? err : cap.cap(), extra);
00146
00147 return cap;
00148 }
00149 #endif
00150
00151 }

```

## 15.199 l4/re/util/event File Reference

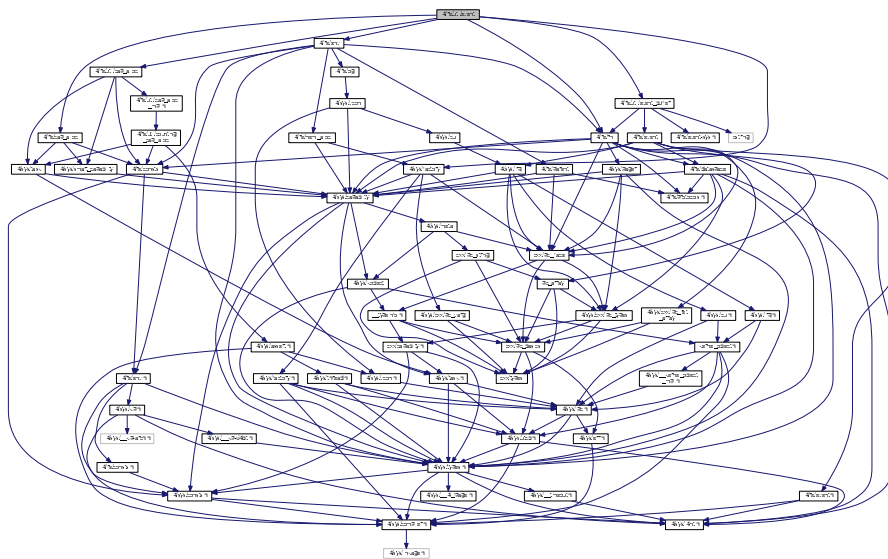
```

#include <l4/re/cap_alloc>
#include <l4/re/util/cap_alloc>
#include <l4/re/env>
#include <l4/re/rm>
#include <l4/re/util/event_buffer>
#include <l4/sys/factory>

```



Include dependency graph for event:



## Data Structures

- class [L4Re::Util::Event\\_t< PAYLOAD >](#)  
*Convenience wrapper for getting access to an event object.*

## Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*

## 15.200 event

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
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 * 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 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
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 #pragma once
00024
00025 #include <14/re/cap_alloc>
00026 #include <14/re/util/cap_alloc>
00027 #include <14/re/env>
00028 #include <14/re/rm>
00029 #include <14/re/util/event_buffer>
00030 #include <14/sys/factory>
```

```

00031
00032 namespace L4Re { namespace Util {
00033
00040 template< typename PAYLOAD >
00041 class Event_t
00042 {
00043 public:
00047 enum Mode
00048 {
00049 Mode_irq,
00050 Mode_polling,
00051 };
00052
00067 template<typename IRQ_TYPE>
00068 int init(L4::Cap<L4Re::Event> event,
00069 L4Re::Env const *env = L4Re::Env::env(),
00070 L4Re::Cap_alloc *ca = L4Re::Cap_alloc::get_cap_alloc
(L4Re::Util::cap_alloc))
00071 {
00072 Auto_cap<L4Re::Dataspace>::Cap ev_ds = ca->alloc<
L4Re::Dataspace>();
00073 if (!ev_ds.is_valid())
00074 return -L4_ENOMEM;
00075
00076 int r;
00077
00078 typename Auto_del_cap<IRQ_TYPE>::Cap ev_irq;
00079 ev_irq = ca->alloc<IRQ_TYPE>();
00080 if (!ev_irq.is_valid())
00081 return -L4_ENOMEM;
00082
00083 if ((r = l4_error(env->factory()->create(ev_irq.get()))))
00084 return r;
00085
00086 if ((r = l4_error(event->bind(0, ev_irq.get()))))
00087 return r;
00088
00089 if ((r = event->get_buffer(ev_ds.get()))
00090 return r;
00091
00092 long sz = ev_ds->size();
00093 if (sz < 0)
00094 return sz;
00095
00096 Rm::Auto_region<void*> buf;
00097
00098 if ((r = env->rm()->attach(&buf, sz, L4Re::Rm::Search_addr,
00099 L4::Ipc::make_cap_rw(ev_ds.get()))))
00100 return r;
00101
00102 _ev_buffer = L4Re::Event_buffer_t<PAYLOAD>(buf.get(), sz);
00103 _ev_ds = ev_ds;
00104 _ev_irq = ev_irq;
00105 _buf = buf;
00106
00107 return 0;
00108 }
00109
00121 int init_poll(L4::Cap<L4Re::Event> event,
00122 L4Re::Env const *env = L4Re::Env::env(),
00123 L4Re::Cap_alloc *ca =
L4Re::Cap_alloc::get_cap_alloc(
L4Re::Util::cap_alloc))
00124 {
00125 Auto_cap<L4Re::Dataspace>::Cap ev_ds = ca->alloc<
L4Re::Dataspace>();
00126 if (!ev_ds.is_valid())
00127 return -L4_ENOMEM;
00128
00129 int r;
00130
00131 if ((r = event->get_buffer(ev_ds.get()))
00132 return r;
00133
00134 long sz = ev_ds->size();
00135 if (sz < 0)
00136 return sz;
00137
00138 Rm::Auto_region<void*> buf;
00139
00140 if ((r = env->rm()->attach(&buf, sz, L4Re::Rm::Search_addr,
00141 L4::Ipc::make_cap_rw(ev_ds.get()))))
00142 return r;
00143
00144 _ev_buffer = L4Re::Event_buffer_t<PAYLOAD>(buf.get(), sz);
00145 _ev_ds = ev_ds;
00146 _buf = buf;

```

```

00147
00148 return 0;
00149 }
00150
00156 L4Re::Event_buffer_t<PAYLOAD> &buffer() { return _ev_buffer; }
00157
00163 L4::Cap<L4::Triggerable> irq() const { return _ev_irq.get(); }
00164
00165 private:
00166 Auto_cap<L4Re::Dataspace>::Cap _ev_ds;
00167 Auto_del_cap<L4::Triggerable>::Cap _ev_irq;
00168 L4Re::Event_buffer_t<PAYLOAD> _ev_buffer;
00169 Rm::Auto_region<void*> _buf;
00170 };
00171
00172 typedef Event_t<Default_event_payload> Event;
00173
00174 }}

```

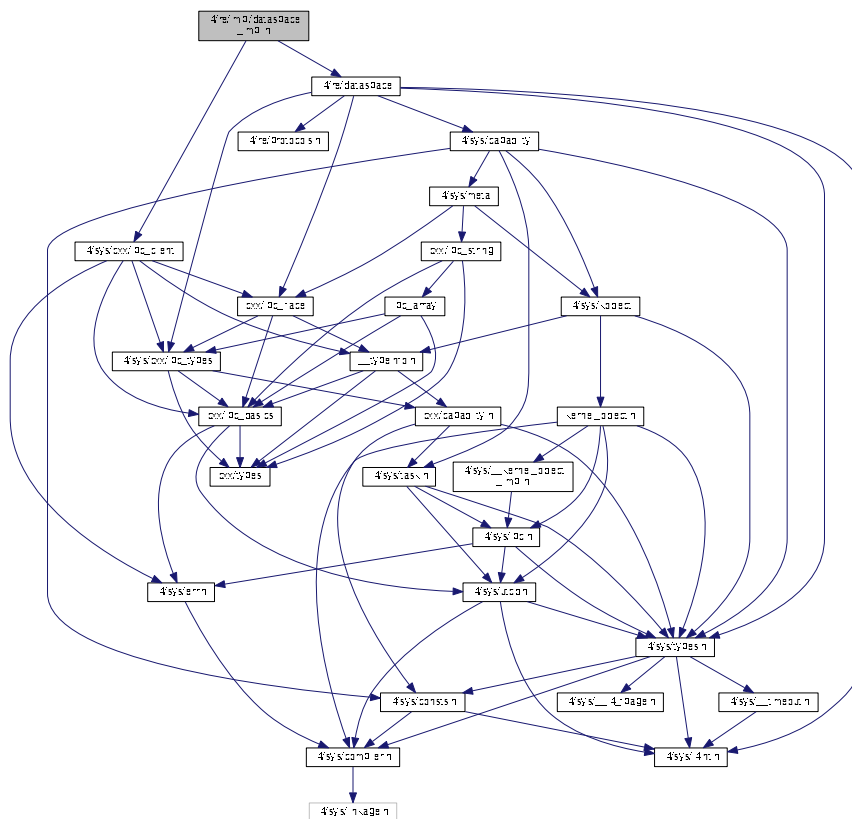
## 15.201 l4/re/impl/dataspace\_impl.h File Reference

Dataspace client stub implementation.

```

#include <l4/re/dataspace>
#include <l4/sys/cxx/ipc_client>
Include dependency graph for dataspace_impl.h:

```



### Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*

### 15.201.1 Detailed Description

Dataspace client stub implementation.

Definition in file [dataspace\\_impl.h](#).

## 15.202 dataspace\_impl.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 * economic rights: Technische Universität Dresden (Germany)
00006 *
00007 * This file is part of TUD:OS and distributed under the terms of the
00008 * GNU General Public License 2.
00009 * Please see the COPYING-GPL-2 file for details.
00010 *
00011 * As a special exception, you may use this file as part of a free software
00012 * library without restriction. Specifically, if other files instantiate
00013 * templates or use macros or inline functions from this file, or you compile
00014 * this file and link it with other files to produce an executable, this
00015 * file does not by itself cause the resulting executable to be covered by
00016 * the GNU General Public License. This exception does not however
00017 * invalidate any other reasons why the executable file might be covered by
00018 * the GNU General Public License.
00019 */
00020 #include <l4/re/dataspace>
00021 #include <l4/sys/cxx/ipc_client>
00022
00023 L4_RPC_DEF(L4Re::Dataspace::clear);
00024 L4_RPC_DEF(L4Re::Dataspace::allocate);
00025 L4_RPC_DEF(L4Re::Dataspace::copy_in);
00026 L4_RPC_DEF(L4Re::Dataspace::phys);
00027 L4_RPC_DEF(L4Re::Dataspace::info);
00028 L4_RPC_DEF(L4Re::Dataspace::take);
00029 L4_RPC_DEF(L4Re::Dataspace::release);
00030
00031 namespace L4Re {
00032
00033 long
00034 Dataspace::__map(unsigned long offset, unsigned char *size, unsigned long flags,
00035 l4_addr_t local_addr) const throw()
00036 {
00037 l4_addr_t spot = local_addr & ~(~0UL << l4_umword_t(*size));
00038 l4_addr_t base = local_addr & (~0UL << l4_umword_t(*size));
00039 L4::Ipc::Rcv_fpage r;
00040 r = L4::Ipc::Rcv_fpage::mem(base, *size, 0);
00041 L4::Ipc::Snd_fpage fp;
00042 long err = map_t::call(c(), offset, spot, flags, r, fp, l4_utcb());
00043 if (L4_UNLIKELY(err < 0))
00044 return err;
00045 *size = fp.rcv_order();
00046 return err;
00047 }
00048
00049 long
00050 Dataspace::map_region(l4_addr_t offset, unsigned long flags,
00051 l4_addr_t min_addr, l4_addr_t max_addr) const throw()
00052 {
00053 min_addr = l4_trunc_page(min_addr);
00054 max_addr = l4_round_page(max_addr);
00055 unsigned char order = L4_LOG2_PAGESIZE;
00056
00057 long err = 0;
00058
00059 while (min_addr < max_addr)
00060 {
00061 unsigned char order_mapped;
00062 order_mapped = order
00063 = l4_fpage_max_order(order, min_addr, min_addr, max_addr, min_addr);
00064 err = __map(offset, &order_mapped, flags, min_addr);
00065 if (L4_UNLIKELY(err < 0))
00066 return err;
00067 if (order > order_mapped)
00068 order = order_mapped;
00069 }
00070 }

```

```

00075
00076 min_addr += 1UL << order;
00077 offset += 1UL << order;
00078
00079 if (min_addr >= max_addr)
00080 return 0;
00081
00082 while (min_addr != l4_trunc_size(min_addr, order)
00083 || max_addr < l4_round_size(min_addr + 1, order))
00084 --order;
00085 }
00086
00087 return 0;
00088 }
00089
00090
00091 long
00092 Dataspace::map(l4_addr_t offset, unsigned long flags,
00093 l4_addr_t local_addr,
00094 l4_addr_t min_addr, l4_addr_t max_addr) const throw()
00095 {
00096 min_addr = l4_trunc_page(min_addr);
00097 max_addr = l4_round_page(max_addr);
00098 local_addr = l4_trunc_page(local_addr);
00099 unsigned char order
00100 = l4_fpage_max_order(L4_LOG2_PAGESIZE, local_addr, min_addr, max_addr
00101 , local_addr);
00102
00103 return __map(offset, &order, flags, local_addr);
00104 }
00105
00106 unsigned long
00107 Dataspace::size() const throw()
00108 {
00109 Stats stats = Stats();
00110 int err = info(&stats);
00111 if (err < 0)
00112 return 0;
00113 return stats.size;
00114 }
00115
00116 long
00117 Dataspace::flags() const throw()
00118 {
00119 Stats stats = Stats();
00120 int err = info(&stats);
00121 if (err < 0)
00122 return err;
00123 return stats.flags;
00124 }
00125 };

```

## 15.203 l4/re/impl/mem\_alloc\_impl.h File Reference

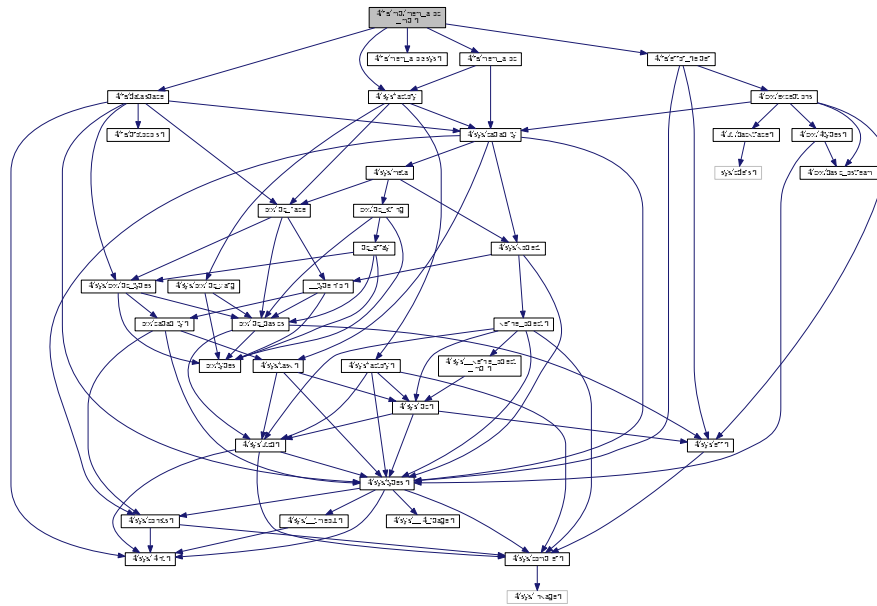
Memory allocator client stub implementation.

```

#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

- [L4Re](#)  
*L4Re C++ Interfaces.*

### 15.203.1 Detailed Description

Memory allocator client stub implementation.

Definition in file [mem\\_alloc\\_impl.h](#).

## 15.204 mem\_alloc\_impl.h

```

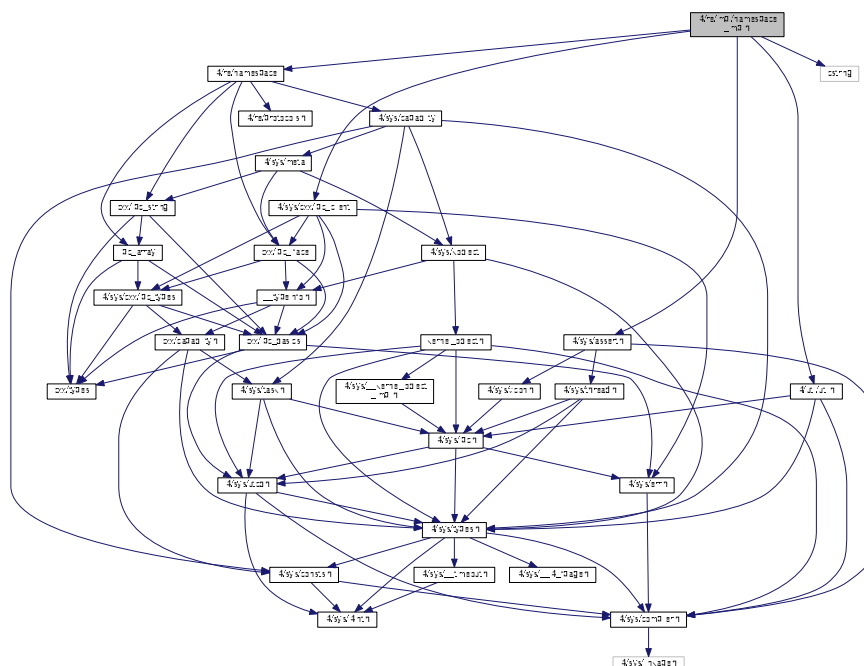
00001
00002 /*
00003 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00004 * Alexander Warg <warg@os.inf.tu-dresden.de>
00005 * economic rights: Technische Universität Dresden (Germany)
00006 *
00007 * This file is part of TUD:OS and distributed under the terms of the
00008 * GNU General Public License 2.
00009 * Please see the COPYING-GPL-2 file for details.
00010 *
00011 * As a special exception, you may use this file as part of a free software
00012 * library without restriction. Specifically, if other files instantiate
00013 * templates or use macros or inline functions from this file, or you compile
00014 * this file and link it with other files to produce an executable, this
00015 * file does not by itself cause the resulting executable to be covered by
00016 * the GNU General Public License. This exception does not however
00017 * invalidate any other reasons why the executable file might be covered by
00018 * the GNU General Public License.
00019 */
00020 #include <l4/re/mem_alloc>
00021 #include <l4/re/mem_alloc-sys.h>
00022 #include <l4/re/dataspace>

```

```
00026 #include <l4/re/error_helper>
00027
00028 #include <l4/sys/factory>
00029
00030
00031 namespace L4Re
00032 {
00033
00034 long
00035 Mem_alloc::alloc(long size,
00036 L4::Cap<Dataspace> mem, unsigned long flags,
00037 unsigned long align) const throw()
00038 {
00039 L4::Cap<L4::Factory> f(cap());
00040 return l4_error(f->create(mem, L4Re::Dataspace::Protocol)
00041 << l4_mword_t(size)
00042 << l4_umword_t(flags)
00043 << l4_umword_t(align));
00044 }
00045
00046 long
00047 Mem_alloc::free(L4::Cap<Dataspace>) const throw()
00048 {
00049 return 0;
00050 }
00051
00052 };
```

Namespace client stub implementation.

```
#include <l4/re/namespace>
#include <l4/util/util.h>
#include <l4/sys/cxx/ipc_client>
#include <l4/sys/assert.h>
#include <cstring>
Include dependency graph for namespace_impl.h:
```



## Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*

### 15.205.1 Detailed Description

Namespace client stub implementation.

Definition in file [namespace\\_impl.h](#).

### 15.206 namespace\_impl.h

```

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 * 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 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
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 #include <l4/re/namespace>
00024
00025 #include <l4/util/util.h>
00026 #include <l4/sys/cxx/ipc_client>
00027 #include <l4/sys/assert.h>
00028
00029 #include <cstring>
00030
00031 L4_RPC_DEF(L4Re::Namespace::query);
00032 L4_RPC_DEF(L4Re::Namespace::register_obj);
00033 L4_RPC_DEF(L4Re::Namespace::unlink);
00034
00035 namespace L4Re {
00036
00037 long
00038 Namespace::_query(char const *name, unsigned len,
00039 L4::Cap<void> const &target,
00040 l4_umword_t *local_id, bool iterate) const throw()
00041 {
00042 l4_assert(target.is_valid());
00043
00044 L4::Cap<Namespace> ns = c();
00045 L4::Ipc::Array<char const, unsigned long> _name(len, name);
00046
00047 while (_name.length > 0)
00048 {
00049 L4::Ipc::Snd_fpage cap;
00050 L4::Opcode dummy;
00051 int err = query_t::call(ns, _name,
00052 L4::Ipc::Small_buf(target.cap(),
00053 local_id
00054 ? L4_RCV_ITEM_LOCAL_ID
00055 : 0),
00056 cap, dummy, _name);
00057 if (err < 0)
00058 return err;
00059
00060 bool const partly = err & Partly_resolved;
00061 if (cap.id_received())
00062 {
00063 *local_id = cap.data();
00064 return _name.length;

```



```

00065 }
00066
00067 if (partly && iterate)
00068 ns = L4::cap_cast<Namespace>(target);
00069 else
00070 return err;
00071 }
00072
00073 return _name.length;
00074 }
00075
00076 long
00077 Namespace::query(char const *name, unsigned len, L4::Cap<void> const &target,
00078 int timeout, l4_umword_t *local_id, bool iterate) const throw()
00079 {
00080 if (L4_UNLIKELY(len == 0))
00081 return -L4_EINVAL;
00082
00083 long ret;
00084 long rem = timeout;
00085 long to = 0;
00086
00087 if (rem)
00088 to = 10;
00089 do
00090 {
00091 ret = _query(name, len, target, local_id, iterate);
00092
00093 if (ret >= 0)
00094 return ret;
00095
00096 if (L4_UNLIKELY(ret < 0 && (ret != -L4_EAGAIN)))
00097 return ret;
00098
00099 if (rem == to)
00100 return ret;
00101
00102 l4_sleep(to);
00103
00104 if (rem > 0)
00105 {
00106 rem -= to;
00107 if (to > rem)
00108 to = rem;
00109 }
00110
00111 if (to < 100)
00112 to += to;
00113 }
00114 while (486);
00115 }
00116
00117 long
00118 Namespace::query(char const *name, L4::Cap<void> const &target,
00119 int timeout, l4_umword_t *local_id,
00120 bool iterate) const throw()
00121 { return query(name, strlen(name), target, timeout, local_id, iterate); }
00122
00123 }

```

## 15.207 l4/re/impl/rm\_impl.h File Reference

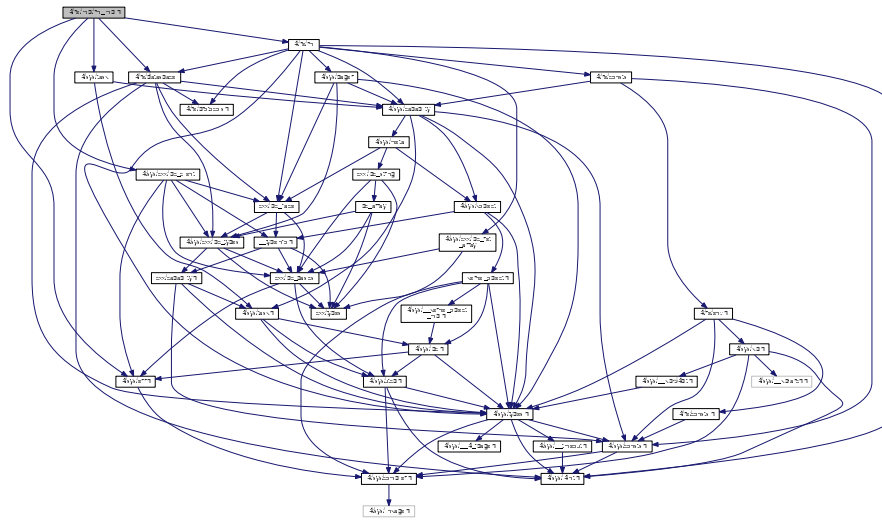
Region map client stub implementation.

```

#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

- [L4Re](#)  
*L4Re C++ Interfaces.*

### 15.207.1 Detailed Description

Region map client stub implementation.

Definition in file [rm\\_impl.h](#).

### 15.208 rm\_impl.h

```

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 * 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 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
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 #include <l4/re/rm>
00024 #include <l4/re/dataspace>
00025
00026 #include <l4/sys/cxx/ipc_client>
00027
00028 #include <l4/sys/task>
00029 #include <l4/sys/err.h>
00030

```

```

00031 L4_RPC_DEF(L4Re::Rm::reserve_area);
00032 L4_RPC_DEF(L4Re::Rm::free_area);
00033 L4_RPC_DEF(L4Re::Rm::attach);
00034 L4_RPC_DEF(L4Re::Rm::detach);
00035 L4_RPC_DEF(L4Re::Rm::get_regions);
00036 L4_RPC_DEF(L4Re::Rm::get_areas);
00037 L4_RPC_DEF(L4Re::Rm::find);
00038
00039 namespace L4Re
00040 {
00041
00042 long
00043 Rm::attach(l4_addr_t *start, unsigned long size, unsigned long flags,
00044 L4::Ipc::Cap<Dataspace> mem, l4_addr_t offs,
00045 unsigned char align) const throw()
00046 {
00047 if (flags & Reserved)
00048 mem = L4::Ipc::Cap<L4Re::Dataspace>();
00049
00050 long e = attach_t::call(c(), start, size, flags, mem, offs, align, mem.cap().cap());
00051 if (e < 0)
00052 return e;
00053
00054 if (flags & Eager_map)
00055 {
00056 unsigned long fl = (flags & Read_only)
00057 ? Dataspace::Map_ro
00058 : Dataspace::Map_rw;
00059 fl |= (flags & Caching) >> Caching_ds_shift;
00060 e = mem.cap()->map_region(offs, fl, *start, *start + size);
00061 }
00062 return e;
00063 }
00064
00065 int
00066 Rm::detach(l4_addr_t start, unsigned long size,
00067 L4::Cap<Dataspace> *mem,
00068 L4::Cap<L4::Task> task, unsigned flags) const throw()
00069 {
00070 l4_addr_t rstart = 0, rsize = 0;
00071 l4_cap_idx_t mem_cap = L4_INVALID_CAP;
00072 long e = detach_t::call(c(), start, size, flags, rstart, rsize, mem_cap);
00073 if (L4_UNLIKELY(e < 0))
00074 return e;
00075
00076 if (mem)
00077 *mem = L4::Cap<L4Re::Dataspace>(mem_cap);
00078
00079 if (!task.is_valid())
00080 return e;
00081
00082 rsize = l4_round_page(rsize);
00083 unsigned order = L4_LOG2_PAGESIZE;
00084 unsigned long sz = (1UL << order);
00085 for (unsigned long p = rstart; rsize; p += sz, rsize -= sz)
00086 {
00087 while (sz > rsize)
00088 {
00089 --order;
00090 sz >>= 1;
00091 }
00092
00093 for (;;)
00094 {
00095 unsigned long m = sz << 1;
00096 if (m > rsize)
00097 break;
00098
00099 if (p & (m - 1))
00100 break;
00101
00102 ++order;
00103 sz <= 1;
00104 }
00105
00106 task->unmap(l4_fpage(p, order, L4_FPAGE_RWX),
00107 L4_FP_ALL_SPACES);
00108 }
00109 return e;
00110 }
00111 }

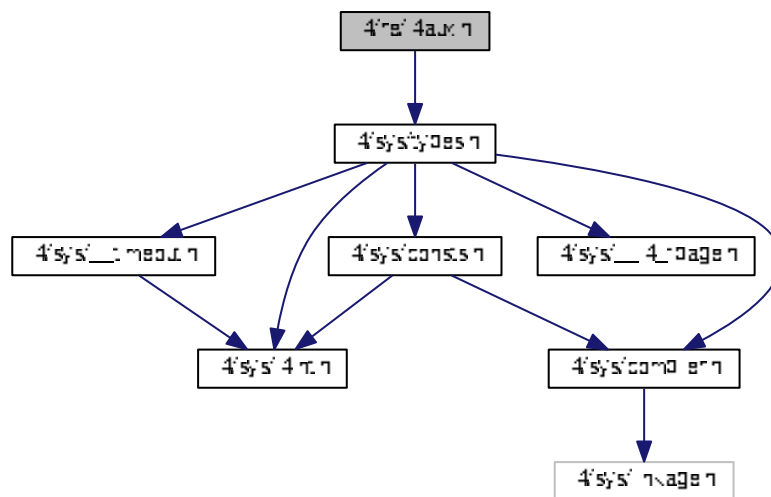
```

## 15.209 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.*

## 15.209.1 Detailed Description

Auxiliary definitions.

Definition in file [l4aux.h](#).

## 15.210 l4aux.h

```

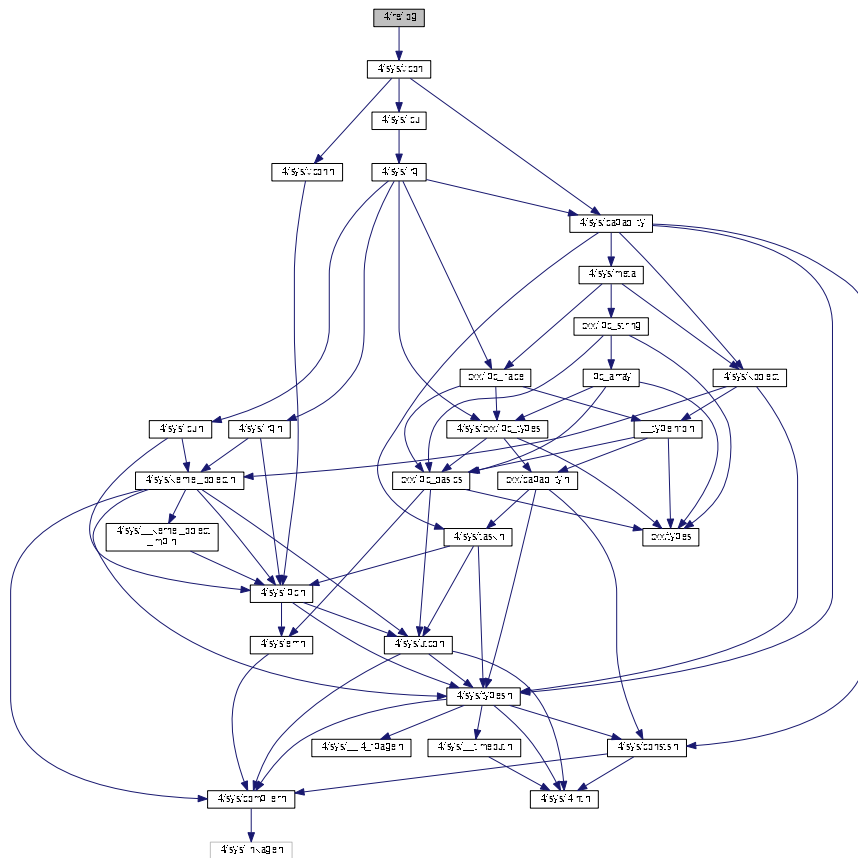
00001 #pragma once
00002
00003 /*
00004 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00005 * Alexander Warg <warg@os.inf.tu-dresden.de>,
00006 * Björn Döbel <doebel@os.inf.tu-dresden.de>
00007 * economic rights: Technische Universität Dresden (Germany)
00008 *
00009 * This file is part of TUD:OS and distributed under the terms of the
00010 * GNU General Public License 2.
00011 * Please see the COPYING-GPL-2 file for details.
00012 *
00013 * As a special exception, you may use this file as part of a free software
00014 * library without restriction. Specifically, if other files instantiate
00015 * templates or use macros or inline functions from this file, or you compile
00016 * this file and link it with other files to produce an executable, this
00017 * file does not by itself cause the resulting executable to be covered by
00018 * the GNU General Public License. This exception does not however
00019 * invalidate any other reasons why the executable file might be covered by
00020 * the GNU General Public License.
00021 */
00022
00023 #include <l4/sys/types.h>
00024
00025 enum l4re_aux_ldr_flags_t
00026 {
00027 L4RE_AUX_LDR_FLAG_EAGER_MAP = 0x1,
00028 L4RE_AUX_LDR_FLAG_ALL_SEGS_COW = 0x2,
00029 L4RE_AUX_LDR_FLAG_PINNED_SEGS = 0x4,
00030 };
00031
00032 typedef struct l4re_aux_t
00033 {
00034 char const * binary;
00035 l4_cap_idx_t kip_ds;
00036 l4_umword_t dbg_lvl;
00037 l4_umword_t ldr_flags;
00038 } l4re_aux_t;
00039

```

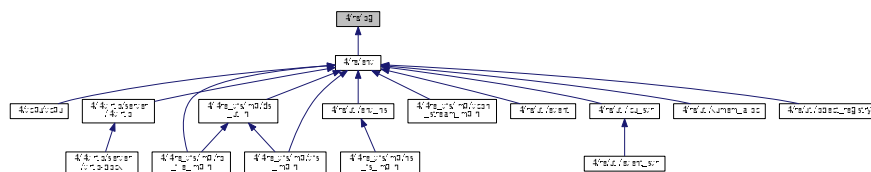
## 15.211 l4/re/log File Reference

Log interface.

```
#include <14/sys/vcon>
Include dependency graph for log:
```



This graph shows which files directly or indirectly include this file:



## Data Structures

- class `L4Re::Log`  
*Log* interface class.

## Namespaces

- L4Re
- L4Re C++ Interfaces.*

### 15.211.1 Detailed Description

Log interface.

Definition in file [log](#).

## 15.212 log

```

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 * This file is part of TUD:OS and distributed under the terms of the
00009 * GNU General Public License 2.
00010 * Please see the COPYING-GPL-2 file for details.
00011 *
00012 * As a special exception, you may use this file as part of a free software
00013 * library without restriction. Specifically, if other files instantiate
00014 * templates or use macros or inline functions from this file, or you compile
00015 * this file and link it with other files to produce an executable, this
00016 * file does not by itself cause the resulting executable to be covered by
00017 * the GNU General Public License. This exception does not however
00018 * invalidate any other reasons why the executable file might be covered by
00019 * the GNU General Public License.
00020 */
00021 #pragma once
00022
00023 #include <l4/sys/vcon>
00024
00025 namespace L4Re {
00026
00027 class L4_EXPORT Log : public L4::Kobject_t<Log, L4::Vcon, L4::PROTO_EMPTY>
00028 {
00029 public:
00030 void printn(char const *string, int len) const throw();
00031 void print(char const *string) const throw();
00032 };
00033
00034 }
```

## 15.213 l4/re/log-sys.h File Reference

Log protocol definition.

### Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*

### Enumerations

- enum [L4Re::Log\\_::Opcodes](#)  
*Logging-service communication-protocol opcodes.*

### 15.213.1 Detailed Description

Log protocol definition.

Definition in file [log-sys.h](#).

## 15.214 log-sys.h

```

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 * 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 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
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 #pragma once
00024
00025 namespace L4Re
00026 {
00027 namespace Log_
00028 {
00034 enum Opcodes { Print };
00035 };
00036 };

```

## 15.215 l4/re/mem\_alloc File Reference

Memory allocator interface.

```

#include <l4/sys/capability>
#include <l4/sys/factory>

```



[illegible]

- class `L4Re::Mem_alloc`  
*Memory allocation interface.*

- **L4Re**  
*L4Re C++ Interfaces.*

### 15.215.1 Detailed Description

Memory allocator interface.

Definition in file [mem\\_alloc](#).

## 15.216 mem\_alloc

```

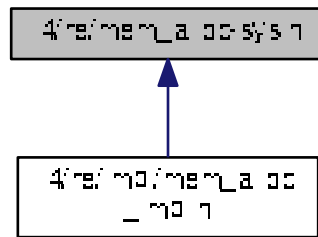
00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00003 /*
00004 * Copyright (C) 2015 Kernkonzept GmbH.
00005 * Author(s): Alexander Warg <alexander.warg@kernkonzept.com>
00006 */
00007 /*
00008 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00009 * Alexander Warg <warg@os.inf.tu-dresden.de>,
00010 * Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00011 * economic rights: Technische Universität Dresden (Germany)
00012 *
00013 * This file is part of TUD:OS and distributed under the terms of the
00014 * GNU General Public License 2.
00015 * Please see the COPYING-GPL-2 file for details.
00016 *
00017 * As a special exception, you may use this file as part of a free software
00018 * library without restriction. Specifically, if other files instantiate
00019 * templates or use macros or inline functions from this file, or you compile
00020 * this file and link it with other files to produce an executable, this
00021 * file does not by itself cause the resulting executable to be covered by
00022 * the GNU General Public License. This exception does not however
00023 * invalidate any other reasons why the executable file might be covered by
00024 * the GNU General Public License.
00025 */
00026 #pragma once
00027
00028 #include <l4/sys/capability>
00029 #include <l4/sys/factory>
00030
00031 namespace L4Re {
00032 class Dataspace;
00033
00034 // MISSING:
00035 // * alignment constraints
00036 // * shall we support superpages in noncont memory?
00037
00038 class L4_EXPORT Mem_alloc :
00039 public L4::Kobject_t<Mem_alloc, L4::Factory, L4::PROTO_EMPTY>
00040 {
00041 public:
00042 enum Mem_alloc_flags
00043 {
00044 Continuous = 0x01,
00045 Pinned = 0x02,
00046 Super_pages = 0x04,
00047 };
00048
00049 long alloc(long size, L4::Cap<Dataspace> mem,
00050 unsigned long flags = 0, unsigned long align = 0) const throw();
00051
00052 /* Deprecation message added Q4 2016 */
00053 long free(L4::Cap<Dataspace> mem) const throw()
00054 {
00055 L4_DEPRECATED("This function is an empty stub and remains for backward compatibility only.
00056 Check documentation for details.");
00057 }
00058 };
00059
00060 };
00061
00062 };

```

## 15.217 l4/re/mem\_alloc-sys.h File Reference

Memory allocator protocol definitions.

This graph shows which files directly or indirectly include this file:



## Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*

## Enumerations

- enum [L4Re::Mem\\_alloc::\\_Opcodes](#)  
*Memory-allocator communication-protocol opcodes.*

### 15.217.1 Detailed Description

Memory allocator protocol definitions.

Definition in file [mem\\_alloc-sys.h](#).

## 15.218 mem\_alloc-sys.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 * economic rights: Technische Universität Dresden (Germany)
00006 *
00007 * This file is part of TUD:OS and distributed under the terms of the
00008 * GNU General Public License 2.
00009 * Please see the COPYING-GPL-2 file for details.
00010 *
00011 * As a special exception, you may use this file as part of a free software
00012 * library without restriction. Specifically, if other files instantiate
00013 * templates or use macros or inline functions from this file, or you compile
00014 * this file and link it with other files to produce an executable, this
00015 * file does not by itself cause the resulting executable to be covered by
00016 * the GNU General Public License. This exception does not however
00017 * invalidate any other reasons why the executable file might be covered by
00018 * the GNU General Public License.
00019 */
00020 #pragma once
00021
00022 namespace L4Re
00023 {
00024 namespace Mem_alloc_
00025 {
00026 enum Opcodes { Alloc, Free };
00027 };
00028 };

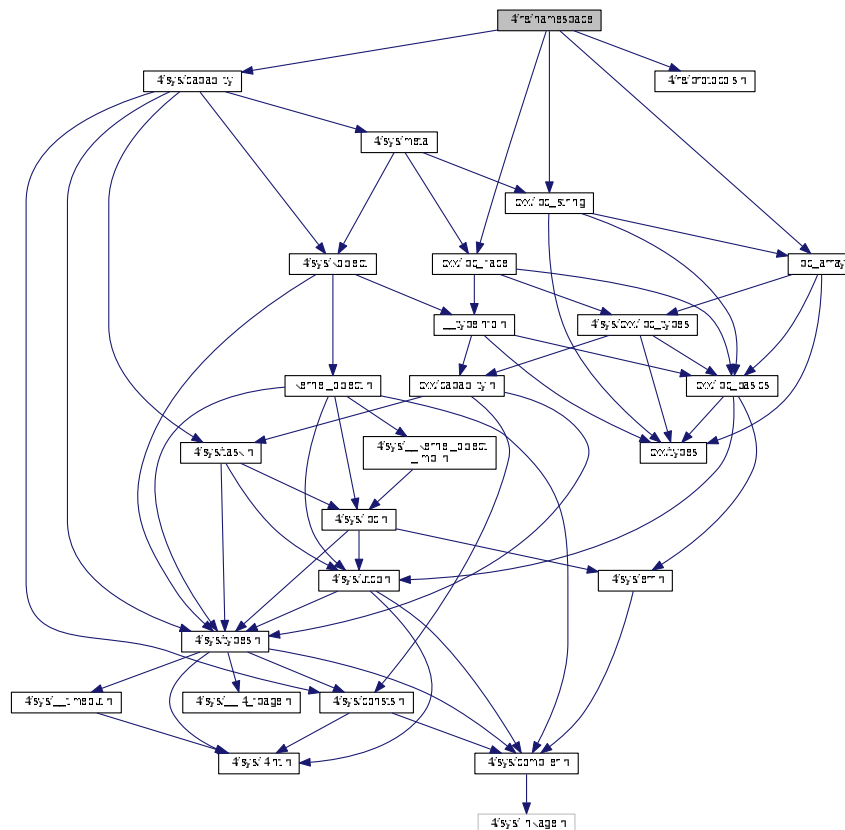
```

## 15.219 l4/re/namespace File Reference

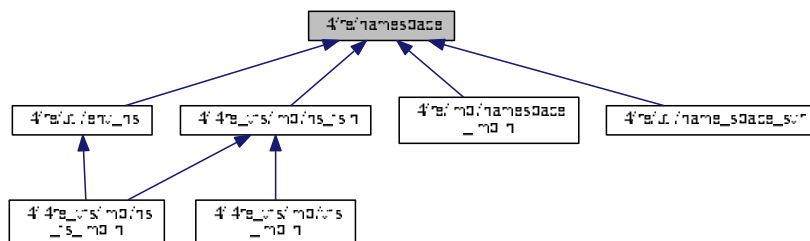
Namespace interface.

```
#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>
```

Include dependency graph for namespace:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [L4Re::Namespace](#)  
*Name-space interface.*

## Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*

### 15.219.1 Detailed Description

Namespace interface.

Definition in file [namespace](#).

## 15.220 namespace

```

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 * This file is part of TUD:OS and distributed under the terms of the
00014 * GNU General Public License 2.
00015 * Please see the COPYING-GPL-2 file for details.
00016 *
00017 * As a special exception, you may use this file as part of a free software
00018 * library without restriction. Specifically, if other files instantiate
00019 * templates or use macros or inline functions from this file, or you compile
00020 * this file and link it with other files to produce an executable, this
00021 * file does not by itself cause the resulting executable to be covered by
00022 * the GNU General Public License. This exception does not however
00023 * invalidate any other reasons why the executable file might be covered by
00024 * the GNU General Public License.
00025 */
00026 #pragma once
00027
00028 #include <l4/sys/capability>
00029 #include <l4/re/protocols.h>
00030 #include <l4/sys/cxx/ipc_iface>
00031 #include <l4/sys/cxx/ipc_array>
00032 #include <l4/sys/cxx/ipc_string>
00033
00034 namespace L4Re {
00035
00060 class L4_EXPORT Namespace :
00061 public L4::Kobject_t<Namespace, L4::Kobject, L4RE_PROTO_NAMESPACE,
00062 L4::Type_info::Demand_t<1> >
00063 {
00064 public:
00068 enum Register_flags
00069 {
00070 Ro = L4_CAP_FPAGE_RO,
00071 Rw = L4_CAP_FPAGE_RW,
00072 Rs = L4_CAP_FPAGE_RS,
00073 Rws = L4_CAP_FPAGE_RWS,
00074 Strong = L4_CAP_FPAGE_S,
00075 Trusted = 0x008,
00076
00077 Cap_flags = Ro | Rw | Strong | Trusted,
00078
00079 Link = 0x100,
00080 Overwrite = 0x200,
00081 };
00082

```

```

00088 enum Query_result_flags
00089 {
00090 Partly_resolved = 0x020,
00091 };
00092
00093 enum Query_timeout
00094 {
00095 To_default = 3600000,
00096 To_non_blocking = 0,
00097 To_forever = -1,
00098 };
00099
00100 L4_RPC_NF(
00101 long, query, (L4::Ipc::Array_ref<char const, unsigned long>
00102 name,
00103 L4::Ipc::Small_buf cap,
00104 L4::Ipc::Snd_fpage &snd_cap,
00105 L4::Ipc::Opt<L4::Opcode &> dummy,
00106 L4::Ipc::Opt<
00107 L4::Ipc::Array_ref<char const, unsigned long> &> out_name));
00108
00109 long query(char const *name, L4::Cap<void> const &cap,
00110 int timeout = To_default,
00111 l4_umword_t *local_id = 0, bool iterate = true) const throw();
00112
00113 long query(char const *name, unsigned len, L4::Cap<void> const &cap,
00114 int timeout = To_default,
00115 l4_umword_t *local_id = 0, bool iterate = true) const throw();
00116
00117 L4_RPC_NF(long, register_obj, (unsigned flags,
00118 L4::Ipc::Array<char const, unsigned long>
00119 name,
00120 L4::Ipc::Opt< L4::Ipc::Cap<void> > obj),
00121 L4::Ipc::Call_t<L4_CAP_FPAGE_W>);
00122
00123 long register_obj(char const *name, L4::Ipc::Cap<void> obj,
00124 unsigned flags = Rw) const throw()
00125 {
00126 return register_obj_t::call(c(), flags,
00127 L4::Ipc::Array<char const, unsigned long>
00128 (
00129 __builtin_strlen(name), name),
00130 obj);
00131 }
00132
00133 L4_RPC_NF_OP(3, // backward compatibility opcode
00134 long, unlink, (L4::Ipc::Array<char const, unsigned long>
00135 name),
00136 L4::Ipc::Call_t<L4_CAP_FPAGE_W>);
00137
00138 long unlink(char const* name)
00139 {
00140 return unlink_t::call(c(), L4::Ipc::Array<char const, unsigned long>
00141 (
00142 __builtin_strlen(name), name));
00143 }
00144
00145 typedef L4::Typeid::Rpc<query_t, register_obj_t, unlink_t>
00146 Rpc;
00147
00148 private:
00149 long _query(char const *name, unsigned len,
00150 L4::Cap<void> const &target, l4_umword_t *local_id,
00151 bool iterate) const throw();
00152 };
00153
00154 };
00155
00156
00157
00158

```

## 15.221 l4/re/namespace-sys.h File Reference

Namespace protocol definitions.

### Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*

## Enumerations

- enum [L4Re::Namespace\\_::Opcodes](#)  
*Name-space communication-protocol opcodes.*

### 15.221.1 Detailed Description

Namespace protocol definitions.

Definition in file [namespace-sys.h](#).

## 15.222 namespace-sys.h

```

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 * 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 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
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 #pragma once
00024
00025 namespace L4Re {
00026 namespace Namespace_
00027 {
00033 enum Opcodes { Query, Register, Link, Unlink };
00034 };
00035 };

```

## 15.223 l4/re/parent File Reference

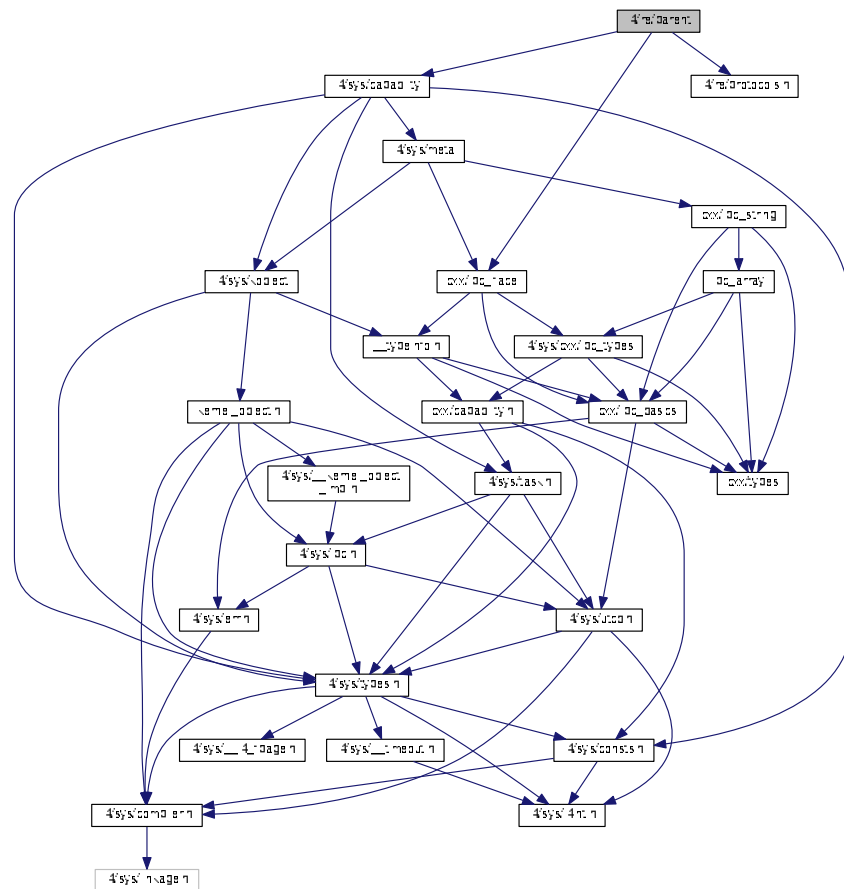
Parent interface.

```

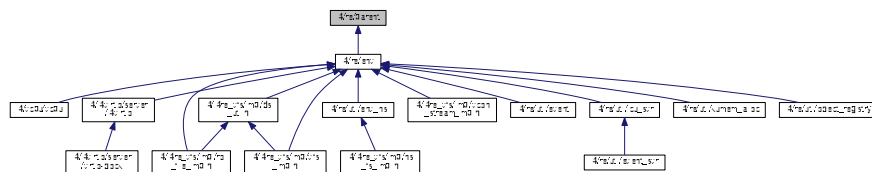
#include <l4/sys/capability>
#include <l4/re/protocols.h>

```

```
#include <linux/sys/cxx/ipc_iface>
```



This graph shows which files directly or indirectly include this file:



## Data Structures

- class `L4Re::Parent`  
*Parent* interface.

## Namespaces

- L4Re
- L4Re C++ Interfaces.*



### 15.223.1 Detailed Description

Parent interface.

Definition in file [parent](#).

## 15.224 parent

```

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 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025 #pragma once
00026
00027 #include <l4/sys/capability>
00028 #include <l4/re/protocols.h>
00029 #include <l4/sys/cxx/ipc_iface>
00030
00031 namespace L4Re {
00032
00051 class L4_EXPORT Parent :
00052 public L4::Kobject_t<Parent, L4::Kobject, L4RE_PROTO_PARENT>
00053 {
00054 public:
00064 L4_INLINE_RPC(long, signal, (unsigned long sig, unsigned long val));
00065 typedef L4::Typeid::Rpc<signal_t> Rpc;
00066 };
00067 };
00068

```

## 15.225 l4/re/parent-sys.h File Reference

Parent protocol definition.

### Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*

### Enumerations

- enum [L4Re::Parent\\_::Opcodes](#)  
*Parent communication-protocol opcodes.*

### 15.225.1 Detailed Description

Parent protocol definition.

Definition in file [parent-sys.h](#).

## 15.226 parent-sys.h

```

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 * 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 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
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 #pragma once
00024
00025 namespace L4Re
00026 {
00027 namespace Parent_
00028 {
00034 enum Opcodes { Signal };
00035 };
00036 };

```

## 15.227 l4/re/rm File Reference

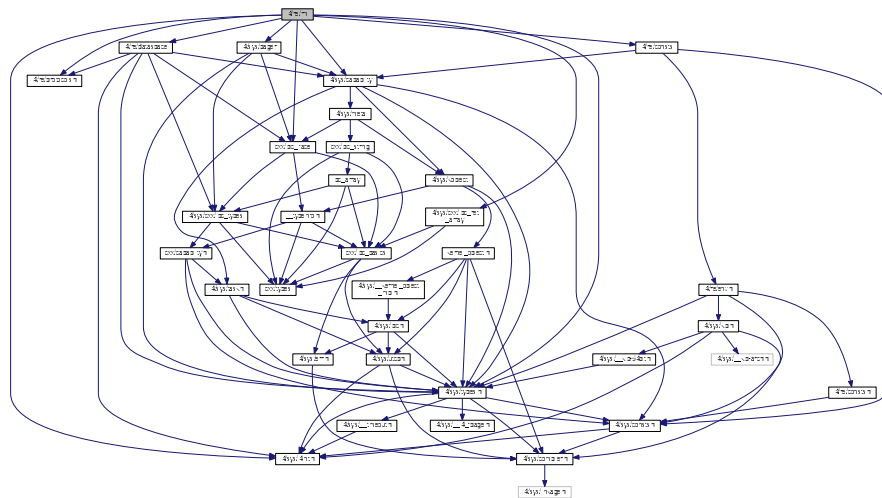
Region mapper interface.

```

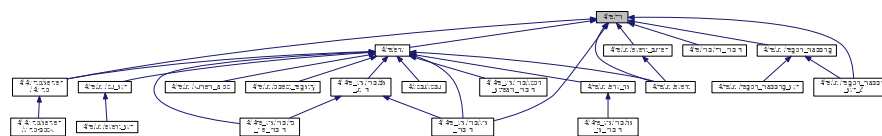
#include <l4/sys/types.h>
#include <l4/sys/l4int.h>
#include <l4/sys/capability>
#include <l4/re/protocols.h>
#include <l4/sys/pager>
#include <l4/sys/cxx/ipc_iface>
#include <l4/sys/cxx/ipc_ret_array>
#include <l4/re/consts>
#include <l4/re/dataspace>

```

Include dependency graph for rm:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class `L4Re::Rm`  
*Region map.*

## Namespaces

- L4Re
- L4Re C++ Interfaces.*

### 15.227.1 Detailed Description

### Region mapper interface.

Definition in file [rm](#).

## 15.228 rm

```

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 * Torsten Frenzel <frenzel@os.inf.tu-dresden.de>
00012 * economic rights: Technische Universität Dresden (Germany)
00013 *
00014 * This file is part of TUD:OS and distributed under the terms of the
00015 * GNU General Public License 2.
00016 * Please see the COPYING-GPL-2 file for details.
00017 *
00018 * As a special exception, you may use this file as part of a free software
00019 * library without restriction. Specifically, if other files instantiate
00020 * templates or use macros or inline functions from this file, or you compile
00021 * this file and link it with other files to produce an executable, this
00022 * file does not by itself cause the resulting executable to be covered by
00023 * the GNU General Public License. This exception does not however
00024 * invalidate any other reasons why the executable file might be covered by
00025 * the GNU General Public License.
00026 */
00027 #pragma once
00028
00029 #include <l4/sys/types.h>
00030 #include <l4/sys/l4int.h>
00031 #include <l4/sys/capability>
00032 #include <l4/re/protocols.h>
00033 #include <l4/sys/pager>
00034 #include <l4/sys/cxx/ipc_iface>
00035 #include <l4/sys/cxx/ipc_ret_array>
00036 #include <l4/re/consts>
00037 #include <l4/re/dataspace>
00038
00039 namespace L4Re {
00040
00069 class L4_EXPORT Rm :
00070 public L4::Kobject_t<Rm, L4::Pager, L4RE_PROTO_RM,
00071 L4::Type_info::Demand_t<1> >
00072 {
00073 public:
00075 enum Detach_result
00076 {
00077 Detached_ds = 0,
00078 Kept_ds = 1,
00079 Split_ds = 2,
00080 Detach_result_mask = 3,
00081
00082 Detach_again = 4,
00083 };
00084
00086 enum Region_flags
00087 {
00088 Read_only = 0x01,
00089 Detach_free = 0x02,
00091 Pager = 0x04,
00092 Reserved = 0x08,
00093
00095 Caching_shift = 8,
00097 Caching_ds_shift = Caching_shift - Dataspace::Map_caching_shift,
00099 Caching = Dataspace::Map_caching_mask << Caching_ds_shift,
00101 Cache_normal = Dataspace::Map_normal << Caching_ds_shift,
00103 Cache_buffered = Dataspace::Map_bufferable << Caching_ds_shift,
00105 Cache_uncached = Dataspace::Map_uncacheable << Caching_ds_shift,
00106
00107 Region_flags = Caching | 0x0f,
00108 };
00109
00111 enum Attach_flags
00112 {
00113 Search_addr = 0x20,
00114 In_area = 0x40,
00115 Eager_map = 0x80,
00116
00117 Attach_flags = 0xf0,
00118 };
00119
00121 enum Detach_flags
00122 {
00132 Detach_exact = 1,
00142 Detach_overlap = 2,
00143
00151 Detach_keep = 4,
00152 };

```

```

00153
00154
00155 template< typename T >
00156 class Auto_region
00157 {
00158 private:
00159 T _addr;
00160 mutable L4::Cap<Rm> _rm;
00161
00162 public:
00163 Auto_region() throw()
00164 : _addr(0), _rm(L4::Cap<Rm>::Invalid) {}
00165
00166 explicit Auto_region(T addr) throw()
00167 : _addr(addr), _rm(L4::Cap<Rm>::Invalid) {}
00168
00169 Auto_region(T addr, L4::Cap<Rm> const &rm) throw()
00170 : _addr(addr), _rm(rm) {}
00171
00172 Auto_region(Auto_region const &o) throw() : _addr(o.get()), _rm(o._rm)
00173 { o.release(); }
00174
00175 Auto_region &operator = (Auto_region const &o) throw()
00176 {
00177 if (&o != this)
00178 {
00179 if (_rm.is_valid())
00180 _rm->detach(l4_addr_t(_addr), 0);
00181 _rm = o._rm;
00182 _addr = o.release();
00183 }
00184 return *this;
00185 }
00186
00187 ~Auto_region() throw()
00188 {
00189 if (_rm.is_valid())
00190 _rm->detach(l4_addr_t(_addr), 0);
00191 }
00192
00193 T get() const throw() { return _addr; }
00194 T release() const throw() { _rm = L4::Cap<Rm>::Invalid; return _addr; }
00195 void reset(T addr, L4::Cap<Rm> const &rm) throw()
00196 {
00197 if (_rm.is_valid())
00198 _rm->detach(l4_addr_t(_addr), 0);
00199
00200 _rm = rm;
00201 _addr = addr;
00202 }
00203
00204 void reset() throw()
00205 { reset(0, L4::Cap<Rm>::Invalid); }
00206
00208 T operator * () const throw() { return _addr; }
00209
00211 T operator -> () const throw() { return _addr; }
00212
00213 };
00214
00239 long reserve_area(l4_addr_t *start, unsigned long size,
00240 unsigned flags = 0,
00241 unsigned char align = L4::PAGESHIFT) const throw()
00242 { return reserve_area_t::call(c(), start, size, flags, align); }
00243
00244 L4_RPC_NF(long, reserve_area, (L4::Ipc::In_out<l4_addr_t *> start,
00245 unsigned long size,
00246 unsigned flags,
00247 unsigned char align));
00248
00264 template< typename T >
00265 long reserve_area(T **start, unsigned long size,
00266 unsigned flags = 0,
00267 unsigned char align = L4::PAGESHIFT) const throw()
00268 { return reserve_area_t::call(c(), (l4_addr_t*)start, size, flags, align); }
00269
00282 L4_RPC(long, free_area, (l4_addr_t addr));
00283
00284 L4_RPC_NF(long, attach, (L4::Ipc::In_out<l4_addr_t *> start,
00285 unsigned long size, unsigned long flags,
00286 L4::Ipc::Opt<L4::Ipc::Cap<Dataspace> > mem,
00287 l4_addr_t offs, unsigned char align,
00288 L4::Ipc::Opt<l4_cap_idx_t> client_cap));
00289
00290 L4_RPC_NF(long, detach, (l4_addr_t addr, unsigned long size, unsigned flags,
00291 l4_addr_t &start, l4_addr_t &rsz,
00292 l4_cap_idx_t &mem_cap));

```

```

00293
00337 long attach(l4_addr_t *start, unsigned long size, unsigned long flags,
00338 L4::Ipc::Cap<Dataspace> mem, l4_addr_t offs = 0,
00339 unsigned char align = L4_PAGESHIFT) const throw();
00340
00344 template< typename T >
00345 long attach(T **start, unsigned long size, unsigned long flags,
00346 L4::Ipc::Cap<Dataspace> mem, l4_addr_t offs = 0,
00347 unsigned char align = L4_PAGESHIFT) const throw()
00348 {
00349 union X { l4_addr_t a; T* t; };
00350 X *x = reinterpret_cast<X*>(start);
00351 return attach(&x->a, size, flags, mem, offs, align);
00352 }
00353
00354 template< typename T >
00355 long attach(Auto_region<T> *start, unsigned long size, unsigned long flags,
00356 L4::Ipc::Cap<Dataspace> mem, l4_addr_t offs = 0,
00357 unsigned char align = L4_PAGESHIFT) const throw()
00358 {
00359 l4_addr_t addr = (l4_addr_t)start->get();
00360
00361 long res = attach(&addr, size, flags, mem, offs, align);
00362 if (res < 0)
00363 return res;
00364
00365 start->reset((T)addr, L4::Cap<Rm>(cap()));
00366 return res;
00367 }
00368
00386 int detach(l4_addr_t addr, L4::Cap<Dataspace> *mem,
00387 L4::Cap<L4::Task> const &task = This_task) const throw();
00388
00392 int detach(void *addr, L4::Cap<Dataspace> *mem,
00393 L4::Cap<L4::Task> const &task = This_task) const throw();
00394
00414 int detach(l4_addr_t start, unsigned long size, L4::Cap<Dataspace> *mem,
00415 L4::Cap<L4::Task> const &task) const throw();
00416
00461 int find(l4_addr_t *addr, unsigned long *size, l4_addr_t *offset,
00462 unsigned *flags, L4::Cap<Dataspace> *m) throw()
00463 { return find_t::call(c(), addr, size, flags, offset, m); }
00464
00465 L4_RPC_NF(int, find, (L4::Ipc::In_out<l4_addr_t *> addr,
00466 L4::Ipc::In_out<unsigned long *> size,
00467 unsigned *flags, l4_addr_t *offset,
00468 L4::Ipc::As_value<L4::Cap<Dataspace> > *m));
00469
00470 struct Region
00471 {
00472 l4_addr_t start;
00473 l4_addr_t end;
00474 l4_addr_t offset;
00475 L4::Cap<Dataspace> ds;
00476 };
00477
00478 struct Area
00479 {
00480 l4_addr_t start;
00481 l4_addr_t end;
00482 };
00483
00484 L4_RPC(long, get_regions, (l4_addr_t start,
00485 L4::Ipc::Ret_array<Region> regions));
00486
00487 L4_RPC(long, get_areas, (l4_addr_t start,
00488 L4::Ipc::Ret_array<Area> areas));
00489
00487 int detach(l4_addr_t start, unsigned long size, L4::Cap<Dataspace> *mem,
00488 L4::Cap<L4::Task> task, unsigned flags) const throw();
00489
00490 typedef L4::Typeid::Rpcs<attach_t, detach_t, find_t,
00491 reserve_area_t, free_area_t,
00492 get_regions_t, get_areas_t> Rpcs;
00493 };
00494
00495
00496 inline int
00497 Rm::detach(l4_addr_t addr, L4::Cap<Dataspace> *mem,
00498 L4::Cap<L4::Task> const &task) const throw()
00499 { return detach(addr, 1, mem, task, Detach_overlap); }
00500
00501 inline int
00502 Rm::detach(void *addr, L4::Cap<Dataspace> *mem,
00503 L4::Cap<L4::Task> const &task) const throw()
00504 { return detach((l4_addr_t)addr, 1, mem, task, Detach_overlap); }
00505
00506 inline int

```

```

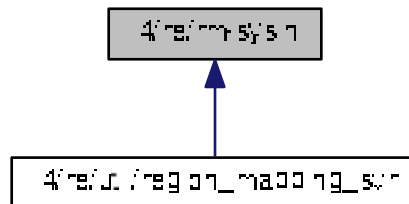
00507 Rm::detach(l4_addr_t addr, unsigned long size,
00508 L4::Cap<Dataspace> *mem,
00509 L4::Cap<L4::Task> const &task) const throw()
00509 { return detach(addr, size, mem, task, Detach_exact); }
00510
00511 };
00512

```

## 15.229 l4/re/rm-sys.h File Reference

Region mapper protocol definitions.

This graph shows which files directly or indirectly include this file:



### Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*

### Enumerations

- enum [L4Re::Rm\\_::Opcodes](#)  
*Region-map communication-protocol opcodes.*

#### 15.229.1 Detailed Description

Region mapper protocol definitions.

Definition in file [rm-sys.h](#).

## 15.230 rm-sys.h

```
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 * 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 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
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 #pragma once
00024
00025 namespace L4Re
00026 {
00027 namespace Rm_
00028 {
00034 enum Opcodes
00035 {
00036 Attach, Detach, Find, Attach_area, Detach_area, Get_regions, Get_areas
00037 };
00038 };
00039 };
```

## 15.231 l4/re/util/bitmap\_cap\_alloc File Reference

Bitmap capability allocator.

```
#include <l4/re/util/item_alloc>
#include <l4/sys/capability>
#include <l4/sys/task.h>
```



- class `L4Re::Util::Cap_alloc_base`  
*Capability allocator.*

- L4Re

*L4Re C++ Interfaces.*

Definition in file [bitmap\\_cap\\_alloc](#).

## 15.232 bitmap\_cap\_alloc

```

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 * This file is part of TUD:OS and distributed under the terms of the
00009 * GNU General Public License 2.
00010 * Please see the COPYING-GPL-2 file for details.
00011 *
00012 * As a special exception, you may use this file as part of a free software
00013 * library without restriction. Specifically, if other files instantiate
00014 * templates or use macros or inline functions from this file, or you compile
00015 * this file and link it with other files to produce an executable, this
00016 * file does not by itself cause the resulting executable to be covered by
00017 * the GNU General Public License. This exception does not however
00018 * invalidate any other reasons why the executable file might be covered by
00019 * the GNU General Public License.
00020 */
00021
00022 #pragma once
00023
00024 #include <l4/re/util/item_alloc>
00025 #include <l4/sys/capability>
00026 #include <l4/sys/task.h>
00027
00028 namespace L4Re { namespace Util {
00029
00030 class Cap_alloc_base
00031 {
00032 private:
00033 long _bias;
00034 Item_alloc_base _items;
00035
00036 public:
00037 enum State { Free = 0, Allocated, Unknown };
00038 Cap_alloc_base(long max, void *mem, long bias = 0)
00039 throw() : _bias(bias), _items(max, mem) {}
00040
00041 L4::Cap<void> alloc() throw()
00042 {
00043 long cap = _items.alloc();
00044 if (cap < 0)
00045 return L4::Cap<void>::Invalid;
00046 return L4::Cap<void>((cap + _bias) << L4_CAP_SHIFT);
00047 }
00048
00049 long hint() const { return _items.hint(); }
00050
00051 template< typename T >
00052 L4::Cap<T> alloc() throw()
00053 { return L4::Cap<T>(alloc().cap()); }
00054
00055 State is_allocated(L4::Cap<void> c) const throw()
00056 {
00057 long idx = (c.cap() >> L4_CAP_SHIFT);
00058
00059 if (idx < _bias)
00060 return Unknown;
00061
00062 idx -= _bias;
00063 return _items.is_allocated(idx) ? Allocated : Free;
00064 }
00065
00066 template< typename T >
00067 void free(L4::Cap<T> const &cap, l4_cap_idx_t task = -1UL,
00068 l4_umword_t unmap_flags = L4_FP_ALL_SPACES) throw()
00069 {
00070 long idx = (cap.cap() >> L4_CAP_SHIFT);
00071 if (idx < _bias)
00072 return;
00073
00074 idx -= _bias;
00075
00076 _items.free(idx);
00077
00078 if (task != -1UL)
00079 l4_task_unmap(task, cap.fpage(), unmap_flags | 2);
00080 }
00081
00082 // since we hav no counters assume couter always > 0
00083 void take(L4::Cap<void>) throw() {}

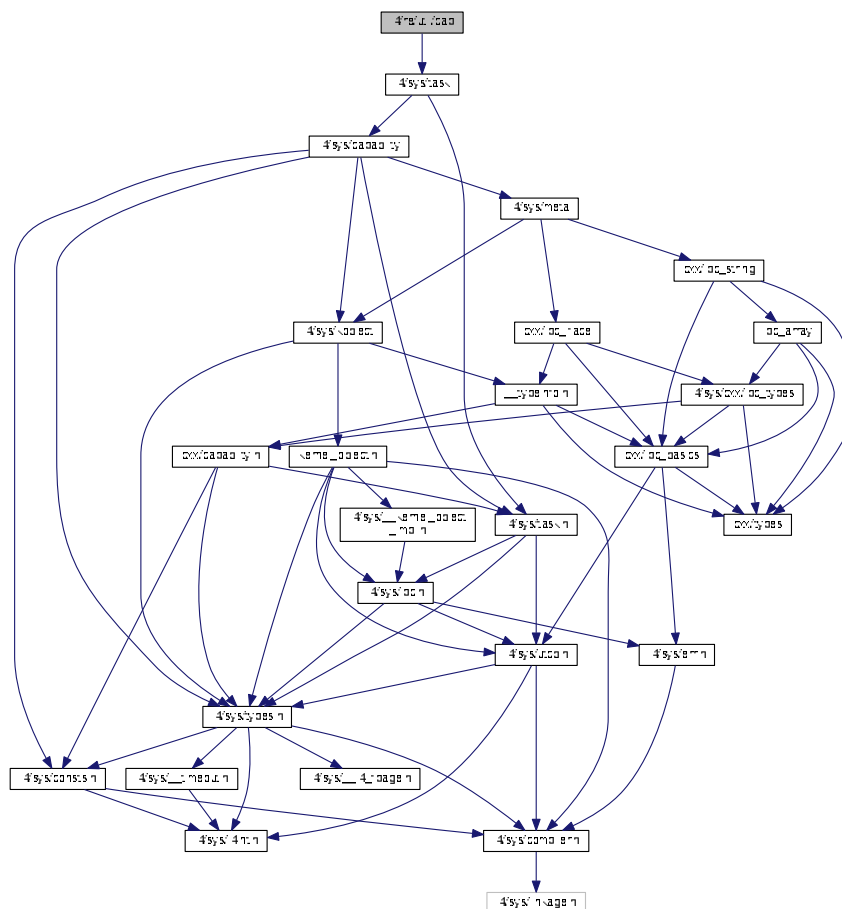
```

```
00099 bool release(L4::Cap<void>, l4_cap_idx_t task =
00100 L4_INVALID_CAP,
00101 unsigned unmap_flags = L4_FP_ALL_SPACES) throw()
00102 { (void)task; (void)unmap_flags; return false; }
00103
00104 long last() throw()
00105 {
00106 return _items.size() + _bias - 1;
00107 }
00108 };
00109
00109 template< long Size >
00110 class Cap_alloc : public Cap_alloc_base
00111 {
00112 private:
00113 typename Bitmap_base::Word<Size>::Type _bits[Bitmap_base::Word<Size>::Size];
00114
00115 public:
00116 explicit Cap_alloc(long bias = 0) throw()
00117 : Cap_alloc_base(Size, _bits, bias) {}
00118
00119 };
00120
00121 }
00122 }
```

### 15.233 I4/re/util/cap File Reference

### Capability utility functions.

```
#include <l4/sys/task>
Include dependency graph for cap:
```



## Namespaces

- [L4Re](#)

*[L4Re](#) C++ Interfaces.*

### 15.233.1 Detailed Description

Capability utility functions.

Definition in file [cap](#).

## 15.234 cap

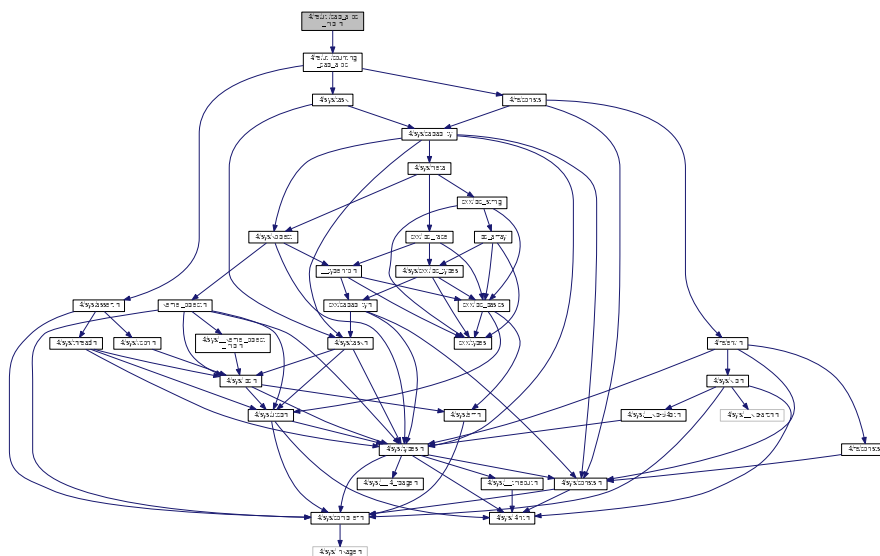
```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008 * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00009 * economic rights: Technische Universität Dresden (Germany)
00010 *
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024
00025 #pragma once
00026
00027 #include <l4/sys/task>
00028
00029 namespace L4Re { namespace Util {
00030
00038 L4_CV static inline l4_msgtag_t cap_release(L4::Cap<void> cap)
00039 {
00040 return l4_task_unmap(L4_BASE_TASK_CAP,
00041 l4_obj_fpage(cap.cap(), 0, L4_FPAGE_RWX),
00042 L4_FP_ALL_SPACES);
00043 }
00044
00045 }}
```

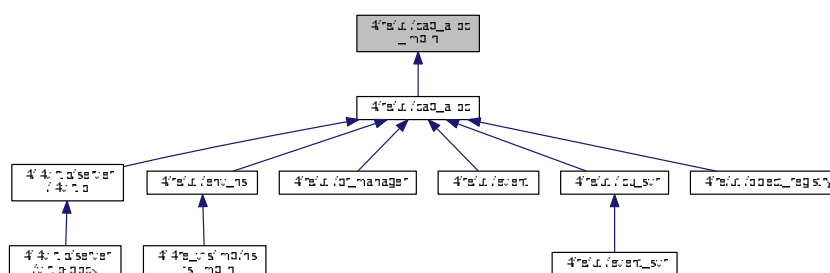
### 15.235 l4/re/util/cap\_alloc\_impl.h File Reference

Capability allocator implementation.

```
#include <l4/re/util/counting_cap_alloc>
```



This graph shows which files directly or indirectly include this file:



## Namespaces

- L4Re
- L4Re C++ Interfaces.*

### 15.235.1 Detailed Description

### Capability allocator implementation.

Definition in file [cap\\_alloc\\_impl.h](#).

## 15.236 cap\_alloc\_impl.h

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008 * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00009 * economic rights: Technische Universität Dresden (Germany)
00010 *
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024
00025 #pragma once
00026
00027 // #define L4RE_STATIC_CAP_ALLOC
00028 #if defined(L4RE_STATIC_CAP_ALLOC)
00029
00030 #include <l4/re/util/bitmap_cap_alloc>
00031
00032 namespace L4Re { namespace Util {
00033
00034 typedef Cap_alloc_base _Cap_alloc;
00035
00036 }}
00037
00038 #else
00039 #include <l4/re/util/counting_cap_alloc>
00040
00041 namespace L4Re { namespace Util {
00042
00043 typedef Counting_cap_alloc<L4Re::Util::Counter<unsigned char> > _Cap_alloc;
00044
00045 }}
00046 #endif
00047
00048

```

## 15.237 l4/re/util/counting\_cap\_alloc File Reference

Reference-counting capability allocator.

```

#include <l4/sys/task>
#include <l4/sys/assert.h>
#include <l4/re/consts>

```

- `class L4Re::Util::Counting_cap_alloc< COUNTERTYPE >`  
*Reference-counting cap allocator.*

- L4Re

*L4Re C++ Interfaces.*

Definition in file [counting\\_cap\\_alloc](#).

## 15.238 counting\_cap\_alloc

```

00001 // -*- Mode: C++ -*-
00002 // vim:ft=cpp
00007 /*
00008 * (c) 2008-2010 Alexander Warg <warg@os.inf.tu-dresden.de>
00009 * economic rights: Technische Universität Dresden (Germany)
00010 *
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024
00025 #pragma once
00026
00027 #include <l4/sys/task>
00028 #include <l4/sys/assert.h>
00029 #include <l4/re/consts>
00030
00031 namespace L4Re { namespace Util {
00032
00033 template< typename COUNTER = unsigned char >
00034 struct Counter
00035 {
00036 typedef COUNTER Type;
00037 Type _cnt;
00038
00039 static Type nil() { return 0; }
00040
00041 void free() { _cnt = 0; }
00042 bool is_free() const { return _cnt == 0; }
00043 void inc() { ++_cnt; }
00044 Type dec() { return --_cnt; }
00045 void alloc() { _cnt = 1; }
00046 };
00047
00055 template<typename COUNTERTYPE = L4Re::Util::Counter<unsigned char> >
00056 class Counting_cap_alloc
00057 {
00058 private:
00059 void operator = (Counting_cap_alloc const &) { }
00060 typedef COUNTERTYPE Counter;
00061
00062 COUNTERTYPE *_items;
00063 long _free_hint;
00064 long _bias;
00065 long _capacity;
00066
00067 public:
00068
00070 template <unsigned COUNT>
00071 struct Counter_storage
00072 {
00073 COUNTERTYPE _buf[COUNT];
00074 typedef COUNTERTYPE Buf_type[COUNT];
00075 enum { Size = COUNT };
00076 };
00077
00078 protected:
00079
00085 Counting_cap_alloc() throw()
00086 : _items(0), _free_hint(0), _bias(0), _capacity(0)
00087 {}
00088
00096 void setup(void *m, long capacity, long bias) throw()
00097 {
00098 _items = (Counter*)m;
00099 _capacity = capacity;
00100 _bias = bias;
00101 }
00102
00103 public:
00110 L4::Cap<void> alloc() throw()
00111 {
00112 if (_free_hint >= _capacity)
00113 return L4::Cap_base::Invalid;

```



```

00114
00115 for (long i = _free_hint; i < _capacity; ++i)
00116 {
00117 if (_items[i].is_free())
00118 {
00119 _items[i].alloc();
00120 _free_hint = i + 1;
00121
00122 return L4::Cap<void>((i + _bias) << L4_CAP_SHIFT);
00123 }
00124 }
00125
00126 return L4::Cap<void>::Invalid;
00127 }
00128
00130 template <typename T>
00131 L4::Cap<T> alloc() throw()
00132 {
00133 return L4::cap_cast<T>(alloc());
00134 }
00135
00136
00146 void take(L4::Cap<void> cap) throw()
00147 {
00148 long c = cap.cap() >> L4_CAP_SHIFT;
00149 if (c < _bias)
00150 return;
00151
00152 c -= _bias;
00153 if (c >= _capacity)
00154 return;
00155
00156 _items[c].inc();
00157 }
00158
00159
00172 bool free(L4::Cap<void> cap, l4_cap_idx_t task =
L4_INVALID_CAP,
00173 unsigned unmap_flags = L4_FP_ALL_SPACES) throw()
00174 {
00175 long c = cap.cap() >> L4_CAP_SHIFT;
00176 if (c < _bias)
00177 return false;
00178
00179 c -= _bias;
00180
00181 if (c >= _capacity)
00182 return false;
00183
00184 l4_assert(!_items[c].is_free());
00185
00186 if (task != L4_INVALID_CAP)
00187 l4_task_unmap(task, cap.fpage(), unmap_flags);
00188
00189 if (c < _free_hint)
00190 _free_hint = c;
00191
00192 _items[c].free();
00193
00194 return true;
00195 }
00196
00211 bool release(L4::Cap<void> cap, l4_cap_idx_t task =
L4_INVALID_CAP,
00212 unsigned unmap_flags = L4_FP_ALL_SPACES) throw()
00213 {
00214 long c = cap.cap() >> L4_CAP_SHIFT;
00215 if (c < _bias)
00216 return false;
00217
00218 c -= _bias;
00219
00220 if (c >= _capacity)
00221 return false;
00222
00223 l4_assert(!_items[c].is_free());
00224
00225 if (_items[c].dec() == Counter::nil())
00226 {
00227 if (task != L4_INVALID_CAP)
00228 l4_task_unmap(task, cap.fpage(), unmap_flags);
00229
00230 if (c < _free_hint)
00231 _free_hint = c;
00232
00233 return true;
00234 }

```

```

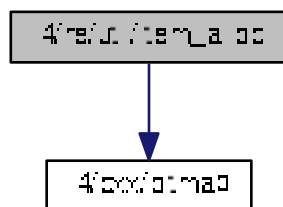
00235 return false;
00236 }
00237
00238
00242 long last() throw()
00243 {
00244 return _capacity + _bias - 1;
00245 }
00246 };
00247
00248 }}
00249

```

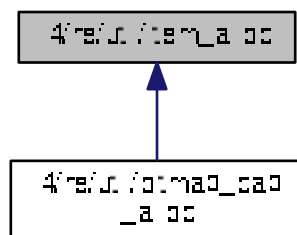
## 15.239 l4/re/util/item\_alloc File Reference

Item allocator.

```
#include <l4/cxx/bitmap>
Include dependency graph for item_alloc:
```



This graph shows which files directly or indirectly include this file:



## Data Structures

- class [L4Re::Util::Item\\_alloc\\_base](#)  
*Item allocator.*

## Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*

### 15.239.1 Detailed Description

Item allocator.

Definition in file [item\\_alloc](#).

## 15.240 item\_alloc

```

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 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025
00026 #pragma once
00027
00028 #include <l4/cxx/bitmap>
00029
00030 namespace L4Re { namespace Util {
00031
00032 using cxx::Bitmap_base;
00033 using cxx::Bitmap;
00034
00038 class Item_alloc_base
00039 {
00040 private:
00041 long _capacity;
00042 long _free_hint;
00043 Bitmap_base _bits;
00044
00045 public:
00046 bool is_allocated(long item) const throw()
00047 { return _bits[item]; }
00048
00049 long hint() const { return _free_hint; }
00050
00051 bool alloc(long item) throw()
00052 {
00053 if (!_bits[item])
00054 {
00055 _bits.set_bit(item);
00056 return true;
00057 }
00058 return false;
00059 }
00060
00061 void free(long item) throw()
00062 {
00063 if (item < _free_hint)
00064 _free_hint = item;
00065
00066 _bits.clear_bit(item);
00067 }
00068

```

```

00069 Item_alloc_base(long size, void *mem) throw()
00070 : _capacity(size), _free_hint(0), _bits(mem)
00071 {}
00072
00073 long alloc() throw()
00074 {
00075 if (_free_hint >= _capacity)
00076 return -1;
00077
00078 long free = _bits.scan_zero(_capacity, _free_hint);
00079 if (free >= 0)
00080 {
00081 _bits.set_bit(free);
00082 _free_hint += 1;
00083 }
00084 return free;
00085 }
00086
00087 long size() const throw()
00088 {
00089 return _capacity;
00090 }
00091 };
00092
00093 template< long Bits >
00094 class Item_alloc : public Item_alloc_base
00095 {
00096 private:
00097 typename Bitmap_base::Word<Bits>::Type _bits[Bitmap_base::Word<Bits>::Size];
00098
00099 public:
00100 Item_alloc() throw() : Item_alloc_base(Bits, _bits) {}
00101 };
00102
00103 }

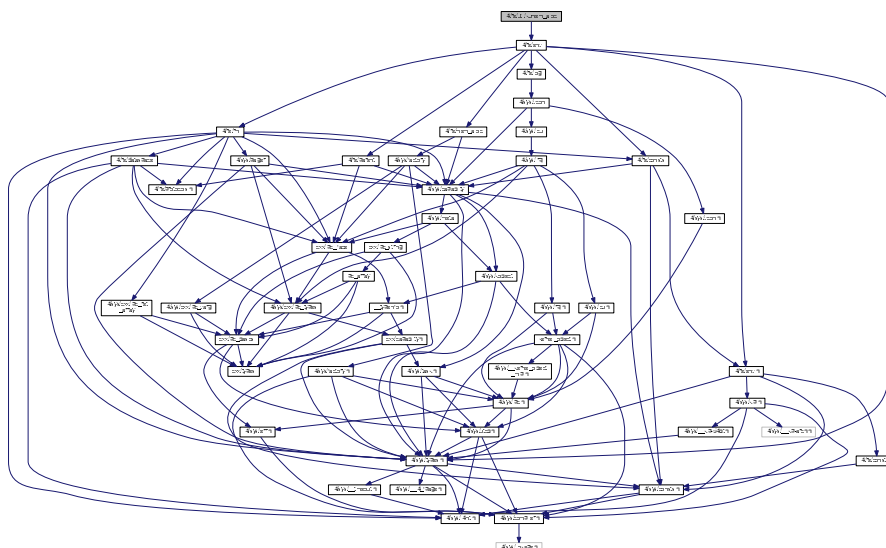
```

## 15.241 l4/re/util/kumem\_alloc File Reference

Kumem allocator helper.

```
#include <l4/re/env>
```

Include dependency graph for kumem\_alloc:



## Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*

## 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()) throw ()`

*Allocate state area.*

### 15.241.1 Detailed Description

Kumem allocator helper.

Definition in file [kumem\\_alloc](#).

## 15.242 kumem\_alloc

```

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 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025
00026 #pragma once
00027
00028 #include <l4/re/env>
00029
00030 namespace L4Re { namespace Util {
00031
00037
00048 int
00049 kumem_alloc(l4_addr_t *mem, unsigned pages_order,
00050 L4::Cap<L4::Task> task = L4Re::Env::env()->task(),
00051 L4::Cap<L4Re::Rm> rm = L4Re::Env::env()->rm()) throw();
00052
00054 }}
```

## 15.243 l4/re/util/region\_mapping File Reference

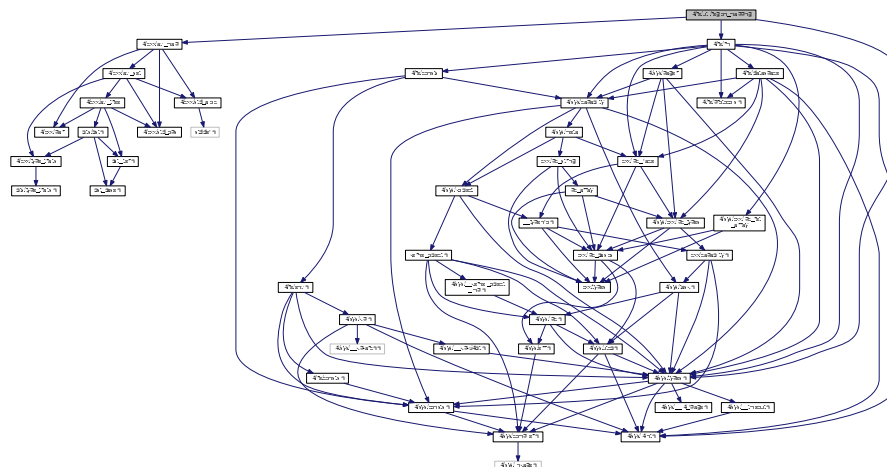
Region handling.

```

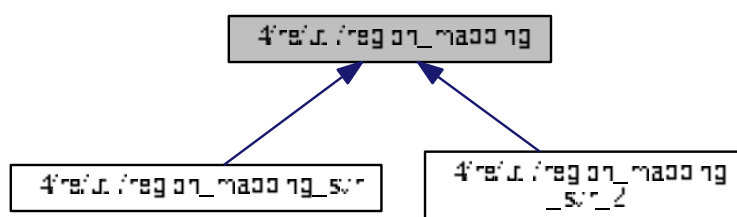
#include <l4/cxx/avl_map>
#include <l4/sys/types.h>
```

```
#include <l4/re/rm>
```

Include dependency graph for region\_mapping:



This graph shows which files directly or indirectly include this file:



## Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*

### 15.243.1 Detailed Description

Region handling.

Definition in file [region\\_mapping](#).

## 15.244 region\_mapping

```

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 * Björn Döbel <doebel@os.inf.tu-dresden.de>
00007 * economic rights: Technische Universität Dresden (Germany)
00008 *
00009 * This file is part of TUD:OS and distributed under the terms of the
00010 * GNU General Public License 2.
00011 * Please see the COPYING-GPL-2 file for details.
00012 *
00013 * As a special exception, you may use this file as part of a free software
00014 * library without restriction. Specifically, if other files instantiate
00015 * templates or use macros or inline functions from this file, or you compile
00016 * this file and link it with other files to produce an executable, this
00017 * file does not by itself cause the resulting executable to be covered by
00018 * the GNU General Public License. This exception does not however
00019 * invalidate any other reasons why the executable file might be covered by
00020 * the GNU General Public License.
00021 */
00022
00023 #pragma once
00024
00025 #include <l4/cxx/avl_map>
00026 #include <l4/sys/types.h>
00027 #include <l4/re/rm>
00028
00029 namespace L4Re { namespace Util {
00030 class Region
00031 {
00032 private:
00033 l4_addr_t _start, _end;
00034
00035 public:
00036 Region() throw() : _start(~0UL), _end(~0UL) {}
00037 Region(l4_addr_t addr) throw() : _start(addr), _end(addr) {}
00038 Region(l4_addr_t start, l4_addr_t end) throw()
00039 : _start(start), _end(end) {}
00040 l4_addr_t start() const throw() { return _start; }
00041 l4_addr_t end() const throw() { return _end; }
00042 unsigned long size() const throw() { return end() - start() + 1; }
00043 bool invalid() const throw() { return _start == ~0UL && _end == ~0UL; }
00044 bool operator < (Region const &o) const throw()
00045 { return end() < o.start(); }
00046 bool contains(Region const &o) const throw()
00047 { return o.start() >= start() && o.end() <= end(); }
00048 bool operator == (Region const &o) const throw()
00049 { return o.start() == start() && o.end() == end(); }
00050 ~Region() throw() {}
00051 };
00052
00053 template< typename DS, typename OPS >
00054 class Region_handler
00055 {
00056 private:
00057 l4_addr_t _offs;
00058 DS _mem;
00059 l4_cap_idx_t _client_cap = L4_INVALID_CAP;
00060 unsigned short _flags;
00061
00062 public:
00063 typedef DS Dataspace;
00064 typedef OPS Ops;
00065 typedef typename OPS::Map_result Map_result;
00066
00067 Region_handler() throw() : _offs(0), _mem(), _flags() {}
00068 Region_handler(Dataspace const &mem, l4_cap_idx_t client_cap,
00069 l4_addr_t offset = 0, unsigned flags = 0) throw()
00070 : _offs(offset), _mem(mem), _client_cap(client_cap), _flags(flags)
00071 {}
00072 Dataspace const &memory() const throw() { return _mem; }
00073 l4_cap_idx_t client_cap_idx() const throw() { return _client_cap; }
00074 l4_addr_t offset() const throw() { return _offs; }
00075 l4_addr_t is_ro() const throw() { return _flags & L4Re::Rm::Read_only; }
00076 unsigned long caching() const throw() { return _flags & L4Re::Rm::Caching; }
00077 unsigned flags() const throw() { return _flags; }
00078
00079 Region_handler operator + (long offset) const throw()
00080 { Region_handler n = *this; n._offs += offset; return n; }
00081
00082 void unmap(l4_addr_t va, l4_addr_t ds_offs, unsigned long size) const throw()
00083 { Ops::unmap(this, va, ds_offs, size); }

```

```

00089
00090 void free(l4_addr_t start, unsigned long size) const throw()
00091 { Ops::free(this, start, size); }
00092
00093 void take() const { Ops::take(this); }
00094 void release() const { Ops::release(this); }
00095
00096 int map(l4_addr_t addr, Region const &r, bool writable, Map_result *result) const
00097 { return Ops::map(this, addr, r, writable, result); }
00098
00099 };
00100
00101
00102 template< typename Hdlr, template<typename T> class Alloc >
00103 class Region_map
00104 {
00105 protected:
00106 typedef cxx::Avl_map< Region, Hdlr, cxx::Lt_functor, Alloc >
00107 Tree;
00108 Tree _rm;
00109 Tree _am;
00110 private:
00111 l4_addr_t _start;
00112 l4_addr_t _end;
00113 protected:
00114 void set_limits(l4_addr_t start, l4_addr_t end) throw()
00115 {
00116 _start = start;
00117 _end = end;
00118 }
00119 public:
00120 typedef typename Tree::Item_type Item;
00121 typedef typename Tree::Node Node;
00122 typedef typename Tree::Key_type Key_type;
00123 typedef Hdlr Region_handler;
00124
00125 typedef typename Tree::Iterator Iterator;
00126 typedef typename Tree::Const_iterator Const_iterator;
00127 typedef typename Tree::Rev_iterator Rev_iterator;
00128 typedef typename Tree::Const_rev_iterator Const_rev_iterator;
00129
00130 Iterator begin() throw() { return _rm.begin(); }
00131 Const_iterator begin() const throw() { return _rm.begin(); }
00132 Iterator end() throw() { return _rm.end(); }
00133 Const_iterator end() const throw() { return _rm.end(); }
00134
00135 Iterator area_begin() throw() { return _am.begin(); }
00136 Const_iterator area_begin() const throw() { return _am.begin(); }
00137 Iterator area_end() throw() { return _am.end(); }
00138 Const_iterator area_end() const throw() { return _am.end(); }
00139 Node area_find(Key_type const &c) const throw() { return _am.find_node(c); }
00140
00141 enum Attach_flags
00142 {
00143 None = 0,
00144 Search = L4Re::Rm::Search_addr,
00145 In_area = L4Re::Rm::In_area,
00146 };
00147
00148 l4_addr_t min_addr() const throw() { return _start; }
00149 l4_addr_t max_addr() const throw() { return _end; }
00150
00151 Region_map(l4_addr_t start, l4_addr_t end) throw() : _start(start), _end(end) {}
00152
00153 Node find(Key_type const &key) const throw()
00154 {
00155 Node n = _rm.find_node(key);
00156 if (!n)
00157 return Node();
00158
00159 // 'find' should find any region overlapping with the searched one, the
00160 // caller should check for further requirements
00161 if (0)
00162 if (!n->first.contains(key))
00163 return Node();
00164
00165 return n;
00166 }
00167
00168 Node lower_bound(Key_type const &key) const throw()
00169 {
00170 Node n = _rm.lower_bound_node(key);
00171 return n;
00172 }

```



```

00175 }
00176
00177 Node lower_bound_area(Key_type const &key) const throw()
00178 {
00179 Node n = _am.lower_bound_node(key);
00180 return n;
00181 }
00182
00183 l4_addr_t attach_area(l4_addr_t addr, unsigned long size,
00184 unsigned flags = None,
00185 unsigned char align = L4_PAGESHIFT) throw()
00186 {
00187 if (size < 2)
00188 return L4_INVALID_ADDR;
00189
00190 Region c;
00191
00192 if (!(flags & Search))
00193 {
00194 c = Region(addr, addr + size - 1);
00195 Node r = _am.find_node(c);
00196 if (r)
00197 return L4_INVALID_ADDR;
00198 }
00199
00200 while (flags & Search)
00201 {
00202 if (addr < min_addr() || (addr + size - 1) > max_addr())
00203 addr = min_addr();
00204 addr = find_free(addr, max_addr(), size, align, flags);
00205 if (addr == L4_INVALID_ADDR)
00206 return L4_INVALID_ADDR;
00207
00208 c = Region(addr, addr + size - 1);
00209 Node r = _am.find_node(c);
00210 if (!r)
00211 break;
00212
00213 if (r->first.end() >= max_addr())
00214 return L4_INVALID_ADDR;
00215
00216 addr = r->first.end() + 1;
00217 }
00218
00219 if (_am.insert(c, Hdlr(typename Hdlr::Dataspace(), 0, flags)).second == 0)
00220 return addr;
00221
00222 return L4_INVALID_ADDR;
00223 }
00224
00225 bool detach_area(l4_addr_t addr) throw()
00226 {
00227 if (_am.remove(addr))
00228 return false;
00229
00230 return true;
00231 }
00232
00233 void *attach(void *addr, unsigned long size, Hdlr const &hdlr,
00234 unsigned flags = None, unsigned char align = L4_PAGESHIFT) throw()
00235 {
00236 if (size < 2)
00237 return L4_INVALID_PTR;
00238
00239 l4_addr_t end = max_addr();
00240 l4_addr_t beg = (l4_addr_t)addr;
00241
00242 if (flags & In_area)
00243 {
00244 Node r = _am.find_node(Region(beg, beg + size - 1));
00245 if (!r || (r->second.flags() & L4Re::Rm::Reserved))
00246 return L4_INVALID_PTR;
00247
00248 end = r->first.end();
00249 }
00250
00251 if (flags & Search)
00252 {
00253 beg = find_free(beg, end, size, align, flags);
00254 if (beg == L4_INVALID_ADDR)
00255 return L4_INVALID_PTR;
00256 }
00257
00258 if (!(flags & (Search | In_area)) && _am.find_node(Region(beg, beg + size - 1)))
00259 return L4_INVALID_PTR;
00260
00261

```

```

00262 if (beg < min_addr() || beg + size - 1 > end)
00263 return L4_INVALID_PTR;
00264
00265 if (_rm.insert(Region(beg, beg + size - 1), hdlr).second == 0)
00266 return (void*)beg;
00267
00268 return L4_INVALID_PTR;
00269 }
00270
00271 int detach(void *addr, unsigned long sz, unsigned flags,
00272 Region *reg, Hdlr *hdlr) throw()
00273 {
00274 Region dr((l4_addr_t)addr, (l4_addr_t)addr + sz - 1);
00275 Region res(~0UL, 0);
00276
00277 Node r = find(dr);
00278 if (!r)
00279 return -L4_ENOENT;
00280
00281 Region g = r->first;
00282 Hdlr const &h = r->second;
00283
00284 if (flags & L4Re::Rm::Detach_overlap || dr.contains(g))
00285 {
00286 if (_rm.remove(g))
00287 return -L4_ENOENT;
00288
00289 if (!(flags & L4Re::Rm::Detach_keep) && (h.flags() &
L4Re::Rm::Detach_free))
00290 h.free(0, g.size());
00291
00292 if (hdlr) *hdlr = h;
00293 if (reg) *reg = g;
00294
00295 if (find(dr))
00296 return Rm::Detached_ds | Rm::Detach_again;
00297 else
00298 return Rm::Detached_ds;
00299 }
00300 else if (dr.start() <= g.start())
00301 {
00302 // move the start of a region
00303
00304 if (!(flags & L4Re::Rm::Detach_keep) && (h.flags() & L4Re::Rm::Detach_free))
00305 h.free(0, dr.end() + 1 - g.start());
00306
00307 unsigned long sz = dr.end() + 1 - g.start();
00308 Item *cn = const_cast<Item*>((Item const *)r);
00309 cn->first = Region(dr.end() + 1, g.end());
00310 cn->second = cn->second + sz;
00311 if (hdlr) *hdlr = Hdlr();
00312 if (reg) *reg = Region(g.start(), dr.end());
00313 if (find(dr))
00314 return Rm::Kept_ds | Rm::Detach_again;
00315 else
00316 return Rm::Kept_ds;
00317 }
00318 else if (dr.end() >= g.end())
00319 {
00320 // move the end of a region
00321
00322 if (!(flags & L4Re::Rm::Detach_keep) && (h.flags() & L4Re::Rm::Detach_free))
00323 h.free(dr.start() - g.start(), g.end() + 1 - dr.start());
00324
00325 Item *cn = const_cast<Item*>((Item const *)r);
00326 cn->first = Region(g.start(), dr.start() - 1);
00327 if (hdlr) *hdlr = Hdlr();
00328 if (reg) *reg = Region(dr.start(), g.end());
00329
00330 if (find(dr))
00331 return Rm::Kept_ds | Rm::Detach_again;
00332 else
00333 return Rm::Kept_ds;
00334 }
00335 else if (g.contains(dr))
00336 {
00337 // split a single region that contains the new region
00338
00339 if (!(flags & L4Re::Rm::Detach_keep) && (h.flags() & L4Re::Rm::Detach_free))
00340 h.free(dr.start() - g.start(), dr.size());
00341
00342 // first move the end off the existing region before the new one
00343 const_cast<Item*>((Item const *)r)->first = Region(g.start(), dr.start() - 1);
00344
00345 int err;
00346
00347 // insert a second region for the remaining tail of

```

```

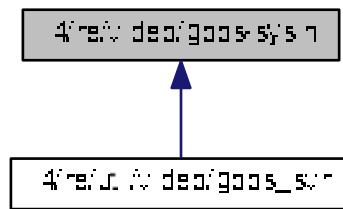
00348 // the old existing region
00349 err = _rm.insert(Region(dr.end() + 1, g.end()), h + (dr.end() + 1 - g.start()).second;
00350
00351 if (err)
00352 return err;
00353
00354 if (hdlr) *hdlr = h;
00355 if (reg) *reg = dr;
00356 return Rm::Split_ds;
00357 }
00358 return -L4_ENOENT;
00359 }
00360
00361 l4_addr_t find_free(l4_addr_t start, l4_addr_t end,
00362 l4_addr_t size,
00363 unsigned char align, unsigned flags) const throw();
00364 };
00365
00366
00367 template< typename Hdlr, template<typename T> class Alloc >
00368 l4_addr_t
00369 Region_map<Hdlr, Alloc>::find_free(l4_addr_t start, l4_addr_t end,
00370 unsigned long size, unsigned char align, unsigned flags) const throw()
00371 {
00372 l4_addr_t addr = start;
00373
00374 if (addr == ~0UL || addr < min_addr() || addr >= end)
00375 addr = min_addr();
00376
00377 addr = l4_round_size(addr, align);
00378 Node r;
00379
00380 for(;;)
00381 {
00382 if (addr > 0 && addr - 1 > end - size)
00383 return L4_INVALID_ADDR;
00384
00385 Region c(addr, addr + size - 1);
00386 r = _rm.find_node(c);
00387
00388 if (!r)
00389 {
00390 if (!(flags & In_area) && (r = _am.find_node(c)))
00391 {
00392 if (r->first.end() > end - size)
00393 return L4_INVALID_ADDR;
00394
00395 addr = l4_round_size(r->first.end() + 1, align);
00396 continue;
00397 }
00398 break;
00399 }
00400 else if (r->first.end() > end - size)
00401 return L4_INVALID_ADDR;
00402
00403 addr = l4_round_size(r->first.end() + 1, align);
00404 }
00405
00406 if (!r)
00407 return addr;
00408
00409 return L4_INVALID_ADDR;
00410 }
00411
00412 }

```

## 15.245 l4/re/video/goos-sys.h File Reference

Goos protocol definition.

This graph shows which files directly or indirectly include this file:



## Namespaces

- [L4Re](#)  
*L4Re C++ Interfaces.*

## Enumerations

- enum [L4Re::Video::Goos\\_::Opcodes](#)  
*Frame buffer communication-protocol opcodes.*

### 15.245.1 Detailed Description

Goos protocol definition.

Definition in file [goos-sys.h](#).

## 15.246 goos-sys.h

```

00001 #pragma once
00002
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 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025
00026 namespace L4Re { namespace Video {

```

```

00027 namespace Goos_
00028 {
00034 enum Opcodes
00035 {
00036 Info, Get_buffer, Create_buffer, Create_view,
00037 Delete_buffer, Delete_view,
00038 View_info, View_set_info, View_stack, View_refresh,
00039 Screen_refresh
00040 };
00041 };
00042 }

```

## 15.247 l4/shmc/shmc.h File Reference

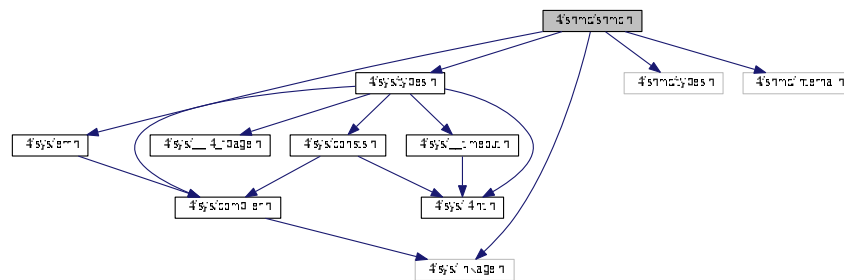
Shared memory library header file.

```

#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:



## Functions

- long [l4shmc\\_create](#) (const char \*shmc\_name, [l4\\_umword\\_t](#) shm\_size)  
*Create a shared memory area.*
- long [l4shmc\\_attach](#) (const char \*shmc\_name, [l4shmc\\_area\\_t](#) \*shmarea)  
*Attach to a shared memory area.*
- long [l4shmc\\_attach\\_to](#) (const char \*shmc\_name, [l4\\_umword\\_t](#) timeout\_ms, [l4shmc\\_area\\_t](#) \*shmarea)  
*Attach to a shared memory area, with limited waiting.*
- long [l4shmc\\_add\\_chunk](#) ([l4shmc\\_area\\_t](#) \*shmarea, const char \*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, const char \*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\\_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, const char \*chunk\_name, l4shmc\_chunk\_t \*chunk)  
*Get chunk out of shared memory area.*
- long [l4shmc\\_get\\_chunk\\_to](#) (l4shmc\_area\_t \*shmarea, const char \*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 \*shmarea, const char \*\*chunk\_name, long offs)  
*Iterate over names of all existing chunks.*
- long [l4shmc\\_attach\\_signal](#) (l4shmc\_area\_t \*shmarea, const char \*signal\_name, [l4\\_cap\\_idx\\_t](#) thread, l4shmc\_signal\_t \*signal)  
*Attach to signal.*
- long [l4shmc\\_attach\\_signal\\_to](#) (l4shmc\_area\_t \*shmarea, const char \*signal\_name, [l4\\_cap\\_idx\\_t](#) thread, [l4\\_umword\\_t](#) timeout\_ms, l4shmc\_signal\_t \*signal)  
*Attach to signal, with timeout.*
- long [l4shmc\\_get\\_signal\\_to](#) (l4shmc\_area\_t \*shmarea, const char \*signal\_name, [l4\\_umword\\_t](#) timeout\_ms, 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 \*chunk)  
*Check whether data is available.*
- long [l4shmc\\_is\\_chunk\\_clear](#) (l4shmc\_chunk\_t \*chunk)  
*Check whether chunk is free.*
- void \* [l4shmc\\_chunk\\_ptr](#) (l4shmc\_chunk\_t \*chunk)  
*Get data pointer to chunk.*
- long [l4shmc\\_chunk\\_size](#) (l4shmc\_chunk\_t \*chunk)

- Get current size of a chunk.*

  - long [l4shmc\\_chunk\\_capacity](#) (l4shmc\_chunk\_t \*chunk)
- Get capacity of a chunk.*

  - l4shmc\_signal\_t \* [l4shmc\\_chunk\\_signal](#) (l4shmc\_chunk\_t \*chunk)
- Get the signal of a chunk.*

  - [l4\\_cap\\_idx\\_t](#) [l4shmc\\_signal\\_cap](#) (l4shmc\_signal\_t \*signal)
- Get the signal capability of a signal.*

  - long [l4shmc\\_check\\_magic](#) (l4shmc\_chunk\_t \*chunk)
- Check magic value of a chunk.*

  - long [l4shmc\\_area\\_size](#) (l4shmc\_area\_t \*shmarea)
- Get size of shared memory area.*

  - long [l4shmc\\_area\\_size\\_free](#) (l4shmc\_area\_t \*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.*

### 15.247.1 Detailed Description

Shared memory library header file.

Definition in file [shmc.h](#).

## 15.248 shmc.h

```

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/linkage.h>
00016 #include <l4/sys/types.h>
00017 #include <l4/sys/err.h>
00018
00069 #define __INCLUDED_FROM_L4SHMC_H__
00070 #include <l4/shmc/types.h>
00071
00072 __BEGIN_DECLS
00073
00084 L4_CV long
00085 l4shmc_create(const char *shmc_name, l4_umword_t shm_size);
00086
00098 L4_CV L4_INLINE long
00099 l4shmc_attach(const char *shmc_name, l4shmc_area_t *shmarea);
00100
00113 L4_CV long
00114 l4shmc_attach_to(const char *shmc_name, l4_umword_t timeout_ms,
00115 l4shmc_area_t *shmarea);
00116
00129 L4_CV long
00130 l4shmc_add_chunk(l4shmc_area_t *shmarea,
00131 const char *chunk_name,
00132 l4_umword_t chunk_capacity,
00133 l4shmc_chunk_t *chunk);
00134
00146 L4_CV long
00147 l4shmc_add_signal(l4shmc_area_t *shmarea,
```

```

00148 const char *signal_name,
00149 l4shmc_signal_t *signal);
00150
00160 L4_CV L4_INLINE long
00161 l4shmc_trigger(l4shmc_signal_t *signal);
00162
00172 L4_CV L4_INLINE long
00173 l4shmc_chunk_try_to_take(l4shmc_chunk_t *chunk);
00174
00185 L4_CV L4_INLINE long
00186 l4shmc_chunk_ready(l4shmc_chunk_t *chunk, l4_umword_t size);
00187
00198 L4_CV L4_INLINE long
00199 l4shmc_chunk_ready_sig(l4shmc_chunk_t *chunk, l4_umword_t size);
00200
00212 L4_CV L4_INLINE long
00213 l4shmc_get_chunk(l4shmc_area_t *shmarea,
00214 const char *chunk_name,
00215 l4shmc_chunk_t *chunk);
00216
00230 L4_CV long
00231 l4shmc_get_chunk_to(l4shmc_area_t *shmarea,
00232 const char *chunk_name,
00233 l4_umword_t timeout_ms,
00234 l4shmc_chunk_t *chunk);
00235
00249 L4_CV long
00250 l4shmc_iterate_chunk(l4shmc_area_t *shmarea, const char **chunk_name,
00251 long offs);
00252
00265 L4_CV L4_INLINE long
00266 l4shmc_attach_signal(l4shmc_area_t *shmarea,
00267 const char *signal_name,
00268 l4_cap_idx_t thread,
00269 l4shmc_signal_t *signal);
00270
00285 L4_CV long
00286 l4shmc_attach_signal_to(l4shmc_area_t *shmarea,
00287 const char *signal_name,
00288 l4_cap_idx_t thread,
00289 l4_umword_t timeout_ms,
00290 l4shmc_signal_t *signal);
00291
00305 L4_CV long
00306 l4shmc_get_signal_to(l4shmc_area_t *shmarea,
00307 const char *signal_name,
00308 l4_umword_t timeout_ms,
00309 l4shmc_signal_t *signal);
00310
00311 L4_CV L4_INLINE long
00312 l4shmc_get_signal(l4shmc_area_t *shmarea,
00313 const char *signal_name,
00314 l4shmc_signal_t *signal);
00315
00316
00330 L4_CV long
00331 l4shmc_enable_signal(l4shmc_signal_t *signal);
00332
00346 L4_CV long
00347 l4shmc_enable_chunk(l4shmc_chunk_t *chunk);
00348
00358 L4_CV L4_INLINE long
00359 l4shmc_wait_any(l4shmc_signal_t **retsignal);
00360
00374 L4_CV L4_INLINE long
00375 l4shmc_wait_any_try(l4shmc_signal_t **retsignal);
00376
00391 L4_CV long
00392 l4shmc_wait_any_to(l4_timeout_t timeout, l4shmc_signal_t **retsignal);
00393
00403 L4_CV L4_INLINE long
00404 l4shmc_wait_signal(l4shmc_signal_t *signal);
00405
00416 L4_CV long
00417 l4shmc_wait_signal_to(l4shmc_signal_t *signal, l4_timeout_t timeout);
00418
00432 L4_CV L4_INLINE long
00433 l4shmc_wait_signal_try(l4shmc_signal_t *signal);
00434
00444 L4_CV L4_INLINE long
00445 l4shmc_wait_chunk(l4shmc_chunk_t *chunk);
00446
00461 L4_CV long
00462 l4shmc_wait_chunk_to(l4shmc_chunk_t *chunk, l4_timeout_t timeout);
00463
00477 L4_CV L4_INLINE long
00478 l4shmc_wait_chunk_try(l4shmc_chunk_t *chunk);

```



```

00479
00489 L4_CV L4_INLINE long
00490 l4shmc_chunk_consumed(l4shmc_chunk_t *chunk);
00491
00501 L4_CV long
00502 l4shmc_connect_chunk_signal(l4shmc_chunk_t *chunk,
00503 l4shmc_signal_t *signal);
00504
00514 L4_CV L4_INLINE long
00515 l4shmc_is_chunk_ready(l4shmc_chunk_t *chunk);
00516
00526 L4_CV L4_INLINE long
00527 l4shmc_is_chunk_clear(l4shmc_chunk_t *chunk);
00528
00538 L4_CV L4_INLINE void *
00539 l4shmc_chunk_ptr(l4shmc_chunk_t *chunk);
00540
00550 L4_CV L4_INLINE long
00551 l4shmc_chunk_size(l4shmc_chunk_t *chunk);
00552
00562 L4_CV L4_INLINE long
00563 l4shmc_chunk_capacity(l4shmc_chunk_t *chunk);
00564
00574 L4_CV L4_INLINE l4shmc_signal_t *
00575 l4shmc_chunk_signal(l4shmc_chunk_t *chunk);
00576
00585 L4_CV L4_INLINE l4_cap_idx_t
00586 l4shmc_signal_cap(l4shmc_signal_t *signal);
00587
00597 L4_CV L4_INLINE long
00598 l4shmc_check_magic(l4shmc_chunk_t *chunk);
00599
00609 L4_CV L4_INLINE long
00610 l4shmc_area_size(l4shmc_area_t *shmarea);
00611
00623 L4_CV long
00624 l4shmc_area_size_free(l4shmc_area_t *shmarea);
00625
00632 L4_CV long
00633 l4shmc_area_overhead(void);
00634
00642 L4_CV long
00643 l4shmc_chunk_overhead(void);
00644
00645 #include <l4/shmc/internal.h>
00646
00647 __END_DECLS

```

## 15.249 l4/sigma0/sigma0.h File Reference

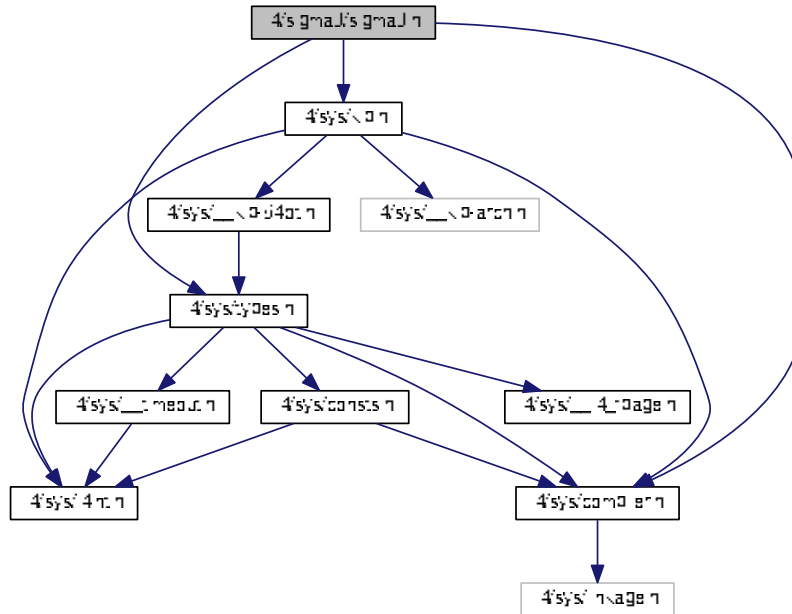
Sigma0 interface.

```

#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_TBUF 0xB0`  
*TBUF.*
- `#define SIGMA0_REQ_ID_DEBUG_DUMP 0xC0`  
*Debug dump.*
- `#define SIGMA0_REQ_ID_NEW_CLIENT 0xD0`  
*New client.*
- `#define SIGMA0_IS_MAGIC_REQ(d1) ((d1 & SIGMA0_REQ_MASK) == SIGMA0_REQ_MAGIC)`  
*Check if magic.*

- `#define SIGMA0_REQ(x) (SIGMA0_REQ_MAGIC + SIGMA0_REQ_ID_ ## 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_TBUF (SIGMA0_REQ(TBUF))`  
*TBUF.*
- `#define SIGMA0_REQ_DEBUG_DUMP (SIGMA0_REQ(DEBUG_DUMP))`  
*Debug dump.*
- `#define SIGMA0_REQ_NEW_CLIENT (SIGMA0_REQ(NEW_CLIENT))`  
*New client.*

## Enumerations

- `enum l4sigma0_return_flags_t {`  
`L4SIGMA0_OK, L4SIGMA0_NOTALIGNED, L4SIGMA0_IPCERROR, L4SIGMA0_NOFPAGE ,`  
`L4SIGMA0_SMALLERFPAGE }`  
*Return flags of libsigma0 functions.*

## Functions

- `l4_kernel_info_t * l4sigma0_map_kip (l4_cap_idx_t sigma0, void *addr, unsigned log2_size)`  
*Map the kernel info page from pager 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.*
- `int l4sigma0_map_tbuf (l4_cap_idx_t sigma0, l4_addr_t virt)`  
*Request Fiasco trace buffer.*
- `void l4sigma0_debug_dump (l4_cap_idx_t sigma0)`  
*Request sigma0 to dump internal debug information.*
- `int l4sigma0_new_client (l4_cap_idx_t sigma0, l4_cap_idx_t gate)`  
*Create a new IPC gate for a new Sigma0 client.*
- `char const * l4sigma0_map_errstr (int err)`  
*Get a user readable error messages for the return codes.*

### 15.249.1 Detailed Description

Sigma0 interface.

Definition in file [sigma0.h](#).

## 15.250 sigma0.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 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU Lesser General Public License 2.1.
00013 * Please see the COPYING-LGPL-2.1 file for details.
00014 */
00015 #ifndef __L4_SIGMA0_SIGMA0_H
00016 #define __L4_SIGMA0_SIGMA0_H
00017
00026 #include <l4/sys/compiler.h>
00027 #include <l4/sys/types.h>
00028 #include <l4/sys/kip.h>
00029
00036 #undef SIGMA0_REQ_MAGIC
00037 #undef SIGMA0_REQ_MASK
00038
00039 # define SIGMA0_REQ_MAGIC ~0xFFUL
00040 # define SIGMA0_REQ_MASK ~0xFFUL
00042 /* Starting with 0x60 allows to detect components which still use the old
00043 * constants (0x00 ... 0x50) */
00044 #define SIGMA0_REQ_ID_MASK 0xF0
00045 #define SIGMA0_REQ_ID_FPAGE_RAM 0x60
00046 #define SIGMA0_REQ_ID_FPAGE_IOMEM 0x70
00047 #define SIGMA0_REQ_ID_FPAGE_IOMEM_CACHED 0x80
00048 #define SIGMA0_REQ_ID_FPAGE_ANY 0x90
00049 #define SIGMA0_REQ_ID_KIP 0xA0
00050 #define SIGMA0_REQ_ID_TBUF 0xB0
00051 #define SIGMA0_REQ_ID_DEBUG_DUMP 0xC0
00052 #define SIGMA0_REQ_ID_NEW_CLIENT 0xD0
00054 #define SIGMA0_IS_MAGIC_REQ(d1) \
00055 ((d1 & SIGMA0_REQ_MASK) == SIGMA0_REQ_MAGIC)
00057 #define SIGMA0_REQ(x) \
00058 (SIGMA0_REQ_MAGIC + SIGMA0_REQ_ID_ ## x)
00060 /* Use these constants in your code! */
00061 #define SIGMA0_REQ_FPAGE_RAM (SIGMA0_REQ(FPAGE_RAM))
00062 #define SIGMA0_REQ_FPAGE_IOMEM (SIGMA0_REQ(FPAGE_IOMEM))
00063 #define SIGMA0_REQ_FPAGE_IOMEM_CACHED (SIGMA0_REQ(FPAGE_IOMEM_CACHED))
00064 #define SIGMA0_REQ_FPAGE_ANY (SIGMA0_REQ(FPAGE_ANY))
00065 #define SIGMA0_REQ_KIP (SIGMA0_REQ(KIP))
00066 #define SIGMA0_REQ_TBUF (SIGMA0_REQ(TBUF))
00067 #define SIGMA0_REQ_DEBUG_DUMP (SIGMA0_REQ(DEBUG_DUMP))
00068 #define SIGMA0_REQ_NEW_CLIENT (SIGMA0_REQ(NEW_CLIENT))
00070
00071
00075
00079 enum l4sigma0_return_flags_t {
00080 L4SIGMA0_OK,
00081 L4SIGMA0_NOTALIGNED,
00082 L4SIGMA0_IPCERROR,
00083 L4SIGMA0_NOFPAGE,
00084 L4SIGMA0_4,
00085 L4SIGMA0_5,
00086 L4SIGMA0_SMALLERFPAGE,
00087 };
00088
00089 EXTERN_C_BEGIN
00090
00099 L4_CV l4_kernel_info_t *
00100 l4sigma0_map_kip(l4_cap_idx_t sigma0, void *addr, unsigned log2_size);
00101
00115 L4_CV int l4sigma0_map_mem(l4_cap_idx_t sigma0,
00116 l4_addr_t phys, l4_addr_t virt,
00117 l4_addr_t size);
00117
00141 L4_CV int l4sigma0_map_iomem(l4_cap_idx_t sigma0,
00142 l4_addr_t phys,
00143 l4_addr_t virt, l4_addr_t size, int cached);
00163 L4_CV int l4sigma0_map_anypage(l4_cap_idx_t sigma0,
00164 l4_addr_t map_area,
00165 unsigned log2_map_size, l4_addr_t *base,
00166 unsigned sz);
00166
00179 L4_CV int l4sigma0_map_tbuf(l4_cap_idx_t sigma0,
00180 l4_addr_t virt);
00180
00189 L4_CV void l4sigma0_debug_dump(l4_cap_idx_t sigma0);
00190
00196 L4_CV int l4sigma0_new_client(l4_cap_idx_t sigma0,
00197 l4_cap_idx_t gate);
00197

```



### 15.251.1 Detailed Description

Low-level kernel debugger functions.

Definition in file [\\_\\_kernel\\_object\\_impl.h](#).

## 15.252 [\\_\\_kernel\\_object\\_impl.h](#)

```

00001
00005 #pragma once
00006
00007 #include <l4/sys/ipc.h>
00008
00009 L4_INLINE l4_msgtag_t
00010 l4_invoke_debugger(l4_cap_idx_t obj, l4_msgtag_t tag,
00011 l4_utcb_t *utcb) L4_NOTHROW
00012 {
00013 l4_msgtag_t t2;
00014 l4_msg_regs_t *mr = l4_utcb_mr_u(utcb);
00015 if (l4_is_invalid_cap(obj))
00016 return l4_msgtag(~L4_EINVAL, 0, 0, 0);
00017
00018 mr->mr[0] += 0x100;
00019 mr->mr[l4_msgtag_words(tag)] = L4_ITEM_MAP;
00020 mr->mr[l4_msgtag_words(tag) + 1] = l4_obj_fpage(obj, 0,
00021 L4_FPAGE_RWX).raw;
00022 t2 = l4_msgtag(L4_PROTO_DEBUGGER, l4_msgtag_words(tag),
00023 1, l4_msgtag_flags(tag));
00024 return l4_ipc_call(L4_BASE_DEBUGGER_CAP, utcb, t2,
00025 L4_IPC_NEVER);
00026 }

```

## 15.253 [l4/sys/\\_\\_ktrace-impl.h](#) File Reference

[L4](#) kernel event tracing.

```

#include <l4/sys/types.h>
#include <l4/sys/kdebug.h>

```



- 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.*

### 15.253.1 Detailed Description

[L4](#) kernel event tracing.

Definition in file [\\_\\_ktrace-impl.h](#).

## 15.254 [\\_\\_ktrace-impl.h](#)

```

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 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025 #pragma once
00026
00027 #include <l4/sys/types.h>
00028 #include <l4/sys/kdebug.h>
00029
00030 /*****
00031 *** Implementation
00032 *****/
00033
00034 L4_INLINE l4_tracebuffer_status_t *
00035 fiasco_tbuf_get_status(void)
00036 {
00037 return 0;
00038 /* Not implemented */
00039 }
00040
00041 L4_INLINE l4_addr_t
00042 fiasco_tbuf_get_status_phys(void)
00043 {
00044 return ~0UL;
00045 }
00046
00047 L4_INLINE l4_umword_t
00048 fiasco_tbuf_log(const char *text)
00049 {
00050 enum { TBUF_LOG = 0x201 };
00051 return l4_error(__kdebug_text(TBUF_LOG, text, __builtin_strlen(text)));
00052 }
00053
00054 L4_INLINE l4_umword_t
00055 fiasco_tbuf_log_3val(const char *text, l4_umword_t v1,
00056 l4_umword_t v2,
00057 l4_umword_t v3)
00058 {
00059 enum { TBUF_LOG_3VAL = 0x204 };
00060 return l4_error(__kdebug_3_text(TBUF_LOG_3VAL, text,
00061 __builtin_strlen(text), v1, v2, v3));

```



```

00061 }
00062
00063 L4_INLINE void
00064 fiasco_tbuf_clear(void)
00065 {
00066 enum { TBUF_CLEAR = 0x202 };
00067 __kdebug_op(TBUF_CLEAR);
00068 }
00069
00070 L4_INLINE void
00071 fiasco_tbuf_dump(void)
00072 {
00073 enum { TBUF_DUMP = 0x203 };
00074 __kdebug_op(TBUF_DUMP);
00075 }
00076
00077 L4_INLINE l4_umword_t
00078 fiasco_tbuf_log_binary(const unsigned char *data)
00079 {
00080 enum { TBUF_LOG_BIN = 0x208 };
00081 return l4_error(__kdebug_text(TBUF_LOG_BIN, (const char *)data, 24));
00082 }
00083

```

## 15.255 l4/sys/\_\_typeinfo.h File Reference

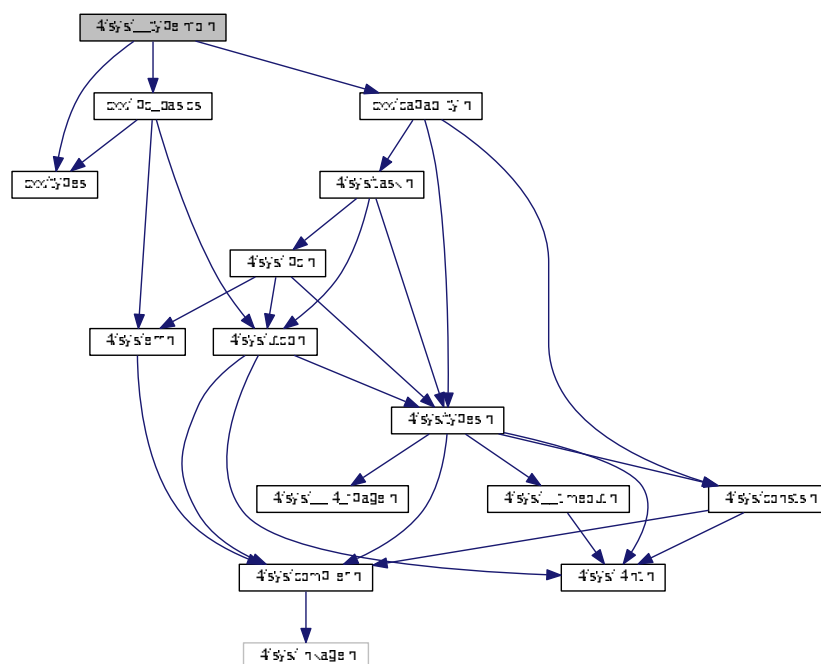
Type information handling.

```

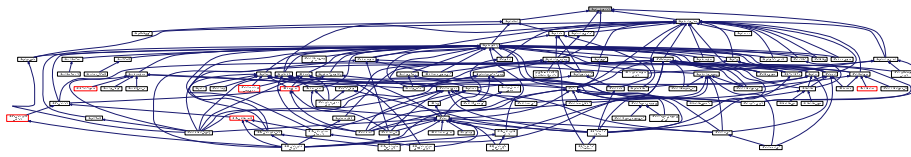
#include "cxx/types"
#include "cxx/ipc_basics"
#include "cxx/capability.h"

```

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.*
- 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\\_demand< T >](#)

*Get the combined server-side resource requirements for all type T...*

- struct [L4::Proto\\_t](#)< P >

*Data type for defining protocol numbers.*

- struct [L4::Kobject\\_x](#)< Derived, ARGS >

*Generic [Kobject](#) inheritance template.*

## Namespaces

- [L4](#)

*[L4](#) low-level kernel interface.*

- [L4::Typeid](#)

*Definition of interface data-type helpers.*

## Typedefs

- typedef int [L4::Opcode](#)

*Data type for RPC opcodes.*

## Enumerations

- enum { [L4::PROTO\\_ANY](#) = 0, [L4::PROTO\\_EMPTY](#) = -19 }

## Functions

- template<typename T >  
Type\_info const \* [L4::kobject\\_typeid](#) ()

*Get the [L4::Type\\_info](#) for the [L4Re](#) interface given in T.*

## 15.255.1 Detailed Description

Type information handling.

Definition in file [\\_\\_typeinfo.h](#).

## 15.256 \_\_typeinfo.h

```

00001
00005 /*
00006 * Copyright (C) 2015 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 * This file is part of TUD:OS and distributed under the terms of the
00014 * GNU General Public License 2.
00015 * Please see the COPYING-GPL-2 file for details.
00016 *
00017 * As a special exception, you may use this file as part of a free software
00018 * library without restriction. Specifically, if other files instantiate
00019 * templates or use macros or inline functions from this file, or you compile
00020 * this file and link it with other files to produce an executable, this
00021 * file does not by itself cause the resulting executable to be covered by
00022 * the GNU General Public License. This exception does not however
00023 * invalidate any other reasons why the executable file might be covered by
00024 * the GNU General Public License.
00025 */
00026 #pragma once
00027 #pragma GCC system_header
00028
00029 #include "cxx/types"
00030 #include "cxx/ipc_basics"
00031 #include "cxx/capability.h"
00032
00033 #if defined(__GXX_RTTI) && !defined(L4_NO_RTTI)
00034 # include <typeinfo>
00035 typedef std::type_info const *L4_std_type_info_ptr;
00036 # define L4_KOBJECT_META_RTTI(type) (&typeid(type))
00037 inline char const *L4_kobject_type_name(L4_std_type_info_ptr n)
00038 { return n ? n->name() : 0; }
00039 #else
00040 typedef void const *L4_std_type_info_ptr;
00041 # define L4_KOBJECT_META_RTTI(type) (0)
00042 inline char const *L4_kobject_type_name(L4_std_type_info_ptr)
00043 { return 0; }
00044 #endif
00045
00046 namespace L4 {
00047 typedef int Opcode;
00048 // internal max helpers
00049 namespace __I {
00050 // internal max of A and B helper
00051 template< unsigned char A, unsigned char B>
00052 struct Max { enum { Res = A > B ? A : B }; };
00053 } // namespace __I
00054
00055 enum
00056 {
00057 PROTO_ANY = 0,
00058 PROTO_EMPTY = -19,
00059 };
00060
00061 namespace Typeid {
00062 using namespace L4::Types;
00063
00064 /*****
00065 *
00066 */
00067 template<long P, typename T>
00068 struct Iface
00069 {
00070 typedef Iface type;
00071 typedef T iface_type;
00072 enum { Proto = P };
00073 };
00074
00075 /*****
00076 *
00077 */
00078 struct Iface_list_end
00079 {
00080 typedef Iface_list_end type;
00081 static bool contains(long) { return false; }
00082 };
00083
00084 template<typename I, typename N = Iface_list_end>
00085 struct Iface_list
00086 {
00087 typedef Iface_list<I, N> type;
00088
00089 typedef typename I::iface_type iface_type;
00090 };
00091 }
00092 }

```

```

00128 typedef N Next;
00129
00130 enum { Proto = I::Proto };
00131
00132 static bool contains(long proto)
00133 { return (proto == Proto) || Next::contains(proto); }
00134 };
00135
00136 // do not insert PROTO_EMPTY interfaces
00137 template<typename I, typename N>
00138 struct Iface_list<Iface<PROTO_EMPTY, I>, N> : N {};
00139
00140 // do not insert 'void' type interfaces
00141 template<long P, typename N>
00142 struct Iface_list<Iface<P, void>, N> : N {};
00143
00144
00145 /*****/
00146 /*
00147 * \internal
00148 * Test if an interface I is in list L
00149 * \tparam I Interface for lookup
00150 * \tparam L Iface_list for search
00151 */
00152 template< typename I, typename L >
00153 struct _In_list;
00154
00155 template< typename I >
00156 struct _In_list<I, Iface_list_end> : False {};
00157
00158 template< typename I, typename N >
00159 struct _In_list<I, Iface_list<I, N> > : True {};
00160
00161 template< typename I, typename I2, typename N >
00162 struct _In_list<I, Iface_list<I2, N> > : _In_list<I, typename N::type> {};
00163
00164 template<typename I, typename L>
00165 struct In_list : _In_list<typename I::type, typename L::type> {};
00166
00167
00168 /*****/
00169 /*
00170 * \internal
00171 * Add Helper: add I to interface list L if ADD is true
00172 * \ingroup l4_cxx_ipc_internal
00173 */
00174 template< bool ADD, typename I, typename L>
00175 struct _Iface_list_add;
00176
00177 template< typename I, typename L>
00178 struct _Iface_list_add<false, I, L> : L {};
00179
00180 template< typename I, typename L>
00181 struct _Iface_list_add<true, I, L> : Iface_list<I, L> {};
00182
00183 /*
00184 * \internal
00185 * Add Helper: add I to interface list L if not already in L.
00186 * \ingroup l4_cxx_ipc_internal
00187 */
00188 template< typename I, typename L >
00189 struct Iface_list_add :
00190 _Iface_list_add<
00191 !In_list<I, typename L::type>::value, I, typename L::type>
00192 {};
00193
00194 /*****/
00195 /*
00196 * \internal
00197 * Helper: checking for a conflict between I2 and I2.
00198 * A conflict means I1 and I2 have the same protocol ID but a different
00199 * iface_type.
00200 */
00201 template< typename I1, typename I2 >
00202 struct __Iface_conflict : Bool<I1::Proto != PROTO_EMPTY && I1::Proto == I2::Proto> {};
00203
00204 template< typename I >
00205 struct __Iface_conflict<I, I> : False {};
00206
00207 /*
00208 * \internal
00209 * Helper: checking for a conflict between I and any interface in LIST.
00210 */
00211 template< typename I, typename LIST >
00212 struct _Iface_conflict;
00213
00214 template< typename I >

```

```

00215 struct _Iface_conflict<I, Iface_list_end> : False {};
00216
00217 template< typename I, typename I2, typename LIST >
00218 struct _Iface_conflict<I, Iface_list<I2, LIST> > :
00219 Bool<__Iface_conflict<I, I2>::value || _Iface_conflict<I, typename LIST::type>::value>
00220 {};
00221
00222 template< typename I, typename LIST >
00223 struct Iface_conflict : _Iface_conflict<typename I::type, typename LIST::type> {};
00224
00225 /*****
00226 */
00227 /*
00228 * \internal
00229 * Helper: merge two interface lists
00230 */
00231 template< typename L1, typename L2 >
00232 struct _Merge_list;
00233
00234 template< typename L >
00235 struct _Merge_list<Iface_list_end, L> : L {};
00236
00237 template< typename I, typename L1, typename L2 >
00238 struct _Merge_list<Iface_list<I, L1>, L2> :
00239 _Merge_list<typename L1::type, typename Iface_list_add<I, L2>::type> {};
00240
00241 template<typename L1, typename L2>
00242 struct Merge_list : _Merge_list<typename L1::type, typename L2::type> {};
00243
00244 /*****
00245 */
00246 /*
00247 * \internal
00248 * check for conflicts among all interfaces in L1 with any interfaces in L2.
00249 */
00250 template< typename L1, typename L2 >
00251 struct _Conflict;
00252
00253 template< typename L >
00254 struct _Conflict<Iface_list_end, L> : False {};
00255
00256 template< typename I, typename L1, typename L2 >
00257 struct _Conflict<Iface_list<I, L1>, L2> :
00258 Bool<Iface_conflict<I, typename L2::type>::value
00259 || _Conflict<typename L1::type, typename L2::type>::value> {};
00260
00261 template< typename L1, typename L2 >
00262 struct Conflict : _Conflict<typename L1::type, typename L2::type> {};
00263
00264 // to be removed -----
00265 // p_dispatch code -- for legacy dispatch -----
00266 /*****
00267 */
00268 /*
00269 * \internal
00270 * helper: Dispatch helper for calling server-side p_dispatch() functions.
00271 */
00272 template<typename LIST>
00273 struct _P_dispatch;
00274
00275 // No matching dispatcher found
00276 template<>
00277 struct _P_dispatch<Iface_list_end>
00278 {
00279 template< typename THIS, typename A1, typename A2 >
00280 static int f(THIS *, long, A1, A2 &)
00281 { return -L4_EBADPROTO; }
00282 };
00283
00284 // call matching p_dispatch() function
00285 template< typename I, typename LIST >
00286 struct _P_dispatch<Iface_list<I, LIST> >
00287 {
00288 // special handling for the meta protocol, to avoid 'using' murx
00289 template< typename THIS, typename A1, typename A2 >
00290 static int _f(THIS self, A1, A2 &a2, True::type)
00291 {
00292 return self->dispatch_meta_request(a2);
00293 }
00294
00295 // normal p_dispatch() dispatching
00296 template< typename THIS, typename A1, typename A2 >
00297 static int _f(THIS self, A1 a1, A2 &a2, False::type)
00298 {
00299 return self->p_dispatch(reinterpret_cast<typename I::iface_type *>(0),
00300 a1, a2);
00301 }
00302
00303 // dispatch function with switch for meta protocol
00304
00305

```

```

00306 template< typename THIS, typename A1, typename A2 >
00307 static int f(THIS *self, long proto, A1 a1, A2 &a2)
00308 {
00309 if (I::Proto == proto)
00310 return _f(self, a1, a2, Bool<I::Proto == (long)L4_PROTO_META>());
00311
00312 return _P_dispatch<typename LIST::type>::f(self, proto, a1, a2);
00313 }
00314 };
00315
00316 template<typename LIST>
00317 struct P_dispatch : _P_dispatch<typename LIST::type> {};
00318 // end: p_dispatch -----
00319 // end: to be removed -----
00320
00321 template<typename RPC> struct Default_op;
00322
00323 namespace Detail {
00324
00325 struct Rpc_end
00326 {
00327 typedef void opcode_type;
00328 typedef Rpc_end rpc;
00329 typedef Rpc_end type;
00330 };
00331
00332 template<typename O1, typename O2, typename RPCS>
00333 struct _Rpc : _Rpc<typename RPCS::next::rpc, O2, typename RPCS::next::type> {};
00334
00335 template<typename O1, typename O2>
00336 struct _Rpc<O1, O2, Rpc_end> {};
00337
00338 template<typename OP, typename RPCS>
00339 struct _Rpc<OP, OP, RPCS> : RPCS
00340 {
00341 typedef _Rpc type;
00342 };
00343
00344 template<typename OP, typename RPCS>
00345 struct Rpc : _Rpc<typename RPCS::rpc, OP, RPCS> {};
00346
00347 template<typename T, unsigned CODE>
00348 struct _Get_opcode
00349 {
00350 template<bool, typename> struct Invalid_opcode {};
00351 template<typename X> struct Invalid_opcode<true, X>;
00352
00353 private:
00354 template<typename U, U> struct _chk;
00355 template<typename U> static long _opc(_chk<int, U::Opcode> *);
00356 template<typename U> static char _opc(...);
00357
00358 template<unsigned SZ, typename U>
00359 struct _Opc { enum { value = CODE }; };
00360
00361 template<typename U>
00362 struct _Opc<sizeof(long), U> { enum { value = U::Opcode }; };
00363
00364 public:
00365 enum { value = _Opc<sizeof(_opc<T>)(0), T::value };
00366 Invalid_opcode<(value < CODE), T> invalid_opcode;
00367 };
00368
00369 template<typename OPCODE, unsigned O, typename ...X>
00370 struct _Rpc : Rpc_end {};
00371
00372 template<typename OPCODE, unsigned O, typename R, typename ...X>
00373 struct _Rpc<OPCODE, O, R, X...>
00374 {
00375 typedef _Rpc type;
00376 typedef OPCODE opcode_type;
00377 typedef R rpc;
00378 typedef typename _Rpc<OPCODE, _Get_opcode<R, O>::value + 1, X...
00379 >::type next;
00380 enum { Opcode = _Get_opcode<R, O>::value };
00381 template<typename Y> struct Rpc : Typeid::Detail::Rpc<Y, _Rpc> {};
00382 };
00383
00384 template<typename OPCODE, unsigned O, typename R>
00385 struct _Rpc<OPCODE, O, Default_op<R> >
00386 {
00387 typedef _Rpc type;
00388 typedef void opcode_type;
00389 typedef R rpc;
00390 typedef Rpc_end next;
00391 enum { Opcode = -99 };
00392 template<typename Y> struct Rpc : Typeid::Detail::Rpc<Y, _Rpc> {};
00393 };

```

```

00410 };
00411
00412 } // namespace Detail
00413
00421 template<typename CLASS>
00422 struct Raw_ipc
00423 {
00424 typedef Raw_ipc type;
00425 typedef Detail::_Rpccs_end next;
00426 typedef void opcode_type;
00427 };
00428
00437 template<typename ...RPCS>
00438 struct Rpccs : Detail::_Rpccs<L4::Opcode, 0, RPCS...> {};
00439
00448 template<typename OPCODE_TYPE>
00449 struct Rpccs_code
00450 {
00454 template<typename ...RPCS>
00455 struct F : Detail::_Rpccs<OPCODE_TYPE, 0, RPCS...> {};
00456 };
00457
00463 template<typename OPERATION>
00464 struct Rpc_nocode : Detail::_Rpccs<void, 0, OPERATION> {};
00465
00474 template<typename ...ARG>
00475 struct Rpccs_sys : Detail::_Rpccs<l4_umword_t, 0, ARG...> {};
00476
00477 template<typename CLASS>
00478 struct Rights
00479 {
00480 unsigned rights;
00481 Rights(unsigned rights) : rights(rights) {}
00482 unsigned operator & (unsigned rhs) const { return rights & rhs; }
00483 };
00484
00485 } // namespace Typeid
00486
00509 struct L4_EXPORT Type_info
00510 {
00516 class L4_EXPORT Demand
00517 {
00518 private:
00520 static unsigned char max(unsigned char a, unsigned char b)
00521 { return a > b ? a : b; }
00522
00523 public:
00524 unsigned char caps;
00525 unsigned char flags;
00526 unsigned char mem;
00527 unsigned char ports;
00528
00536 explicit
00537 Demand(unsigned char caps = 0, unsigned char flags = 0,
00538 unsigned char mem = 0, unsigned char ports = 0)
00539 : caps(caps), flags(flags), mem(mem), ports(ports) {}
00540
00542 bool no_demand() const
00543 { return caps == 0 && mem == 0 && ports == 0 && flags == 0; }
00544
00546 Demand operator | (Demand const &rhs) const
00547 {
00548 return Demand(max(caps, rhs.caps), flags | rhs.flags,
00549 max(mem, rhs.mem), max(ports, rhs.ports));
00550 }
00551 };
00552
00561 template<unsigned char CAPS = 0, unsigned char FLAGS = 0,
00562 unsigned char MEM = 0, unsigned char PORTS = 0>
00563 struct Demand_t : Demand
00564 {
00565 enum
00566 {
00567 Caps = CAPS,
00568 Flags = FLAGS,
00569 Mem = MEM,
00570 Ports = PORTS
00571 };
00572 Demand_t() : Demand(CAPS, FLAGS, MEM, PORTS) {}
00573 };
00574
00582 template<typename D1, typename D2>
00583 struct Demand_union_t : Demand_t<__I::Max<D1::Caps, D2::Caps>::Res,
00584 D1::Flags | D2::Flags,
00585 __I::Max<D1::Mem, D2::Mem>::Res,
00586 __I::Max<D1::Ports, D2::Ports>::Res>
00587 {};

```



```

00588
00589 L4_std_type_info_ptr _type;
00590 Type_info const *const *_bases;
00591 unsigned _num_bases;
00592 long _proto;
00593
00594 L4_std_type_info_ptr type() const { return _type; }
00595 Type_info const *base(unsigned idx) const { return _bases[idx]; }
00596 unsigned num_bases() const { return _num_bases; }
00597 long proto() const { return _proto; }
00598 char const *name() const { return L4_kobject_type_name(type()); }
00599 bool has_proto(long proto) const
00600 {
00601 if (_proto && _proto == proto)
00602 return true;
00603
00604 if (!proto)
00605 return false;
00606
00607 for (unsigned i = 0; i < _num_bases; ++i)
00608 if (base(i)->has_proto(proto))
00609 return true;
00610
00611 return false;
00612 }
00613 };
00614
00620 template<typename T> struct Kobject_typeid
00621 {
00632 typedef typename T::__Kobject_typeid::Demand Demand;
00633 typedef typename T::__Iface::iface_type Iface;
00634 typedef typename T::__Iface_list Iface_list;
00635
00640 static Type_info const *id() { return &T::__Kobject_typeid::_m; }
00641
00649 static Type_info::Demand demand()
00650 { return T::__Kobject_typeid::Demand(); }
00651
00652 // to be removed -----
00653 // p_dispatch -----
00669 template<typename THIS, typename A1, typename A2>
00670 static int proto_dispatch(THIS *self, long proto, A1 a1, A2 &a2)
00671 { return typeid::P_dispatch<typename T::__Iface_list>::f(
00672 self, proto, a1, a2); }
00672 // p_dispatch -----
00673 // end: to be removed -----
00674 };
00675
00684 template<typename T>
00685 inline
00686 Type_info const *kobject_typeid()
00687 { return Kobject_typeid<T>::id(); }
00688
00693 #define L4___GEN_TI(t...) //
00694 Type_info const t::__Kobject_typeid::_m = //
00695 { //
00696 L4_KOBJECT_META_RTTI(Derived), //
00697 &t::__Kobject_typeid::_b[0], //
00698 sizeof(t::__Kobject_typeid::_b) / sizeof(t::__Kobject_typeid::_b[0]), //
00699 PROTO //
00700 } //
00701
00706 #define L4___GEN_TI_MEMBERS(BASE_DEMAND...) //
00707 private: //
00708 template< typename T > friend struct Kobject_typeid; //
00709 protected: //
00710 struct __Kobject_typeid { //
00711 typedef Type_info::Demand_union_t<S_DEMAND, BASE_DEMAND> Demand; //
00712 static Type_info const *const _b[]; //
00713 static Type_info const _m; //
00714 }; //
00715 public: //
00716 static long const Protocol = PROTO; //
00717 typedef L4::Typeid::Rights<Class> Rights; //
00718
00747 template<
00748 typename Derived,
00749 typename Base,
00750 long PROTO = PROTO_ANY,
00751 typename S_DEMAND = Type_info::Demand_t<>
00752 >
00753 class Kobject_t : public Base
00754 {
00755 protected:
00757 typedef Derived Class;
00759 typedef Typeid::Iface<PROTO, Derived> __Iface;
00761 typedef Typeid::Merge_list<

```

```

00762 Typeid::Iface_list<__Iface>, typename Base::__Iface_list
00763 > __Iface_list;
00764
00766 static void __check_protocols__()
00767 {
00768 typedef Typeid::Iface_conflict<__Iface, typename Base::__Iface_list> Base_conflict;
00769 static_assert(!Base_conflict::value, "ambiguous protocol ID: protocol also used by Base");
00770 }
00771
00773 L4::Cap<Class> c() const { return L4::Cap<Class>(this->cap()); }
00774
00775 // Generate the remaining type information
00776 L4__GEN_TI_MEMBERS(typename Base::__Kobject_typeid::Demand)
00777 };
00778
00779
00780 template< typename Derived, typename Base, long PROTO, typename S_DEMAND>
00781 Type_info const *const
00782 Kobject_t<Derived, Base, PROTO, S_DEMAND>::
00783 __Kobject_typeid::b[] = { &Base::__Kobject_typeid::m };
00784
00789 template< typename Derived, typename Base, long PROTO, typename S_DEMAND>
00790 L4__GEN_TI(Kobject_t<Derived, Base, PROTO, S_DEMAND>);
00791
00792
00822 template<
00823 typename Derived,
00824 typename Base1,
00825 typename Base2,
00826 long PROTO = PROTO_ANY,
00827 typename S_DEMAND = Type_info::Demand_t<>
00828 >
00829 class Kobject_2t : public Base1, public Base2
00830 {
00831 protected:
00833 typedef Derived Class;
00835 typedef Typeid::Iface<PROTO, Derived> __Iface;
00837 typedef Typeid::Merge_list<
00838 Typeid::Iface_list<__Iface>,
00839 Typeid::Merge_list<
00840 typename Base1::__Iface_list,
00841 typename Base2::__Iface_list
00842 >
00843 > __Iface_list;
00844
00846 static void __check_protocols__()
00847 {
00848 typedef typename Base1::__Iface_list Base1_proto_list;
00849 typedef typename Base2::__Iface_list Base2_proto_list;
00850
00851 typedef Typeid::Iface_conflict<__Iface, Base1_proto_list> Base1_conflict;
00852 typedef Typeid::Iface_conflict<__Iface, Base2_proto_list> Base2_conflict;
00853 static_assert(!Base1_conflict::value, "ambiguous protocol ID, also in Base1");
00854 static_assert(!Base2_conflict::value, "ambiguous protocol ID, also in Base2");
00855
00856 typedef Typeid::Conflict<Base1_proto_list, Base2_proto_list> Bases_conflict;
00857 static_assert(!Bases_conflict::value, "ambiguous protocol IDs in base classes");
00858 }
00859
00860 // disambiguate cap()
00861 l4_cap_idx_t cap() const throw()
00862 { return Base1::cap(); }
00863
00865 L4::Cap<Class> c() const { return L4::Cap<Class>(this->cap()); }
00866
00867 L4__GEN_TI_MEMBERS(Type_info::Demand_union_t<
00868 typename Base1::__Kobject_typeid::Demand,
00869 typename Base2::__Kobject_typeid::Demand>
00870)
00871
00872 public:
00873 // Provide non-ambiguous conversion to Kobject
00874 operator Kobject const & () const
00875 { return *static_cast<Base1 const *>(this); }
00876
00877 // Provide non-ambiguous access of dec_refcnt()
00878 l4_msgtag_t dec_refcnt(l4_mword_t diff, l4_utcb_t *utcb =
00879 l4_utcb())
00880 { return Base1::dec_refcnt(diff, utcb); }
00881 };
00882
00883 template< typename Derived, typename Base1, typename Base2,
00884 long PROTO, typename S_DEMAND >
00885 Type_info const *const
00886 Kobject_2t<Derived, Base1, Base2, PROTO, S_DEMAND>::__Kobject_typeid::b
00887 [] =

```

```

00887 {
00888 &Base1::__Kobject_typeid::_m,
00889 &Base2::__Kobject_typeid::_m
00890 };
00891
00896 template< typename Derived, typename Base1, typename Base2,
00897 long PROTO, typename S_DEMAND >
00898 L4_____GEN_TI(Kobject_2t<Derived, Base1, Base2, PROTO, S_DEMAND>
00899);
00900
00901
00921 template<
00922 typename Derived,
00923 typename Base1,
00924 typename Base2,
00925 typename Base3,
00926 long PROTO = PROTO_ANY,
00927 typename S_DEMAND = Type_info::Demand_t<>
00928 >
00929 struct Kobject_3t : Base1, Base2, Base3
00930 {
00931 protected:
00932 typedef Derived Class;
00933 typedef Typeid::Iface<PROTO, Derived> __Iface;
00934 typedef Typeid::Merge_list<
00935 Typeid::Iface_list<__Iface>,
00936 Typeid::Merge_list<
00937 typename Base1::__Iface_list,
00938 Typeid::Merge_list<
00939 typename Base2::__Iface_list,
00940 typename Base3::__Iface_list
00941 >
00942 >
00943 >
00944 >
00945 >
00946 > __Iface_list;
00947
00949 static void __check_protocols__()
00950 {
00951 typedef typename Base1::__Iface_list Base1_proto_list;
00952 typedef typename Base2::__Iface_list Base2_proto_list;
00953 typedef typename Base3::__Iface_list Base3_proto_list;
00954
00955 typedef Typeid::Iface_conflict<__Iface, Base1_proto_list> Base1_conflict;
00956 typedef Typeid::Iface_conflict<__Iface, Base2_proto_list> Base2_conflict;
00957 typedef Typeid::Iface_conflict<__Iface, Base3_proto_list> Base3_conflict;
00958
00959 static_assert(!Base1_conflict::value, "ambiguous protocol ID, also in Base1");
00960 static_assert(!Base2_conflict::value, "ambiguous protocol ID, also in Base2");
00961 static_assert(!Base3_conflict::value, "ambiguous protocol ID, also in Base3");
00962
00963 typedef Typeid::Conflict<Base1_proto_list, Base2_proto_list> Conflict_bases12;
00964 typedef Typeid::Conflict<Base1_proto_list, Base3_proto_list> Conflict_bases13;
00965 typedef Typeid::Conflict<Base2_proto_list, Base3_proto_list> Conflict_bases23;
00966
00967 static_assert(!Conflict_bases12::value, "ambiguous protocol IDs in base classes: Base1 and Base2");
00968 static_assert(!Conflict_bases13::value, "ambiguous protocol IDs in base classes: Base1 and Base3");
00969 static_assert(!Conflict_bases23::value, "ambiguous protocol IDs in base classes: Base2 and Base3");
00970 }
00971
00972 // disambiguate cap()
00973 l4_cap_idx_t cap() const throw()
00974 { return Base1::cap(); }
00975
00977 L4::Cap<Class> c() const { return L4::Cap<Class>(this->cap()); }
00978
00979 L4_____GEN_TI_MEMBERS(Type_info::Demand_union_t<
Type_info::Demand_union_t<
00980 typename Base1::__Kobject_typeid::Demand,
00981 typename Base2::__Kobject_typeid::Demand>,
00982 typename Base3::__Kobject_typeid::Demand>
00983)
00984
00985 public:
00986 // Provide non-ambiguous conversion to Kobject
00987 operator Kobject const & () const
00988 { return *static_cast<Base1 const *>(this); }
00989
00990 // Provide non-ambiguous access of dec_refcnt()
00991 l4_msgtag_t dec_refcnt(l4_mword_t diff, l4_utcb_t *utcb =
l4_utcb())
00992 { return Base1::dec_refcnt(diff, utcb); }
00993 };
00994
00995
00996 template< typename Derived, typename Base1, typename Base2, typename Base3,
00997 long PROTO, typename S_DEMAND >
00998 Type_info const *const

```

```

00999 Kobject_3t<Derived, Base1, Base2, Base3, PROTO, S_DEMAND>::__Kobject_typeid::_b
 [] =
01000 {
01001 &Base1::__Kobject_typeid::_m,
01002 &Base2::__Kobject_typeid::_m,
01003 &Base3::__Kobject_typeid::_m
01004 };
01005
01010 template< typename Derived, typename Base1, typename Base2, typename Base3,
01011 long PROTO, typename S_DEMAND >
01012 L4_____GEN_TI(Kobject_3t<Derived, Base1, Base2, Base3, PROTO, S_DEMAND>
01013);
01014 }
01015
01016 #if __cplusplus >= 201103L
01017 namespace L4 {
01018
01026 template< typename ...T >
01027 struct Kobject_demand;
01028
01029 template<>
01030 struct Kobject_demand<> : Type_info::Demand_t<> {};
01031
01032 template<typename T>
01033 struct Kobject_demand<T> : Kobject_typeid<T>::Demand {};
01034
01035 template<typename T1, typename ...T2>
01036 struct Kobject_demand<T1, T2...> :
01037 Type_info::Demand_union_t<typename Kobject_typeid<T1>::Demand,
01038 Kobject_demand<T2...> >
01039 {};
01040
01041 namespace Typeid_xx {
01042
01043 template<typename ...LISTS>
01044 struct Merge_list;
01045
01046 template<typename L>
01047 struct Merge_list<L> : L {};
01048
01049 template<typename L1, typename L2>
01050 struct Merge_list<L1, L2> : Typeid::Merge_list<L1, L2> {};
01051
01052 template<typename L1, typename L2, typename ...LISTS>
01053 struct Merge_list<L1, L2, LISTS...> :
01054 Merge_list<typename Typeid::Merge_list<L1, L2>::type, LISTS...> {};
01055
01056 template< typename I, typename ...LIST >
01057 struct Iface_conflict;
01058
01059 template< typename I >
01060 struct Iface_conflict<I> : Typeid::False {};
01061
01062 template< typename I, typename L, typename ...LIST >
01063 struct Iface_conflict<I, L, LIST...> :
01064 Typeid::Bool<Typeid::Iface_conflict<typename I::type, typename L::type>::value
01065 || Iface_conflict<I, LIST...>::value>
01066 {};
01067
01068 template< typename ...LIST >
01069 struct Conflict;
01070
01071 template< typename L >
01072 struct Conflict<L> : Typeid::False {};
01073
01074 template< typename L1, typename L2, typename ...LIST >
01075 struct Conflict<L1, L2, LIST...> :
01076 Typeid::Bool<Typeid::Conflict<typename L1::type, typename L2::type>::value
01077 || Conflict<L1, LIST...>::value
01078 || Conflict<L2, LIST...>::value>
01079 {};
01080
01081 template< typename T >
01082 struct Is_demand
01083 {
01084 static long test(Type_info::Demand const *);
01085 static char test(...);
01086 enum { value = sizeof(test((T*)0)) == sizeof(long) };
01087 };
01088
01089 template< typename T, typename ... >
01090 struct First : T { typedef T type; };
01091 } // Typeid
01092
01098 template< typename Derived, long PROTO, typename S_DEMAND, typename ...BASES>

```

```

01099 struct __Kobject_base : BASES...
01100 {
01101 protected:
01102 typedef Derived Class;
01103 typedef Typeid::Iface<PROTO, Derived> __Iface;
01104 typedef Typeid_xx::Merge_list<
01105 Typeid::Iface_list<__Iface>,
01106 typename BASES::__Iface_list...
01107 > __Iface_list;
01108
01109 static void __check_protocols__()
01110 {
01111 typedef Typeid_xx::Iface_conflict<__Iface, typename BASES::__Iface_list...> Conflict;
01112 static_assert(!Conflict::value, "ambiguous protocol ID, protocol also used in base class");
01113
01114 typedef Typeid_xx::Conflict<typename BASES::__Iface_list...> Base_conflict;
01115 static_assert(!Base_conflict::value, "ambiguous protocol IDs in base classes");
01116 }
01117
01118 // disambiguate cap()
01119 l4_cap_idx_t cap() const throw()
01120 { return Typeid_xx::First<BASES...>::type::cap(); }
01121
01122 L4::Cap<Class> c() const { return L4::Cap<Class>(this->cap()); }
01123
01124 L4_____GEN_TI_MEMBERS(Kobject_demand<BASES...>)
01125
01126 private:
01127 // This function returns the first base class (used below)
01128 template<typename B1, typename ...> struct Basel { typedef B1 type; };
01129
01130 public:
01131 // Provide non-ambiguous conversion to Kobject
01132 operator Kobject const & () const
01133 { return *static_cast<typename Basel<BASES...>::type const *>(this); }
01134
01135 // Provide non-ambiguous access of dec_refcnt()
01136 l4_msgtag_t dec_refcnt(l4_mword_t diff, l4_utcb_t *utcb =
01137 l4_utcb())
01137 { return Basel<BASES...>::type::dec_refcnt(diff, utcb); }
01138 };
01139
01140 template< typename Derived, long PROTO, typename S_DEMAND, typename ...BASES>
01141 Type_info const *const
01142 __Kobject_base<Derived, PROTO, S_DEMAND, BASES...>::__Kobject_typeid::_b[] =
01143 {
01144 (&BASES::__Kobject_typeid::_m)...
01145 };
01146
01147 template< typename Derived, long PROTO, typename S_DEMAND, typename ...BASES>
01148 L4_____GEN_TI(__Kobject_base<Derived, PROTO, S_DEMAND, BASES...>);
01149
01150
01151 // Test if there is a Demand argument to Kobject_x
01152 template< typename Derived, long PROTO, bool HAS_DEMAND, typename DEMAND, typename ...ARGS >
01153 struct __Kobject_x_proto;
01154
01155 // YES: pass it to __Kobject_base
01156 template< typename Derived, long PROTO, typename DEMAND, typename ...BASES>
01157 struct __Kobject_x_proto<Derived, PROTO, true, DEMAND, BASES...> :
01158 __Kobject_base<Derived, PROTO, DEMAND, BASES...> {};
01159
01160 // NO: pass it empty Type_info::Demand_t
01161 template< typename Derived, long PROTO, typename B1, typename ...BASES>
01162 struct __Kobject_x_proto<Derived, PROTO, false, B1, BASES...> :
01163 __Kobject_base<Derived, PROTO, Type_info::Demand_t<>, B1, BASES...> {};
01164
01172 template< long P = PROTO_EMPTY >
01173 struct Proto_t {};
01174
01188 template< typename Derived, typename ...ARGS >
01189 struct Kobject_x;
01190
01191 template< typename Derived, typename A, typename ...ARGS >
01192 struct Kobject_x<Derived, A, ARGS...> :
01193 __Kobject_x_proto<Derived, PROTO_ANY, Typeid_xx::Is_demand<A>::value, A, ARGS...>
01194 {};
01195
01196 template< typename Derived, long PROTO, typename A, typename ...ARGS >
01197 struct Kobject_x<Derived, Proto_t<PROTO>, A, ARGS...> :
01198 __Kobject_x_proto<Derived, PROTO, Typeid_xx::Is_demand<A>::value, A, ARGS...>
01199 {};
01200
01201 }
01202 #endif
01203
01204 #undef L4_____GEN_TI

```

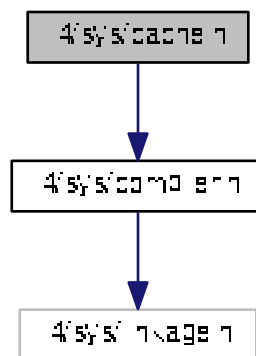
```
01205 #undef L4___GEN_TI_MEMBERS
01206
```

## 15.257 l4/sys/cache.h File Reference

Cache-consistency 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.*
- int [l4\\_cache\\_flush\\_data](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Cache flush a range.*
- int [l4\\_cache\\_inv\\_data](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Cache invalidate a range.*
- int [l4\\_cache\\_coherent](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Make memory coherent between I-cache and D-cache.*
- int [l4\\_cache\\_dma\\_coherent](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Make memory coherent for use with external memory.*
- int [l4\\_cache\\_dma\\_coherent\\_full](#) (void) [L4\\_NOTHROW](#)  
*Make memory coherent for use with external memory.*

### 15.257.1 Detailed Description

Cache-consistency functions.

Date

2007-11

Author

Adam Lackorzynski [adam@os.inf.tu-dresden.de](mailto:adam@os.inf.tu-dresden.de)

Definition in file [cache.h](#).

## 15.258 cache.h

```

00001
00011 /*
00012 * (c) 2007-2009 Author(s)
00013 * economic rights: Technische Universität Dresden (Germany)
00014 *
00015 * This file is part of TUD:OS and distributed under the terms of the
00016 * GNU General Public License 2.
00017 * Please see the COPYING-GPL-2 file for details.
00018 *
00019 * As a special exception, you may use this file as part of a free software
00020 * library without restriction. Specifically, if other files instantiate
00021 * templates or use macros or inline functions from this file, or you compile
00022 * this file and link it with other files to produce an executable, this
00023 * file does not by itself cause the resulting executable to be covered by
00024 * the GNU General Public License. This exception does not however
00025 * invalidate any other reasons why the executable file might be covered by
00026 * the GNU General Public License.
00027 */
00028
00029 #ifndef __L4SYS__INCLUDE__CACHE_H__
00030 #define __L4SYS__INCLUDE__CACHE_H__
00031
00032 #include <l4/sys/compiler.h>
00033
00042 EXTERN_C_BEGIN
00043
00055 L4_INLINE int
00056 l4_cache_clean_data(unsigned long start,
00057 unsigned long end) L4_NOTHROW;
00058
00070 L4_INLINE int
00071 l4_cache_flush_data(unsigned long start,
00072 unsigned long end) L4_NOTHROW;
00073
00085 L4_INLINE int
00086 l4_cache_inv_data(unsigned long start,
00087 unsigned long end) L4_NOTHROW;
00088
00100 L4_INLINE int
00101 l4_cache_coherent(unsigned long start,
00102 unsigned long end) L4_NOTHROW;
00103
00115 L4_INLINE int
00116 l4_cache_dma_coherent(unsigned long start,
00117 unsigned long end) L4_NOTHROW;
00118
00123 L4_INLINE int
00124 l4_cache_dma_coherent_full(void) L4_NOTHROW;
00125
00126 EXTERN_C_END
00127
00128 #endif /* ! __L4SYS__INCLUDE__CACHE_H__ */

```

## 15.259 arm/l4/sys/cache.h File Reference

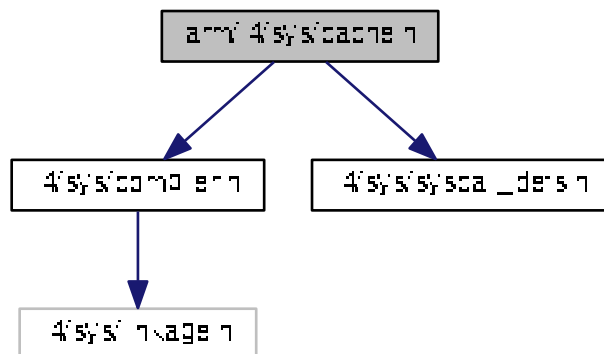
Cache functions.

```

#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.*
- `int l4_cache_flush_data` (unsigned long start, unsigned long end) `L4_NOTHROW`  
*Cache flush a range.*
- `int l4_cache_inv_data` (unsigned long start, unsigned long end) `L4_NOTHROW`  
*Cache invalidate a range.*
- `int l4_cache_coherent` (unsigned long start, unsigned long end) `L4_NOTHROW`  
*Make memory coherent between I-cache and D-cache.*
- `int l4_cache_dma_coherent` (unsigned long start, unsigned long end) `L4_NOTHROW`  
*Make memory coherent for use with external memory.*
- `int l4_cache_dma_coherent_full` (void) `L4_NOTHROW`  
*Make memory coherent for use with external memory.*

### 15.259.1 Detailed Description

Cache functions.

Date

2007-11

Author

Adam Lackorzynski [adam@os.inf.tu-dresden.de](mailto:adam@os.inf.tu-dresden.de)

Definition in file [cache.h](#).



## 15.260 cache.h

```

00001
00009 /*
00010 * (c) 2007-2009 Author(s)
00011 * economic rights: Technische Universität Dresden (Germany)
00012 *
00013 * This file is part of TUD:OS and distributed under the terms of the
00014 * GNU General Public License 2.
00015 * Please see the COPYING-GPL-2 file for details.
00016 *
00017 * As a special exception, you may use this file as part of a free software
00018 * library without restriction. Specifically, if other files instantiate
00019 * templates or use macros or inline functions from this file, or you compile
00020 * this file and link it with other files to produce an executable, this
00021 * file does not by itself cause the resulting executable to be covered by
00022 * the GNU General Public License. This exception does not however
00023 * invalidate any other reasons why the executable file might be covered by
00024 * the GNU General Public License.
00025 */
00026 #ifndef __L4SYS__INCLUDE__ARCH_ARM__CACHE_H__
00027 #define __L4SYS__INCLUDE__ARCH_ARM__CACHE_H__
00028
00029 #include <l4/sys/compiler.h>
00030 #include <l4/sys/syscall_defs.h>
00031
00032 #include_next <l4/sys/cache.h>
00033
00037 L4_INLINE void
00038 l4_cache_op_arm_call(unsigned long op,
00039 unsigned long start,
00040 unsigned long end);
00041
00042 L4_INLINE void
00043 l4_cache_op_arm_call(unsigned long op,
00044 unsigned long start,
00045 unsigned long end)
00046 {
00047 register unsigned long _op __asm__ ("r0") = op;
00048 register unsigned long _start __asm__ ("r1") = start;
00049 register unsigned long _end __asm__ ("r2") = end;
00050
00051 __asm__ __volatile__
00052 ("@ l4_cache_op_arm_call(start) \n\t"
00053 "mov lr, pc \n\t"
00054 "mov pc, %[sc] \n\t"
00055 "@ l4_cache_op_arm_call(end) \n\t"
00056 :
00057 "=r" (_op),
00058 "=r" (_start),
00059 "=r" (_end)
00060 :
00061 [sc] "i" (L4_SYSCALL_MEM_OP),
00062 "0" (_op),
00063 "1" (_start),
00064 "2" (_end)
00065 :
00066 "cc", "memory", "lr"
00067);
00068 }
00069
00070 enum L4_mem_cache_ops
00071 {
00072 L4_MEM_CACHE_OP_CLEAN_DATA = 0,
00073 L4_MEM_CACHE_OP_FLUSH_DATA = 1,
00074 L4_MEM_CACHE_OP_INV_DATA = 2,
00075 L4_MEM_CACHE_OP_COHERENT = 3,
00076 L4_MEM_CACHE_OP_DMA_COHERENT = 4,
00077 L4_MEM_CACHE_OP_DMA_COHERENT_FULL = 5,
00078 L4_MEM_CACHE_OP_L2_CLEAN = 6,
00079 L4_MEM_CACHE_OP_L2_FLUSH = 7,
00080 L4_MEM_CACHE_OP_L2_INV = 8,
00081 };
00082
00083 L4_INLINE int
00084 l4_cache_clean_data(unsigned long start,
00085 unsigned long end) L4_NOTHROW
00086 {
00087 l4_cache_op_arm_call(L4_MEM_CACHE_OP_CLEAN_DATA, start, end);
00088 return 0;
00089 }
00090
00091 L4_INLINE int
00092 l4_cache_flush_data(unsigned long start,
00093 unsigned long end) L4_NOTHROW
00094 {

```

```

00095 l4_cache_op_arm_call(L4_MEM_CACHE_OP_FLUSH_DATA, start, end);
00096 return 0;
00097 }
00098
00099 L4_INLINE int
00100 l4_cache_inv_data(unsigned long start,
00101 unsigned long end) L4_NOTHROW
00102 {
00103 l4_cache_op_arm_call(L4_MEM_CACHE_OP_INV_DATA, start, end);
00104 return 0;
00105 }
00106
00107 L4_INLINE int
00108 l4_cache_coherent(unsigned long start,
00109 unsigned long end) L4_NOTHROW
00110 {
00111 l4_cache_op_arm_call(L4_MEM_CACHE_OP_COHERENT, start, end);
00112 return 0;
00113 }
00114
00115 L4_INLINE int
00116 l4_cache_dma_coherent(unsigned long start,
00117 unsigned long end) L4_NOTHROW
00118 {
00119 l4_cache_op_arm_call(L4_MEM_CACHE_OP_DMA_COHERENT, start, end);
00120 return 0;
00121 }
00122
00123 L4_INLINE int
00124 l4_cache_dma_coherent_full(void) L4_NOTHROW
00125 {
00126 l4_cache_op_arm_call(L4_MEM_CACHE_OP_DMA_COHERENT_FULL, 0, 0);
00127 return 0;
00128 }
00129
00130 L4_INLINE int
00131 l4_cache_l2_clean(unsigned long start,
00132 unsigned long end) L4_NOTHROW
00133 {
00134 l4_cache_op_arm_call(L4_MEM_CACHE_OP_L2_CLEAN, start, end);
00135 return 0;
00136 }
00137
00138 L4_INLINE int
00139 l4_cache_l2_flush(unsigned long start,
00140 unsigned long end) L4_NOTHROW
00141 {
00142 l4_cache_op_arm_call(L4_MEM_CACHE_OP_L2_FLUSH, start, end);
00143 return 0;
00144 }
00145
00146 L4_INLINE int
00147 l4_cache_l2_inv(unsigned long start,
00148 unsigned long end) L4_NOTHROW
00149 {
00150 l4_cache_op_arm_call(L4_MEM_CACHE_OP_L2_INV, start, end);
00151 return 0;
00152 }
00153
00154 #endif /* ! __L4SYS__INCLUDE__ARCH_ARM__CACHE_H__ */

```

## 15.261 amd64/l4/sys/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.*
- int [l4\\_cache\\_flush\\_data](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Cache flush a range.*
- int [l4\\_cache\\_inv\\_data](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Cache invalidate a range.*

- int [l4\\_cache\\_coherent](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Make memory coherent between I-cache and D-cache.*
- int [l4\\_cache\\_dma\\_coherent](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Make memory coherent for use with external memory.*
- int [l4\\_cache\\_dma\\_coherent\\_full](#) (void) [L4\\_NOTHROW](#)  
*Make memory coherent for use with external memory.*

### 15.261.1 Detailed Description

Cache functions.

Definition in file [cache.h](#).

## 15.262 cache.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007 * economic rights: Technische Universität Dresden (Germany)
00008 *
00009 * This file is part of TUD:OS and distributed under the terms of the
00010 * GNU General Public License 2.
00011 * Please see the COPYING-GPL-2 file for details.
00012 *
00013 * As a special exception, you may use this file as part of a free software
00014 * library without restriction. Specifically, if other files instantiate
00015 * templates or use macros or inline functions from this file, or you compile
00016 * this file and link it with other files to produce an executable, this
00017 * file does not by itself cause the resulting executable to be covered by
00018 * the GNU General Public License. This exception does not however
00019 * invalidate any other reasons why the executable file might be covered by
00020 * the GNU General Public License.
00021 */
00022 #ifndef __L4SYS__INCLUDE__ARCH_AMD64__CACHE_H__
00023 #define __L4SYS__INCLUDE__ARCH_AMD64__CACHE_H__
00024
00025 #include_next <l4/sys/cache.h>
00026
00027 L4_INLINE int
00028 l4_cache_clean_data(unsigned long start,
00029 unsigned long end) L4_NOTHROW
00030 {
00031 (void)start; (void)end;
00032 return 0;
00033 }
00034
00035 L4_INLINE int
00036 l4_cache_flush_data(unsigned long start,
00037 unsigned long end) L4_NOTHROW
00038 {
00039 (void)start; (void)end;
00040 return 0;
00041 }
00042
00043 L4_INLINE int
00044 l4_cache_inv_data(unsigned long start,
00045 unsigned long end) L4_NOTHROW
00046 {
00047 (void)start; (void)end;
00048 return 0;
00049 }
00050
00051 L4_INLINE int
00052 l4_cache_coherent(unsigned long start,
00053 unsigned long end) L4_NOTHROW
00054 {
00055 (void)start; (void)end;
00056 return 0;
00057 }
00058
00059 L4_INLINE int
00060 l4_cache_dma_coherent(unsigned long start,
```

```

00061 unsigned long end) L4_NOTHROW
00062 {
00063 (void)start; (void)end;
00064 return 0;
00065 }
00066
00067 L4_INLINE int
00068 l4_cache_dma_coherent_full(void) L4_NOTHROW
00069 {
00070 return 0;
00071 }
00072
00073 #endif /* ! __L4SYS__INCLUDE__ARCH_AMD64__CACHE_H__ */

```

## 15.263 x86/I4/sys/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.*
- int [l4\\_cache\\_flush\\_data](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Cache flush a range.*
- int [l4\\_cache\\_inv\\_data](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Cache invalidate a range.*
- int [l4\\_cache\\_coherent](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Make memory coherent between I-cache and D-cache.*
- int [l4\\_cache\\_dma\\_coherent](#) (unsigned long start, unsigned long end) [L4\\_NOTHROW](#)  
*Make memory coherent for use with external memory.*
- int [l4\\_cache\\_dma\\_coherent\\_full](#) (void) [L4\\_NOTHROW](#)  
*Make memory coherent for use with external memory.*

### 15.263.1 Detailed Description

Cache functions.

Definition in file [cache.h](#).

## 15.264 cache.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007 * economic rights: Technische Universität Dresden (Germany)
00008 *
00009 * This file is part of TUD:OS and distributed under the terms of the
00010 * GNU General Public License 2.
00011 * Please see the COPYING-GPL-2 file for details.
00012 *
00013 * As a special exception, you may use this file as part of a free software
00014 * library without restriction. Specifically, if other files instantiate
00015 * templates or use macros or inline functions from this file, or you compile
00016 * this file and link it with other files to produce an executable, this
00017 * file does not by itself cause the resulting executable to be covered by
00018 * the GNU General Public License. This exception does not however
00019 * invalidate any other reasons why the executable file might be covered by
00020 * the GNU General Public License.

```

```

00021 */
00022 #ifndef __L4SYS__INCLUDE__ARCH_X86__CACHE_H__
00023 #define __L4SYS__INCLUDE__ARCH_X86__CACHE_H__
00024
00025 #include_next <l4/sys/cache.h>
00026
00027 L4_INLINE int
00028 l4_cache_clean_data(unsigned long start,
00029 unsigned long end) L4_NOTHROW
00030 {
00031 (void)start; (void)end;
00032 return 0;
00033 }
00034
00035 L4_INLINE int
00036 l4_cache_flush_data(unsigned long start,
00037 unsigned long end) L4_NOTHROW
00038 {
00039 (void)start; (void)end;
00040 return 0;
00041 }
00042
00043 L4_INLINE int
00044 l4_cache_inv_data(unsigned long start,
00045 unsigned long end) L4_NOTHROW
00046 {
00047 (void)start; (void)end;
00048 return 0;
00049 }
00050
00051 L4_INLINE int
00052 l4_cache_coherent(unsigned long start,
00053 unsigned long end) L4_NOTHROW
00054 {
00055 (void)start; (void)end;
00056 return 0;
00057 }
00058
00059 L4_INLINE int
00060 l4_cache_dma_coherent(unsigned long start,
00061 unsigned long end) L4_NOTHROW
00062 {
00063 (void)start; (void)end;
00064 return 0;
00065 }
00066
00067 L4_INLINE int
00068 l4_cache_dma_coherent_full(void) L4_NOTHROW
00069 {
00070 return 0;
00071 }
00072
00073 #endif /* ! __L4SYS__INCLUDE__ARCH_X86__CACHE_H__ */

```

## 15.265 l4/sys/capability File Reference

[L4::Cap](#) related definitions.

```

#include <l4/sys/consts.h>
#include <l4/sys/types.h>
#include <l4/sys/kobject>
#include <l4/sys/task.h>
#include <l4/sys/meta>

```

The figure displays a large, intricate network graph. The nodes are organized into several distinct clusters or communities. A central cluster contains a high density of nodes and edges, suggesting a highly interactive core. Other clusters are more sparsely connected and often serve as peripheries to the central group. Nodes are represented by small circles, and their labels consist of text identifying each entity. Edges are thin lines connecting the nodes, representing the relationships between them. Some nodes are highlighted with a red border, while others have a blue border. The overall structure is complex and non-linear, typical of social or organizational networks.

- L4

*L4 low-level kernel interface.*

- `#define L4_DISABLE_COPY(_class)`  
*Disable copy of a class.*

- `template<typename T, typename F>`  
`Cap<T> L4::cap_dynamic_cast (Cap< F > const &c) throw ()`  
*dynamic\_cast for capabilities.*

## 15.265.1 Detailed Description

[L4::Cap](#) related definitions.

### Author

Alexander Warg [alexander.warg@os.inf.tu-dresden.de](mailto:alexander.warg@os.inf.tu-dresden.de)

Definition in file [capability](#).

## 15.265.2 Macro Definition Documentation

### 15.265.2.1 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 61 of file [capability](#).

## 15.266 capability

```
00001 // vim:set ft=cpp:
00009 /*
00010 * (c) 2008-2009,2015 Author(s)
00011 * economic rights: Technische Universität Dresden (Germany)
00012 *
```

```

00013 * This file is part of TUD:OS and distributed under the terms of the
00014 * GNU General Public License 2.
00015 * Please see the COPYING-GPL-2 file for details.
00016 *
00017 * As a special exception, you may use this file as part of a free software
00018 * library without restriction. Specifically, if other files instantiate
00019 * templates or use macros or inline functions from this file, or you compile
00020 * this file and link it with other files to produce an executable, this
00021 * file does not by itself cause the resulting executable to be covered by
00022 * the GNU General Public License. This exception does not however
00023 * invalidate any other reasons why the executable file might be covered by
00024 * the GNU General Public License.
00025 */
00026 #pragma once
00027
00028 #include <l4/sys/consts.h>
00029 #include <l4/sys/types.h>
00030 #include <l4/sys/kobject>
00031 #include <l4/sys/task.h>
00032
00033 namespace L4
00034 {
00035
00036 /* Forward declarations for our kernel object classes. */
00037 class Task;
00038 class Thread;
00039 class Factory;
00040 class Irq;
00041 class Log;
00042 class Vm;
00043 class Kobject;
00044
00060 #if __cplusplus >= 201103L
00061 # define L4_DISABLE_COPY(_class) \
00062 public: \
00063 _class(_class const &) = delete; \
00064 _class operator = (_class const &) = delete; \
00065 private:
00066 #else
00067 # define L4_DISABLE_COPY(_class) \
00068 private: \
00069 _class(_class const &); \
00070 _class operator = (_class const &);
00071 #endif
00072
00073
00074 #define L4_KOBJECT_DISABLE_COPY(_class) \
00075 protected: \
00076 _class(); \
00077 L4_DISABLE_COPY(_class)
00078
00079
00080 #define L4_KOBJECT(_class) L4_KOBJECT_DISABLE_COPY(_class)
00081
00082 inline l4_msgtag_t
00083 Cap_base::validate(Cap<Task> task, l4_utcb_t *u) const throw()
00084 { return l4_task_cap_valid_u(task.cap(), _c, u); }
00085
00086 inline l4_msgtag_t
00087 Cap_base::validate(l4_utcb_t *u) const throw()
00088 { return l4_task_cap_valid_u(L4_BASE_TASK_CAP, _c, u); }
00089
00090 }; // namespace L4
00091
00092 #include <l4/sys/meta>
00093
00094 namespace L4 {
00095
00117 template< typename T, typename F >
00118 inline
00119 Cap<T> cap_dynamic_cast(Cap<F> const &c) throw()
00120 {
00121 if (!c.is_valid())
00122 return Cap<T>::Invalid;
00123
00124 Cap<Meta> mc = cap_reinterpret_cast<Meta>(c);
00125 Type_info const *m = kobject_typeid<T>();
00126 if (m->proto() && l4_error(mc->supports(m->proto())) > 0)
00127 return Cap<T>(c.cap());
00128
00129 // FIXME: use generic checker
00130 #if 0
00131 if (l4_error(mc->supports(T::kobject_proto())) > 0)
00132 return Cap<T>(c.cap());
00133 #endif
00134
00135 return Cap<T>::Invalid;

```

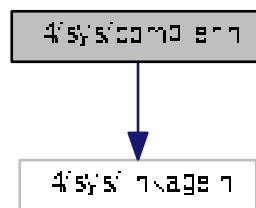


```
00136 }
00137
00138 }
```

## 15.267 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_ALWAYS_INLINE`  
*L4 Inline function attribute.*
- `#define L4_NOTHROW`  
*Mark a function declaration and definition as never throwing an exception.*
- `#define EXTERN_C_BEGIN`  
*Start section with C types and functions.*
- `#define EXTERN_C_END`  
*End section with C types and functions.*
- `#define EXTERN_C`  
*Mark C types and functions.*
- `#define __END_DECLS`  
*End section with C types and functions.*
- `#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_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

- `void l4_barrier (void)`  
*Memory barrier.*
- `void l4_mb (void)`  
*Memory barrier.*
- `void l4_wmb (void)`  
*Write memory barrier.*

## 15.267.1 Detailed Description

[L4](#) compiler related defines.

Definition in file [compiler.h](#).

## 15.268 compiler.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 * Jork Löser <jork@os.inf.tu-dresden.de>,
00007 * Ronald Aigner <ra3@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 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
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 */
00025 #ifndef __L4_COMPILER_H__
00026 #define __L4_COMPILER_H__
00027
00028 #if !defined(__ASSEMBLY__) && !defined(__ASSEMBLER__)
00029

```

```

00040
00045 #ifndef L4_INLINE
00046 #ifndef __cplusplus
00047 # ifdef __OPTIMIZE__
00048 # define L4_INLINE_STATIC static __inline__
00049 # define L4_INLINE_EXTERN extern __inline__
00050 # ifdef __GNUC_STDC_INLINE__
00051 # define L4_INLINE L4_INLINE_STATIC
00052 # else
00053 # define L4_INLINE L4_INLINE_EXTERN
00054 # endif
00055 # else /* ! __OPTIMIZE__ */
00056 # define L4_INLINE static
00057 # endif /* ! __OPTIMIZE__ */
00058 #else /* __cplusplus */
00059 # define L4_INLINE inline
00060 #endif /* __cplusplus */
00061 #endif /* L4_INLINE */
00062
00067 #define L4_ALWAYS_INLINE L4_INLINE __attribute__((__always_inline__))
00068
00069
00070 #define L4_DECLARE_CONSTRUCTOR(func, prio) \
00071 static inline __attribute__((constructor(prio))) void func ## _ctor_func(void) { func(); }
00072
00073
00171 #ifndef __cplusplus
00172 # define L4_NOTHROW_A __attribute__((nothrow))
00173 # define L4_NOTHROW
00174 # define EXTERN_C_BEGIN
00175 # define EXTERN_C_END
00176 # define EXTERN_C
00177 # ifndef __BEGIN_DECLS
00178 # define __BEGIN_DECLS
00179 # endif
00180 # ifndef __END_DECLS
00181 # define __END_DECLS
00182 # endif
00183 # define L4_DEFAULT_PARAM(x)
00184 #else /* __cplusplus */
00185 # define L4_NOTHROW throw()
00186 # define EXTERN_C_BEGIN extern "C" {
00187 # define EXTERN_C_END }
00188 # define EXTERN_C extern "C"
00189 # ifndef __BEGIN_DECLS
00190 # define __BEGIN_DECLS extern "C" {
00191 # endif
00192 # ifndef __END_DECLS
00193 # define __END_DECLS }
00194 # endif
00195 # define L4_DEFAULT_PARAM(x) = x
00196 #endif /* __cplusplus */
00197
00202 #define L4_NORETURN __attribute__((noreturn))
00203
00204 #define L4_PURE __attribute__((pure))
00205
00210 #define L4_NOINSTRUMENT __attribute__((no_instrument_function))
00211 #ifndef L4_HIDDEN
00212 # define L4_HIDDEN __attribute__((visibility("hidden")))
00213 #endif
00214 #ifndef L4_EXPORT
00215 # define L4_EXPORT __attribute__((visibility("default")))
00216 #endif
00217 #ifndef L4_EXPORT_TYPE
00218 # ifdef __cplusplus
00219 # define L4_EXPORT_TYPE __attribute__((visibility("default")))
00220 # else
00221 # define L4_EXPORT_TYPE
00222 # endif
00223 #endif
00224 #define L4_STRONG_ALIAS(name, aliasname) L4__STRONG_ALIAS(name, aliasname)
00225 #define L4__STRONG_ALIAS(name, aliasname) \
00226 extern __typeof (name) aliasname __attribute__((alias (#name)));
00227
00228
00229 #endif /* !__ASSEMBLY__ */
00230
00231 #include <l4/sys/linkage.h>
00232
00233 #define L4_LIKELY(x) __builtin_expect((x),1)
00234 #define L4_UNLIKELY(x) __builtin_expect((x),0)
00235
00236 /* Make sure that the function is not removed by optimization. Without the
00237 * "used" attribute, unreferenced static functions are removed. */
00238 #define L4_STICKY(x) __attribute__((used)) x
00239 #define L4_DEPRECATED(s) __attribute__((deprecated(s)))

```

```

00240
00241 #ifndef __GXX_EXPERIMENTAL_CXX0X__
00242 #ifndef static_assert
00243 #define static_assert(x, y) \
00244 do { (void)sizeof(char[-!(x)]); } while (0)
00245 #endif
00246 #endif
00247
00248 #define L4_stringify_helper(x) #x
00249 #define L4_stringify(x) L4_stringify_helper(x)
00250
00251 #ifndef __ASSEMBLER__
00252
00255 L4_INLINE void l4_barrier(void);
00256
00260 L4_INLINE void l4_mb(void);
00261
00265 L4_INLINE void l4_wmb(void);
00266
00267
00268 /* Implementations */
00269 L4_INLINE void l4_barrier(void)
00270 {
00271 __asm__ __volatile__ (" : : : \"memory\"");
00272 }
00273
00274 L4_INLINE void l4_mb(void)
00275 {
00276 __asm__ __volatile__ (" : : : \"memory\"");
00277 }
00278
00279 L4_INLINE void l4_wmb(void)
00280 {
00281 __asm__ __volatile__ (" : : : \"memory\"");
00282 }
00283 #endif
00284
00287 #endif /* !__L4_COMPILER_H__ */

```

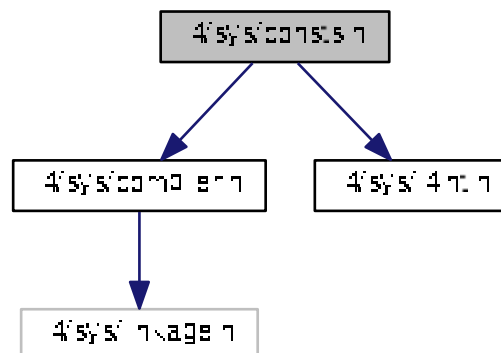
## 15.269 l4/sys/consts.h File Reference

Common constants.

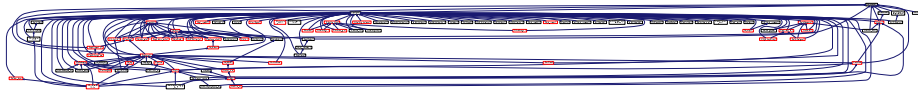
```
#include <l4/sys/compiler.h>
```

```
#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_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_syscall_flags_t {`  
`L4_SYSF_NONE, L4_SYSF_SEND, L4_SYSF_RECV, L4_SYSF_OPEN_WAIT,`  
`L4_SYSF_REPLY, L4_SYSF_CALL, L4_SYSF_WAIT, L4_SYSF_SEND_AND_WAIT,`  
`L4_SYSF_REPLY_AND_WAIT }`  
*Capability selector flags.*
- `enum l4_cap_consts_t { L4_CAP_SHIFT, L4_CAP_SIZE, L4_CAP_MASK, L4_INVALID_CAP }`  
*Constants related to capability selectors.*
- `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_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 }`  
*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_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 flex page 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 \(l4\\_umword\\_t value, unsigned char bits\)](#) [L4\\_NOTHROW](#)  
*Round value up to the next alignment with bits size.*

### 15.269.1 Detailed Description

Common constants.

Definition in file [consts.h](#).

### 15.270 consts.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 * This file is part of TUD:OS and distributed under the terms of the
00014 * GNU General Public License 2.
00015 * Please see the COPYING-GPL-2 file for details.
00016 *
00017 * As a special exception, you may use this file as part of a free software
00018 * library without restriction. Specifically, if other files instantiate
00019 * templates or use macros or inline functions from this file, or you compile
00020 * this file and link it with other files to produce an executable, this
00021 * file does not by itself cause the resulting executable to be covered by
00022 * the GNU General Public License. This exception does not however
00023 * invalidate any other reasons why the executable file might be covered by
00024 * the GNU General Public License.
00025 */
00026 #ifndef __L4_SYS__INCLUDE__CONSTS_H__
00027 #define __L4_SYS__INCLUDE__CONSTS_H__
00028
00029 #include <l4/sys/compiler.h>
00030 #include <l4/sys/l4int.h>
00031
00039 enum l4_syscall_flags_t
00040 {
00048 L4_SYSF_NONE = 0x00,
00049
00058 L4_SYSF_SEND = 0x01,
00059
00069 L4_SYSF_RECV = 0x02,
00070
00080 L4_SYSF_OPEN_WAIT = 0x04,
00081
00089 L4_SYSF_REPLY = 0x08,
00090
00097 L4_SYSF_CALL = L4_SYSF_SEND | L4_SYSF_RECV,
00098
00105 L4_SYSF_WAIT = L4_SYSF_OPEN_WAIT |
00106 L4_SYSF_RECV,
00113 L4_SYSF_SEND_AND_WAIT = L4_SYSF_OPEN_WAIT |
00114 L4_SYSF_CALL,
00121 L4_SYSF_REPLY_AND_WAIT = L4_SYSF_WAIT |
00122 L4_SYSF_SEND | L4_SYSF_REPLY
00123 };

```

```

00128 enum l4_cap_consts_t
00129 {
00131 L4_CAP_SHIFT = 12UL,
00133 L4_CAP_SIZE = 1UL << L4_CAP_SHIFT,
00134 L4_CAP_OFFSET = 1UL << L4_CAP_SHIFT,
00139 L4_CAP_MASK = ~0UL << (L4_CAP_SHIFT - 1),
00141 L4_INVALID_CAP = ~0UL << (L4_CAP_SHIFT - 1),
00142
00143 L4_INVALID_CAP_BIT = 1UL << (L4_CAP_SHIFT - 1),
00144 };
00145
00146 enum l4_sched_consts_t
00147 {
00148 L4_SCHED_MIN_PRIO = 0,
00149 L4_SCHED_MAX_PRIO = 255,
00150 };
00151
00157 enum l4_unmap_flags_t
00158 {
00165 L4_FP_ALL_SPACES = 0x80000000UL,
00166
00173 L4_FP_DELETE_OBJ = 0xc0000000UL,
00174
00180 L4_FP_OTHER_SPACES = 0x0UL
00181 };
00182
00187 enum l4_msg_item_consts_t
00188 {
00189 L4_ITEM_MAP = 8,
00190
00195 L4_ITEM_CONT = 1,
00196
00197 // send
00198 L4_MAP_ITEM_GRANT = 2,
00199 L4_MAP_ITEM_MAP = 0,
00200
00201 // receive
00206 L4_RCV_ITEM_SINGLE_CAP = L4_ITEM_MAP | 2,
00207
00212 L4_RCV_ITEM_LOCAL_ID = 4,
00213 };
00214
00219 enum l4_buffer_desc_consts_t
00220 {
00221 L4_BDR_MEM_SHIFT = 0,
00222 L4_BDR_IO_SHIFT = 5,
00223 L4_BDR_OBJ_SHIFT = 10,
00224 L4_BDR_OFFSET_MASK = (1UL << 20) - 1,
00225 };
00226
00240 enum l4_default_caps_t
00241 {
00243 L4_BASE_TASK_CAP = 1UL << L4_CAP_SHIFT,
00245 L4_BASE_FACTORY_CAP = 2UL << L4_CAP_SHIFT,
00247 L4_BASE_THREAD_CAP = 3UL << L4_CAP_SHIFT,
00249 L4_BASE_PAGER_CAP = 4UL << L4_CAP_SHIFT,
00251 L4_BASE_LOG_CAP = 5UL << L4_CAP_SHIFT,
00253 L4_BASE_ICU_CAP = 6UL << L4_CAP_SHIFT,
00255 L4_BASE_SCHEDULER_CAP = 7UL << L4_CAP_SHIFT,
00257 L4_BASE_IOMMU_CAP = 8UL << L4_CAP_SHIFT,
00259 L4_BASE_DEBUGGER_CAP = 10UL << L4_CAP_SHIFT,
00260
00262 L4_BASE_CAPS_LAST_P1,
00264 L4_BASE_CAPS_LAST = L4_BASE_CAPS_LAST_P1 - 1
00265 };
00266
00277 #define L4_PAGE_SIZE (1UL << L4_PAGE_SHIFT)
00278
00286 #define L4_PAGE_MASK (~ (L4_PAGE_SIZE - 1))
00287
00295 #define L4_LOG2_PAGE_SIZE L4_PAGE_SHIFT
00296
00304 #define L4_SUPERPAGE_SIZE (1UL << L4_SUPERPAGE_SHIFT)
00305
00313 #define L4_SUPERPAGE_MASK (~ (L4_SUPERPAGE_SIZE - 1))
00314
00321 #define L4_LOG2_SUPERPAGE_SIZE L4_SUPERPAGE_SHIFT
00322
00333 L4_INLINE l4_addr_t l4_trunc_page(l4_addr_t address)
00334 L4_NOTHROW;
00334 L4_INLINE l4_addr_t l4_trunc_page(l4_addr_t address) L4_NOTHROW
00335 { return address & L4_PAGE_MASK; }
00336
00344 L4_INLINE l4_addr_t l4_trunc_size(l4_addr_t address, unsigned char bits)
00345 L4_NOTHROW;
00345 L4_INLINE l4_addr_t l4_trunc_size(l4_addr_t address, unsigned char bits)
00346 L4_NOTHROW

```

```

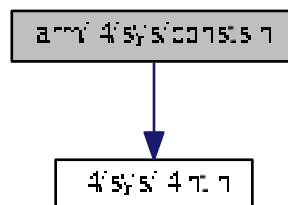
00346 { return address & (~0UL << bits); }
00347
00358 L4_INLINE l4_addr_t l4_round_page(l4_addr_t address)
00359 L4_NOTHROW;
00359 L4_INLINE l4_addr_t l4_round_page(l4_addr_t address) L4_NOTHROW
00360 { return (address + L4_PAGESIZE - 1) & L4_PAGEMASK; }
00361
00369 L4_INLINE l4_addr_t l4_round_size(l4_umword_t value, unsigned char bits)
00370 L4_NOTHROW;
00370 L4_INLINE l4_addr_t l4_round_size(l4_umword_t value, unsigned char bits)
00371 L4_NOTHROW
00371 { return (value + (1UL << bits) - 1) & (~0UL << bits); }
00372
00377 enum l4_addr_consts_t {
00379 L4_INVALID_ADDR = ~0UL
00380 };
00381
00386 #define L4_INVALID_PTR ((void *)L4_INVALID_ADDR)
00387
00388 #ifndef NULL
00389 #ifndef __cplusplus
00390 # define NULL ((void *)0)
00394 #else
00395 # define NULL 0
00396 #endif
00397 #endif
00398
00399 #endif /* ! __L4_SYS__INCLUDE__CONSTS_H__ */

```

## 15.271 arm/l4/sys/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.*

### 15.271.1 Detailed Description

Common [L4](#) constants, arm version.

Definition in file [consts.h](#).



## 15.272 consts.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 * economic rights: Technische Universität Dresden (Germany)
00011 *
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025 #ifndef __L4_SYS_CONSTS_H
00026 #define __L4_SYS_CONSTS_H
00027
00028 /* L4 includes */
00029 #include <l4/sys/l4int.h>
00037 #define L4_PAGESHIFT 12
00038
00042 #define L4_SUPERPAGESHIFT 21
00043
00046 #include_next <l4/sys/consts.h>
00047
00048 #endif /* !__L4_SYS_CONSTS_H */

```

## 15.273 amd64/l4/sys/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.*

### 15.273.1 Detailed Description

Common [L4](#) constants, amd64 version.

Definition in file [consts.h](#).

## 15.274 consts.h

```

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 * This file is part of TUD:OS and distributed under the terms of the
00015 * GNU General Public License 2.
00016 * Please see the COPYING-GPL-2 file for details.
00017 *
00018 * As a special exception, you may use this file as part of a free software
00019 * library without restriction. Specifically, if other files instantiate
00020 * templates or use macros or inline functions from this file, or you compile
00021 * this file and link it with other files to produce an executable, this
00022 * file does not by itself cause the resulting executable to be covered by
00023 * the GNU General Public License. This exception does not however
00024 * invalidate any other reasons why the executable file might be covered by
00025 * the GNU General Public License.
00026 */
00027 /*****
00028 #ifndef __L4SYS__INCLUDE__ARCH_AMD64__CONSTS_H__
00029 #define __L4SYS__INCLUDE__ARCH_AMD64__CONSTS_H__
00030
00035 #define L4_PAGESHIFT 12
00036
00041 #define L4_SUPERPAGESHIFT 21
00042
00043 #include_next <l4/sys/consts.h>
00044
00045 #endif /* ! __L4SYS__INCLUDE__ARCH_AMD64__CONSTS_H__ */

```

## 15.275 x86/l4/sys/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.*

### 15.275.1 Detailed Description

Common [L4](#) constants, x86 version.

Definition in file [consts.h](#).

## 15.276 consts.h

```

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 * This file is part of TUD:OS and distributed under the terms of the
00015 * GNU General Public License 2.
00016 * Please see the COPYING-GPL-2 file for details.
00017 *
00018 * As a special exception, you may use this file as part of a free software
00019 * library without restriction. Specifically, if other files instantiate
00020 * templates or use macros or inline functions from this file, or you compile
00021 * this file and link it with other files to produce an executable, this
00022 * file does not by itself cause the resulting executable to be covered by
00023 * the GNU General Public License. This exception does not however
00024 * invalidate any other reasons why the executable file might be covered by
00025 * the GNU General Public License.
00026 */
00027 /*****
00028 #ifndef __L4SYS__INCLUDE__ARCH_X86__CONSTS_H__
00029 #define __L4SYS__INCLUDE__ARCH_X86__CONSTS_H__
00030
00035 #define L4_PAGESHIFT 12
00036
00041 #define L4_SUPERPAGESHIFT 22
00042
00043 #include_next <l4/sys/consts.h>
00044
00045 #endif /* ! __L4SYS__INCLUDE__ARCH_X86__CONSTS_H__ */

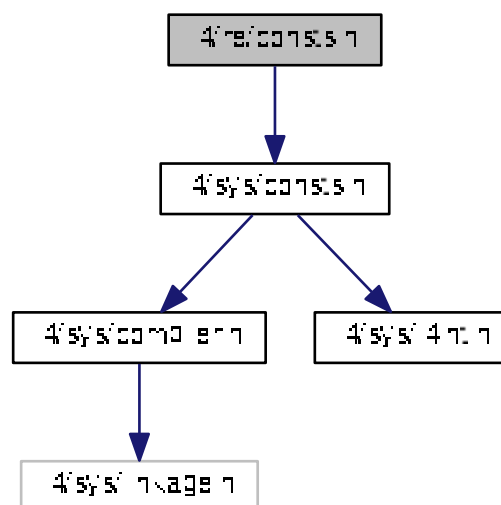
```

## 15.277 l4/re/consts.h File Reference

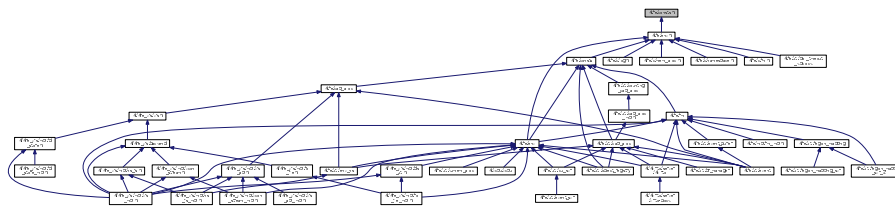
Constants.

```
#include <l4/sys/consts.h>
```

Include dependency graph for consts.h:



This graph shows which files directly or indirectly include this file:



### 15.277.1 Detailed Description

Constants.

Definition in file [consts.h](#).

## 15.278 consts.h

```

00001
00005 /*
00006 * (c) 2008-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00007 * economic rights: Technische Universität Dresden (Germany)
00008 *
00009 * This file is part of TUD:OS and distributed under the terms of the
00010 * GNU General Public License 2.
00011 * Please see the COPYING-GPL-2 file for details.
00012 *
00013 * As a special exception, you may use this file as part of a free software
00014 * library without restriction. Specifically, if other files instantiate
00015 * templates or use macros or inline functions from this file, or you compile
00016 * this file and link it with other files to produce an executable, this
00017 * file does not by itself cause the resulting executable to be covered by
00018 * the GNU General Public License. This exception does not however
00019 * invalidate any other reasons why the executable file might be covered by
00020 * the GNU General Public License.
00021 */
00022 #pragma once
00023
00024 #include <l4/sys/consts.h>
00025
00026 enum
00027 {
00028 L4RE_THIS_TASK_CAP = 1UL << L4_CAP_SHIFT,
00029 };
00030

```

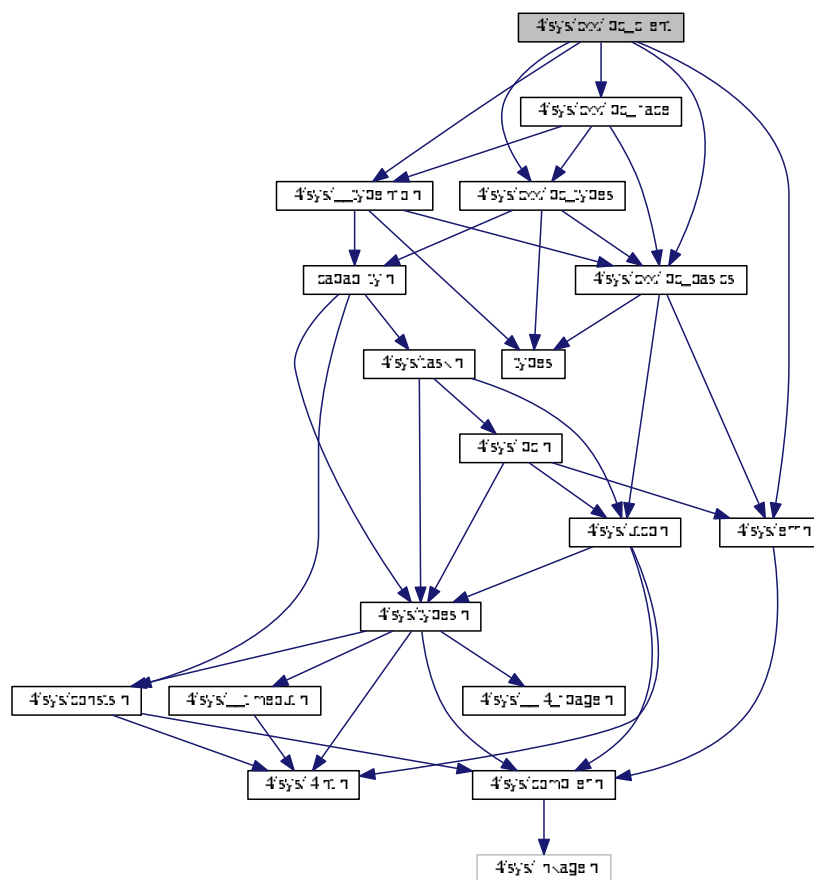
## 15.279 l4/sys/cxx/ipc\_client File Reference

```

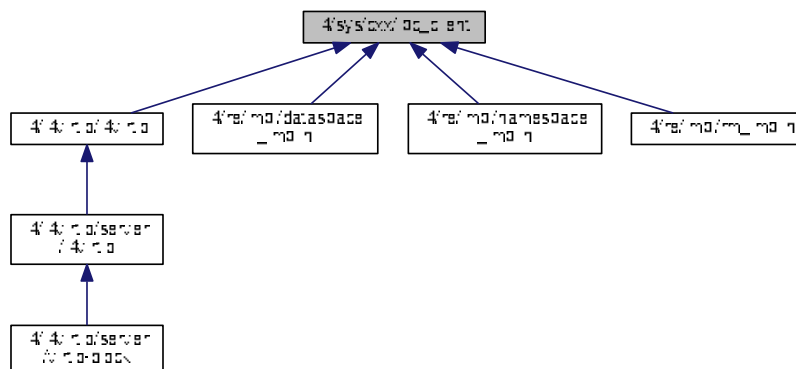
#include <l4/sys/cxx/ipc_basics>
#include <l4/sys/cxx/ipc_types>
#include <l4/sys/cxx/ipc_iface>
#include <l4/sys/__typeinfo.h>

```

```
#include <l4/sys/err.h>
Include dependency graph for ipc_client:
```



This graph shows which files directly or indirectly include this file:



## Namespaces

- [L4](#)  
*L4 low-level kernel interface.*
- [L4::ipc](#)  
*IPC related functionality.*
- [L4::ipc::Msg](#)  
*IPC Message related functionality.*

## Macros

- `#define L4_RPC_DEF(name)`  
*Generate the definition of an RPC stub.*

### 15.279.1 Macro Definition Documentation

#### 15.279.1.1 L4\_RPC\_DEF

```
#define L4_RPC_DEF(
 name)
```

#### Value:

```
template struct L4::Ipcc::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 [43](#) of file [ipc\\_client](#).

### 15.280 ipc\_client

```
00001 // vi:set ft=c++: -- Mode: C++ --
00002 /*
00003 * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004 *
00005 * This file is part of TUD:OS and distributed under the terms of the
00006 * GNU General Public License 2.
00007 * Please see the COPYING-GPL-2 file for details.
00008 *
00009 * As a special exception, you may use this file as part of a free software
00010 * library without restriction. Specifically, if other files instantiate
```

```

00011 * templates or use macros or inline functions from this file, or you compile
00012 * this file and link it with other files to produce an executable, this
00013 * file does not by itself cause the resulting executable to be covered by
00014 * the GNU General Public License. This exception does not however
00015 * invalidate any other reasons why the executable file might be covered by
00016 * the GNU General Public License.
00017 */
00018 #pragma once
00019 #pragma GCC system_header
00020
00021 #include <l4/sys/cxx/ipc_basics>
00022 #include <l4/sys/cxx/ipc_types>
00023 #include <l4/sys/cxx/ipc_iface>
00024 #include <l4/sys/__typeinfo.h>
00025 #include <l4/sys/err.h>
00026
00031 namespace L4 { namespace Ipc { namespace Msg {
00032 //-----
00033
00043 #define L4_RPC_DEF(name) \
00044 template struct L4::Ipc::Msg::Rpc_call \
00045 <name##_t, name##_t::class_type, name##_t::ipc_type, name##_t::flags_type>
00046
00047
00049 //-----
00050 //Implementation of the RPC call
00051 template<typename OP, typename C, typename FLAGS, typename R, typename ...ARGS>
00052 R L4_EXPORT
00053 Rpc_call<OP, C, R (ARGS...), FLAGS>::
00054 call(L4::Cap<C> cap, typename _Elem<ARGS>::arg_type ...a, l4_utcb_t *utcb) throw()
00055 {
00056 return Rpc_inline_call<OP, C, R (ARGS...), FLAGS>::call(cap, a..., utcb);
00057 }
00059
00060 } // namespace Msg
00061 } // namespace Ipc
00062 } // namespace L4
00063

```

## 15.281 l4/sys/cxx/ipc\_iface File Reference

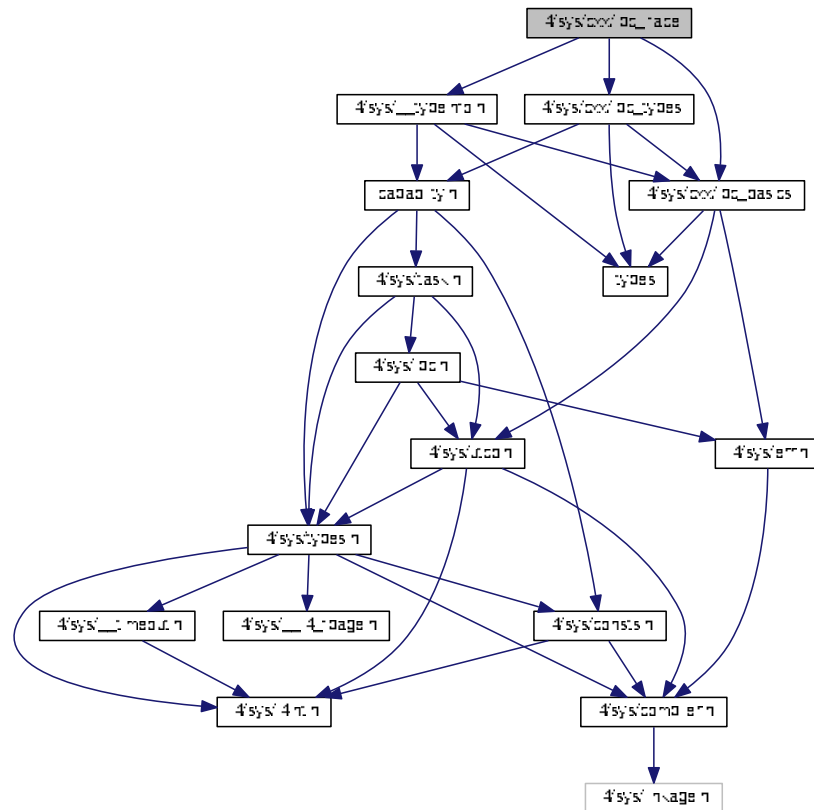
Interface Definition Language.

```

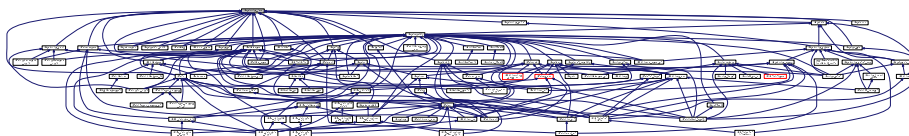
#include <l4/sys/cxx/ipc_basics>
#include <l4/sys/cxx/ipc_types>
#include <l4/sys/__typeinfo.h>

```

Include dependency graph for ipc\_iface:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct `L4::ipc::Call`  
*RPC attribute for a standard RPC call.*
- struct `L4::ipc::Call_zero_send_timeout`  
*RPC attribute for an RPC call, with zero send timeout.*
- struct `L4::ipc::Call_t< RIGHTS >`  
*RPC attribute for an RPC call with required rights.*
- struct `L4::ipc::Send_only`  
*RPC attribute for a send-only RPC.*



## Namespaces

- [L4](#)  
*L4 low-level kernel interface.*
- [L4::ipc](#)  
*IPC related functionality.*
- [L4::ipc::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...) res name args`  
*Define an inline RPC call (type and callable).*
- `#define L4_INLINE_RPC_OP(op, res, name, args, attr...) res name args`  
*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...) res name args`  
*Define an RPC call (type and callable).*
- `#define L4_RPC_OP(op, res, name, args, attr...) res name args`  
*Define an RPC call with specific opcode (type and callable).*

### 15.281.1 Detailed Description

Interface Definition Language.

See also

[L4\\_RPC](#), [L4\\_INLINE\\_RPC](#), [L4::ipc::Call](#) [L4::ipc::Send\\_only](#), [L4::ipc::Msg::Rpc\\_call](#), [L4::ipc::Msg::Rpc\\_↔  
inline\\_call](#)

Definition in file [ipc\\_iface](#).

### 15.281.2 Macro Definition Documentation

#### 15.281.2.1 L4\_INLINE\_RPC

```
#define L4_INLINE_RPC(
 res,
 name,
 args,
 attr...) 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 ( <a href="#">L4::lpc::Call</a> , <a href="#">L4::lpc::Call_t</a> etc.). |

Definition at line 457 of file [ipc\\_iface](#).

Referenced by [L4Re::Video::Goos::delete\\_view\(\)](#).

## 15.281.2.2 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> \
{
 typedef Rpc_inline_call type;
 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 ( <a href="#">L4::lpc::Call</a> , <a href="#">L4::lpc::Call_t</a> 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 429 of file [ipc\\_iface](#).

Referenced by [L4::Factory::create\(\)](#), [L4Re::Video::Goos::create\\_view\(\)](#), and [L4Re::Inhibitor::next\\_lock\\_info\(\)](#).

## 15.281.2.3 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> \
{
 typedef Rpc_inline_call type; 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 ( <a href="#">L4::Ipc::Call</a> , <a href="#">L4::Ipc::Call_t</a> 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 442 of file [ipc\\_iface](#).

**15.281.2.4 L4\_INLINE\_RPC\_OP**

```

#define L4_INLINE_RPC_OP(
 op,
 res,
 name,
 args,
 attr...) 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 ( <a href="#">L4::Ipc::Call</a> , <a href="#">L4::Ipc::Call_t</a> etc.). |

Definition at line 472 of file [ipc\\_iface](#).

Referenced by [L4::Scheduler::info\(\)](#), and [L4::Icu::info\(\)](#).

## 15.281.2.5 L4\_RPC

```
#define L4_RPC(
 res,
 name,
 args,
 attr...) 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 ( <code>name_t</code> is used for the type.)                            |
| <i>args</i> | The argument list of the RPC function.                                                           |
| <i>attr</i> | Optional RPC attributes ( <a href="#">L4::ipc::Call</a> , <a href="#">L4::ipc::Call_t</a> etc.). |

Definition at line 515 of file [ipc\\_iface](#).

Referenced by [L4Re::Rm::find\(\)](#), and [L4Re::Rm::reserve\\_area\(\)](#).

## 15.281.2.6 L4\_RPC\_NF

```
#define L4_RPC_NF(
 res,
 name,
 args...)
```

## Value:

```
struct name##_t : L4::Ipc::Msg::Rpc_call<name##_t, Class, res args>
{
 typedef Rpc_call type;
 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 ( <code>name_t</code> is used for the type.)                                                              |
| <i>args</i> | The argument list of the RPC function, and RPC attributes ( <a href="#">L4::ipc::Call</a> , <a href="#">L4::ipc::Call_t</a> etc.). |

Definition at line 485 of file [ipc\\_iface](#).

Referenced by [L4Re::Rm::find\(\)](#), and [L4Re::Rm::reserve\\_area\(\)](#).

## 15.281.2.7 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>
{
 typedef Rpc_call type; 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 ( <a href="#">L4::Ipc::Call</a> , <a href="#">L4::Ipc::Call_t</a> etc.). |

Definition at line 500 of file [ipc\\_iface](#).

Referenced by [L4::Icu::bind\(\)](#), [L4::Icu::info\(\)](#), [L4::Icu::mask\(\)](#), [L4Re::Namespace::register\\_obj\(\)](#), [L4::Icu::set\\_mode\(\)](#), and [L4::Icu::unbind\(\)](#).

## 15.281.2.8 L4\_RPC\_OP

```
#define L4_RPC_OP (
 op,
 res,
 name,
 args,
 attr...) 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 ( <a href="#">L4::Ipc::Call</a> , <a href="#">L4::Ipc::Call_t</a> etc.). |

Definition at line 530 of file [ipc\\_iface](#).

## 15.282 ipc\_iface

```

00001 // vi:set ft=c++: -- Mode: C++ --
00002 /*
00003 * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004 *
00005 * This file is part of TUD:OS and distributed under the terms of the
00006 * GNU General Public License 2.
00007 * Please see the COPYING-GPL-2 file for details.
00008 *
00009 * As a special exception, you may use this file as part of a free software
00010 * library without restriction. Specifically, if other files instantiate
00011 * templates or use macros or inline functions from this file, or you compile
00012 * this file and link it with other files to produce an executable, this
00013 * file does not by itself cause the resulting executable to be covered by
00014 * the GNU General Public License. This exception does not however
00015 * invalidate any other reasons why the executable file might be covered by
00016 * the GNU General Public License.
00017 */
00018 #pragma once
00019 #pragma GCC system_header
00020
00021 #include <l4/sys/cxx/ipc_basics>
00022 #include <l4/sys/cxx/ipc_types>
00023 #include <l4/sys/__typeinfo.h>
00024
00196 // TODO: add some more documentation
00197 namespace L4 { namespace Ipc {
00198
00215 struct L4_EXPORT Call
00216 {
00217 enum { Is_call = true };
00218 enum { Rights = 0 };
00219 static l4_timeout_t timeout() { return L4_IPC_NEVER; }
00220 };
00221
00225 struct L4_EXPORT Call_zero_send_timeout : Call
00226 {
00227 static l4_timeout_t timeout() { return L4_IPC_SEND_TIMEOUT_0; }
00228 };
00229
00245 template<unsigned RIGHTS>
00246 struct L4_EXPORT Call_t : Call
00247 {
00248 enum { Rights = RIGHTS };
00249 };
00250
00263 struct L4_EXPORT Send_only
00264 {
00265 enum { Is_call = false };
00266 enum { Rights = 0 };
00267 static l4_timeout_t timeout() { return L4_IPC_NEVER; }
00268 };
00269
00270 namespace Msg {
00271
00282 template<typename OP, typename CLASS, typename SIG, typename FLAGS = Call>
00283 struct L4_EXPORT Rpc_inline_call;
00284
00289 template<typename OP, typename CLASS, typename FLAGS, typename R,
00290 typename ...ARGS>
00291 struct L4_EXPORT Rpc_inline_call<OP, CLASS, R (ARGS...), FLAGS>
00292 {
00293 template<typename T> struct Result { typedef T result_type; };
00294 enum
00295 {
00296 Return_tag = L4::Types::Same<R, l4_msgtag_t>::value
00297 };
00298
00300 typedef Rpc_inline_call type;
00301 typedef OP op_type;
00302 typedef CLASS class_type;
00303 typedef typename Result<R>::result_type result_type;
00304 typedef R ipc_type (ARGS...);
00305 typedef result_type func_type (typename _Elem<ARGS>::arg_type...);
00306
00313 typedef FLAGS flags_type;
00314
00315 template<typename RES>
00316 static typename L4::Types::Enable_if< Return_tag, RES >::type
00317 return_err(long err) { return l4_msgtag(err, 0, 0, 0); }
00318
00319 template<typename RES>
00320 static typename L4::Types::Enable_if< Return_tag, RES >::type
00321 return_ipc_err(l4_msgtag_t tag, l4_utcb_t const *) { return tag; }
00322

```

```

00323 template<typename RES>
00324 static typename L4::Types::Enable_if< Return_tag, RES >::type
00325 return_code(l4_msgtag_t tag) { return tag; }
00326
00327 template<typename RES>
00328 static typename L4::Types::Enable_if< !Return_tag, RES >::type
00329 return_err(long err) { return err; }
00330
00331 template<typename RES>
00332 static typename L4::Types::Enable_if< !Return_tag, RES >::type
00333 return_ipc_err(l4_msgtag_t, l4_utcb_t *utcb)
00334 { return l4_ipc_to_errno(l4_ipc_error_code(utcb)); }
00335
00336 template<typename RES>
00337 static typename L4::Types::Enable_if< !Return_tag, RES >::type
00338 return_code(l4_msgtag_t tag) { return tag.label(); }
00339
00340 static R call(L4::Cap<class_type> cap,
00341 typename _Elem<ARGS>::arg_type ...a,
00342 l4_utcb_t *utcb = l4_utcb()) throw();
00343 };
00344
00349 template<typename OP, typename CLASS, typename SIG, typename FLAGS = Call>
00350 struct L4_EXPORT Rpc_call;
00351
00359 template<typename IPC, typename SIG> struct _Call;
00360
00362 template<typename IPC, typename R, typename ...ARGS>
00363 struct _Call<IPC, R (ARGS...)>
00364 {
00365 public:
00366 typedef typename IPC::class_type class_type;
00367 typedef typename IPC::result_type result_type;
00368
00369 private:
00370 L4::Cap<class_type> cap() const
00371 {
00372 return L4::Cap<class_type>(reinterpret_cast<l4_cap_idx_t>(this)
00373 & L4_CAP_MASK);
00374 }
00375
00376 public:
00377 result_type operator () (ARGS ...a, l4_utcb_t *utcb = l4_utcb()) const throw()
00378 { return IPC::call(cap(), a..., utcb); }
00379 };
00380
00381 template<typename IPC> struct Call : _Call<IPC, typename IPC::func_type> {};
00382
00394 template<typename OP,
00395 typename CLASS,
00396 typename FLAGS,
00397 typename R,
00398 typename ...ARGS>
00399 struct L4_EXPORT Rpc_call<OP, CLASS, R (ARGS...), FLAGS> :
00400 Rpc_inline_call<OP, CLASS, R (ARGS...), FLAGS>
00401 {
00402 static R call(L4::Cap<CLASS> cap,
00403 typename _Elem<ARGS>::arg_type ...a,
00404 l4_utcb_t *utcb = l4_utcb()) throw();
00405 };
00406
00407 #define L4_INLINE_RPC_SRV_FORWARD(name)
00408 template<typename OBJ> struct fwd
00409 {
00410 OBJ *o;
00411 fwd(OBJ *o) : o(o) {}
00412 template<typename ...ARGS> long call(ARGS ...a)
00413 { return o->op_##name(a...); }
00414 }
00415
00416 #define L4_INLINE_RPC_NF(res, name, args...)
00429 struct name##_t : L4::Ip::Msg::Rpc_inline_call<name##_t, Class, res args>
00430 {
00431 typedef Rpc_inline_call type;
00432 L4_INLINE_RPC_SRV_FORWARD(name);
00433 }
00434
00435 #define L4_INLINE_RPC_NF_OP(op, res, name, args...)
00442 struct name##_t : L4::Ip::Msg::Rpc_inline_call<name##_t, Class, res args>
00443 {
00444 typedef Rpc_inline_call type; enum { Opcode = (op) };
00445 L4_INLINE_RPC_SRV_FORWARD(name);
00446 }
00447
00448 #ifndef DOXYGEN
00449
00450

```

```

00457 #define L4_INLINE_RPC(res, name, args, attr...) res name args
00458 #else
00459 #define L4_INLINE_RPC(res, name, args...) \
00460 L4_INLINE_RPC_NF(res, name, args); L4::Ipcc::Msg::Call<name##_t> name
00461 #endif
00462
00463 #ifdef DOXYGEN
00464
00472 #define L4_INLINE_RPC_OP(op, res, name, args, attr...) res name args
00473 #else
00474 #define L4_INLINE_RPC_OP(op, res, name, args...) \
00475 L4_INLINE_RPC_NF_OP(op, res, name, args); L4::Ipcc::Msg::Call<name##_t> name
00476 #endif
00477
00485 #define L4_RPC_NF(res, name, args...) \
00486 struct name##_t : L4::Ipcc::Msg::Rpc_call<name##_t, Class, res args> \
00487 { \
00488 typedef Rpc_call type; \
00489 L4_INLINE_RPC_SRV_FORWARD(name); \
00490 } \
00491
00500 #define L4_RPC_NF_OP(op, res, name, args...) \
00501 struct name##_t : L4::Ipcc::Msg::Rpc_call<name##_t, Class, res args> \
00502 { \
00503 typedef Rpc_call type; enum { Opcode = (op) }; \
00504 L4_INLINE_RPC_SRV_FORWARD(name); \
00505 } \
00506
00507 #ifdef DOXYGEN
00508
00515 #define L4_RPC(res, name, args, attr...) res name args
00516 #else
00517 #define L4_RPC(res, name, args...) \
00518 L4_RPC_NF(res, name, args); L4::Ipcc::Msg::Call<name##_t> name
00519 #endif
00520
00521 #ifdef DOXYGEN
00522
00530 #define L4_RPC_OP(op, res, name, args, attr...) res name args
00531 #else
00532 #define L4_RPC_OP(op, res, name, args...) \
00533 L4_RPC_NF_OP(op, res, name, args); L4::Ipcc::Msg::Call<name##_t> name
00534 #endif
00535
00536
00541 namespace Detail {
00542
00546 template<typename ...ARGS>
00547 struct Buf
00548 {
00549 public:
00550 template<typename DIR>
00551 static int write(char *, int offset, int)
00552 { return offset; }
00553
00554 template<typename DIR>
00555 static int read(char *, int offset, int, long)
00556 { return offset; }
00557
00558 typedef void Base;
00559 };
00560
00561 template<typename A, typename ...M>
00562 struct Buf<A, M...> : Buf<M...>
00563 {
00564 typedef Buf<M...> Base;
00565
00566 typedef Clnt_xmit<A> xmit;
00567 typedef typename _Elem<A>::arg_type arg_type;
00568 typedef Detail::_Plain<arg_type> plain;
00569
00570 template<typename DIR>
00571 static int
00572 write(char *base, int offset, int limit,
00573 arg_type a, typename _Elem<M>::arg_type ...m)
00574 {
00575 offset = xmit::to_msg(base, offset, limit, plain::deref(a),
00576 typename DIR::dir(), typename DIR::cls());
00577 return Base::template write<DIR>(base, offset, limit, m...);
00578 }
00579
00580 template<typename DIR>
00581 static int
00582 read(char *base, int offset, int limit, long ret,
00583 arg_type a, typename _Elem<M>::arg_type ...m)
00584 {
00585 int r = xmit::from_msg(base, offset, limit, ret, plain::deref(a),

```



```

00586 typename DIR::dir(), typename DIR::cls());
00587 if (L4_LIKELY(r >= 0))
00588 return Base::template read<DIR>(base, r, limit, ret, m...);
00589
00590 if (!_Elem<A>::Is_optional)
00591 return Base::template read<DIR>(base, offset, limit, ret, m...);
00592
00593 return r;
00594 }
00595 };
00596
00597 template <typename ...ARGS> struct _Part
00598 {
00600 typedef Buf<ARGS...> Data;
00601
00602 template<typename DIR>
00603 static int write(void *b, int offset, int limit,
00604 typename _Elem<ARGS>::arg_type ...m)
00605 {
00606 int r = Data::template write<DIR>((char *)b, offset, limit, m...);
00607 if (L4_LIKELY(r >= offset))
00608 return r - offset;
00609 return r;
00610 }
00611
00612 template<typename DIR>
00613 static int read(void *b, int offset, int limit, long ret,
00614 typename _Elem<ARGS>::arg_type ...m)
00615 {
00616 int r = Data::template read<DIR>((char *)b, offset, limit, ret, m...);
00617 if (L4_LIKELY(r >= offset))
00618 return r - offset;
00619 return r;
00620 }
00621 };
00622
00623 template<typename IPC_TYPE, typename OPCODE = void>
00624 struct Part;
00625
00626 // The version without an op-code
00627 template<typename R, typename ...ARGS>
00628 struct Part<R (ARGS...), void> : _Part<ARGS...>
00629 {
00630 typedef Buf<ARGS...> Data;
00631
00632 // write arguments, skipping the dummy opcode
00633 template<typename DIR>
00634 static int write_op(void *b, int offset, int limit,
00635 int /*placeholder for op*/,
00636 typename _Elem<ARGS>::arg_type ...m)
00637 {
00638 int r = Data::template write<DIR>((char *)b, offset, limit, m...);
00639 if (L4_LIKELY(r >= offset))
00640 return r - offset;
00641 return r;
00642 }
00643 };
00644
00645 // Message part with additional opcode
00646 template<typename OPCODE, typename R, typename ...ARGS>
00647 struct Part<R (ARGS...), OPCODE> : _Part<ARGS...>
00648 {
00649 typedef OPCODE opcode_type;
00650 typedef Buf<opcode_type, ARGS...> Data;
00651
00652 // write arguments, including the opcode
00653 template<typename DIR>
00654 static int write_op(void *b, int offset, int limit,
00655 opcode_type op, typename _Elem<ARGS>::arg_type ...m)
00656 {
00657 int r = Data::template write<DIR>((char *)b, offset, limit, op, m...);
00658 if (L4_LIKELY(r >= offset))
00659 return r - offset;
00660 return r;
00661 }
00662 };
00663
00664 } // namespace Detail
00665
00666 //-----
00667 // Implementation of the RPC call
00668 // TODO: Add support for timeout via special RPC argument
00669 // TODO: Add support for passing the UTCB pointer as argument
00670 //
00671 template<typename OP, typename CLASS, typename FLAGS, typename R,
00672 typename ...ARGS>

```

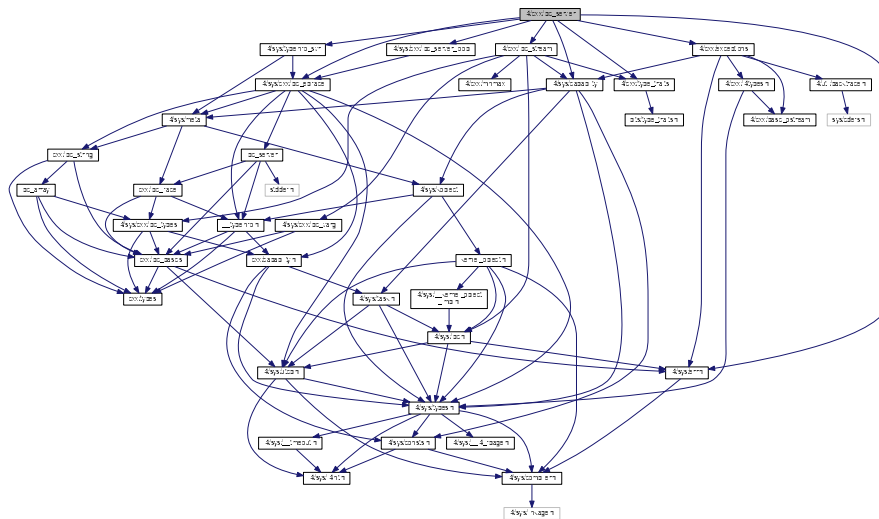
```

00682 inline R
00683 Rpc_inline_call<OP, CLASS, R (ARGS...), FLAGS>::
00684 call(L4::Cap<CLASS> cap,
00685 typename _Elem<ARGS>::arg_type ...a,
00686 l4_utcb_t *utcb) throw()
00687 {
00688 using namespace Ipc::Msg;
00689
00690 typedef typename Kobject_typeid<CLASS>::Iface::Rpc Rpc;
00691 typedef typename Rpc::template Rpc<OP> Opt;
00692 typedef Detail::Part<ipc_type, typename Rpc::opcode_type> Args;
00693
00694 l4_msg_regs_t *mrs = l4_utcb_mr_u(utcb);
00695
00696 // handle in-data part of the arguments
00697 int send_bytes =
00698 Args::template write_op<Do_in_data>(mrs->mr, 0, Mr_bytes,
00699 Opt::Opcode, a...);
00700
00701 if (L4_UNLIKELY(send_bytes < 0))
00702 return return_err<R>(send_bytes);
00703
00704 send_bytes = align_to<l4_umword_t>(send_bytes);
00705 int const send_words = send_bytes / Word_bytes;
00706 // write the in-items part of the message if there is one
00707 int item_bytes =
00708 Args::template write<Do_in_items>(&mrs->mr[send_words], 0,
00709 Mr_bytes - send_bytes, a...);
00710
00711 if (L4_UNLIKELY(item_bytes < 0))
00712 return return_err<R>(item_bytes);
00713
00714 int send_items = item_bytes / Item_bytes;
00715
00716 {
00717 // setup the receive buffers for the RPC call
00718 l4_buf_regs_t *brs = l4_utcb_br_u(utcb);
00719 // XXX: we currently support only one type of receive buffers per call
00720 brs->bdr = 0; // we always start at br[0]
00721
00722 // the limit leaves us at least one register for the zero terminator
00723 // add the buffers given as arguments to the buffer registers
00724 int bytes =
00725 Args::template write<Do_rcv_buffers>(brs->br, 0, Br_bytes - Word_bytes,
00726 a...);
00727
00728 if (L4_UNLIKELY(bytes < 0))
00729 return return_err<R>(bytes);
00730
00731 brs->br[bytes / Word_bytes] = 0;
00732 }
00733
00734 // here we do the actual IPC -----
00735 l4_msgtag_t t;
00736 t = l4_msgtag(CLASS::Protocol, send_words, send_items, 0);
00737 // do the call (Q: do we need support for timeouts?)
00738 if (flags_type::Is_call)
00739 t = l4_ipc_call(cap.cap(), utcb, t, flags_type::timeout());
00740 else
00741 {
00742 t = l4_ipc_send(cap.cap(), utcb, t, flags_type::timeout());
00743 if (L4_UNLIKELY(t.has_error()))
00744 return return_ipc_err<R>(t, utcb);
00745
00746 return return_code<R>(l4_msgtag(0, 0, 0, t.flags()));
00747 }
00748
00749 // unmarshalling starts here -----
00750
00751 // bail out early in the case of an IPC error
00752 if (L4_UNLIKELY(t.has_error()))
00753 return return_ipc_err<R>(t, utcb);
00754
00755 // take the label as return value
00756 long r = t.label();
00757
00758 // bail out on negative error codes too
00759 if (L4_UNLIKELY(r < 0))
00760 return return_err<R>(r);
00761
00762 int const rcv_bytes = t.words() * Word_bytes;
00763
00764 // read the static out-data values to the arguments
00765 int err = Args::template read<Do_out_data>(mrs->mr, 0, rcv_bytes, r, a...);
00766
00767 int const item_limit = t.items() * Item_bytes;

```

## 15.283 I4/cxx/ipc\_server File Reference

```
#include <l4/sys/capability>
#include <l4/sys/typeinfo_svr>
#include <l4/sys/err.h>
#include <l4/cxx/ipc_stream>
#include <l4/sys/cxx/ipc_epiface>
#include <l4/sys/cxx/ipc_server_loop>
#include <l4/cxx/type_traits>
#include <l4/cxx/exceptions>
Include dependency graph for ipc_server:
```



- 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.*

## Namespaces

- [L4](#)  
*L4 low-level kernel interface.*

### 15.283.1 Detailed Description

IPC server loop.

Definition in file [ipc\\_server](#).

## 15.284 ipc\_server

```

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 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025
00026 #pragma once
00027
00028 #include <l4/sys/capability>
00029 #include <l4/sys/typeinfo_svr>
00030 #include <l4/sys/err.h>
00031 #include <l4/cxx/ipc_stream>
00032 #include <l4/sys/cxx/ipc_epiface>
00033 #include <l4/sys/cxx/ipc_server_loop>
00034 #include <l4/cxx/type_traits>
00035 #include <l4/cxx/exceptions>
00036
00037 namespace L4 {
00038
00050 class Server_object : public Epiface
00051 {
00052 public:
00070 virtual int dispatch(unsigned long rights, Ipc::Iostream &ios) = 0;
00071
00072 l4_msgtag_t dispatch(l4_msgtag_t tag, unsigned rights,
00073 l4_utcb_t *utcb)
00074 {
00075 L4::Ipc::Iostream ios(utcb);
00076 ios.tag() = tag;
00077 int r = dispatch(rights, ios);
00078 return ios.prepare_ipc(r);
00079 }
00080
00080 Cap<Kobject> obj_cap() const
00081 { return cap_cast<Kobject>(Epiface::obj_cap()); }
00082 };

```

```

00083
00091 template<typename IFACE, typename BASE = L4::Server_object>
00092 struct Server_object_t : BASE
00093 {
00094 typedef IFACE Interface;
00095
00096 typename BASE::Demand get_buffer_demand() const
00097 { return typename L4::Kobject_typeid<IFACE>::Demand(); }
00098
00099 int dispatch_meta_request(L4::Ipc::Iostream &ios)
00100 { return L4::Util::handle_meta_request<IFACE>(ios); }
00101
00102 template<typename THIS>
00103 static int proto_dispatch(THIS *self, l4_umword_t rights,
00104 L4::Ipc::Iostream &ios)
00105 {
00106 l4_msgtag_t t;
00107 ios >> t;
00108 return Kobject_typeid<IFACE>::proto_dispatch(self, t.label(),
00109 rights, ios);
00110 }
00111 };
00112
00113 template<typename Derived, typename IFACE, typename BASE = L4::Server_object>
00114 struct Server_object_x : Server_object_t<IFACE, BASE>
00115 {
00116 int dispatch(l4_umword_t r, L4::Ipc::Iostream &ios)
00117 {
00118 return Server_object_t<IFACE, BASE>::proto_dispatch(
00119 static_cast<Derived *>(this),
00120 r, ios);
00121 }
00122 };
00123
00124 struct Irq_handler_object : Server_object_t<Kobject> {};
00125
00126 }

```

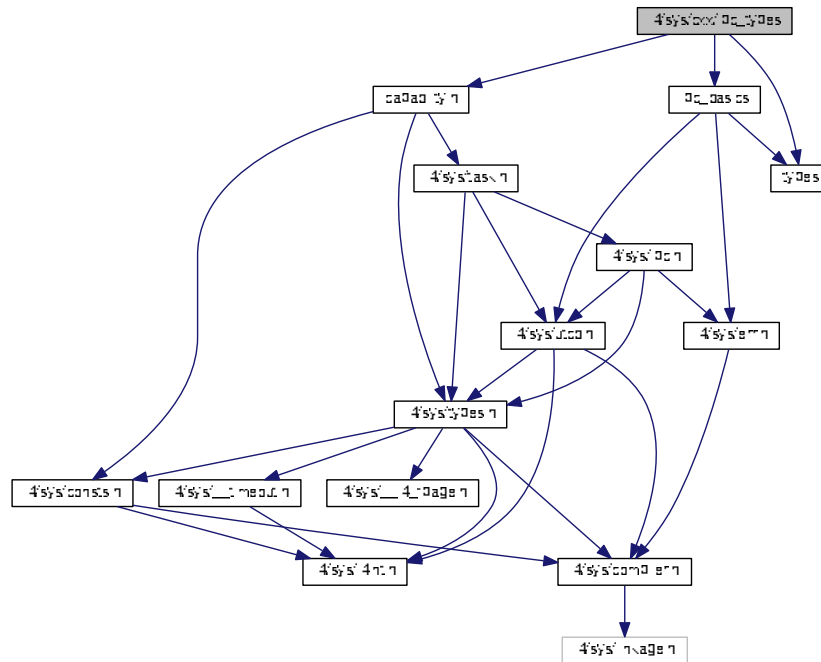
## 15.285 l4/sys/cxx/ipc\_types File Reference

```

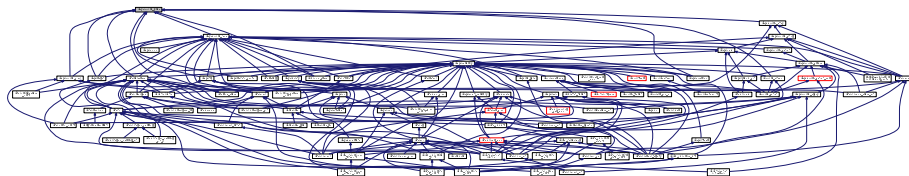
#include "capability.h"
#include "types"
#include "ipc_basics"

```

Include dependency graph for ipc\_types:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct `L4::lpc::Out< T >`  
*Mark an argument as a output value in an RPC signature.*
- 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 capability.*
- class `L4::lpc::Snd_item`  
*RPC wrapper for a send item.*
- class `L4::lpc::Buf_item`  
*RPC wrapper for a receive item.*

- class [L4::ipc::Gen\\_fpage< T >](#)  
*Generic RPC wrapper for [L4](#) flex-pages.*
- class [L4::ipc::Cap< T >](#)  
*Capability type for RPC interfaces (see [L4::Cap< T >](#)).*

## Namespaces

- [L4](#)  
*[L4](#) low-level kernel interface.*
- [L4::ipc](#)  
*IPC related functionality.*
- [L4::ipc::Msg](#)  
*IPC Message related functionality.*

## Typedefs

- typedef Gen\_fpage< Snd\_item > [L4::ipc::Snd\\_fpage](#)  
*Send flex-page.*
- typedef Gen\_fpage< Buf\_item > [L4::ipc::Rcv\\_fpage](#)  
*Rcv flex-page.*

## Functions

- template<typename T >  
[Cap< T >](#) [L4::ipc::make\\_cap](#) ([L4::Cap< T >](#) cap, unsigned rights)  
*Make an [L4::ipc::Cap< T >](#) for the given capability and rights.*
- template<typename T >  
[Cap< T >](#) [L4::ipc::make\\_cap\\_rw](#) ([L4::Cap< T >](#) cap)  
*Make an [L4::ipc::Cap< T >](#) for the given capability with [L4\\_CAP\\_FPAGE\\_RW](#) rights.*
- template<typename T >  
[Cap< T >](#) [L4::ipc::make\\_cap\\_rws](#) ([L4::Cap< T >](#) cap)  
*Make an [L4::ipc::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)  
*Make an [L4::IPC::Cap< T >](#) for the given capability with full fpage and object-specific rights.*

## 15.286 ipc\_types

```

00001 // vi:set ft=c++: -- Mode: C++ --
00002 /*
00003 * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004 *
00005 * This file is part of TUD:OS and distributed under the terms of the
00006 * GNU General Public License 2.
00007 * Please see the COPYING-GPL-2 file for details.
00008 *
00009 * As a special exception, you may use this file as part of a free software
00010 * library without restriction. Specifically, if other files instantiate
00011 * templates or use macros or inline functions from this file, or you compile
00012 * this file and link it with other files to produce an executable, this
00013 * file does not by itself cause the resulting executable to be covered by
00014 * the GNU General Public License. This exception does not however
00015 * invalidate any other reasons why the executable file might be covered by
00016 * the GNU General Public License.

```

```

00017 */
00018 #pragma once
00019
00020 #include "capability.h"
00021 #include "types"
00022 #include "ipc_basics"
00027 namespace L4 {
00028
00030 typedef int Opcode;
00031
00032 namespace Ipc {
00033
00042 template<typename T> struct L4_EXPORT Out;
00043
00044
00052 template<typename T> struct L4_EXPORT In_out
00053 {
00054 T v;
00055 In_out() {}
00056 In_out(T v) : v(v) {}
00057 operator T () const { return v; }
00058 operator T & () { return v; }
00059 };
00060
00061 namespace Msg {
00062 template<typename A> struct Elem< In_out<A *> > : Elem<A *> {};
00063
00064 template<typename A>
00065 struct Svr_xmit< In_out<A *> > : Svr_xmit<A *>, Svr_xmit<A const *>
00066 {
00067 using Svr_xmit<A *>::from_svr;
00068 using Svr_xmit<A const *>::to_svr;
00069 };
00070
00071 template<typename A>
00072 struct Clnt_xmit< In_out<A *> > : Clnt_xmit<A *>, Clnt_xmit<A const *>
00073 {
00074 using Clnt_xmit<A *>::from_msg;
00075 using Clnt_xmit<A const *>::to_msg;
00076 };
00077
00078 template<typename A>
00079 struct Is_valid_rpc_type< In_out<A *> > : Is_valid_rpc_type<A *> {};
00080 template<typename A>
00081 struct Is_valid_rpc_type< In_out<A const *> > : L4::Types::False {};
00082
00083 #ifdef CONFIG_ALLOW_REFS
00084 template<typename A> struct Elem< In_out<A &> > : Elem<A &> {};
00085
00086 template<typename A>
00087 struct Svr_xmit< In_out<A &> > : Svr_xmit<A &>, Svr_xmit<A const &>
00088 {
00089 using Svr_xmit<A &>::from_svr;
00090 using Svr_xmit<A const &>::to_svr;
00091 };
00092
00093 template<typename A>
00094 struct Clnt_xmit< In_out<A &> > : Clnt_xmit<A &>, Clnt_xmit<A const &>
00095 {
00096 using Clnt_xmit<A &>::from_msg;
00097 using Clnt_xmit<A const &>::to_msg;
00098 };
00099
00100 template<typename A>
00101 struct Is_valid_rpc_type< In_out<A &> > : Is_valid_rpc_type<A &> {};
00102 template<typename A>
00103 struct Is_valid_rpc_type< In_out<A const &> > : L4::Types::False {};
00104
00105 #else
00106
00107 template<typename A>
00108 struct Is_valid_rpc_type< In_out<A &> > : L4::Types::False {};
00109
00110 #endif
00111
00112 // Value types don't make sense for output.
00113 template<typename A>
00114 struct Is_valid_rpc_type< In_out<A > > : L4::Types::False {};
00115
00116 }
00117
00118
00127 template<typename T> struct L4_EXPORT As_value
00128 {
00129 typedef T value_type;
00130 T v;
00131 As_value() {}

```



```

00132 As_value(T v) : v(v) {}
00133 operator T () const { return v; }
00134 operator T & () { return v; }
00135 };
00136
00137 namespace Msg {
00138 template<typename T> struct Class< As_value<T> > : Cls_data {};
00139 template<typename T> struct Elem< As_value<T> > : Elem<T> {};
00140 template<typename T> struct Elem< As_value<T> *> : Elem<T *> {};
00141 }
00142
00143
00147 template<typename T> struct L4_EXPORT Opt
00148 {
00149 T _value;
00150 bool _valid;
00151
00153 Opt() : _valid(false) {}
00154
00156 Opt(T value) : _value(value), _valid(true) {}
00157
00159 Opt &operator = (T value)
00160 {
00161 this->_value = value;
00162 this->_valid = true;
00163 return *this;
00164 }
00165
00167 void set_valid(bool valid = true) { _valid = valid; }
00168
00170 T *operator -> () { return &this->_value; }
00172 T const *operator -> () const { return &this->_value; }
00174 T value() const { return this->_value; }
00176 T &value() { return this->_value; }
00178 bool is_valid() const { return this->_valid; }
00179 };
00180
00181 namespace Msg {
00182 template<typename T> struct Elem< Opt<T &> > : Elem<T &>
00183 {
00184 enum { Is_optional = true };
00185 typedef Opt<typename Elem<T &>::svr_type> &svr_arg_type;
00186 typedef Opt<typename Elem<T &>::svr_type> svr_type;
00187 };
00188
00189 template<typename T> struct Elem< Opt<T *> > : Elem<T *>
00190 {
00191 enum { Is_optional = true };
00192 typedef Opt<typename Elem<T *>::svr_type> &svr_arg_type;
00193 typedef Opt<typename Elem<T *>::svr_type> svr_type;
00194 };
00195
00196
00197
00198 template<typename T, typename CLASS>
00199 struct Svr_val_ops<Opt<T>, Dir_out, CLASS> : Svr_noops< Opt<T> >
00200 {
00201 typedef Opt<T> svr_type;
00202 typedef Svr_val_ops<T, Dir_out, CLASS> Native;
00203
00204 using Svr_noops< Opt<T> >::to_svr;
00205 static int to_svr(char *msg, unsigned offset, unsigned limit,
00206 Opt<T> &arg, Dir_out, CLASS)
00207 {
00208 return Native::to_svr(msg, offset, limit, arg.value(), Dir_out(), CLASS());
00209 }
00210
00211 using Svr_noops< Opt<T> >::from_svr;
00212 static int from_svr(char *msg, unsigned offset, unsigned limit, long ret,
00213 svr_type &arg, Dir_out, CLASS)
00214 {
00215 if (arg.is_valid())
00216 return Native::from_svr(msg, offset, limit, ret, arg.value(),
00217 Dir_out(), CLASS());
00218 return offset;
00219 }
00220 };
00221
00222 template<typename T> struct Elem< Opt<T> > : Elem<T>
00223 {
00224 enum { Is_optional = true };
00225 typedef Opt<T> arg_type;
00226 };
00227
00228 template<typename T> struct Elem< Opt<T const *> > : Elem<T const *>
00229 {
00230 enum { Is_optional = true };

```

```

00231 typedef Opt<T const *> arg_type;
00232 };
00233
00234 template<typename T>
00235 struct Is_valid_rpc_type< Opt<T const &> > : L4::Types::False {};
00236
00237 template<typename T, typename CLASS>
00238 struct Clnt_val_ops<Opt<T>, Dir_in, CLASS> : Clnt_noops< Opt<T> >
00239 {
00240 typedef Opt<T> arg_type;
00241 typedef Detail::_Clnt_val_ops<typename Elem<T>::arg_type, Dir_in, CLASS> Native;
00242
00243 using Clnt_noops< Opt<T> >::to_msg;
00244 static int to_msg(char *msg, unsigned offset, unsigned limit,
00245 arg_type arg, Dir_in, CLASS)
00246 {
00247 if (arg.is_valid())
00248 return Native::to_msg(msg, offset, limit,
00249 Detail::_Plain<T>::deref(arg.value()),
00250 Dir_in(), CLASS());
00251 return offset;
00252 }
00253 };
00254
00255 template<typename T> struct Class< Opt<T> > :
00256 Class< typename Detail::_Plain<T>::type > {};
00257 template<typename T> struct Direction< Opt<T> > : Direction<T> {};
00258 }
00259
00260 class L4_EXPORT Small_buf
00261 {
00262 public:
00263 explicit Small_buf(L4::Cap<void> cap, unsigned long flags = 0)
00264 : _data(cap.cap() | L4_RCV_ITEM_SINGLE_CAP | flags) {}
00265
00266 explicit Small_buf(l4_cap_idx_t cap, unsigned long flags = 0)
00267 : _data(cap | L4_RCV_ITEM_SINGLE_CAP | flags) {}
00268
00269 l4_umword_t raw() const { return _data; }
00270 private:
00271 l4_umword_t _data;
00272 };
00273
00274 class Snd_item
00275 {
00276 public:
00277 Snd_item(l4_umword_t base, l4_umword_t data) : _base(base), _data(data) {}
00278
00279 protected:
00280 l4_umword_t _base;
00281 l4_umword_t _data;
00282 };
00283
00284 class Buf_item
00285 {
00286 public:
00287 Buf_item(l4_umword_t base, l4_umword_t data) : _base(base), _data(data) {}
00288
00289 protected:
00290 l4_umword_t _base;
00291 l4_umword_t _data;
00292 };
00293
00294 template< typename T >
00295 class L4_EXPORT Gen_fpage : public T
00296 {
00297 public:
00298 enum Type
00299 {
00300 Special = L4_FPAGE_SPECIAL << 4,
00301 Memory = L4_FPAGE_MEMORY << 4,
00302 Io = L4_FPAGE_IO << 4,
00303 Obj = L4_FPAGE_OBJ << 4
00304 };
00305
00306 enum Map_type
00307 {
00308 Map = L4_MAP_ITEM_MAP,
00309 Grant = L4_MAP_ITEM_GRANT,
00310 };
00311
00312 enum Cacheopt
00313 {
00314 None = 0,
00315 Cached = L4_FPAGE_CACHEABLE << 4,
00316 Buffered = L4_FPAGE_BUFFERABLE << 4,
00317 Uncached = L4_FPAGE_UNCACHEABLE << 4
00318 }

```

```

00347 };
00348
00349 enum Continue
00350 {
00351 Single = 0,
00352 Last = 0,
00353 More = L4_ITEM_CONT,
00354 Compound = L4_ITEM_CONT,
00355 };
00356
00357 private:
00358 Gen_fpage(l4_umword_t d, l4_umword_t fp) : T(d, fp) {}
00359
00360 Gen_fpage(Type type, l4_addr_t base, int order,
00361 unsigned char rights,
00362 l4_addr_t snd_base,
00363 Map_type map_type,
00364 Cacheopt cache, Continue cont)
00365 : T(L4_ITEM_MAP | (snd_base & (~0UL << 10)) | l4_umword_t(map_type) |
00366 l4_umword_t(cache)
00367 | l4_umword_t(cont),
00368 base | l4_umword_t(type) | rights | (l4_umword_t(order) << 6))
00369 {}
00370 public:
00371 Gen_fpage() : T(0, 0) {}
00372 Gen_fpage(l4_fpage_t const &fp, l4_addr_t snd_base = 0,
00373 Map_type map_type = Map,
00374 Cacheopt cache = None, Continue cont = Last)
00375 : T(L4_ITEM_MAP | (snd_base & (~0UL << 10)) | l4_umword_t(map_type) |
00376 l4_umword_t(cache)
00377 | l4_umword_t(cont),
00378 fp.raw)
00379 {}
00380
00381 Gen_fpage(L4::Cap<void> cap, unsigned rights, Map_type map_type = Map)
00382 : T(L4_ITEM_MAP | l4_umword_t(map_type) | (rights & 0xf0),
00383 cap.fpage(rights).raw)
00384 {}
00385
00386 static Gen_fpage<T> obj(l4_addr_t base, int order,
00387 unsigned char rights,
00388 l4_addr_t snd_base = 0,
00389 Map_type map_type = Map,
00390 Continue cont = Last)
00391 {
00392 return Gen_fpage<T>(Obj, base << 12, order, rights, snd_base, map_type, None, cont);
00393 }
00394
00395 static Gen_fpage<T> mem(l4_addr_t base, int order,
00396 unsigned char rights,
00397 l4_addr_t snd_base = 0,
00398 Map_type map_type = Map,
00399 Cacheopt cache = None, Continue cont = Last)
00400 {
00401 return Gen_fpage<T>(Memory, base, order, rights, snd_base,
00402 map_type, cache, cont);
00403 }
00404
00405 static Gen_fpage<T> rmem(l4_addr_t base, int order,
00406 l4_addr_t snd_base,
00407 unsigned char rights, unsigned cap_br)
00408 {
00409 return Gen_fpage<T>({
00410 L4_ITEM_MAP | (snd_base & (~0UL << 10)) | l4_umword_t(Map)
00411 | l4_umword_t(None) | l4_umword_t(Compound) | (cap_br << 8),
00412 base | l4_umword_t(Memory) | rights | (l4_umword_t(order) << 6)});
00413 }
00414
00415 static Gen_fpage<T> io(l4_addr_t base, int order,
00416 unsigned char rights,
00417 l4_addr_t snd_base = 0,
00418 Map_type map_type = Map,
00419 Continue cont = Last)
00420 {
00421 return Gen_fpage<T>(Io, base << 12, order, rights, snd_base, map_type, None, cont);
00422 }
00423
00424 unsigned order() const { return (T::_data >> 6) & 0x3f; }
00425 unsigned snd_order() const { return (T::_data >> 6) & 0x3f; }
00426 unsigned rcv_order() const { return (T::_base >> 6) & 0x3f; }
00427 l4_addr_t base() const { return T::_data & (~0UL << 12); }
00428 l4_addr_t snd_base() const { return T::_base & (~0UL << 10); }
00429 void snd_base(l4_addr_t b) { T::_base = (T::_base & ~(~0UL << 10)) | (b & (~0UL << 10)); }
00430
00431 bool is_valid() const { return T::_base & L4_ITEM_MAP; }

```

```

00438 bool cap_received() const { return (T::_base & 0x3e) == 0x38; }
00450 bool id_received() const { return (T::_base & 0x3e) == 0x3c; }
00460 bool local_id_received() const { return (T::_base & 0x3e) == 0x3e; }
00461
00468 bool is_compound() const { return T::_base & 1; }
00470 l4_umword_t data() const { return T::_data; }
00472 l4_umword_t base_x() const { return T::_base; }
00473 };
00474
00475
00477 typedef Gen_fpage<Snd_item> Snd_fpage;
00479 typedef Gen_fpage<Buf_item> Rcv_fpage;
00480
00481 #ifdef L4_CXX_IPC_SUPPORT_STRINGS
00482 template <typename T, typename B>
00483 class Gen_string : public T
00484 {
00485 public:
00486 Gen_string() : T(0, 0) {}
00487 Gen_string(B buf, unsigned long size)
00488 : T(size << 10, l4_umword_t(buf))
00489 {}
00490
00491 unsigned long len() const { return T::_base >> 10; }
00492 };
00493
00494 typedef Gen_string<Snd_item, void const *> Snd_string;
00495 typedef Gen_string<Buf_item, void *> Rcv_string;
00496 #endif
00497
00498
00499 namespace Msg {
00500
00501 // Snd_fpage are out items
00502 template<> struct Class<L4::Ipc::Snd_fpage> : Cls_item {};
00503
00504 // Rcv_fpage are buffer items
00505 template<> struct Class<L4::Ipc::Rcv_fpage> : Cls_buffer {};
00506
00507 // Remove receive buffers from server-side arguments
00508 template<> struct Elem<L4::Ipc::Rcv_fpage>
00509 {
00510 typedef L4::Ipc::Rcv_fpage arg_type;
00511 typedef void svr_type;
00512 typedef void svr_arg_type;
00513 enum { Is_optional = false };
00514 };
00515
00516 // Rcv_fpage are buffer items
00517 template<> struct Class<L4::Ipc::Small_buf> : Cls_buffer {};
00518
00519 // Remove receive buffers from server-side arguments
00520 template<> struct Elem<L4::Ipc::Small_buf>
00521 {
00522 typedef L4::Ipc::Small_buf arg_type;
00523 typedef void svr_type;
00524 typedef void svr_arg_type;
00525 enum { Is_optional = false };
00526 };
00527 } // namespace Msg
00528
00529 // L4::Cap<> handling
00530
00541 template<typename T> class Cap
00542 {
00543 template<typename O> friend class Cap;
00544 l4_umword_t _cap_n_rights;
00545
00546 public:
00547 enum
00548 {
00554 Rights_mask = 0xff,
00555
00560 Cap_mask = L4_CAP_MASK
00561 };
00562
00564 template<typename O>
00565 Cap(Cap<O> const &o) : _cap_n_rights(o._cap_n_rights)
00566 { T *x = (O*)1; (void)x; }
00567
00569 Cap(L4::Cap<T> cap) : _cap_n_rights(cap.cap() & Cap_mask) {}
00570
00572 template<typename O>
00573 Cap(L4::Cap<O> cap) : _cap_n_rights(cap.cap() & Cap_mask)
00574 { T *x = (O*)1; (void)x; }
00575
00577 Cap() : _cap_n_rights(L4_INVALID_CAP) {}

```

```

00578
00586 Cap(L4::Cap<T> cap, unsigned char rights)
00587 : _cap_n_rights((cap.cap() & Cap_mask) | (rights & Rights_mask)) {}
00588
00594 static Cap from_ci(l4_cap_idx_t c)
00595 { return Cap(L4::Cap<T>(c & Cap_mask), c & Rights_mask); }
00596
00598 L4::Cap<T> cap() const
00599 { return L4::Cap<T>(_cap_n_rights & Cap_mask); }
00600
00602 unsigned rights() const
00603 { return _cap_n_rights & Rights_mask; }
00604
00606 L4::Ipc::Snd_fpage fpage() const
00607 { return L4::Ipc::Snd_fpage(cap(), rights()); }
00608
00610 bool is_valid() const throw()
00611 { return !(_cap_n_rights & L4_INVALID_CAP_BIT); }
00612 };
00613
00620 template<typename T>
00621 Cap<T> make_cap(L4::Cap<T> cap, unsigned rights)
00622 { return Cap<T>(cap, rights); }
00623
00630 template<typename T>
00631 Cap<T> make_cap_rw(L4::Cap<T> cap)
00632 { return Cap<T>(cap, L4_CAP_FPAGE_RW); }
00633
00640 template<typename T>
00641 Cap<T> make_cap_rws(L4::Cap<T> cap)
00642 { return Cap<T>(cap, L4_CAP_FPAGE_RWS); }
00643
00658 template<typename T>
00659 Cap<T> make_cap_full(L4::Cap<T> cap)
00660 { return Cap<T>(cap, L4_CAP_FPAGE_RWSD |
00661 L4_FPAGE_C_OBJ_RIGHTS); }
00661
00662 // caps are special the have an invalid representation
00663 template<typename T> struct L4_EXPORT Opt< Cap<T> >
00664 {
00665 Cap<T> _value;
00666 Opt() {}
00667 Opt(Cap<T> value) : _value(value) {}
00668 Opt(L4::Cap<T> value) : _value(value) {}
00669 Opt &operator = (Cap<T> value)
00670 { this->_value = value; }
00671 Opt &operator = (L4::Cap<T> value)
00672 { this->_value = value; }
00673
00674 Cap<T> value() const { return this->_value; }
00675 bool is_valid() const { return this->_value.is_valid(); }
00676 };
00677
00678
00679 namespace Msg {
00680 // prohibit L4::Cap as argument
00681 template<typename A>
00682 struct Is_valid_rpc_type< L4::Cap<A> > : L4::Types::False {};
00683
00684 template<typename A> struct Class< Cap<A> > : Cls_item {};
00685 template<typename A> struct Elem< Cap<A> >
00686 {
00687 enum { Is_optional = false };
00688 typedef Cap<A> arg_type;
00689 typedef L4::Ipc::Snd_fpage svr_type;
00690 typedef L4::Ipc::Snd_fpage svr_arg_type;
00691 };
00692
00693
00694 template<typename A, typename CLASS>
00695 struct Svr_val_ops<Cap<A>, Dir_in, CLASS> :
00696 Svr_val_ops<L4::Ipc::Snd_fpage, Dir_in, CLASS>
00697 {};
00698
00699 template<typename A, typename CLASS>
00700 struct Clnt_val_ops<Cap<A>, Dir_in, CLASS> :
00701 Clnt_noops< Cap<A> >
00702 {
00703 using Clnt_noops< Cap<A> >::to_msg;
00704
00705 static int to_msg(char *msg, unsigned offset, unsigned limit,
00706 Cap<A> arg, Dir_in, Cls_item)
00707 {
00708 // passing an invalid cap as mandatory argument is an error
00709 // XXX: This checks for a client calling error, we could
00710 // also just ignore this for performance reasons and
00711 // let the client fail badly (Alex: I'd prefer this)

```

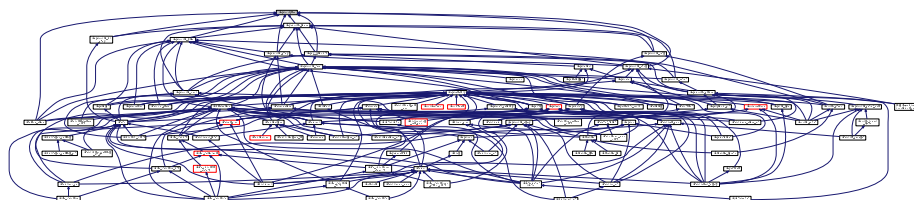
```

00712 if (L4_UNLIKELY(!arg.is_valid()))
00713 return -L4_MSGMISSARG;
00714
00715 return msg_add(msg, offset, limit, arg.fpage());
00716 }
00717 };
00718
00719 template<typename A>
00720 struct Elem<Out<L4::Cap<A> > >
00721 {
00722 enum { Is_optional = false };
00723 typedef L4::Cap<A> arg_type;
00724 typedef Ipc::Cap<A> svr_type;
00725 typedef svr_type &svr_arg_type;
00726 };
00727
00728 template<typename A> struct Direction< Out< L4::Cap<A> > > : Dir_out {};
00729 template<typename A> struct Class< Out< L4::Cap<A> > > : Cls_item {};
00730
00731 template<typename A>
00732 struct Clnt_val_ops< L4::Cap<A>, Dir_out, Cls_item > :
00733 Clnt_noops< L4::Cap<A> >
00734 {
00735 using Clnt_noops< L4::Cap<A> >::to_msg;
00736 static int to_msg(char *msg, unsigned offset, unsigned limit,
00737 L4::Cap<A> arg, Dir_in, Cls_buffer)
00738 {
00739 if (L4_UNLIKELY(!arg.is_valid()))
00740 return -L4_MSGMISSARG; // no buffer inserted
00741 return msg_add(msg, offset, limit, Small_buf(arg));
00742 }
00743 };
00744
00745 template<typename A>
00746 struct Svr_val_ops< L4::Ipc::Cap<A>, Dir_out, Cls_item > :
00747 Svr_noops<Cap<A> &>
00748 {
00749 using Svr_noops<Cap<A> &>::from_svr;
00750 static int from_svr(char *msg, unsigned offset, unsigned limit, long,
00751 Cap<A> arg, Dir_out, Cls_item)
00752 {
00753 if (L4_UNLIKELY(!arg.is_valid()))
00754 return offset; // do not map anything
00755 return msg_add(msg, offset, limit, arg.fpage());
00756 }
00757 };
00758 };
00759
00760 // prohibit a UTCB pointer as normal RPC argument
00761 template<> struct Is_valid_rpc_type<l4_utcb_t *> : L4::Types::False {};
00762
00763 } // namespace Msg
00764 } // namespace Ipc
00765 } // namespace L4
00766

```

## 15.287 l4/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::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.

## Namespaces

- `L4`

*L4* low-level kernel interface.

- `L4::Types`

*L4* basic type helpers for C++.

## 15.288 types

```
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003 * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00004 *
00005 * This file is part of TUD:OS and distributed under the terms of the
00006 * GNU General Public License 2.
00007 * Please see the COPYING-GPL-2 file for details.
00008 *
00009 * As a special exception, you may use this file as part of a free software
00010 * library without restriction. Specifically, if other files instantiate
00011 * templates or use macros or inline functions from this file, or you compile
00012 * this file and link it with other files to produce an executable, this
00013 * file does not by itself cause the resulting executable to be covered by
00014 * the GNU General Public License. This exception does not however
00015 * invalidate any other reasons why the executable file might be covered by
00016 * the GNU General Public License.
00017 */
00018
00019
00020
00021 #pragma once
00022
00023 // very simple type traits for basic L4 functions, for a more complete set
00024 // use <l4/cxx/type_traits> or the standard <type_traits>.
00025
00026 namespace L4 {
00027
00031 namespace Types {
00032
00062 template<typename BITS_ENUM, typename UNDERLYING = unsigned long>
00063 class Flags
00064 {
00065 public:
00067 typedef UNDERLYING value_type;
00069 typedef BITS_ENUM bits_enum_type;
00071 typedef Flags<BITS_ENUM, UNDERLYING> type;
00072
00073 private:
00074 struct Private_bool;
00075 value_type _v;
00076 explicit Flags(value_type v) : _v(v) {}
00077
00078 public:
00080 enum None_type { None };
00081
00090 Flags(None_type) : _v(0) {}
00091
00093 Flags() : _v(0) {}
00094
00103 Flags(BITS_ENUM e) : _v(((value_type)1) << e) {}
00104

```

```

00110 static type from_raw(value_type v) { return type(v); }
00111
00113 operator Private_bool * () const
00114 { return _v != 0 ? (Private_bool *)1 : 0; }
00115
00117 bool operator ! () const { return _v == 0; }
00118
00120 friend type operator | (type lhs, type rhs)
00121 { return type(lhs._v | rhs._v); }
00122
00124 friend type operator | (type lhs, bits_enum_type rhs)
00125 { return lhs | type(rhs); }
00126
00128 friend type operator & (type lhs, type rhs)
00129 { return type(lhs._v & rhs._v); }
00130
00132 friend type operator & (type lhs, bits_enum_type rhs)
00133 { return lhs & type(rhs); }
00134
00136 type &operator |= (type rhs) { _v |= rhs._v; return *this; }
00138 type &operator |= (bits_enum_type rhs) { return operator |= (
type(rhs)); }
00139
00141 type &operator &= (type rhs) { _v &= rhs._v; return *this; }
00143 type &operator &= (bits_enum_type rhs) { return operator &= (
type(rhs)); }
00144
00146 type operator ~ () const { return type(~_v); }
00147
00154 type &clear(bits_enum_type flag) { return operator &= (~
type(flag)); }
00155
00157 value_type as_value() const { return _v; }
00158 };
00159
00165 template< bool V > struct Bool
00166 {
00167 typedef Bool<V> type;
00168 enum { value = V };
00169 };
00170
00173 struct False : Bool<false> {};
00174
00177 struct True : Bool<true> {};
00178
00179 /*****/
00188 template<typename A, typename B>
00189 struct Same : False {};
00190
00191 template<typename A>
00192 struct Same<A, A> : True {};
00193
00194 template<bool EXP, typename T = void> struct Enable_if {};
00195 template<typename T> struct Enable_if<true, T> { typedef T type; };
00196
00197 template<typename T1, typename T2, typename T = void>
00198 struct Enable_if_same : Enable_if<Same<T1, T2>::value, T> {};
00199
00200 template<typename T> struct Remove_const { typedef T type; };
00201 template<typename T> struct Remove_const<T const> { typedef T type; };
00202 template<typename T> struct Remove_volatile { typedef T type; };
00203 template<typename T> struct Remove_volatile<T volatile> { typedef T type; };
00204 template<typename T> struct Remove_cv
00205 { typedef typename Remove_const<typename Remove_volatile<T>::type>::type
type; };
00206
00207 template<typename T> struct Remove_pointer { typedef T type; };
00208 template<typename T> struct Remove_pointer<T*> { typedef T type; };
00209 template<typename T> struct Remove_reference { typedef T type; };
00210 template<typename T> struct Remove_reference<T&> { typedef T type; };
00211 template<typename T> struct Remove_pr { typedef T type; };
00212 template<typename T> struct Remove_pr<T&> { typedef T type; };
00213 template<typename T> struct Remove_pr<T*> { typedef T type; };
00214 } // Types
00215 } // L4

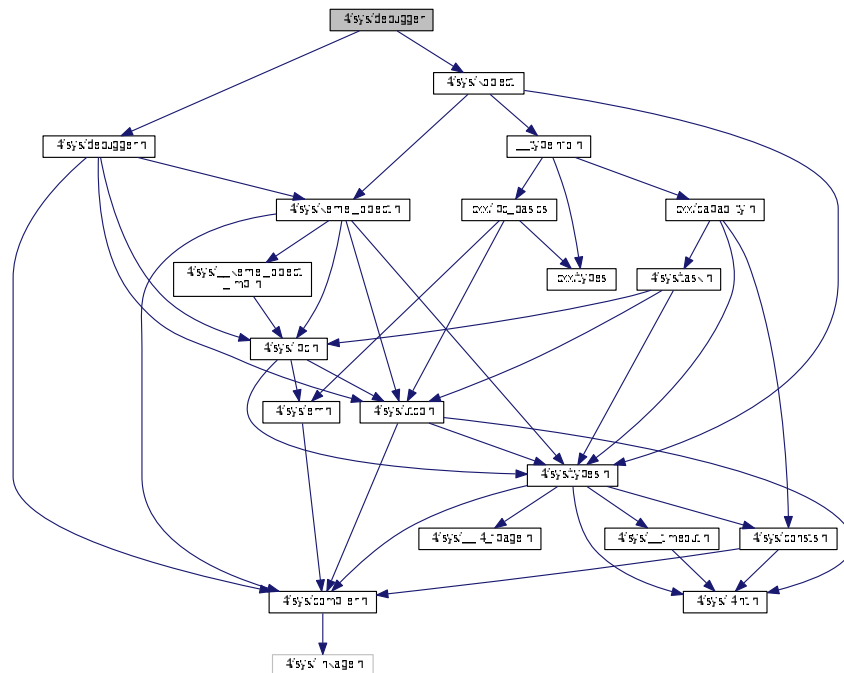
```

## 15.289 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:
```



## Data Structures

- class `L4::Debugger`  
*C++ debugger interface.*

## Namespaces

- L4
- L4 low-level kernel interface.*

### 15.289.1 Detailed Description

The debugger interface specifies common debugging related definitions.

Definition in file [debugger](#).

## 15.290 debugger

```

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 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024 #pragma once
00025
00026 #include <l4/sys/debugger.h>
00027 #include <l4/sys/kobject>
00028
00029 namespace L4 {
00030
00039 class Debugger : public Kobject_t<Debugger, Kobject, L4_PROTO_DEBUGGER>
00040 {
00041 public:
00042 enum
00043 {
00044 Switch_log_on = L4_DEBUGGER_SWITCH_LOG_ON,
00045 Switch_log_off = L4_DEBUGGER_SWITCH_LOG_OFF,
00046 };
00047
00056 l4_msgtag_t set_object_name(const char *name,
00057 l4_utcb_t *utcb = l4_utcb()) throw()
00058 { return l4_debugger_set_object_name_u(cap(), name, utcb); }
00059
00068 unsigned long global_id(l4_utcb_t *utcb = l4_utcb()) throw()
00069 { return l4_debugger_global_id_u(cap(), utcb); }
00070
00080 unsigned long kobj_to_id(l4_addr_t kobjp,
00081 l4_utcb_t *utcb = l4_utcb()) throw()
00082 { return l4_debugger_kobj_to_id_u(cap(), kobjp, utcb); }
00083
00094 int query_log_typeid(const char *name, unsigned idx,
00095 l4_utcb_t *utcb = l4_utcb()) throw()
00096 { return l4_debugger_query_log_typeid_u(cap(), name, idx, utcb); }
00097
00113 int query_log_name(unsigned idx,
00114 char *name, unsigned namelen,
00115 char *shortname, unsigned shortnamelen,
00116 l4_utcb_t *utcb = l4_utcb()) throw()
00117 {
00118 return l4_debugger_query_log_name_u(cap(), idx, name, namelen,
00119 shortname, shortnamelen, utcb);
00120 }
00121
00130 l4_msgtag_t switch_log(const char *name, unsigned on_off,
00131 l4_utcb_t *utcb = l4_utcb()) throw()
00132 { return l4_debugger_switch_log_u(cap(), name, on_off, utcb); }
00133
00145 l4_msgtag_t get_object_name(unsigned id, char *name, unsigned size,
00146 l4_utcb_t *utcb = l4_utcb()) throw()
00147 { return l4_debugger_get_object_name_u(cap(), id, name, size, utcb); }
00148 };
00149 }

```

## 15.291 l4/sys/debugger.h File Reference

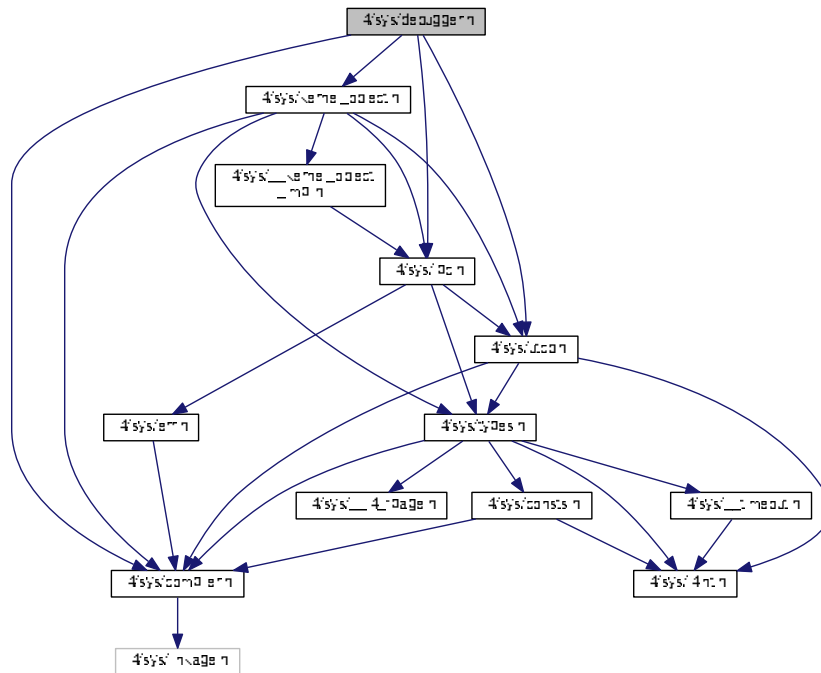
Debugger related definitions.

```

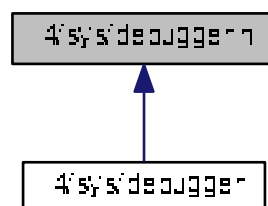
#include <l4/sys/compiler.h>
#include <l4/sys/utcb.h>

```

```
#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:



## 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_get_object_name (l4_cap_idx_t cap, unsigned id, char *name, unsigned size)` `L4_NOTHROW`  
*Get name of the kernel object with Id `id`.*
- `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.*

- int [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.*

- int [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.*

## 15.291.1 Detailed Description

Debugger related definitions.

Definition in file [debugger.h](#).

## 15.291.2 Function Documentation

### 15.291.2.1 l4\_debugger\_get\_object\_name()

```
l4_msgtag_t l4_debugger_get_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 <code>name</code> buffer.                                   |

#### Returns

Syscall return tag

Definition at line [366](#) of file [debugger.h](#).

## 15.291.2.2 l4\_debugger\_query\_log\_name()

```
int 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 shortname to. |
| <i>shortnamelen</i> | Buffer length of shortname.  |

## Return values

|    |         |
|----|---------|
| 0  | Success |
| <0 | Error   |

This is a debugging facility, the call might be invalid.

Definition at line 350 of file [debugger.h](#).

## 15.291.2.3 l4\_debugger\_query\_log\_typeid()

```
int 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 343 of file [debugger.h](#).

#### 15.291.2.4 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 359 of file [debugger.h](#).

## 15.292 debugger.h

```
00001 #pragma once
00002
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 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025
00026 #include <l4/sys/compiler.h>
00027 #include <l4/sys/utcb.h>
00028 #include <l4/sys/ipc.h>
00029
00053 L4_INLINE l4_msgtag_t
00054 l4_debugger_set_object_name(l4_cap_idx_t cap, const char *name)
00055 L4_NOTHROW;
00056
00059 L4_INLINE l4_msgtag_t
00060 l4_debugger_set_object_name_u(l4_cap_idx_t cap, const char *name,
00061 l4_utcb_t *utcb) L4_NOTHROW;
```

```

00073 L4_INLINE l4_msgtag_t
00074 l4_debugger_get_object_name(l4_cap_idx_t cap, unsigned id,
00075 char *name, unsigned size) L4_NOTHROW;
00076
00080 L4_INLINE l4_msgtag_t
00081 l4_debugger_get_object_name_u(l4_cap_idx_t cap, unsigned id,
00082 char *name, unsigned size,
00083 l4_utcb_t *utcb) L4_NOTHROW;
00084
00096 L4_INLINE unsigned long
00097 l4_debugger_global_id(l4_cap_idx_t cap)
00098 L4_NOTHROW;
00099
00102 L4_INLINE unsigned long
00103 l4_debugger_global_id_u(l4_cap_idx_t cap, l4_utcb_t *utcb)
00104 L4_NOTHROW;
00105
00117 L4_INLINE unsigned long
00118 l4_debugger_kobj_to_id(l4_cap_idx_t cap,
00119 l4_addr_t kobjp) L4_NOTHROW;
00120
00123 L4_INLINE unsigned long
00124 l4_debugger_kobj_to_id_u(l4_cap_idx_t cap, l4_addr_t kobjp,
00125 l4_utcb_t *utcb) L4_NOTHROW;
00126
00137 L4_INLINE int
00138 l4_debugger_query_log_typeid(l4_cap_idx_t cap, const char *name,
00139 unsigned idx) L4_NOTHROW;
00140
00144 L4_INLINE int
00145 l4_debugger_query_log_typeid_u(l4_cap_idx_t cap, const char *name,
00146 unsigned idx, l4_utcb_t *utcb)
00147 L4_NOTHROW;
00148
00163 L4_INLINE int
00164 l4_debugger_query_log_name(l4_cap_idx_t cap, unsigned idx,
00165 char *name, unsigned namelen,
00166 char *shortname, unsigned shortnamelen) L4_NOTHROW;
00167
00171 L4_INLINE int
00172 l4_debugger_query_log_name_u(l4_cap_idx_t cap, unsigned idx,
00173 char *name, unsigned namelen,
00174 char *shortname, unsigned shortnamelen,
00175 l4_utcb_t *utcb) L4_NOTHROW;
00176
00186 L4_INLINE l4_msgtag_t
00187 l4_debugger_switch_log(l4_cap_idx_t cap, const char *name,
00188 int on_off) L4_NOTHROW;
00189
00193 L4_INLINE l4_msgtag_t
00194 l4_debugger_switch_log_u(l4_cap_idx_t cap, const char *name, int on_off,
00195 l4_utcb_t *utcb) L4_NOTHROW;
00196
00197 enum
00198 {
00199 L4_DEBUGGER_NAME_SET_OP = 0UL,
00200 L4_DEBUGGER_GLOBAL_ID_OP = 1UL,
00201 L4_DEBUGGER_KOBJ_TO_ID_OP = 2UL,
00202 L4_DEBUGGER_QUERY_LOG_TYPEID_OP = 3UL,
00203 L4_DEBUGGER_SWITCH_LOG_OP = 4UL,
00204 L4_DEBUGGER_NAME_GET_OP = 5UL,
00205 L4_DEBUGGER_QUERY_LOG_NAME_OP = 6UL,
00206 };
00207
00208 enum
00209 {
00210 L4_DEBUGGER_SWITCH_LOG_ON = 1,
00211 L4_DEBUGGER_SWITCH_LOG_OFF = 0,
00212 };
00213
00214 /* IMPLEMENTATION -----*/
00215
00216 #include <l4/sys/kernel_object.h>
00217
00218 L4_INLINE l4_msgtag_t
00219 l4_debugger_set_object_name_u(unsigned long cap,
00220 const char *name, l4_utcb_t *utcb)
00221 L4_NOTHROW
00222 {
00223 unsigned int i;
00224 char *s = (char *)&l4_utcb_mr_u(utcb)->mr[1];
00225 l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_NAME_SET_OP;
00226 for (i = 0;
00227 *name && i < (L4_UTCB_GENERIC_DATA_SIZE - 2) * sizeof(
00228 l4_umword_t) - 1;
00229 ++i, ++name, ++s)
00230 *s = *name;

```

```

00229 *s = 0;
00230 i = (i + sizeof(l4_umword_t) - 1) / sizeof(l4_umword_t);
00231 return l4_invoke_debugger(cap, l4_msgtag(0, i + 1, 0, 0), utcb);
00232 }
00233
00234 L4_INLINE unsigned long
00235 l4_debugger_global_id_u(l4_cap_idx_t cap, l4_utcb_t *utcb)
00236 L4_NOTHROW
00237 {
00238 l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_GLOBAL_ID_OP;
00239 if (l4_error_u(l4_invoke_debugger(cap, l4_msgtag(0, 1, 0, 0), utcb), utcb))
00240 return ~0UL;
00241 return l4_utcb_mr_u(utcb)->mr[0];
00242 }
00243
00244 L4_INLINE unsigned long
00245 l4_debugger_kobj_to_id_u(l4_cap_idx_t cap, l4_addr_t kobjp,
00246 l4_utcb_t *utcb) L4_NOTHROW
00247 {
00248 l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_KOBJ_TO_ID_OP;
00249 l4_utcb_mr_u(utcb)->mr[1] = kobjp;
00250 if (l4_error_u(l4_invoke_debugger(cap, l4_msgtag(0, 2, 0, 0), utcb), utcb))
00251 return ~0UL;
00252 return l4_utcb_mr_u(utcb)->mr[0];
00253 }
00254
00255 L4_INLINE int
00256 l4_debugger_query_log_typeid_u(l4_cap_idx_t cap, const char *name,
00257 unsigned idx,
00258 l4_utcb_t *utcb) L4_NOTHROW
00259 {
00260 unsigned l;
00261 int e;
00262 l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_QUERY_LOG_TYPEID_OP;
00263 l4_utcb_mr_u(utcb)->mr[1] = idx;
00264 l = __builtin_strlen(name);
00265 l = l > 31 ? 31 : l;
00266 __builtin_strncpy((char *)&l4_utcb_mr_u(utcb)->mr[2], name, 31);
00267 l = (l + 1 + sizeof(l4_umword_t) - 1) / sizeof(l4_umword_t);
00268 e = l4_error_u(l4_invoke_debugger(cap, l4_msgtag(0, 2 + l, 0, 0), utcb), utcb);
00269 if (e < 0)
00270 return e;
00271 return l4_utcb_mr_u(utcb)->mr[0];
00272 }
00273
00274 L4_INLINE int
00275 l4_debugger_query_log_name_u(l4_cap_idx_t cap, unsigned idx,
00276 char *name, unsigned namelen,
00277 char *shortname, unsigned shortnamelen,
00278 l4_utcb_t *utcb) L4_NOTHROW
00279 {
00280 int e;
00281 char *n;
00282 l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_QUERY_LOG_NAME_OP;
00283 l4_utcb_mr_u(utcb)->mr[1] = idx;
00284 e = l4_error_u(l4_invoke_debugger(cap, l4_msgtag(0, 2, 0, 0), utcb), utcb);
00285 if (e < 0)
00286 return e;
00287 n = (char *)&l4_utcb_mr_u(utcb)->mr[0];
00288 __builtin_strncpy(name, n, namelen);
00289 name[namelen - 1] = 0;
00290 __builtin_strncpy(shortname, n + __builtin_strlen(n) + 1, shortnamelen);
00291 shortname[shortnamelen - 1] = 0;
00292 return 0;
00293 }
00294
00295 L4_INLINE l4_msgtag_t
00296 l4_debugger_switch_log_u(l4_cap_idx_t cap, const char *name, int on_off,
00297 l4_utcb_t *utcb) L4_NOTHROW
00298 {
00299 unsigned l;
00300 l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_SWITCH_LOG_OP;
00301 l4_utcb_mr_u(utcb)->mr[1] = on_off;
00302 l = __builtin_strlen(name);
00303 l = l > 31 ? 31 : l;
00304 __builtin_strncpy((char *)&l4_utcb_mr_u(utcb)->mr[2], name, 31);
00305 l = (l + 1 + sizeof(l4_umword_t) - 1) / sizeof(l4_umword_t);
00306 return l4_invoke_debugger(cap, l4_msgtag(0, 2 + l, 0, 0), utcb);
00307 }
00308
00309 L4_INLINE l4_msgtag_t
00310 l4_debugger_get_object_name_u(l4_cap_idx_t cap, unsigned id,
00311 char *name, unsigned size,
00312 l4_utcb_t *utcb) L4_NOTHROW
00313 {
00314 l4_msgtag_t t;

```



```

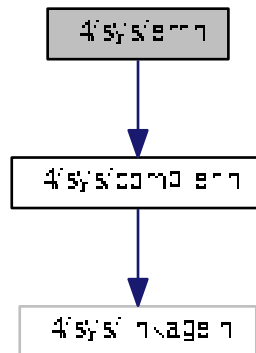
00314 l4_utcb_mr_u(utcb)->mr[0] = L4_DEBUGGER_NAME_GET_OP;
00315 l4_utcb_mr_u(utcb)->mr[1] = id;
00316 t = l4_invoke_debugger(cap, l4_msgtag(0, 2, 0, 0), utcb);
00317 __builtin_strncpy(name, (char *)&l4_utcb_mr_u(utcb)->mr[0], size);
00318 name[size - 1] = 0;
00319 return t;
00320 }
00321
00322
00323 L4_INLINE l4_msgtag_t
00324 l4_debugger_set_object_name(unsigned long cap,
00325 const char *name) L4_NOTHROW
00326 {
00327 return l4_debugger_set_object_name_u(cap, name, l4_utcb());
00328 }
00329
00330 L4_INLINE unsigned long
00331 l4_debugger_global_id(l4_cap_idx_t cap)
00332 L4_NOTHROW
00333 {
00334 return l4_debugger_global_id_u(cap, l4_utcb());
00335 }
00336 L4_INLINE unsigned long
00337 l4_debugger_kobj_to_id(l4_cap_idx_t cap,
00338 l4_addr_t kobjp) L4_NOTHROW
00339 {
00340 return l4_debugger_kobj_to_id_u(cap, kobjp, l4_utcb());
00341 }
00342 L4_INLINE int
00343 l4_debugger_query_log_typeid(l4_cap_idx_t cap, const char *name,
00344 unsigned idx) L4_NOTHROW
00345 {
00346 return l4_debugger_query_log_typeid_u(cap, name, idx, l4_utcb());
00347 }
00348
00349 L4_INLINE int
00350 l4_debugger_query_log_name(l4_cap_idx_t cap, unsigned idx,
00351 char *name, unsigned namelen,
00352 char *shortname, unsigned shortnamelen) L4_NOTHROW
00353 {
00354 return l4_debugger_query_log_name_u(cap, idx, name, namelen,
00355 shortname, shortnamelen, l4_utcb());
00356 }
00357
00358 L4_INLINE l4_msgtag_t
00359 l4_debugger_switch_log(l4_cap_idx_t cap, const char *name,
00360 int on_off) L4_NOTHROW
00361 {
00362 return l4_debugger_switch_log_u(cap, name, on_off, l4_utcb());
00363 }
00364
00365 L4_INLINE l4_msgtag_t
00366 l4_debugger_get_object_name(l4_cap_idx_t cap, unsigned id,
00367 char *name, unsigned size) L4_NOTHROW
00368 {
00369 return l4_debugger_get_object_name_u(cap, id, name, size, l4_utcb());
00370 }

```

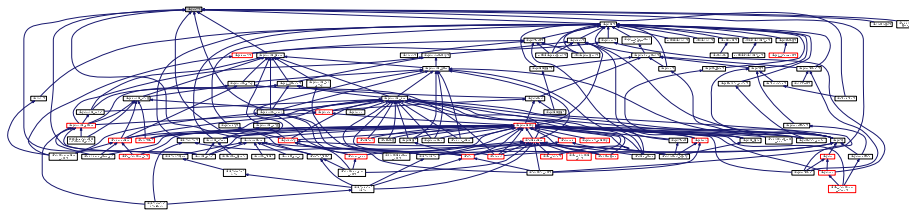
## 15.293 l4/sys/err.h File Reference

Error codes.

```
#include <l4/sys/compiler.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_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_EIPC_LO` = 2000, `L4_EIPC_HI` = 2000 + 0x1f }

*L4 error codes.*

### 15.293.1 Detailed Description

Error codes.

Definition in file `err.h`.

## 15.294 err.h

```

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 * 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 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
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 #pragma once
00024
00025 #include <l4/sys/compiler.h>
00026
00041 enum l4_error_code_t
00042 {
00043 L4_EOK = 0,
00044 L4_EPERM = 1,
00045 L4_ENOENT = 2,
00046 L4_EIO = 5,
00047 L4_ENXIO = 6,
00048 L4_E2BIG = 7,
00049 L4_EAGAIN = 11,
00050 L4_ENOMEM = 12,
00051 L4_EACCESS = 13,
00052 L4_EFAULT = 14,
00053 L4_EBUSY = 16,
00054 L4_EEXIST = 17,
00055 L4_ENODEV = 19,
00056 L4_EINVAL = 22,
00057 L4_ENOSPC = 28,
00058 L4_ERANGE = 34,
00059 L4_ENAMETOOLONG = 36,
00060 L4_ENOSYS = 38,
00061 L4_EBADPROTO = 39,
00062 L4_EADDRNOTAVAIL = 99,
00063 L4_ERRNOMAX = 100,
00065 L4_ENOREPLY = 1000,
00066 L4_EMSGTOOSHORT = 1001,
00067 L4_EMSGTOOLONG = 1002,
00068 L4_EMSSGMISARG = 1003,
00070 L4_EIPC_LO = 2000,
00071 L4_EIPC_HI = 2000 + 0x1f,
00072 };
00073
00074 __BEGIN_DECLS
00075 L4_CV char const *l4sys_errtostr(long err) L4_NOTHROW;
00076 __END_DECLS
00077
00078

```

## 15.295 l4/sys/factory File Reference

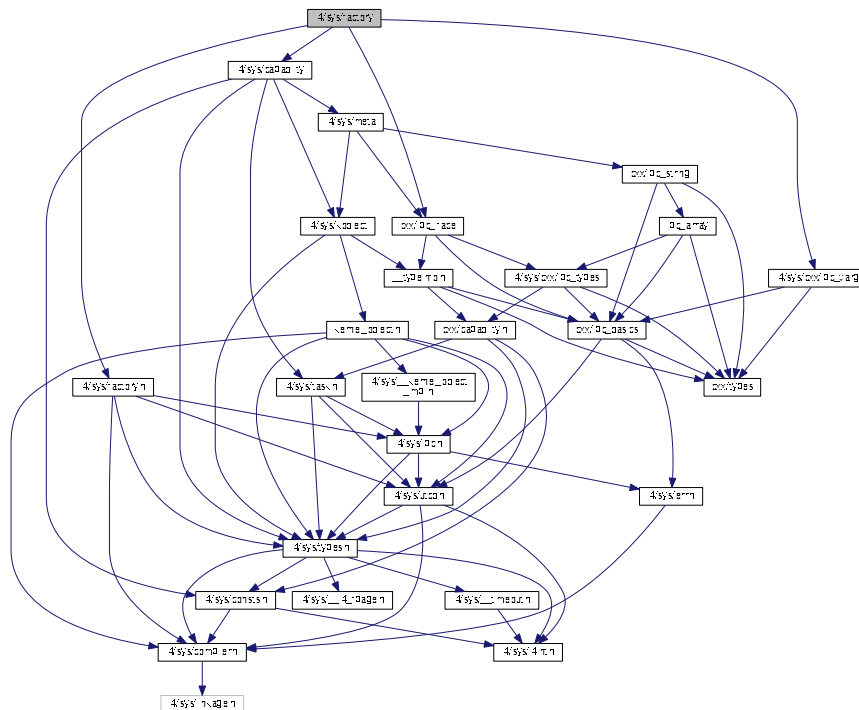
Common factory related definitions.

```

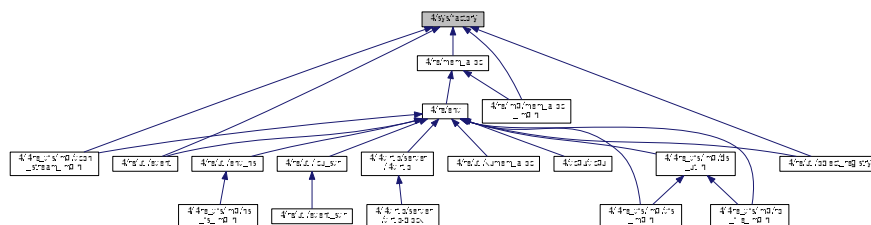
#include <l4/sys/factory.h>
#include <l4/sys/capability>
#include <l4/sys/cxx/ipc_iface>
#include <l4/sys/cxx/ipc_varg>

```

Include dependency graph for factory:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class `L4::Factory`  
*C++ `Factory` interface to create kernel objects.*
- 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

- L4  
*L4 low-level kernel interface.*

### 15.295.1 Detailed Description

Common factory related definitions.

Definition in file [factory](#).

## 15.296 factory

```

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 * This file is part of TUD:OS and distributed under the terms of the
00008 * GNU General Public License 2.
00009 * Please see the COPYING-GPL-2 file for details.
00010 *
00011 * As a special exception, you may use this file as part of a free software
00012 * library without restriction. Specifically, if other files instantiate
00013 * templates or use macros or inline functions from this file, or you compile
00014 * this file and link it with other files to produce an executable, this
00015 * file does not by itself cause the resulting executable to be covered by
00016 * the GNU General Public License. This exception does not however
00017 * invalidate any other reasons why the executable file might be covered by
00018 * the GNU General Public License.
00019 */
00020 #pragma once
00021 #include <l4/sys/factory.h>
00022 #include <l4/sys/capability>
00023 #include <l4/sys/cxx/ipc_iface>
00024 #include <l4/sys/cxx/ipc_varg>
00025 namespace L4 {
00026 class Factory : public Kobject_t<Factory, Kobject, L4_PROTO_FACTORY>
00027 {
00028 public:
00029 typedef l4_mword_t Proto;
00030 struct Nil {};
00031 struct Lstr
00032 {
00033 char const *s;
00034 int len;
00035 Lstr(char const *s, int len) : s(s), len(len) {}
00036 };
00037 class S
00038 {
00039 private:
00040 l4_utcb_t *u;
00041 l4_msgtag_t t;
00042 l4_cap_idx_t f;
00043 public:
00044 S(S const &o)
00045 : u(o.u), t(o.t), f(o.f)
00046 { const_cast<S&>(o).t.raw = 0; }
00047 S(l4_cap_idx_t f, long obj, L4::Cap<void> target,
00048 l4_utcb_t *utcb) throw()
00049 : u(utcb), t(l4_factory_create_start_u(obj, target.cap(), u)), f(f)
00050 {}
00051 ~S()
00052 {
00053 if (t.raw)
00054 l4_factory_create_commit_u(f, t, u);
00055 }
00056 operator l4_msgtag_t ()
00057 {

```

```

00146 l4_msgtag_t r = l4_factory_create_commit_u(f, t, u);
00147 t.raw = 0;
00148 return r;
00149 }
00150
00158 S &operator << (l4_mword_t i)
00159 {
00160 l4_factory_create_add_int_u(i, &t, u);
00161 return *this;
00162 }
00163
00171 S &operator << (l4_umword_t i)
00172 {
00173 l4_factory_create_add_uint_u(i, &t, u);
00174 return *this;
00175 }
00176
00184 S &operator << (char const *s)
00185 {
00186 l4_factory_create_add_str_u(s, &t, u);
00187 return *this;
00188 }
00189
00197 S &operator << (Lstr const &s)
00198 {
00199 l4_factory_create_add_lstr_u(s.s, s.len, &t, u);
00200 return *this;
00201 }
00202
00208 S &operator << (Nil)
00209 {
00210 l4_factory_create_add_nil_u(&t, u);
00211 return *this;
00212 }
00213
00221 S &operator << (l4_fpage_t d)
00222 {
00223 l4_factory_create_add_fpage_u(d, &t, u);
00224 return *this;
00225 }
00226 };
00227
00228
00229 public:
00230
00253 S create(Cap<void> target, long obj, l4_utcb_t *utcb =
l4_utcb()) throw()
00254 {
00255 return S(cap(), obj, target, utcb);
00256 }
00257
00269 template<typename OBJ>
00270 S create(Cap<OBJ> target, l4_utcb_t *utcb = l4_utcb()) throw()
00271 {
00272 return S(cap(), OBJ::Protocol, target, utcb);
00273 }
00274
00275 L4_INLINE_RPC_NF(
00276 l4_msgtag_t, create, (L4::Ipc::Out<
L4::Cap<void> > target, l4_mword_t obj,
00277 L4::Ipc::Varg const *args),
00278 L4::Ipc::Call_t<L4_CAP_FPAGE_S>);
00279
00300 l4_msgtag_t create_task(Cap<Task> const &target_cap,
00301 l4_fpage_t const &utcb_area,
00302 l4_utcb_t *utcb = l4_utcb()) throw()
00303 { return l4_factory_create_task_u(cap(), target_cap.
cap(), utcb_area, utcb); }
00304
00318 l4_msgtag_t create_thread(Cap<Thread> const &target_cap,
00319 l4_utcb_t *utcb = l4_utcb()) throw()
00320 { L4_DEPRECATED("Call create with Cap<Thread> as argument instead.")
00321 { return l4_factory_create_thread_u(cap(), target_cap.
cap(), utcb); } }
00322
00336 l4_msgtag_t create_factory(Cap<Factory> const &target_cap,
00337 unsigned long limit,
00338 l4_utcb_t *utcb = l4_utcb()) throw()
00339 { return l4_factory_create_factory_u(cap(), target_cap.
cap(), limit, utcb); }
00340
00366 l4_msgtag_t create_gate(Cap<Kobject> const &target_cap,
00367 Cap<Thread> const &thread_cap, l4_umword_t label,
00368 l4_utcb_t *utcb = l4_utcb()) throw()
00369 { return l4_factory_create_gate_u(cap(), target_cap.
cap(), thread_cap.cap(), label, utcb); }
00370

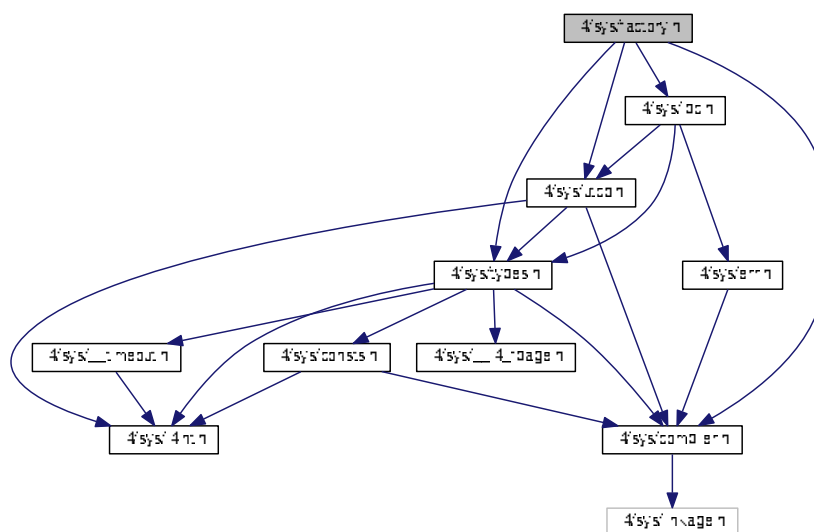
```

```
00384 14_msgtag_t create_irq(Cap<Irq>const &target_cap,
00385 14_utcb_t *utcb = 14_utcb()) throw()
00386 { L4_DEPRECATED("Call create with Cap<Irq> as argument instead.")
00387 { return 14_factory_create_irq_u(cap(), target_cap.
00388 cap(), utcb); }
00389
00400 14_msgtag_t create_vm(Cap<Vm>const &target_cap,
00401 14_utcb_t *utcb = 14_utcb()) throw()
00402 { L4_DEPRECATED("Call create with Cap<Vm> as argument instead.")
00403 { return 14_factory_create_vm_u(cap(), target_cap.cap(), utcb); }
00404
00405 typedef L4::Typeid::Rpc_nocode<create_t> Rpc;
00406
00407 };
00408
00409
00410 }
```

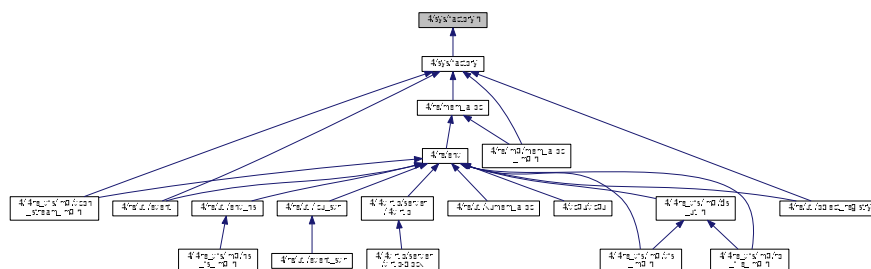
## 15.297 l4/sys/factory.h File Reference

### 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:
```



This graph shows which files directly or indirectly include this file:



## Functions

- [l4\\_msgtag\\_t l4\\_factory\\_create\\_task](#) ([l4\\_cap\\_idx\\_t](#) factory, [l4\\_cap\\_idx\\_t](#) target\_cap, [l4\\_fpage\\_t](#) const utcb\_↵  
area) [L4\\_NOTHROW](#)  
*Create a new task.*
- [l4\\_msgtag\\_t l4\\_factory\\_create\\_task\\_u](#) ([l4\\_cap\\_idx\\_t](#) factory, [l4\\_cap\\_idx\\_t](#) target\_cap, [l4\\_fpage\\_t](#) const  
utcb\_area, [l4\\_utcb\\_t](#) \*utcb) [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\\_thread\\_u](#) ([l4\\_cap\\_idx\\_t](#) factory, [l4\\_cap\\_idx\\_t](#) target\_cap, [l4\\_utcb\\_t](#) \*utcb)  
[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\\_factory\\_u](#) ([l4\\_cap\\_idx\\_t](#) factory, [l4\\_cap\\_idx\\_t](#) target\_cap, unsigned long limit,  
[l4\\_utcb\\_t](#) \*utcb) [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](#) thread\_cap,  
[l4\\_umword\\_t](#) label) [L4\\_NOTHROW](#)  
*Create a new IPC gate.*
- [l4\\_msgtag\\_t l4\\_factory\\_create\\_gate\\_u](#) ([l4\\_cap\\_idx\\_t](#) factory, [l4\\_cap\\_idx\\_t](#) target\_cap, [l4\\_cap\\_idx\\_t](#) thread\_↵  
\_cap, [l4\\_umword\\_t](#) label, [l4\\_utcb\\_t](#) \*utcb) [L4\\_NOTHROW](#)  
*Create a new IPC gate.*
- [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\\_irq\\_u](#) ([l4\\_cap\\_idx\\_t](#) factory, [l4\\_cap\\_idx\\_t](#) target\_cap, [l4\\_utcb\\_t](#) \*utcb) [L4\\_↵](#)  
[NOTHROW](#)  
*Create a new IRQ.*
- [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\\_vm\\_u](#) ([l4\\_cap\\_idx\\_t](#) factory, [l4\\_cap\\_idx\\_t](#) target\_cap, [l4\\_utcb\\_t](#) \*utcb) [L4\\_↵](#)  
[NOTHROW](#)  
*Create a new virtual machine.*

### 15.297.1 Detailed Description

Common factory related definitions.

Definition in file [factory.h](#).

## 15.298 factory.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 * Henning Schild <hschild@os.inf.tu-dresden.de>
00012 * economic rights: Technische Universität Dresden (Germany)
00013 *
```



```

00014 * This file is part of TUD:OS and distributed under the terms of the
00015 * GNU General Public License 2.
00016 * Please see the COPYING-GPL-2 file for details.
00017 *
00018 * As a special exception, you may use this file as part of a free software
00019 * library without restriction. Specifically, if other files instantiate
00020 * templates or use macros or inline functions from this file, or you compile
00021 * this file and link it with other files to produce an executable, this
00022 * file does not by itself cause the resulting executable to be covered by
00023 * the GNU General Public License. This exception does not however
00024 * invalidate any other reasons why the executable file might be covered by
00025 * the GNU General Public License.
00026 */
00027 #pragma once
00028
00029 #include <l4/sys/compiler.h>
00030 #include <l4/sys/types.h>
00031 #include <l4/sys/utcb.h>
00032
00086 L4_INLINE l4_msgtag_t
00087 l4_factory_create_task(l4_cap_idx_t factory,
00088 l4_cap_idx_t target_cap, l4_fpage_t const utcb_area)
00089 L4_NOTHROW;
00096 L4_INLINE l4_msgtag_t
00097 l4_factory_create_task_u(l4_cap_idx_t factory,
00098 l4_cap_idx_t target_cap,
00099 l4_fpage_t const utcb_area, l4_utcb_t *utcb)
00100 L4_NOTHROW;
00111 L4_INLINE l4_msgtag_t
00112 l4_factory_create_thread(l4_cap_idx_t factory,
00113 l4_cap_idx_t target_cap) L4_NOTHROW;
00114
00121 L4_INLINE l4_msgtag_t
00122 l4_factory_create_thread_u(l4_cap_idx_t factory,
00123 l4_cap_idx_t target_cap, l4_utcb_t *utcb)
00124 L4_NOTHROW;
00138 L4_INLINE l4_msgtag_t
00139 l4_factory_create_factory(l4_cap_idx_t factory,
00140 l4_cap_idx_t target_cap,
00141 unsigned long limit) L4_NOTHROW;
00148 L4_INLINE l4_msgtag_t
00149 l4_factory_create_factory_u(l4_cap_idx_t factory,
00150 l4_cap_idx_t target_cap,
00151 unsigned long limit, l4_utcb_t *utcb)
00152 L4_NOTHROW;
00176 L4_INLINE l4_msgtag_t
00177 l4_factory_create_gate(l4_cap_idx_t factory,
00178 l4_cap_idx_t target_cap,
00179 l4_cap_idx_t thread_cap, l4_umword_t label)
00180 L4_NOTHROW;
00187 L4_INLINE l4_msgtag_t
00188 l4_factory_create_gate_u(l4_cap_idx_t factory,
00189 l4_cap_idx_t target_cap,
00190 l4_cap_idx_t thread_cap, l4_umword_t label,
00191 l4_utcb_t *utcb) L4_NOTHROW;
00192
00205 L4_INLINE l4_msgtag_t
00206 l4_factory_create_irq(l4_cap_idx_t factory,
00207 l4_cap_idx_t target_cap) L4_NOTHROW;
00208
00215 L4_INLINE l4_msgtag_t
00216 l4_factory_create_irq_u(l4_cap_idx_t factory,
00217 l4_cap_idx_t target_cap, l4_utcb_t *utcb)
00218 L4_NOTHROW;
00230 L4_INLINE l4_msgtag_t
00231 l4_factory_create_vm(l4_cap_idx_t factory,
00232 l4_cap_idx_t target_cap) L4_NOTHROW;
00233
00240 L4_INLINE l4_msgtag_t
00241 l4_factory_create_vm_u(l4_cap_idx_t factory,
00242 l4_cap_idx_t target_cap, l4_utcb_t *utcb)
00243 L4_NOTHROW;
00244 L4_INLINE l4_msgtag_t
00245 l4_factory_create_start_u(long obj, l4_cap_idx_t target,
00246 l4_utcb_t *utcb) L4_NOTHROW;
00247
00248 L4_INLINE int
00249 l4_factory_create_add_fpage_u(l4_fpage_t d, l4_msgtag_t *tag,
00250 l4_utcb_t *utcb) L4_NOTHROW;

```

```

00251
00252 L4_INLINE int
00253 l4_factory_create_add_int_u(l4_mword_t d, l4_msgtag_t *tag,
00254 l4_utcb_t *utcb) L4_NOTHROW;
00255
00256 L4_INLINE int
00257 l4_factory_create_add_uint_u(l4_umword_t d, l4_msgtag_t *tag,
00258 l4_utcb_t *utcb) L4_NOTHROW;
00259
00260 L4_INLINE int
00261 l4_factory_create_add_str_u(char const *s, l4_msgtag_t *tag,
00262 l4_utcb_t *utcb) L4_NOTHROW;
00263
00264 L4_INLINE int
00265 l4_factory_create_add_lstr_u(char const *s, int len, l4_msgtag_t *tag,
00266 l4_utcb_t *utcb) L4_NOTHROW;
00267
00268 L4_INLINE int
00269 l4_factory_create_add_nil_u(l4_msgtag_t *tag, l4_utcb_t *utcb)
00270 L4_NOTHROW;
00271
00272 L4_INLINE l4_msgtag_t
00273 l4_factory_create_commit_u(l4_cap_idx_t factory, l4_msgtag_t tag,
00274 l4_utcb_t *utcb) L4_NOTHROW;
00275
00276 L4_INLINE l4_msgtag_t
00277 l4_factory_create_u(l4_cap_idx_t factory, long obj, l4_cap_idx_t target,
00278 l4_utcb_t *utcb) L4_NOTHROW;
00279
00280 L4_INLINE l4_msgtag_t
00281 l4_factory_create(l4_cap_idx_t factory, long obj,
00282 l4_cap_idx_t target) L4_NOTHROW;
00283
00284 /* IMPLEMENTATION ----- */
00285
00286 #include <l4/sys/ipc.h>
00287
00288 L4_INLINE l4_msgtag_t
00289 l4_factory_create_task_u(l4_cap_idx_t factory,
00290 l4_cap_idx_t target_cap, l4_fpage_t utcb_area,
00291 l4_utcb_t *u) L4_NOTHROW
00292 {
00293 l4_msgtag_t t;
00294 t = l4_factory_create_start_u(L4_PROTO_TASK, target_cap, u);
00295 l4_factory_create_add_fpage_u(utcb_area, &t, u);
00296 return l4_factory_create_commit_u(factory, t, u);
00297 }
00298
00299 L4_INLINE l4_msgtag_t
00300 l4_factory_create_thread_u(l4_cap_idx_t factory,
00301 l4_cap_idx_t target_cap, l4_utcb_t *u)
00302 L4_NOTHROW
00303 {
00304 return l4_factory_create_u(factory, L4_PROTO_THREAD, target_cap, u);
00305 }
00306
00307 L4_INLINE l4_msgtag_t
00308 l4_factory_create_factory_u(l4_cap_idx_t factory,
00309 l4_cap_idx_t target_cap, unsigned long limit,
00310 l4_utcb_t *u) L4_NOTHROW
00311 {
00312 l4_msgtag_t t;
00313 t = l4_factory_create_start_u(L4_PROTO_FACTORY, target_cap, u);
00314 l4_factory_create_add_uint_u(limit, &t, u);
00315 return l4_factory_create_commit_u(factory, t, u);
00316 }
00317
00318 L4_INLINE l4_msgtag_t
00319 l4_factory_create_gate_u(l4_cap_idx_t factory,
00320 l4_cap_idx_t target_cap,
00321 l4_cap_idx_t thread_cap, l4_umword_t label,
00322 l4_utcb_t *u) L4_NOTHROW
00323 {
00324 l4_msgtag_t t;
00325 l4_msg_regs_t *v;
00326 int items = 0;
00327 t = l4_factory_create_start_u(0, target_cap, u);
00328 l4_factory_create_add_uint_u(label, &t, u);
00329 v = l4_utcb_mr_u(u);
00330 if (!(thread_cap & L4_INVALID_CAP_BIT))
00331 {
00332 items = 1;
00333 v->mr[3] = l4_map_obj_control(0,0);
00334 v->mr[4] = l4_obj_fpage(thread_cap, 0, L4_FPAGE_RWX).
00335 raw;
00336 }
00337 return l4_factory_create_commit_u(factory, t, u);
00338 }

```

```

00335 t = l4_msgtag(l4_msgtag_label(t), l4_msgtag_words(t), items,
00336 l4_msgtag_flags(t));
00337 return l4_factory_create_commit_u(factory, t, u);
00338 }
00339 L4_INLINE l4_msgtag_t
00340 l4_factory_create_irq_u(l4_cap_idx_t factory,
00341 l4_cap_idx_t target_cap, l4_utcb_t *u)
00342 {
00343 L4_NOTHROW
00344 return l4_factory_create_u(factory, L4_PROTO_IRQ_SENDER, target_cap, u);
00345 }
00346 L4_INLINE l4_msgtag_t
00347 l4_factory_create_vm_u(l4_cap_idx_t factory,
00348 l4_cap_idx_t target_cap,
00349 l4_utcb_t *u) L4_NOTHROW
00350 {
00351 return l4_factory_create_u(factory, L4_PROTO_VM, target_cap, u);
00352 }
00353
00354
00355
00356
00357
00358 L4_INLINE l4_msgtag_t
00359 l4_factory_create_task(l4_cap_idx_t factory,
00360 l4_cap_idx_t target_cap, l4_fpage_t const utcb_area)
00361 {
00362 L4_NOTHROW
00363 return l4_factory_create_task_u(factory, target_cap, utcb_area,
00364 l4_utcb());
00365 }
00366 L4_INLINE l4_msgtag_t
00367 l4_factory_create_thread(l4_cap_idx_t factory,
00368 l4_cap_idx_t target_cap) L4_NOTHROW
00369 {
00370 return l4_factory_create_thread_u(factory, target_cap,
00371 l4_utcb());
00372 }
00373 L4_INLINE l4_msgtag_t
00374 l4_factory_create_factory(l4_cap_idx_t factory,
00375 l4_cap_idx_t target_cap, unsigned long limit)
00376 {
00377 L4_NOTHROW
00378 return l4_factory_create_factory_u(factory, target_cap, limit,
00379 l4_utcb());
00380 }
00381 L4_INLINE l4_msgtag_t
00382 l4_factory_create_gate(l4_cap_idx_t factory,
00383 l4_cap_idx_t target_cap,
00384 l4_cap_idx_t thread_cap, l4_umword_t label)
00385 {
00386 L4_NOTHROW
00387 return l4_factory_create_gate_u(factory, target_cap, thread_cap, label,
00388 l4_utcb());
00389 }
00390 L4_INLINE l4_msgtag_t
00391 l4_factory_create_irq(l4_cap_idx_t factory,
00392 l4_cap_idx_t target_cap) L4_NOTHROW
00393 {
00394 return l4_factory_create_irq_u(factory, target_cap,
00395 l4_utcb());
00396 }
00397 L4_INLINE l4_msgtag_t
00398 l4_factory_create_vm(l4_cap_idx_t factory,
00399 l4_cap_idx_t target_cap) L4_NOTHROW
00400 {
00401 return l4_factory_create_vm_u(factory, target_cap,
00402 l4_utcb());
00403 }
00404 L4_INLINE l4_msgtag_t
00405 l4_factory_create_start_u(long obj, l4_cap_idx_t target_cap,
00406 l4_utcb_t *u) L4_NOTHROW
00407 {
00408 l4_msg_regs_t *v = l4_utcb_mr_u(u);
00409 l4_buf_regs_t *b = l4_utcb_br_u(u);
00410 v->mr[0] = obj;
00411 b->bdr = 0;
00412 b->br[0] = target_cap | L4_RCV_ITEM_SINGLE_CAP;

```

```

00411 return l4_msgtag(L4_PROTO_FACTORY, 1, 0, 0);
00412 }
00413
00414 L4_INLINE int
00415 l4_factory_create_add_fpage_u(l4_fpage_t d, l4_msgtag_t *tag,
00416 l4_utcb_t *u) L4_NOTHROW
00417 {
00418 l4_msg_regs_t *v = l4_utcb_mr_u(u);
00419 int w = l4_msgtag_words(*tag);
00420 if (w + 2 > L4_UTCB_GENERIC_DATA_SIZE)
00421 return 0;
00422 v->mr[w] = L4_VARG_TYPE_FPAGE | (sizeof(l4_fpage_t) << 16);
00423 v->mr[w + 1] = d.raw;
00424 w += 2;
00425 tag->raw = (tag->raw & ~0x3fUL) | (w & 0x3f);
00426 return 1;
00427 }
00428
00429 L4_INLINE int
00430 l4_factory_create_add_int_u(l4_mword_t d, l4_msgtag_t *tag,
00431 l4_utcb_t *u) L4_NOTHROW
00432 {
00433 l4_msg_regs_t *v = l4_utcb_mr_u(u);
00434 int w = l4_msgtag_words(*tag);
00435 if (w + 2 > L4_UTCB_GENERIC_DATA_SIZE)
00436 return 0;
00437 v->mr[w] = L4_VARG_TYPE_MWORD | (sizeof(l4_mword_t) << 16);
00438 v->mr[w + 1] = d;
00439 w += 2;
00440 tag->raw = (tag->raw & ~0x3fUL) | (w & 0x3f);
00441 return 1;
00442 }
00443
00444 L4_INLINE int
00445 l4_factory_create_add_uint_u(l4_umword_t d, l4_msgtag_t *tag,
00446 l4_utcb_t *u) L4_NOTHROW
00447 {
00448 l4_msg_regs_t *v = l4_utcb_mr_u(u);
00449 int w = l4_msgtag_words(*tag);
00450 if (w + 2 > L4_UTCB_GENERIC_DATA_SIZE)
00451 return 0;
00452 v->mr[w] = L4_VARG_TYPE_UMWORD | (sizeof(l4_umword_t) << 16);
00453 v->mr[w + 1] = d;
00454 w += 2;
00455 tag->raw = (tag->raw & ~0x3fUL) | (w & 0x3f);
00456 return 1;
00457 }
00458
00459 L4_INLINE int
00460 l4_factory_create_add_str_u(char const *s, l4_msgtag_t *tag,
00461 l4_utcb_t *u) L4_NOTHROW
00462 {
00463 return l4_factory_create_add_lstr_u(s, __builtin_strlen(s), tag, u);
00464 }
00465
00466 L4_INLINE int
00467 l4_factory_create_add_lstr_u(char const *s, int len, l4_msgtag_t *tag,
00468 l4_utcb_t *u) L4_NOTHROW
00469 {
00470 l4_msg_regs_t *v = l4_utcb_mr_u(u);
00471 int w = l4_msgtag_words(*tag);
00472 char *c;
00473 int i;
00474
00475 if (w + 1 + (len + sizeof(l4_umword_t) - 1) / sizeof(l4_umword_t)
00476 > L4_UTCB_GENERIC_DATA_SIZE)
00477 return 0;
00478
00479 v->mr[w] = L4_VARG_TYPE_STRING | (len << 16);
00480 c = (char*)&v->mr[w + 1];
00481 for (i = 0; i < len; ++i)
00482 *c++ = *s++;
00483
00484 w = w + 1 + (len + sizeof(l4_umword_t) - 1) / sizeof(l4_umword_t);
00485
00486 tag->raw = (tag->raw & ~0x3fUL) | (w & 0x3f);
00487 return 1;
00488 }
00489
00490
00491 L4_INLINE int
00492 l4_factory_create_add_nil_u(l4_msgtag_t *tag, l4_utcb_t *utcb)
00493 L4_NOTHROW
00494 {
00495 l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00496 int w = l4_msgtag_words(*tag);
00497 v->mr[w] = L4_VARG_TYPE_NIL;

```

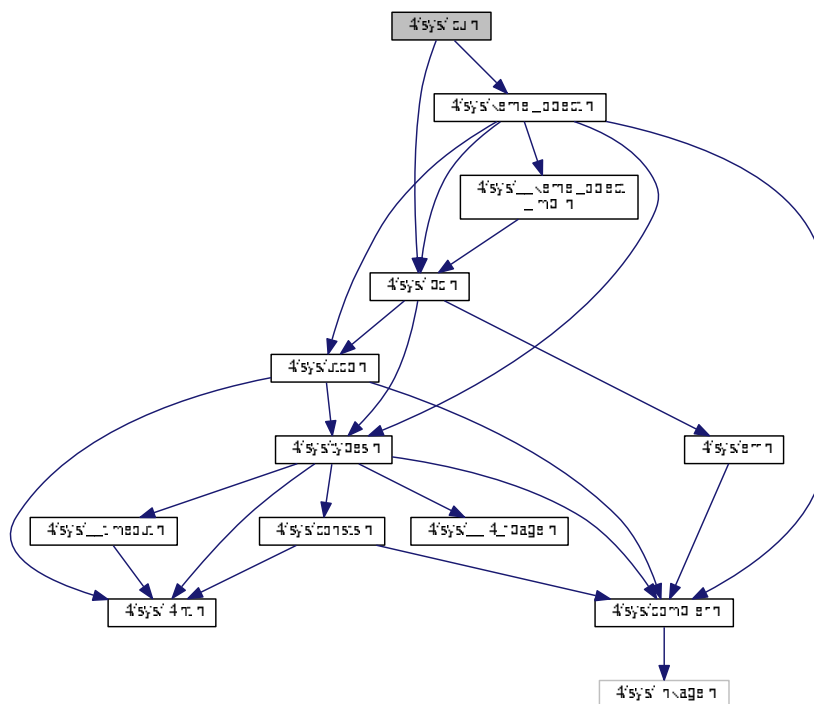
```

00497 ++w;
00498 tag->raw = (tag->raw & ~0x3FUL) | (w & 0x3f);
00499 return 1;
00500 }
00501
00502
00503 L4_INLINE l4_msgtag_t
00504 l4_factory_create_commit_u(l4_cap_idx_t factory, l4_msgtag_t tag,
00505 l4_utcb_t *u) L4_NOTHROW
00506 {
00507 return l4_ipc_call(factory, u, tag, L4_IPC_NEVER);
00508 }
00509
00510 L4_INLINE l4_msgtag_t
00511 l4_factory_create_u(l4_cap_idx_t factory, long obj, l4_cap_idx_t target,
00512 l4_utcb_t *utcb) L4_NOTHROW
00513 {
00514 l4_msgtag_t t = l4_factory_create_start_u(obj, target, utcb);
00515 return l4_factory_create_commit_u(factory, t, utcb);
00516 }
00517
00518
00519 L4_INLINE l4_msgtag_t
00520 l4_factory_create(l4_cap_idx_t factory, long obj,
00521 l4_cap_idx_t target) L4_NOTHROW
00522 {
00523 return l4_factory_create_u(factory, obj, target, l4_utcb());
00524 }

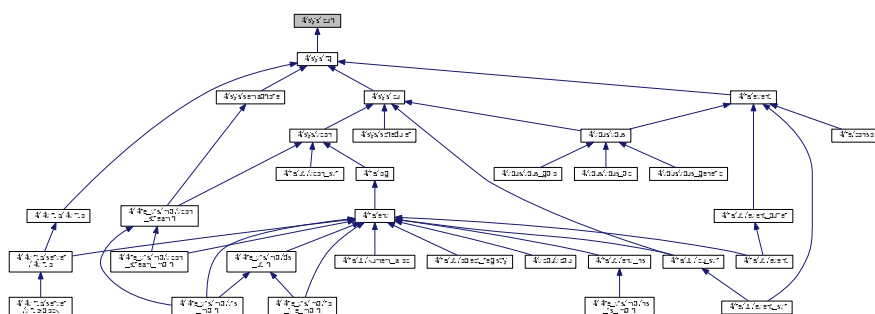
```



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\\_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](#) = 0x3, [L4\\_IRQ\\_F\\_LEVEL\\_LOW](#)  
= 0x7,  
[L4\\_IRQ\\_F\\_POS\\_EDGE](#) = 0x1, [L4\\_IRQ\\_F\\_NEG\\_EDGE](#) = 0x5, [L4\\_IRQ\\_F\\_BOTH\\_EDGE](#) = 0x9, [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 capabilities of the ICU.*
- [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 capabilities of the ICU.*
- [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`  
*Acknowledge 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_N↵ OTHROW`  
*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.*

### 15.301.1 Detailed Description

Interrupt controller.

Definition in file [icu.h](#).

## 15.302 icu.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 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025 #pragma once
00026
00027 #include <l4/sys/kernel_object.h>
00028 #include <l4/sys/ipc.h>
00029
00050 enum L4_icu_flags
00051 {
00059 L4_ICU_FLAG_MSI = 0x80000000,
00060 };
00061
00062
00067 enum L4_irq_mode
00068 {
00070 L4_IRQ_F_NONE = 0,
00071 L4_IRQ_F_LEVEL = 0x2,
00072 L4_IRQ_F_EDGE = 0x0,
00073 L4_IRQ_F_POS = 0x0,
00074 L4_IRQ_F_NEG = 0x4,
00075 L4_IRQ_F_BOTH = 0x8,
00076 L4_IRQ_F_LEVEL_HIGH = 0x3,
00077 L4_IRQ_F_LEVEL_LOW = 0x7,
00078 L4_IRQ_F_POS_EDGE = 0x1,
00079 L4_IRQ_F_NEG_EDGE = 0x5,
00080 L4_IRQ_F_BOTH_EDGE = 0x9,
00081 L4_IRQ_F_MASK = 0xf,
00084 L4_IRQ_F_SET_WAKEUP = 0x10,
00085 L4_IRQ_F_CLEAR_WAKEUP = 0x20,
00086 };

```

```

00087
00088
00093 enum L4_icu_opcode
00094 {
00100 L4_ICU_OP_BIND = 0,
00101
00107 L4_ICU_OP_UNBIND = 1,
00108
00114 L4_ICU_OP_INFO = 2,
00115
00121 L4_ICU_OP_MSI_INFO = 3,
00122
00128 L4_ICU_OP_UNMASK = 4,
00129
00135 L4_ICU_OP_MASK = 5,
00136
00142 L4_ICU_OP_SET_MODE = 6,
00143 };
00144
00145 enum L4_icu_ctl_op
00146 {
00147 L4_ICU_CTL_UNMASK = 0,
00148 L4_ICU_CTL_MASK = 1
00149 };
00150
00151
00159 typedef struct l4_icu_info_t
00160 {
00166 unsigned features;
00167
00171 unsigned nr_irqs;
00172
00176 unsigned nr_msis;
00177 } l4_icu_info_t;
00178
00180 typedef struct l4_icu_msi_info_t
00181 {
00183 l4_uint64_t msi_addr;
00185 l4_uint32_t msi_data;
00186 } l4_icu_msi_info_t;
00187
00204 L4_INLINE l4_msgtag_t
00205 l4_icu_bind(l4_cap_idx_t icu, unsigned irqnum,
00206 l4_cap_idx_t irq) L4_NOTHROW;
00207
00213 L4_INLINE l4_msgtag_t
00214 l4_icu_bind_u(l4_cap_idx_t icu, unsigned irqnum,
00215 l4_cap_idx_t irq,
00216 l4_utcb_t *utcb) L4_NOTHROW;
00217
00227 L4_INLINE l4_msgtag_t
00228 l4_icu_unbind(l4_cap_idx_t icu, unsigned irqnum,
00229 l4_cap_idx_t irq) L4_NOTHROW;
00230
00236 L4_INLINE l4_msgtag_t
00237 l4_icu_unbind_u(l4_cap_idx_t icu, unsigned irqnum,
00238 l4_cap_idx_t irq,
00239 l4_utcb_t *utcb) L4_NOTHROW;
00240
00250 L4_INLINE l4_msgtag_t
00251 l4_icu_set_mode(l4_cap_idx_t icu, unsigned irqnum,
00252 l4_umword_t mode) L4_NOTHROW;
00253
00259 L4_INLINE l4_msgtag_t
00260 l4_icu_set_mode_u(l4_cap_idx_t icu, unsigned irqnum,
00261 l4_umword_t mode,
00262 l4_utcb_t *utcb) L4_NOTHROW;
00263
00274 L4_INLINE l4_msgtag_t
00275 l4_icu_info(l4_cap_idx_t icu, l4_icu_info_t *info)
00276 L4_NOTHROW;
00277
00283 L4_INLINE l4_msgtag_t
00284 l4_icu_info_u(l4_cap_idx_t icu, l4_icu_info_t *info,
00285 l4_utcb_t *utcb) L4_NOTHROW;
00286
00293 L4_INLINE l4_msgtag_t
00294 l4_icu_msi_info(l4_cap_idx_t icu, unsigned irqnum,
00295 l4_uint64_t source,
00296 l4_icu_msi_info_t *msi_info) L4_NOTHROW;
00297
00303 L4_INLINE l4_msgtag_t
00304 l4_icu_msi_info_u(l4_cap_idx_t icu, unsigned irqnum,
00305 l4_uint64_t source,
00306 l4_icu_msi_info_t *msi_info, l4_utcb_t *utcb)
00307 L4_NOTHROW;
00308
00309

```

```

00307
00320 L4_INLINE l4_msgtag_t
00321 l4_icu_unmask(l4_cap_idx_t icu, unsigned irqnum,
00322 l4_umword_t *label,
00323 l4_timeout_t to) L4_NOTHROW;
00323
00330 L4_INLINE l4_msgtag_t
00331 l4_icu_unmask_u(l4_cap_idx_t icu, unsigned irqnum,
00332 l4_umword_t *label,
00333 l4_timeout_t to, l4_utcb_t *utcb)
00334 L4_NOTHROW;
00333
00345 L4_INLINE l4_msgtag_t
00346 l4_icu_mask(l4_cap_idx_t icu, unsigned irqnum,
00347 l4_umword_t *label,
00348 l4_timeout_t to) L4_NOTHROW;
00348
00355 L4_INLINE l4_msgtag_t
00356 l4_icu_mask_u(l4_cap_idx_t icu, unsigned irqnum,
00357 l4_umword_t *label,
00358 l4_timeout_t to, l4_utcb_t *utcb) L4_NOTHROW;
00358
00362 L4_INLINE l4_msgtag_t
00363 l4_icu_control_u(l4_cap_idx_t icu, unsigned irqnum, unsigned op,
00364 l4_umword_t *label, l4_timeout_t to,
00365 l4_utcb_t *utcb) L4_NOTHROW;
00366
00367
00368 /*****
00369 * Implementations
00370 */
00371
00372 L4_INLINE l4_msgtag_t
00373 l4_icu_bind_u(l4_cap_idx_t icu, unsigned irqnum,
00374 l4_cap_idx_t irq,
00375 l4_utcb_t *utcb) L4_NOTHROW
00376 {
00377 l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00378 m->mr[0] = L4_ICU_OP_BIND;
00379 m->mr[1] = irqnum;
00380 m->mr[2] = l4_map_obj_control(0, 0);
00381 m->mr[3] = l4_obj_fpage(irq, 0, L4_FPAGE_RWX).raw;
00382 return l4_ipc_call(icu, utcb, l4_msgtag(L4_PROTO_IRQ, 2, 1, 0),
00383 L4_IPC_NEVER);
00384 }
00383
00384 L4_INLINE l4_msgtag_t
00385 l4_icu_unbind_u(l4_cap_idx_t icu, unsigned irqnum,
00386 l4_cap_idx_t irq,
00387 l4_utcb_t *utcb) L4_NOTHROW
00388 {
00389 l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00390 m->mr[0] = L4_ICU_OP_UNBIND;
00391 m->mr[1] = irqnum;
00392 m->mr[2] = l4_map_obj_control(0, 0);
00393 m->mr[3] = l4_obj_fpage(irq, 0, L4_FPAGE_RWX).raw;
00394 return l4_ipc_call(icu, utcb, l4_msgtag(L4_PROTO_IRQ, 2, 1, 0),
00395 L4_IPC_NEVER);
00396 }
00395
00396 L4_INLINE l4_msgtag_t
00397 l4_icu_info_u(l4_cap_idx_t icu, l4_icu_info_t *info,
00398 l4_utcb_t *utcb) L4_NOTHROW
00399 {
00400 l4_msgtag_t res;
00401 l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00402 m->mr[0] = L4_ICU_OP_INFO;
00403 res = l4_ipc_call(icu, utcb, l4_msgtag(L4_PROTO_IRQ, 1, 0, 0),
00404 L4_IPC_NEVER);
00405 info->features = m->mr[0];
00406 info->nr_irqs = m->mr[1];
00407 info->nr_msis = m->mr[2];
00408 return res;
00409 }
00409
00410 L4_INLINE l4_msgtag_t
00411 l4_icu_msi_info_u(l4_cap_idx_t icu, unsigned irqnum,
00412 l4_uint64_t source,
00413 l4_icu_msi_info_t *msi_info, l4_utcb_t *utcb)
00414 L4_NOTHROW
00415 {
00416 l4_msgtag_t res;
00417 l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00418 m->mr[0] = L4_ICU_OP_MSI_INFO;
00419 m->mr[1] = irqnum;
00420 m->mr64[l4_utcb_mr64_idx(2)] = source;
00421 res = l4_ipc_call(icu, utcb, l4_msgtag(L4_PROTO_IRQ,

```

```

00420 2 + 1 * sizeof(l4_uint64_t)
00421 / sizeof(l4_umword_t),
00422 0, 0), L4_IPC_NEVER);
00423 if (L4_UNLIKELY(l4_msgtag_has_error(res)))
00424 return res;
00425
00426 if (L4_UNLIKELY(l4_msgtag_words(res) * sizeof(
l4_umword_t) < sizeof(*msi_info)))
00427 return res;
00428
00429 __builtin_memcpy(msi_info, &m->mr[0], sizeof(*msi_info));
00430 return res;
00431 }
00432
00433 L4_INLINE l4_msgtag_t
00434 l4_icu_set_mode_u(l4_cap_idx_t icu, unsigned irqnum,
l4_umword_t mode,
00435 l4_utcb_t *utcb) L4_NOTHROW
00436 {
00437 l4_msg_regs_t *mr = l4_utcb_mr_u(utcb);
00438 mr->mr[0] = L4_ICU_OP_SET_MODE;
00439 mr->mr[1] = irqnum;
00440 mr->mr[2] = mode;
00441 return l4_ipc_call(icu, utcb, l4_msgtag(L4_PROTO_IRQ, 3, 0, 0),
L4_IPC_NEVER);
00442 }
00443
00444 L4_INLINE l4_msgtag_t
00445 l4_icu_control_u(l4_cap_idx_t icu, unsigned irqnum, unsigned op,
l4_umword_t *label, l4_timeout_t to,
00446 l4_utcb_t *utcb) L4_NOTHROW
00447 {
00448 l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00449 m->mr[0] = L4_ICU_OP_UNMASK + op;
00450 m->mr[1] = irqnum;
00451 if (label)
00452 return l4_ipc_send_and_wait(icu, utcb, l4_msgtag(
L4_PROTO_IRQ, 2, 0, 0),
00453 label, to);
00454 else
00455 return l4_ipc_send(icu, utcb, l4_msgtag(L4_PROTO_IRQ, 2, 0, 0), to);
00456 }
00457
00458
00459 L4_INLINE l4_msgtag_t
00460 l4_icu_mask_u(l4_cap_idx_t icu, unsigned irqnum,
l4_umword_t *label,
00461 l4_timeout_t to, l4_utcb_t *utcb) L4_NOTHROW
00462 { return l4_icu_control_u(icu, irqnum, L4_ICU_CTL_MASK, label, to, utcb); }
00463
00464 L4_INLINE l4_msgtag_t
00465 l4_icu_unmask_u(l4_cap_idx_t icu, unsigned irqnum,
l4_umword_t *label,
00466 l4_timeout_t to, l4_utcb_t *utcb)
L4_NOTHROW
00467 { return l4_icu_control_u(icu, irqnum, L4_ICU_CTL_UNMASK, label, to, utcb); }
00468
00469
00470
00471
00472 L4_INLINE l4_msgtag_t
00473 l4_icu_bind(l4_cap_idx_t icu, unsigned irqnum,
l4_cap_idx_t irq) L4_NOTHROW
00474 { return l4_icu_bind_u(icu, irqnum, irq, l4_utcb()); }
00475
00476 L4_INLINE l4_msgtag_t
00477 l4_icu_unbind(l4_cap_idx_t icu, unsigned irqnum,
l4_cap_idx_t irq) L4_NOTHROW
00478 { return l4_icu_unbind_u(icu, irqnum, irq, l4_utcb()); }
00479
00480 L4_INLINE l4_msgtag_t
00481 l4_icu_info(l4_cap_idx_t icu, l4_icu_info_t *info)
L4_NOTHROW
00482 { return l4_icu_info_u(icu, info, l4_utcb()); }
00483
00484 L4_INLINE l4_msgtag_t
00485 l4_icu_msi_info(l4_cap_idx_t icu, unsigned irqnum,
l4_uint64_t source,
00486 l4_icu_msi_info_t *msi_info) L4_NOTHROW
00487 { return l4_icu_msi_info_u(icu, irqnum, source, msi_info,
l4_utcb()); }
00488
00489 L4_INLINE l4_msgtag_t
00490 l4_icu_unmask(l4_cap_idx_t icu, unsigned irqnum,
l4_umword_t *label,
00491 l4_timeout_t to) L4_NOTHROW
00492 { return l4_icu_control_u(icu, irqnum, L4_ICU_CTL_UNMASK, label, to, l4_utcb()); }
00493

```

```

00494 L4_INLINE l4_msgtag_t
00495 l4_icu_mask(l4_cap_idx_t icu, unsigned irqnum,
00496 l4_umword_t *label,
00497 l4_timeout_t to) L4_NOTHROW
00498 { return l4_icu_control_u(icu, irqnum, L4_ICU_CTL_MASK, label, to, l4_utcb()); }
00499 L4_INLINE l4_msgtag_t
00500 l4_icu_set_mode(l4_cap_idx_t icu, unsigned irqnum,
00501 l4_umword_t mode) L4_NOTHROW
00502 { return l4_icu_set_mode_u(icu, irqnum, mode, l4_utcb()); }
00503 }

```

## 15.303 l4/sys/ipc.h File Reference

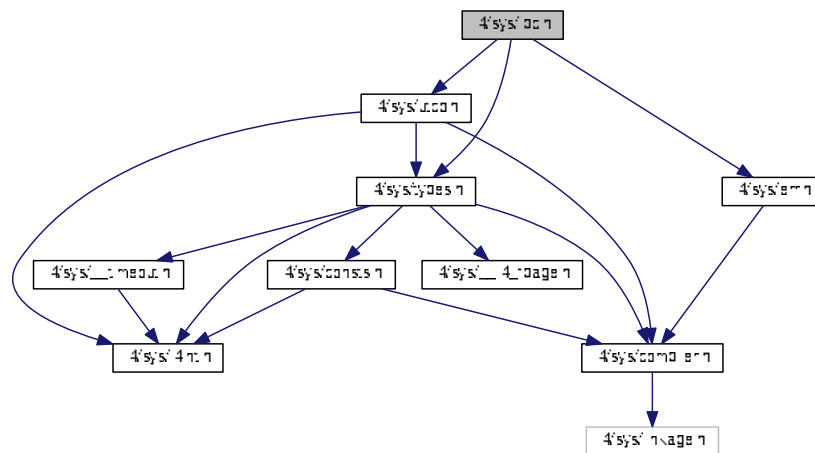
Common IPC interface.

```

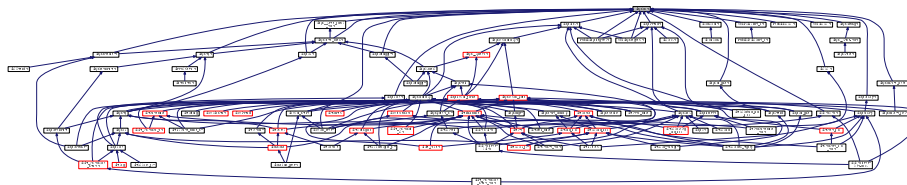
#include <l4/sys/types.h>
#include <l4/sys/utcb.h>
#include <l4/sys/err.h>

```

Include 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_RE↵
MSGCUT = 0x09,
L4_IPC_SEMSGCUT = 0x08 }

```

*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 error code for an object invocation.*
- `long l4_error (l4_msgtag_t tag) L4_NOTHROW`  
*Return error code of a system call return message tag.*
- `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.*
- `long 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_N↵  
OTHROW`  
*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_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_ALWAYS_INLINE l4_ipc (l4_cap_idx_t dest, l4_utcb_t *utcb, l4_umword_t flags, l4_↵  
umword_t label, 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.*
- `int l4_sndfpage_add (l4_fpage_t const snd_fpage, unsigned long snd_base, l4_msgtag_t *tag) L4_NOTH↵  
ROW`  
*Add a flex-page to be sent to the UTCB.*

### 15.303.1 Detailed Description

Common IPC interface.

Definition in file `ipc.h`.

## 15.303.2 Function Documentation

### 15.303.2.1 l4\_ipc\_to\_errno()

```
long l4_ipc_to_errno (
 unsigned long ipc_error_code) [inline]
```

Get a negative error code for the given IPC error code.

#### Parameters

|                             |                                                                                                               |
|-----------------------------|---------------------------------------------------------------------------------------------------------------|
| <code>ipc_error_code</code> | IPC error code as delivered by the kernel. (or returned by the <a href="#">l4_ipc_error_code()</a> function). |
|-----------------------------|---------------------------------------------------------------------------------------------------------------|

#### Returns

negative error code in the range of L4\_EIPC\_LO to L4\_EIPC\_HI.

Definition at line 441 of file [ipc.h](#).

References [L4\\_EIPC\\_LO](#).

## 15.304 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 * 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 * This file is part of TUD:OS and distributed under the terms of the
00014 * GNU General Public License 2.
00015 * Please see the COPYING-GPL-2 file for details.
00016 *
00017 * As a special exception, you may use this file as part of a free software
00018 * library without restriction. Specifically, if other files instantiate
00019 * templates or use macros or inline functions from this file, or you compile
00020 * this file and link it with other files to produce an executable, this
00021 * file does not by itself cause the resulting executable to be covered by
00022 * the GNU General Public License. This exception does not however
00023 * invalidate any other reasons why the executable file might be covered by
00024 * the GNU General Public License.
00025 */
00026 #ifndef __L4SYS__INCLUDE__L4API_FIASCO__IPC_H__
00027 #define __L4SYS__INCLUDE__L4API_FIASCO__IPC_H__
00028
00029 #include <l4/sys/types.h>
00030 #include <l4/sys/utcb.h>
00031 #include <l4/sys/err.h>
00032
00056 /*****
00057 *** IPC result checking
00058 *****/
00059
00075 enum l4_ipc_tcr_error_t
00076 {
00077 L4_IPC_ERROR_MASK = 0x1F,
00078 L4_IPC_SND_ERR_MASK = 0x01,
00080 L4_IPC_ENOT_EXISTENT = 0x04,
00083 L4_IPC_RETIMEOUT = 0x03,
00086 L4_IPC_SETTIMEOUT = 0x02,
```

```

00089 L4_IPC_RECANCELED = 0x07,
00092 L4_IPC_SECANCELED = 0x06,
00095 L4_IPC_REMAPFAILED = 0x11,
00099 L4_IPC_SEMAPFAILED = 0x10,
00102 L4_IPC_RESNDPFTO = 0x0b,
00106 L4_IPC_SESNDPFTO = 0x0a,
00110 L4_IPC_RERCVPFTO = 0x0d,
00114 L4_IPC_SERCVPFTO = 0x0c,
00118 L4_IPC_REABORTED = 0x0f,
00121 L4_IPC_SEABORTED = 0x0e,
00124 L4_IPC_REMSGCUT = 0x09,
00128 L4_IPC_SEMSGCUT = 0x08,
00132 };
00133
00134
00145 L4_INLINE l4_umword_t
00146 l4_ipc_error(l4_msgtag_t tag, l4_utcb_t *utcb)
00147 L4_NOTHROW;
00148
00149
00155 L4_INLINE long
00156 l4_error(l4_msgtag_t tag) L4_NOTHROW;
00157
00158 L4_INLINE long
00159 l4_error_u(l4_msgtag_t tag, l4_utcb_t *utcb) L4_NOTHROW;
00160
00161 /*****
00162 *** IPC results
00163 *****/
00164
00174 L4_INLINE int l4_ipc_is_snd_error(l4_utcb_t *utcb)
00175 L4_NOTHROW;
00176
00185 L4_INLINE int l4_ipc_is_rcv_error(l4_utcb_t *utcb)
00186 L4_NOTHROW;
00187
00196 L4_INLINE int l4_ipc_error_code(l4_utcb_t *utcb)
00197 L4_NOTHROW;
00198
00204 L4_INLINE long l4_ipc_to_errno(unsigned long ipc_error_code)
00205 L4_NOTHROW;
00206
00207 /*****
00208 *** IPC calls
00209 *****/
00210
00228 L4_INLINE l4_msgtag_t
00229 l4_ipc_send(l4_cap_idx_t dest, l4_utcb_t *utcb,
00230 l4_msgtag_t tag,
00231 l4_timeout_t timeout) L4_NOTHROW;
00232
00253 L4_INLINE l4_msgtag_t
00254 l4_ipc_wait(l4_utcb_t *utcb, l4_umword_t *label,
00255 l4_timeout_t timeout) L4_NOTHROW;
00256
00257
00277 L4_INLINE l4_msgtag_t
00278 l4_ipc_receive(l4_cap_idx_t object, l4_utcb_t *utcb,
00279 l4_timeout_t timeout) L4_NOTHROW;
00280
00297 L4_INLINE l4_msgtag_t
00298 l4_ipc_call(l4_cap_idx_t object, l4_utcb_t *utcb,
00299 l4_msgtag_t tag,
00300 l4_timeout_t timeout) L4_NOTHROW;
00301
00321 L4_INLINE l4_msgtag_t
00322 l4_ipc_reply_and_wait(l4_utcb_t *utcb, l4_msgtag_t tag,
00323 l4_umword_t *label, l4_timeout_t timeout)
00324 L4_NOTHROW;
00325
00344 L4_INLINE l4_msgtag_t
00345 l4_ipc_send_and_wait(l4_cap_idx_t dest,
00346 l4_utcb_t *utcb, l4_msgtag_t tag,
00347 l4_umword_t *label, l4_timeout_t timeout)
00348 L4_NOTHROW;
00349
00354 #if 0
00355
00365 L4_INLINE l4_msgtag_t
00366 l4_ipc_wait_next_period(l4_utcb_t *utcb,
00367 l4_umword_t *label,
00368 l4_timeout_t timeout);
00369
00370 #endif

```



```

00371
00386 L4_ALWAYS_INLINE l4_msgtag_t
00387 l4_ipc(l4_cap_idx_t dest,
00388 l4_utcb_t *utcb,
00389 l4_umword_t flags,
00390 l4_umword_t slabel,
00391 l4_msgtag_t tag,
00392 l4_umword_t *rlabel,
00393 l4_timeout_t timeout) L4_NOTHROW;
00394
00409 L4_INLINE l4_msgtag_t
00410 l4_ipc_sleep(l4_timeout_t timeout) L4_NOTHROW;
00411
00424 L4_INLINE int
00425 l4_sndfpage_add(l4_fpage_t const snd_fpage, unsigned long snd_base,
00426 l4_msgtag_t *tag) L4_NOTHROW;
00427
00428 /*
00429 * \internal
00430 * \ingroup l4_ipc_api
00431 */
00432 L4_INLINE int
00433 l4_sndfpage_add_u(l4_fpage_t const snd_fpage, unsigned long snd_base,
00434 l4_msgtag_t *tag, l4_utcb_t *utcb)
00435 L4_NOTHROW;
00436
00437 /*****
00438 * Implementations
00439 *****/
00440
00441 L4_INLINE long l4_ipc_to_errno(unsigned long ipc_error_code)
00442 L4_NOTHROW
00443 { return -(L4_EIPC_LO + ipc_error_code); }
00444
00444 L4_INLINE l4_msgtag_t
00445 l4_ipc_call(l4_cap_idx_t dest, l4_utcb_t *utcb,
00446 l4_msgtag_t tag,
00447 l4_timeout_t timeout) L4_NOTHROW
00448 {
00449 return l4_ipc(dest, utcb, L4_SYSF_CALL, 0, tag, 0, timeout);
00450 }
00451
00452 L4_INLINE l4_msgtag_t
00453 l4_ipc_reply_and_wait(l4_utcb_t *utcb, l4_msgtag_t tag,
00454 l4_umword_t *label,
00455 l4_timeout_t timeout) L4_NOTHROW
00456 {
00457 return l4_ipc(L4_INVALID_CAP, utcb, L4_SYSF_REPLY_AND_WAIT, 0,
00458 tag, label, timeout);
00459 }
00460
00460 L4_INLINE l4_msgtag_t
00461 l4_ipc_send_and_wait(l4_cap_idx_t dest,
00462 l4_utcb_t *utcb,
00463 l4_msgtag_t tag,
00464 l4_umword_t *src,
00465 l4_timeout_t timeout) L4_NOTHROW
00466 {
00467 return l4_ipc(dest, utcb, L4_SYSF_SEND_AND_WAIT, 0, tag, src, timeout);
00468 }
00469
00469 L4_INLINE l4_msgtag_t
00470 l4_ipc_send(l4_cap_idx_t dest, l4_utcb_t *utcb,
00471 l4_msgtag_t tag,
00472 l4_timeout_t timeout) L4_NOTHROW
00473 {
00474 return l4_ipc(dest, utcb, L4_SYSF_SEND, 0, tag, 0, timeout);
00475 }
00476
00477 L4_INLINE l4_msgtag_t
00478 l4_ipc_wait(l4_utcb_t *utcb, l4_umword_t *src,
00479 l4_timeout_t timeout) L4_NOTHROW
00480 {
00481 l4_msgtag_t t;
00482 t.raw = 0;
00483 return l4_ipc(L4_INVALID_CAP, utcb, L4_SYSF_WAIT, 0, t, src, timeout);
00484 }
00485
00486 L4_INLINE l4_msgtag_t
00487 l4_ipc_receive(l4_cap_idx_t src, l4_utcb_t *utcb,
00488 l4_timeout_t timeout) L4_NOTHROW
00489 {
00490 l4_msgtag_t t;
00491 t.raw = 0;
00492 return l4_ipc(src, utcb, L4_SYSF_RECV, 0, t, 0, timeout);
00493 }

```

```

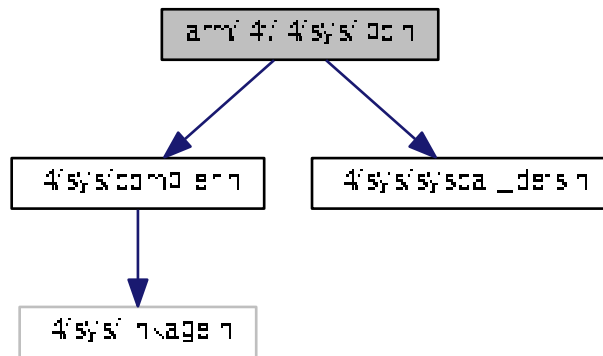
00494
00495 L4_INLINE l4_msgtag_t
00496 l4_ipc_sleep(l4_timeout_t timeout) L4_NOTHROW
00497 { return l4_ipc_receive(L4_INVALID_CAP, NULL, timeout); }
00498
00499 L4_INLINE l4_umword_t
00500 l4_ipc_error(l4_msgtag_t tag, l4_utcb_t *utcb)
00501 L4_NOTHROW
00502 {
00503 if (!l4_msgtag_has_error(tag))
00504 return 0;
00505 return l4_utcb_tcr_u(utcb)->error & L4_IPC_ERROR_MASK;
00506 }
00507 L4_INLINE long
00508 l4_error_u(l4_msgtag_t tag, l4_utcb_t *u) L4_NOTHROW
00509 {
00510 if (l4_msgtag_has_error(tag))
00511 return l4_ipc_to_errno(l4_utcb_tcr_u(u)->error &
00512 L4_IPC_ERROR_MASK);
00513 return l4_msgtag_label(tag);
00514 }
00515
00516 L4_INLINE long
00517 l4_error(l4_msgtag_t tag) L4_NOTHROW
00518 {
00519 return l4_error_u(tag, l4_utcb());
00520 }
00521
00522
00523 L4_INLINE int l4_ipc_is_snd_error(l4_utcb_t *u)
00524 L4_NOTHROW
00525 { return (l4_utcb_tcr_u(u)->error & 1) != 0; }
00526
00527 L4_INLINE int l4_ipc_is_rcv_error(l4_utcb_t *u)
00528 L4_NOTHROW
00529 { return l4_utcb_tcr_u(u)->error & 1; }
00530
00531 L4_INLINE int l4_ipc_error_code(l4_utcb_t *u)
00532 L4_NOTHROW
00533 { return l4_utcb_tcr_u(u)->error & L4_IPC_ERROR_MASK; }
00534
00535 /*
00536 * \internal
00537 * \ingroup l4_ipc_api
00538 */
00539 L4_INLINE int
00540 l4_sndfpage_add_u(l4_fpage_t const snd_fpage, unsigned long snd_base,
00541 l4_msgtag_t *tag, l4_utcb_t *utcb)
00542 L4_NOTHROW
00543 {
00544 l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00545 int i = l4_msgtag_words(*tag) + 2 * l4_msgtag_items(*tag);
00546 if (i >= L4_UTCB_GENERIC_DATA_SIZE - 1)
00547 return -L4_ENOMEM;
00548 v->mr[i] = snd_base | L4_ITEM_MAP | L4_ITEM_CONT;
00549 v->mr[i + 1] = snd_fpage.raw;
00550 *tag = l4_msgtag(l4_msgtag_label(*tag),
00551 l4_msgtag_words(*tag),
00552 l4_msgtag_items(*tag) + 1, l4_msgtag_flags(*tag));
00553 return 0;
00554 }
00555
00556 L4_INLINE int
00557 l4_sndfpage_add(l4_fpage_t const snd_fpage, unsigned long snd_base,
00558 l4_msgtag_t *tag) L4_NOTHROW
00559 {
00560 return l4_sndfpage_add_u(snd_fpage, snd_base, tag, l4_utcb());
00561 }
00562
00563 #endif /* ! __L4SYS__INCLUDE__L4API_FIASCO__IPC_H__ */

```

## 15.305 arm/l4f/l4/sys/ipc.h File Reference

[L4 IPC System Calls, ARM.](#)

```
#include <l4/sys/compiler.h>
#include <l4/sys/syscall_defs.h>
Include dependency graph for ipc.h:
```



## Functions

- [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.

### 15.305.1 Detailed Description

[L4](#) IPC System Calls, ARM.

Definition in file [ipc.h](#).

## 15.306 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 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024 #pragma once
00025
00026 #include_next <l4/sys/ipc.h>
```

```

00027
00028 #ifdef __GNUC__
00029
00030 #include <l4/sys/compiler.h>
00031 #include <l4/sys/syscall_defs.h>
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 register l4_umword_t _dest __asm__("r2") = dest | flags;
00042 register l4_umword_t _timeout __asm__("r3") = timeout.raw;
00043 register l4_umword_t _tag __asm__("r0") = tag.raw;
00044 register l4_umword_t _label __asm__("r4") = slabel;
00045 (void)utcb;
00046
00047 __asm__ __volatile__
00048 ("mov lr, pc\n"
00049 "mov pc, %[sc]\n"
00050 :
00051 "+r" (_dest),
00052 "+r" (_timeout),
00053 "+r" (_label),
00054 "+r" (_tag)
00055 :
00056 [sc] "i" (L4_SYSCALL_INVOKE)
00057 :
00058 "cc", "memory", "lr");
00059
00060 if (rlabel)
00061 *rlabel = _label;
00062 tag.raw = _tag;
00063
00064 return tag;
00065 }
00066
00067 #endif //__GNUC__

```

## 15.307 x86/l4/l4/sys/ipc.h File Reference

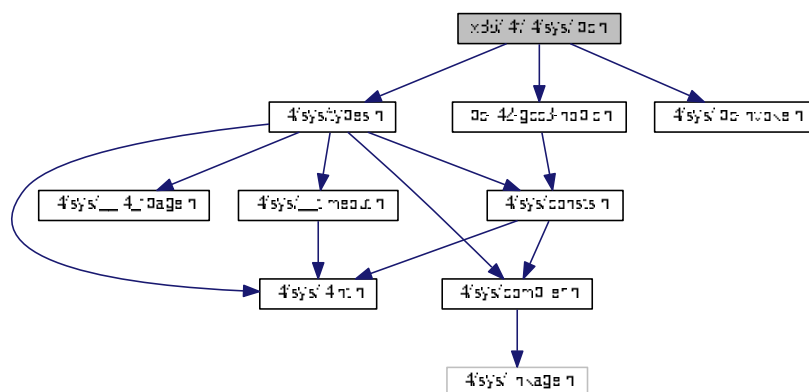
[L4 IPC System Calls, x86.](#)

```

#include <l4/sys/types.h>
#include <l4/sys/ipc-invoke.h>
#include "ipc-l42-gcc3-nopic.h"

```

Include dependency graph for ipc.h:



### 15.307.1 Detailed Description

[L4 IPC System Calls](#), x86.

Definition in file [ipc.h](#).

## 15.308 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 * Lars Reuther <reuther@os.inf.tu-dresden.de>
00010 * economic rights: Technische Universität Dresden (Germany)
00011 *
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025 #ifndef __L4_IPC_H__
00026 #define __L4_IPC_H__
00027
00028 #include <l4/sys/types.h>
00029
00030 #include_next <l4/sys/ipc.h>
00031
00032 /*****
00033 *** Implementation
00034 *****/
00035
00036 #include <l4/sys/ipc-invoke.h>
00037 #include "ipc-l42-gcc3-nopic.h"
00038
00039 #endif /* !__L4_IPC_H__ */

```

## 15.309 l4/sys/ipc\_gate File Reference

The C++ IPC gate interface.

```

#include <l4/sys/ipc_gate.h>
#include <l4/sys/capability>
#include <l4/sys/cxx/ipc_iface>

```



### 15.309.1 Detailed Description

The C++ IPC gate interface.

Definition in file [ipc\\_gate](#).

## 15.310 ipc\_gate

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00002 /*
00003 * (c) 2009-2010 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 * This file is part of TUD:OS and distributed under the terms of the
00008 * GNU General Public License 2.
00009 * Please see the COPYING-GPL-2 file for details.
00010 *
00011 * As a special exception, you may use this file as part of a free software
00012 * library without restriction. Specifically, if other files instantiate
00013 * templates or use macros or inline functions from this file, or you compile
00014 * this file and link it with other files to produce an executable, this
00015 * file does not by itself cause the resulting executable to be covered by
00016 * the GNU General Public License. This exception does not however
00017 * invalidate any other reasons why the executable file might be covered by
00018 * the GNU General Public License.
00019 */
00020 #pragma once
00021
00022 #include <l4/sys/ipc_gate.h>
00023 #include <l4/sys/capability>
00024 #include <l4/sys/cxx/ipc_iface>
00025
00026 namespace L4 {
00027
00028 class Thread;
00029
00030 class L4_EXPORT Ipc_gate :
00031 public Kobject_t<Ipc_gate, Kobject, L4_PROTO_KOBJECT,
00032 Type_info::Demand_t<1> >
00033 {
00034 public:
00035 L4_INLINE_RPC_OP(L4_IPC_GATE_BIND_OP,
00036 l4_msgtag_t, bind_thread, (Ipc::Cap<Thread> t,
00037 l4_umword_t label));
00038
00039 L4_INLINE_RPC_OP(L4_IPC_GATE_GET_INFO_OP,
00040 l4_msgtag_t, get_infos, (l4_umword_t *label));
00041
00042 typedef L4::Typeid::Rpcsys<bind_thread_t, get_infos_t>
00043 Rpcsys;
00044 };
00045 }
00046

```

## 15.311 l4/sys/ipc\_gate.h File Reference

The C IPC gate interface.

```

#include <l4/sys/utcb.h>
#include <l4/sys/types.h>

```





### 15.311.1 Detailed Description

The C IPC gate interface.

IPC gates are used to create secure communication channels between threads. An IPC gate object can be created using the [Factory](#) interface. With [l4\\_ipc\\_gate\\_bind\\_thread\(\)](#) a thread is bound to an IPC gate which then receives all messages sent to that IPC gate.

The [l4\\_ipc\\_gate\\_bind\\_thread\(\)](#) call allows to assign each IPC gate a kernel protected, machine-word sized payload called a *label*. It securely identifies the gate. The lower two bits of the *label* can be used to encode rights bits. The kernel combines these bits with the capability rights, so a programmer usually should not pick the lower two bits for the *label*. The *label* is only visible in the task which is running the thread the IPC gate was bound to and cannot be altered by the sender.

Definition in file [ipc\\_gate.h](#).

## 15.312 ipc\_gate.h

```

00001
00018 /*
00019 * (c) 2009-2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00020 * Alexander Warg <warg@os.inf.tu-dresden.de>
00021 * economic rights: Technische Universität Dresden (Germany)
00022 *
00023 * This file is part of TUD:OS and distributed under the terms of the
00024 * GNU General Public License 2.
00025 * Please see the COPYING-GPL-2 file for details.
00026 *
00027 * As a special exception, you may use this file as part of a free software
00028 * library without restriction. Specifically, if other files instantiate
00029 * templates or use macros or inline functions from this file, or you compile
00030 * this file and link it with other files to produce an executable, this
00031 * file does not by itself cause the resulting executable to be covered by
00032 * the GNU General Public License. This exception does not however
00033 * invalidate any other reasons why the executable file might be covered by
00034 * the GNU General Public License.
00035 */
00036 #pragma once
00037
00038 #include <l4/sys/utcb.h>
00039 #include <l4/sys/types.h>
00040
00057 L4_INLINE l4_msgtag_t
00058 l4_ipc_gate_bind_thread(l4_cap_idx_t gate,
00059 l4_cap_idx_t thread,
00060 l4_umword_t label);
00061
00062 L4_INLINE l4_msgtag_t
00063 l4_ipc_gate_bind_thread_u(l4_cap_idx_t gate, l4_cap_idx_t thread,
00064 l4_umword_t label, l4_utcb_t *utcb);
00065
00075 L4_INLINE l4_msgtag_t
00076 l4_ipc_gate_get_infos(l4_cap_idx_t gate,
00077 l4_umword_t *label);
00078
00082 L4_INLINE l4_msgtag_t
00083 l4_ipc_gate_get_infos_u(l4_cap_idx_t gate, l4_umword_t *label,
00084 l4_utcb_t *utcb);
00085
00091 enum l4_ipc_gate_ops
00092 {
00093 L4_IPC_GATE_BIND_OP = 0x10,
00094 L4_IPC_GATE_GET_INFO_OP = 0x11,
00095 };
00096
00097
00098 /* IMPLEMENTATION -----*/
00099
00100 #include <l4/sys/ipc.h>
00101
00102 L4_INLINE l4_msgtag_t
00103 l4_ipc_gate_bind_thread_u(l4_cap_idx_t gate,
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_IPC_GATE_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_FPAGE_RWX).raw;
00112 return l4_ipc_call(gate, utcb, l4_msgtag(L4_PROTO_KOBJECT, 2, 1, 0),
00113 L4_IPC_NEVER);
00114 }
00115
00116 L4_INLINE l4_msgtag_t
00117 l4_ipc_gate_get_infos_u(l4_cap_idx_t gate, l4_umword_t *label,
00118 l4_utcb_t *utcb)
00119 {
00120 l4_msgtag_t tag;
00121 l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00122 m->mr[0] = L4_IPC_GATE_GET_INFO_OP;
00123 tag = l4_ipc_call(gate, utcb, l4_msgtag(L4_PROTO_KOBJECT, 1, 0, 0),
00124 L4_IPC_NEVER);
00125 if (!l4_msgtag_has_error(tag) && l4_msgtag_label(tag) >= 0)
00126 *label = m->mr[0];
00127 return tag;
00128 }
00129
00130
00131
00132 L4_INLINE l4_msgtag_t
00133 l4_ipc_gate_bind_thread(l4_cap_idx_t gate,
00134 l4_cap_idx_t thread,
00135 l4_umword_t label)
00136 {
00137 return l4_ipc_gate_bind_thread_u(gate, thread, label, l4_utcb());
00138 }
00139
00140 L4_INLINE l4_msgtag_t
00141 l4_ipc_gate_get_infos(l4_cap_idx_t gate,
00142 l4_umword_t *label)
00143 {
00144 return l4_ipc_gate_get_infos_u(gate, label, l4_utcb());
00145 }

```

## 15.313 l4/sys/irq File Reference

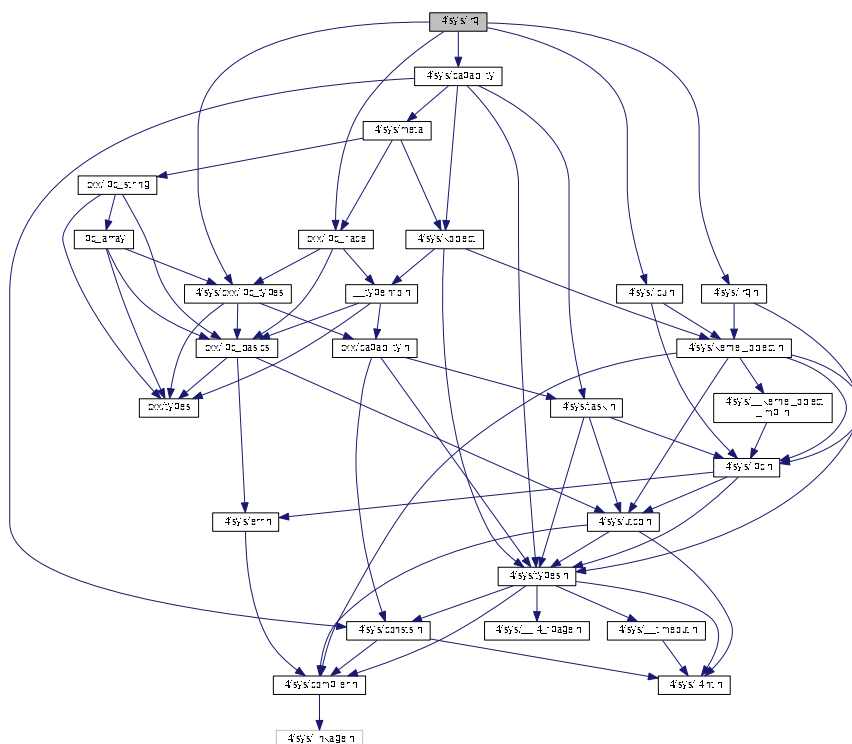
C++ Irq interface.

```

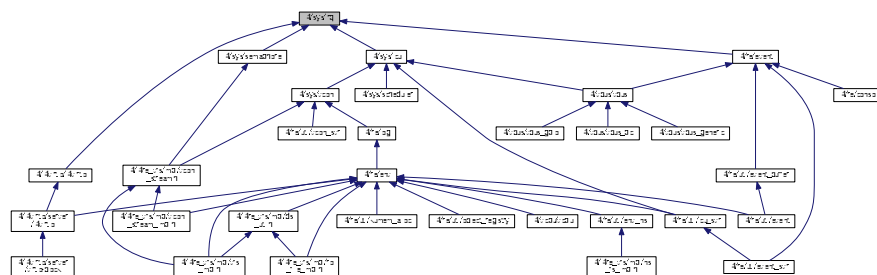
#include <l4/sys/icu.h>
#include <l4/sys/irq.h>
#include <l4/sys/capability>
#include <l4/sys/cxx/ipc_iface>
#include <l4/sys/cxx/ipc_types>

```

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 acknowledge message to an object.*
- struct `L4::Triggerable`  
*Interface that allows an object to be triggered by some source.*
- class `L4::Irq`  
*C++ `Irq` interface.*
- struct `L4::Irq_mux`  
*IRQ multiplexer for shared IRQs.*
- class `L4::Icu`

C++ *lcu* interface.

- class `L4::lcu::Info`

*This class encapsulates information about an ICU.*

## Namespaces

- `L4`

*L4 low-level kernel interface.*

## 15.313.1 Detailed Description

C++ Irq interface.

Definition in file `irq`.

## 15.314 irq

```
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 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024
00025 #pragma once
00026
00027 #include <l4/sys/icu.h>
00028 #include <l4/sys/irq.h>
00029 #include <l4/sys/capability>
00030 #include <l4/sys/cxx/ipc_iface>
00031 #include <l4/sys/cxx/ipc_types>
00032
00033 namespace L4 {
00034
00040 class Irq_eoi : public Kobject_0t<Irq_eoi, L4::PROTO_EMPTY>
00041 {
00042 public:
00062 l4_msgtag_t unmask(unsigned irqnum, l4_umword_t *label = 0,
00063 l4_timeout_t to = L4_IPC_NEVER,
00064 l4_utcb_t *utcb = l4_utcb()) throw()
00065 {
00066 return l4_icu_control_u(cap(), irqnum, L4_ICU_CTL_UNMASK, label, to, utcb);
00067 }
00068 };
00069
00075 struct Triggerable : Kobject_t<Triggerable, Irq_eoi, L4_PROTO_IRQ>
00076 {
00090 l4_msgtag_t trigger(l4_utcb_t *utcb = l4_utcb()) throw()
00091 { return l4_irq_trigger_u(cap(), utcb); }
00092 };
00093
00111 class Irq : public Kobject_t<Irq, Triggerable, L4_PROTO_IRQ_SENDER>
00112 {
00113 public:
00114 using Triggerable::unmask;
00115 }
```

```

00130 l4_msgtag_t attach(l4_umword_t label,
00131 Cap<Thread> const &thread = Cap<Thread>::Invalid,
00132 l4_utcb_t *utcb = l4_utcb()) throw()
00133 { return l4_irq_attach_u(cap(), label, thread.cap(), utcb); }
00134
00142 l4_msgtag_t detach(l4_utcb_t *utcb = l4_utcb()) throw()
00143 { return l4_irq_detach_u(cap(), utcb); }
00144
00145
00157 l4_msgtag_t receive(l4_timeout_t timeout =
L4_IPC_NEVER,
00158 l4_utcb_t *utcb = l4_utcb()) throw()
00159 { return l4_irq_receive_u(cap(), timeout, utcb); }
00160
00170 l4_msgtag_t wait(l4_umword_t *label, l4_timeout_t timeout =
L4_IPC_NEVER,
00171 l4_utcb_t *utcb = l4_utcb()) throw()
00172 { return unmask(-1, label, timeout, utcb); }
00173
00193 l4_msgtag_t unmask(l4_utcb_t *utcb = l4_utcb()) throw()
00194 { return unmask(-1, 0, L4_IPC_NEVER, utcb); }
00195 };
00196
00210 struct Irq_mux : Kobject_t<Irq_mux, Triggerable, L4_PROTO_IRQ_MUX>
00211 {
00225 l4_msgtag_t chain(Cap<Triggerable> const &slave,
00226 l4_utcb_t *utcb = l4_utcb()) throw()
00227 { return l4_irq_mux_chain_u(cap(), slave.cap(), utcb); }
00228 };
00229
00230
00242 class Icu :
00243 public Kobject_t<Icu, Irq_eoi, L4_PROTO_IRQ,
00244 Type_info::Demand_t<1> >
00245 {
00246 public:
00247 enum Mode
00248 {
00249 F_none = L4_IRQ_F_NONE,
00250 F_level_high = L4_IRQ_F_LEVEL_HIGH,
00251 F_level_low = L4_IRQ_F_LEVEL_LOW,
00252 F_pos_edge = L4_IRQ_F_POS_EDGE,
00253 F_neg_edge = L4_IRQ_F_NEG_EDGE,
00254 F_both_edge = L4_IRQ_F_BOTH_EDGE,
00255 F_mask = L4_IRQ_F_MASK,
00256
00257 F_set_wakeup = L4_IRQ_F_SET_WAKEUP,
00258 F_clear_wakeup = L4_IRQ_F_CLEAR_WAKEUP,
00259 };
00260
00261 enum Flags
00262 {
00263 F_msi = L4_ICU_FLAG_MSI
00264 };
00265
00269 class Info : public l4_icu_info_t
00270 {
00271 public:
00272 bool supports_msi() const { return features & F_msi; }
00273 };
00274
00290 l4_msgtag_t bind(unsigned irqnum, L4::Cap<Triggerable> irq,
00291 l4_utcb_t *utcb = l4_utcb()) throw()
00292 { return l4_icu_bind_u(cap(), irqnum, irq.cap(), utcb); }
00293
00294 L4_RPC_NF_OP(L4_ICU_OP_BIND,
00295 l4_msgtag_t, bind, (l4_umword_t irqnum,
Ipc::Cap<Irq> irq));
00296
00306 l4_msgtag_t unbind(unsigned irqnum, L4::Cap<Triggerable> irq,
00307 l4_utcb_t *utcb = l4_utcb()) throw()
00308 { return l4_icu_unbind_u(cap(), irqnum, irq.cap(), utcb); }
00309
00310 L4_RPC_NF_OP(L4_ICU_OP_UNBIND,
00311 l4_msgtag_t, unbind, (l4_umword_t irqnum,
Ipc::Cap<Irq> irq));
00312
00321 l4_msgtag_t info(l4_icu_info_t *info, l4_utcb_t *utcb =
l4_utcb()) throw()
00322 { return l4_icu_info_u(cap(), info, utcb); }
00323
00324 struct _Info { l4_umword_t features, nr_irqs, nr_msis; };
00325 L4_RPC_NF_OP(L4_ICU_OP_INFO, l4_msgtag_t, info, (_Info *info));
00326
00339 L4_INLINE_RPC_OP(L4_ICU_OP_MSI_INFO,
00340 l4_msgtag_t, msi_info, (l4_umword_t irqnum,
l4_uint64_t source,

```

```

00341 l4_icu_msi_info_t *msi_info));
00342
00346 l4_msgtag_t control(unsigned irqnum, unsigned op, l4_umword_t *label,
00347 l4_timeout_t to, l4_utcb_t *utcb =
00348 l4_utcb()) throw()
00349 { return l4_icu_control_u(cap(), irqnum, op, label, to, utcb); }
00349
00364 l4_msgtag_t mask(unsigned irqnum,
00365 l4_umword_t *label = 0,
00366 l4_timeout_t to = L4_IPC_NEVER,
00367 l4_utcb_t *utcb = l4_utcb()) throw()
00368 { return l4_icu_mask_u(cap(), irqnum, label, to, utcb); }
00369
00370 L4_RPC_NF_OP(L4_ICU_OP_MASK, l4_msgtag_t, mask, (
00371 l4_umword_t irqnum),
00372 L4::Ipc::Send_only);
00372
00373
00374 L4_RPC_NF_OP(L4_ICU_OP_UNMASK, l4_msgtag_t,
00375 unmask, (l4_umword_t irqnum),
00376 L4::Ipc::Send_only);
00376
00386 l4_msgtag_t set_mode(unsigned irqnum, l4_umword_t mode,
00387 l4_utcb_t *utcb = l4_utcb()) throw()
00388 { return l4_icu_set_mode_u(cap(), irqnum, mode, utcb); }
00389
00390 L4_RPC_NF_OP(L4_ICU_OP_SET_MODE,
00391 l4_msgtag_t, set_mode, (l4_umword_t irqnum,
00392 l4_umword_t mode));
00392
00393 typedef L4::Typeid::Rpcs_sys<
00394 bind_t, unbind_t, info_t, msi_info_t, unmask_t, mask_t, set_mode_t
00395 > Rpcs;
00396 };
00397
00398 }

```

## 15.315 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>

```

```
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 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
```

```

00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024 #ifndef __L4SYS__KERNEL_OBJECT_H__
00025 #define __L4SYS__KERNEL_OBJECT_H__
00026
00027 #include <l4/sys/types.h>
00028 #include <l4/sys/compiler.h>
00029 #include <l4/sys/utcb.h>
00030
00049 L4_INLINE l4_msgtag_t
00050 l4_invoke_debugger(l4_cap_idx_t obj, l4_msgtag_t tag,
00051 l4_utcb_t *utcb) L4_NOTHROW;
00052
00053 /*****
00054 * Implementation
00055 *****/
00056
00057 #include <l4/sys/__kernel_object_impl.h>
00058 #include <l4/sys/ipc.h>
00059
00060 enum L4_kobject_op {
00061 L4_KOBJECT_OP_DEC_REFCNT = 0,
00062 L4_KOBJECT_OP_REGISTER_IRQ,
00063 };
00064
00065 L4_INLINE l4_msgtag_t
00066 l4_kobject_dec_refcnt_u(l4_cap_idx_t obj, l4_mword_t diff,
00067 l4_utcb_t *u) L4_NOTHROW;
00068
00069 L4_INLINE l4_msgtag_t
00070 l4_kobject_dec_refcnt(l4_cap_idx_t obj, l4_mword_t diff)
00071 L4_NOTHROW;
00072
00073 L4_INLINE l4_msgtag_t
00074 l4_kobject_dec_refcnt_u(l4_cap_idx_t obj, l4_mword_t diff,
00075 l4_utcb_t *u) L4_NOTHROW
00076 {
00077 l4_msg_regs_t *m = l4_utcb_mr_u(u);
00078 m->mr[0] = L4_KOBJECT_OP_DEC_REFCNT;
00079 m->mr[1] = diff;
00080 return l4_ipc_call(obj, u, l4_msgtag(L4_PROTO_KOBJECT, 2, 0, 0),
00081 L4_IPC_NEVER);
00082 }
00083
00084 L4_INLINE l4_msgtag_t
00085 l4_kobject_dec_refcnt(l4_cap_idx_t obj, l4_mword_t diff)
00086 L4_NOTHROW
00087 {
00088 return l4_kobject_dec_refcnt_u(obj, diff, l4_utcb());
00089 }
00090
00091 #endif /* ! __L4SYS__KERNEL_OBJECT_H__ */

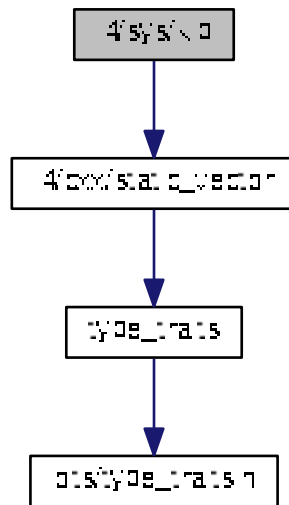
```

## 15.317 l4/sys/kip File Reference

```
#include <l4/cxx/static_vector>
```



Include dependency graph for kip:



## Data Structures

- class [L4::Kip::Mem\\_desc](#)

*Memory descriptors stored in the kernel interface page.*

## Namespaces

- [L4](#)

*[L4](#) low-level kernel interface.*

## 15.317.1 Detailed Description

L4::Kip class, memory descriptors.

### Author

Alexander Warg [alexander.warg@os.inf.tu-dresden.de](mailto:alexander.warg@os.inf.tu-dresden.de)

Definition in file [kip](#).

## 15.318 kip

```

00001 // vim:set ft=cpp:
00010 /*
00011 * (c) 2008-2009 Author(s)
00012 * economic rights: Technische Universität Dresden (Germany)
00013 *
00014 * This file is part of TUD:OS and distributed under the terms of the
00015 * GNU General Public License 2.
00016 * Please see the COPYING-GPL-2 file for details.
00017 *
00018 * As a special exception, you may use this file as part of a free software
00019 * library without restriction. Specifically, if other files instantiate
00020 * templates or use macros or inline functions from this file, or you compile
00021 * this file and link it with other files to produce an executable, this
00022 * file does not by itself cause the resulting executable to be covered by
00023 * the GNU General Public License. This exception does not however
00024 * invalidate any other reasons why the executable file might be covered by
00025 * the GNU General Public License.
00026 */
00027 #ifndef L4_SYS_KIP_H__
00028 #define L4_SYS_KIP_H__
00029
00030 #include <l4/cxx/static_vector>
00031
00032 /* C++ version of memory descriptors */
00033
00043 namespace L4
00044 {
00045 namespace Kip
00046 {
00053 class Mem_desc
00054 {
00055 public:
00059 enum Mem_type
00060 {
00061 Undefined = 0x0,
00062 Conventional = 0x1,
00063 Reserved = 0x2,
00064 Dedicated = 0x3,
00065 Shared = 0x4,
00066
00067 Info = 0xd,
00068 Bootloader = 0xe,
00069 Arch = 0xf
00070 };
00071
00075 enum Info_sub_type
00076 {
00077 Info_acpi_rsdp = 0
00078 };
00079
00080 private:
00081 unsigned long _l, _h;
00082
00083 static unsigned long &memory_info(void *kip) throw()
00084 { return *((unsigned long *)kip + 21); }
00085
00086 static unsigned long memory_info(void const *kip) throw()
00087 { return *((unsigned long const *)kip + 21); }
00088
00089 public:
00097 static Mem_desc *first(void *kip) throw()
00098 {
00099 return (Mem_desc *)((char *)kip
00100 + (memory_info(kip) >> ((sizeof(unsigned long) / 2) * 8)));
00101 }
00102
00103 static Mem_desc const *first(void const *kip) throw()
00104 {
00105 return (Mem_desc const *)((char const *)kip
00106 + (memory_info(kip) >> ((sizeof(unsigned long) / 2) * 8)));
00107 }
00108
00116 static unsigned long count(void const *kip) throw()
00117 {
00118 return memory_info(kip)
00119 & ((1UL << ((sizeof(unsigned long) / 2) * 8)) - 1);
00120 }
00121
00128 static void count(void *kip, unsigned count) throw()
00129 {
00130 unsigned long &mi = memory_info(kip);
00131 mi = (mi & ~((1UL << ((sizeof(unsigned long) / 2) * 8)) - 1)) | count;
00132 }
00133

```

```

00139 static inline cxx::static_vector<Mem_desc const>
00140 all(void const *kip)
00141 {
00142 return cxx::static_vector<Mem_desc const>(
00143 Mem_desc::first(kip),
00144 Mem_desc::count(kip));
00145 }
00146
00147 static inline cxx::static_vector<Mem_desc> all(void *kip)
00148 {
00149 return cxx::static_vector<Mem_desc>(
00150 Mem_desc::first(kip),
00151 Mem_desc::count(kip));
00152 }
00153
00154 Mem_desc(unsigned long start, unsigned long end,
00155 Mem_type t, unsigned char st = 0, bool virt = false) throw()
00156 : _l((start & ~0x3ffUL) | (t & 0x0f) | ((st << 4) & 0x0f0)
00157 | (virt ? 0x0200 : 0x0)), _h(end | 0x3ffUL)
00158 {}
00159
00160 unsigned long start() const throw() { return _l & ~0x3ffUL; }
00161
00162 unsigned long end() const throw() { return _h | 0x3ffUL; }
00163
00164 unsigned long size() const throw() { return end() + 1 - start(); }
00165
00166 Mem_type type() const throw() { return (Mem_type)(_l & 0x0f); }
00167
00168 unsigned char sub_type() const throw() { return (_l >> 4) & 0x0f; }
00169
00170 unsigned is_virtual() const throw() { return _l & 0x200; }
00171
00172 void set(unsigned long start, unsigned long end,
00173 Mem_type t, unsigned char st = 0, bool virt = false) throw()
00174 {
00175 _l = (start & ~0x3ffUL) | (t & 0x0f) | ((st << 4) & 0x0f0)
00176 | (virt?0x0200:0x0);
00177 _h = end | 0x3ffUL;
00178 }
00179 };
00180
00181 #endif

```

## 15.319 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>

```



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.

## 15.319.1 Detailed Description

[L4](#) kernel event tracing.

Definition in file [ktrace.h](#).

## 15.320 ktrace.h

```

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 * This file is part of TUD:OS and distributed under the terms of the
00014 * GNU General Public License 2.
00015 * Please see the COPYING-GPL-2 file for details.
00016 *
00017 * As a special exception, you may use this file as part of a free software
00018 * library without restriction. Specifically, if other files instantiate
00019 * templates or use macros or inline functions from this file, or you compile
00020 * this file and link it with other files to produce an executable, this
00021 * file does not by itself cause the resulting executable to be covered by
00022 * the GNU General Public License. This exception does not however
00023 * invalidate any other reasons why the executable file might be covered by
00024 * the GNU General Public License.
00025 */
00026 /*****
00027 #ifndef __L4_KTRACE_H__
00028 #define __L4_KTRACE_H__
00029
00030 #include <l4/sys/types.h>
00031 #include <l4/sys/ktrace_events.h>
00032
00037 enum
00038 {
00039 LOG_EVENT_CONTEXT_SWITCH = 0,
00040 LOG_EVENT_IPC_SHORTCUT = 1,
00041 LOG_EVENT_IRQ_RAISED = 2,
00042 LOG_EVENT_TIMER_IRQ = 3,
00043 LOG_EVENT_THREAD_EX_REGS = 4,
00044 LOG_EVENT_MAX_EVENTS = 16,
00045 };
00046
00051 // keep in sync with fiasco/src/jabi/jdb_ktrace.cpp
00052 typedef struct
00053 {
00055 l4_tracebuffer_entry_t *tracebuffer;
00057 l4_umword_t size;
00059 volatile l4_uint64_t version;
00060 } l4_tracebuffer_status_window_t;
00061
00066 // keep in sync with fiasco/src/jabi/jdb_ktrace.cpp
00067 typedef struct
00068 {
00069 l4_tracebuffer_status_window_t window[2];
00071 volatile l4_tracebuffer_entry_t * current_entry;
00073 l4_uint32_t logevents[LOG_EVENT_MAX_EVENTS];

```

```

00074
00076 l4_uint32_t scaler_tsc_to_ns;
00078 l4_uint32_t scaler_tsc_to_us;
00080 l4_uint32_t scaler_ns_to_tsc;
00081
00083 volatile l4_uint32_t cnt_context_switch;
00085 volatile l4_uint32_t cnt_addr_space_switch;
00087 volatile l4_uint32_t cnt_shortcut_failed;
00089 volatile l4_uint32_t cnt_shortcut_success;
00091 volatile l4_uint32_t cnt_irq;
00093 volatile l4_uint32_t cnt_ipc_long;
00095 volatile l4_uint32_t cnt_page_fault;
00098 volatile l4_uint32_t cnt_io_fault;
00100 volatile l4_uint32_t cnt_task_create;
00102 volatile l4_uint32_t cnt_schedule;
00106 volatile l4_uint32_t cnt_iobmap_tlb_flush;
00107
00108 } l4_tracebuffer_status_t;
00109
00116 L4_INLINE l4_tracebuffer_status_t *
00117 fiasco_tbuf_get_status(void);
00118
00125 L4_INLINE l4_addr_t
00126 fiasco_tbuf_get_status_phys(void);
00127
00135 L4_INLINE l4_umword_t
00136 fiasco_tbuf_log(const char *text);
00137
00149 L4_INLINE l4_umword_t
00150 fiasco_tbuf_log_3val(const char *text, l4_umword_t v1,
00151 l4_umword_t v2, l4_umword_t v3);
00151
00159 L4_INLINE l4_umword_t
00160 fiasco_tbuf_log_binary(const unsigned char *data);
00161
00166 L4_INLINE void
00167 fiasco_tbuf_clear(void);
00168
00173 L4_INLINE void
00174 fiasco_tbuf_dump(void);
00175
00176 #include <l4/sys/__ktrace-impl.h>
00177
00178 #endif

```

## 15.321 l4/sys/l4int.h File Reference

Fixed sized integer types, generic version.

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.*

### 15.321.1 Detailed Description

Fixed sized integer types, generic version.

Definition in file [l4int.h](#).

## 15.322 l4int.h

```

00001
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 * This file is part of TUD:OS and distributed under the terms of the
00019 * GNU General Public License 2.
00020 * Please see the COPYING-GPL-2 file for details.
00021 *
00022 * As a special exception, you may use this file as part of a free software
00023 * library without restriction. Specifically, if other files instantiate
00024 * templates or use macros or inline functions from this file, or you compile
00025 * this file and link it with other files to produce an executable, this
00026 * file does not by itself cause the resulting executable to be covered by
00027 * the GNU General Public License. This exception does not however
00028 * invalidate any other reasons why the executable file might be covered by
00029 * the GNU General Public License.
00030 */
00031 #ifndef __L4_SYS_L4INT_H__
00032 #define __L4_SYS_L4INT_H__
00033
00034 /* fixed sized data types */
00035 typedef signed char l4_int8_t;
00036 typedef unsigned char l4_uint8_t;
00037 typedef signed short int l4_int16_t;
00038 typedef unsigned short int l4_uint16_t;
00039 typedef signed int l4_int32_t;
00040 typedef unsigned int l4_uint32_t;
00041 typedef signed long long l4_int64_t;
00042 typedef unsigned long long l4_uint64_t;
00043
00044 /* some common data types */
00045 typedef unsigned long l4_addr_t;
00046 //do-we-need-this?//typedef unsigned long l4_offs_t; /**< Address offset type \ingroup
00047 l4_basic_types */
00048
00049 typedef signed long l4_mword_t;
00050 typedef unsigned long l4_umword_t;
00051 typedef l4_uint64_t l4_cpu_time_t;
00052
00053 typedef l4_uint64_t l4_kernel_clock_t;
00054
00055 #endif /* !__L4_SYS_L4INT_H__ */

```

## 15.323 arm/l4/sys/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.*

### 15.323.1 Detailed Description

Fixed sized integer types, arm version.

Definition in file [l4int.h](#).

## 15.324 l4int.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 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024 #pragma once
00025
00026 #include_next <l4/sys/l4int.h>
00027
00032
00033 #define L4_MWORD_BITS 32
00035 typedef unsigned int l4_size_t;
00036 typedef signed int l4_ssize_t;
00038

```

## 15.325 amd64/l4/sys/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.*

### 15.325.1 Detailed Description

Fixed sized integer types, amd64 version.

Definition in file [l4int.h](#).

## 15.326 l4int.h

```

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 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025 #pragma once
00026
00027 #include_next <l4/sys/l4int.h>
00028
00033
00034 #define L4_MWORD_BITS 64
00036 typedef unsigned long l4_size_t;
00037 typedef signed long l4_ssize_t;
00039
```

### 15.327 x86/l4/sys/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.*

### 15.327.1 Detailed Description

Fixed sized integer types, x86 version.

Definition in file [l4int.h](#).

## 15.328 l4int.h

```

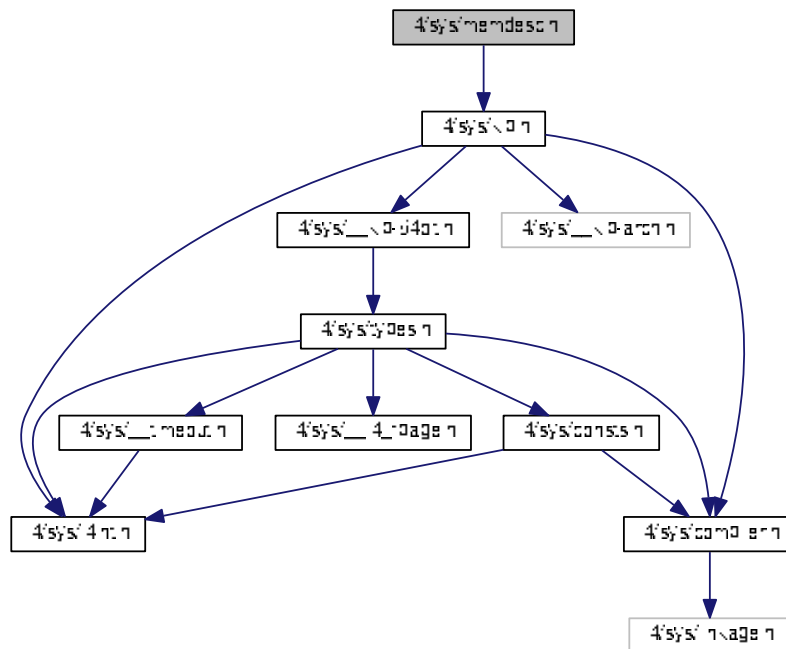
00001
00006 /*
00007 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00008 * Alexander Warg <warg@os.inf.tu-dresden.de>
00009 * economic rights: Technische Universität Dresden (Germany)
00010 *
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024 #pragma once
00025
00026 #include_next <l4/sys/l4int.h>
00027
00032
00033 #define L4_MWORD_BITS 32
00035 typedef unsigned int l4_size_t;
00036 typedef signed int l4_ssize_t;
00038

```

### 15.329 l4/sys/memdesc.h File Reference

Memory description functions.

Include dependency graph for memdesc.h:



- struct `l4_kernel_info_mem_desc_t`  
*Memory descriptor data structure.*

- typedef struct `l4_kernel_info_mem_desc_t` `l4_kernel_info_mem_desc_t`  
*Memory descriptor data structure.*

- 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 }

*Memory sub types for l4\_mem\_type\_info 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.*

### 15.329.1 Detailed Description

Memory description functions.

Definition in file [memdesc.h](#).

## 15.330 memdesc.h

```

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 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024 #ifndef __L4SYS_MEMDESC_H__
00025 #define __L4SYS_MEMDESC_H__
00026
00027 #include <l4/sys/kip.h>
00028
00044 enum l4_mem_type_t
00045 {
00046 l4_mem_type_undefined = 0x0,
00047 l4_mem_type_conventional = 0x1,
00048 l4_mem_type_reserved = 0x2,
00049 l4_mem_type_dedicated = 0x3,
00050 l4_mem_type_shared = 0x4,
00051
00052 l4_mem_type_info = 0xd,
00053 l4_mem_type_bootloader = 0xe,
00054 l4_mem_type_archspecific = 0xf,

```

```

00055 };
00056
00061 enum l4_mem_info_sub_type_t
00062 {
00063 l4_mem_info_acpi_rsdp = 0
00064 };
00065
00066
00074 typedef struct l4_kernel_info_mem_desc_t
00075 {
00077 l4_umword_t l;
00079 l4_umword_t h;
00080 } l4_kernel_info_mem_desc_t;
00081
00082
00087 L4_INLINE
00088 l4_kernel_info_mem_desc_t *
00089 l4_kernel_info_get_mem_descs(l4_kernel_info_t *kip)
00090 L4_NOTHROW;
00091
00097 L4_INLINE
00098 unsigned
00099 l4_kernel_info_get_num_mem_descs(l4_kernel_info_t *kip)
00100 L4_NOTHROW;
00101
00102
00112 L4_INLINE
00113 void
00114 l4_kernel_info_set_mem_desc(l4_kernel_info_mem_desc_t *
00115 md,
00116 l4_addr_t start,
00117 l4_addr_t end,
00118 unsigned type,
00119 unsigned virt,
00120 unsigned sub_type) L4_NOTHROW;
00121
00122
00127 L4_INLINE
00128 l4_umword_t
00129 l4_kernel_info_get_mem_desc_start(
00130 l4_kernel_info_mem_desc_t *md) L4_NOTHROW;
00131
00132
00137 L4_INLINE
00138 l4_umword_t
00139 l4_kernel_info_get_mem_desc_end(
00140 l4_kernel_info_mem_desc_t *md) L4_NOTHROW;
00141
00142
00147 L4_INLINE
00148 l4_umword_t
00149 l4_kernel_info_get_mem_desc_type(
00150 l4_kernel_info_mem_desc_t *md) L4_NOTHROW;
00151
00152
00160 L4_INLINE
00161 l4_umword_t
00162 l4_kernel_info_get_mem_desc_subtype(
00163 l4_kernel_info_mem_desc_t *md) L4_NOTHROW;
00164
00165
00170 L4_INLINE
00171 l4_umword_t
00172 l4_kernel_info_get_mem_desc_is_virtual(
00173 l4_kernel_info_mem_desc_t *md) L4_NOTHROW;
00174
00175
00176 /*
00177 * Implementations
00178 */
00179
00178 L4_INLINE
00179 l4_kernel_info_mem_desc_t *
00180 l4_kernel_info_get_mem_descs(l4_kernel_info_t *kip)
00181 L4_NOTHROW
00182 {
00183 return (l4_kernel_info_mem_desc_t *)(((l4_addr_t)kip)
00184 + (kip->mem_info >> (sizeof(l4_umword_t) * 4)));
00185 }
00186
00187 L4_INLINE
00188 unsigned
00189 l4_kernel_info_get_num_mem_descs(l4_kernel_info_t *kip)
00190 L4_NOTHROW
00191 {
00192 return kip->mem_info & ((1UL << (sizeof(l4_umword_t)*4)) -1);
00193 }
00194
00195 L4_INLINE
00196 void
00197 l4_kernel_info_set_mem_desc(l4_kernel_info_mem_desc_t *
00198 md,
00199 l4_addr_t start,
00200 l4_addr_t end,

```

```

00198 unsigned type,
00199 unsigned virt,
00200 unsigned sub_type) L4_NOTHROW
00201 {
00202 md->l = (start & ~0x3ffUL) | (type & 0x0f) | ((sub_type << 4) & 0x0f0)
00203 | (virt ? 0x200 : 0x0);
00204 md->h = end;
00205 }
00206
00207
00208 L4_INLINE
00209 l4_umword_t
00210 l4_kernel_info_get_mem_desc_start(
00211 l4_kernel_info_mem_desc_t *md) L4_NOTHROW
00212 {
00213 return md->l & ~0x3ffUL;
00214 }
00215 L4_INLINE
00216 l4_umword_t
00217 l4_kernel_info_get_mem_desc_end(
00218 l4_kernel_info_mem_desc_t *md) L4_NOTHROW
00219 {
00220 return md->h | 0x3ffUL;
00221 }
00222 L4_INLINE
00223 l4_umword_t
00224 l4_kernel_info_get_mem_desc_type(
00225 l4_kernel_info_mem_desc_t *md) L4_NOTHROW
00226 {
00227 return md->l & 0xf;
00228 }
00229 L4_INLINE
00230 l4_umword_t
00231 l4_kernel_info_get_mem_desc_subtype(
00232 l4_kernel_info_mem_desc_t *md) L4_NOTHROW
00233 {
00234 return (md->l & 0xf0) >> 4;
00235 }
00236 L4_INLINE
00237 l4_umword_t
00238 l4_kernel_info_get_mem_desc_is_virtual(
00239 l4_kernel_info_mem_desc_t *md) L4_NOTHROW
00240 {
00241 return md->l & 0x200;
00242 }
00243 #endif /* ! __L4SYS__MEMDESC_H__ */

```

## 15.331 l4/sys/meta File Reference

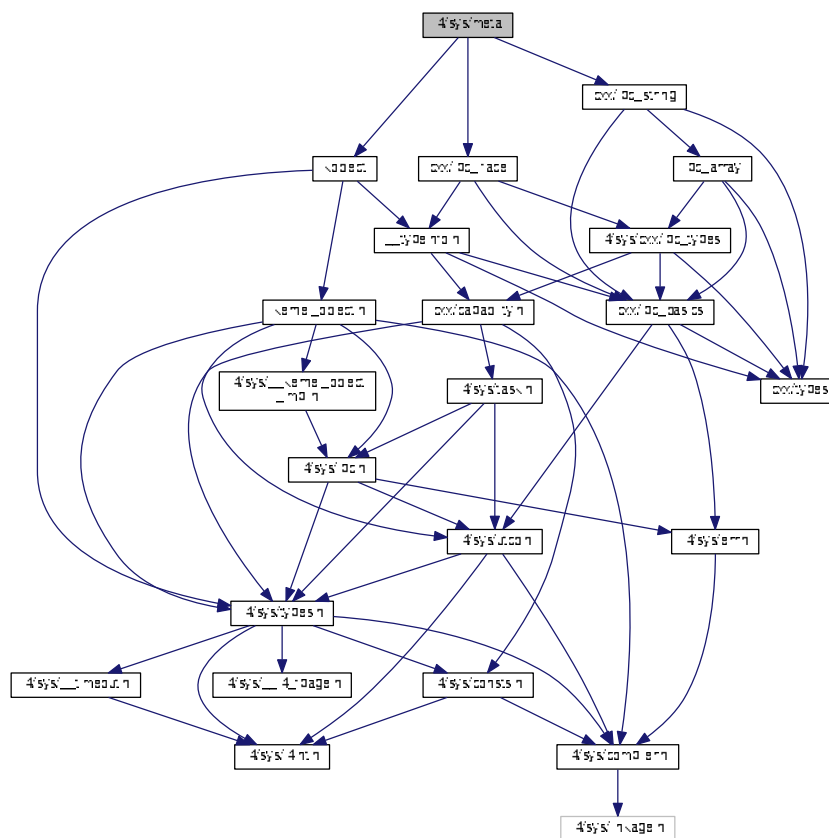
Meta interface for getting dynamic type information about objects behind capabilities.

```

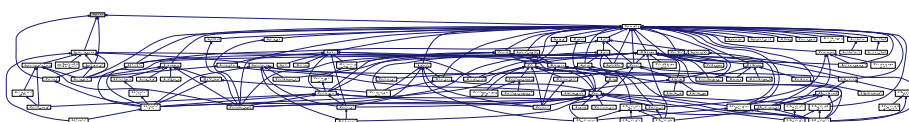
#include "kobject"
#include "cxx/ipc_iface"
#include "cxx/ipc_string"

```

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

- L4

*L4* low-level kernel interface.

### 15.331.1 Detailed Description

Meta interface for getting dynamic type information about objects behind capabilities.

Definition in file [meta](#).

## 15.332 meta

```

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 * This file is part of TUD:OS and distributed under the terms of the
00007 * GNU General Public License 2.
00008 * Please see the COPYING-GPL-2 file for details.
00009 *
00010 * As a special exception, you may use this file as part of a free software
00011 * library without restriction. Specifically, if other files instantiate
00012 * templates or use macros or inline functions from this file, or you compile
00013 * this file and link it with other files to produce an executable, this
00014 * file does not by itself cause the resulting executable to be covered by
00015 * the GNU General Public License. This exception does not however
00016 * invalidate any other reasons why the executable file might be covered by
00017 * the GNU General Public License.
00018 */
00019 #pragma once
00020
00021 #include "kobject"
00022 #include "cxx/ipc_iface"
00023 #include "cxx/ipc_string"
00024
00025 namespace L4 {
00026
00027 class Meta : public Kobject_t<Meta, Kobject, L4_PROTO_META>
00028 {
00029 public:
00030 L4_INLINE_RPC(l4_msgtag_t, num_interfaces, ());
00031
00032 L4_INLINE_RPC(l4_msgtag_t, interface, (
00033 l4_umword_t idx, long *proto,
00034 L4::Ipc::String<char> *name));
00035
00036 L4_INLINE_RPC(l4_msgtag_t, supports, (
00037 l4_mword_t protocol));
00038
00039 typedef L4::Typeid::Rpc<num_interfaces_t, interface_t, supports_t>
00040 Rpc;
00041 };
00042
00043 }
```

## 15.333 l4/sys/pager File Reference

Pager and lo\_pager C++ interface.

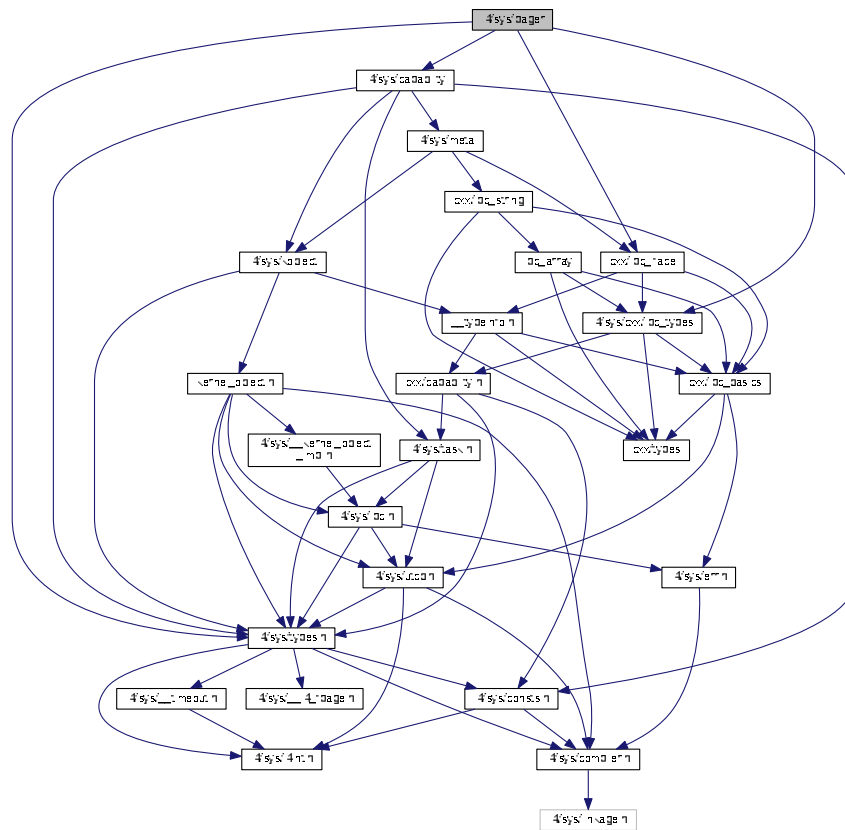
```

#include <l4/sys/capability>
#include <l4/sys/types.h>
#include <l4/sys/cxx/ipc_types>
```

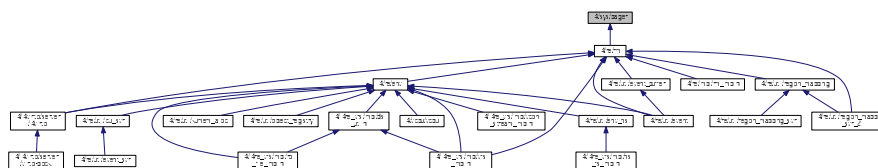


```
#include <l4/sys/cxx/ipc_iface>
```

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

- [L4](#)  
*L4* low-level kernel interface.

### 15.333.1 Detailed Description

Pager and `Io_pager` C++ interface.

Definition in file [pager](#).

## 15.334 pager

```

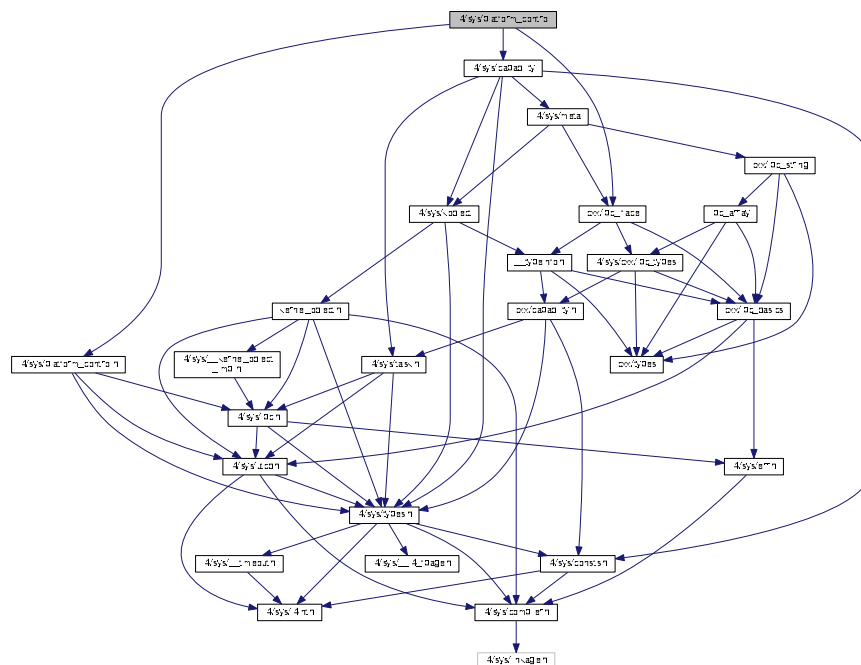
00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007 * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00008 *
00009 * This file is part of TUD:OS and distributed under the terms of the
00010 * GNU General Public License 2.
00011 * Please see the COPYING-GPL-2 file for details.
00012 *
00013 * As a special exception, you may use this file as part of a free software
00014 * library without restriction. Specifically, if other files instantiate
00015 * templates or use macros or inline functions from this file, or you compile
00016 * this file and link it with other files to produce an executable, this
00017 * file does not by itself cause the resulting executable to be covered by
00018 * the GNU General Public License. This exception does not however
00019 * invalidate any other reasons why the executable file might be covered by
00020 * the GNU General Public License.
00021 */
00022
00023 #pragma once
00024
00025 #include <l4/sys/capability>
00026 #include <l4/sys/types.h>
00027 #include <l4/sys/cxx/ipc_types>
00028 #include <l4/sys/cxx/ipc_iface>
00029
00030 namespace L4 {
00031
00035 class L4_EXPORT Io_pager :
00036 public Kobject_0t<Io_pager, L4_PROTO_IO_PAGE_FAULT>
00037 {
00038 public:
00055 L4_INLINE_RPC(
00056 l4_msgtag_t, io_page_fault, (l4_fpage_t io_pfa,
00057 l4_umword_t pc,
00058 L4::Ipc::Opt<l4_mword_t &> result,
00059 L4::Ipc::Rcv_fpage rwin,
00060 L4::Ipc::Opt<L4::Ipc::Snd_fpage &> fp)
00061);
00062
00063 typedef L4::Typeid::Rpc_nocode<io_page_fault_t>
00064 Rpccs;
00065 };
00066
00067 class L4_EXPORT Pager :
00068 public Kobject_t<Pager, Io_pager, L4_PROTO_PAGE_FAULT>
00069 {
00070 public:
00103 L4_INLINE_RPC(
00104 l4_msgtag_t, page_fault, (l4_umword_t pfa,
00105 l4_umword_t pc,
00106 L4::Ipc::Opt<l4_mword_t &> result,
00107 L4::Ipc::Rcv_fpage rwin,
00108 L4::Ipc::Opt<L4::Ipc::Snd_fpage &> fp));
00109
00110 typedef L4::Typeid::Rpc_nocode<page_fault_t>
00111 Rpccs;
00112 };

```

## 15.335 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.*

## Namespaces

- **L4**  
*L4 low-level kernel interface.*

### 15.335.1 Detailed Description

Platform control object.

Definition in file [platform\\_control](#).

## 15.336 platform\_control

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007 * (c) 2014 Steffen Liebergeld <steffen.liebergeld@kernkonzept.com>
00008 * 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 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025
00026 #pragma once
00027
00028 #include <l4/sys/capability>
00029 #include <l4/sys/platform_control.h>
00030 #include <l4/sys/cxx/ipc_iface>
00031
00032 namespace L4 {
00033
00046 class L4_EXPORT Platform_control
00047 : public Kobject_t<Platform_control, Kobject, L4_PROTO_PLATFORM_CTL>
00048 {
00049 public:
00051 enum Opcode
00052 {
00053 Suspend = L4_PLATFORM_CTL_SYS_SUSPEND_OP,
00054 Shutdown = L4_PLATFORM_CTL_SYS_SHUTDOWN_OP,
00055 Cpu_enable = L4_PLATFORM_CTL_CPU_ENABLE_OP,
00056 Cpu_disable = L4_PLATFORM_CTL_CPU_DISABLE_OP
00057 };
00058
00065 L4_INLINE_RPC_OP(L4_PLATFORM_CTL_SYS_SUSPEND_OP,
00066 l4_msgtag_t, system_suspend, (l4_umword_t extras));
00067
00073 L4_INLINE_RPC_OP(L4_PLATFORM_CTL_SYS_SHUTDOWN_OP,
00074 l4_msgtag_t, system_shutdown, (l4_umword_t reboot));
00075
00083 L4_INLINE_RPC_OP(L4_PLATFORM_CTL_CPU_ENABLE_OP,
00084 l4_msgtag_t, cpu_enable, (l4_umword_t phys_id));
00085
00093 L4_INLINE_RPC_OP(L4_PLATFORM_CTL_CPU_DISABLE_OP,
00094 l4_msgtag_t, cpu_disable, (l4_umword_t phys_id));
00095
00096 typedef L4::Typeid::Rpcsys<system_suspend_t, system_shutdown_t,
00097 cpu_enable_t, cpu_disable_t> Rpcps;
00098 };
00099
00100 }
00101

```

## 15.337 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>

```



- [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\\_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.*

### 15.337.1 Detailed Description

Platform control object.

Definition in file [platform\\_control.h](#).

## 15.338 platform\_control.h

```

00001
00005 /*
00006 * (c) 2014 Alexander Warg <alexander.warg@kernkonzept.com>
00007 *
00008 * This file is part of TUD:OS and distributed under the terms of the
00009 * GNU General Public License 2.
00010 * Please see the COPYING-GPL-2 file for details.
00011 *
00012 * As a special exception, you may use this file as part of a free software
00013 * library without restriction. Specifically, if other files instantiate
00014 * templates or use macros or inline functions from this file, or you compile
00015 * this file and link it with other files to produce an executable, this
00016 * file does not by itself cause the resulting executable to be covered by
00017 * the GNU General Public License. This exception does not however
00018 * invalidate any other reasons why the executable file might be covered by
00019 * the GNU General Public License.
00020 */
00021
00022 #pragma once
00023
00024 #include <l4/sys/types.h>
00025 #include <l4/sys/utcb.h>
00026
00051 L4_INLINE l4_msgtag_t
00052 l4_platform_ctl_system_suspend(l4_cap_idx_t pfc,
00053 l4_umword_t extras) L4_NOTHROW;
00054
00058 L4_INLINE l4_msgtag_t
00059 l4_platform_ctl_system_suspend_u(l4_cap_idx_t pfc,
00060 l4_umword_t extras,
00061 l4_utcb_t *utcb) L4_NOTHROW;
00062
00063
00072 L4_INLINE l4_msgtag_t
00073 l4_platform_ctl_system_shutdown(l4_cap_idx_t pfc,
00074 l4_umword_t reboot) L4_NOTHROW;
00075
00079 L4_INLINE l4_msgtag_t
00080 l4_platform_ctl_system_shutdown_u(l4_cap_idx_t pfc,
00081 l4_umword_t reboot,
00082 l4_utcb_t *utcb) L4_NOTHROW;
00083
00092 L4_INLINE l4_msgtag_t
00093 l4_platform_ctl_cpu_enable(l4_cap_idx_t pfc,
00094 l4_umword_t phys_id) L4_NOTHROW;
00095
00099 L4_INLINE l4_msgtag_t
00100 l4_platform_ctl_cpu_enable_u(l4_cap_idx_t pfc,
00101 l4_umword_t phys_id,
00102 l4_utcb_t *utcb) L4_NOTHROW;
00103
00112 L4_INLINE l4_msgtag_t
00113 l4_platform_ctl_cpu_disable(l4_cap_idx_t pfc,
00114 l4_umword_t phys_id) L4_NOTHROW;
00115
00119 L4_INLINE l4_msgtag_t
00120 l4_platform_ctl_cpu_disable_u(l4_cap_idx_t pfc,

```

```

00121 l4_umword_t phys_id,
00122 l4_utcb_t *utcb) L4_NOTHROW;
00123 /* ends l4_platform_control_api group */
00125
00126
00135 enum L4_platform_ctl_ops
00136 {
00137 L4_PLATFORM_CTL_SYS_SUSPEND_OP = 0UL,
00138 L4_PLATFORM_CTL_SYS_SHUTDOWN_OP = 1UL,
00139 L4_PLATFORM_CTL_CPU_ENABLE_OP = 3UL,
00140 L4_PLATFORM_CTL_CPU_DISABLE_OP = 4UL,
00141 };
00142
00147 enum L4_platform_ctl_proto
00148 {
00154 L4_PROTO_PLATFORM_CTL = 0
00155 };
00156
00157 /* IMPLEMENTATION -----*/
00158
00159 #include <l4/sys/ipc.h>
00160
00161 L4_INLINE l4_msgtag_t
00162 l4_platform_ctl_system_suspend_u(l4_cap_idx_t pfc,
00163 l4_umword_t extras,
00164 l4_utcb_t *utcb) L4_NOTHROW
00165 {
00166 l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00167 v->mr[0] = L4_PLATFORM_CTL_SYS_SUSPEND_OP;
00168 v->mr[1] = extras;
00169 return l4_ipc_call(pfc, utcb, l4_msgtag(
00170 L4_PROTO_PLATFORM_CTL, 2, 0, 0),
00171 L4_IPC_NEVER);
00172 }
00173
00174 L4_INLINE l4_msgtag_t
00175 l4_platform_ctl_system_shutdown_u(l4_cap_idx_t pfc,
00176 l4_umword_t reboot,
00177 l4_utcb_t *utcb) L4_NOTHROW
00178 {
00179 l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00180 v->mr[0] = L4_PLATFORM_CTL_SYS_SHUTDOWN_OP;
00181 v->mr[1] = reboot;
00182 return l4_ipc_call(pfc, utcb, l4_msgtag(
00183 L4_PROTO_PLATFORM_CTL, 2, 0, 0),
00184 L4_IPC_NEVER);
00185 }
00186
00187 L4_INLINE l4_msgtag_t
00188 l4_platform_ctl_system_suspend(l4_cap_idx_t pfc,
00189 l4_umword_t extras) L4_NOTHROW
00190 {
00191 return l4_platform_ctl_system_suspend_u(pfc, extras, l4_utcb());
00192 }
00193
00194 L4_INLINE l4_msgtag_t
00195 l4_platform_ctl_system_shutdown(l4_cap_idx_t pfc,
00196 l4_umword_t reboot) L4_NOTHROW
00197 {
00198 return l4_platform_ctl_system_shutdown_u(pfc, reboot, l4_utcb());
00199 }
00200
00201 L4_INLINE l4_msgtag_t
00202 l4_platform_ctl_cpu_enable_u(l4_cap_idx_t pfc,
00203 l4_umword_t phys_id,
00204 l4_utcb_t *utcb) L4_NOTHROW
00205 {
00206 l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00207 v->mr[0] = L4_PLATFORM_CTL_CPU_ENABLE_OP;
00208 v->mr[1] = phys_id;
00209 return l4_ipc_call(pfc, utcb, l4_msgtag(
00210 L4_PROTO_PLATFORM_CTL, 2, 0, 0),
00211 L4_IPC_NEVER);
00212 }
00213
00214 L4_INLINE l4_msgtag_t
00215 l4_platform_ctl_cpu_disable_u(l4_cap_idx_t pfc,
00216 l4_umword_t phys_id,
00217 l4_utcb_t *utcb) L4_NOTHROW
00218 {
00219 l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00220 v->mr[0] = L4_PLATFORM_CTL_CPU_DISABLE_OP;
00221 v->mr[1] = phys_id;
00222 return l4_ipc_call(pfc, utcb, l4_msgtag(
00223 L4_PROTO_PLATFORM_CTL, 2, 0, 0),
00224 L4_IPC_NEVER);

```

```

00222 }
00223
00224 L4_INLINE l4_msgtag_t
00225 l4_platform_ctl_cpu_enable(l4_cap_idx_t pfc,
00226 l4_umword_t phys_id) L4_NOTHROW
00227 {
00228 return l4_platform_ctl_cpu_enable_u(pfc, phys_id, l4_utcb());
00229 }
00230
00231 L4_INLINE l4_msgtag_t
00232 l4_platform_ctl_cpu_disable(l4_cap_idx_t pfc,
00233 l4_umword_t phys_id) L4_NOTHROW
00234 {
00235 return l4_platform_ctl_cpu_disable_u(pfc, phys_id, l4_utcb());
00236 }

```

## 15.339 l4/sys/scheduler File Reference

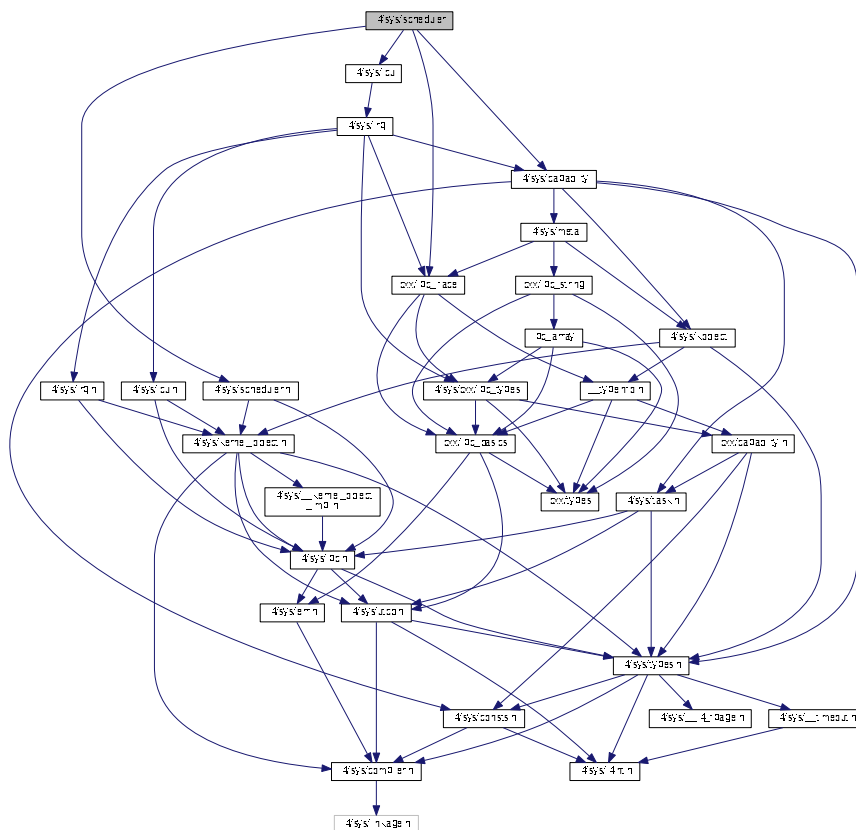
Scheduler object functions.

```

#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.*



## Namespaces

- [L4](#)

[L4](#) *low-level kernel interface.*

### 15.339.1 Detailed Description

Scheduler object functions.

Definition in file [scheduler](#).

## 15.340 scheduler

```

00001 // vi:set ft=c++: -- 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 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024 #pragma once
00025
00026 #include <l4/sys/icu>
00027 #include <l4/sys/scheduler.h>
00028 #include <l4/sys/capability>
00029 #include <l4/sys/cxx/ipc_iface>
00030
00031 namespace L4 {
00032
00042 class L4_EXPORT Scheduler :
00043 public Kobject_t<Scheduler, Icu, L4_PROTO_SCHEDULER,
00044 Type_info::Demand_t<1> >
00045 {
00046 public:
00047 // ABI function for 'info' call
00048 L4_INLINE_RPC_NF_OP(L4_SCHEDULER_INFO_OP,
00049 l4_msgtag_t, info, (l4_umword_t gran_offset,
00050 l4_umword_t *map,
00051 l4_umword_t *cpu_max));
00052
00066 l4_msgtag_t info(l4_umword_t *cpu_max,
00067 l4_sched_cpu_set_t *cpus,
00068 l4_utcb_t *utcb = l4_utcb()) const throw()
00069 {
00070 l4_umword_t max = 0;
00071 l4_msgtag_t t =
00072 info_t::call(c(), cpus->gran_offset, &cpus->map, &max, utcb);
00073 if (cpu_max) *cpu_max = max;
00074 return t;
00075 }
00076
00097 L4_INLINE_RPC_OP(L4_SCHEDULER_RUN_THREAD_OP,
00098 l4_msgtag_t, run_thread, (Ipc::Cap<Thread> thread,
00099 l4_sched_param_t const &sp));
00100
00126 L4_INLINE_RPC_OP(L4_SCHEDULER_IDLE_TIME_OP,
00127 l4_msgtag_t, idle_time, (l4_sched_cpu_set_t const &cpus,
00128 l4_kernel_clock_t *us));
00129
00139 bool is_online(l4_umword_t cpu, l4_utcb_t *utcb =
00140 l4_utcb()) const throw()
00141 { return l4_scheduler_is_online_u(cap(), cpu, utcb); }
00142
00142 typedef L4::Typeid::Rpcs_sys<info_t, run_thread_t, idle_time_t>
00143 Rpcs;
00144 };

```



## 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_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_sched_param_t` `l4_sched_param` (unsigned prio, `l4_cpu_time_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  $\mu$ 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.*

### 15.341.1 Detailed Description

Scheduler object functions.

Definition in file `scheduler.h`.

## 15.342 scheduler.h

```

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 * 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 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by

```

```

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 #pragma once
00024
00025 #include <l4/sys/kernel_object.h>
00026 #include <l4/sys/ipc.h>
00027
00044 typedef struct l4_sched_cpu_set_t
00045 {
00057 l4_umword_t gran_offset;
00058
00062 l4_umword_t map;
00063
00064 #ifdef __cplusplus
00065 unsigned char granularity() const { return gran_offset >> 24; }
00068 unsigned offset() const { return gran_offset & 0x00ffffff; }
00070 void set(unsigned char granularity, unsigned offset)
00071 { gran_offset = ((l4_umword_t)granularity << 24) | (offset & 0x00ffffff); }
00072 #endif
00073 } l4_sched_cpu_set_t;
00074
00085 L4_INLINE l4_sched_cpu_set_t
00086 l4_sched_cpu_set(l4_umword_t offset, unsigned char
granularity,
00087 l4_umword_t map L4_DEFAULT_PARAM(1)) L4_NOTHROW;
00088
00103 L4_INLINE l4_msgtag_t
00104 l4_scheduler_info(l4_cap_idx_t scheduler,
l4_umword_t *cpu_max,
00105 l4_sched_cpu_set_t *cpus) L4_NOTHROW;
00106
00110 L4_INLINE l4_msgtag_t
00111 l4_scheduler_info_u(l4_cap_idx_t scheduler, l4_umword_t *cpu_max,
00112 l4_sched_cpu_set_t *cpus, l4_utcb_t *utcb)
L4_NOTHROW;
00113
00114
00119 typedef struct l4_sched_param_t
00120 {
00121 l4_sched_cpu_set_t affinity;
00122 l4_umword_t prio;
00123 l4_umword_t quantum;
00124 } l4_sched_param_t;
00125
00130 L4_INLINE l4_sched_param_t
00131 l4_sched_param(unsigned prio,
l4_cpu_time_t quantum L4_DEFAULT_PARAM(0))
00132 L4_NOTHROW;
00133
00141 L4_INLINE l4_msgtag_t
00142 l4_scheduler_run_thread(l4_cap_idx_t scheduler,
l4_cap_idx_t thread, l4_sched_param_t const *sp)
00143 L4_NOTHROW;
00144
00148 L4_INLINE l4_msgtag_t
00149 l4_scheduler_run_thread_u(l4_cap_idx_t scheduler, l4_cap_idx_t thread,
l4_sched_param_t const *sp, l4_utcb_t *utcb) L4_NOTHROW;
00150
00151
00159 L4_INLINE l4_msgtag_t
00160 l4_scheduler_idle_time(l4_cap_idx_t scheduler,
l4_sched_cpu_set_t const *cpus,
00161 l4_kernel_clock_t *us) L4_NOTHROW;
00162
00166 L4_INLINE l4_msgtag_t
00167 l4_scheduler_idle_time_u(l4_cap_idx_t scheduler, l4_sched_cpu_set_t const *
cpus,
00168 l4_kernel_clock_t *us, l4_utcb_t *utcb) L4_NOTHROW;
00169
00170
00171
00182 L4_INLINE int
00183 l4_scheduler_is_online(l4_cap_idx_t scheduler,
l4_umword_t cpu) L4_NOTHROW;
00184
00188 L4_INLINE int
00189 l4_scheduler_is_online_u(l4_cap_idx_t scheduler, l4_umword_t cpu,
l4_utcb_t *utcb) L4_NOTHROW;
00190
00191
00192
00193
00200 enum l4_scheduler_ops
00201 {
00202 L4_SCHEDULER_INFO_OP = 0UL,
00203 L4_SCHEDULER_RUN_THREAD_OP = 1UL,
00204 L4_SCHEDULER_IDLE_TIME_OP = 2UL,

```

```

00205 };
00206
00207 /***** Implementations *****/
00208
00209 L4_INLINE l4_sched_cpu_set_t
00210 l4_sched_cpu_set(l4_umword_t offset, unsigned char
granularity,
00211 l4_umword_t map) L4_NOTHROW
00212 {
00213 l4_sched_cpu_set_t cs;
00214 cs.gran_offset = ((l4_umword_t)granularity << 24) |
offset;
00215 cs.map = map;
00216 return cs;
00217 }
00218
00219 L4_INLINE l4_sched_param_t
00220 l4_sched_param(unsigned prio, l4_cpu_time_t quantum)
L4_NOTHROW
00221 {
00222 l4_sched_param_t sp;
00223 sp.prio = prio;
00224 sp.quantum = quantum;
00225 sp.affinity = l4_sched_cpu_set(0, ~0, 1);
00226 return sp;
00227 }
00228
00229
00230 L4_INLINE l4_msgtag_t
00231 l4_scheduler_info_u(l4_cap_idx_t scheduler, l4_umword_t *cpu_max,
00232 l4_sched_cpu_set_t *cpus, l4_utcb_t *utcb)
L4_NOTHROW
00233 {
00234 l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00235 l4_msgtag_t res;
00236
00237 m->mr[0] = L4_SCHEDULER_INFO_OP;
00238 m->mr[1] = cpus->gran_offset;
00239
00240 res = l4_ipc_call(scheduler, utcb, l4_msgtag(
L4_PROTO_SCHEDULER, 2, 0, 0), L4_IPC_NEVER);
00241
00242 if (l4_msgtag_has_error(res))
00243 return res;
00244
00245 cpus->map = m->mr[0];
00246
00247 if (cpu_max)
00248 *cpu_max = m->mr[1];
00249
00250 return res;
00251 }
00252
00253 L4_INLINE l4_msgtag_t
00254 l4_scheduler_run_thread_u(l4_cap_idx_t scheduler, l4_cap_idx_t thread,
00255 l4_sched_param_t const *sp, l4_utcb_t *utcb)
L4_NOTHROW
00256 {
00257 l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00258 m->mr[0] = L4_SCHEDULER_RUN_THREAD_OP;
00259 m->mr[1] = sp->affinity.gran_offset;
00260 m->mr[2] = sp->affinity.map;
00261 m->mr[3] = sp->prio;
00262 m->mr[4] = sp->quantum;
00263 m->mr[5] = l4_map_obj_control(0, 0);
00264 m->mr[6] = l4_obj_fpage(thread, 0, L4_FPAGE_RWX).raw;
00265
00266 return l4_ipc_call(scheduler, utcb, l4_msgtag(
L4_PROTO_SCHEDULER, 5, 1, 0), L4_IPC_NEVER);
00267 }
00268
00269 L4_INLINE l4_msgtag_t
00270 l4_scheduler_idle_time_u(l4_cap_idx_t scheduler, l4_sched_cpu_set_t const *
cpus,
00271 l4_kernel_clock_t *us, l4_utcb_t *utcb)
L4_NOTHROW
00272 {
00273 l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00274 l4_msgtag_t res;
00275
00276 v->mr[0] = L4_SCHEDULER_IDLE_TIME_OP;
00277 v->mr[1] = cpus->gran_offset;
00278 v->mr[2] = cpus->map;
00279
00280 res = l4_ipc_call(scheduler, utcb,
00281 l4_msgtag(L4_PROTO_SCHEDULER, 3, 0, 0),
L4_IPC_NEVER);

```

```

00282
00283 if (l4_msgtag_has_error(res))
00284 return res;
00285
00286 *us = v->mr64[l4_utcb_mr64_idx(0)];
00287
00288 return res;
00289 }
00290
00291
00292 L4_INLINE int
00293 l4_scheduler_is_online_u(l4_cap_idx_t scheduler, l4_umword_t cpu,
00294 l4_utcb_t *utcb) L4_NOTHROW
00295 {
00296 l4_sched_cpu_set_t s;
00297 l4_msgtag_t r;
00298 s.gran_offset = cpu;
00299 r = l4_scheduler_info_u(scheduler, NULL, &s, utcb);
00300 if (l4_msgtag_has_error(r) || l4_msgtag_label(r) < 0)
00301 return 0;
00302
00303 return s.map & 1;
00304 }
00305
00306
00307 L4_INLINE l4_msgtag_t
00308 l4_scheduler_info(l4_cap_idx_t scheduler,
00309 l4_umword_t *cpu_max,
00310 l4_sched_cpu_set_t *cpus) L4_NOTHROW
00311 {
00312 return l4_scheduler_info_u(scheduler, cpu_max, cpus, l4_utcb());
00313 }
00314
00315 L4_INLINE l4_msgtag_t
00316 l4_scheduler_run_thread(l4_cap_idx_t scheduler,
00317 l4_cap_idx_t thread, l4_sched_param_t const *sp)
00318 L4_NOTHROW
00319 {
00320 return l4_scheduler_run_thread_u(scheduler, thread, sp, l4_utcb());
00321 }
00322
00323 L4_INLINE l4_msgtag_t
00324 l4_scheduler_idle_time(l4_cap_idx_t scheduler,
00325 l4_sched_cpu_set_t const *cpus,
00326 l4_kernel_clock_t *us) L4_NOTHROW
00327 {
00328 return l4_scheduler_idle_time_u(scheduler, cpus, us, l4_utcb());
00329 }
00330
00331 L4_INLINE int
00332 l4_scheduler_is_online(l4_cap_idx_t scheduler,
00333 l4_umword_t cpu) L4_NOTHROW
00334 {
00335 return l4_scheduler_is_online_u(scheduler, cpu, l4_utcb());
00336 }

```

## 15.343 l4/sys/semaphore File Reference

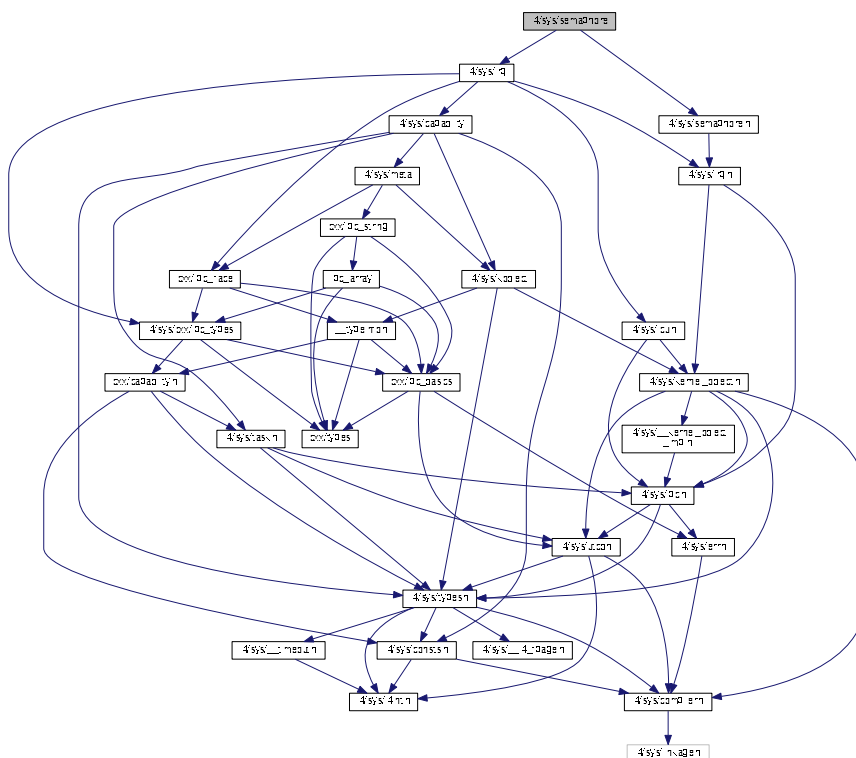
Semaphore class definition.

```

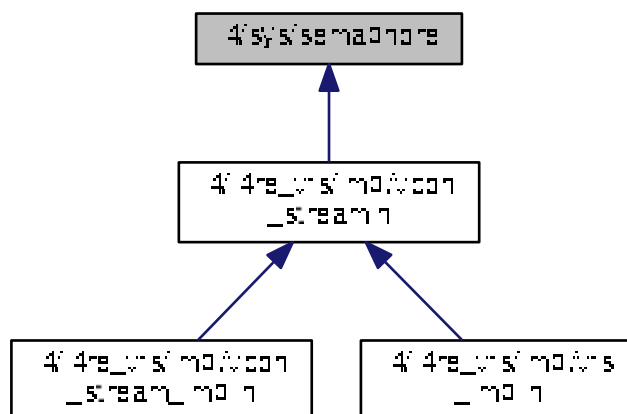
#include <l4/sys/irq>
#include <l4/sys/semaphore.h>

```

Include dependency graph for semaphore:



This graph shows which files directly or indirectly include this file:



## Data Structures

- struct L4::Semaphore  
*Kernel-provided semaphore object.*

## Namespaces

- [L4](#)

[L4](#) low-level kernel interface.

### 15.343.1 Detailed Description

Semaphore class definition.

Definition in file [semaphore](#).

## 15.344 semaphore

```

00001 // vi:set ft=c++: -- Mode: C++ --
00006 /*
00007 * (c) 2015 Alexander Warg <alexander.warg@kernkonzept.com>
00008 *
00009 * This file is part of TUD:OS and distributed under the terms of the
00010 * GNU General Public License 2.
00011 * Please see the COPYING-GPL-2 file for details.
00012 *
00013 * As a special exception, you may use this file as part of a free software
00014 * library without restriction. Specifically, if other files instantiate
00015 * templates or use macros or inline functions from this file, or you compile
00016 * this file and link it with other files to produce an executable, this
00017 * file does not by itself cause the resulting executable to be covered by
00018 * the GNU General Public License. This exception does not however
00019 * invalidate any other reasons why the executable file might be covered by
00020 * the GNU General Public License.
00021 */
00022
00023 #pragma once
00024
00025 #include <l4/sys/irq>
00026 #include <l4/sys/semaphore.h>
00027
00028 namespace L4 {
00029
00051 struct Semaphore : Kobject_t<Semaphore, Triggerable, L4_PROTO_SEMAPHORE>
00052 {
00065 l4_msgtag_t up(l4_utcb_t *utcb = l4_utcb()) throw()
00066 { return trigger(utcb); }
00067
00084 l4_msgtag_t down(l4_timeout_t timeout =
00085 L4_IPC_NEVER,
00086 l4_utcb_t *utcb = l4_utcb()) throw()
00087 { return l4_semaphore_down_u(cap(), timeout, utcb); }
00088 };
00089

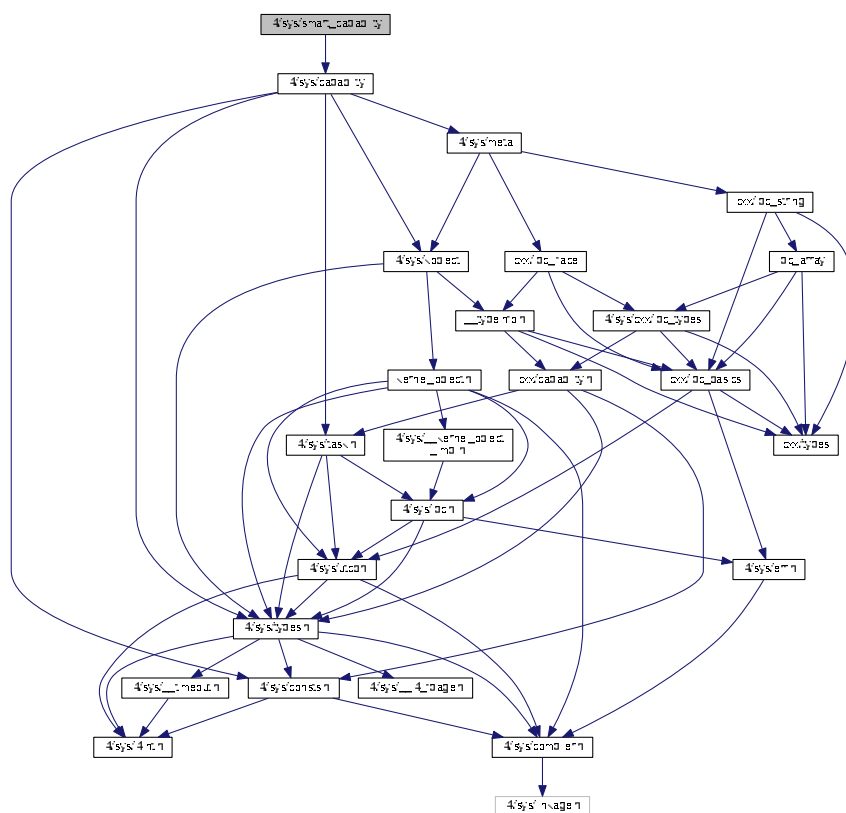
```

### 15.345 l4/sys/smart\_capability File Reference

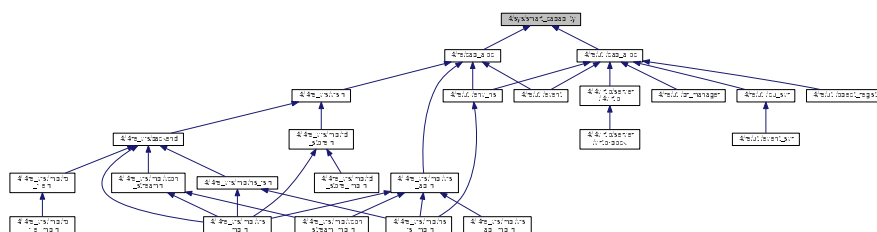
L4::Capability class.



Include dependency graph for smart\_capability:



This graph shows which files directly or indirectly include this file:



## Data Structures

- class L4::Smart\_cap< T, SMART >

*Smart capability class.*

## Namespaces

- 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) throw ()`  
*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) throw ()`  
*reinterpret\_cast for (smart) capabilities.*

### 15.345.1 Detailed Description

L4::Capability class.

#### Author

Alexander Warg [alexander.warg@os.inf.tu-dresden.de](mailto:alexander.warg@os.inf.tu-dresden.de)

Definition in file [smart\\_capability](#).

### 15.346 smart\_capability

```

00001 // vim:set ft=cpp:
00009 /*
00010 * (c) 2008-2009 Author(s)
00011 * economic rights: Technische Universität Dresden (Germany)
00012 *
00013 * This file is part of TUD:OS and distributed under the terms of the
00014 * GNU General Public License 2.
00015 * Please see the COPYING-GPL-2 file for details.
00016 *
00017 * As a special exception, you may use this file as part of a free software
00018 * library without restriction. Specifically, if other files instantiate
00019 * templates or use macros or inline functions from this file, or you compile
00020 * this file and link it with other files to produce an executable, this
00021 * file does not by itself cause the resulting executable to be covered by
00022 * the GNU General Public License. This exception does not however
00023 * invalidate any other reasons why the executable file might be covered by
00024 * the GNU General Public License.
00025 */
00026 #pragma once
00027
00028 #include <l4/sys/capability>
00029
00030 namespace L4 {
00031
00032 template< typename T, typename SMART >
00033 class Smart_cap : public Cap_base, private SMART
00034 {
00035 public:
00036 SMART const &smart() const { return *this; }
00037
00038 void _delete() throw()
00039 {
00040 SMART::free(const_cast<Smart_cap<T, SMART>&>(*this));
00041 }
00042
00043 Cap<T> release() const throw()
00044 {
00045 l4_cap_idx_t r = cap();
00046 SMART::invalidate(const_cast<Smart_cap<T, SMART>&>(*this));
00047
00048 return Cap<T>(r);
00049 }
00050
00051 void reset()
00052 {
00053 _c = L4_INVALID_CAP;
00054 }
00055
00056 private:
00057 l4_cap_idx_t cap() const { return _c; }
00058 void _c = L4_INVALID_CAP;
00059 };

```

```

00058 }
00059
00060 Smart_cap() throw() : Cap_base(Invalid) {}
00061
00062 Smart_cap(Cap_base::Cap_type t) throw() : Cap_base(t) {}
00063
00072 template< typename O >
00073 Smart_cap(Cap<O> const &p) throw() : Cap_base(p.cap())
00074 { T* __t = ((O*)100); (void)__t; }
00075
00076 template< typename O >
00077 Smart_cap(Cap<O> const &p, SMART const &smart) throw()
00078 : Cap_base(p.cap()), SMART(smart)
00079 { T* __t = ((O*)100); (void)__t; }
00080
00081 template< typename O >
00082 Smart_cap(Smart_cap<O, SMART> const &o) throw()
00083 : Cap_base(SMART::copy(o), SMART(o.smart()))
00084 { T* __t = ((O*)100); (void)__t; }
00085
00086 Smart_cap(Smart_cap const &o) throw()
00087 : Cap_base(SMART::copy(o), SMART(o.smart()))
00088 { }
00089
00090 template< typename O >
00091 Smart_cap(typename Cap<O>::Cap_type cap) throw() :
00092 Cap_base(cap)
00093 { T* __t = ((O*)100); (void)__t; }
00094
00095 void operator = (typename Cap<T>::Cap_type cap) throw()
00096 {
00097 _delete();
00098 _c = cap;
00099 }
00100
00101 template< typename O >
00102 void operator = (Smart_cap<O, SMART> const &o) throw()
00103 {
00104 _delete();
00105 _c = this->SMART::copy(o).cap();
00106 this->SMART::operator = (o.smart());
00107 // return *this;
00108 }
00109
00110 Smart_cap const &operator = (Smart_cap const &o) throw()
00111 {
00112 if (&o == this)
00113 return *this;
00114
00115 _delete();
00116 _c = this->SMART::copy(o).cap();
00117 this->SMART::operator = (o.smart());
00118 return *this;
00119 }
00120
00121 #if __cplusplus >= 201103L
00122 template< typename O >
00123 Smart_cap(Smart_cap<O, SMART> &&o) throw()
00124 : Cap_base(o.release()), SMART(o.smart())
00125 { T* __t = ((O*)100); (void)__t; }
00126
00127 Smart_cap(Smart_cap &&o) throw()
00128 : Cap_base(o.release()), SMART(o.smart())
00129 { }
00130
00131 template< typename O >
00132 void operator = (Smart_cap<O, SMART> &&o) throw()
00133 {
00134 _delete();
00135 _c = o.release().cap();
00136 this->SMART::operator = (o.smart());
00137 // return *this;
00138 }
00139
00140 Smart_cap const &operator = (Smart_cap &&o) throw()
00141 {
00142 if (&o == this)
00143 return *this;
00144
00145 _delete();
00146 _c = o.release().cap();
00147 this->SMART::operator = (o.smart());
00148 return *this;
00149 }
00150 #endif
00151
00152 Cap<T> operator -> () const throw() { return Cap<T>(_c); }

```

```

00155
00156 Cap<T> get() const throw() { return Cap<T>(_c); }
00157
00158 ~Smart_cap() throw() { _delete(); }
00159 };
00160
00161 template< typename T >
00162 class Weak_cap : public Cap_base
00163 {
00164 public:
00165 Weak_cap() : Cap_base(Invalid) {}
00166
00167 template< typename O >
00168 Weak_cap(typename Cap<O>::Cap_type t) : Cap_base(t)
00169 { T* __t = ((O*)100); (void)__t; }
00170
00171 template< typename O, typename S >
00172 Weak_cap(Smart_cap<O, S> const &c) : Cap_base(c.cap())
00173 { T* __t = ((O*)100); (void)__t; }
00174
00175 Weak_cap(Weak_cap const &o) : Cap_base(o) {}
00176
00177 template< typename O >
00178 Weak_cap(Weak_cap<O> const &o) : Cap_base(o)
00179 { T* __t = ((O*)100); (void)__t; }
00180
00181 };
00182
00183 namespace Cap_traits {
00184 template< typename T1, typename T2 >
00185 struct Type { enum { Equal = false }; };
00186
00187 template< typename T1 >
00188 struct Type<T1,T1> { enum { Equal = true }; };
00189 };
00190
00201 template< typename T, typename F, typename SMART >
00202 inline
00203 Smart_cap<T, SMART> cap_cast(Smart_cap<F, SMART> const &c)
00204 throw()
00205 {
00206 (void)static_cast<T const *>(reinterpret_cast<F const *>(100));
00207 return Smart_cap<T, SMART>(Cap<T>(SMART::copy(c).cap()));
00208 }
00209
00220 template< typename T, typename F, typename SMART >
00221 inline
00222 Smart_cap<T, SMART> cap_reinterpret_cast(
00223 Smart_cap<F, SMART> const &c) throw()
00224 {
00225 return Smart_cap<T, SMART>(Cap<T>(SMART::copy(c).cap()));
00226 }
00227
00228 }
00229

```

## 15.347 l4/sys/task File Reference

Common task related definitions.

```

#include <l4/sys/task.h>
#include <l4/sys/capability>

```

- class `L4::Task`  
*C++ interface of the `Task` kernel object.*

- **L4**  
*L4 low-level kernel interface.*

### 15.347.1 Detailed Description

Common task related definitions.

Definition in file [task](#).

## 15.348 task

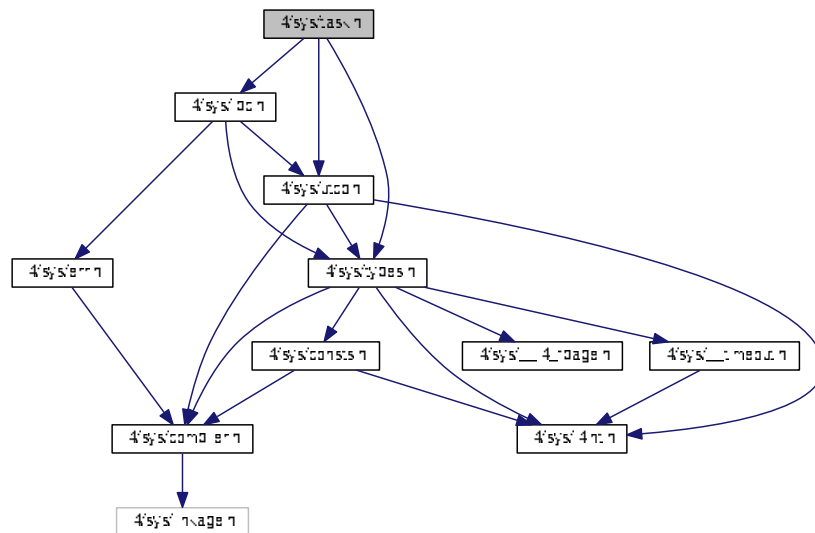
```

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 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024
00025 #pragma once
00026
00027 #include <l4/sys/task.h>
00028 #include <l4/sys/capability>
00029
00030 namespace L4 {
00031
00043 class Task :
00044 public Kobject<Task, Kobject, L4_PROTO_TASK,
00045 Type_info::Demand_t<2> >
00046 {
00047 public:
00067 l4_msgtag_t map(Cap<Task> const &src_task,
00068 l4_fpage_t const &snd_fpage, l4_addr_t snd_base,
00069 l4_utcb_t *utcb = l4_utcb()) throw()
00070 { return l4_task_map_u(cap(), src_task.cap(), snd_fpage, snd_base, utcb); }
00071
00090 l4_msgtag_t unmap(l4_fpage_t const &fpage,
00091 l4_umword_t map_mask,
00092 l4_utcb_t *utcb = l4_utcb()) throw()
00093 { return l4_task_unmap_u(cap(), fpage, map_mask, utcb); }
00094
00115 l4_msgtag_t unmap_batch(l4_fpage_t const *fpages,
00116 unsigned num_fpages,
00117 l4_umword_t map_mask,
00118 l4_utcb_t *utcb = l4_utcb()) throw()
00119 { return l4_task_unmap_batch_u(cap(), fpages, num_fpages, map_mask, utcb); }
00120
00133 l4_msgtag_t delete_obj(L4::Cap<void> obj,
00134 l4_utcb_t *utcb = l4_utcb()) throw()
00135 { return l4_task_delete_obj_u(cap(), obj.cap(), utcb); }
00136
00147 l4_msgtag_t release_cap(L4::Cap<void> cap,
00148 l4_utcb_t *utcb = l4_utcb()) throw()
00149 { return l4_task_release_cap_u(this->cap(), cap.cap(), utcb); }
00150
00163 l4_msgtag_t cap_valid(Cap<void> const &cap,
00164 l4_utcb_t *utcb = l4_utcb()) throw()
00165 { return l4_task_cap_valid_u(this->cap(), cap.cap(), utcb); }
00166
00179 l4_msgtag_t cap_has_child(Cap<void> const &cap,
00180 l4_utcb_t *utcb = l4_utcb()) throw()
00181 { L4_DEPRECATED("Do not use. Future uncertain.");
00182 return l4_task_cap_has_child_u(this->cap(), cap.cap(), utcb); }
00183
00196 l4_msgtag_t cap_equal(Cap<void> const &cap_a,
00197 Cap<void> const &cap_b,
00198 l4_utcb_t *utcb = l4_utcb()) throw()
00199 { return l4_task_cap_equal_u(cap(), cap_a.cap(), cap_b.cap(), utcb); }
00200
00209 l4_msgtag_t add_ku_mem(l4_fpage_t const &fpage,

```

## 15.349 I4/sys/task.h File Reference

```
#include <linux/sys/types.h>
#include <linux/sys/utcb.h>
#include <linux/sys/ipc.h>
Include dependency graph for task.h:
```

[illegible]

- 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 }  
*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_addr_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, unsigned long 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 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_has_child (l4_cap_idx_t task, l4_cap_idx_t cap) L4_NOTHROW`  
*Test whether a capability has child mappings (in another task).*
- `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 rights.*
- `l4_msgtag_t l4_task_add_ku_mem (l4_cap_idx_t task, l4_fpage_t ku_mem) L4_NOTHROW`  
*Add kernel-user memory.*

### 15.349.1 Detailed Description

Common task related definitions.

Definition in file [task.h](#).

### 15.350 task.h

```

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 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025 #pragma once
00026 #include <l4/sys/types.h>
00027 #include <l4/sys/utcb.h>
00028
00060 L4_INLINE l4_msgtag_t
00061 l4_task_map(l4_cap_idx_t dst_task, l4_cap_idx_t src_task,
00062 l4_fpage_t snd_fpage, l4_addr_t snd_base)
00063 L4_NOTHROW;
```



```

00063
00067 L4_INLINE l4_msgtag_t
00068 l4_task_map_u(l4_cap_idx_t dst_task, l4_cap_idx_t src_task,
00069 l4_fpage_t snd_fpage, l4_addr_t snd_base,
00070 l4_utcb_t *utcb) L4_NOTHROW;
00070
00089 L4_INLINE l4_msgtag_t
00090 l4_task_unmap(l4_cap_idx_t task, l4_fpage_t fpage,
00091 l4_umword_t map_mask) L4_NOTHROW;
00092
00096 L4_INLINE l4_msgtag_t
00097 l4_task_unmap_u(l4_cap_idx_t task, l4_fpage_t fpage,
00098 l4_umword_t map_mask, l4_utcb_t *utcb)
00099 L4_NOTHROW;
00099
00123 L4_INLINE l4_msgtag_t
00124 l4_task_unmap_batch(l4_cap_idx_t task, l4_fpage_t const *fpages,
00125 unsigned num_fpages, unsigned long map_mask) L4_NOTHROW;
00126
00130 L4_INLINE l4_msgtag_t
00131 l4_task_unmap_batch_u(l4_cap_idx_t task, l4_fpage_t const *fpages,
00132 unsigned num_fpages, unsigned long map_mask,
00133 l4_utcb_t *u) L4_NOTHROW;
00134
00150 L4_INLINE l4_msgtag_t
00151 l4_task_delete_obj(l4_cap_idx_t task, l4_cap_idx_t obj)
00152 L4_NOTHROW;
00152
00156 L4_INLINE l4_msgtag_t
00157 l4_task_delete_obj_u(l4_cap_idx_t task, l4_cap_idx_t obj,
00158 l4_utcb_t *u) L4_NOTHROW;
00159
00171 L4_INLINE l4_msgtag_t
00172 l4_task_release_cap(l4_cap_idx_t task,
00173 l4_cap_idx_t cap) L4_NOTHROW;
00173
00177 L4_INLINE l4_msgtag_t
00178 l4_task_release_cap_u(l4_cap_idx_t task, l4_cap_idx_t cap,
00179 l4_utcb_t *u) L4_NOTHROW;
00180
00181
00195 L4_INLINE l4_msgtag_t
00196 l4_task_cap_valid(l4_cap_idx_t task, l4_cap_idx_t cap)
00197 L4_NOTHROW;
00197
00201 L4_INLINE l4_msgtag_t
00202 l4_task_cap_valid_u(l4_cap_idx_t task, l4_cap_idx_t cap,
00203 l4_utcb_t *utcb) L4_NOTHROW;
00203
00216 L4_INLINE l4_msgtag_t
00217 l4_task_cap_has_child(l4_cap_idx_t task,
00218 l4_cap_idx_t cap) L4_NOTHROW
00219 L4_DEPRECATED("Do not use. Future uncertain.");
00219
00223 L4_INLINE l4_msgtag_t
00224 l4_task_cap_has_child_u(l4_cap_idx_t task, l4_cap_idx_t cap,
00225 l4_utcb_t *utcb) L4_NOTHROW;
00225
00238 L4_INLINE l4_msgtag_t
00239 l4_task_cap_equal(l4_cap_idx_t task, l4_cap_idx_t cap_a,
00240 l4_cap_idx_t cap_b) L4_NOTHROW;
00241
00245 L4_INLINE l4_msgtag_t
00246 l4_task_add_ku_mem_u(l4_cap_idx_t task, l4_fpage_t ku_mem,
00247 l4_utcb_t *u) L4_NOTHROW;
00248
00258 L4_INLINE l4_msgtag_t
00259 l4_task_add_ku_mem(l4_cap_idx_t task, l4_fpage_t ku_mem)
00260 L4_NOTHROW;
00260
00261
00265 L4_INLINE l4_msgtag_t
00266 l4_task_cap_equal_u(l4_cap_idx_t task, l4_cap_idx_t cap_a,
00267 l4_cap_idx_t cap_b, l4_utcb_t *utcb)
00268 L4_NOTHROW;
00268
00273 enum L4_task_ops
00274 {
00275 L4_TASK_MAP_OP = 0UL,
00276 L4_TASK_UNMAP_OP = 1UL,
00277 L4_TASK_CAP_INFO_OP = 2UL,
00278 L4_TASK_ADD_KU_MEM_OP = 3UL,
00279 L4_TASK_LDT_SET_X86_OP = 0x11UL,
00280 };
00281
00282
00283 /* IMPLEMENTATION ----- */

```

```

00284
00285 #include <l4/sys/ipc.h>
00286
00287
00288 L4_INLINE l4_msgtag_t
00289 l4_task_map_u(l4_cap_idx_t dst_task, l4_cap_idx_t src_task,
00290 l4_fpage_t snd_fpage, unsigned long snd_base, l4_utcb_t *u)
00291 L4_NOTHROW
00292 {
00293 l4_msg_regs_t *v = l4_utcb_mr_u(u);
00294 v->mr[0] = L4_TASK_MAP_OP;
00295 v->mr[3] = l4_map_obj_control(0,0);
00296 v->mr[4] = l4_obj_fpage(src_task, 0, L4_FPAGE_RWX).
00297 raw;
00298 v->mr[1] = snd_base;
00299 v->mr[2] = snd_fpage.raw;
00300 return l4_ipc_call(dst_task, u, l4_msgtag(L4_PROTO_TASK, 3, 1, 0),
00301 L4_IPC_NEVER);
00302 }
00303
00304 L4_INLINE l4_msgtag_t
00305 l4_task_unmap_u(l4_cap_idx_t task, l4_fpage_t fpage,
00306 unsigned long map_mask, l4_utcb_t *u) L4_NOTHROW
00307 {
00308 l4_msg_regs_t *v = l4_utcb_mr_u(u);
00309 v->mr[0] = L4_TASK_UNMAP_OP;
00310 v->mr[1] = map_mask;
00311 v->mr[2] = fpage.raw;
00312 return l4_ipc_call(task, u, l4_msgtag(L4_PROTO_TASK, 3, 0, 0),
00313 L4_IPC_NEVER);
00314 }
00315
00316 L4_INLINE l4_msgtag_t
00317 l4_task_unmap_batch_u(l4_cap_idx_t task, l4_fpage_t const *fpages,
00318 unsigned num_fpages, unsigned long map_mask,
00319 l4_utcb_t *u) L4_NOTHROW
00320 {
00321 l4_msg_regs_t *v = l4_utcb_mr_u(u);
00322 v->mr[0] = L4_TASK_UNMAP_OP;
00323 v->mr[1] = map_mask;
00324 __builtin_memcpy(&v->mr[2], fpages, num_fpages * sizeof(l4_fpage_t));
00325 return l4_ipc_call(task, u, l4_msgtag(L4_PROTO_TASK, 2 + num_fpages, 0,
00326 0), L4_IPC_NEVER);
00327 }
00328
00329 L4_INLINE l4_msgtag_t
00330 l4_task_cap_valid_u(l4_cap_idx_t task, l4_cap_idx_t cap,
00331 l4_utcb_t *u) L4_NOTHROW
00332 {
00333 l4_msg_regs_t *v = l4_utcb_mr_u(u);
00334 v->mr[0] = L4_TASK_CAP_INFO_OP;
00335 v->mr[1] = cap & ~1UL;
00336 return l4_ipc_call(task, u, l4_msgtag(L4_PROTO_TASK, 2, 0, 0),
00337 L4_IPC_NEVER);
00338 }
00339
00340 L4_INLINE l4_msgtag_t
00341 l4_task_cap_has_child_u(l4_cap_idx_t task, l4_cap_idx_t cap,
00342 l4_utcb_t *u) L4_NOTHROW
00343 {
00344 l4_msg_regs_t *v = l4_utcb_mr_u(u);
00345 v->mr[0] = L4_TASK_CAP_INFO_OP;
00346 v->mr[1] = cap | 1UL;
00347 return l4_ipc_call(task, u, l4_msgtag(L4_PROTO_TASK, 2, 0, 0),
00348 L4_IPC_NEVER);
00349 }
00350
00351 L4_INLINE l4_msgtag_t
00352 l4_task_cap_equal_u(l4_cap_idx_t task, l4_cap_idx_t cap_a,
00353 l4_cap_idx_t cap_b, l4_utcb_t *u)
00354 L4_NOTHROW
00355 {
00356 l4_msg_regs_t *v = l4_utcb_mr_u(u);
00357 v->mr[0] = L4_TASK_CAP_INFO_OP;
00358 v->mr[1] = cap_a;
00359 v->mr[2] = cap_b;
00360 return l4_ipc_call(task, u, l4_msgtag(L4_PROTO_TASK, 3, 0, 0),
00361 L4_IPC_NEVER);
00362 }
00363
00364 L4_INLINE l4_msgtag_t
00365 l4_task_add_ku_mem_u(l4_cap_idx_t task, l4_fpage_t ku_mem,
00366 l4_utcb_t *u) L4_NOTHROW
00367 {
00368 l4_msg_regs_t *v = l4_utcb_mr_u(u);
00369 v->mr[0] = L4_TASK_ADD_KU_MEM_OP;
00370 v->mr[1] = ku_mem.raw;

```

```

00360 return l4_ipc_call(task, u, l4_msgtag(L4_PROTO_TASK, 2, 0, 0),
00361 L4_IPC_NEVER);
00362 }
00363
00364
00365 L4_INLINE l4_msgtag_t
00366 l4_task_map(l4_cap_idx_t dst_task, l4_cap_idx_t src_task,
00367 l4_fpage_t snd_fpage, unsigned long snd_base) L4_NOTHROW
00368 {
00369 return l4_task_map_u(dst_task, src_task, snd_fpage, snd_base, l4_utcb());
00370 }
00371
00372 L4_INLINE l4_msgtag_t
00373 l4_task_unmap(l4_cap_idx_t task, l4_fpage_t fpage,
00374 unsigned long map_mask) L4_NOTHROW
00375 {
00376 return l4_task_unmap_u(task, fpage, map_mask, l4_utcb());
00377 }
00378
00379 L4_INLINE l4_msgtag_t
00380 l4_task_unmap_batch(l4_cap_idx_t task, l4_fpage_t const *fpages,
00381 unsigned num_fpages, unsigned long map_mask) L4_NOTHROW
00382 {
00383 return l4_task_unmap_batch_u(task, fpages, num_fpages, map_mask,
00384 l4_utcb());
00385 }
00386
00387 L4_INLINE l4_msgtag_t
00388 l4_task_delete_obj_u(l4_cap_idx_t task, l4_cap_idx_t obj,
00389 l4_utcb_t *u) L4_NOTHROW
00390 {
00391 return l4_task_unmap_u(task, l4_obj_fpage(obj, 0,
00392 L4_CAP_FPAGE_RWSD),
00393 L4_FP_DELETE_OBJ, u);
00394 }
00395
00396 L4_INLINE l4_msgtag_t
00397 l4_task_delete_obj(l4_cap_idx_t task, l4_cap_idx_t obj)
00398 L4_NOTHROW
00399 {
00400 return l4_task_delete_obj_u(task, obj, l4_utcb());
00401 }
00402
00403 L4_INLINE l4_msgtag_t
00404 l4_task_release_cap_u(l4_cap_idx_t task, l4_cap_idx_t cap,
00405 l4_utcb_t *u) L4_NOTHROW
00406 {
00407 return l4_task_unmap_u(task, l4_obj_fpage(cap, 0,
00408 L4_CAP_FPAGE_RWSD),
00409 L4_FP_ALL_SPACES, u);
00410 }
00411
00412 L4_INLINE l4_msgtag_t
00413 l4_task_release_cap(l4_cap_idx_t task,
00414 l4_cap_idx_t cap) L4_NOTHROW
00415 {
00416 return l4_task_release_cap_u(task, cap, l4_utcb());
00417 }
00418
00419 L4_INLINE l4_msgtag_t
00420 l4_task_cap_valid(l4_cap_idx_t task, l4_cap_idx_t cap)
00421 L4_NOTHROW
00422 {
00423 return l4_task_cap_valid_u(task, cap, l4_utcb());
00424 }
00425
00426 L4_INLINE l4_msgtag_t
00427 l4_task_cap_has_child(l4_cap_idx_t task,
00428 l4_cap_idx_t cap) L4_NOTHROW
00429 {
00430 return l4_task_cap_has_child_u(task, cap, l4_utcb());
00431 }
00432
00433 L4_INLINE l4_msgtag_t
00434 l4_task_cap_equal(l4_cap_idx_t task, l4_cap_idx_t cap_a,
00435 l4_cap_idx_t cap_b) L4_NOTHROW
00436 {
00437 return l4_task_cap_equal_u(task, cap_a, cap_b, l4_utcb());
00438 }
00439
00440 L4_INLINE l4_msgtag_t
00441 l4_task_add_ku_mem(l4_cap_idx_t task, l4_fpage_t ku_mem)
00442 L4_NOTHROW
00443 {
00444 return l4_task_add_ku_mem_u(task, ku_mem, l4_utcb());
00445 }

```



## Data Structures

- class [L4::Thread](#)  
*C++ L4 kernel thread interface.*
- class [L4::Thread::Attr](#)  
*Thread attributes used for control\_commit().*
- class [L4::Thread::Modify\\_senders](#)  
*Wrapper class for modifying senders.*

## Namespaces

- [L4](#)  
*L4 low-level kernel interface.*

### 15.351.1 Detailed Description

Common thread related definitions.

Definition in file [thread](#).

## 15.352 thread

```

00001 // vi:set ft=c++: -- 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 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024
00025 #pragma once
00026
00027 #include <l4/sys/capability>
00028 #include <l4/sys/thread.h>
00029
00030 namespace L4 {
00031
00058 class Thread :
00059 public Kobject_t<Thread, Kobject, L4_PROTO_THREAD,
00060 Type_info::Demand_t<1> >
00061 {
00062 public:
00085 l4_msgtag_t ex_regs(l4_addr_t ip, l4_addr_t sp,
00086 l4_umword_t flags,
00087 l4_utcb_t *utcb = l4_utcb()) throw()
00088 { return l4_thread_ex_regs_u(cap(), ip, sp, flags, utcb); }
00089
00111 l4_msgtag_t ex_regs(l4_addr_t *ip, l4_addr_t *sp,
00112 l4_umword_t *flags,
00113 l4_utcb_t *utcb = l4_utcb()) throw()
00114 { return l4_thread_ex_regs_ret_u(cap(), ip, sp, flags, utcb); }
00115
00116
00125 class Attr

```

```

00126 {
00127 private:
00128 friend class L4::Thread;
00129 l4_utcb_t *_u;
00130
00131 public:
00132 explicit Attr(l4_utcb_t *utcb = l4_utcb()) throw() : _u(utcb)
00133 { l4_thread_control_start_u(utcb); }
00134
00135 void pager(Cap<void> const &pager) throw()
00136 { l4_thread_control_pager_u(pager.cap(), _u); }
00137
00138 Cap<void> pager() throw()
00139 { return Cap<void>(l4_utcb_mr_u(_u)->mr[1]); }
00140
00141 void exc_handler(Cap<void> const &exc_handler) throw()
00142 { l4_thread_control_exc_handler_u(exc_handler.cap(), _u); }
00143
00144 Cap<void> exc_handler() throw()
00145 { return Cap<void>(l4_utcb_mr_u(_u)->mr[2]); }
00146
00147 void bind(l4_utcb_t *thread_utcb, Cap<Task> const &task) throw()
00148 { l4_thread_control_bind_u(thread_utcb, task.cap(), _u); }
00149
00150 void alien(int on) throw()
00151 { l4_thread_control_alien_u(_u, on); }
00152
00153 void ux_host_syscall(int on) throw()
00154 { l4_thread_control_ux_host_syscall_u(_u, on); }
00155
00156 };
00157
00158 l4_msgtag_t control(Attr const &attr) throw()
00159 { return l4_thread_control_commit_u(cap(), attr._u); }
00160
00161 l4_msgtag_t switch_to(l4_utcb_t *utcb = l4_utcb()) throw()
00162 { return l4_thread_switch_u(cap(), utcb); }
00163
00164 l4_msgtag_t stats_time(l4_kernel_clock_t *us,
00165 l4_utcb_t *utcb = l4_utcb()) throw()
00166 { return l4_thread_stats_time_u(cap(), us, utcb); }
00167
00168 l4_msgtag_t vcpu_resume_start(l4_utcb_t *utcb =
00169 l4_utcb()) throw()
00170 { return l4_thread_vcpu_resume_start_u(utcb); }
00171
00172 l4_msgtag_t vcpu_resume_commit(l4_msgtag_t tag,
00173 l4_utcb_t *utcb = l4_utcb()) throw()
00174 { return l4_thread_vcpu_resume_commit_u(cap(), tag, utcb); }
00175
00176 l4_msgtag_t vcpu_control(l4_addr_t vcpu_state,
00177 l4_utcb_t *utcb = l4_utcb())
00178 throw()
00179 { return l4_thread_vcpu_control_u(cap(), vcpu_state, utcb); }
00180
00181 l4_msgtag_t vcpu_control_ext(l4_addr_t ext_vcpu_state,
00182 l4_utcb_t *utcb = l4_utcb()) throw()
00183 { return l4_thread_vcpu_control_ext_u(cap(), ext_vcpu_state, utcb); }
00184
00185 l4_msgtag_t register_del_irq(Cap<Irq> irq,
00186 l4_utcb_t *u = l4_utcb()) throw()
00187 { return l4_thread_register_del_irq_u(cap(), irq.cap(), u); }
00188
00189 class Modify_senders
00190 {
00191 private:
00192 friend class Thread;
00193 l4_utcb_t *utcb;
00194 unsigned cnt;
00195
00196 public:
00197 explicit Modify_senders(l4_utcb_t *u = l4_utcb()) throw()
00198 : utcb(u), cnt(1)
00199 {
00200 l4_utcb_mr_u(utcb)->mr[0] = L4_THREAD_MODIFY_SENDER_OP;
00201 }
00202
00203 int add(l4_umword_t match_mask, l4_umword_t match,
00204 l4_umword_t del_bits, l4_umword_t add_bits) throw()
00205 {
00206 l4_msg_regs_t *m = l4_utcb_mr_u(utcb);
00207 if (cnt >= L4_UTCB_GENERIC_DATA_SIZE - 4)
00208 return -L4_ENOMEM;
00209 m->mr[cnt++] = match_mask;
00210 m->mr[cnt++] = match;
00211 m->mr[cnt++] = del_bits;
00212 m->mr[cnt++] = add_bits;
00213 }
00214

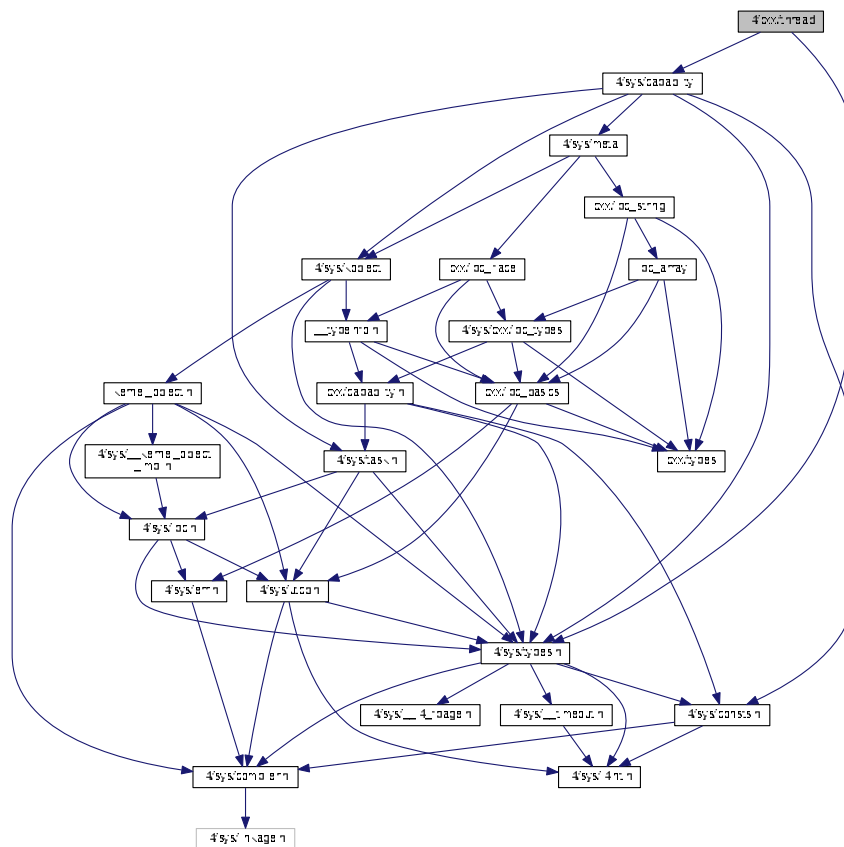
```

```
00362 return 0;
00363 }
00364 };
00365
00373 l4_msgtag_t modify_senders(Modify_senders const &todo) throw()
00374 {
00375 return l4_ipc_call(cap(), todo.utcb, l4_msgtag(
00376 L4_PROTO_THREAD, todo.cnt, 0, 0), L4_IPC_NEVER);
00377 };
00378 }
```

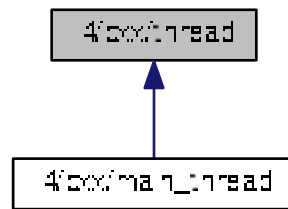
### 15.353 I4/cxx/thread File Reference

### Thread implementation.

```
#include <linux/sys/capability>
#include <linux/sys/types.h>
Include dependency graph for thread:
```



This graph shows which files directly or indirectly include this file:



## Namespaces

- [cxx](#)

*Our C++ library.*

### 15.353.1 Detailed Description

Thread implementation.

Definition in file [thread](#).

## 15.354 thread

```

00001
00005 /*
00006 * (c) 2004-2009 Alexander Warg <warg@os.inf.tu-dresden.de>
00007 * economic rights: Technische Universität Dresden (Germany)
00008 * This file is part of TUD:OS and distributed under the terms of the
00009 * GNU Lesser General Public License 2.1.
00010 * Please see the COPYING-LGPL-2.1 file for details.
00011 */
00012
00013 #ifndef CXX_THREAD_H__
00014 #define CXX_THREAD_H__
00015
00016 #include <l4/sys/capability>
00017 #include <l4/sys/types.h>
00018
00019 namespace cxx {
00020
00021 class Thread
00022 {
00023 public:
00024
00025 enum State
00026 {
00027 Dead = 0,
00028 Running = 1,
00029 Stopped = 2,
00030 };
00031
00032 Thread(bool initiate);
00033 Thread(void *stack);
00034 Thread(void *stack, L4::Cap<L4::Thread> const &cap);
00035 virtual ~Thread();
00036 void execute() asm ("L4_Thread_execute");

```



```

00037 virtual void run() = 0;
00038 virtual void shutdown() asm ("L4_Thread_shutdown");
00039 void start();
00040 void stop();
00041
00042 L4::Cap<L4::Thread> self() const throw()
00043 { return _cap; }
00044
00045 State state() const
00046 { return _state; }
00047
00048 static void start_cxx_thread(Thread *_this)
00049 asm ("L4_Thread_start_cxx_thread");
00050
00051 static void kill_cxx_thread(Thread *_this)
00052 asm ("L4_Thread_kill_cxx_thread");
00053
00054 static void set_pager(L4::Cap<void>const &p) throw()
00055 { _pager = p; }
00056
00057 private:
00058 int create();
00059
00060 L4::Cap<L4::Thread> _cap;
00061 State _state;
00062
00063 protected:
00064 void *_stack;
00065
00066 private:
00067 static L4::Cap<void> _pager;
00068 static L4::Cap<void> _master;
00069 };
00070
00071 };
00072
00073 #endif /* CXX_THREAD_H__ */
00074

```

## 15.355 l4/sys/typeinfo\_svr File Reference

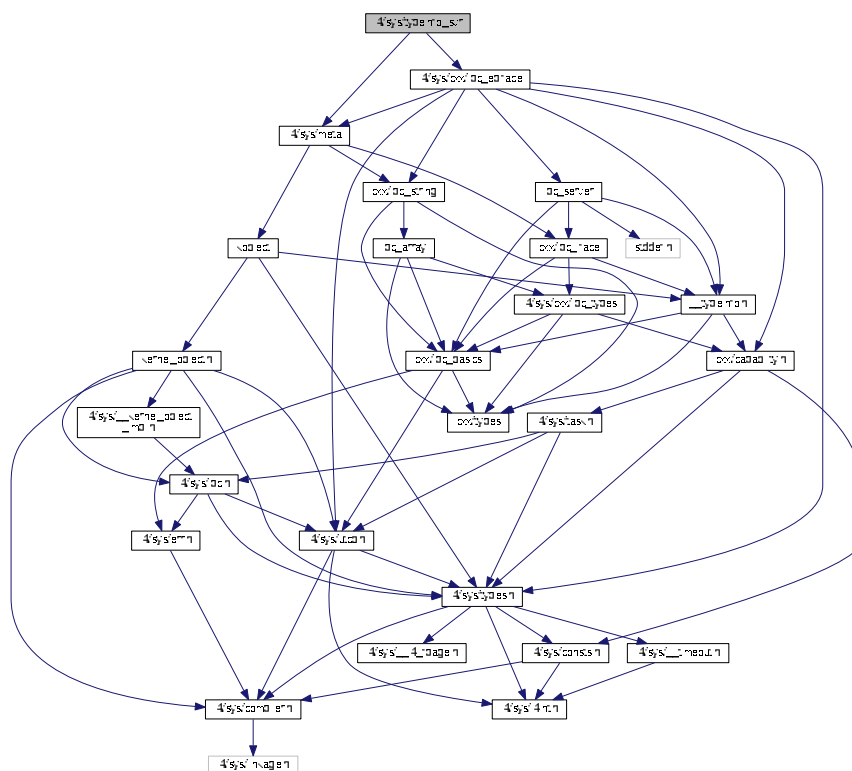
Type information server template.

```

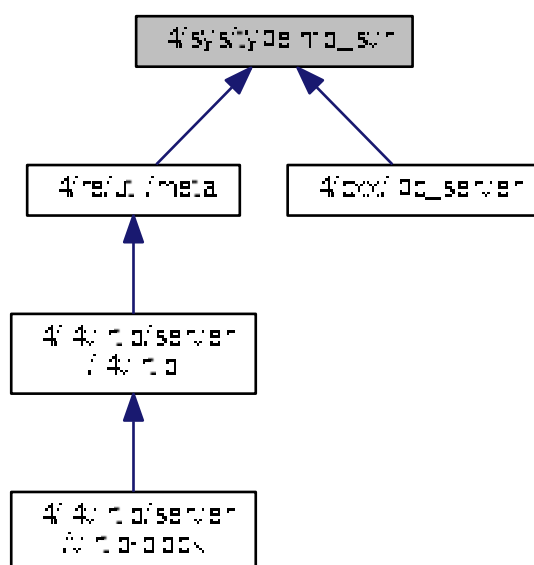
#include <l4/sys/meta>
#include <l4/sys/cxx/ipc_epiface>

```

Include dependency graph for typeinfo\_svr:



This graph shows which files directly or indirectly include this file:



## Namespaces

- [L4](#)

[L4](#) low-level kernel interface.

### 15.355.1 Detailed Description

Type information server template.

Definition in file [typeinfo\\_svr](#).

## 15.356 typeinfo\_svr

```

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 * 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 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
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 #pragma once
00025
00026 #include <l4/sys/meta>
00027 #include <l4/sys/cxx/ipc_epiface>
00028
00029 namespace L4 { namespace Util {
00030
00031 template<typename KO, typename IOS>
00032 long handle_meta_request(IOS &ios)
00033 {
00034 using L4::Ipc::Msg::dispatch_call;
00035 typedef L4::Ipc::Detail::Meta_svr<KO> Msvr;
00036 l4_msgtag_t tag = dispatch_call<L4::Meta::Rpcs>((Msvr *)0, ios.utcb(),
00037 ios.tag(), 0);
00038 ios.set_ipc_params(tag);
00039 return tag.label();
00040 }
00041
00042 }}
```

## 15.357 l4/sys/utcb.h File Reference

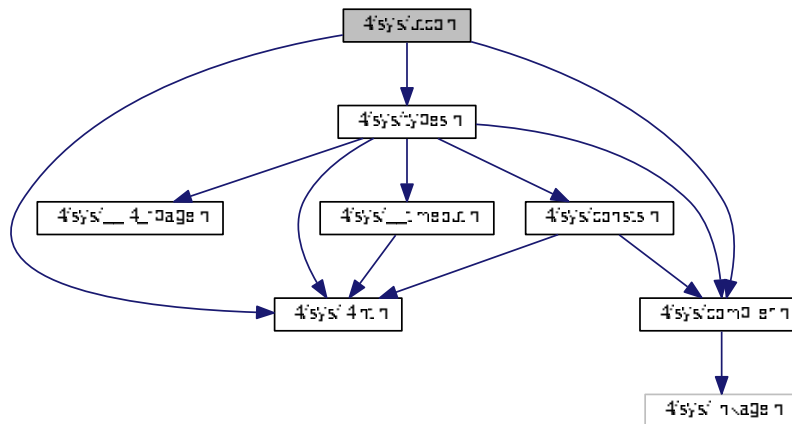
UTCB definitions.

```

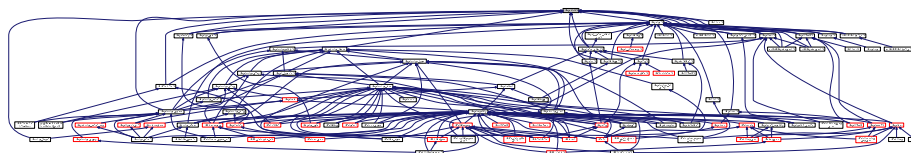
#include <l4/sys/types.h>
#include <l4/sys/compiler.h>
```

```
#include <l4/sys/l4int.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_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).*
- `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()`.*
- `void l4_utcb_inherit_fpu (int switch_on) L4_NOTHROW`  
*Enable or disable inheritance of FPU state to receiver.*
- `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.*

### 15.357.1 Detailed Description

UTCB definitions.

Definition in file `utcb.h`.

## 15.358 utcb.h

```

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 *
00013 * This file is part of TUD:OS and distributed under the terms of the
00014 * GNU General Public License 2.
00015 * Please see the COPYING-GPL-2 file for details.
00016 *
00017 * As a special exception, you may use this file as part of a free software
00018 * library without restriction. Specifically, if other files instantiate
00019 * templates or use macros or inline functions from this file, or you compile
00020 * this file and link it with other files to produce an executable, this
00021 * file does not by itself cause the resulting executable to be covered by

```

```

00022 * the GNU General Public License. This exception does not however
00023 * invalidate any other reasons why the executable file might be covered by
00024 * the GNU General Public License.
00025 */
00026 /*****
00027 #ifndef _L4_SYS_UTCB_H
00028 #define _L4_SYS_UTCB_H
00029
00030 #include <l4/sys/types.h>
00031 #include <l4/sys/compiler.h>
00032 #include <l4/sys/l4int.h>
00033
00067 typedef struct l4_utcb_t l4_utcb_t;
00068
00078 typedef union l4_msg_regs_t
00079 {
00080 l4_umword_t mr[L4_UTCB_GENERIC_DATA_SIZE];
00081 l4_uint64_t mr64[L4_UTCB_GENERIC_DATA_SIZE / (sizeof(
00082 l4_uint64_t)/sizeof(l4_umword_t))];
00083 } l4_msg_regs_t;
00083
00093 typedef struct l4_buf_regs_t
00094 {
00096 l4_umword_t bdr;
00097
00099 l4_umword_t br[L4_UTCB_GENERIC_BUFFERS_SIZE];
00100 } l4_buf_regs_t;
00101
00110 typedef struct l4_thread_regs_t
00111 {
00113 l4_umword_t error;
00115 l4_timeout_t xfer;
00117 l4_umword_t user[3];
00118 } l4_thread_regs_t;
00119
00120 __BEGIN_DECLS
00121
00132 L4_CV l4_utcb_t *l4_utcb_wrap(void) L4_NOTHROW L4_PURE;
00133
00139 L4_INLINE l4_utcb_t *l4_utcb_direct(void) L4_NOTHROW L4_PURE;
00140
00145 L4_INLINE l4_utcb_t *l4_utcb(void) L4_NOTHROW L4_PURE;
00146
00152 L4_INLINE l4_msg_regs_t *l4_utcb_mr(void) L4_NOTHROW L4_PURE;
00153
00158 L4_INLINE l4_msg_regs_t *l4_utcb_mr_u(l4_utcb_t *u) L4_NOTHROW L4_PURE;
00159
00166 L4_INLINE l4_buf_regs_t *l4_utcb_br(void) L4_NOTHROW L4_PURE;
00167
00172 L4_INLINE l4_buf_regs_t *l4_utcb_br_u(l4_utcb_t *u)
00173 L4_NOTHROW L4_PURE;
00173
00179 L4_INLINE l4_thread_regs_t *l4_utcb_tcr(void)
00180 L4_NOTHROW L4_PURE;
00180
00185 L4_INLINE l4_thread_regs_t *l4_utcb_tcr_u(l4_utcb_t *u)
00186 L4_NOTHROW L4_PURE;
00186
00199 L4_INLINE l4_exc_regs_t *l4_utcb_exc(void) L4_NOTHROW L4_PURE;
00200
00205 L4_INLINE l4_exc_regs_t *l4_utcb_exc_u(l4_utcb_t *u)
00206 L4_NOTHROW L4_PURE;
00206
00214 L4_INLINE l4_umword_t l4_utcb_exc_pc(l4_exc_regs_t const *u)
00215 L4_NOTHROW L4_PURE;
00215
00224 L4_INLINE void l4_utcb_exc_pc_set(l4_exc_regs_t *u,
00225 l4_addr_t pc) L4_NOTHROW;
00225
00230 L4_INLINE unsigned long l4_utcb_exc_typeval(l4_exc_regs_t const *u)
00231 L4_NOTHROW L4_PURE;
00231
00241 L4_INLINE int l4_utcb_exc_is_pf(l4_exc_regs_t const *u) L4_NOTHROW L4_PURE;
00242
00247 L4_INLINE l4_addr_t l4_utcb_exc_pfa(l4_exc_regs_t const *u) L4_NOTHROW
00248 L4_PURE;
00248
00249
00260 L4_INLINE int l4_utcb_exc_is_ex_regs_exception(
00261 l4_exc_regs_t const *u) L4_NOTHROW L4_PURE;
00261
00266 L4_INLINE void l4_utcb_inherit_fpu(int switch_on) L4_NOTHROW;
00267
00271 L4_INLINE void l4_utcb_inherit_fpu_u(l4_utcb_t *u, int switch_on)
00272 L4_NOTHROW;
00272
00287 L4_INLINE

```

```

00288 l4_timeout_s l4_timeout_abs_u(l4_kernel_clock_t pint, int br,
00289 l4_utcb_t *utcb) L4_NOTHROW;
00303 L4_INLINE
00304 l4_timeout_s l4_timeout_abs(l4_kernel_clock_t pint, int br)
00305 L4_NOTHROW;
00313 L4_INLINE
00314 unsigned l4_utcb_mr64_idx(unsigned idx) L4_NOTHROW;
00315
00316 /*****
00317 * Implementations
00318 *****/
00319
00320 L4_INLINE l4_msg_regs_t *l4_utcb_mr_u(l4_utcb_t *u) L4_NOTHROW
00321 { return (l4_msg_regs_t*)((char*)u + L4_UTCB_MSG_REGS_OFFSET); }
00322
00323 L4_INLINE l4_buf_regs_t *l4_utcb_br_u(l4_utcb_t *u) L4_NOTHROW
00324 { return (l4_buf_regs_t*)((char*)u + L4_UTCB_BUF_REGS_OFFSET); }
00325
00326 L4_INLINE l4_thread_regs_t *l4_utcb_tcr_u(l4_utcb_t *u) L4_NOTHROW
00327 { return (l4_thread_regs_t*)((char*)u +
00328 L4_UTCB_THREAD_REGS_OFFSET); }
00328
00329 L4_INLINE l4_exc_regs_t *l4_utcb_exc_u(l4_utcb_t *u) L4_NOTHROW
00330 { return (l4_exc_regs_t*)((char*)u + L4_UTCB_MSG_REGS_OFFSET); }
00331
00332 L4_INLINE void l4_utcb_inherit_fpu_u(l4_utcb_t *u, int switch_on) L4_NOTHROW
00333 {
00334 if (switch_on)
00335 l4_utcb_br_u(u)->bdr |= L4_UTCB_INHERIT_FPU;
00336 else
00337 l4_utcb_br_u(u)->bdr &= ~L4_UTCB_INHERIT_FPU;
00338 }
00339
00340 L4_INLINE l4_utcb_t *l4_utcb(void) L4_NOTHROW
00341 {
00342 #ifdef L4SYS_USE_UTCB_WRAP
00343 return l4_utcb_wrap();
00344 #else
00345 return l4_utcb_direct();
00346 #endif
00347 }
00348
00349
00350
00351
00352 L4_INLINE l4_msg_regs_t *l4_utcb_mr(void) L4_NOTHROW
00353 { return l4_utcb_mr_u(l4_utcb()); }
00354
00355 L4_INLINE l4_buf_regs_t *l4_utcb_br(void) L4_NOTHROW
00356 { return l4_utcb_br_u(l4_utcb()); }
00357
00358 L4_INLINE l4_thread_regs_t *l4_utcb_tcr(void) L4_NOTHROW
00359 { return l4_utcb_tcr_u(l4_utcb()); }
00360
00361 L4_INLINE l4_exc_regs_t *l4_utcb_exc(void) L4_NOTHROW
00362 { return l4_utcb_exc_u(l4_utcb()); }
00363
00364 L4_INLINE void l4_utcb_inherit_fpu(int switch_on) L4_NOTHROW
00365 { l4_utcb_inherit_fpu_u(l4_utcb(), switch_on); }
00366
00367 L4_INLINE
00368 l4_timeout_s l4_timeout_abs_u(l4_kernel_clock_t val, int pos,
00369 l4_utcb_t *utcb) L4_NOTHROW
00370 {
00371 union T
00372 {
00373 l4_kernel_clock_t t;
00374 l4_umword_t m[sizeof(l4_kernel_clock_t)/sizeof(
00375 l4_umword_t)];
00376 };
00377 l4_timeout_s to;
00378 to.t = 0x8000 | pos;
00379 ((union T*)(l4_utcb_br_u(utcb)->br + pos))->t = val;
00380 return to;
00381 }
00382 L4_INLINE
00383 l4_timeout_s l4_timeout_abs(l4_kernel_clock_t val, int pos)
00384 L4_NOTHROW
00385 { return l4_timeout_abs_u(val, pos, l4_utcb()); }
00386
00387 L4_INLINE unsigned l4_utcb_mr64_idx(unsigned idx) L4_NOTHROW
00388 { return idx / (sizeof(l4_uint64_t) / sizeof(l4_umword_t)); }
00389
00390 __END_DECLS

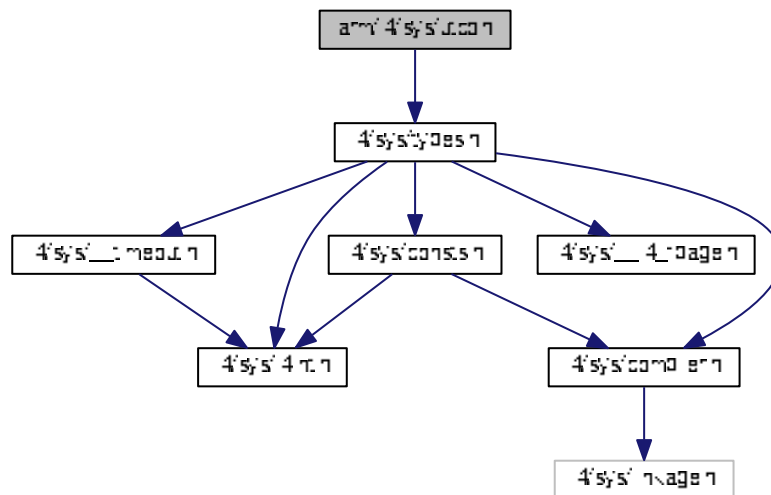
```

```
00391 #endif /* ! _L4_SYS_UTCB_H */
```

## 15.359 arm/l4/sys/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.*

### Typedefs

- typedef struct [l4\\_exc\\_regs\\_t](#) [l4\\_exc\\_regs\\_t](#)  
*UTCB structure for exceptions.*

### Enumerations

- enum [L4\\_utcb\\_consts\\_arm](#)  
*UTCB constants for ARM.*



## 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()`.

### 15.359.1 Detailed Description

UTCB definitions for ARM.

Definition in file `utcb.h`.

## 15.360 utcb.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 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024 #ifndef __L4_SYS__INCLUDE__ARCH_ARM__UTCB_H__
00025 #define __L4_SYS__INCLUDE__ARCH_ARM__UTCB_H__
00026
00027 #include <l4/sys/types.h>
00028
00038 typedef struct l4_exc_regs_t
00039 {
00040 l4_umword_t pfa;
00041 l4_umword_t err;
00043 l4_umword_t r[13];
00044 l4_umword_t sp;
00045 l4_umword_t ulr;
00046 l4_umword_t _dummy1;
00047 l4_umword_t pc;
00048 l4_umword_t cpsr;
00049 l4_umword_t tpidruro;
00050 } l4_exc_regs_t;
00051
00057 enum L4_utcb_consts_arm
00058 {
00059 L4_UTCB_EXCEPTION_REGS_SIZE = sizeof(
00060 l4_exc_regs_t) / sizeof(l4_umword_t),
00060 L4_UTCB_GENERIC_DATA_SIZE = 63,
00061 L4_UTCB_GENERIC_BUFFERS_SIZE = 58,
00062

```

```

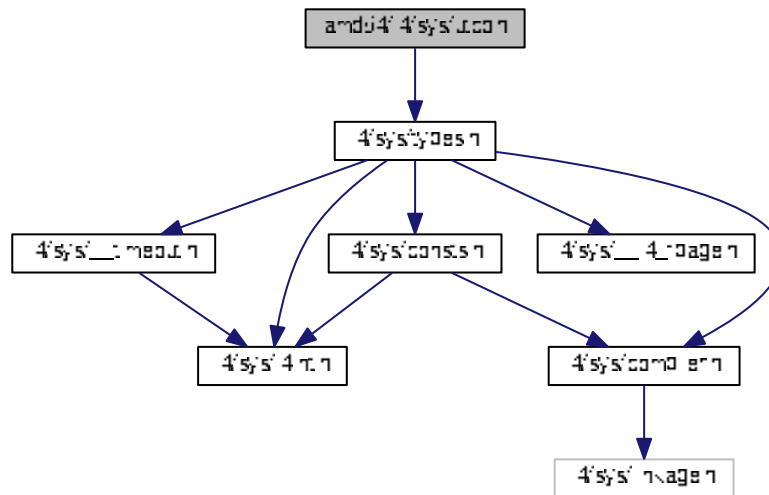
00063 L4_UTCB_MSG_REGS_OFFSET = 0,
00064 L4_UTCB_BUF_REGS_OFFSET = 64 * sizeof(
00065 l4_umword_t),
00066 L4_UTCB_THREAD_REGS_OFFSET = 123 * sizeof(
00067 l4_umword_t),
00068 L4_UTCB_INHERIT_FPU = 1UL << 24,
00069 L4_UTCB_OFFSET = 512,
00070 };
00071
00072 #include_next <l4/sys/utcb.h>
00073
00074 /*
00075 * =====
00076 * Implementations.
00077 */
00078
00079 #ifdef __GNUC__
00080 L4_INLINE l4_utcb_t *l4_utcb_direct(void) L4_NOTHROW
00081 {
00082 register l4_utcb_t *utcb __asm__ ("r0");
00083 __asm__ ("mov lr, pc\n"
00084 "mov pc, #0xffffffff00\n"
00085 : "=r"(utcb) : : "lr");
00086 return utcb;
00087 }
00088 #endif
00089
00090 L4_INLINE l4_umword_t l4_utcb_exc_pc(l4_exc_regs_t const *u)
00091 L4_NOTHROW
00092 {
00093 return u->pc;
00094 }
00095
00096 L4_INLINE void l4_utcb_exc_pc_set(l4_exc_regs_t *u,
00097 l4_addr_t pc) L4_NOTHROW
00098 {
00099 u->pc = pc;
00100 }
00101
00102 L4_INLINE l4_umword_t l4_utcb_exc_typeval(
00103 l4_exc_regs_t const *u) L4_NOTHROW
00104 {
00105 return u->err >> 26;
00106 }
00107
00108 L4_INLINE int l4_utcb_exc_is_pf(l4_exc_regs_t const *u)
00109 L4_NOTHROW
00110 {
00111 return ((u->err >> 26) & 0x30) == 0x20;
00112 }
00113
00114 L4_INLINE l4_addr_t l4_utcb_exc_pfa(l4_exc_regs_t const *u)
00115 L4_NOTHROW
00116 {
00117 return (u->pfa & ~7UL) | ((u->err >> 5) & 2);
00118 }
00119
00120 L4_INLINE int l4_utcb_exc_is_ex_regs_exception(
00121 l4_exc_regs_t const *u) L4_NOTHROW
00122 {
00123 return l4_utcb_exc_typeval(u) == 0x3e;
00124 }
00125
00126 #endif /* ! __L4_SYS__INCLUDE__ARCH_ARM__UTCB_H__ */

```

## 15.361 amd64/l4/sys/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.*

## Typedefs

- typedef struct [l4\\_exc\\_regs\\_t](#) [l4\\_exc\\_regs\\_t](#)  
*UTCB structure for exceptions.*

## Enumerations

- enum [L4\\_utcb\\_consts\\_amd64](#)  
*UTCB constants for AMD64.*

## 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\(\)](#).*

### 15.361.1 Detailed Description

UTCB definitions for amd64.

Definition in file [utcb.h](#).

## 15.362 utcb.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 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024 /*****
00025 * #ifndef __L4_SYS__INCLUDE__ARCH_AMD64__UTCB_H__
00026 * #define __L4_SYS__INCLUDE__ARCH_AMD64__UTCB_H__
00027 *
00028 * #include <l4/sys/types.h>
00029 *
00039 * enum L4_utcb_consts_amd64
00040 * {
00041 * L4_UTCB_EXCEPTION_REGS_SIZE = 26,
00042 * L4_UTCB_GENERIC_DATA_SIZE = 63,
00043 * L4_UTCB_GENERIC_BUFFERS_SIZE = 58,
00044 *
00045 * L4_UTCB_MSG_REGS_OFFSET = 0,
00046 * L4_UTCB_BUF_REGS_OFFSET = 64 * sizeof(
00047 * l4_umword_t),
00048 * L4_UTCB_THREAD_REGS_OFFSET = 123 * sizeof(
00049 * l4_umword_t),
00050 *
00051 * L4_UTCB_INHERIT_FPU = 1UL << 24,
00052 * L4_UTCB_OFFSET = 1024,
00053 * };
00054 *
00055 * typedef struct l4_exc_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 * l4_umword_t rdi;
00066 * l4_umword_t rsi;
00067 * l4_umword_t rbp;
00068 * l4_umword_t pfa;
00069 * l4_umword_t rbx;
00070 * l4_umword_t rdx;
00071 * l4_umword_t rcx;
00072 * l4_umword_t rax;
00073 * l4_umword_t trapno;
00074 * l4_umword_t err;
00075 * l4_umword_t ip;
00076 * l4_umword_t dummy1;
00077 * l4_umword_t flags;
00078 * l4_umword_t sp;
00079 * l4_umword_t ss;
00080 * l4_umword_t fs_base;
00081 * l4_umword_t gs_base;
00082 * l4_uint16_t ds, es, fs, gs;
00083 * } l4_exc_regs_t;
00084 */
00085

```

```

00088
00089 #include_next <l4/sys/utcb.h>
00090
00091 /*
00092 * =====
00093 * Implementations.
00094 */
00095
00096 L4_INLINE l4_utcb_t *l4_utcb_direct(void) L4_NOTHROW
00097 {
00098 l4_utcb_t *res;
00099 __asm__ ("mov %%gs:0, %0 \n" : "=r"(res));
00100 return res;
00101 }
00102
00103 L4_INLINE l4_umword_t l4_utcb_exc_pc(l4_exc_regs_t const *u)
00104 L4_NOTHROW
00105 {
00106 return u->ip;
00107 }
00108 L4_INLINE void l4_utcb_exc_pc_set(l4_exc_regs_t *u,
00109 l4_addr_t pc) L4_NOTHROW
00110 {
00111 u->ip = pc;
00112 }
00113 L4_INLINE l4_umword_t l4_utcb_exc_typeval(
00114 l4_exc_regs_t const *u) L4_NOTHROW
00115 {
00116 return u->trapno;
00117 }
00118 L4_INLINE int l4_utcb_exc_is_pf(l4_exc_regs_t const *u)
00119 L4_NOTHROW
00120 {
00121 return u->trapno == 14;
00122 }
00123 L4_INLINE l4_addr_t l4_utcb_exc_pfa(l4_exc_regs_t const *u)
00124 L4_NOTHROW
00125 {
00126 return (u->pfa & ~3UL) | (u->err & 2);
00127 }
00128 L4_INLINE int l4_utcb_exc_is_ex_regs_exception(
00129 l4_exc_regs_t const *u) L4_NOTHROW
00130 {
00131 return l4_utcb_exc_typeval(u) == 0xff;
00132 }
00133 #endif /* ! __L4_SYS__INCLUDE__ARCH_AMD64__UTCB_H__ */

```

## 15.363 x86/I4/sys/utcb.h File Reference

UTCB definitions for X86.



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()`.

### 15.363.1 Detailed Description

UTCB definitions for X86.

Definition in file [utcb.h](#).

## 15.364 utcb.h

```

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 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025 /*****
00026 #ifndef __L4_SYS__INCLUDE__ARCH_X86__UTCB_H__
00027 #define __L4_SYS__INCLUDE__ARCH_X86__UTCB_H__
00028
00029 #include <l4/sys/types.h>
00030
00041 enum l4_utcb_consts_x86
00042 {
00044 L4_UTCB_EXCEPTION_REGS_SIZE = 19,
00045
00047 L4_UTCB_GENERIC_DATA_SIZE = 63,
00048
00050 L4_UTCB_GENERIC_BUFFERS_SIZE = 58,
00051
00053 L4_UTCB_MSG_REGS_OFFSET = 0,
00054
00056 L4_UTCB_BUF_REGS_OFFSET = 64 * sizeof(
00057 l4_umword_t),
00059 L4_UTCB_THREAD_REGS_OFFSET = 123 * sizeof(
00060 l4_umword_t),
00062 L4_UTCB_INHERIT_FPU = 1UL << 24,
00063
00065 L4_UTCB_OFFSET = 512,
00066 };
00067
00072 typedef struct l4_exc_regs_t
00073 {
00074 l4_umword_t es;
00075 l4_umword_t ds;
00076 l4_umword_t gs;
00077 l4_umword_t fs;
00079 l4_umword_t edi;
00080 l4_umword_t esi;
00081 l4_umword_t ebp;

```

```

00082 l4_umword_t pfa;
00083 l4_umword_t ebx;
00084 l4_umword_t edx;
00085 l4_umword_t ecx;
00086 l4_umword_t eax;
00088 l4_umword_t trapno;
00089 l4_umword_t err;
00091 l4_umword_t ip;
00092 l4_umword_t dummy1;
00093 l4_umword_t flags;
00094 l4_umword_t sp;
00095 l4_umword_t ss;
00096 } l4_exc_regs_t;
00097
00098 #include_next <l4/sys/utcb.h>
00099
00100 /*
00101 * =====
00102 * Implementations.
00103 */
00104
00105 L4_INLINE l4_utcb_t *l4_utcb_direct(void) L4_NOTHROW
00106 {
00107 l4_utcb_t *utcb;
00108 __asm__ ("mov %%fs:0, %0" : "=r" (utcb));
00109 return utcb;
00110 }
00111
00112 L4_INLINE l4_umword_t l4_utcb_exc_pc(l4_exc_regs_t const *u)
00113 L4_NOTHROW
00114 {
00115 return u->ip;
00116 }
00117 L4_INLINE void l4_utcb_exc_pc_set(l4_exc_regs_t *u,
00118 l4_addr_t pc) L4_NOTHROW
00119 {
00120 u->ip = pc;
00121 }
00122 L4_INLINE void l4_utcb_exc_sp_set(l4_exc_regs_t *u, l4_addr_t
00123 sp) L4_NOTHROW
00124 {
00125 u->sp = sp;
00126 }
00127 L4_INLINE l4_umword_t l4_utcb_exc_typeval(
00128 l4_exc_regs_t const *u) L4_NOTHROW
00129 {
00130 return u->trapno;
00131 }
00132 L4_INLINE int l4_utcb_exc_is_pf(l4_exc_regs_t const *u)
00133 L4_NOTHROW
00134 {
00135 return u->trapno == 14;
00136 }
00137 L4_INLINE l4_addr_t l4_utcb_exc_pfa(l4_exc_regs_t const *u)
00138 L4_NOTHROW
00139 {
00140 return (u->pfa & ~3UL) | (u->err & 2);
00141 }
00142 L4_INLINE int l4_utcb_exc_is_ex_regs_exception(
00143 l4_exc_regs_t const *u) L4_NOTHROW
00144 {
00145 return l4_utcb_exc_typeval(u) == 0xff;
00146 }
00147 #endif /* ! __L4_SYS__INCLUDE__ARCH_X86__UTCB_H__ */

```

## 15.365 l4/sys/vcon File Reference

Virtual console interface.

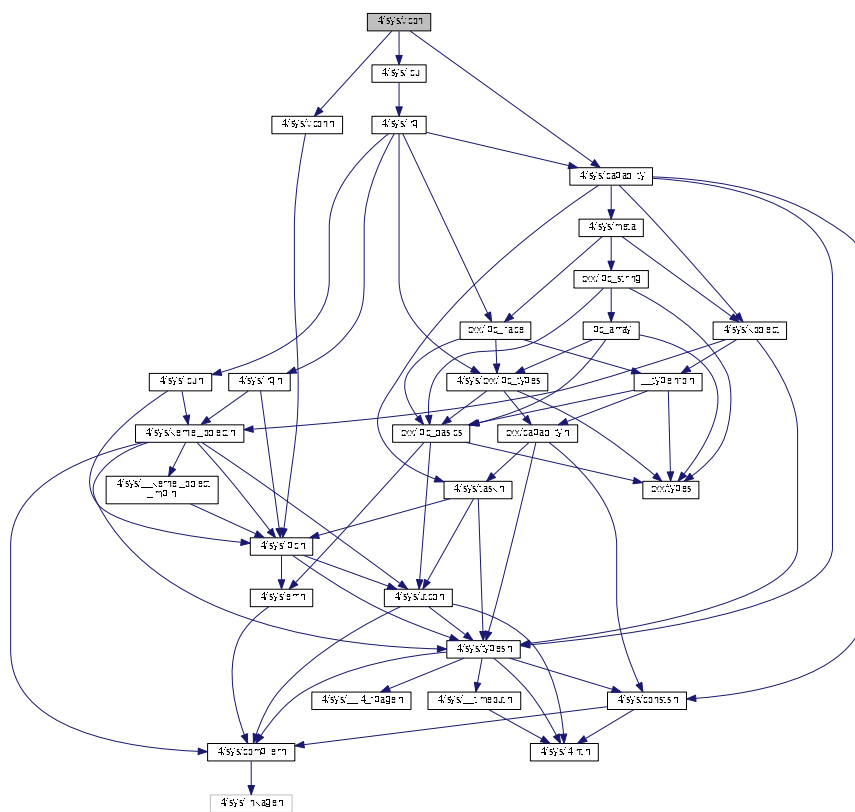
```

#include <l4/sys/icu>
#include <l4/sys/vcon.h>

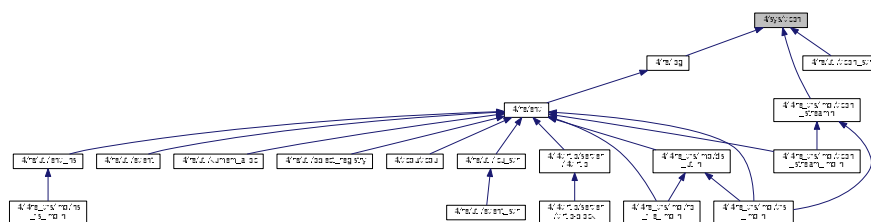
```



```
#include <l4/sys/capability>
Include dependency graph for vcon:
```



This graph shows which files directly or indirectly include this file:



## Data Structures

- class `L4::Vcon`  
*C++ L4 Vcon interface.*

## Namespaces

- L4  
*L4 low-level kernel interface.*

### 15.365.1 Detailed Description

Virtual console interface.

Definition in file [vcon](#).

## 15.366 vcon

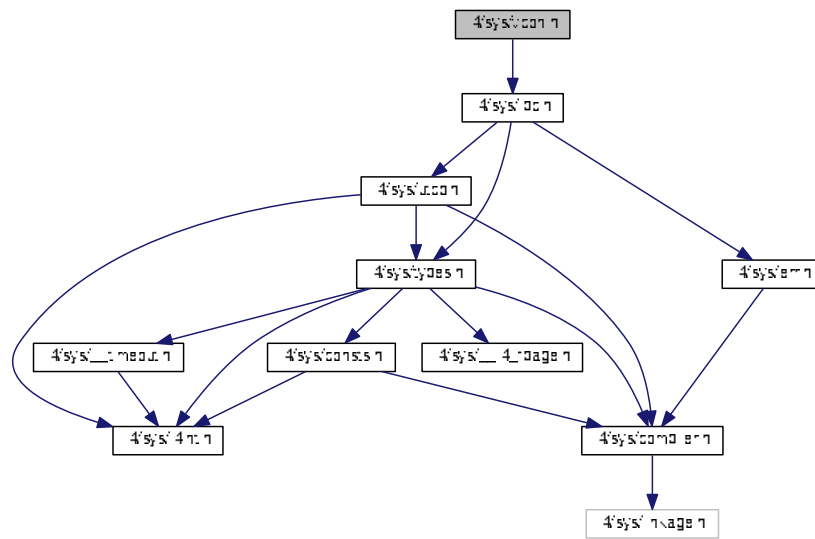
```

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 * This file is part of TUD:OS and distributed under the terms of the
00009 * GNU General Public License 2.
00010 * Please see the COPYING-GPL-2 file for details.
00011 *
00012 * As a special exception, you may use this file as part of a free software
00013 * library without restriction. Specifically, if other files instantiate
00014 * templates or use macros or inline functions from this file, or you compile
00015 * this file and link it with other files to produce an executable, this
00016 * file does not by itself cause the resulting executable to be covered by
00017 * the GNU General Public License. This exception does not however
00018 * invalidate any other reasons why the executable file might be covered by
00019 * the GNU General Public License.
00020 */
00021 #pragma once
00022
00023 #include <l4/sys/icu>
00024 #include <l4/sys/vcon.h>
00025 #include <l4/sys/capability>
00026
00027 namespace L4 {
00028
00029 class Vcon :
00030 public Kobject_t<Vcon, Icu, L4_PROTO_LOG>
00031 {
00032 public:
00033 l4_msgtag_t
00034 send(char const *buf, unsigned size, l4_utcb_t *utcb = l4_utcb()) const throw()
00035 { return l4_vcon_send_u(cap(), buf, size, utcb); }
00036
00037 long
00038 write(char const *buf, unsigned size, l4_utcb_t *utcb = l4_utcb()) const throw()
00039 { return l4_vcon_write_u(cap(), buf, size, utcb); }
00040
00041 int
00042 read(char *buf, unsigned size, l4_utcb_t *utcb = l4_utcb()) const throw()
00043 { return l4_vcon_read_u(cap(), buf, size, utcb); }
00044
00045 int
00046 read_with_flags(char *buf, unsigned size, l4_utcb_t *utcb =
00047 l4_utcb()) const throw()
00048 { return l4_vcon_read_with_flags_u(cap(), buf, size, utcb); }
00049
00050 l4_msgtag_t
00051 set_attr(l4_vcon_attr_t const *attr, l4_utcb_t *utcb =
00052 l4_utcb()) const throw()
00053 { return l4_vcon_set_attr_u(cap(), attr, utcb); }
00054
00055 l4_msgtag_t
00056 get_attr(l4_vcon_attr_t *attr, l4_utcb_t *utcb =
00057 l4_utcb()) const throw()
00058 { return l4_vcon_get_attr_u(cap(), attr, utcb); }
00059
00060 typedef L4::Typeid::Raw_ipc<Vcon> Rpcs;
00061 };
00062
00063 }
```

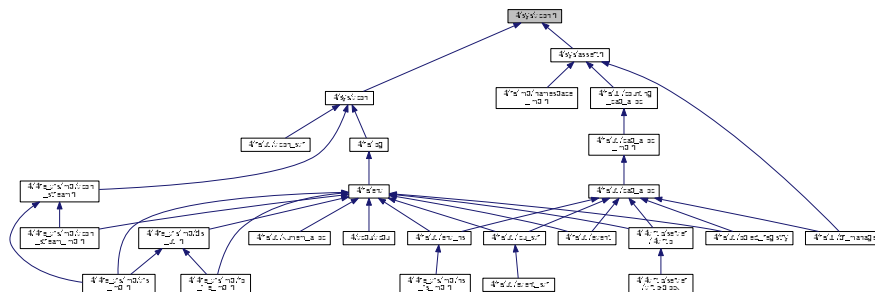
## 15.367 l4/sys/vcon.h File Reference

Virtual console interface.

```
#include <linux/sys/ipc.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.

### 15.367.1 Detailed Description

Virtual console interface.

Definition in file [vcon.h](#).

## 15.367.2 Enumeration Type Documentation

### 15.367.2.1 L4\_vcon\_read\_flags

enum [L4\\_vcon\\_read\\_flags](#)

Vcon read flags.

#### Enumerator

|                                         |                       |
|-----------------------------------------|-----------------------|
| <a href="#">L4_VCON_READ_SIZE_MASK</a>  | Size mask.            |
| <a href="#">L4_VCON_READ_STAT_BREAK</a> | Break condition flag. |
| <a href="#">L4_VCON_READ_STAT_DONE</a>  | Done condition flag.  |

Definition at line [167](#) of file [vcon.h](#).

## 15.368 vcon.h

```

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 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024 #pragma once
00025
00026 #include <l4/sys/ipc.h>
00027
00056 L4_INLINE l4_msgtag_t
00057 l4_vcon_send(l4_cap_idx_t vcon, char const *buf, unsigned size)
00058 L4_NOTHROW;
00059
00065 L4_INLINE l4_msgtag_t
00066 l4_vcon_send_u(l4_cap_idx_t vcon, char const *buf, unsigned size,
00067 l4_utcb_t *utcb) L4_NOTHROW;
00068
00079 L4_INLINE long
00080 l4_vcon_write(l4_cap_idx_t vcon, char const *buf, unsigned size)
00081 L4_NOTHROW;
00082
00088 L4_INLINE long
00089 l4_vcon_write_u(l4_cap_idx_t vcon, char const *buf, unsigned size,
00090 l4_utcb_t *utcb) L4_NOTHROW;
00091
00095 enum L4_vcon_size_consts
00096 {
00098 L4_VCON_WRITE_SIZE = (L4_UTCB_GENERIC_DATA_SIZE - 2) * sizeof(
00099 l4_umword_t),
00100 L4_VCON_READ_SIZE = (L4_UTCB_GENERIC_DATA_SIZE - 1) * sizeof(
00101 l4_umword_t),
00102 };

```

```

00102
00118 L4_INLINE int
00119 l4_vcon_read(l4_cap_idx_t vcon, char *buf, unsigned size)
 L4_NOTHROW;
00120
00127 L4_INLINE int
00128 l4_vcon_read_u(l4_cap_idx_t vcon, char *buf, unsigned size,
 l4_utcb_t *utcb) L4_NOTHROW;
00129
00154 L4_INLINE int
00155 l4_vcon_read_with_flags(l4_cap_idx_t vcon, char *buf, unsigned size)
 L4_NOTHROW;
00156
00160 L4_INLINE int
00161 l4_vcon_read_with_flags_u(l4_cap_idx_t vcon, char *buf, unsigned size,
 l4_utcb_t *utcb) L4_NOTHROW;
00162
00163
00167 enum L4_vcon_read_flags
00168 {
00169 L4_VCON_READ_SIZE_MASK = 0x3fffffff,
00170 L4_VCON_READ_STAT_BREAK = 1 << 30,
00171 L4_VCON_READ_STAT_DONE = 1 << 31,
00172 };
00173
00178 typedef struct l4_vcon_attr_t
00179 {
00180 l4_umword_t i_flags;
00181 l4_umword_t o_flags;
00182 l4_umword_t l_flags;
00183 } l4_vcon_attr_t;
00184
00189 enum L4_vcon_i_flags
00190 {
00191 L4_VCON_INLCR = 000100,
00192 L4_VCON_IGNCR = 000200,
00193 L4_VCON_ICRNL = 000400,
00194 };
00195
00200 enum L4_vcon_o_flags
00201 {
00202 L4_VCON_ONLCR = 000004,
00203 L4_VCON_OCRNL = 000010,
00204 L4_VCON_ONLRET = 000040,
00205 };
00206
00211 enum L4_vcon_l_flags
00212 {
00213 L4_VCON_ICANON = 000002,
00214 L4_VCON_ECHO = 000010,
00215 };
00216
00225 L4_INLINE l4_msgtag_t
00226 l4_vcon_set_attr(l4_cap_idx_t vcon, l4_vcon_attr_t const *attr)
 L4_NOTHROW;
00227
00234 L4_INLINE l4_msgtag_t
00235 l4_vcon_set_attr_u(l4_cap_idx_t vcon,
 l4_vcon_attr_t const *attr,
00236 l4_utcb_t *utcb) L4_NOTHROW;
00237
00246 L4_INLINE l4_msgtag_t
00247 l4_vcon_get_attr(l4_cap_idx_t vcon, l4_vcon_attr_t *attr)
 L4_NOTHROW;
00248
00255 L4_INLINE l4_msgtag_t
00256 l4_vcon_get_attr_u(l4_cap_idx_t vcon,
 l4_vcon_attr_t *attr,
00257 l4_utcb_t *utcb) L4_NOTHROW;
00258
00259
00264 enum L4_vcon_ops
00265 {
00266 L4_VCON_WRITE_OP = 0UL,
00267 L4_VCON_READ_OP = 1UL,
00268 L4_VCON_SET_ATTR_OP = 2UL,
00269 L4_VCON_GET_ATTR_OP = 3UL,
00270 };
00271
00272 /***** Implementations *****/
00273
00274 L4_INLINE l4_msgtag_t
00275 l4_vcon_send_u(l4_cap_idx_t vcon, char const *buf, unsigned size,
 l4_utcb_t *utcb) L4_NOTHROW
00276 {
00277 l4_msg_regs_t *mr = l4_utcb_mr_u(utcb);
00278 mr->mr[0] = L4_VCON_WRITE_OP;
00279 mr->mr[1] = size;

```

```

00280 __builtin_memcpy(&mr->mr[2], buf, size);
00281 return l4_ipc_send(vcon, utcb,
00282 l4_msgtag(L4_PROTO_LOG,
00283 2 + (size + sizeof(l4_umword_t) - 1) / sizeof(
00284 l4_umword_t),
00285 0, L4_MSGTAG_SCHEDULE),
00286);
00287
00288 L4_INLINE l4_msgtag_t
00289 l4_vcon_send(l4_cap_idx_t vcon, char const *buf, unsigned size) L4_NOTHROW
00290 {
00291 return l4_vcon_send_u(vcon, buf, size, l4_utcb());
00292 }
00293
00294 L4_INLINE long
00295 l4_vcon_write_u(l4_cap_idx_t vcon, char const *buf, unsigned size,
00296 l4_utcb_t *utcb) L4_NOTHROW
00297 {
00298 l4_msgtag_t t;
00299 if (size > L4_VCON_WRITE_SIZE)
00300 size = L4_VCON_WRITE_SIZE;
00301 t = l4_vcon_send_u(vcon, buf, size, utcb);
00302 if (l4_msgtag_has_error(t))
00303 return l4_error(t);
00304 return (long) size;
00305 }
00306
00307 L4_INLINE long
00308 l4_vcon_write(l4_cap_idx_t vcon, char const *buf, unsigned size) L4_NOTHROW
00309 {
00310 return l4_vcon_write_u(vcon, buf, size, l4_utcb());
00311 }
00312
00313 L4_INLINE int
00314 l4_vcon_read_with_flags_u(l4_cap_idx_t vcon, char *buf, unsigned size,
00315 l4_utcb_t *utcb) L4_NOTHROW
00316 {
00317 int ret;
00318 unsigned r;
00319 l4_msg_regs_t *mr;
00320 mr = l4_utcb_mr_u(utcb);
00321 mr->mr[0] = (size << 16) | L4_VCON_READ_OP;
00322 ret = l4_error_u(l4_ipc_call(vcon, utcb,
00323 l4_msgtag(L4_PROTO_LOG, 1, 0, 0),
00324 L4_IPC_NEVER),
00325 utcb);
00326 if (ret < 0)
00327 return ret;
00328 r = mr->mr[0] & L4_VCON_READ_SIZE_MASK;
00329 if (!(mr->mr[0] & L4_VCON_READ_STAT_DONE)) // !eof
00330 ret = size + 1;
00331 else if (r < size)
00332 ret = r;
00333 else
00334 ret = size;
00335 if (L4_LIKELY(buf != NULL))
00336 __builtin_memcpy(buf, &mr->mr[1], r < size ? r : size);
00337 return ret | (mr->mr[0] & ~(L4_VCON_READ_STAT_DONE |
00338 L4_VCON_READ_SIZE_MASK));
00339 }
00340
00341 L4_INLINE int
00342 l4_vcon_read_with_flags(l4_cap_idx_t vcon, char *buf, unsigned size)
00343 L4_NOTHROW
00344 {
00345 return l4_vcon_read_with_flags_u(vcon, buf, size, l4_utcb());
00346 }
00347
00348 L4_INLINE int
00349 l4_vcon_read_u(l4_cap_idx_t vcon, char *buf, unsigned size,
00350 l4_utcb_t *utcb) L4_NOTHROW
00351 {
00352 int r = l4_vcon_read_with_flags_u(vcon, buf, size, utcb);
00353 if (r < 0)
00354 return r;
00355 return r & L4_VCON_READ_SIZE_MASK;
00356 }

```

```

00362 }
00363
00364 L4_INLINE int
00365 l4_vcon_read(l4_cap_idx_t vcon, char *buf, unsigned size) L4_NOTHROW
00366 {
00367 return l4_vcon_read_u(vcon, buf, size, l4_utcb());
00368 }
00369
00370 L4_INLINE l4_msgtag_t
00371 l4_vcon_set_attr_u(l4_cap_idx_t vcon,
00372 l4_vcon_attr_t const *attr,
00373 l4_utcb_t *utcb) L4_NOTHROW
00374 {
00375 l4_msg_regs_t *mr = l4_utcb_mr_u(utcb);
00376 mr->mr[0] = L4_VCON_SET_ATTR_OP;
00377 __builtin_memcpy(&mr->mr[1], attr, sizeof(*attr));
00378
00379 return l4_ipc_call(vcon, utcb,
00380 l4_msgtag(L4_PROTO_LOG, 4, 0, 0),
00381 L4_IPC_NEVER);
00382 }
00383
00384 L4_INLINE l4_msgtag_t
00385 l4_vcon_set_attr(l4_cap_idx_t vcon, l4_vcon_attr_t const *attr)
00386 L4_NOTHROW
00387 {
00388 return l4_vcon_set_attr_u(vcon, attr, l4_utcb());
00389 }
00390 L4_INLINE l4_msgtag_t
00391 l4_vcon_get_attr_u(l4_cap_idx_t vcon,
00392 l4_vcon_attr_t *attr,
00393 l4_utcb_t *utcb) L4_NOTHROW
00394 {
00395 l4_msgtag_t res;
00396 l4_msg_regs_t *mr = l4_utcb_mr_u(utcb);
00397 mr->mr[0] = L4_VCON_GET_ATTR_OP;
00398
00399 res = l4_ipc_call(vcon, utcb,
00400 l4_msgtag(L4_PROTO_LOG, 1, 0, 0),
00401 L4_IPC_NEVER);
00402 if (l4_error_u(res, utcb) >= 0)
00403 __builtin_memcpy(attr, &mr->mr[1], sizeof(*attr));
00404
00405 return res;
00406 }
00407
00408 L4_INLINE l4_msgtag_t
00409 l4_vcon_get_attr(l4_cap_idx_t vcon, l4_vcon_attr_t *attr)
00410 L4_NOTHROW
00411 {
00412 return l4_vcon_get_attr_u(vcon, attr, l4_utcb());
00413 }

```

## 15.369 l4/sys/vhw.h File Reference

Descriptors for virtual hardware (under UX).

```

#include <l4/sys/types.h>
#include <l4/sys/kip.h>

```





## 15.370 vhw.h

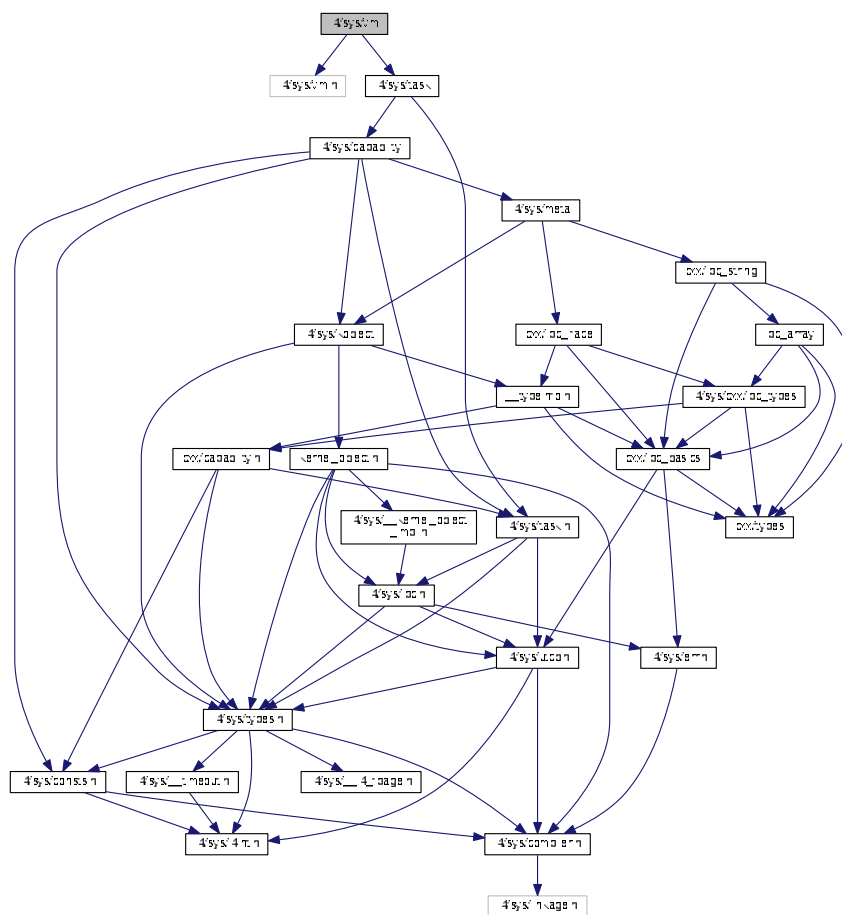
```

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 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */
00025 /*****
00026 #ifndef _L4_SYS_VHW_H
00027 #define _L4_SYS_VHW_H
00028
00029 #include <l4/sys/types.h>
00030 #include <l4/sys/kip.h>
00031
00044 enum l4_vhw_entry_type {
00045 L4_TYPE_VHW_NONE,
00046 L4_TYPE_VHW_FRAMEBUFFER,
00047 L4_TYPE_VHW_INPUT,
00048 L4_TYPE_VHW_NET,
00049 };
00050
00055 struct l4_vhw_entry {
00056 enum l4_vhw_entry_type type;
00057 l4_uint32_t provider_pid;
00059 l4_addr_t mem_start;
00060 l4_addr_t mem_size;
00062 l4_uint32_t irq_no;
00063 l4_uint32_t fd;
00064 };
00065
00070 struct l4_vhw_descriptor {
00071 l4_uint32_t magic;
00072 l4_uint8_t version;
00073 l4_uint8_t count;
00074 l4_uint8_t pad1;
00075 l4_uint8_t pad2;
00077 struct l4_vhw_entry descs[];
00078 };
00079
00080 enum {
00081 L4_VHW_MAGIC = 0x56687765,
00082 };
00083
00084 static inline struct l4_vhw_descriptor *
00085 l4_vhw_get(l4_kernel_info_t *kip) L4_NOTHROW
00086 {
00087 struct l4_vhw_descriptor *v
00088 = (struct l4_vhw_descriptor *)(((unsigned long)kip) + kip->vhw_offset);
00089
00090 if (v->magic == L4_VHW_MAGIC)
00091 return v;
00092
00093 return NULL;
00094 }
00095
00096 static inline struct l4_vhw_entry *
00097 l4_vhw_get_entry(struct l4_vhw_descriptor *v, int entry)
00098 L4_NOTHROW
00099 {
00100 return v->descs + entry;
00101 }
00102
00103 static inline struct l4_vhw_entry *
00104 l4_vhw_get_entry_type(struct l4_vhw_descriptor *v, enum
00105 l4_vhw_entry_type t) L4_NOTHROW
00106 {
00107 int i;
00108 struct l4_vhw_entry *e = v->descs;
00109
00110 for (i = 0; i < v->count; i++, e++)
00111 if (e->type == t)
00112 return e;

```

## 15.371 I4/sys/vm File Reference

```
#include <14/sys/vm.h>
#include <14/sys/task>
Include dependency graph for vm:
```



- class `L4::Vm`  
*Virtual machine.*

- L4  
*L4 low-level kernel interface.*

### 15.371.1 Detailed Description

Virtualization interface.

Definition in file [vm](#).

## 15.372 vm

```

00001 // vi:set ft=cpp: -*- Mode: C++ -*-
00006 /*
00007 * (c) 2008-2010 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 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024
00025 #pragma once
00026
00027 #include <l4/sys/vm.h>
00028 #include <l4/sys/task>
00029
00030 namespace L4 {
00031
00035 class Vm : public Kobject_t<Vm, Task, L4_PROTO_VM>
00036 {
00037 protected:
00038 Vm();
00039
00040 private:
00041 Vm(Vm const &);
00042 void operator = (Vm const &);
00043 };
00044
00045 };

```

## 15.373 l4/util/alloc.h File Reference

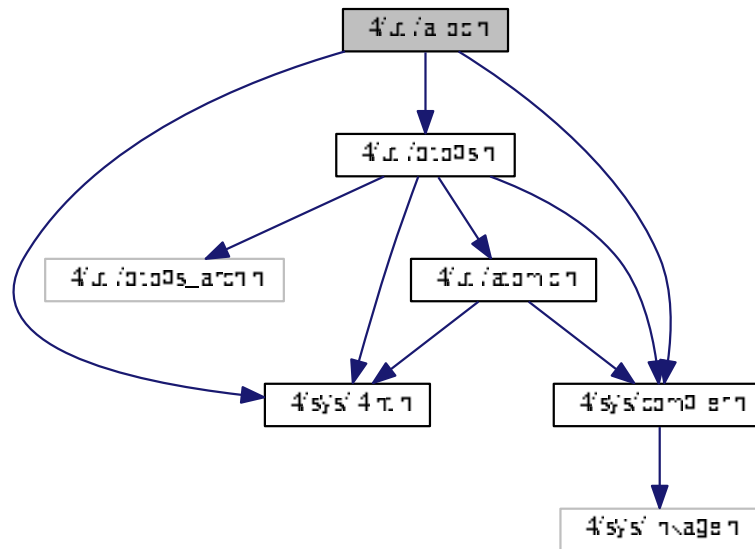
Allocator using a bit-array.

```

#include <l4/sys/l4int.h>
#include <l4/util/bitops.h>
#include <l4/sys/compiler.h>

```

Include dependency graph for alloc.h:



### 15.373.1 Detailed Description

Allocator using a bit-array.

Date

09/14/2004

Author

Jork Loeser [jork.loeser@inf.tu-dresden.de](mailto:jork.loeser@inf.tu-dresden.de)

Definition in file [alloc.h](#).

## 15.374 alloc.h

```

00001
00009 /*
00010 * (c) 2004-2009 Author(s)
00011 * economic rights: Technische Universität Dresden (Germany)
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU Lesser General Public License 2.1.
00014 * Please see the COPYING-LGPL-2.1 file for details.
00015 */
00016 #ifndef __UTIL_INCLUDE_ALLOC_H_
00017 #define __UTIL_INCLUDE_ALLOC_H_
00018 #include <l4/sys/l4int.h>
00019 #include <l4/util/bitops.h>
00020 #include <l4/sys/compiler.h>
00021
00022 EXTERN_C_BEGIN

```

```

00023
00024 typedef struct {
00025 int base, count, next_elem;
00026 l4_umword_t *bits;
00027 } l4util_alloc_t;
00028
00029 #define L4UTIL_ALLOC_BITS_SIZE (8 * sizeof(l4_umword_t))
00030
00031 L4_CV l4util_alloc_t *l4util_alloc_init(int count, int base);
00032 L4_CV int l4util_alloc_avail(l4util_alloc_t *alloc, int elem);
00033 L4_CV int l4util_alloc_occupy(l4util_alloc_t *alloc, int elem);
00034 L4_CV int l4util_alloc_alloc(l4util_alloc_t *alloc);
00035 L4_CV int l4util_alloc_free(l4util_alloc_t *alloc, int elem);
00036
00037 EXTERN_C_END
00038 #endif

```

## 15.375 l4/cxx/alloc.h File Reference

Alloc list.

### Data Structures

- class [L4::Alloc\\_list](#)  
*A simple list-based allocator.*

### Namespaces

- [L4](#)  
*L4 low-level kernel interface.*

## 15.375.1 Detailed Description

Alloc list.

Definition in file [alloc.h](#).

## 15.376 alloc.h

```

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 * 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 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
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_ALLOC_H__

```

```

00025 #define L4_CXX_ALLOC_H__
00026
00027 namespace L4 {
00028
00033 class Alloc_list
00034 {
00035 public:
00036 Alloc_list() : _free(0) {}
00037 Alloc_list(void *blk, unsigned long size) : _free(0)
00038 { free(blk, size); }
00039
00040 void free(void *blk, unsigned long size);
00041 void *alloc(unsigned long size);
00042
00043 private:
00044 struct Elem
00045 {
00046 Elem *next;
00047 unsigned long size;
00048 };
00049
00050 Elem *_free;
00051 };
00052 };
00053
00054 #endif // L4_CXX_ALLOC_H__
00055

```

## 15.377 l4/util/assert.h File Reference

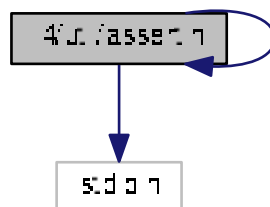
Some useful assert-style macros.

```

#include <stdio.h>
#include <assert.h>

```

Include dependency graph for assert.h:



This graph shows which files directly or indirectly include this file:



### 15.377.1 Detailed Description

Some useful assert-style macros.

#### Date

09/2009

#### Author

Bjoern Doebel [doebel@tudos.org](mailto:doebel@tudos.org)

Definition in file [assert.h](#).

## 15.378 assert.h

```

00001 /*****
00009 */
00010 * (c) 2009 Author(s)
00011 * economic rights: Technische Universität Dresden (Germany)
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU Lesser General Public License 2.1.
00014 * Please see the COPYING-LGPL-2.1 file for details.
00015 */
00016
00017 /*****
00018 #pragma once
00019
00020 #ifndef NDEBUG
00021
00022 #define DO_NOTHING do {} while (0)
00023 #define ASSERT_VALID(c) DO_NOTHING
00024 #define ASSERT_EQUAL(a,b) DO_NOTHING
00025 #define ASSERT_NOT_EQUAL(a,b) DO_NOTHING
00026 #define ASSERT_LOWER_EQ(a,b) DO_NOTHING
00027 #define ASSERT_GREATER_EQ(a,b) DO_NOTHING
00028 #define ASSERT_BETWEEN(a,b,c) DO_NOTHING
00029 #define ASSERT_IPC_OK(i) DO_NOTHING
00030 #define ASSERT_OK(e) do { (void)e; } while (0)
00031 #define ASSERT_NOT_NULL(p) DO_NOTHING
00032 #ifndef assert
00033 #define assert(cond) DO_NOTHING
00034 #endif
00035
00036 #else // NDEBUG
00037
00038 #ifndef ASSERT_PRINTF
00039 #include <stdio.h>
00040 #define ASSERT_PRINTF printf
00041 #endif
00042 #ifndef ASSERT_ASSERT
00043 #include <assert.h>
00044 #define ASSERT_ASSERT(x) assert(x)
00045 #endif
00046
00047 #define ASSERT_VALID(cap) \
00048 do { \
00049 typeof(cap) _cap = cap; \
00050 if (!l4_is_invalid_cap(_cap)) { \
00051 ASSERT_PRINTF("%s: Cap invalid.\n", __func__); \
00052 ASSERT_ASSERT(!l4_is_invalid_cap(_cap)); \
00053 } \
00054 } while (0)
00055
00056
00057 #define ASSERT_EQUAL(a, b) \
00058 do { \
00059 typeof(a) _a = a; \
00060 typeof(b) _b = b; \
00061 if (_a != _b) { \
00062 ASSERT_PRINTF("%s:\n", __func__); \
00063 ASSERT_PRINTF(" "#a" (%lx) != "#b" (%lx)\n", (unsigned long)_a, (unsigned long)_b); \
00064 ASSERT_ASSERT(_a == _b); \

```



```

00065 } \
00066 } while (0)
00067
00068
00069 #define ASSERT_NOT_EQUAL(a, b) \
00070 do { \
00071 typeof(a) _a = a; \
00072 typeof(b) _b = b; \
00073 if (_a == _b) { \
00074 ASSERT_PRINTF("%s:\n", __func__); \
00075 ASSERT_PRINTF(" "#a" (%lx) == "#b" (%lx)\n", (unsigned long)_a, (unsigned long)_b); \
00076 ASSERT_ASSERT(_a != _b); \
00077 } \
00078 } while (0)
00079
00080
00081 #define ASSERT_LOWER_EQ(val, max) \
00082 do { \
00083 typeof(val) _val = val; \
00084 typeof(max) _max = max; \
00085 if (_val > _max) { \
00086 ASSERT_PRINTF("%s:\n", __func__); \
00087 ASSERT_PRINTF(" "#val" (%lx) > "#max" (%lx)\n", (unsigned long)_val, (unsigned long)_max); \
00088 ASSERT_ASSERT(_val <= _max); \
00089 } \
00090 } while (0)
00091
00092
00093 #define ASSERT_GREATER_EQ(val, min) \
00094 do { \
00095 typeof(val) _val = val; \
00096 typeof(min) _min = min; \
00097 if (_val < _min) { \
00098 ASSERT_PRINTF("%s:\n", __func__); \
00099 ASSERT_PRINTF(" "#val" (%lx) < "#min" (%lx)\n", (unsigned long)_val, (unsigned long)_min); \
00100 ASSERT_ASSERT(_val >= _min); \
00101 } \
00102 } while (0)
00103
00104
00105 #define ASSERT_BETWEEN(val, min, max) \
00106 ASSERT_LOWER_EQ((val), (max)); \
00107 ASSERT_GREATER_EQ((val), (min));
00108
00109
00110 #define ASSERT_IPC_OK(msgtag) \
00111 do { \
00112 int _r = l4_ipc_error(msgtag, l4_utcb()); \
00113 if (_r) { \
00114 ASSERT_PRINTF("%s: IPC Error: %lx\n", __func__, _r); \
00115 ASSERT_ASSERT(_r == 0); \
00116 } \
00117 } while (0)
00118
00119 #define ASSERT_OK(val) ASSERT_EQUAL((val), 0)
00120 #define ASSERT_NOT_NULL(ptr) ASSERT_NOT_EQUAL((ptr), (void *)0)
00121
00122 #endif // NDEBUG

```

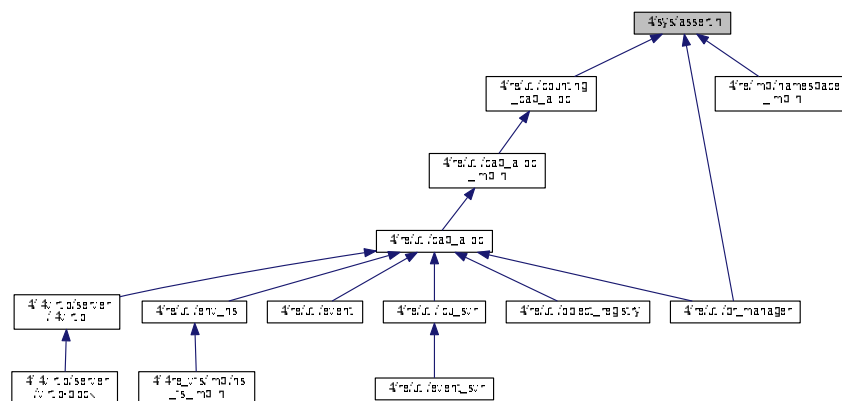
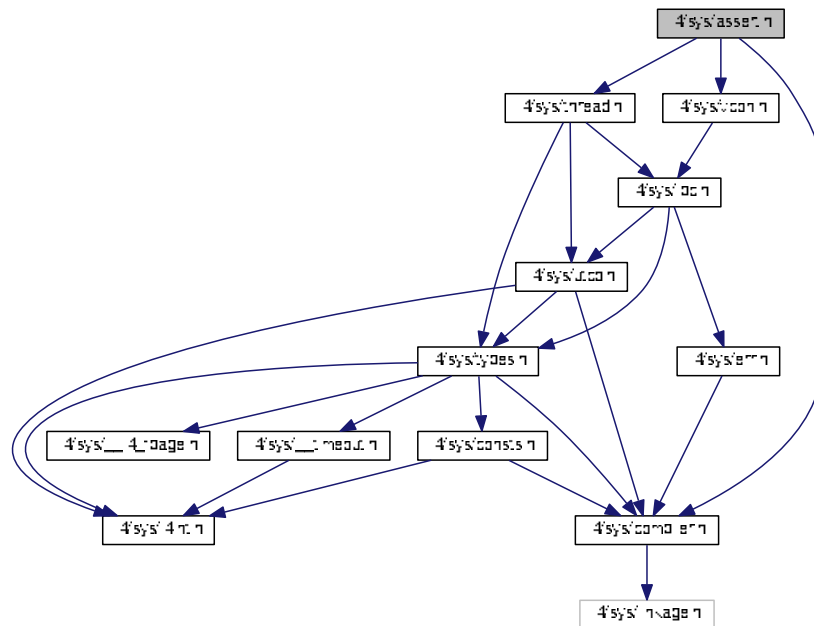
## 15.379 l4/sys/assert.h File Reference

Low-level assert implementation.

```

#include <l4/sys/compiler.h>
#include <l4/sys/thread.h>
#include <l4/sys/vcon.h>

```



- `#define l4_assert(expr)`  
*Low-level assert.*

Low-level assert implementation.  
Definition in file [assert.h](#).

## 15.379.2 Macro Definition Documentation

### 15.379.2.1 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 evaluate 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 [43](#) of file [assert.h](#).

Referenced by [L4Re::Util::Counting\\_cap\\_alloc< COUNTERTYPE >::free\(\)](#), [L4Re::Util::Counting\\_cap\\_alloc< C←OUNTERTYPE >::release\(\)](#), and [L4Re::Util::Br\\_manager::set\\_rcv\\_cap\\_flags\(\)](#).

## 15.380 assert.h

```
00001
00005 /*
00006 * (c) 2015 Adam Lackorzynski <adam@l4re.org>
00007 *
00008 * This file is part of L4Re and distributed under the terms of the
00009 * GNU General Public License 2.
00010 * Please see the COPYING-GPL-2 file for details.
00011 *
00012 * As a special exception, you may use this file as part of a free software
00013 * library without restriction. Specifically, if other files instantiate
00014 * templates or use macros or inline functions from this file, or you compile
00015 * this file and link it with other files to produce an executable, this
00016 * file does not by itself cause the resulting executable to be covered by
00017 * the GNU General Public License. This exception does not however
00018 * invalidate any other reasons why the executable file might be covered by
00019 * the GNU General Public License.
00020 */
00021 #pragma once
00022
00023 #ifdef NDEBUG
00024
00025 #define l4_assert(x) do { } while (0)
00026 #define l4_check(x) do { (void)(x); } while (0)
00027
00028 #else
00029
00030 #include <l4/sys/compiler.h>
00031 #include <l4/sys/thread.h>
00032 #include <l4/sys/vcon.h>
00033
00043 #define l4_assert(expr) \
```

```

00044 l4_assert_fn(expr, __FILE__ ":" L4_stringify(__LINE__) ": Assertion \"" \
00045 L4_stringify(expr) "\" failed.\n")
00046
00047 #define l4_check(expr) l4_assert(expr)
00048
00052 L4_ALWAYS_INLINE
00053 void l4_assert_fn(bool expr, const char *text) L4_NOTHROW;
00054
00058 L4_INLINE L4_NORETURN
00059 void l4_assert_abort(const char *text) L4_NOTHROW;
00060
00061
00062 /* IMPLEMENTATION ----- */
00063
00064 L4_INLINE L4_NORETURN
00065 void l4_assert_abort(const char *text) L4_NOTHROW
00066 {
00067 l4_vcon_write(L4_BASE_LOG_CAP, text, __builtin_strlen(text));
00068 for (;;)
00069 l4_thread_ex_regs(L4_INVALID_CAP, ~0UL, ~0UL,
00070 L4_THREAD_EX_REGS_TRIGGER_EXCEPTION);
00071 }
00072
00073 L4_ALWAYS_INLINE
00074 void l4_assert_fn(bool expr, const char *text) L4_NOTHROW
00075 {
00076 if (L4_LIKELY(expr))
00077 return;
00078 l4_assert_abort(text);
00079 }
00080 }
00081
00082 #endif /* NDEBUG */

```

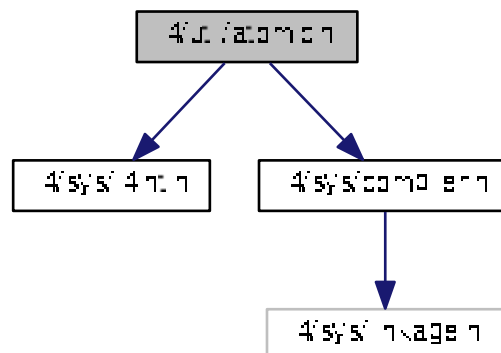
## 15.381 l4/util/atomic.h File Reference

atomic operations header and generic implementations

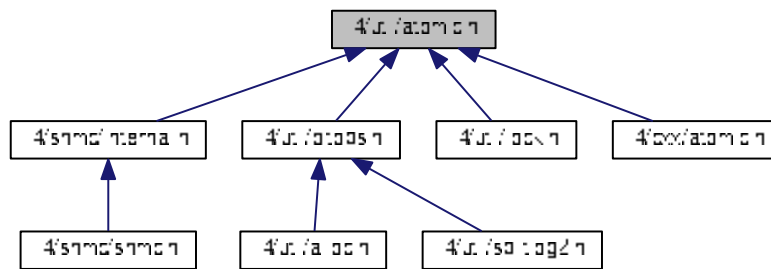
```

#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_cmpxchg64 (volatile l4_uint64_t *dest, l4_uint64_t cmp_val, l4_uint64_t new_val)`  
*Atomic compare and exchange (64 bit version)*
- `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)

### 15.381.1 Detailed Description

atomic operations header and generic implementations

#### Date

10/20/2000

#### Author

Lars Reuther [reuther@os.inf.tu-dresden.de](mailto:reuther@os.inf.tu-dresden.de), Jork Loeser [jork@os.inf.tu-dresden.de](mailto:jork@os.inf.tu-dresden.de)

Definition in file [atomic.h](#).

## 15.382 atomic.h

```

00001 /*****
00010 */
00011 * (c) 2000-2009 Author(s)
00012 * economic rights: Technische Universität Dresden (Germany)
00013 * This file is part of TUD:OS and distributed under the terms of the
00014 * GNU Lesser General Public License 2.1.
00015 * Please see the COPYING-LGPL-2.1 file for details.
00016 */
00017
00018 /*****
00019 #ifndef __L4UTIL__INCLUDE__ATOMIC_H__
00020 #define __L4UTIL__INCLUDE__ATOMIC_H__
00021
00022 #include <l4/sys/l4int.h>
00023 #include <l4/sys/compiler.h>
00024
00025 /*****
00026 *** Prototypes
00027 *****/
00028
00029 EXTERN_C_BEGIN
00030
00049 L4_INLINE int
00050 l4util_cmpxchg64(volatile l4_uint64_t * dest,
00051 l4_uint64_t cmp_val, l4_uint64_t new_val);
00052
00066 L4_INLINE int
00067 l4util_cmpxchg32(volatile l4_uint32_t * dest,
00068 l4_uint32_t cmp_val, l4_uint32_t new_val);
00069
00083 L4_INLINE int
00084 l4util_cmpxchg16(volatile l4_uint16_t * dest,
00085 l4_uint16_t cmp_val, l4_uint16_t new_val);
00086
00100 L4_INLINE int
00101 l4util_cmpxchg8(volatile l4_uint8_t * dest,
00102 l4_uint8_t cmp_val, l4_uint8_t new_val);
00103
00117 L4_INLINE int
00118 l4util_cmpxchg(volatile l4_umword_t * dest,
00119 l4_umword_t cmp_val, l4_umword_t new_val);
00120
00130 L4_INLINE l4_uint32_t
00131 l4util_xchg32(volatile l4_uint32_t * dest, l4_uint32_t val);
00132
00142 L4_INLINE l4_uint16_t
00143 l4util_xchg16(volatile l4_uint16_t * dest, l4_uint16_t val);
00144
00154 L4_INLINE l4_uint8_t
00155 l4util_xchg8(volatile l4_uint8_t * dest, l4_uint8_t val);
00156
00166 L4_INLINE l4_umword_t
00167 l4util_xchg(volatile l4_umword_t * dest, l4_umword_t val);
00168
00170
00176 L4_INLINE void
00177 l4util_add8(volatile l4_uint8_t *dest, l4_uint8_t val);
00178 L4_INLINE void
00179 l4util_add16(volatile l4_uint16_t *dest, l4_uint16_t val);
00180 L4_INLINE void
00181 l4util_add32(volatile l4_uint32_t *dest, l4_uint32_t val);
00182 L4_INLINE void
00183 l4util_sub8(volatile l4_uint8_t *dest, l4_uint8_t val);
00184 L4_INLINE void
00185 l4util_sub16(volatile l4_uint16_t *dest, l4_uint16_t val);
00186 L4_INLINE void
00187 l4util_sub32(volatile l4_uint32_t *dest, l4_uint32_t val);
00188 L4_INLINE void
00189 l4util_and8(volatile l4_uint8_t *dest, l4_uint8_t val);
00190 L4_INLINE void
00191 l4util_and16(volatile l4_uint16_t *dest, l4_uint16_t val);
00192 L4_INLINE void
00193 l4util_and32(volatile l4_uint32_t *dest, l4_uint32_t val);
00194 L4_INLINE void
00195 l4util_or8(volatile l4_uint8_t *dest, l4_uint8_t val);
00196 L4_INLINE void
00197 l4util_or16(volatile l4_uint16_t *dest, l4_uint16_t val);
00198 L4_INLINE void
00199 l4util_or32(volatile l4_uint32_t *dest, l4_uint32_t val);
00201
00203
00210 L4_INLINE l4_uint8_t
00211 l4util_add8_res(volatile l4_uint8_t *dest, l4_uint8_t val);
00212 L4_INLINE l4_uint16_t

```

```

00213 l4util_addl6_res(volatile l4_uint16_t *dest, l4_uint16_t val);
00214 L4_INLINE l4_uint32_t
00215 l4util_add32_res(volatile l4_uint32_t *dest, l4_uint32_t val);
00216 L4_INLINE l4_uint8_t
00217 l4util_sub8_res(volatile l4_uint8_t *dest, l4_uint8_t val);
00218 L4_INLINE l4_uint16_t
00219 l4util_subl6_res(volatile l4_uint16_t *dest, l4_uint16_t val);
00220 L4_INLINE l4_uint32_t
00221 l4util_sub32_res(volatile l4_uint32_t *dest, l4_uint32_t val);
00222 L4_INLINE l4_uint8_t
00223 l4util_and8_res(volatile l4_uint8_t *dest, l4_uint8_t val);
00224 L4_INLINE l4_uint16_t
00225 l4util_andl6_res(volatile l4_uint16_t *dest, l4_uint16_t val);
00226 L4_INLINE l4_uint32_t
00227 l4util_and32_res(volatile l4_uint32_t *dest, l4_uint32_t val);
00228 L4_INLINE l4_uint8_t
00229 l4util_or8_res(volatile l4_uint8_t *dest, l4_uint8_t val);
00230 L4_INLINE l4_uint16_t
00231 l4util_orl6_res(volatile l4_uint16_t *dest, l4_uint16_t val);
00232 L4_INLINE l4_uint32_t
00233 l4util_or32_res(volatile l4_uint32_t *dest, l4_uint32_t val);
00235
00237
00242 L4_INLINE void
00243 l4util_inc8(volatile l4_uint8_t *dest);
00244 L4_INLINE void
00245 l4util_incl6(volatile l4_uint16_t *dest);
00246 L4_INLINE void
00247 l4util_inc32(volatile l4_uint32_t *dest);
00248 L4_INLINE void
00249 l4util_dec8(volatile l4_uint8_t *dest);
00250 L4_INLINE void
00251 l4util_decl6(volatile l4_uint16_t *dest);
00252 L4_INLINE void
00253 l4util_dec32(volatile l4_uint32_t *dest);
00255
00257
00263 L4_INLINE l4_uint8_t
00264 l4util_inc8_res(volatile l4_uint8_t *dest);
00265 L4_INLINE l4_uint16_t
00266 l4util_incl6_res(volatile l4_uint16_t *dest);
00267 L4_INLINE l4_uint32_t
00268 l4util_inc32_res(volatile l4_uint32_t *dest);
00269 L4_INLINE l4_uint8_t
00270 l4util_dec8_res(volatile l4_uint8_t *dest);
00271 L4_INLINE l4_uint16_t
00272 l4util_decl6_res(volatile l4_uint16_t *dest);
00273 L4_INLINE l4_uint32_t
00274 l4util_dec32_res(volatile l4_uint32_t *dest);
00276
00284 L4_INLINE void
00285 l4util_atomic_add(volatile long *dest, long val);
00286
00293 L4_INLINE void
00294 l4util_atomic_inc(volatile long *dest);
00295
00296 EXTERN_C_END
00297
00298 /*****
00299 * IMPLEMENTATION
00300 *****/
00301
00302 L4_INLINE int
00303 l4util_cmpxchg64(volatile l4_uint64_t * dest,
00304 l4_uint64_t cmp_val, l4_uint64_t new_val)
00305 {
00306 return __atomic_compare_exchange_n(dest, &cmp_val, new_val, 0,
00307 __ATOMIC_SEQ_CST, __ATOMIC_SEQ_CST);
00308 }
00309
00310 L4_INLINE int
00311 l4util_cmpxchg32(volatile l4_uint32_t * dest,
00312 l4_uint32_t cmp_val, l4_uint32_t new_val)
00313 {
00314 return __atomic_compare_exchange_n(dest, &cmp_val, new_val, 0,
00315 __ATOMIC_SEQ_CST, __ATOMIC_SEQ_CST);
00316 }
00317
00318 L4_INLINE int
00319 l4util_cmpxchg16(volatile l4_uint16_t * dest,
00320 l4_uint16_t cmp_val, l4_uint16_t new_val)
00321 {
00322 return __atomic_compare_exchange_n(dest, &cmp_val, new_val, 0,
00323 __ATOMIC_SEQ_CST, __ATOMIC_SEQ_CST);
00324 }
00325
00326 L4_INLINE int

```



```

00327 l4util_cmpxchg8(volatile l4_uint8_t * dest,
00328 l4_uint8_t cmp_val, l4_uint8_t new_val)
00329 {
00330 return __atomic_compare_exchange_n(dest, &cmp_val, new_val, 0,
00331 __ATOMIC_SEQ_CST, __ATOMIC_SEQ_CST);
00332 }
00333
00334 L4_INLINE int
00335 l4util_cmpxchg(volatile l4_umword_t * dest,
00336 l4_umword_t cmp_val, l4_umword_t new_val)
00337 {
00338 return __atomic_compare_exchange_n(dest, &cmp_val, new_val, 0,
00339 __ATOMIC_SEQ_CST, __ATOMIC_SEQ_CST);
00340 }
00341
00342 L4_INLINE l4_uint32_t
00343 l4util_xchg32(volatile l4_uint32_t * dest, l4_uint32_t val)
00344 {
00345 return __atomic_exchange_n(dest, val, __ATOMIC_SEQ_CST);
00346 }
00347
00348 L4_INLINE l4_uint16_t
00349 l4util_xchg16(volatile l4_uint16_t * dest, l4_uint16_t val)
00350 {
00351 return __atomic_exchange_n(dest, val, __ATOMIC_SEQ_CST);
00352 }
00353
00354 L4_INLINE l4_uint8_t
00355 l4util_xchg8(volatile l4_uint8_t * dest, l4_uint8_t val)
00356 {
00357 return __atomic_exchange_n(dest, val, __ATOMIC_SEQ_CST);
00358 }
00359
00360 L4_INLINE l4_umword_t
00361 l4util_xchg(volatile l4_umword_t * dest, l4_umword_t val)
00362 {
00363 return __atomic_exchange_n(dest, val, __ATOMIC_SEQ_CST);
00364 }
00365
00366 L4_INLINE void
00367 l4util_inc8(volatile l4_uint8_t *dest)
00368 { __atomic_fetch_add(dest, 1, __ATOMIC_SEQ_CST); }
00369
00370 L4_INLINE void
00371 l4util_inc16(volatile l4_uint16_t *dest)
00372 { __atomic_fetch_add(dest, 1, __ATOMIC_SEQ_CST); }
00373
00374 L4_INLINE void
00375 l4util_inc32(volatile l4_uint32_t *dest)
00376 { __atomic_fetch_add(dest, 1, __ATOMIC_SEQ_CST); }
00377
00378 L4_INLINE void
00379 l4util_atomic_inc(volatile long *dest)
00380 { __atomic_fetch_add(dest, 1, __ATOMIC_SEQ_CST); }
00381
00382 L4_INLINE void
00383 l4util_dec8(volatile l4_uint8_t *dest)
00384 { __atomic_fetch_sub(dest, 1, __ATOMIC_SEQ_CST); }
00385
00386 L4_INLINE void
00387 l4util_dec16(volatile l4_uint16_t *dest)
00388 { __atomic_fetch_sub(dest, 1, __ATOMIC_SEQ_CST); }
00389
00390 L4_INLINE void
00391 l4util_dec32(volatile l4_uint32_t *dest)
00392 { __atomic_fetch_sub(dest, 1, __ATOMIC_SEQ_CST); }
00393
00394
00395 L4_INLINE l4_uint8_t
00396 l4util_inc8_res(volatile l4_uint8_t *dest)
00397 { return __atomic_add_fetch(dest, 1, __ATOMIC_SEQ_CST); }
00398
00399 L4_INLINE l4_uint16_t
00400 l4util_inc16_res(volatile l4_uint16_t *dest)
00401 { return __atomic_add_fetch(dest, 1, __ATOMIC_SEQ_CST); }
00402
00403 L4_INLINE l4_uint32_t
00404 l4util_inc32_res(volatile l4_uint32_t *dest)
00405 { return __atomic_add_fetch(dest, 1, __ATOMIC_SEQ_CST); }
00406
00407 L4_INLINE l4_uint8_t
00408 l4util_dec8_res(volatile l4_uint8_t *dest)
00409 { return __atomic_sub_fetch(dest, 1, __ATOMIC_SEQ_CST); }
00410
00411 L4_INLINE l4_uint16_t
00412 l4util_dec16_res(volatile l4_uint16_t *dest)
00413 { return __atomic_sub_fetch(dest, 1, __ATOMIC_SEQ_CST); }

```

```

00414
00415 L4_INLINE l4_uint32_t
00416 l4util_dec32_res(volatile l4_uint32_t *dest)
00417 { return __atomic_sub_fetch(dest, 1, __ATOMIC_SEQ_CST); }
00418
00419 L4_INLINE l4_umword_t
00420 l4util_dec_res(volatile l4_umword_t *dest)
00421 { return __atomic_sub_fetch(dest, 1, __ATOMIC_SEQ_CST); }
00422
00423 L4_INLINE void
00424 l4util_add8(volatile l4_uint8_t *dest, l4_uint8_t val)
00425 { __atomic_fetch_add(dest, val, __ATOMIC_SEQ_CST); }
00426
00427 L4_INLINE void
00428 l4util_add16(volatile l4_uint16_t *dest, l4_uint16_t val)
00429 { __atomic_fetch_add(dest, val, __ATOMIC_SEQ_CST); }
00430
00431 L4_INLINE void
00432 l4util_add32(volatile l4_uint32_t *dest, l4_uint32_t val)
00433 { __atomic_fetch_add(dest, val, __ATOMIC_SEQ_CST); }
00434
00435 L4_INLINE void
00436 l4util_atomic_add(volatile long *dest, long val)
00437 { __atomic_fetch_add(dest, val, __ATOMIC_SEQ_CST); }
00438
00439 L4_INLINE void
00440 l4util_sub8(volatile l4_uint8_t *dest, l4_uint8_t val)
00441 { __atomic_fetch_sub(dest, val, __ATOMIC_SEQ_CST); }
00442
00443 L4_INLINE void
00444 l4util_sub16(volatile l4_uint16_t *dest, l4_uint16_t val)
00445 { __atomic_fetch_sub(dest, val, __ATOMIC_SEQ_CST); }
00446
00447 L4_INLINE void
00448 l4util_sub32(volatile l4_uint32_t *dest, l4_uint32_t val)
00449 { __atomic_fetch_sub(dest, val, __ATOMIC_SEQ_CST); }
00450
00451 L4_INLINE void
00452 l4util_and8(volatile l4_uint8_t *dest, l4_uint8_t val)
00453 { __atomic_fetch_and(dest, val, __ATOMIC_SEQ_CST); }
00454
00455 L4_INLINE void
00456 l4util_and16(volatile l4_uint16_t *dest, l4_uint16_t val)
00457 { __atomic_fetch_and(dest, val, __ATOMIC_SEQ_CST); }
00458
00459 L4_INLINE void
00460 l4util_and32(volatile l4_uint32_t *dest, l4_uint32_t val)
00461 { __atomic_fetch_and(dest, val, __ATOMIC_SEQ_CST); }
00462
00463 L4_INLINE void
00464 l4util_or8(volatile l4_uint8_t *dest, l4_uint8_t val)
00465 { __atomic_fetch_or(dest, val, __ATOMIC_SEQ_CST); }
00466
00467 L4_INLINE void
00468 l4util_or16(volatile l4_uint16_t *dest, l4_uint16_t val)
00469 { __atomic_fetch_or(dest, val, __ATOMIC_SEQ_CST); }
00470
00471 L4_INLINE void
00472 l4util_or32(volatile l4_uint32_t *dest, l4_uint32_t val)
00473 { __atomic_fetch_or(dest, val, __ATOMIC_SEQ_CST); }
00474
00475 L4_INLINE l4_uint8_t
00476 l4util_add8_res(volatile l4_uint8_t *dest, l4_uint8_t val)
00477 { return __atomic_add_fetch(dest, val, __ATOMIC_SEQ_CST); }
00478
00479 L4_INLINE l4_uint16_t
00480 l4util_add16_res(volatile l4_uint16_t *dest, l4_uint16_t val)
00481 { return __atomic_add_fetch(dest, val, __ATOMIC_SEQ_CST); }
00482
00483 L4_INLINE l4_uint32_t
00484 l4util_add32_res(volatile l4_uint32_t *dest, l4_uint32_t val)
00485 { return __atomic_add_fetch(dest, val, __ATOMIC_SEQ_CST); }
00486
00487 L4_INLINE l4_uint8_t
00488 l4util_sub8_res(volatile l4_uint8_t *dest, l4_uint8_t val)
00489 { return __atomic_sub_fetch(dest, val, __ATOMIC_SEQ_CST); }
00490
00491 L4_INLINE l4_uint16_t
00492 l4util_sub16_res(volatile l4_uint16_t *dest, l4_uint16_t val)
00493 { return __atomic_sub_fetch(dest, val, __ATOMIC_SEQ_CST); }
00494
00495 L4_INLINE l4_uint32_t
00496 l4util_sub32_res(volatile l4_uint32_t *dest, l4_uint32_t val)
00497 { return __atomic_sub_fetch(dest, val, __ATOMIC_SEQ_CST); }
00498
00499 L4_INLINE l4_uint8_t
00500 l4util_and8_res(volatile l4_uint8_t *dest, l4_uint8_t val)

```

```

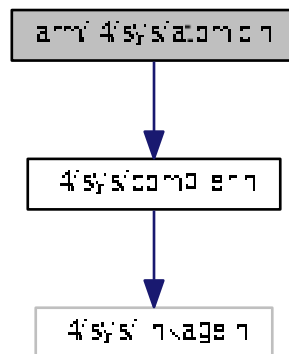
00501 { return __atomic_and_fetch(dest, val, __ATOMIC_SEQ_CST); }
00502
00503 L4_INLINE l4_uint16_t
00504 l4util_and16_res(volatile l4_uint16_t *dest, l4_uint16_t val)
00505 { return __atomic_and_fetch(dest, val, __ATOMIC_SEQ_CST); }
00506
00507 L4_INLINE l4_uint32_t
00508 l4util_and32_res(volatile l4_uint32_t *dest, l4_uint32_t val)
00509 { return __atomic_and_fetch(dest, val, __ATOMIC_SEQ_CST); }
00510
00511 L4_INLINE l4_uint8_t
00512 l4util_or8_res(volatile l4_uint8_t *dest, l4_uint8_t val)
00513 { return __atomic_or_fetch(dest, val, __ATOMIC_SEQ_CST); }
00514
00515 L4_INLINE l4_uint16_t
00516 l4util_or16_res(volatile l4_uint16_t *dest, l4_uint16_t val)
00517 { return __atomic_or_fetch(dest, val, __ATOMIC_SEQ_CST); }
00518
00519 L4_INLINE l4_uint32_t
00520 l4util_or32_res(volatile l4_uint32_t *dest, l4_uint32_t val)
00521 { return __atomic_or_fetch(dest, val, __ATOMIC_SEQ_CST); }
00522
00523 #endif /* ! __L4UTIL__INCLUDE__ATOMIC_H__ */

```

## 15.383 arm/l4/sys/atomic.h File Reference

Atomic memory modifications.

#include <l4/sys/compiler.h>  
 Include dependency graph for atomic.h:



### 15.383.1 Detailed Description

Atomic memory modifications.

Definition in file [atomic.h](#).

## 15.384 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 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.
00023 */
00024 #pragma once
00025
00026 #include <l4/sys/compiler.h>
00027
00028 EXTERN_C long int
00029 l4_atomic_add(volatile long int* mem, long int offset) L4_NOTHROW L4_LONG_CALL;
00030
00031 EXTERN_C long int
00032 l4_atomic_xchg(volatile long int* mem, long int newval) L4_NOTHROW L4_LONG_CALL;
00033
00034 EXTERN_C long int
00035 l4_atomic_cmpxchg(volatile long int* mem, long int oldval, long int newval)
00036 L4_NOTHROW L4_LONG_CALL;

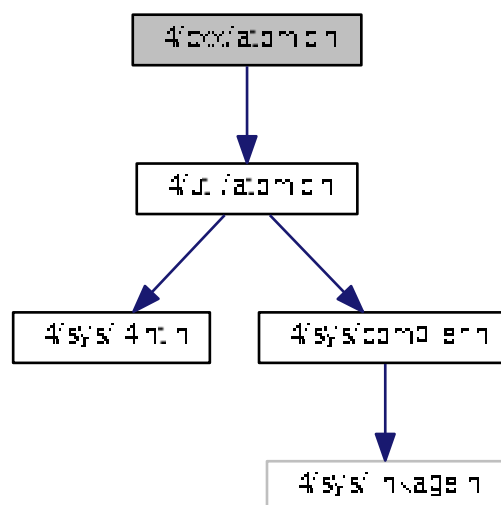
```

## 15.385 l4/cxx/atomic.h File Reference

Atomic template.

```
#include <l4/util/atomic.h>
```

Include dependency graph for atomic.h:



## Namespaces

- [L4](#)

[L4](#) low-level kernel interface.

## 15.385.1 Detailed Description

Atomic template.

Definition in file [atomic.h](#).

## 15.386 atomic.h

```

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 * 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 * As a special exception, you may use this file as part of a free software
00015 * library without restriction. Specifically, if other files instantiate
00016 * templates or use macros or inline functions from this file, or you compile
00017 * this file and link it with other files to produce an executable, this
00018 * file does not by itself cause the resulting executable to be covered by
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_ATOMIC_H__
00025 #define L4_CXX_ATOMIC_H__
00026
00027 #include <l4/util/atomic.h>
00028
00029 extern "C" void ____error_compare_and_swap_does_not_support_3_bytes____();
00030 extern "C" void ____error_compare_and_swap_does_not_support_more_than_4_bytes____();
00031
00032 namespace L4
00033 {
00034 template< typename X >
00035 inline int compare_and_swap(X volatile *dst, X old_val, X new_val)
00036 {
00037 switch (sizeof(X))
00038 {
00039 case 1:
00040 return l4util_cmpxchg8((l4_uint8_t volatile*)dst, old_val, new_val);
00041 case 2:
00042 return l4util_cmpxchg16((l4_uint16_t volatile *)dst, old_val, new_val);
00043 case 3: ____error_compare_and_swap_does_not_support_3_bytes____();
00044 case 4:
00045 return l4util_cmpxchg32((l4_uint32_t volatile*)dst, old_val, new_val);
00046 default:
00047 ____error_compare_and_swap_does_not_support_more_than_4_bytes____();
00048 }
00049 return 0;
00050 }
00051 };
00052
00053 #endif // L4_CXX_ATOMIC_H__

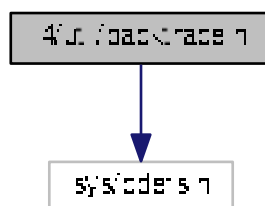
```

## 15.387 l4/util/backtrace.h File Reference

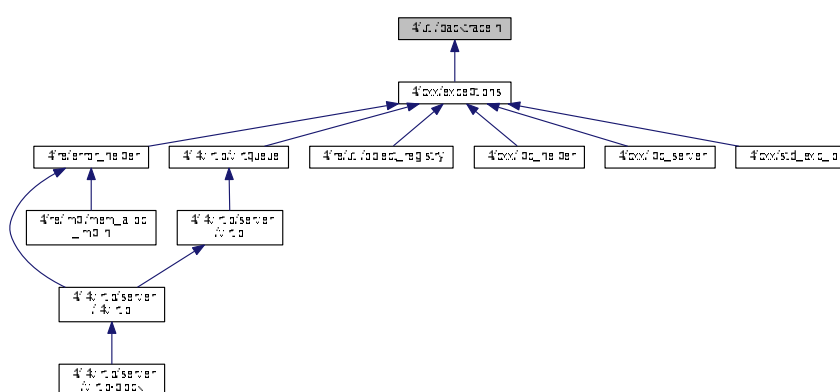
Backtrace.

```
#include <sys/cdefs.h>
```

Include dependency graph for backtrace.h:



This graph shows which files directly or indirectly include this file:



### Functions

- [int l4util\\_backtrace](#) (void \*\*pc\_array, int max\_len)  
*Fill backtrace structure.*

### 15.387.1 Detailed Description

Backtrace.

Definition in file [backtrace.h](#).

## 15.387.2 Function Documentation

### 15.387.2.1 l4util\_backtrace()

```
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

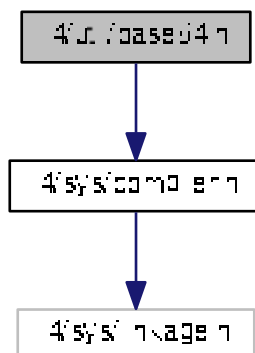
## 15.388 backtrace.h

```
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 * 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 <sys/cdefs.h>
00016
00017 __BEGIN_DECLS
00018
00026 int l4util_backtrace(void **pc_array, int max_len);
00027
00028 __END_DECLS
```

## 15.389 l4/util/base64.h File Reference

base 64 encoding and decoding functions adapted from Bob Trower 08/04/01

```
#include <lib/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*

### 15.389.1 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](mailto:jn6@os.inf.tu-dresden.de)

Definition in file [base64.h](#).



## 15.390 base64.h

```

00001
00010 /*
00011 * (c) 2008-2009 Author(s)
00012 * economic rights: Technische Universität Dresden (Germany)
00013 * This file is part of TUD:OS and distributed under the terms of the
00014 * GNU Lesser General Public License 2.1.
00015 * Please see the COPYING-LGPL-2.1 file for details.
00016 */
00017
00018 #ifndef B64_EN_DECODE
00019 #define B64_EN_DECODE
00020
00021 #include <l4/sys/compiler.h>
00022
00023 EXTERN_C_BEGIN
00024
00030
00041 L4_CV void base64_encode(const char *infile, unsigned int in_size, char **outfile);
00042
00053 L4_CV void base64_decode(const char *infile, unsigned int in_size, char **outfile);
00054
00055 EXTERN_C_END
00056
00058 #endif //B64_EN_DECODE

```

## 15.391 l4/util/bitops.h File Reference

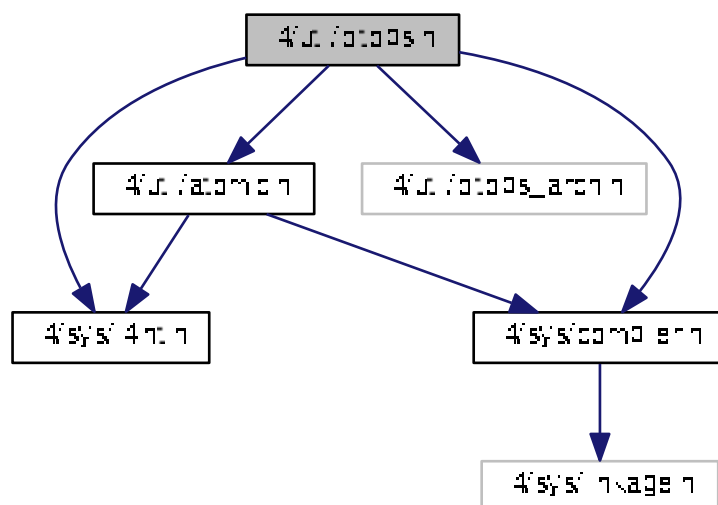
bit manipulation functions

```

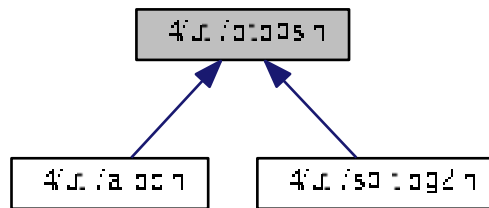
#include <l4/sys/l4int.h>
#include <l4/sys/compiler.h>
#include <l4/util/bitops_arch.h>
#include <l4/util/atomic.h>

```

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) l4util_btr(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` (const unsigned long val)  
*Find the next power of 2 for a given number.*

### 15.391.1 Detailed Description

bit manipulation functions

#### Date

07/03/2001

#### Author

Lars Reuther [reuther@os.inf.tu-dresden.de](mailto:reuther@os.inf.tu-dresden.de)

Definition in file [bitops.h](#).

## 15.392 bitops.h

```

00001 /*****
00009 */
00010 * (c) 2000-2009 Author(s)
00011 * economic rights: Technische Universität Dresden (Germany)
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU Lesser General Public License 2.1.
00014 * Please see the COPYING-LGPL-2.1 file for details.
00015 */
00016
00017 /*****
00018 #ifndef __L4UTIL__INCLUDE__BITOPS_H__
00019 #define __L4UTIL__INCLUDE__BITOPS_H__
00020
00021 */ L4 includes */
00022 #include <l4/sys/l4int.h>
00023 #include <l4/sys/compiler.h>
00024
00026 #define l4util_test_and_clear_bit(b, dest) l4util_btr(b, dest)
00027 #define l4util_test_and_set_bit(b, dest) l4util_bts(b, dest)
00028 #define l4util_test_and_change_bit(b, dest) l4util_btc(b, dest)
00029 #define l4util_log2(word) l4util_bsr(word)
00030
00031 /*****
00032 *** Prototypes
00033 *****/
00034
00035 EXTERN_C_BEGIN
00036
00049 L4_INLINE void
00050 l4util_set_bit(int b, volatile l4_umword_t * dest);
00051
00059 L4_INLINE void
00060 l4util_clear_bit(int b, volatile l4_umword_t * dest);
00061
00069 L4_INLINE void
00070 l4util_complement_bit(int b, volatile l4_umword_t * dest);
00071
00081 L4_INLINE int
00082 l4util_test_bit(int b, const volatile l4_umword_t * dest);
00083
00095 L4_INLINE int
00096 l4util_bts(int b, volatile l4_umword_t * dest);
00097
00109 L4_INLINE int
00110 l4util_btr(int b, volatile l4_umword_t * dest);
00111
00123 L4_INLINE int
00124 l4util_btc(int b, volatile l4_umword_t * dest);
00125
00137 L4_INLINE int
00138 l4util_bsr(l4_umword_t word);
00139
00151 L4_INLINE int
00152 l4util_bsf(l4_umword_t word);
00153
00164 L4_INLINE int
00165 l4util_find_first_set_bit(const void * dest, l4_size_t size);

```

```

00166
00177 L4_INLINE int
00178 l4util_find_first_zero_bit(const void * dest,
00179 l4_size_t size);
00179
00180
00189 L4_INLINE int
00190 l4util_next_power2(const unsigned long val);
00191
00192 EXTERN_C_END
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 return -1;
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 return -1;
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,
00334 l4_size_t size)
00335 {
00336 l4_size_t i, j;
00337 unsigned long *v = (unsigned long*)dest;
00338
00339 if (!size)
00340 return 0;
00341
00342 size = (size + 31) & ~0x1f; /* Grmbl: adapt to x86 implementation... */
00343
00344 for (i = j = 0; i < size; i++, j++)
00345 {
00346 if (j >= sizeof(*v) * 8)
00347 {
00348 j = 0;
00349 v++;
00350 }
00351 if (!((1UL << j) & *v))
00352 return i;
00353 }
00354 return size + 1;
00355 }
00356 #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++)
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 0; }
00396 #endif
00397
00398 #ifndef __L4UTIL_BITOPS_HAVE_ARCH_FIND_FIRST_SET_BIT
00399 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(); return 0; }
00402 #endif
00403
00404 #endif /* ! __L4UTIL__INCLUDE__BITOPS_H__ */

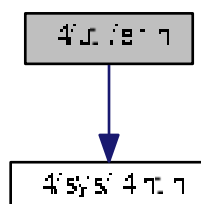
```

## 15.393 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 - figure 1-9, page 1-9.*
- 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\\_Sym](#)  
*ELF32 symbol table entry.*
- struct [Elf64\\_Sym](#)  
*ELF64 symbol table entry.*

## Macros

- #define [EI\\_NIDENT](#) 16  
*number of characters*
- #define [EI\\_CLASS](#) 4  
*ELF class byte index.*
- #define [ELFCLASSNONE](#) 0  
*Invalid ELF class.*
- #define [ELFCLASS32](#) 1  
*32-bit objects*
- #define [ELFCLASS64](#) 2  
*64-bit objects*
- #define [ELFCLASSNUM](#) 3  
*Mask for 32-bit or 64-bit class.*
- #define [EI\\_DATA](#) 5  
*Data encoding byte index.*
- #define [ELFDATANONE](#) 0  
*Invalid data encoding.*
- #define [ELFDATA2LSB](#) 1  
*2's complement, little endian*
- #define [ELFDATA2MSB](#) 2  
*2's complement, big endian*
- #define [EI\\_VERSION](#) 6  
*File version byte index.*
- #define [EI\\_OSABI](#) 7  
*OS ABI identification.*
- #define [ELFOSABI\\_NONE](#) 0

- *UNIX System V ABI.*
- #define [ELFOSABI\\_SYSV](#) 0
- *Alias.*
- #define [ELFOSABI\\_HPUX](#) 1
- *HP-UX.*
- #define [ELFOSABI\\_NETBSD](#) 2
- *NetBSD.*
- #define [ELFOSABI\\_LINUX](#) 3
- *Linux.*
- #define [ELFOSABI\\_SOLARIS](#) 6
- *Sun Solaris.*
- #define [ELFOSABI\\_AIX](#) 7
- *IBM AIX.*
- #define [ELFOSABI\\_IRIX](#) 8
- *SGI Irix.*
- #define [ELFOSABI\\_FREEBSD](#) 9
- *FreeBSD.*
- #define [ELFOSABI\\_TRU64](#) 10
- *Compaq TRU64 UNIX.*
- #define [ELFOSABI\\_MODESTO](#) 11
- *Novell Modesto.*
- #define [ELFOSABI\\_OPENBSD](#) 12
- *OpenBSD.*
- #define [ELFOSABI\\_ARM](#) 97
- *ARM.*
- #define [ELFOSABI\\_STANDALONE](#) 255
- *Standalone (embedded) application.*
- #define [EI\\_ABIVERSION](#) 8
- *ABI version.*
- #define [EI\\_PAD](#) 9
- *Byte index of padding bytes.*
- #define [ET\\_NONE](#) 0
- *no file type*
- #define [ET\\_REL](#) 1
- *relocatable file*
- #define [ET\\_EXEC](#) 2
- *executable file*
- #define [ET\\_DYN](#) 3
- *shared object file*
- #define [ET\\_CORE](#) 4
- *core file*
- #define [ET\\_LOPROC](#) 0xff00
- *processor-specific*
- #define [ET\\_HIPROC](#) 0xffff
- *processor-specific*
- #define [EM\\_NONE](#) 0
- *no machine*
- #define [EM\\_M32](#) 1
- *AT&T WE 32100.*
- #define [EM\\_SPARC](#) 2
- *SPARC.*



- #define [EM\\_386](#) 3  
*Intel 80386.*
- #define [EM\\_68K](#) 4  
*Motorola 68000.*
- #define [EM\\_88K](#) 5  
*Motorola 88000.*
- #define [EM\\_860](#) 7  
*Intel 80860.*
- #define [EM\\_MIPS](#) 8  
*MIPS RS3000 big-endian.*
- #define [EM\\_MIPS\\_RS4\\_BE](#) 10  
*MIPS RS4000 big-endian.*
- #define [EM\\_SPARC64](#) 11  
*SPARC 64-bit.*
- #define [EM\\_PARISC](#) 15  
*HP PA-RISC.*
- #define [EM\\_VPP500](#) 17  
*Fujitsu VPP500.*
- #define [EM\\_SPARC32PLUS](#) 18  
*Sun's V8plus.*
- #define [EM\\_960](#) 19  
*Intel 80960.*
- #define [EM\\_PPC](#) 20  
*PowerPC.*
- #define [EM\\_V800](#) 36  
*NEC V800.*
- #define [EM\\_FR20](#) 37  
*Fujitsu FR20.*
- #define [EM\\_RH32](#) 38  
*TRW RH-32.*
- #define [EM\\_RCE](#) 39  
*Motorola RCE.*
- #define [EM\\_ARM](#) 40  
*Advanced RISC Machines ARM.*
- #define [EM\\_ALPHA](#) 41  
*Digital Alpha.*
- #define [EM\\_SH](#) 42  
*Hitachi SuperH.*
- #define [EM\\_SPARCV9](#) 43  
*SPARC v9 64-bit.*
- #define [EM\\_TRICORE](#) 44  
*Siemens Tricore embedded processor.*
- #define [EM\\_ARC](#) 45  
*Argonaut RISC Core, Argonaut Techn Inc.*
- #define [EM\\_H8\\_300](#) 46  
*Hitachi H8/300.*
- #define [EM\\_H8\\_300H](#) 47  
*Hitachi H8/300H.*
- #define [EM\\_H8S](#) 48  
*Hitachi H8/S.*
- #define [EM\\_H8\\_500](#) 49

- Hitachi H8/500.*
- #define [EM\\_IA\\_64](#) 50
- HP/Intel IA-64.*
- #define [EM\\_MIPS\\_X](#) 51
- Stanford MIPS-X.*
- #define [EM\\_COLDFIRE](#) 52
- Motorola Coldfire.*
- #define [EM\\_68HC12](#) 53
- Motorola M68HC12.*
- #define [EM\\_X86\\_64](#) 62
- Advanced Micro Devices x86-64.*
- #define [EM\\_PDSP](#) 63
- Sony DSP Processor.*
- #define [EM\\_FX66](#) 66
- Siemens FX66 microcontroller.*
- #define [EM\\_ST9PLUS](#) 67
- STMicroelectronics ST9+ 8/16 mc.*
- #define [EM\\_ST7](#) 68
- STmicroelectronics ST7 8 bit mc.*
- #define [EM\\_68HC16](#) 69
- Motorola MC68HC16 microcontroller.*
- #define [EM\\_68HC11](#) 70
- Motorola MC68HC11 microcontroller.*
- #define [EM\\_68HC08](#) 71
- Motorola MC68HC08 microcontroller.*
- #define [EM\\_68HC05](#) 72
- Motorola MC68HC05 microcontroller.*
- #define [EM\\_SVX](#) 73
- Silicon Graphics SVx.*
- #define [EM\\_ST19](#) 74
- STMicroelectronics ST19 8 bit mc.*
- #define [EM\\_VAX](#) 75
- Digital VAX.*
- #define [EM\\_CRIS](#) 76
- Axis Communications 32-bit embedded processor.*
- #define [EM\\_JAVELIN](#) 77
- Infineon Technologies 32-bit embedded processor.*
- #define [EM\\_FIREPATH](#) 78
- Element 14 64-bit DSP Processor.*
- #define [EM\\_ZSP](#) 79
- LSI Logic 16-bit DSP Processor.*
- #define [EM\\_MMIX](#) 80
- Donald Knuth's educational 64-bit processor.*
- #define [EM\\_HUANY](#) 81
- Harvard University machine-independent object files.*
- #define [EM\\_PRISM](#) 82
- SiTera Prism.*
- #define [EM\\_AVR](#) 83
- Atmel AVR 8-bit microcontroller.*
- #define [EM\\_FR30](#) 84
- Fujitsu FR30.*

- #define [EM\\_D10V](#) 85  
*Mitsubishi D10V.*
- #define [EM\\_D30V](#) 86  
*Mitsubishi D30V.*
- #define [EM\\_V850](#) 87  
*NEC v850.*
- #define [EM\\_M32R](#) 88  
*Mitsubishi M32R.*
- #define [EM\\_MN10300](#) 89  
*Matsushita MN10300.*
- #define [EM\\_MN10200](#) 90  
*Matsushita MN10200.*
- #define [EM\\_PJ](#) 91  
*picoJava*
- #define [EM\\_OPENRISC](#) 92  
*OpenRISC 32-bit embedded processor.*
- #define [EM\\_ARC\\_A5](#) 93  
*ARC Cores Tangent-A5.*
- #define [EM\\_XTENSA](#) 94  
*Tensilica Xtensa Architecture.*
- #define [EM\\_ALTERA\\_NIOS2](#) 113  
*Altera Nios II.*
- #define [EM\\_AARCH64](#) 183  
*ARM AARCH64.*
- #define [EM\\_TILEPRO](#) 188  
*Tilera TILEPro.*
- #define [EM\\_MICROBLAZE](#) 189  
*Xilinx MicroBlaze.*
- #define [EM\\_TILEGX](#) 191  
*Tilera TILE-Gx.*
- #define [EV\\_NONE](#) 0  
*Invalid version.*
- #define [EV\\_CURRENT](#) 1  
*Current version.*
- #define [EI\\_MAG0](#) 0  
*file id*
- #define [EI\\_MAG1](#) 1  
*file id*
- #define [EI\\_MAG2](#) 2  
*file id*
- #define [EI\\_MAG3](#) 3  
*file id*
- #define [EI\\_CLASS](#) 4  
*ELF class byte index.*
- #define [EI\\_DATA](#) 5  
*Data encoding byte index.*
- #define [EI\\_VERSION](#) 6  
*File version byte index.*
- #define [EI\\_OSABI](#) 7  
*OS ABI identification.*
- #define [EI\\_ABIVERSION](#) 8

- ABI version.
- #define `EL_PAD` 9
  - Byte index of padding bytes.
- #define `ELFMAG0` 0x7f
  - `e_ident[EI_MAG0]`
- #define `ELFMAG1` 'E'
  - `e_ident[EI_MAG1]`
- #define `ELFMAG2` 'L'
  - `e_ident[EI_MAG2]`
- #define `ELFMAG3` 'F'
  - `e_ident[EI_MAG3]`
- #define `ELFCLASSNONE` 0
  - Invalid ELF class.
- #define `ELFCLASS32` 1
  - 32-bit object
- #define `ELFCLASS64` 2
  - 64-bit object
- #define `ELFDATANONE` 0
  - Invalid data encoding.
- #define `ELFDATA2LSB` 1
  - 2's complement, little endian
- #define `ELFDATA2MSB` 2
  - 2's complement, big endian
- #define `ELFOSABI_SYSV` 0
  - Alias.
- #define `ELFOSABI_HPUX` 1
  - HP-UX.
- #define `ELFOSABI_STANDALONE` 255
  - Standalone (embedded) application.
- #define `SHN_UNDEF` 0
  - undefined section header entry
- #define `SHN_LORESERVE` 0xff00
  - lower bound of reserved indexes
- #define `SHN_LOPROC` 0xff00
  - lower bound of proc spec entr
- #define `SHN_HIPROC` 0xff1f
  - upper bound of proc spec entr
- #define `SHN_ABS` 0xffff
  - absolute values for ref
- #define `SHN_COMMON` 0xffff2
  - common symbols
- #define `SHN_HIRESERVE` 0xffff
  - upper bound of reserved indexes
- #define `SHT_INIT_ARRAY` 14
  - Array of constructors.
- #define `SHT_FINI_ARRAY` 15
  - Array of destructors.
- #define `SHT_PREINIT_ARRAY` 16
  - Array of pre-constructors.
- #define `SHT_GROUP` 17
  - Section group.

- #define [SHT\\_SYMTAB\\_SHNDX](#) 18  
*Extended section indeces.*
- #define [SHT\\_NUM](#) 19  
*Number of defined types.*
- #define [SHF\\_WRITE](#) 0x1  
*writeable during execution*
- #define [SHF\\_ALLOC](#) 0x2  
*section occupies virt memory*
- #define [SHF\\_EXECINSTR](#) 0x4  
*code section*
- #define [SHF\\_MERGE](#) 0x10  
*Might be merged.*
- #define [SHF\\_STRINGS](#) 0x20  
*Contains nul-terminated strings.*
- #define [SHF\\_INFO\\_LINK](#) 0x40  
*'sh\_info' contains SHT index*
- #define [SHF\\_LINK\\_ORDER](#) 0x80  
*Preserve order after combining.*
- #define [SHF\\_OS\\_NONCONFORMING](#) 0x100  
*Non-standard OS specific handling required.*
- #define [SHF\\_GROUP](#) 0x200  
*Section is member of a group.*
- #define [SHF\\_TLS](#) 0x400  
*Section hold thread-local data.*
- #define [SHF\\_MASKOS](#) 0x0ff00000  
*OS-specific.*
- #define [SHF\\_MASKPROC](#) 0xf0000000  
*proc spec mask*
- #define [PT\\_NULL](#) 0  
*array is unused*
- #define [PT\\_LOAD](#) 1  
*loadable*
- #define [PT\\_DYNAMIC](#) 2  
*dynamic linking information*
- #define [PT\\_INTERP](#) 3  
*path to interpreter*
- #define [PT\\_NOTE](#) 4  
*auxiliary information*
- #define [PT\\_SHLIB](#) 5  
*reserved*
- #define [PT\\_PHDR](#) 6  
*location of the pht itself*
- #define [PT\\_TLS](#) 7  
*Thread-local storage segment.*
- #define [PT\\_NUM](#) 8  
*Number of defined types.*
- #define [PT\\_LOOS](#) 0x60000000  
*os spec.*
- #define [PT\\_HIOS](#) 0x6fffffff  
*os spec.*
- #define [PT\\_LOPROC](#) 0x70000000

- processor spec.*
- #define `PT_HIPROC` 0x7fffffff
- processor spec.*
- #define `PT_GNU_EH_FRAME` (`PT_LOOS` + 0x474e550)
- EH frame information.*
- #define `PT_GNU_STACK` (`PT_LOOS` + 0x474e551)
- Flags for stack.*
- #define `PT_GNU_RELRO` (`PT_LOOS` + 0x474e552)
- Read only after reloc.*
- #define `PT_L4_STACK` (`PT_LOOS` + 0x12)
- Address of the stack.*
- #define `PT_L4_KIP` (`PT_LOOS` + 0x13)
- Address of the KIP.*
- #define `PT_L4_AUX` (`PT_LOOS` + 0x14)
- Address of the AUX structures.*
- #define `NT_PRSTATUS` 1
- Contains copy of prstatus struct.*
- #define `NT_FPREGSET` 2
- Contains copy of fpregset struct.*
- #define `NT_PRPSINFO` 3
- Contains copy of prpsinfo struct.*
- #define `NT_PRXREG` 4
- Contains copy of prxregset struct.*
- #define `NT_TASKSTRUCT` 4
- Contains copy of task structure.*
- #define `NT_PLATFORM` 5
- String from sysinfo(SI\_PLATFORM)*
- #define `NT_AUXV` 6
- Contains copy of auxv array.*
- #define `NT_GWINDOWS` 7
- Contains copy of gwindows struct.*
- #define `NT_ASRS` 8
- Contains copy of asrset struct.*
- #define `NT_PSTATUS` 10
- Contains copy of pstatus struct.*
- #define `NT_PSINFO` 13
- Contains copy of psinfo struct.*
- #define `NT_PRCRED` 14
- Contains copy of prcred struct.*
- #define `NT_UTSNAME` 15
- Contains copy of utsname struct.*
- #define `NT_LWPSTATUS` 16
- Contains copy of lwpstatus struct.*
- #define `NT_LWPSINFO` 17
- Contains copy of lwpinfo struct.*
- #define `NT_PRFPXREG` 20
- Contains copy of fprxregset struct.*
- #define `NT_VERSION` 1
- Contains a version string.*
- #define `DT_NULL` 0

*Dynamic Array Tags, d\_tag - figure 2-10, page 2-12.*

- #define [DT\\_NEEDED](#) 1  
*name of a needed library*
- #define [DT\\_PLTRELSZ](#) 2  
*total size of relocation entry*
- #define [DT\\_PLTGOT](#) 3  
*address assoc with prog link table*
- #define [DT\\_HASH](#) 4  
*address of symbol hash table*
- #define [DT\\_STRTAB](#) 5  
*address of string table*
- #define [DT\\_SYMTAB](#) 6  
*address of symbol table*
- #define [DT\\_RELA](#) 7  
*address of relocation table*
- #define [DT\\_RELASZ](#) 8  
*total size of relocation table*
- #define [DT\\_RELAENT](#) 9  
*size of DT\_RELA relocation entry*
- #define [DT\\_STRSZ](#) 10  
*size of the string table*
- #define [DT\\_SYMENT](#) 11  
*size of a symbol table entry*
- #define [DT\\_INIT](#) 12  
*address of initialization function*
- #define [DT\\_FINI](#) 13  
*address of termination function*
- #define [DT\\_SONAME](#) 14  
*name of the shared object*
- #define [DT\\_RPATH](#) 15  
*search library path*
- #define [DT\\_SYMBOLIC](#) 16  
*alter symbol resolution algorithm*
- #define [DT\\_REL](#) 17  
*address of relocation table*
- #define [DT\\_RELSZ](#) 18  
*total size of DT\_REL relocation table*
- #define [DT\\_RELENT](#) 19  
*size of the DT\_REL relocation entry*
- #define [DT\\_PTRREL](#) 20  
*type of relocation entry*
- #define [DT\\_DEBUG](#) 21  
*for debugging purposes*
- #define [DT\\_TEXTREL](#) 22  
*at least on entry changes r/o section*
- #define [DT\\_JMPREL](#) 23  
*address of relocation entries*
- #define [DT\\_BIND\\_NOW](#) 24  
*Process relocations of object.*
- #define [DT\\_INIT\\_ARRAY](#) 25  
*Array with addresses of init fct.*
- #define [DT\\_FINI\\_ARRAY](#) 26

- *Array with addresses of fini fct.*
- #define [DT\\_INIT\\_ARRAYSZ](#) 27  
*Size in bytes of DT\_INIT\_ARRAY.*
- #define [DT\\_FINI\\_ARRAYSZ](#) 28  
*Size in bytes of DT\_FINI\_ARRAY.*
- #define [DT\\_RUNPATH](#) 29  
*Library search path.*
- #define [DT\\_FLAGS](#) 30  
*Flags for the object being loaded.*
- #define [DT\\_ENCODING](#) 32  
*Start of encoded range.*
- #define [DT\\_PREINIT\\_ARRAY](#) 32  
*Array with addresses of preinit fct.*
- #define [DT\\_PREINIT\\_ARRAYSZ](#) 33  
*size in bytes of DT\_PREINIT\_ARRAY*
- #define [DT\\_NUM](#) 34  
*Number used.*
- #define [DT\\_LOOS](#) 0x6000000d  
*Start of OS-specific.*
- #define [DT\\_HIOS](#) 0x6ffff000  
*End of OS-specific.*
- #define [DT\\_LOPROC](#) 0x70000000  
*processor spec.*
- #define [DT\\_HIPROC](#) 0x7ffffff
- #define [DF\\_ORIGIN](#) 0x00000001  
*Object may use DF\_ORIGIN.*
- #define [DF\\_SYMBOLIC](#) 0x00000002  
*Symbol resolutions starts here.*
- #define [DF\\_TEXTREL](#) 0x00000004  
*Object contains text relocations.*
- #define [DF\\_BIND\\_NOW](#) 0x00000008  
*No lazy binding for this object.*
- #define [DF\\_STATIC\\_TLS](#) 0x00000010  
*Module uses the static TLS model.*
- #define [DF\\_1\\_NOW](#) 0x00000001  
*Set RTLD\_NOW for this object.*
- #define [DF\\_1\\_GLOBAL](#) 0x00000002  
*Set RTLD\_GLOBAL for this object.*
- #define [DF\\_1\\_GROUP](#) 0x00000004  
*Set RTLD\_GROUP for this object.*
- #define [DF\\_1\\_NODELETE](#) 0x00000008  
*Set RTLD\_NODELETE for this object.*
- #define [DF\\_1\\_LOADFLTR](#) 0x00000010  
*Trigger filtee loading at runtime.*
- #define [DF\\_1\\_INITFIRST](#) 0x00000020  
*Set RTLD\_INITFIRST for this object.*
- #define [DF\\_1\\_NOOPEN](#) 0x00000040  
*Set RTLD\_NOOPEN for this object.*
- #define [DF\\_1\\_ORIGIN](#) 0x00000080  
*\$ORIGIN must be handled.*



- #define [DF\\_1\\_DIRECT](#) 0x00000100  
*Direct binding enabled.*
- #define [DF\\_1\\_INTERPOSE](#) 0x00000400  
*Object is used to interpose.*
- #define [DF\\_1\\_NODEFLIB](#) 0x00000800  
*Ignore default lib search path.*
- #define [DF\\_1\\_NODUMP](#) 0x00001000  
*Object can't be dldump'ed.*
- #define [DF\\_1\\_CONFALT](#) 0x00002000  
*Configuration alternative created.*
- #define [DF\\_1\\_ENDFILTEE](#) 0x00004000  
*Filtee terminates filters search.*
- #define [DF\\_1\\_DISPRELDNE](#) 0x00008000  
*Disp reloc applied at build time.*
- #define [DF\\_1\\_DISPRELPND](#) 0x00010000  
*Disp reloc applied at run-time.*
- #define [DF\\_P1\\_LAZYLOAD](#) 0x00000001  
*Lazyload following object.*
- #define [DF\\_P1\\_GROUPPERM](#) 0x00000002  
*Symbols from next object are not generally available.*
- #define [R\\_386\\_NONE](#) 0  
*none*
- #define [R\\_386\\_32](#) 1  
*S + A.*
- #define [R\\_386\\_PC32](#) 2  
*S + A - P*
- #define [R\\_386\\_GOT32](#) 3  
*G + A - P.*
- #define [R\\_386\\_PLT32](#) 4  
*L + A - P.*
- #define [R\\_386\\_COPY](#) 5  
*none*
- #define [R\\_386\\_GLOB\\_DAT](#) 6  
*S.*
- #define [R\\_386\\_JMP\\_SLOT](#) 7  
*S.*
- #define [R\\_386\\_RELATIVE](#) 8  
*B + A.*
- #define [R\\_386\\_GOTOFF](#) 9  
*S + A - GOT.*
- #define [R\\_386\\_GOTPC](#) 10  
*GOT + A - P.*
- #define [STB\\_LOCAL](#) 0  
*not visible outside object file*
- #define [STB\\_GLOBAL](#) 1  
*visible to all objects beeing combined*
- #define [STB\\_WEAK](#) 2  
*resemble global symbols*
- #define [STB\\_LOOS](#) 10  
*os specific*
- #define [STB\\_HIOS](#) 12

- os specific*
- #define [STB\\_LOPROC](#) 13
- proc specific*
- #define [STB\\_HIPROC](#) 15
- proc specific*
- #define [STT\\_NOTYPE](#) 0
- symbol's type not specified*
- #define [STT\\_OBJECT](#) 1
- associated with a data object*
- #define [STT\\_FUNC](#) 2
- associated with a function or other code*
- #define [STT\\_SECTION](#) 3
- associated with a section*
- #define [STT\\_FILE](#) 4
- source file name associated with object*
- #define [STT\\_LOOS](#) 10
- os specific*
- #define [STT\\_HIOS](#) 12
- os specific*
- #define [STT\\_LOPROC](#) 13
- proc specific*
- #define [STT\\_HIPROC](#) 15
- proc specific*

## Typedefs

### 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*

### 15.393.1 Detailed Description

ELF definition.

Date

08/18/2000

Author

Frank Mehnert [fm3@os.inf.tu-dresden.de](mailto:fm3@os.inf.tu-dresden.de) Alexander Warg [aw11@os.inf.tu-dresden.de](mailto: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](#).

## 15.394 elf.h

```

00001
00019 /*
00020 * (c) 2008-2009 Author(s)
00021 * economic rights: Technische Universität Dresden (Germany)
00022 * This file is part of TUD:OS and distributed under the terms of the
00023 * GNU Lesser General Public License 2.1.
00024 * Please see the COPYING-LGPL-2.1 file for details.
00025 */
00026
00027 /* (c) 2003-2006 Technische Universitaet Dresden
00028 * This file is part of the exec package, which is distributed under
00029 * the terms of the GNU General Public License 2. Please see the
00030 * COPYING file for details. */
00031
00032 #ifndef _L4_EXEC_ELF_H
00033 #define _L4_EXEC_ELF_H
00034
00035 #include <l4/sys/l4int.h>
00036
00047 typedef l4_uint32_t Elf32_Addr;
00048 typedef l4_uint32_t Elf32_Off;
00049 typedef l4_uint16_t Elf32_Half;
00050 typedef l4_uint32_t Elf32_Word;
00051 typedef l4_int32_t Elf32_Sword;
00052 typedef l4_uint64_t Elf64_Addr;
00053 typedef l4_uint64_t Elf64_Off;
00054 typedef l4_uint16_t Elf64_Half;
00055 typedef l4_uint32_t Elf64_Word;
00056 typedef l4_int32_t Elf64_Sword;
00057 typedef l4_uint64_t Elf64_Xword;
00058 typedef l4_int64_t Elf64_Sxword;
00060
00061 #if L4_MWORD_BITS == 64
00062
00066 #define ElfW(type) _ElfW(Elf, 64, type)
00067 #else
00068 #define ElfW(type) _ElfW(Elf, 32, type)
00069 #endif
00070 #define _ElfW(e,w,t) __ElfW(e, w, _##t)
00071 #define __ElfW(e,w,t) e##w##t
00072
00073 #if defined(ARCH_x86)
00074 #define L4_ARCH_EI_DATA ELFDATA2LSB
00075 #define L4_ARCH_E_MACHINE EM_386
00076 #define L4_ARCH_EI_CLASS ELFCLASS32
00077 #elif defined(ARCH_amd64)
00078 #define L4_ARCH_EI_DATA ELFDATA2LSB
00079 #define L4_ARCH_E_MACHINE EM_X86_64
00080 #define L4_ARCH_EI_CLASS ELFCLASS64
00081 #elif defined(ARCH_arm)
00082 #define L4_ARCH_EI_DATA ELFDATA2LSB

```

```

00083 #define L4_ARCH_E_MACHINE EM_ARM
00084 #define L4_ARCH_EI_CLASS ELFCLASS32
00085 #elif defined(ARCH_arm64)
00086 #define L4_ARCH_EI_DATA ELFDATA2LSB
00087 #define L4_ARCH_E_MACHINE EM_AARCH64
00088 #define L4_ARCH_EI_CLASS ELFCLASS64
00089 #elif defined(ARCH_ppc32)
00090 #define L4_ARCH_EI_DATA ELFDATA2MSB
00091 #define L4_ARCH_E_MACHINE EM_PPC
00092 #define L4_ARCH_EI_CLASS ELFCLASS32
00093 #elif defined(ARCH_sparc)
00094 #define L4_ARCH_EI_DATA ELFDATA2MSB
00095 #define L4_ARCH_E_MACHINE EM_SPARC
00096 #define L4_ARCH_EI_CLASS ELFCLASS32
00097 #elif defined(ARCH_mips)
00098 #define L4_ARCH_EI_DATA ELFDATA2LSB
00099 #define L4_ARCH_E_MACHINE EM_MIPS
00100 #ifdef __mips64
00101 #define L4_ARCH_EI_CLASS ELFCLASS64
00102 #else
00103 #define L4_ARCH_EI_CLASS ELFCLASS32
00104 #endif
00105 #else
00106 #warning elf.h: Unsupported build architecture!
00107 #endif
00108
00109
00110 /*****
00111 * ELF Header - figure 1-3, page 1-3 *
00112 *****/
00113
00117
00118 #define EI_NIDENT 16
00122 typedef struct {
00123 unsigned char e_ident[EI_NIDENT];
00124 Elf32_Half e_type;
00125 Elf32_Half e_machine;
00126 Elf32_Word e_version;
00127 Elf32_Addr e_entry;
00128 Elf32_Off e_phoff;
00129 Elf32_Off e_shoff;
00130 Elf32_Word e_flags;
00131 Elf32_Half e_ehsize;
00132 Elf32_Half e_phentsize;
00133 Elf32_Half e_phnum;
00134 Elf32_Half e_shentsize;
00135 Elf32_Half e_shnum;
00136 Elf32_Half e_shstrndx;
00137 } Elf32_Ehdr;
00138
00142 typedef struct {
00143 unsigned char e_ident[EI_NIDENT];
00144 Elf64_Half e_type;
00145 Elf64_Half e_machine;
00146 Elf64_Word e_version;
00147 Elf64_Addr e_entry;
00148 Elf64_Off e_phoff;
00149 Elf64_Off e_shoff;
00150 Elf64_Word e_flags;
00151 Elf64_Half e_ehsize;
00152 Elf64_Half e_phentsize;
00153 Elf64_Half e_phnum;
00154 Elf64_Half e_shentsize;
00155 Elf64_Half e_shnum;
00156 Elf64_Half e_shstrndx;
00157 } Elf64_Ehdr;
00158
00159 #define EI_CLASS 4
00160 #define ELFCLASSNONE 0
00161 #define ELFCLASS32 1
00162 #define ELFCLASS64 2
00163 #define ELFCLASSNUM 3
00165 #define EI_DATA 5
00166 #define ELFDATANONE 0
00167 #define ELFDATA2LSB 1
00168 #define ELFDATA2MSB 2
00169 #define ELFDATANUM 3
00170
00171 #define EI_VERSION 6
00174 #define EI_OSABI 7
00175 #define ELFOSABI_NONE 0
00176 #define ELFOSABI_SYSV 0
00177 #define ELFOSABI_HPUX 1
00178 #define ELFOSABI_NETBSD 2
00179 #define ELFOSABI_LINUX 3
00180 #define ELFOSABI_SOLARIS 6
00181 #define ELFOSABI_AIX 7

```

```
00182 #define ELFOSABI_IRIX 8
00183 #define ELFOSABI_FREEBSD 9
00184 #define ELFOSABI_TRU64 10
00185 #define ELFOSABI_MODESTO 11
00186 #define ELFOSABI_OPENBSD 12
00187 #define ELFOSABI_ARM 97
00188 #define ELFOSABI_STANDALONE 255
00190 #define EI_ABIVERSION 8
00192 #define EI_PAD 9
00194 /* object file type - page 1-3 (e_type) */
00195
00196 #define ET_NONE 0
00197 #define ET_REL 1
00198 #define ET_EXEC 2
00199 #define ET_DYN 3
00200 #define ET_CORE 4
00201 #define ET_LOPROC 0xffff00
00202 #define ET_HIPROC 0xffff
00204 /* required architecture - page 1-4 (e_machine) */
00205
00206 #define EM_NONE 0
00207 #define EM_M32 1
00208 #define EM_SPARC 2
00209 #define EM_386 3
00210 #define EM_68K 4
00211 #define EM_88K 5
00212 #define EM_860 7
00213 #define EM_MIPS 8
00214 #define EM_MIPS_RS4_BE 10
00215 #define EM_SPARC64 11
00216 #define EM_PARISC 15
00217 #define EM_VPP500 17
00218 #define EM_SPARC32PLUS 18
00219 #define EM_960 19
00220 #define EM_PPC 20
00221 #define EM_V800 36
00222 #define EM_FR20 37
00223 #define EM_RH32 38
00224 #define EM_RCE 39
00225 #define EM_ARM 40
00226 #define EM_ALPHA 41
00227 #define EM_SH 42
00228 #define EM_SPARCV9 43
00229 #define EM_TRICORE 44
00230 #define EM_ARC 45
00231 #define EM_H8_300 46
00232 #define EM_H8_300H 47
00233 #define EM_H8S 48
00234 #define EM_H8_500 49
00235 #define EM_IA_64 50
00236 #define EM_MIPS_X 51
00237 #define EM_COLDFIRE 52
00238 #define EM_68HC12 53
00239 #define EM_X86_64 62
00240 #define EM_PDSP 63
00241 #define EM_FX66 66
00242 #define EM_ST9PLUS 67
00243 #define EM_ST7 68
00244 #define EM_68HC16 69
00245 #define EM_68HC11 70
00246 #define EM_68HC08 71
00247 #define EM_68HC05 72
00248 #define EM_SVX 73
00249 #define EM_ST19 74
00250 #define EM_VAX 75
00251 #define EM_CRIS 76
00252 #define EM_JAVELIN 77
00253 #define EM_FIREPATH 78
00254 #define EM_ZSP 79
00255 #define EM_MMIX 80
00256 #define EM_HUANY 81
00257 #define EM_PRISM 82
00258 #define EM_AVR 83
00259 #define EM_FR30 84
00260 #define EM_D10V 85
00261 #define EM_D30V 86
00262 #define EM_V850 87
00263 #define EM_M32R 88
00264 #define EM_MN10300 89
00265 #define EM_MN10200 90
00266 #define EM_PJ 91
00267 #define EM_OPENRISC 92
00268 #define EM_ARC_A5 93
00269 #define EM_XTENSA 94
00270 #define EM_ALTERA_NIOS2 113
00271 #define EM_AARCH64 183
00272 #define EM_TILEPRO 188
```

```

00273 #define EM_MICROBLAZE 189
00274 #define EM_TILEGX 191
00275 #define EM_NUM 192
00276
00277 #if 0
00278 #define EM_ALPHA 0x9026 /* interim value used by Linux until the
00279 committee comes up with a final number */
00280 #define EM_S390 0xA390 /* interim value used for IBM S390 */
00281 #endif
00282
00283 /* object file version - page 1-4 (e_version) */
00284
00285 #define EV_NONE 0
00286 #define EV_CURRENT 1
00288 /* e_ident[] Identification Indexes - figure 1-4, page 1-5 */
00289
00290 #define EI_MAG0 0
00291 #define EI_MAG1 1
00292 #define EI_MAG2 2
00293 #define EI_MAG3 3
00294 #define EI_CLASS 4
00295 #define EI_DATA 5
00296 #define EI_VERSION 6
00297 #define EI_OSABI 7
00298 #define EI_ABIVERSION 8
00299 #define EI_PAD 9
00301 /* magic number - page 1-5 */
00302
00303 #define ELFMAG0 0x7f
00304 #define ELFMAG1 'E'
00305 #define ELFMAG2 'L'
00306 #define ELFMAG3 'F'
00308 /* file class or capacity - page 1-6 */
00309
00310 #define ELFCLASSNONE 0
00311 #define ELFCLASS32 1
00312 #define ELFCLASS64 2
00314 /* data encoding - page 1-6 */
00315
00316 #define ELFDATANONE 0
00317 #define ELFDATA2LSB 1
00318 #define ELFDATA2MSB 2
00320 /* Identify operating system and ABI to which the object is targeted */
00321
00322 #define ELFOSABI_SYSV 0
00323 #define ELFOSABI_HPUX 1
00324 #define ELFOSABI_STANDALONE 255
00327 /******
00328 /* Sections - page 1-8 */
00329 /******
00330
00331 /* special section indexes */
00332
00333 #define SHN_UNDEF 0
00334 #define SHN_LORESERVE 0xff00
00335 #define SHN_LOPROC 0xff00
00336 #define SHN_HIPROC 0xffff
00337 #define SHN_ABS 0xffff1
00338 #define SHN_COMMON 0xffff2
00339 #define SHN_HIRESERVE 0xfffff
00342 typedef struct {
00343 Elf32_Word sh_name;
00344 Elf32_Word sh_type;
00345 Elf32_Word sh_flags;
00346 Elf32_Addr sh_addr;
00347 Elf32_Off sh_offset;
00348 Elf32_Word sh_size;
00349 Elf32_Word sh_link;
00350 Elf32_Word sh_info;
00351 Elf32_Word sh_addralign;
00352 Elf32_Word sh_entsize;
00353 } Elf32_Shdr;
00354
00356 typedef struct {
00357 Elf64_Word sh_name;
00358 Elf64_Word sh_type;
00359 Elf64_Xword sh_flags;
00360 Elf64_Addr sh_addr;
00361 Elf64_Off sh_offset;
00362 Elf64_Xword sh_size;
00363 Elf64_Word sh_link;
00364 Elf64_Word sh_info;
00365 Elf64_Xword sh_addralign;
00366 Elf64_Xword sh_entsize;
00367 } Elf64_Shdr;
00368
00369 /* section type - figure 1-10, page 1-10 */

```

```

00370
00371 #define SHT_NULL 0
00372 #define SHT_PROGBITS 1
00373 #define SHT_SYMTAB 2
00374 #define SHT_STRTAB 3
00375 #define SHT_RELA 4
00376 #define SHT_HASH 5
00377 #define SHT_DYNAMIC 6
00378 #define SHT_NOTE 7
00379 #define SHT_NOBITS 8
00380 #define SHT_REL 9
00381 #define SHT_SHLIB 10
00382 #define SHT_DYNSYM 11
00383 #define SHT_INIT_ARRAY 14
00384 #define SHT_FINI_ARRAY 15
00385 #define SHT_PREINIT_ARRAY 16
00386 #define SHT_GROUP 17
00387 #define SHT_SYMTAB_SHNDX 18
00388 #define SHT_NUM 19
00389 #define SHT_LOOS 0x60000000
00390 #define SHT_HIOS 0x6fffffff
00391 #define SHT_LOPROC 0x70000000
00392 #define SHT_HIPROC 0x7fffffff
00393 #define SHT_LOUSER 0x80000000
00394 #define SHT_HIUSER 0xffffffff
00395
00396 /* section attribute flags - page 1-12, figure 1-12 */
00397
00398 #define SHF_WRITE 0x1
00399 #define SHF_ALLOC 0x2
00400 #define SHF_EXECINSTR 0x4
00401 #define SHF_MERGE 0x10
00402 #define SHF_STRINGS 0x20
00403 #define SHF_INFO_LINK 0x40
00404 #define SHF_LINK_ORDER 0x80
00405 #define SHF_OS_NONCONFORMING 0x100
00407 #define SHF_GROUP 0x200
00408 #define SHF_TLS 0x400
00409 #define SHF_MASKOS 0x0ff00000
00410 #define SHF_MASKPROC 0xf0000000
00413 /*****
00414 */
00415 /*****
00416
00417 typedef struct {
00418 Elf32_Word p_type;
00419 Elf32_Off p_offset;
00420 Elf32_Addr p_vaddr;
00421 Elf32_Addr p_paddr;
00422 Elf32_Word p_filesz;
00423 Elf32_Word p_memsz;
00424 Elf32_Word p_flags;
00425 Elf32_Word p_align;
00426 } Elf32_Phdr;
00427
00428
00429 typedef struct {
00430 Elf64_Word p_type;
00431 Elf64_Word p_flags;
00432 Elf64_Off p_offset;
00433 Elf64_Addr p_vaddr;
00434 Elf64_Addr p_paddr;
00435 Elf64_Xword p_filesz;
00436 Elf64_Xword p_memsz;
00437 Elf64_Xword p_align;
00438 } Elf64_Phdr;
00439
00440
00441 /* segment types - figure 2-2, page 2-3 */
00442
00443 #define PT_NULL 0
00444 #define PT_LOAD 1
00445 #define PT_DYNAMIC 2
00446 #define PT_INTERP 3
00447 #define PT_NOTE 4
00448 #define PT_SHLIB 5
00449 #define PT_PHDR 6
00450 #define PT_TLS 7
00451 #define PT_NUM 8
00452 #define PT_LOOS 0x60000000
00453 #define PT_HIOS 0x6fffffff
00454 #define PT_LOPROC 0x70000000
00455 #define PT_HIPROC 0x7fffffff
00457 #define PT_GNU_EH_FRAME (PT_LOOS + 0x474e550)
00458 #define PT_GNU_STACK (PT_LOOS + 0x474e551)
00459 #define PT_GNU_RELRO (PT_LOOS + 0x474e552)
00461 #define PT_L4_STACK (PT_LOOS + 0x12)
00462 #define PT_L4_KIP (PT_LOOS + 0x13)
00463 #define PT_L4_AUX (PT_LOOS + 0x14)

```

```

00465 /* segment permissions - page 2-3 */
00466
00467 #define PF_X 0x1
00468 #define PF_W 0x2
00469 #define PF_R 0x4
00470 #define PF_MASKOS 0x0ff00000
00471 #define PF_MASKPROC 0x7fffffff
00472
00473 /* Legal values for note segment descriptor types for core files. */
00474
00475 #define NT_PRSTATUS 1
00476 #define NT_FPREGSET 2
00477 #define NT_PRPSINFO 3
00478 #define NT_PRXREG 4
00479 #define NT_TASKSTRUCT 4
00480 #define NT_PLATFORM 5
00481 #define NT_AUXV 6
00482 #define NT_GWINDOWS 7
00483 #define NT_ASRS 8
00484 #define NT_PSTATUS 10
00485 #define NT_PSINFO 13
00486 #define NT_PRCRED 14
00487 #define NT_UTSNAME 15
00488 #define NT_LWPSTATUS 16
00489 #define NT_LWPINFO 17
00490 #define NT_PRFPXREG 20
00492 /* Legal values for the note segment descriptor types for object files. */
00493
00494 #define NT_VERSION 1
00496 /* Dynamic structure - figure 2-9, page 2-12 */
00497
00498 typedef struct {
00500 Elf32_Sword d_tag;
00501 union {
00502 Elf32_Word d_val;
00503 Elf32_Addr d_ptr;
00504 } d_un;
00505 } Elf32_Dyn;
00506
00508 typedef struct {
00509 Elf64_Sxword d_tag;
00510 union {
00511 Elf64_Xword d_val;
00512 Elf64_Addr d_ptr;
00513 } d_un;
00514 } Elf64_Dyn;
00515
00518 #define DT_NULL 0
00519 #define DT_NEEDED 1
00520 #define DT_PLTRELSZ 2
00521 #define DT_PLTGOT 3
00522 #define DT_HASH 4
00523 #define DT_STRTAB 5
00524 #define DT_SYMTAB 6
00525 #define DT_RELA 7
00526 #define DT_RELASZ 8
00527 #define DT_RELAENT 9
00528 #define DT_STRSZ 10
00529 #define DT_SYMENT 11
00530 #define DT_INIT 12
00531 #define DT_FINI 13
00532 #define DT_SONAME 14
00533 #define DT_RPATH 15
00534 #define DT_SYMBOLIC 16
00535 #define DT_REL 17
00536 #define DT_RELSZ 18
00537 #define DT_RELENT 19
00538 #define DT_PIRREL 20
00539 #define DT_DEBUG 21
00540 #define DT_TEXTREL 22
00541 #define DT_JMPREL 23
00542 #define DT_BIND_NOW 24
00543 #define DT_INIT_ARRAY 25
00544 #define DT_FINI_ARRAY 26
00545 #define DT_INIT_ARRAYSZ 27
00546 #define DT_FINI_ARRAYSZ 28
00547 #define DT_RUNPATH 29
00548 #define DT_FLAGS 30
00549 #define DT_ENCODING 32
00550 #define DT_PREINIT_ARRAY 32
00551 #define DT_PREINIT_ARRAYSZ 33
00552 #define DT_NUM 34
00553 #define DT_LOOS 0x6000000d
00554 #define DT_HIOS 0x6ffff000
00555 #define DT_LOPROC 0x70000000
00556 #define DT_HIPROC 0x7fffffff
00558 /* Values of 'd_un.d_val' in the DT_FLAGS entry. */

```



```

00559 #define DF_ORIGIN 0x00000001
00560 #define DF_SYMBOLIC 0x00000002
00561 #define DF_TEXTREL 0x00000004
00562 #define DF_BIND_NOW 0x00000008
00563 #define DF_STATIC_TLS 0x00000010
00565 /* State flags selectable in the 'd_un.d_val' element of the DT_FLAGS_1
00566 entry in the dynamic section. */
00567 #define DF_1_NOW 0x00000001
00568 #define DF_1_GLOBAL 0x00000002
00569 #define DF_1_GROUP 0x00000004
00570 #define DF_1_NODELETE 0x00000008
00571 #define DF_1_LOADFLTR 0x00000010
00572 #define DF_1_INITFIRST 0x00000020
00573 #define DF_1_NOOPEN 0x00000040
00574 #define DF_1_ORIGIN 0x00000080
00575 #define DF_1_DIRECT 0x00000100
00576 #define DF_1_TRANS 0x00000200
00577 #define DF_1_INTERPOSE 0x00000400
00578 #define DF_1_NODEFLIB 0x00000800
00579 #define DF_1_NODUMP 0x00001000
00580 #define DF_1_CONFALT 0x00002000
00581 #define DF_1_ENDFILTEE 0x00004000
00582 #define DF_1_DISPRELDNE 0x00008000
00583 #define DF_1_DISPRELPND 0x00010000
00585 /* Flags for the feature selection in DT_FEATURE_1. */
00586 #define DTF_1_PARINIT 0x00000001
00587 #define DTF_1_CONFEXP 0x00000002
00588
00589 /* Flags in the DT_POSFLAG_1 entry effecting only the next DT_* entry. */
00590 #define DF_P1_LAZYLOAD 0x00000001
00591 #define DF_P1_GROUPPERM 0x00000002
00594 /* Relocation - page 1-21, figure 1-20 */
00595
00596 typedef struct {
00597 Elf32_Addr r_offset;
00598 Elf32_Word r_info;
00599 } Elf32_Rel;
00600
00601 typedef struct {
00602 Elf32_Addr r_offset;
00603 Elf32_Word r_info;
00604 Elf32_Sword r_addend;
00605 } Elf32_Rela;
00606
00607 typedef struct {
00608 Elf64_Addr r_offset;
00609 Elf64_Xword r_info;
00610 } Elf64_Rel;
00611
00612 typedef struct {
00613 Elf64_Addr r_offset;
00614 Elf64_Xword r_info;
00615 Elf64_Sxword r_addend;
00616 } Elf64_Rela;
00617
00618 #define ELF32_R_SYM(i) ((i)>>8)
00619 #define ELF32_R_TYPE(i) ((unsigned char)(i))
00620 #define ELF32_R_INFO(s,t) (((s)<<8)+(unsigned char)(t))
00621
00622 #define ELF64_R_SYM(i) ((i)>>32)
00623 #define ELF64_R_TYPE(i) ((i)&0xffffffffL)
00624 #define ELF64_R_INFO(s,t) (((s)<<32)+(t)&0xffffffffL)
00625
00626 /* Relocation types (processor specific) - page 1-23, figure 1-22 */
00627
00628 #define R_386_NONE 0
00629 #define R_386_32 1
00630 #define R_386_PC32 2
00631 #define R_386_GOT32 3
00632 #define R_386_PLT32 4
00633 #define R_386_COPY 5
00634 #define R_386_GLOB_DAT 6
00635 #define R_386_JMP_SLOT 7
00636 #define R_386_RELATIVE 8
00637 #define R_386_GOTOFF 9
00638 #define R_386_GOTPC 10
00639 #define R_386_32PLT 11
00640 #define R_386_TLS_TPOFF 14 /* Offset in static TLS block */
00641 #define R_386_TLS_IE 15 /* Address of GOT entry for static TLS
00642 block offset */
00643 #define R_386_TLS_GOTIE 16 /* GOT entry for static TLS block
00644 offset */
00645 #define R_386_TLS_LE 17 /* Offset relative to static TLS
00646 block */
00647 #define R_386_TLS_GD 18 /* Direct 32 bit for GNU version of
00648 general dynamic thread local data */
00649 #define R_386_TLS_LDM 19 /* Direct 32 bit for GNU version of

```

```

00650 local dynamic thread local data
00651 in LE code */
00652 #define R_386_16 20
00653 #define R_386_PC16 21
00654 #define R_386_8 22
00655 #define R_386_PC8 23
00656 #define R_386_TLS_GD_32 24 /* Direct 32 bit for general dynamic
00657 thread local data */
00658 #define R_386_TLS_GD_PUSH 25 /* Tag for pushl in GD TLS code */
00659 #define R_386_TLS_GD_CALL 26 /* Relocation for call to
00660 __tls_get_addr() */
00661 #define R_386_TLS_GD_POP 27 /* Tag for popl in GD TLS code */
00662 #define R_386_TLS_LDM_32 28 /* Direct 32 bit for local dynamic
00663 thread local data in LE code */
00664 #define R_386_TLS_LDM_PUSH 29 /* Tag for pushl in LDM TLS code */
00665 #define R_386_TLS_LDM_CALL 30 /* Relocation for call to
00666 __tls_get_addr() in LDM code */
00667 #define R_386_TLS_LDM_POP 31 /* Tag for popl in LDM TLS code */
00668 #define R_386_TLS_LDO_32 32 /* Offset relative to TLS block */
00669 #define R_386_TLS_IE_32 33 /* GOT entry for negated static TLS
00670 block offset */
00671 #define R_386_TLS_LE_32 34 /* Negated offset relative to static
00672 TLS block */
00673 #define R_386_TLS_DTPMOD32 35 /* ID of module containing symbol */
00674 #define R_386_TLS_DTPOFF32 36 /* Offset in TLS block */
00675 #define R_386_TLS_TPOFF32 37 /* Negated offset in static TLS block */
00676 /* Keep this the last entry. */
00677 #define R_386_NUM 38
00678
00679 /* ARM specific declarations */
00680
00681 /* Processor specific flags for the ELF header e_flags field. */
00682 #define EF_ARM_RELEXEC 0x01
00683 #define EF_ARM_HASENTRY 0x02
00684 #define EF_ARM_INTERWORK 0x04
00685 #define EF_ARM_APCS_26 0x08
00686 #define EF_ARM_APCS_FLOAT 0x10
00687 #define EF_ARM_PIC 0x20
00688 #define EF_ARM_ALIGN8 0x40 /* 8-bit structure alignment is in use */
00689 #define EF_ARM_NEW_ABI 0x80
00690 #define EF_ARM_OLD_ABI 0x100
00691
00692 /* Other constants defined in the ARM ELF spec. version B-01. */
00693 /* NB. These conflict with values defined above. */
00694 #define EF_ARM_SYMSARESORTED 0x04
00695 #define EF_ARM_DYN_SYMS_USE_SEGIDX 0x08
00696 #define EF_ARM_MAPSYMSFIRST 0x10
00697 #define EF_ARM_EABIMASK 0xFF000000
00698
00699 #define EF_ARM_EABI_VERSION(flags) ((flags) & EF_ARM_EABIMASK)
00700 #define EF_ARM_EABI_UNKNOWN 0x00000000
00701 #define EF_ARM_EABI_VER1 0x01000000
00702 #define EF_ARM_EABI_VER2 0x02000000
00703
00704 /* Additional symbol types for Thumb */
00705 #define STT_ARM_TFUNC 0xd
00706
00707 /* ARM-specific values for sh_flags */
00708 #define SHF_ARM_ENTRYSECT 0x10000000 /* Section contains an entry point */
00709 #define SHF_ARM_COMDEF 0x80000000 /* Section may be multiply defined
00710 in the input to a link step */
00711
00712 /* ARM-specific program header flags */
00713 #define PF_ARM_SB 0x10000000 /* Segment contains the location
00714 addressed by the static base */
00715
00716 /* ARM relocs. */
00717 #define R_ARM_NONE 0 /* No reloc */
00718 #define R_ARM_PC24 1 /* PC relative 26 bit branch */
00719 #define R_ARM_ABS32 2 /* Direct 32 bit */
00720 #define R_ARM_REL32 3 /* PC relative 32 bit */
00721 #define R_ARM_PC13 4
00722 #define R_ARM_ABS16 5 /* Direct 16 bit */
00723 #define R_ARM_ABS12 6 /* Direct 12 bit */
00724 #define R_ARM_THM_ABS5 7
00725 #define R_ARM_ABS8 8 /* Direct 8 bit */
00726 #define R_ARM_SBREL32 9
00727 #define R_ARM_THM_PC22 10
00728 #define R_ARM_THM_PC8 11
00729 #define R_ARM_AMP_VCALL9 12
00730 #define R_ARM_SWI24 13
00731 #define R_ARM_THM_SWI8 14
00732 #define R_ARM_XPC25 15
00733 #define R_ARM_THM_XPC22 16
00734 #define R_ARM_COPY 20 /* Copy symbol at runtime */
00735 #define R_ARM_GLOB_DAT 21 /* Create GOT entry */
00736 #define R_ARM_JUMP_SLOT 22 /* Create PLT entry */

```

```

00737 #define R_ARM_RELATIVE 23 /* Adjust by program base */
00738 #define R_ARM_GOTOFF 24 /* 32 bit offset to GOT */
00739 #define R_ARM_GOTPC 25 /* 32 bit PC relative offset to GOT */
00740 #define R_ARM_GOT32 26 /* 32 bit GOT entry */
00741 #define R_ARM_PLT32 27 /* 32 bit PLT address */
00742 #define R_ARM_ALU_PCREL_7_0 32
00743 #define R_ARM_ALU_PCREL_15_8 33
00744 #define R_ARM_ALU_PCREL_23_15 34
00745 #define R_ARM_LDR_SBREL_11_0 35
00746 #define R_ARM_ALU_SBREL_19_12 36
00747 #define R_ARM_ALU_SBREL_27_20 37
00748 #define R_ARM_GNU_VTENTRY 100
00749 #define R_ARM_GNU_VTINHERIT 101
00750 #define R_ARM_THM_PC11 102 /* thumb unconditional branch */
00751 #define R_ARM_THM_PC9 103 /* thumb conditional branch */
00752 #define R_ARM_RXPC25 249
00753 #define R_ARM_RSBREL32 250
00754 #define R_ARM_THM_RPC22 251
00755 #define R_ARM_RREL32 252
00756 #define R_ARM_RABS22 253
00757 #define R_ARM_RPC24 254
00758 #define R_ARM_RBASE 255
00759 /* Keep this the last entry. */
00760 #define R_ARM_NUM 256
00761
00762 /* AMD x86-64 relocations. */
00763 #define R_X86_64_NONE 0 /* No reloc */
00764 #define R_X86_64_64 1 /* Direct 64 bit */
00765 #define R_X86_64_PC32 2 /* PC relative 32 bit signed */
00766 #define R_X86_64_GOT32 3 /* 32 bit GOT entry */
00767 #define R_X86_64_PLT32 4 /* 32 bit PLT address */
00768 #define R_X86_64_COPY 5 /* Copy symbol at runtime */
00769 #define R_X86_64_GLOB_DAT 6 /* Create GOT entry */
00770 #define R_X86_64_JUMP_SLOT 7 /* Create PLT entry */
00771 #define R_X86_64_RELATIVE 8 /* Adjust by program base */
00772 #define R_X86_64_GOTPCREL 9 /* 32 bit signed PC relative
00773 offset to GOT */
00774 #define R_X86_64_32 10 /* Direct 32 bit zero extended */
00775 #define R_X86_64_32S 11 /* Direct 32 bit sign extended */
00776 #define R_X86_64_16 12 /* Direct 16 bit zero extended */
00777 #define R_X86_64_PC16 13 /* 16 bit sign extended pc relative */
00778 #define R_X86_64_8 14 /* Direct 8 bit sign extended */
00779 #define R_X86_64_PC8 15 /* 8 bit sign extended pc relative */
00780 #define R_X86_64_DTPMOD64 16 /* ID of module containing symbol */
00781 #define R_X86_64_DTPOFF64 17 /* Offset in module's TLS block */
00782 #define R_X86_64_TPOFF64 18 /* Offset in initial TLS block */
00783 #define R_X86_64_TLSGD 19 /* 32 bit signed PC relative offset
00784 to two GOT entries for GD symbol */
00785 #define R_X86_64_TLSLD 20 /* 32 bit signed PC relative offset
00786 to two GOT entries for LD symbol */
00787 #define R_X86_64_DTPOFF32 21 /* Offset in TLS block */
00788 #define R_X86_64_GOTTPOFF 22 /* 32 bit signed PC relative offset
00789 to GOT entry for IE symbol */
00790 #define R_X86_64_TPOFF32 23 /* Offset in initial TLS block */
00791
00792 #define R_X86_64_NUM 24
00793
00794 /* Symbol Table Entry - page 1-17, figure 1-16 */
00795
00796 #define STN_UNDEF 0
00797
00798 typedef struct {
00800 Elf32_Word st_name;
00801 Elf32_Addr st_value;
00802 Elf32_Word st_size;
00803 unsigned char st_info;
00804 unsigned char st_other;
00805 Elf32_Half st_shndx;
00806 } Elf32_Sym;
00807
00808 typedef struct {
00810 Elf64_Word st_name;
00811 unsigned char st_info;
00812 unsigned char st_other;
00813 Elf64_Half st_shndx;
00814 Elf64_Addr st_value;
00815 Elf64_Xword st_size;
00816 } Elf64_Sym;
00817
00818 #define ELF32_ST_BIND(i) ((i)>>4)
00819 #define ELF32_ST_TYPE(i) ((i)&0xf)
00820 #define ELF32_ST_INFO(b,t) (((b)<<4)+((t)&0xf))
00821
00822 #define ELF64_ST_BIND(i) ((i)>>4)
00823 #define ELF64_ST_TYPE(i) ((i)&0xf)
00824 #define ELF64_ST_INFO(b,t) (((b)<<4)+((t)&0xf))
00825

```

```

00826 /* Symbol Binding - page 1-18, figure 1-17 */
00827
00828 #define STB_LOCAL 0
00829 #define STB_GLOBAL 1
00830 #define STB_WEAK 2
00831 #define STB_LOOS 10
00832 #define STB_HIOS 12
00833 #define STB_LOPROC 13
00834 #define STB_HIPROC 15
00836 /* Symbol Types - page 1-19, figure 1-18 */
00837
00838 #define STT_NOTYPE 0
00839 #define STT_OBJECT 1
00840 #define STT_FUNC 2
00841 #define STT_SECTION 3
00842 #define STT_FILE 4
00843 #define STT_LOOS 10
00844 #define STT_HIOS 12
00845 #define STT_LOPROC 13
00846 #define STT_HIPROC 15
00848 enum Elf_ATs
00849 {
00850 AT_NULL = 0,
00851 AT_IGNORE = 1,
00852 AT_EXECD = 2,
00853 AT_PHDR = 3,
00854 AT_PHENT = 4,
00855 AT_PHNUM = 5,
00856 AT_PAGESZ = 6,
00857 AT_BASE = 7,
00858 AT_FLAGS = 8,
00859 AT_ENTRY = 9,
00860 AT_NOTELF = 10,
00861 AT_UID = 11,
00862 AT_EUID = 12,
00863 AT_GID = 13,
00864 AT_EGID = 14,
00865
00866 AT_L4_AUX = 0xf0,
00867 AT_L4_ENV = 0xf1,
00868 };
00869
00870 typedef struct Elf32_Auxv
00871 {
00872 Elf32_Word atype;
00873 Elf32_Word avalue;
00874 } Elf32_Auxv;
00875
00876 typedef struct Elf64_Auxv
00877 {
00878 Elf64_Word atype;
00879 Elf64_Word avalue;
00880 } Elf64_Auxv;
00881
00882 /* Some helpers */
00883 static inline int l4util_elf_check_magic(ElfW(Ehdr) *hdr);
00884 static inline int l4util_elf_check_arch(ElfW(Ehdr) *hdr);
00885 static inline ElfW(Phdr) *l4util_elf_phdr(ElfW(Ehdr) *hdr);
00886
00887
00888 /* Implemeantions */
00889 static inline
00890 int l4util_elf_check_magic(ElfW(Ehdr) *hdr)
00891 {
00892 return hdr->e_ident[EI_MAG0] == ELFMAG0
00893 && hdr->e_ident[EI_MAG1] == ELFMAG1
00894 && hdr->e_ident[EI_MAG2] == ELFMAG2
00895 && hdr->e_ident[EI_MAG3] == ELFMAG3;
00896 }
00897
00898 static inline
00899 int l4util_elf_check_arch(ElfW(Ehdr) *hdr)
00900 {
00901 return hdr->e_ident[EI_CLASS] == L4_ARCH_EI_CLASS
00902 && hdr->e_ident[EI_DATA] == L4_ARCH_EI_DATA
00903 && hdr->e_machine == L4_ARCH_E_MACHINE;
00904 }
00905
00906 static inline
00907 ElfW(Phdr) *l4util_elf_phdr(ElfW(Ehdr) *hdr)
00908 {
00909 return (ElfW(Phdr) *) ((char *)hdr + hdr->e_phoff);
00910 }
00913 #endif /* _L4_EXEC_ELF_H */

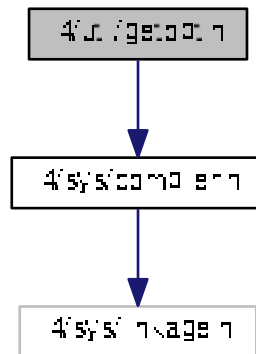
```

## 15.395 l4/util/getopt.h File Reference

getopt

```
#include <l4/sys/compiler.h>
```

Include dependency graph for getopt.h:



### 15.395.1 Detailed Description

getopt

Definition in file [getopt.h](#).

## 15.396 getopt.h

```

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 * 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 #ifndef _GETOPT_H
00014 #define _GETOPT_H
00015
00016 #ifndef NULL
00017 #define NULL 0
00018 #endif
00019
00020 #include <l4/sys/compiler.h>
00021
00022 EXTERN_C_BEGIN
00023
00024 /* For communication from 'getopt' to the caller.
00025 * When 'getopt' finds an option that takes an argument,
00026 * the argument value is returned here.
00027 * Also, when 'ordering' is RETURN_IN_ORDER,
00028 * each non-option ARGV-element is returned here. */
00029
00030 extern char *optarg;

```

```

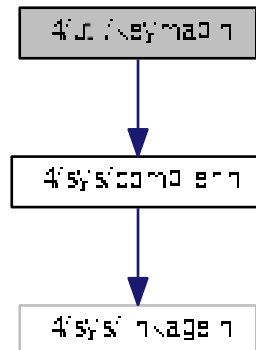
00031
00032 /* Index in ARGV of the next element to be scanned.
00033 This is used for communication to and from the caller
00034 and for communication between successive calls to 'getopt'.
00035
00036 On entry to 'getopt', zero means this is the first call; initialize.
00037
00038 When 'getopt' returns -1, this is the index of the first of the
00039 non-option elements that the caller should itself scan.
00040
00041 Otherwise, 'optind' communicates from one call to the next
00042 how much of ARGV has been scanned so far. */
00043
00044 extern int optind;
00045
00046 /* Callers store zero here to inhibit the error message 'getopt' prints
00047 for unrecognized options. */
00048
00049 extern int opterr;
00050
00051 /* Set to an option character which was unrecognized. */
00052
00053 extern int optopt;
00054
00055 /* Describe the long-named options requested by the application.
00056 The LONG_OPTIONS argument to getopt_long or getopt_long_only is a vector
00057 of 'struct option' terminated by an element containing a name which is
00058 zero.
00059
00060 The field 'has_arg' is:
00061 no_argument (or 0) if the option does not take an argument,
00062 required_argument (or 1) if the option requires an argument,
00063 optional_argument (or 2) if the option takes an optional argument.
00064
00065 If the field 'flag' is not NULL, it points to a variable that is set
00066 to the value given in the field 'val' when the option is found, but
00067 left unchanged if the option is not found.
00068
00069 To have a long-named option do something other than set an 'int' to
00070 a compiled-in constant, such as set a value from 'optarg', set the
00071 option's 'flag' field to zero and its 'val' field to a nonzero
00072 value (the equivalent single-letter option character, if there is
00073 one). For long options that have a zero 'flag' field, 'getopt'
00074 returns the contents of the 'val' field. */
00075
00076 struct option
00077 {
00078 const char *name;
00079 /* has_arg can't be an enum because some compilers complain about
00080 type mismatches in all the code that assumes it is an int. */
00081 int has_arg;
00082 int *flag;
00083 int val;
00084 };
00085
00086 /* Names for the values of the 'has_arg' field of 'struct option'. */
00087
00088 #define no_argument 0
00089 #define required_argument 1
00090 #define optional_argument 2
00091
00092 L4_CV int getopt (int argc, char *const *argv, const char *shortopts);
00093
00094 L4_CV int getopt_long (int argc, char *const *argv, const char *shortopts,
00095 const struct option *longopts, int *longind);
00096 L4_CV int getopt_long_only (int argc, char *const *argv,
00097 const char *shortopts,
00098 const struct option *longopts, int *longind);
00099
00100 L4_CV int _getopt_internal (int argc, char *const *argv,
00101 const char *shortopts,
00102 const struct option *longopts, int *longind,
00103 int long_only);
00104
00105 EXTERN_C_END
00106
00107 #endif /* _GETOPT_H */

```

## 15.397 l4/util/keymap.h File Reference

Event to ASCII key mapping.

```
#include <l4/sys/compiler.h>
Include dependency graph for keymap.h:
```



### 15.397.1 Detailed Description

Event to ASCII key mapping.

Definition in file [keymap.h](#).

## 15.398 keymap.h

```

00001
00005 /*
00006 * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007 * economic rights: Technische Universität Dresden (Germany)
00008 * This file is part of TUD:OS and distributed under the terms of the
00009 * GNU Lesser General Public License 2.1.
00010 * Please see the COPYING-LGPL-2.1 file for details.
00011 */
00012 #ifndef __L4UTIL__KEYMAP_H__
00013 #define __L4UTIL__KEYMAP_H__
00014
00015 #include <l4/sys/compiler.h>
00016
00017 __BEGIN_DECLS
00018
00019 int l4util_map_event_to_keymap(unsigned value, unsigned shift);
00020
00021 __END_DECLS
00022
00023
00024 #endif /* __L4UTIL__KEYMAP_H__ */

```

## 15.399 l4/util/kip.h File Reference

```
#include <l4/sys/kip.h>
#include <l4/sys/compiler.h>
```

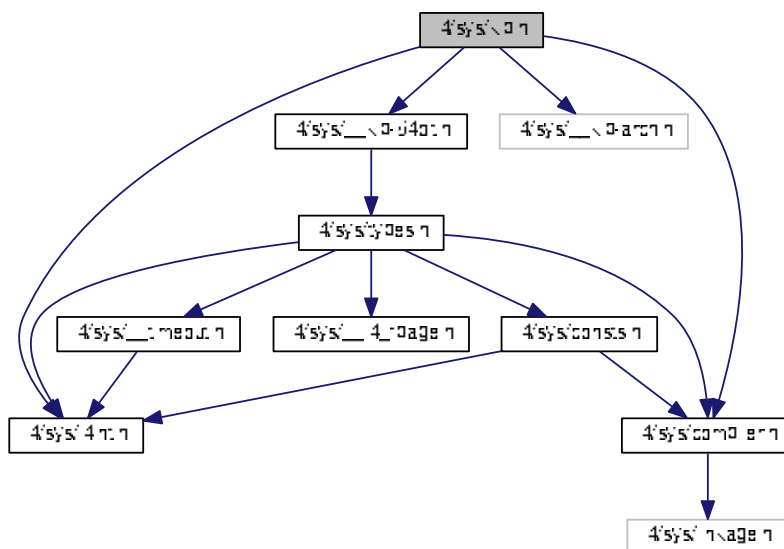




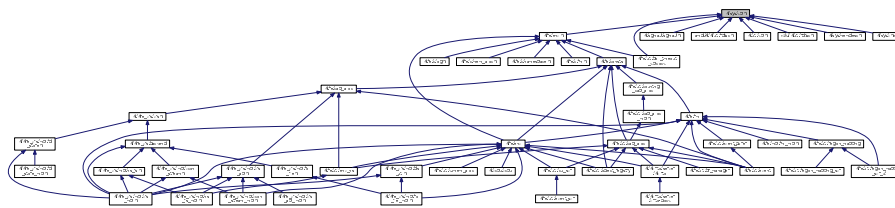
## 15.401 I4/sys/kip.h File Reference

```
#include <l4/sys/compiler.h>
#include <l4/sys/l4int.h>
#include <l4/sys/__kip-arch.h>
#include <l4/sys/__kip-64bit.h>
```

**Include dependency graph for kip.h:**



This graph shows which files directly or indirectly include this file:



## Macros

- `#define L4_KERNEL_INFO_MAGIC (0x4BE6344CL) /* "L4μK" */`  
Kernel Info Page identifier ("L4μK").

## Functions

- `l4_umword_t l4_kip_version (l4_kernel_info_t *kip) L4_NOTHROW`  
Get the kernel version.
- `const char * l4_kip_version_string (l4_kernel_info_t *kip) L4_NOTHROW`  
Get the kernel version string.
- `int l4_kernel_info_version_offset (l4_kernel_info_t *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 *kip) L4_NOTHROW`  
Return clock value from the KIP.
- `l4_umword_t l4_kip_clock_lw (l4_kernel_info_t *kip) L4_NOTHROW`  
Return least significant machine word of clock value from the KIP.

### 15.401.1 Detailed Description

Kernel Info Page access functions.

Definition in file [kip.h](#).

### 15.402 kip.h

```

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 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU General Public License 2.
00013 * Please see the COPYING-GPL-2 file for details.
00014 *
00015 * As a special exception, you may use this file as part of a free software
00016 * library without restriction. Specifically, if other files instantiate
00017 * templates or use macros or inline functions from this file, or you compile
00018 * this file and link it with other files to produce an executable, this
00019 * file does not by itself cause the resulting executable to be covered by
00020 * the GNU General Public License. This exception does not however
00021 * invalidate any other reasons why the executable file might be covered by
00022 * the GNU General Public License.

```

```

00023 */
00024 #pragma once
00025
00026 #include <l4/sys/compiler.h>
00027 #include <l4/sys/l4int.h>
00028
00029 #include <l4/sys/__kip-arch.h>
00030
00031 struct l4_kip_platform_info
00032 {
00033 char name[16];
00034 l4_uint32_t is_mp;
00035 struct l4_kip_platform_info_arch arch;
00036 };
00037
00038 #if L4_MWORD_BITS == 32
00039 #include <l4/sys/__kip-32bit.h>
00040 #else
00041 #include <l4/sys/__kip-64bit.h>
00042 #endif
00043
00044 enum l4_kernel_info_consts_t
00045 {
00046 L4_KIP_VERSION_FIASCO = 0x87004444,
00047 L4_KIP_VERSION_FIASCO_MASK = 0xff00ffff,
00048 };
00049
00050 #define L4_KERNEL_INFO_MAGIC (0x4BE6344CL) /* "L4pK" */
00051
00052 L4_INLINE l4_umword_t l4_kip_version(l4_kernel_info_t *kip)
00053 L4_NOTHROW;
00054
00055 L4_INLINE const char *l4_kip_version_string(l4_kernel_info_t *kip)
00056 L4_NOTHROW;
00057
00058 L4_INLINE int
00059 l4_kernel_info_version_offset(l4_kernel_info_t *kip)
00060 L4_NOTHROW;
00061
00062 L4_INLINE l4_cpu_time_t
00063 l4_kip_clock(l4_kernel_info_t *kip) L4_NOTHROW;
00064
00065 L4_INLINE l4_umword_t
00066 l4_kip_clock_lw(l4_kernel_info_t *kip) L4_NOTHROW;
00067
00068 /*****
00069 * Implementations
00070 *****/
00071
00072 L4_INLINE l4_umword_t
00073 l4_kip_version(l4_kernel_info_t *kip) L4_NOTHROW
00074 {
00075 return kip->version & L4_KIP_VERSION_FIASCO_MASK;
00076 }
00077
00078 L4_INLINE const char*
00079 l4_kip_version_string(l4_kernel_info_t *k) L4_NOTHROW
00080 {
00081 return (const char *)k + l4_kernel_info_version_offset(k);
00082 }
00083
00084 L4_INLINE int
00085 l4_kernel_info_version_offset(l4_kernel_info_t *kip)
00086 L4_NOTHROW
00087 {
00088 return kip->offset_version_strings << 4;
00089 }
00090
00091 L4_INLINE l4_cpu_time_t
00092 l4_kip_clock(l4_kernel_info_t *kip) L4_NOTHROW
00093 {
00094 unsigned long h1, l;
00095 unsigned long *c;
00096
00097 if (sizeof(unsigned long) == 8)
00098 return kip->_clock_val;
00099
00100 c = (unsigned long *)&kip->_clock_val;
00101 do
00102 {
00103 h1 = c[1];
00104 l4_mb();
00105 l = c[0];
00106 l4_mb();
00107 }
00108 while (h1 != c[1]);
00109
00110 return ((unsigned long long)h1 << 32) | l;
00111 }
00112
00113 L4_INLINE l4_umword_t

```

```

00160 l4_kip_clock_lw(l4_kernel_info_t *kip) L4_NOTHROW
00161 {
00162 /* We do the casting because the clock field is volatile */
00163 unsigned long *c = (unsigned long *)&kip->_clock_val;
00164 l4_mb();
00165 return c[0];
00166 }

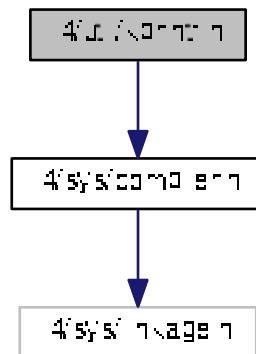
```

## 15.403 l4/util/kprintf.h File Reference

printf using the kernel debugger

```
#include <l4/sys/compiler.h>
```

Include dependency graph for kprintf.h:



### 15.403.1 Detailed Description

printf using the kernel debugger

Date

04/05/2007

Author

Adam Lackorzynski [adam@os.inf.tu-dresden.de](mailto:adam@os.inf.tu-dresden.de),

Definition in file [kprintf.h](#).

## 15.404 kprintf.h

```

00001 /*****
00009 */
00010 * (c) 2007-2009 Author(s)
00011 * economic rights: Technische Universität Dresden (Germany)
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU Lesser General Public License 2.1.
00014 * Please see the COPYING-LGPL-2.1 file for details.
00015 */
00016
00017 #ifndef __L4UTIL__INCLUDE__KPRINTF_H__
00018 #define __L4UTIL__INCLUDE__KPRINTF_H__
00019
00020 #include <l4/sys/compiler.h>
00021
00022 EXTERN_C_BEGIN
00023
00024 L4_CV int l4_kprintf(const char *fmt, ...)
00025 __attribute__((format (printf, 1, 2)));
00026
00027 EXTERN_C_END
00028
00029 #endif /* ! __L4UTIL__INCLUDE__KPRINTF_H__ */

```

## 15.405 l4/util/list\_alloc.h File Reference

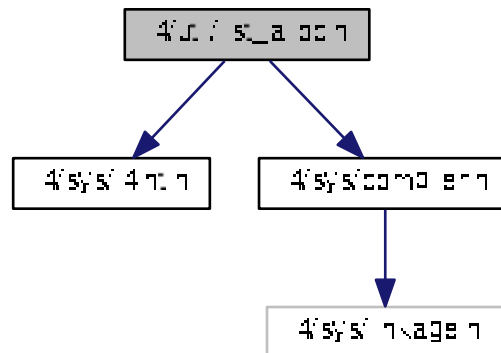
Simple list-based allocator.

```

#include <l4/sys/l4int.h>
#include <l4/sys/compiler.h>

```

Include dependency graph for list\_alloc.h:



## Functions

- 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.*

### 15.405.1 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](mailto:fm3@os.inf.tu-dresden.de)

Definition in file [list\\_alloc.h](#).

### 15.405.2 Function Documentation

#### 15.405.2.1 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                          |

#### 15.405.2.2 l4la\_avail()

```
l4_size_t l4la_avail (
 l4la_free_t ** first)
```

Show available memory in pool.

## Parameters

|              |                 |
|--------------|-----------------|
| <i>first</i> | list identifier |
|--------------|-----------------|

## 15.405.2.3 l4la\_dump()

```
void l4la_dump (
 l4la_free_t ** first)
```

Show all list members.

## Parameters

|              |                 |
|--------------|-----------------|
| <i>first</i> | list identifier |
|--------------|-----------------|

## 15.405.2.4 l4la\_free()

```
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           |

## 15.405.2.5 l4la\_init()

```
void l4la_init (
 l4la_free_t ** first)
```

Init memory pool.

## Parameters

|              |                 |
|--------------|-----------------|
| <i>first</i> | list identifier |
|--------------|-----------------|

## 15.406 list\_alloc.h

```

00001
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 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU Lesser General Public License 2.1.
00014 * Please see the COPYING-LGPL-2.1 file for details.
00015 */
00016
00017 #ifndef L4UTIL_L4LA_H
00018 #define L4UTIL_L4LA_H
00019
00020 #include <l4/sys/l4int.h>
00021 #include <l4/sys/compiler.h>
00022
00023 typedef struct l4la_free_t_s
00024 {
00025 struct l4la_free_t_s *next;
00026 l4_size_t size;
00027 } l4la_free_t;
00028
00029 #define L4LA_INITIALIZER { 0 }
00030
00031 EXTERN_C_BEGIN
00032
00037 L4_CV void l4la_free(l4la_free_t **first, void *block,
00038 l4_size_t size);
00038
00043 L4_CV void* l4la_alloc(l4la_free_t **first, l4_size_t size, unsigned align);
00044
00047 L4_CV void l4la_dump(l4la_free_t **first);
00048
00051 L4_CV void l4la_init(l4la_free_t **first);
00052
00055 L4_CV l4_size_t l4la_avail(l4la_free_t **first);
00056
00057 EXTERN_C_END
00058
00059 #endif

```

## 15.407 l4/util/lock.h File Reference

Simple lock implementation.

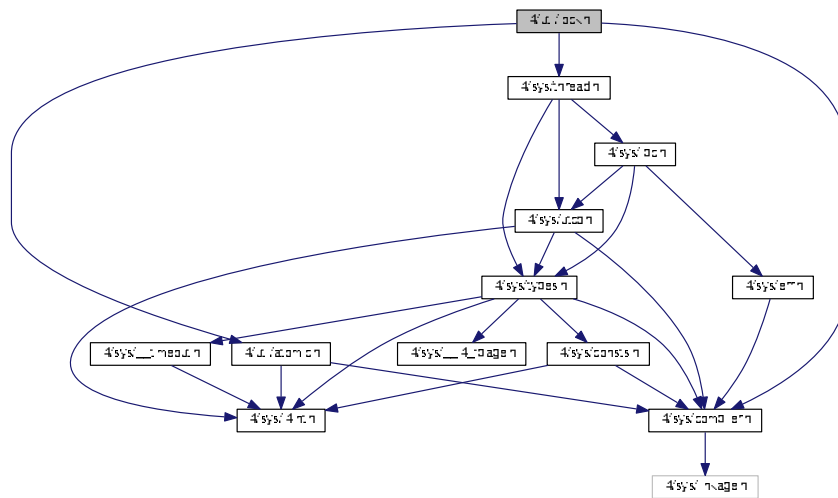
```

#include <l4/sys/thread.h>
#include <l4/sys/compiler.h>
#include <l4/util/atomic.h>

```



Include dependency graph for lock.h:



### 15.407.1 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](mailto:hohmuth@os.inf.tu-dresden.de)

Definition in file [lock.h](#).

## 15.408 lock.h

```

00001 /*****
00009 */
00010 * (c) 2000-2009 Author(s)
00011 * economic rights: Technische Universität Dresden (Germany)
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU Lesser General Public License 2.1.
00014 * Please see the COPYING-LGPL-2.1 file for details.
00015 */
00016
00017 /*****
00018 #ifndef __L4UTIL_LOCK_H__
00019 #define __L4UTIL_LOCK_H__
00020
00021 #include <l4/sys/thread.h>
00022 #include <l4/sys/compiler.h>
00023 #include <l4/util/atomic.h>
00024
00025 EXTERN_C_BEGIN

```

```

00026
00027 typedef l4_uint32_t l4util_simple_lock_t;
00028
00029 L4_INLINE int l4_simple_try_lock(l4util_simple_lock_t *lock);
00030 L4_INLINE void l4_simple_unlock(l4util_simple_lock_t *lock);
00031 L4_INLINE int l4_simple_lock_locked(l4util_simple_lock_t *lock);
00032 L4_INLINE void l4_simple_lock_solid(register l4util_simple_lock_t *p);
00033 L4_INLINE void l4_simple_lock(l4util_simple_lock_t * lock);
00034
00035 L4_INLINE int
00036 l4_simple_try_lock(l4util_simple_lock_t *lock)
00037 {
00038 return l4util_xchg32(lock, 1) == 0;
00039 }
00040
00041 L4_INLINE void
00042 l4_simple_unlock(l4util_simple_lock_t *lock)
00043 {
00044 *lock = 0;
00045 }
00046
00047 L4_INLINE int
00048 l4_simple_lock_locked(l4util_simple_lock_t *lock)
00049 {
00050 return (*lock == 0) ? 0 : 1;
00051 }
00052
00053 L4_INLINE void
00054 l4_simple_lock_solid(register l4util_simple_lock_t *p)
00055 {
00056 while (l4_simple_lock_locked(p) || !l4_simple_try_lock(p))
00057 l4_thread_switch(L4_INVALID_CAP);
00058 }
00059
00060 L4_INLINE void
00061 l4_simple_lock(l4util_simple_lock_t * lock)
00062 {
00063 if (!l4_simple_try_lock(lock))
00064 l4_simple_lock_solid(lock);
00065 }
00066
00067 EXTERN_C_END
00068
00069 #endif

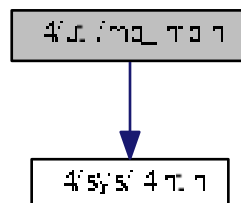
```

## 15.409 l4/util/mb\_info.h File Reference

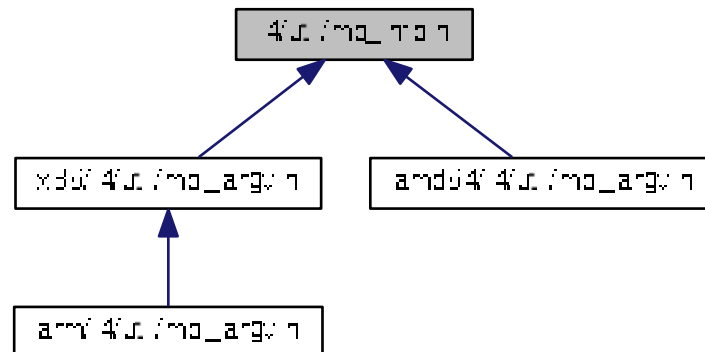
Multiboot info structure as defined by GRUB.

```
#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](#)
- 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](#)

## 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_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.*

### 15.409.1 Detailed Description

Multiboot info structure as defined by GRUB.

Definition in file [mb\\_info.h](#).

### 15.409.2 Macro Definition Documentation

#### 15.409.2.1 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 263 of file [mb\\_info.h](#).

## 15.409.2.2 MB\_ARD\_MEMORY

```
#define MB_ARD_MEMORY 1
```

usable memory "Type", all others are reserved.

Definition at line 62 of file [mb\\_info.h](#).

## 15.409.2.3 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 68 of file [mb\\_info.h](#).

## 15.410 mb\_info.h

```
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 * 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
00014 #ifndef L4UTIL_MB_INFO_H
00015 #define L4UTIL_MB_INFO_H
00016
00017 /*****
00018 * Multiboot (v1)
00019 *****/
00020
00021 #ifndef __ASSEMBLY__
00022
00023 #include <l4/sys/l4int.h>
00024
00031 typedef struct
00032 {
00033 l4_uint32_t mod_start;
00034 l4_uint32_t mod_end;
00035 l4_uint32_t cmdline;
00036 l4_uint32_t pad;
00037 } l4util_mb_mod_t;
00038
00039
00047 typedef struct __attribute__((packed))
00048 {
00049 l4_uint32_t struct_size;
00050 l4_uint64_t addr;
00051 l4_uint64_t size;
00052 l4_uint32_t type;
00053 /* unspecified optional padding... */
00054 } l4util_mb_addr_range_t;
00055
00056 #define l4util_mb_for_each_mmap_entry(i, mbi) \
00057 for (i = (l4util_mb_addr_range_t *) (unsigned long) mbi->mmap_addr; \
00058 (unsigned long) i < (unsigned long) mbi->mmap_addr + mbi->mmap_length; \
00059 i = (l4util_mb_addr_range_t *) ((unsigned long) i + mmap->struct_size + sizeof (mmap->struct_size)))
00060
00062 #define MB_ARD_MEMORY 1
00063
00068 #define MB_ART_MEMORY 1
00069 #define MB_ART_RESERVED 2
```

```
00070 #define MB_ART ACPI 3
00072 #define MB_ART_NV 4
00073 #define MB_ART_UNUSABLE 5
00077 typedef struct
00078 {
00079 l4_uint32_t size;
00080 l4_uint8_t drive_number;
00081 l4_uint8_t drive_mode;
00082 l4_uint16_t drive_cylinders;
00083 l4_uint8_t drive_heads;
00084 l4_uint8_t drive_sectors;
00085 l4_uint16_t drive_ports[0];
00086 } l4util_mb_drive_t;
00087
00088 /* Drive Mode. */
00089 #define MB_DI_CHS_MODE 0
00090 #define MB_DI_LBA_MODE 1
00091
00092
00094 typedef struct
00095 {
00096 l4_uint16_t version;
00097 l4_uint16_t cseg;
00098 l4_uint32_t offset;
00099 l4_uint16_t cseg_l6;
00100 l4_uint16_t dseg_l6;
00101 l4_uint16_t cseg_len;
00102 l4_uint16_t cseg_l6_len;
00103 l4_uint16_t dseg_l6_len;
00104 } l4util_mb_apm_t;
00105
00106
00108 typedef struct
00109 {
00110 l4_uint8_t signature[4];
00111 l4_uint16_t version;
00112 l4_uint32_t oem_string;
00113 l4_uint32_t capabilities;
00114 l4_uint32_t video_mode;
00115 l4_uint16_t total_memory;
00116 l4_uint16_t oem_software_rev;
00117 l4_uint32_t oem_vendor_name;
00118 l4_uint32_t oem_product_name;
00119 l4_uint32_t oem_product_rev;
00120 l4_uint8_t reserved[222];
00121 l4_uint8_t oem_data[256];
00122 } __attribute__((packed)) l4util_mb_vbe_ctrl_t;
00123
00124
00126 typedef struct
00127 {
00130 l4_uint16_t mode_attributes;
00131 l4_uint8_t win_a_attributes;
00132 l4_uint8_t win_b_attributes;
00133 l4_uint16_t win_granularity;
00134 l4_uint16_t win_size;
00135 l4_uint16_t win_a_segment;
00136 l4_uint16_t win_b_segment;
00137 l4_uint32_t win_func;
00138 l4_uint16_t bytes_per_scanline;
00143 l4_uint16_t x_resolution;
00144 l4_uint16_t y_resolution;
00145 l4_uint8_t x_char_size;
00146 l4_uint8_t y_char_size;
00147 l4_uint8_t number_of_planes;
00148 l4_uint8_t bits_per_pixel;
00149 l4_uint8_t number_of_banks;
00150 l4_uint8_t memory_model;
00151 l4_uint8_t bank_size;
00152 l4_uint8_t number_of_image_pages;
00153 l4_uint8_t reserved0;
00158 l4_uint8_t red_mask_size;
00159 l4_uint8_t red_field_position;
00160 l4_uint8_t green_mask_size;
00161 l4_uint8_t green_field_position;
00162 l4_uint8_t blue_mask_size;
00163 l4_uint8_t blue_field_position;
00164 l4_uint8_t reserved_mask_size;
00165 l4_uint8_t reserved_field_position;
00166 l4_uint8_t direct_color_mode_info;
00171 l4_uint32_t phys_base;
00172 l4_uint32_t reserved1;
00173 l4_uint16_t reversed2;
00178 l4_uint16_t linear_bytes_per_scanline;
00179 l4_uint8_t banked_number_of_image_pages;
00180 l4_uint8_t linear_number_of_image_pages;
00181 l4_uint8_t linear_red_mask_size;
```

```

00182 14_uint8_t linear_red_field_position;
00183 14_uint8_t linear_green_mask_size;
00184 14_uint8_t linear_green_field_position;
00185 14_uint8_t linear_blue_mask_size;
00186 14_uint8_t linear_blue_field_position;
00187 14_uint8_t linear_reserved_mask_size;
00188 14_uint8_t linear_reserved_field_position;
00189 14_uint32_t max_pixel_clock;
00190
00191 /* The VBE spec says this structure should have a size of 256 bytes but
00192 * the described structure layout is only 255 bytes... */
00193 14_uint8_t reserved3[189 + 1];
00195 } __attribute__((packed)) 14util_mb_vbe_mode_t;
00196
00197
00206 typedef struct
00207 {
00208 14_uint32_t flags;
00209 14_uint32_t mem_lower;
00210 14_uint32_t mem_upper;
00211 14_uint32_t boot_device;
00212 14_uint32_t cmdline;
00213 14_uint32_t mods_count;
00214 14_uint32_t mods_addr;
00216 union
00217 {
00218 struct
00219 {
00221 14_uint32_t tabsize;
00222 14_uint32_t strsize;
00223 14_uint32_t addr;
00224 14_uint32_t pad;
00225 }
00226 a;
00227
00228 struct
00229 {
00231 14_uint32_t num;
00232 14_uint32_t size;
00233 14_uint32_t addr;
00234 14_uint32_t shndx;
00235 }
00236 e;
00237 }
00238 syms;
00239
00240 14_uint32_t mmap_length;
00241 14_uint32_t mmap_addr;
00242 14_uint32_t drives_length;
00243 14_uint32_t drives_addr;
00244 14_uint32_t config_table;
00245 14_uint32_t boot_loader_name;
00246 14_uint32_t apm_table;
00247 14_uint32_t vbe_ctrl_info;
00248 14_uint32_t vbe_mode_info;
00249 14_uint16_t vbe_mode;
00250 14_uint16_t vbe_interface_seg;
00251 14_uint16_t vbe_interface_off;
00252 14_uint16_t vbe_interface_len;
00253 } 14util_mb_info_t;
00254
00255 #endif /* ! __ASSEMBLY__ */
00256
00262 #define L4UTIL_MB_MEMORY 0x00000001
00263
00265 #define L4UTIL_MB_BOOTDEV 0x00000002
00266
00268 #define L4UTIL_MB_CMDLINE 0x00000004
00269
00271 #define L4UTIL_MB_MODS 0x00000008
00272
00273 /* These next two are mutually exclusive */
00275 #define L4UTIL_MB_AOUT_SYMS 0x00000010
00276
00278 #define L4UTIL_MB_ELF_SHDR 0x00000020
00279
00281 #define L4UTIL_MB_MEM_MAP 0x00000040
00282
00284 #define L4UTIL_MB_DRIVE_INFO 0x00000080
00285
00287 #define L4UTIL_MB_CONFIG_TABLE 0x00000100
00288
00290 #define L4UTIL_MB_BOOT_LOADER_NAME 0x00000200
00291
00293 #define L4UTIL_MB_APM_TABLE 0x00000400
00294
00296 #define L4UTIL_MB_VIDEO_INFO 0x00000800

```

```

00297
00298
00300 #define L4UTIL_MB_VALID 0x2BADB002UL
00301 #define L4UTIL_MB_VALID_ASM 0x2BADB002
00302
00303
00304 /*****
00305 * Multiboot2
00306 *****/
00307
00308 #ifndef __ASSEMBLY__
00309
00310 typedef struct
00311 {
00312 l4_uint32_t total_size;
00313 l4_uint32_t reserved;
00314 } __attribute__((packed)) l4util_mb2_info_t;
00315
00316 typedef struct
00317 {
00318 char string[0];
00319 } __attribute__((packed)) l4util_mb2_cmdline_tag_t;
00320
00321 typedef struct
00322 {
00323 l4_uint32_t mod_start;
00324 l4_uint32_t mod_end;
00325 char string[];
00326 } __attribute__((packed)) l4util_mb2_module_tag_t;
00327
00328 typedef struct
00329 {
00330 l4_uint64_t base_addr;
00331 l4_uint64_t length;
00332 l4_uint32_t type;
00333 l4_uint32_t reserved;
00334 } __attribute__((packed)) l4util_mb2_memmap_entry_t;
00335
00336 typedef struct
00337 {
00338 l4_uint32_t entry_size;
00339 l4_uint32_t entry_version;
00340 l4util_mb2_memmap_entry_t entries[];
00341 } __attribute__((packed)) l4util_mb2_memmap_tag_t;
00342
00343 typedef struct
00344 {
00345 char data[0];
00346 } __attribute__((packed)) l4util_mb2_rsdp_tag_t;
00347
00348 typedef struct
00349 {
00350 l4_uint32_t type;
00351 l4_uint32_t size;
00352
00353 union
00354 {
00355 l4util_mb2_cmdline_tag_t cmdline;
00356 l4util_mb2_module_tag_t module;
00357 l4util_mb2_memmap_tag_t memmap;
00358 l4util_mb2_rsdp_tag_t rsdp;
00359 };
00360 } __attribute__((packed)) l4util_mb2_tag_t;
00361
00362 #endif /* ! __ASSEMBLY__ */
00363
00364
00365 #define L4UTIL_MB2_MAGIC 0xE85250D6
00366 #define L4UTIL_MB2_ARCH_I386 0x0
00367
00368 #define L4UTIL_MB2_TERMINATOR_HEADER_TAG 0
00369 #define L4UTIL_MB2_INFO_REQUEST_HEADER_TAG 1
00370 #define L4UTIL_MB2_ENTRY_ADDRESS_HEADER_TAG 3
00371
00372 #define L4UTIL_MB2_TAG_FLAG_REQUIRED 0
00373
00374 #define L4UTIL_MB2_TAG_ALIGN_SHIFT 3
00375 #define L4UTIL_MB2_TAG_ALIGN 8
00376
00377 #define L4UTIL_MB2_TERMINATOR_INFO_TAG 0
00378 #define L4UTIL_MB2_BOOT_CMDLINE_INFO_TAG 1
00379 #define L4UTIL_MB2_MODULE_INFO_TAG 3
00380 #define L4UTIL_MB2_MEMORY_MAP_INFO_TAG 6
00381 #define L4UTIL_MB2_RSDP_OLD_INFO_TAG 14
00382 #define L4UTIL_MB2_RSDP_NEW_INFO_TAG 15
00383
00384 #endif

```

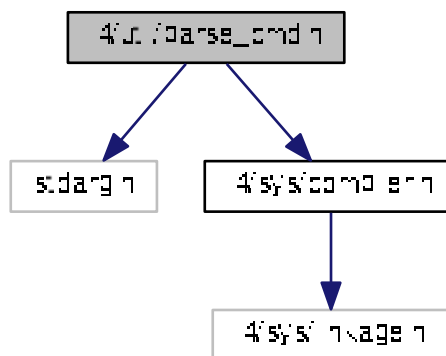


00385

## 15.411 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

- int [parse\\_cmdline](#) (int \*argc, const char \*\*\*argv, char arg0,...)  
*Parse the command-line for specified arguments and store the values into variables.*

### 15.411.1 Detailed Description

comfortable command-line parsing

#### Date

2002

#### Author

Jork Loeser [jork.loeser@inf.tu-dresden.de](mailto:jork.loeser@inf.tu-dresden.de)

Definition in file [parse\\_cmd.h](#).

## 15.412 parse\_cmd.h

```

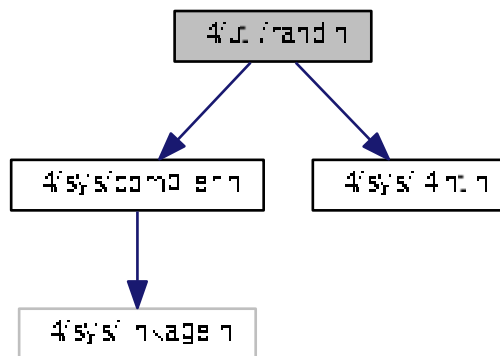
00001
00009 /*
00010 * (c) 2003-2009 Author(s)
00011 * economic rights: Technische Universität Dresden (Germany)
00012 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU Lesser General Public License 2.1.
00014 * Please see the COPYING-LGPL-2.1 file for details.
00015 */
00016
00017 #ifndef __PARSE_CMD_H
00018 #define __PARSE_CMD_H
00019
00020 #include <stdarg.h>
00021 #include <l4/sys/compiler.h>
00022
00028
00032 enum parse_cmd_type {
00033 PARSE_CMD_INT,
00034 PARSE_CMD_SWITCH,
00035 PARSE_CMD_STRING,
00036 PARSE_CMD_FN,
00037 PARSE_CMD_FN_ARG,
00038 PARSE_CMD_INC,
00039 PARSE_CMD_DEC,
00040 };
00041
00045 typedef L4_CV void (*parse_cmd_fn_t)(int);
00046
00050 typedef L4_CV void (*parse_cmd_fn_arg_t)(int, const char*, int);
00051
00052 EXTERN_C_BEGIN
00053
00140 L4_CV int parse_cmdline(int *argc, const char***argv, char arg0, ...);
00141 L4_CV int parse_cmdlinev(int *argc, const char***argv, char arg0, va_list va);
00142 L4_CV int parse_cmdline_extra(const char*argv0, const char*line, char delim,
00143 char arg0,...);
00144
00145 EXTERN_C_END
00148 #endif
00149

```

## 15.413 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.*

### 15.413.1 Detailed Description

Simple Pseudo-Random Number Generator.

#### Date

1998

#### Author

Lars Reuther [reuther@os.inf.tu-dresden.de](mailto:reuther@os.inf.tu-dresden.de)

Definition in file [rand.h](#).

## 15.414 rand.h

```

00001
00008 /*
00009 * (c) 2008-2009 Author(s)
00010 * economic rights: Technische Universität Dresden (Germany)
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU Lesser General Public License 2.1.
00013 * Please see the COPYING-LGPL-2.1 file for details.
00014 */
00015
00016 #ifndef __L4UTIL_RAND_H
00017 #define __L4UTIL_RAND_H
00018
00019 #define L4_RAND_MAX 65535
00020
00021 #include <l4/sys/compiler.h>
00022 #include <l4/sys/l4int.h>
00023
00024 EXTERN_C_BEGIN
00025
00037 l4_cv l4_uint32_t
00038 l4util_rand(void);
00039
00046 l4_cv void
00047 l4util_srand (l4_uint32_t seed);
00048
00049 EXTERN_C_END
00050
00051 #endif /* __L4UTIL_RAND_H */

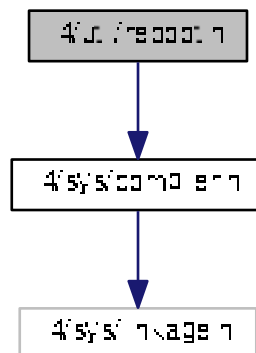
```

## 15.415 l4/util/reboot.h File Reference

Machine restarting functions.

```
#include <l4/sys/compiler.h>
```

Include dependency graph for reboot.h:



## Functions

- void `l4util_reboot` (void)  
*Machine reboot.*

### 15.415.1 Detailed Description

Machine restarting functions.

Definition in file [reboot.h](#).

## 15.416 reboot.h

```

00001
00005 /*
00006 * (c) 2000-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Alexander Warg <warg@os.inf.tu-dresden.de>,
00008 * Norman Feske <nf2@os.inf.tu-dresden.de>
00009 * economic rights; Technische Universität Dresden (Germany)
00010 * This file is part of TUD:OS and distributed under the terms of the
00011 * GNU Lesser General Public License 2.1.
00012 * Please see the COPYING-LGPL-2.1 file for details.
00013 */
00014 /*****
00015 #ifndef _L4UTIL_REBOOT_H
00016 #define _L4UTIL_REBOOT_H
00017
00018 #include <l4/sys/compiler.h>
00019
00029 EXTERN_C_BEGIN
00030 L4_CV void l4util_reboot(void) __attribute__((__noreturn__));
00031 EXTERN_C_END
00032
00033 #endif /* !_L4UTIL_REBOOT_H */

```

## 15.417 l4/util/sll.h File Reference

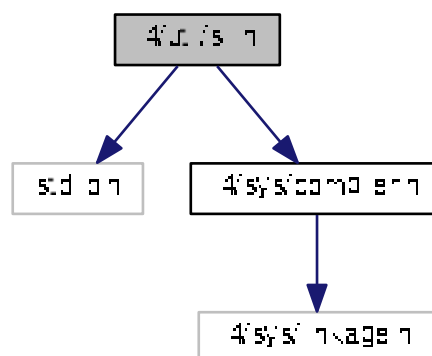
List implementation.

```

#include <stdlib.h>
#include <l4/sys/compiler.h>

```

Include dependency graph for sll.h:



## 15.417.1 Detailed Description

List implementation.

Definition in file [sll.h](#).

## 15.418 sll.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * Ronald Aigner <ra3@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 #ifndef __L4UTIL__SLL_H__
00014 #define __L4UTIL__SLL_H__
00015
00016 #include <stdlib.h>
00017
00018 #include <14/sys/compiler.h>
00019
00020 EXTERN_C_BEGIN
00021
00022 /*
00023 * the linked list structure
00024 */
00025
00026 typedef struct slist_t
00027 {
00028 struct slist_t *next; /* pointer to next node */
00029 void *data; /* void pointer for user data */
00030 } slist_t;
00031
00032 /*
00033 * function prototypes
00034 */
00035 static inline slist_t*
00036 list_new_entry(void *data);
00037
00038 static inline slist_t*
00039 list_append(slist_t *list, slist_t *new_node);
00040
00041 static inline slist_t*
00042 list_remove(slist_t *list, slist_t *node);
00043
00044 static inline void
00045 list_free_entry(slist_t **list);
00046
00047 static inline unsigned char
00048 list_is_empty(slist_t *list);
00049
00050 static inline slist_t*
00051 list_get_at(slist_t *list, int n);
00052
00053 static inline slist_t*
00054 list_add(slist_t *list, slist_t *new_node);
00055
00056 static inline void
00057 list_insert_after(slist_t *after, slist_t *new_node);
00058
00059 static inline int
00060 list_elements(slist_t *head);
00061
00062 /*
00063 * allocateNode()
00064 * allocate a new node.
00065 *
00066 * Parameters:
00067 * void *data a generic pointer to object data
00068 *
00069 * Return Values:
00070 * pointer to slist_t if succeeds
00071 * NULL otherwise
00072 */
00073
00074 static inline slist_t*

```

```

00075 list_new_entry(void *data)
00076 {
00077 slist_t *sll;
00078
00079 sll = (slist_t *)malloc(sizeof(slist_t));
00080 if (!sll)
00081 return ((slist_t *) NULL);
00082
00083 sll->data=data;
00084 sll->next=NULL;
00085
00086 return (sll);
00087 }
00088
00089 /*
00090 * appendNode()
00091 * appends a node to the end of a list
00092 *
00093 * Parameters:
00094 * slist_t *head - modify the list
00095 * slist_t *new - appends this node
00096 *
00097 * Return Values:
00098 * the new list
00099 *
00100 */
00101 static inline slist_t*
00102 list_append(slist_t *head, slist_t *new_node)
00103 {
00104 slist_t *ret = head;
00105 if (!head)
00106 return new_node;
00107
00108 while (head->next)
00109 head = head->next;
00110 head->next = new_node;
00111 return ret;
00112 }
00113
00114 /*
00115 * insertNode()
00116 * insert a node at the beginning of a list
00117 *
00118 * Parameters:
00119 * slist_t *head - modify this list
00120 * slist_t *new - appends this node
00121 *
00122 * Return Values:
00123 * the new list
00124 *
00125 */
00126 static inline slist_t*
00127 list_add(slist_t *head, slist_t *new_node)
00128 {
00129 if (!new_node)
00130 return head;
00131 new_node->next = head;
00132 return new_node;
00133 }
00134
00135 /*
00136 * insertNode()
00137 * insert a node at the beginning of a list
00138 *
00139 * Parameters:
00140 * slist_t *head - modify this list
00141 * slist_t *new - appends this node
00142 *
00143 * Return Values:
00144 * the new list
00145 *
00146 */
00147 static inline void
00148 list_insert_after(slist_t *after, slist_t *new_node)
00149 {
00150 if (!new_node)
00151 return;
00152 if (!after)
00153 return;
00154 new_node->next = after->next;
00155 after->next = new_node;
00156 }
00157
00158 /*
00159 * emptyList()
00160 * check if a list variable is NULL

```

```

00162 *
00163 * Parameters:
00164 * slist_t *list list
00165 *
00166 * Return Values:
00167 * TRUE if empty
00168 * FALSE if not empty
00169 *
00170 */
00171 static inline unsigned char
00172 list_is_empty(slist_t *list)
00173 {
00174 return ((list) ? 0 : 1);
00175 }
00176
00177 /*
00178 * delNode()
00179 * remove a node from a list
00180 *
00181 * Parameters:
00182 * slist_t *head - list to modify
00183 * slist_t *node - node to remove
00184 *
00185 * Return Values:
00186 * none
00187 *
00188 */
00189 static inline slist_t*
00190 list_remove(slist_t *head, slist_t *node)
00191 {
00192 slist_t *ret = head;
00193 if (list_is_empty(head))
00194 return ret;
00195 if (!node)
00196 return ret;
00197
00198 if (head == node)
00199 {
00200 ret = head->next;
00201 }
00202 else
00203 {
00204 while (head && (head->next != node))
00205 head = head->next;
00206 if (!head)
00207 return ret;
00208 else
00209 head->next = node->next;
00210 }
00211 list_free_entry(&node);
00212 return ret;
00213 }
00214
00215 /*
00216 * freeNode()
00217 * frees a node
00218 *
00219 * Parameters:
00220 * slist_t *list node to free
00221 *
00222 * Return Values:
00223 * none
00224 *
00225 */
00226 static inline void
00227 list_free_entry(slist_t **list)
00228 {
00229 if (*list)
00230 {
00231 free((void *) (*list));
00232 (*list)=NULL;
00233 }
00234 }
00235
00236 /*
00237 * getNthNode()
00238 * get nth node in a list
00239 *
00240 * Parameters:
00241 * slist_t *list - the head list
00242 * int n - return the node
00243 *
00244 * Return Values:
00245 * a pointer to the list at position n
00246 * NULL if there's no such node at posion n
00247 *
00248 */

```



```

00249 static inline slist_t*
00250 list_get_at(slist_t *list, int n)
00251 {
00252 int j=0;
00253
00254 while (list)
00255 {
00256 j++;
00257 if (j == n)
00258 return (list);
00259 list = list->next;
00260 }
00261
00262 return ((slist_t *) NULL);
00263 }
00264
00265 /*
00266 * numNodes()
00267 * returns number of nodes in the list
00268 *
00269 * Parameters:
00270 * slist_t *head - the head node of the list
00271 *
00272 * Return Values:
00273 * number of node/s
00274 *
00275 */
00276 static inline int
00277 list_elements(slist_t *head)
00278 {
00279 register int n;
00280 for (n=0; head; head=head->next) n++;
00281 return (n);
00282 }
00283
00284 EXTERN_C_END
00285
00286 #endif /* __L4UTIL__SLL_H__ */

```

## 15.419 l4/util/splitlog2.h File Reference

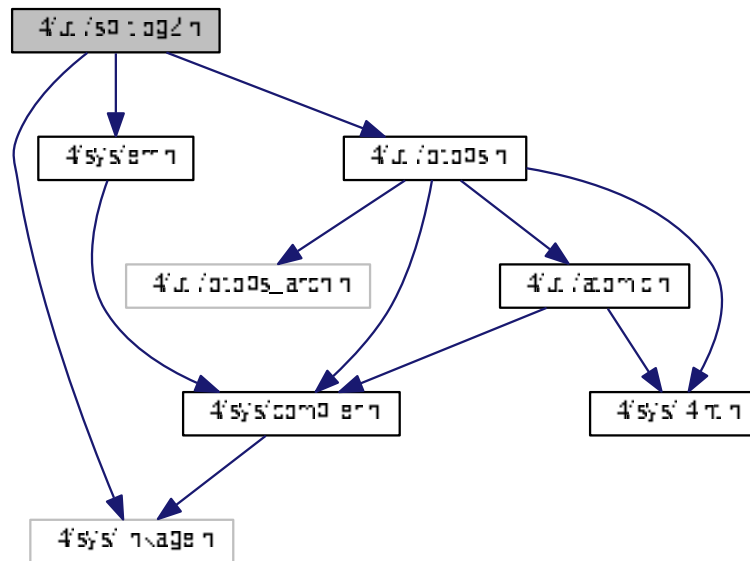
Split a range in log2 aligned and size-aligned chunks.

```

#include <l4/sys/linkage.h>
#include <l4/sys/err.h>
#include <l4/util/bitops.h>

```

Include dependency graph for `splitlog2.h`:



## Functions

- `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.

### 15.419.1 Detailed Description

Split a range in log2 aligned and size-aligned chunks.

Definition in file [splitlog2.h](#).

### 15.420 splitlog2.h

```

00001
00005 /*
00006 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
00007 * economic rights: Technische Universität Dresden (Germany)
00008 * This file is part of TUD:OS and distributed under the terms of the
00009 * GNU Lesser General Public License 2.1.
00010 * Please see the COPYING-LGPL-2.1 file for details.
00011 */
00012 #ifndef __L4UTIL__INCLUDE__SPLITLOG2_H__
00013 #define __L4UTIL__INCLUDE__SPLITLOG2_H__
00014
00015 #include <l4/sys/linkage.h>
00016 #include <l4/sys/err.h>
00017 #include <l4/util/bitops.h>
00018

```

```

00019 EXTERN_C_BEGIN
00020
00033 L4_INLINE long
00034 l4util_splitlog2_hdl(l4_addr_t start, l4_addr_t end,
00035 long (*handler)(l4_addr_t s, l4_addr_t e, int log2size));
00036
00045 L4_INLINE l4_addr_t
00046 l4util_splitlog2_size(l4_addr_t start, l4_addr_t end);
00047
00048 EXTERN_C_END
00049
00050 /* Implementation */
00051
00052 L4_INLINE long
00053 l4util_splitlog2_hdl(l4_addr_t start, l4_addr_t end,
00054 long (*handler)(l4_addr_t s, l4_addr_t e, int log2size))
00055 {
00056 if (end < start)
00057 return -L4_EINVAL;
00058 while (start <= end)
00059 {
00060 long retval;
00061 int len2 = l4util_splitlog2_size(start, end);
00062 l4_addr_t len = 1UL << len2;
00063 if ((retval = handler(start, start + len - 1, len2)))
00064 return retval;
00065 start += len;
00066 }
00067 return 0;
00068 }
00069
00070
00071 L4_INLINE l4_addr_t
00072 l4util_splitlog2_size(l4_addr_t start, l4_addr_t end)
00073 {
00074 int start_bits = l4util_bsf(start);
00075 int len_bits = l4util_bsr(end - start + 1);
00076 if (start_bits != -1 && len_bits > start_bits)
00077 len_bits = start_bits;
00078 return len_bits;
00079 }
00080
00081
00082 #endif /* ! __L4UTIL__INCLUDE__SPLITLOG2_H__ */

```

## 15.421 l4/util/stack.h File Reference

Some helper functions for stack manipulation.

```

#include <l4/sys/types.h>
#include <l4/sys/compiler.h>
#include <l4/util/stack_impl.h>

```



## 15.421.2.1 l4util\_stack\_get\_sp()

```
l4_addr_t l4util_stack_get_sp (
 void) [inline]
```

Get current stack pointer.

## Returns

stack pointer.

Definition at line 23 of file [stack\\_impl.h](#).

References [EXTERN\\_C\\_END](#).

## 15.422 stack.h

```
00001
00012 /*
00013 * (c) 2003-2009 Author(s)
00014 * economic rights: Technische Universität Dresden (Germany)
00015 * This file is part of TUD:OS and distributed under the terms of the
00016 * GNU Lesser General Public License 2.1.
00017 * Please see the COPYING-LGPL-2.1 file for details.
00018 */
00019
00020 #ifndef _L4UTIL_STACK_H
00021 #define _L4UTIL_STACK_H
00022
00023 #include <l4/sys/types.h>
00024 #include <l4/sys/compiler.h>
00025
00026 EXTERN_C_BEGIN
00027
00028 L4_INLINE void l4util_stack_push_mword(l4_addr_t *stack, l4_mword_t val);
00029
00030 /*****
00036 L4_INLINE l4_addr_t l4util_stack_get_sp(void);
00037
00038 */
00039 * Implementations.
00040 */
00041
00042 #include <l4/util/stack_impl.h>
00043
00044 L4_INLINE void
00045 l4util_stack_push_mword(l4_addr_t *stack, l4_mword_t val)
00046 {
00047 l4_mword_t *esp = (l4_mword_t*)(*stack);
00048 *--esp = val;
00049 *stack = (l4_addr_t)esp;
00050 }
00051
00052 EXTERN_C_END
00053
00054 #endif
```

## 15.423 l4/util/thread.h File Reference

Low-level Thread Functions.



## 15.423.2 Macro Definition Documentation

### 15.423.2.1 \_\_L4UTIL\_THREAD\_FUNC

```
#define __L4UTIL_THREAD_FUNC(
 name) 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) { for(;;); }
```

```
thread_cap->ex_regs((l4_umword_t)helper_thread, stack_addr);
```

Definition at line 72 of file [thread.h](#).

## 15.424 thread.h

```
00001
00008 /*
00009 * (c) 2003-2009 Author(s)
00010 * economic rights: Technische Universität Dresden (Germany)
00011 * This file is part of TUD:OS and distributed under the terms of the
00012 * GNU Lesser General Public License 2.1.
00013 * Please see the COPYING-LGPL-2.1 file for details.
00014 */
00015
00016 #ifndef __L4_THREAD_H
00017 #define __L4_THREAD_H
00018
00019 #include <l4/sys/types.h>
00020 #include <l4/sys/scheduler.h>
00021
00022 EXTERN_C_BEGIN
00023
00046 L4_CV long
00047 l4util_create_thread(l4_cap_idx_t id, l4_utcb_t *thread_utcb,
00048 l4_cap_idx_t factory,
00049 l4_umword_t pc, l4_umword_t sp,
00050 l4_cap_idx_t pager,
00051 l4_cap_idx_t task,
00052 l4_cap_idx_t scheduler, l4_sched_param_t scp)
00053 L4_NOTHROW;
00054
00055 EXTERN_C_END
00056
00055 #ifndef L4UTIL_THREAD_FUNC
00056
00072 #define __L4UTIL_THREAD_FUNC(name) void L4_NORETURN name(void)
00073 #define L4UTIL_THREAD_FUNC(name) __L4UTIL_THREAD_FUNC(name)
00074 #define __L4UTIL_THREAD_STATIC_FUNC(name) static L4_NORETURN void name(void)
00075 #define L4UTIL_THREAD_STATIC_FUNC(name) __L4UTIL_THREAD_STATIC_FUNC(name)
00076 #endif
00077
00078 #endif /* __L4_THREAD_H */
```







- `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  $\mu$ 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 or disable 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 or disable 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 or disable 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 or disable 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.*

### 15.425.1 Detailed Description

Common thread related definitions.

Definition in file [thread.h](#).

## 15.426 thread.h

```

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 * This file is part of TUD:OS and distributed under the terms of the
00013 * GNU General Public License 2.
00014 * Please see the COPYING-GPL-2 file for details.
00015 *
00016 * As a special exception, you may use this file as part of a free software
00017 * library without restriction. Specifically, if other files instantiate
00018 * templates or use macros or inline functions from this file, or you compile
00019 * this file and link it with other files to produce an executable, this
00020 * file does not by itself cause the resulting executable to be covered by
00021 * the GNU General Public License. This exception does not however
00022 * invalidate any other reasons why the executable file might be covered by
00023 * the GNU General Public License.
00024 */

```

```

00025 #pragma once
00026
00027 #include <l4/sys/types.h>
00028 #include <l4/sys/utcb.h>
00029 #include <l4/sys/ipc.h>
00030
00087 L4_INLINE l4_msgtag_t
00088 l4_thread_ex_regs(l4_cap_idx_t thread, l4_addr_t ip,
00089 l4_addr_t sp,
00089 l4_umword_t flags) L4_NOTHROW;
00090
00097 L4_INLINE l4_msgtag_t
00098 l4_thread_ex_regs_u(l4_cap_idx_t thread,
00098 l4_addr_t ip, l4_addr_t sp,
00099 l4_umword_t flags, l4_utcb_t *utcb)
00099 L4_NOTHROW;
00100
00124 L4_INLINE l4_msgtag_t
00125 l4_thread_ex_regs_ret(l4_cap_idx_t thread,
00125 l4_addr_t *ip, l4_addr_t *sp,
00126 l4_umword_t *flags) L4_NOTHROW;
00127
00134 L4_INLINE l4_msgtag_t
00135 l4_thread_ex_regs_ret_u(l4_cap_idx_t thread,
00135 l4_addr_t *ip, l4_addr_t *sp,
00136 l4_umword_t *flags, l4_utcb_t *utcb)
00136 L4_NOTHROW;
00137
00138
00139
00187 L4_INLINE void
00188 l4_thread_control_start(void) L4_NOTHROW;
00189
00194 L4_INLINE void
00195 l4_thread_control_start_u(l4_utcb_t *utcb) L4_NOTHROW;
00196
00206 L4_INLINE void
00207 l4_thread_control_pager(l4_cap_idx_t pager)
00207 L4_NOTHROW;
00208
00213 L4_INLINE void
00214 l4_thread_control_pager_u(l4_cap_idx_t pager, l4_utcb_t *utcb)
00214 L4_NOTHROW;
00215
00225 L4_INLINE void
00226 l4_thread_control_exc_handler(l4_cap_idx_t exc_handler)
00226 L4_NOTHROW;
00227
00232 L4_INLINE void
00233 l4_thread_control_exc_handler_u(l4_cap_idx_t exc_handler,
00233 l4_utcb_t *utcb) L4_NOTHROW;
00234
00235
00251 L4_INLINE void
00252 l4_thread_control_bind(l4_utcb_t *thread_utcb,
00252 l4_cap_idx_t task) L4_NOTHROW;
00253
00254
00259 L4_INLINE void
00260 l4_thread_control_bind_u(l4_utcb_t *thread_utcb,
00260 l4_cap_idx_t task, l4_utcb_t *utcb)
00261 L4_NOTHROW;
00262
00277 L4_INLINE void
00278 l4_thread_control_alien(int on) L4_NOTHROW;
00279
00284 L4_INLINE void
00285 l4_thread_control_alien_u(l4_utcb_t *utcb, int on) L4_NOTHROW;
00286
00297 L4_INLINE void
00298 l4_thread_control_ux_host_syscall(int on)
00298 L4_NOTHROW;
00299
00304 L4_INLINE void
00305 l4_thread_control_ux_host_syscall_u(l4_utcb_t *utcb, int on) L4_NOTHROW;
00306
00307
00308
00316 L4_INLINE l4_msgtag_t
00317 l4_thread_control_commit(l4_cap_idx_t thread)
00317 L4_NOTHROW;
00318
00323 L4_INLINE l4_msgtag_t
00324 l4_thread_control_commit_u(l4_cap_idx_t thread, l4_utcb_t *utcb)
00324 L4_NOTHROW;
00325
00332 L4_INLINE l4_msgtag_t
00333 l4_thread_yield(void) L4_NOTHROW;
00334

```

```

00343 L4_INLINE l4_msgtag_t
00344 l4_thread_switch(l4_cap_idx_t to_thread) L4_NOTHROW;
00345
00350 L4_INLINE l4_msgtag_t
00351 l4_thread_switch_u(l4_cap_idx_t to_thread, l4_utcb_t *utcb)
00352 L4_NOTHROW;
00353
00354
00364 L4_INLINE l4_msgtag_t
00365 l4_thread_stats_time(l4_cap_idx_t thread,
00366 l4_kernel_clock_t *us) L4_NOTHROW;
00367
00371 L4_INLINE l4_msgtag_t
00372 l4_thread_stats_time_u(l4_cap_idx_t thread, l4_kernel_clock_t *us,
00373 l4_utcb_t *utcb) L4_NOTHROW;
00374
00375
00386 L4_INLINE l4_msgtag_t
00387 l4_thread_vcpu_resume_start(void) L4_NOTHROW;
00388
00393 L4_INLINE l4_msgtag_t
00394 l4_thread_vcpu_resume_start_u(l4_utcb_t *utcb) L4_NOTHROW;
00395
00422 L4_INLINE l4_msgtag_t
00423 l4_thread_vcpu_resume_commit(l4_cap_idx_t thread,
00424 l4_msgtag_t tag) L4_NOTHROW;
00425
00430 L4_INLINE l4_msgtag_t
00431 l4_thread_vcpu_resume_commit_u(l4_cap_idx_t thread,
00432 l4_msgtag_t tag, l4_utcb_t *utcb)
00433 L4_NOTHROW;
00434
00451 L4_INLINE l4_msgtag_t
00452 l4_thread_vcpu_control(l4_cap_idx_t thread,
00453 l4_addr_t vcpu_state) L4_NOTHROW;
00454
00461 L4_INLINE l4_msgtag_t
00462 l4_thread_vcpu_control_u(l4_cap_idx_t thread,
00463 l4_addr_t vcpu_state,
00464 l4_utcb_t *utcb) L4_NOTHROW;
00465
00487 L4_INLINE l4_msgtag_t
00488 l4_thread_vcpu_control_ext(l4_cap_idx_t thread,
00489 l4_addr_t ext_vcpu_state) L4_NOTHROW;
00490
00497 L4_INLINE l4_msgtag_t
00498 l4_thread_vcpu_control_ext_u(l4_cap_idx_t thread,
00499 l4_addr_t ext_vcpu_state,
00500 l4_utcb_t *utcb) L4_NOTHROW;
00501
00514 L4_INLINE l4_msgtag_t
00515 l4_thread_register_del_irq(l4_cap_idx_t thread,
00516 l4_cap_idx_t irq) L4_NOTHROW;
00517
00521 L4_INLINE l4_msgtag_t
00522 l4_thread_register_del_irq_u(l4_cap_idx_t thread, l4_cap_idx_t irq,
00523 l4_utcb_t *utcb) L4_NOTHROW;
00524
00536 L4_INLINE l4_msgtag_t
00537 l4_thread_modify_sender_start(void) L4_NOTHROW;
00538
00543 L4_INLINE l4_msgtag_t
00544 l4_thread_modify_sender_start_u(l4_utcb_t *u) L4_NOTHROW;
00545
00570 L4_INLINE int
00571 l4_thread_modify_sender_add(l4_umword_t match_mask,
00572 l4_umword_t match,
00573 l4_umword_t del_bits,
00574 l4_umword_t add_bits,
00575 l4_msgtag_t *tag) L4_NOTHROW;
00576
00581 L4_INLINE int
00582 l4_thread_modify_sender_add_u(l4_umword_t match_mask,
00583 l4_umword_t match,
00584 l4_umword_t del_bits,
00585 l4_umword_t add_bits,
00586 l4_msgtag_t *tag, l4_utcb_t *u)
00587 L4_NOTHROW;
00588
00595 L4_INLINE l4_msgtag_t
00596 l4_thread_modify_sender_commit(l4_cap_idx_t thread,
00597 l4_msgtag_t tag) L4_NOTHROW;
00598
00602 L4_INLINE l4_msgtag_t

```

```

00603 l4_thread_modify_sender_commit_u(l4_cap_idx_t thread, l4_msgtag_t tag,
00604 l4_utcb_t *u) L4_NOTHROW;
00605
00612 enum L4_thread_ops
00613 {
00614 L4_THREAD_CONTROL_OP = 0UL,
00615 L4_THREAD_EX_REGS_OP = 1UL,
00616 L4_THREAD_SWITCH_OP = 2UL,
00617 L4_THREAD_STATS_OP = 3UL,
00618 L4_THREAD_VCPU_RESUME_OP = 4UL,
00619 L4_THREAD_REGISTER_DELETE_IRQ_OP = 5UL,
00620 L4_THREAD_MODIFY_SENDER_OP = 6UL,
00621 L4_THREAD_VCPU_CONTROL_OP = 7UL,
00622 L4_THREAD_VCPU_CONTROL_EXT_OP = L4_THREAD_VCPU_CONTROL_OP | 0x10000,
00623 L4_THREAD_X86_GDT_OP = 0x10UL,
00624 L4_THREAD_ARM_TPIDRURO_OP = 0x10UL,
00625 L4_THREAD_AMD64_SET_SEGMENT_BASE_OP = 0x12UL,
00626 L4_THREAD_AMD64_GET_SEGMENT_INFO_OP = 0x13UL,
00627 L4_THREAD_OPCODE_MASK = 0xffff,
00628 };
00629
00640 enum L4_thread_control_flags
00641 {
00643 L4_THREAD_CONTROL_SET_PAGER = 0x00100000,
00645 L4_THREAD_CONTROL_BIND_TASK = 0x02000000,
00647 L4_THREAD_CONTROL_ALIEN = 0x04000000,
00649 L4_THREAD_CONTROL_UX_NATIVE = 0x08000000,
00651 L4_THREAD_CONTROL_SET_EXC_HANDLER = 0x10000000,
00652 };
00653
00663 enum L4_thread_control_mr_indices
00664 {
00665 L4_THREAD_CONTROL_MR_IDX_FLAGS = 0,
00666 L4_THREAD_CONTROL_MR_IDX_PAGER = 1,
00667 L4_THREAD_CONTROL_MR_IDX_EXC_HANDLER = 2,
00668 L4_THREAD_CONTROL_MR_IDX_FLAG_VALS = 4,
00669 L4_THREAD_CONTROL_MR_IDX_BIND_UTCB = 5,
00670 L4_THREAD_CONTROL_MR_IDX_BIND_TASK = 6,
00671 };
00672
00678 enum L4_thread_ex_regs_flags
00679 {
00680 L4_THREAD_EX_REGS_CANCEL = 0x10000UL,
00681 L4_THREAD_EX_REGS_TRIGGER_EXCEPTION = 0x20000UL,
00682 };
00683
00684
00685 /* IMPLEMENTATION ----- */
00686
00687 #include <l4/sys/ipc.h>
00688 #include <l4/sys/types.h>
00689
00690 L4_INLINE l4_msgtag_t
00691 l4_thread_ex_regs_u(l4_cap_idx_t thread,
00692 l4_addr_t ip, l4_addr_t sp,
00693 l4_umword_t flags, l4_utcb_t *utcb)
00694 L4_NOTHROW
00695 {
00696 l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00697 v->mr[0] = L4_THREAD_EX_REGS_OP | flags;
00698 v->mr[1] = ip;
00699 v->mr[2] = sp;
00700 return l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 3, 0, 0),
00701 L4_IPC_NEVER);
00702 }
00703
00704 L4_INLINE l4_msgtag_t
00705 l4_thread_ex_regs_ret_u(l4_cap_idx_t thread,
00706 l4_addr_t *ip, l4_addr_t *sp,
00707 l4_umword_t *flags, l4_utcb_t *utcb)
00708 L4_NOTHROW
00709 {
00710 l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00711 l4_msgtag_t ret = l4_thread_ex_regs_u(thread, *ip, *sp, *flags, utcb);
00712 if (l4_error_u(ret, utcb))
00713 return ret;
00714 *flags = v->mr[0];
00715 *ip = v->mr[1];
00716 *sp = v->mr[2];
00717 return ret;
00718 }
00719
00716 L4_INLINE void
00717 l4_thread_control_start_u(l4_utcb_t *utcb) L4_NOTHROW
00718 {
00719 l4_msg_regs_t *v = l4_utcb_mr_u(utcb);

```

```

00720 v->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS] =
00721 L4_THREAD_CONTROL_OP;
00722 }
00723 L4_INLINE void
00724 l4_thread_control_pager_u(l4_cap_idx_t pager, l4_utcb_t *utcb)
00725 L4_NOTHROW
00726 {
00727 l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00728 v->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS] |=
00729 L4_THREAD_CONTROL_SET_PAGER;
00730 v->mr[L4_THREAD_CONTROL_MR_IDX_PAGER] = pager;
00731 }
00732 L4_INLINE void
00733 l4_thread_control_exc_handler_u(l4_cap_idx_t exc_handler,
00734 l4_utcb_t *utcb) L4_NOTHROW
00735 {
00736 l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00737 v->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS] |=
00738 L4_THREAD_CONTROL_SET_EXC_HANDLER;
00739 v->mr[L4_THREAD_CONTROL_MR_IDX_EXC_HANDLER] = exc_handler;
00740 }
00741 L4_INLINE void
00742 l4_thread_control_bind_u(l4_utcb_t *thread_utcb, l4_cap_idx_t task,
00743 l4_utcb_t *utcb) L4_NOTHROW
00744 {
00745 l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00746 v->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS] |=
00747 L4_THREAD_CONTROL_BIND_TASK;
00748 v->mr[L4_THREAD_CONTROL_MR_IDX_BIND_UTCB] = (
00749 l4_addr_t)thread_utcb;
00750 v->mr[L4_THREAD_CONTROL_MR_IDX_BIND_TASK] =
00751 L4_ITEM_MAP;
00752 v->mr[L4_THREAD_CONTROL_MR_IDX_BIND_TASK + 1] =
00753 l4_obj_fpage(task, 0, L4_FPAGE_RWX).raw;
00754 }
00755 L4_INLINE void
00756 l4_thread_control_alien_u(l4_utcb_t *utcb, int on) L4_NOTHROW
00757 {
00758 l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00759 v->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS] |=
00760 L4_THREAD_CONTROL_ALIEN;
00761 v->mr[L4_THREAD_CONTROL_MR_IDX_FLAG_VALS] |= on ?
00762 L4_THREAD_CONTROL_ALIEN : 0;
00763 }
00764 L4_INLINE void
00765 l4_thread_control_ux_host_syscall_u(l4_utcb_t *utcb, int on) L4_NOTHROW
00766 {
00767 l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00768 v->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS] |=
00769 L4_THREAD_CONTROL_UX_NATIVE;
00770 v->mr[L4_THREAD_CONTROL_MR_IDX_FLAG_VALS] |= on ?
00771 L4_THREAD_CONTROL_UX_NATIVE : 0;
00772 }
00773 L4_INLINE l4_msgtag_t
00774 l4_thread_control_commit_u(l4_cap_idx_t thread, l4_utcb_t *utcb)
00775 L4_NOTHROW
00776 {
00777 int items = 0;
00778 if (l4_utcb_mr_u(utcb)->mr[L4_THREAD_CONTROL_MR_IDX_FLAGS] &
00779 L4_THREAD_CONTROL_BIND_TASK)
00780 items = 1;
00781 return l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 6, items,
00782 0), L4_IPC_NEVER);
00783 }
00784 L4_INLINE l4_msgtag_t
00785 l4_thread_yield(void) L4_NOTHROW
00786 {
00787 l4_ipc_receive(L4_INVALID_CAP, NULL,
00788 L4_IPC_BOTH_TIMEOUT_0);
00789 return l4_msgtag(0, 0, 0, 0);
00790 }
00791 /* Preliminary, to be changed */
00792 L4_INLINE l4_msgtag_t
00793 l4_thread_switch_u(l4_cap_idx_t to_thread, l4_utcb_t *utcb)
00794 L4_NOTHROW
00795 {
00796 l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00797 v->mr[0] = L4_THREAD_SWITCH_OP;

```

```

00790 return l4_ipc_call(to_thread, utcb, l4_msgtag(
00791 L4_PROTO_THREAD, 1, 0, 0), L4_IPC_NEVER);
00792 }
00793
00794 L4_INLINE l4_msgtag_t
00795 l4_thread_stats_time_u(l4_cap_idx_t thread, l4_kernel_clock_t *us,
00796 l4_utcb_t *utcb) L4_NOTHROW
00797 {
00798 l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00799 l4_msgtag_t res;
00800
00801 v->mr[0] = L4_THREAD_STATS_OP;
00802
00803 res = l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 1, 0, 0),
00804 L4_IPC_NEVER);
00805
00806 if (l4_msgtag_has_error(res))
00807 return res;
00808
00809 *us = v->mr64[l4_utcb_mr64_idx(0)];
00810 return res;
00811 }
00812
00813 L4_INLINE l4_msgtag_t
00814 l4_thread_vcpu_resume_start_u(l4_utcb_t *utcb) L4_NOTHROW
00815 {
00816 l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00817 v->mr[0] = L4_THREAD_VCPU_RESUME_OP;
00818 return l4_msgtag(L4_PROTO_THREAD, 1, 0, 0);
00819 }
00820
00821 L4_INLINE l4_msgtag_t
00822 l4_thread_vcpu_resume_commit_u(l4_cap_idx_t thread,
00823 l4_msgtag_t tag, l4_utcb_t *utcb)
00824 L4_NOTHROW
00825 {
00826 return l4_ipc_call(thread, utcb, tag, L4_IPC_NEVER);
00827 }
00828
00829 L4_INLINE l4_msgtag_t
00830 l4_thread_ex_regs(l4_cap_idx_t thread, l4_addr_t ip,
00831 l4_addr_t sp,
00832 l4_umword_t flags) L4_NOTHROW
00833 {
00834 return l4_thread_ex_regs_u(thread, ip, sp, flags, l4_utcb());
00835 }
00836
00837 L4_INLINE l4_msgtag_t
00838 l4_thread_ex_regs_ret(l4_cap_idx_t thread,
00839 l4_addr_t *ip, l4_addr_t *sp,
00840 l4_umword_t *flags) L4_NOTHROW
00841 {
00842 return l4_thread_ex_regs_ret_u(thread, ip, sp, flags,
00843 l4_utcb());
00844 }
00845
00846 L4_INLINE void
00847 l4_thread_control_start(void) L4_NOTHROW
00848 {
00849 l4_thread_control_start_u(l4_utcb());
00850 }
00851
00852 L4_INLINE void
00853 l4_thread_control_pager(l4_cap_idx_t pager)
00854 L4_NOTHROW
00855 {
00856 l4_thread_control_pager_u(pager, l4_utcb());
00857 }
00858
00859 L4_INLINE void
00860 l4_thread_control_exc_handler(l4_cap_idx_t exc_handler)
00861 L4_NOTHROW
00862 {
00863 l4_thread_control_exc_handler_u(exc_handler, l4_utcb());
00864 }
00865
00866 L4_INLINE void
00867 l4_thread_control_bind(l4_utcb_t *thread_utcb,
00868 l4_cap_idx_t task) L4_NOTHROW
00869 {
00870 l4_thread_control_bind_u(thread_utcb, task, l4_utcb());
00871 }
00872
00873 L4_INLINE void

```

```

00868 l4_thread_control_alien(int on) L4_NOTHROW
00869 {
00870 l4_thread_control_alien_u(l4_utcb(), on);
00871 }
00872
00873 L4_INLINE void
00874 l4_thread_control_ux_host_syscall(int on)
00875 L4_NOTHROW
00876 {
00877 l4_thread_control_ux_host_syscall_u(l4_utcb(), on);
00878 }
00879 L4_INLINE l4_msgtag_t
00880 l4_thread_control_commit(l4_cap_idx_t thread)
00881 L4_NOTHROW
00882 {
00883 return l4_thread_control_commit_u(thread, l4_utcb());
00884 }
00885
00886
00887
00888 L4_INLINE l4_msgtag_t
00889 l4_thread_switch(l4_cap_idx_t to_thread) L4_NOTHROW
00890 {
00891 return l4_thread_switch_u(to_thread, l4_utcb());
00892 }
00893
00894
00895
00896
00897 L4_INLINE l4_msgtag_t
00898 l4_thread_stats_time(l4_cap_idx_t thread,
00899 l4_kernel_clock_t *us) L4_NOTHROW
00900 {
00901 return l4_thread_stats_time_u(thread, us, l4_utcb());
00902 }
00903 L4_INLINE l4_msgtag_t
00904 l4_thread_vcpu_resume_start(void) L4_NOTHROW
00905 {
00906 return l4_thread_vcpu_resume_start_u(l4_utcb());
00907 }
00908
00909 L4_INLINE l4_msgtag_t
00910 l4_thread_vcpu_resume_commit(l4_cap_idx_t thread,
00911 l4_msgtag_t tag) L4_NOTHROW
00912 {
00913 return l4_thread_vcpu_resume_commit_u(thread, tag, l4_utcb());
00914 }
00915
00916
00917 L4_INLINE l4_msgtag_t
00918 l4_thread_register_del_irq_u(l4_cap_idx_t thread, l4_cap_idx_t irq,
00919 l4_utcb_t *u) L4_NOTHROW
00920 {
00921 l4_msg_regs_t *m = l4_utcb_mr_u(u);
00922 m->mr[0] = L4_THREAD_REGISTER_DELETE_IRQ_OP;
00923 m->mr[1] = l4_map_obj_control(0, 0);
00924 m->mr[2] = l4_obj_fpage(irq, 0, L4_CAP_FPAGE_RWS).
00925 raw;
00926 return l4_ipc_call(thread, u, l4_msgtag(L4_PROTO_THREAD, 1, 1, 0),
00927 L4_IPC_NEVER);
00928 }
00929
00930 L4_INLINE l4_msgtag_t
00931 l4_thread_register_del_irq(l4_cap_idx_t thread,
00932 l4_cap_idx_t irq) L4_NOTHROW
00933 {
00934 return l4_thread_register_del_irq_u(thread, irq, l4_utcb());
00935 }
00936
00937 L4_INLINE l4_msgtag_t
00938 l4_thread_vcpu_control_u(l4_cap_idx_t thread,
00939 l4_addr_t vcpu_state,
00940 l4_utcb_t *utcb) L4_NOTHROW
00941 {
00942 l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00943 v->mr[0] = L4_THREAD_VCPU_CONTROL_OP;
00944 v->mr[1] = vcpu_state;
00945 return l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 2, 0, 0),
00946 L4_IPC_NEVER);
00947 }
00948
00949 L4_INLINE l4_msgtag_t

```



```

00947 l4_thread_vcpu_control(l4_cap_idx_t thread,
 l4_addr_t vcpu_state) L4_NOTHROW
00948 { return l4_thread_vcpu_control_u(thread, vcpu_state,
 l4_utcb()); }
00949
00950
00951 L4_INLINE l4_msgtag_t
00952 l4_thread_vcpu_control_ext_u(l4_cap_idx_t thread,
 l4_addr_t ext_vcpu_state,
 l4_utcb_t *utcb) L4_NOTHROW
00953 {
00954 l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00955 v->mr[0] = L4_THREAD_VCPU_CONTROL_EXT_OP;
00956 v->mr[1] = ext_vcpu_state;
00957 return l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 2, 0, 0),
 L4_IPC_NEVER);
00958 }
00959
00960
00961 L4_INLINE l4_msgtag_t
00962 l4_thread_vcpu_control_ext(l4_cap_idx_t thread,
 l4_addr_t ext_vcpu_state) L4_NOTHROW
00963 { return l4_thread_vcpu_control_ext_u(thread, ext_vcpu_state,
 l4_utcb()); }
00964
00965 L4_INLINE l4_msgtag_t
00966 l4_thread_modify_sender_start_u(l4_utcb_t *u) L4_NOTHROW
00967 {
00968 l4_msg_regs_t *m = l4_utcb_mr_u(u);
00969 m->mr[0] = L4_THREAD_MODIFY_SENDER_OP;
00970 return l4_msgtag(L4_PROTO_THREAD, 1, 0, 0);
00971 }
00972
00973 L4_INLINE int
00974 l4_thread_modify_sender_add_u(l4_umword_t match_mask,
 l4_umword_t match,
 l4_umword_t del_bits,
 l4_umword_t add_bits,
 l4_msgtag_t *tag, l4_utcb_t *u)
 L4_NOTHROW
00975 {
00976 l4_msg_regs_t *m = l4_utcb_mr_u(u);
00977 unsigned w = l4_msgtag_words(*tag);
00978 if (w >= L4_UTCB_GENERIC_DATA_SIZE - 4)
00979 return -L4_ENOMEM;
00980
00981 m->mr[w] = match_mask;
00982 m->mr[w+1] = match;
00983 m->mr[w+2] = del_bits;
00984 m->mr[w+3] = add_bits;
00985
00986 *tag = l4_msgtag(l4_msgtag_label(*tag), w + 4, 0, 0);
00987
00988 return 0;
00989 }
00990
00991 L4_INLINE l4_msgtag_t
00992 l4_thread_modify_sender_commit_u(l4_cap_idx_t thread, l4_msgtag_t tag,
 l4_utcb_t *u) L4_NOTHROW
00993 {
00994 return l4_ipc_call(thread, u, tag, L4_IPC_NEVER);
00995 }
01000
01001
01002 L4_INLINE l4_msgtag_t
01003 l4_thread_modify_sender_start(void) L4_NOTHROW
01004 {
01005 return l4_thread_modify_sender_start_u(l4_utcb());
01006 }
01007
01008 L4_INLINE int
01009 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
01010 {
01011 return l4_thread_modify_sender_add_u(match_mask, match,
 del_bits, add_bits, tag, l4_utcb());
01012 }
01013
01014 L4_INLINE l4_msgtag_t
01015 l4_thread_modify_sender_commit(l4_cap_idx_t thread,
 l4_msgtag_t tag) L4_NOTHROW
01016 {
01017 return l4_thread_modify_sender_commit_u(thread, tag, l4_utcb());
01018 }
01019
01020
01021
01022
01023

```

## 15.427 arm/l4/sys/thread.h File Reference

ARM-specific thread related definitions.

### 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.

### 15.427.1 Detailed Description

ARM-specific thread related definitions.

Definition in file [thread.h](#).

## 15.428 thread.h

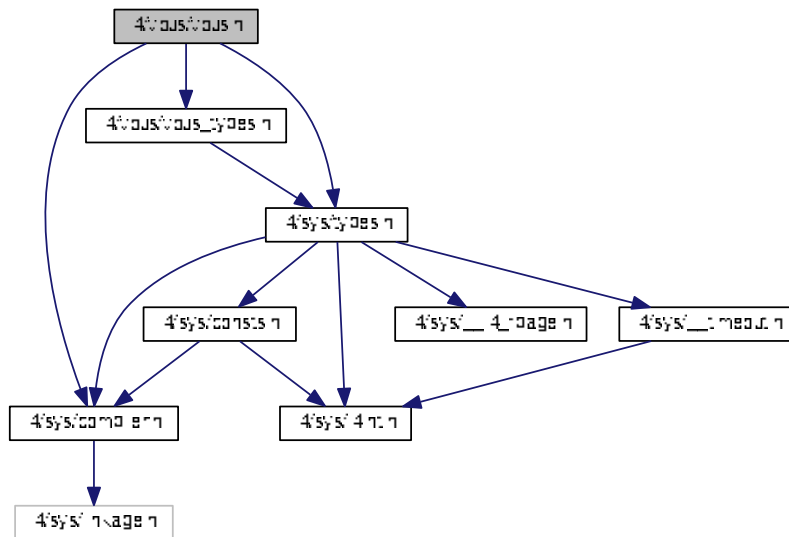
```

00001
00005 /*
00006 * (c) 2013 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
00007 * economic rights: Technische Universität Dresden (Germany)
00008 *
00009 * This file is part of TUD:OS and distributed under the terms of the
00010 * GNU General Public License 2.
00011 * Please see the COPYING-GPL-2 file for details.
00012 *
00013 * As a special exception, you may use this file as part of a free software
00014 * library without restriction. Specifically, if other files instantiate
00015 * templates or use macros or inline functions from this file, or you compile
00016 * this file and link it with other files to produce an executable, this
00017 * file does not by itself cause the resulting executable to be covered by
00018 * the GNU General Public License. This exception does not however
00019 * invalidate any other reasons why the executable file might be covered by
00020 * the GNU General Public License.
00021 */
00022 #pragma once
00023
00024 #include_next <l4/sys/thread.h>
00025
00034 L4_INLINE l4_msgtag_t
00035 l4_thread_arm_set_tpidruro(l4_cap_idx_t thread,
00036 l4_addr_t tpidruro) L4_NOTHROW;
00036
00041 L4_INLINE l4_msgtag_t
00042 l4_thread_arm_set_tpidruro_u(l4_cap_idx_t thread, l4_addr_t tpidruro,
00043 l4_utcb_t *utcb) L4_NOTHROW;
00044
00045 /* IMPLEMENTATION -----*/
00046
00047 L4_INLINE l4_msgtag_t
00048 l4_thread_arm_set_tpidruro_u(l4_cap_idx_t thread, l4_addr_t tpidruro,
00049 l4_utcb_t *utcb) L4_NOTHROW
00050 {
00051 l4_msg_regs_t *v = l4_utcb_mr_u(utcb);
00052 v->mr[0] = L4_THREAD_ARM_TPIDRURO_OP;
00053 v->mr[1] = tpidruro;
00054 return l4_ipc_call(thread, utcb, l4_msgtag(L4_PROTO_THREAD, 2, 0, 0),
00055 L4_IPC_NEVER);
00056 }
00057
00058 L4_INLINE l4_msgtag_t
00059 l4_thread_arm_set_tpidruro(l4_cap_idx_t thread,
00060 l4_addr_t tpidruro) L4_NOTHROW
00061 {
00062 return l4_thread_arm_set_tpidruro_u(thread, tpidruro, l4_utcb());
00063 }

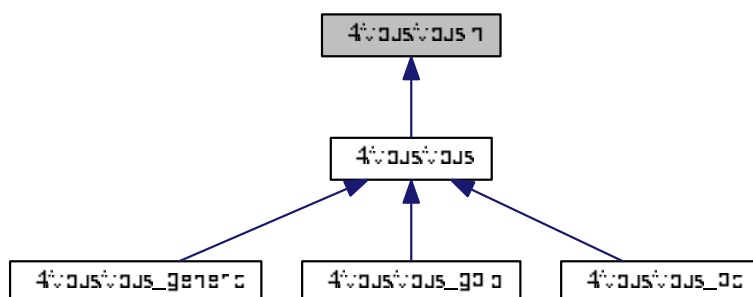
```

## Description of the vbus C API.

Include dependency graph for vbus.h:



This graph shows which files directly or indirectly include this file:



## Enumerations

- enum { L4VBUS\_NULL = 0, L4VBUS\_ROOT\_BUS = 0 }  
*Constants for device nodes.*
- 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

- 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 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, int 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\\_request\\_resource](#) ([l4\\_cap\\_idx\\_t](#) vbus, [l4vbus\\_resource\\_t](#) const \*res, int flags)  
*Request a resource of a specific type.*
- 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 ([L4::Task](#)) or a [L4Re::Dma\\_space](#) to a DMA domain.*
- int [l4vbus\\_release\\_resource](#) ([l4\\_cap\\_idx\\_t](#) vbus, [l4vbus\\_resource\\_t](#) const \*res)  
*Release a previously requested 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.*

### 15.429.1 Detailed Description

Description of the vbus C API.

Definition in file [vbus.h](#).

### 15.429.2 Enumeration Type Documentation

#### 15.429.2.1 anonymous enum

anonymous enum

Constants for device nodes.

Enumerator

|                 |                          |
|-----------------|--------------------------|
| L4VBUS_NULL     | NULL device.             |
| L4VBUS_ROOT_BUS | Root device on the vbus. |

Definition at line 22 of file [vbus.h](#).

## 15.430 vbus.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 * This file is part of TUD:OS and distributed under the terms of the
00008 * GNU General Public License 2.
00009 * Please see the COPYING-GPL-2 file for details.
00010 */
00015 #pragma once
00016
00017 #include <l4/sys/compiler.h>
00018 #include <l4/vbus/vbus_types.h>
00019 #include <l4/sys/types.h>
00020
00022 enum {
00023 L4VBUS_NULL = 0,
00024 L4VBUS_ROOT_BUS = 0,
00025 };
00026
00045 __BEGIN_DECLS
00046
00053 int L4_CV
00054 l4vbus_get_device_by_hid(l4_cap_idx_t vbus, l4vbus_device_handle_t
 parent,
00055 l4vbus_device_handle_t *child, char const *hid,
00056 int depth, l4vbus_device_t *devinfo);
00057
00069 int L4_CV
00070 l4vbus_get_next_device(l4_cap_idx_t vbus, l4vbus_device_handle_t parent,
00071 l4vbus_device_handle_t *child, int depth,
00072 l4vbus_device_t *devinfo);
00073
00088 int L4_CV
00089 l4vbus_get_device(l4_cap_idx_t vbus, l4vbus_device_handle_t dev,
00090 l4vbus_device_t *devinfo);
00091
00100 int L4_CV
00101 l4vbus_get_resource(l4_cap_idx_t vbus, l4vbus_device_handle_t dev,
00102 int res_idx, l4vbus_resource_t *res);
00103
00104
00111 int L4_CV
00112 l4vbus_is_compatible(l4_cap_idx_t vbus, l4vbus_device_handle_t dev,
00113 char const *cid);
00114
00125 int L4_CV
00126 l4vbus_get_hid(l4_cap_idx_t vbus, l4vbus_device_handle_t dev, char *hid,
00127 unsigned long max_len);
00128
00146 int L4_CV
00147 l4vbus_request_resource(l4_cap_idx_t vbus,
00148 l4vbus_resource_t const *res,
00149 int flags);
00149
00153 enum L4vbus_dma_domain_assign_flags
00154 {
00156 L4VBUS_DMAD_UNBIND = 0,
00158 L4VBUS_DMAD_BIND = 1,
00160 L4VBUS_DMAD_L4RE_DMA_SPACE = 0,
00162 L4VBUS_DMAD_KERNEL_DMA_SPACE = 2,
00163 };
00164
00185 int L4_CV
00186 l4vbus_assign_dma_domain(l4_cap_idx_t vbus, unsigned domain_id,
00187 unsigned flags, l4_cap_idx_t dma_space);
00188
00197 int L4_CV
00198 l4vbus_release_resource(l4_cap_idx_t vbus,
00199 l4vbus_resource_t const *res);
00199
00209 int L4_CV
00210 l4vbus_vicu_get_cap(l4_cap_idx_t vbus, l4vbus_device_handle_t icu,
00211 l4_cap_idx_t cap);
00212
00213 __END_DECLS
00214

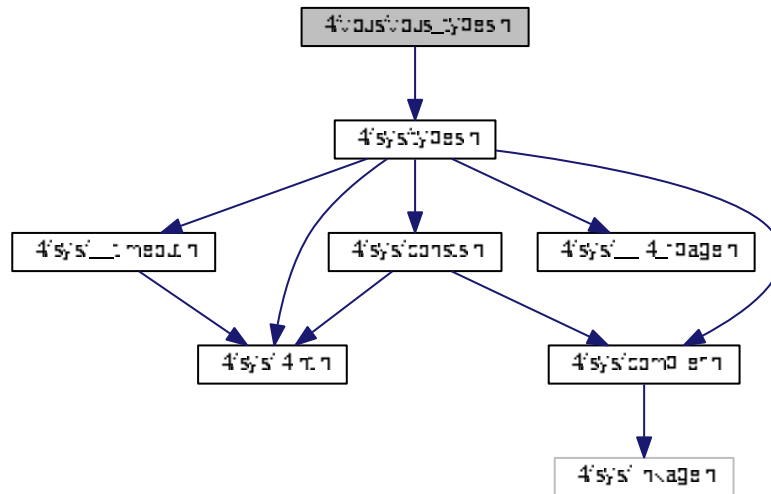
```

## 15.431 l4/vbus/vbus\_types.h File Reference

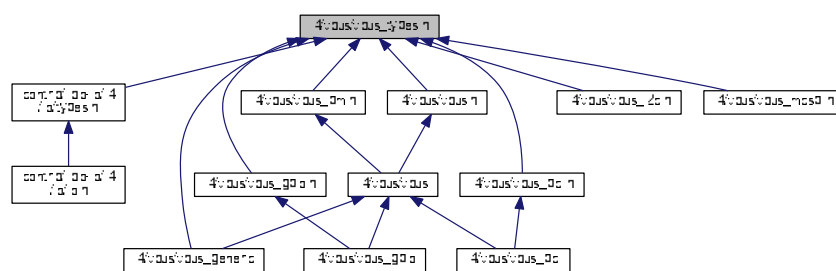
This header file contains descriptions of vbus related data types and constants.

```
#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.

## 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.*

### 15.431.1 Detailed Description

This header file contains descriptions of vbus related data types and constants.

Definition in file [vbus\\_types.h](#).

### 15.431.2 Enumeration Type Documentation

#### 15.431.2.1 l4vbus\_resource\_type\_t

enum `l4vbus_resource_type_t`

Description of vbus resource types.

#### Enumerator

|                                         |                              |
|-----------------------------------------|------------------------------|
| <code>L4VBUS_RESOURCE_INVALID</code>    | Invalid type.                |
| <code>L4VBUS_RESOURCE_IRQ</code>        | Interrupt resource.          |
| <code>L4VBUS_RESOURCE_MEM</code>        | I/O memory resource.         |
| <code>L4VBUS_RESOURCE_PORT</code>       | I/O port resource (x86 only) |
| <code>L4VBUS_RESOURCE_BUS</code>        | Bus resource.                |
| <code>L4VBUS_RESOURCE_GPIO</code>       | Gpio resource.               |
| <code>L4VBUS_RESOURCE_DMA_DOMAIN</code> | DMA domain.                  |
| <code>L4VBUS_RESOURCE_MAX</code>        | Maximum resource id.         |

Definition at line 39 of file [vbus\\_types.h](#).

## 15.432 vbus\_types.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 * This file is part of TUD:OS and distributed under the terms of the
00007 * GNU General Public License 2.
00008 * Please see the COPYING-GPL-2 file for details.
00009 */
```

```
00015 #pragma once
00016
00017 #include <l4/sys/types.h>
00018
00019 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
00050 enum l4vbus_consts_t {
00051 L4VBUS_DEV_NAME_LEN = 64,
00052 L4VBUS_MAX_DEPTH = 100,
00053 };
00054
00056 typedef struct {
00058 l4_uint32_t type;
00060 char name[L4VBUS_DEV_NAME_LEN];
00062 unsigned num_resources;
00064 unsigned flags;
00065 } l4vbus_device_t;
00066
00067 enum l4vbus_device_flags_t {
00068 L4VBUS_DEVICE_F_CHILDREN = 0x10,
00069 };
```



## Chapter 16

# Example Documentation

### 16.1 examples/clntsrv/client.cc

Client/Server example using C++ infrastructure – Client implementation.

```
/*
 * (c) 2008-2009 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 General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#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;
}
```

## 16.2 examples/clntsrv/clntsrv.cfg

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");
```

## 16.3 examples/clntsrv/server.cc

Client/Server example using C++ infrastructure – Server implementation.

```
/*
 * (c) 2008-2009 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 General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#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<L4Re::Util::Br_manager_hooks>
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 substractions and negations.\n");

 // Wait for client requests
 server.loop();

 return 0;
}

```

## 16.4 examples/libs/l4re/c++/mem\_alloc/ma+rm.cc

Coarse grained memory allocation, in C++.

```

/*
 * (c) 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.
 */

#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::Search_addr,
 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;
}

```

## 16.5 examples/libs/l4re/c++/shared\_ds/ds\_clnt.cc

Sharing memory between applications, client side.

```

/*
 * (c) 2009 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 General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */

#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
 */
L4Re::Cap<L4Re::Irq> irq = L4Re::Util::cap_alloc.
 alloc<L4Re::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::Search_addr,
 L4Re::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;
}

```

## 16.6 examples/libs/l4re/c++/shared\_ds/ds\_srv.cc

Sharing memory between applications, server/creator side.

```

/*
 * (c) 2009 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 General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */

```

```

#include <l4/re/env>
#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/sys/typeinfo_svr>

#include <cstring>
#include <cstdio>
#include <unistd.h>

#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;
}

static L4Re::Util::Registry_server<> server;

```

```

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::Search_addr,
 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 * const msg = "[DS] Hello from server!";
 snprintf(_addr, strlen(msg) + 1, msg);

 return _addr;
}

int main()
{
 L4::Cap<L4Re::Dataspace> ds;
 char *addr;

 if (!(addr = get_ds(&ds)))
 return 2;

 // 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();
 return 0;
}

```

## 16.7 examples/libs/l4re/c++/shared\_ds/shared\_ds.cfg

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"
);

```

## 16.8 examples/libs/l4re/c/ma+rm.c

Coarse grained memory allocation, in C.

```

/*
 * (c) 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.
 */

#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_SEARCH_ADDR, 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;
}

```

## 16.9 examples/libs/l4re/streammap/client.cc

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.

```

/*
 * (c) 2008-2009 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 General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#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::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;
}

```

## 16.10 examples/libs/l4re/streammap/server.cc

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.

```

/*
 * (c) 2008-2009 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 General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#include <stdio.h>
#include <l4/re/env>
#include <l4/re/util/cap_alloc>
#include <l4/re/util/object_registry>
#include <l4/cxx/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;
}

```

## 16.11 examples/libs/l4re/streammap/streammap.cfg

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");

```

## 16.12 examples/libs/libirq/async\_isr.c

libirq usage example using asynchronous ISR handler functionality.

```

/*
 * (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 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;
}

```

## 16.13 examples/libs/libirq/loop.c

libirq usage example using a self-created thread.

```

/*
 * (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.
 */

#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;
}

```

## 16.14 examples/libs/shmc/prodcons.c

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>

// a small helper
#define CHK(func) if (func) { printf("failure: %d\n", __LINE__); return (void *)-1; }

static const char some_data[] = "Hi consumer!";

static void *thread_producer(void *d)
{
 (void)d;
 l4shmc_chunk_t p_one;
 l4shmc_signal_t s_one, s_done;
 l4shmc_area_t shmarea;

 // 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, "prod", &s_one));

 CHK(l4shmc_attach_signal_to(&shmarea, "done",
 pthread_l4_cap(pthread_self()), 10000, &s_done));

 // connect chunk and signal
 CHK(l4shmc_connect_chunk_signal(&p_one, &s_one));
}

```

```

printf("PRODUCER: 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)));

 printf("PRODUCER: Sent data\n");

 CHK(l4shmc_wait_signal(&s_done));
}

l4_sleep_forever();
return NULL;
}

static void *thread_consume(void *d)
{
 (void)d;
 l4shmc_area_t shmarea;
 l4shmc_chunk_t p_one;
 l4shmc_signal_t s_one, s_done;

 // 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, "done", &s_done));

 // attach signal to this thread
 CHK(l4shmc_attach_signal_to(&shmarea, "prod",
 pthread_l4_cap(pthread_self()), 10000, &s_one));

 // connect chunk and signal
 CHK(l4shmc_connect_chunk_signal(&p_one, &s_one));

 while (1)
 {
 CHK(l4shmc_wait_chunk(&p_one));

 printf("CONSUMER: 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;

 // create new shared memory area, 8K in size
 if (l4shmc_create("testshm", 8192))
 return 1;

 // create two threads, one for producer, one for consumer
 pthread_create(&one, 0, thread_producer, 0);
 pthread_create(&two, 0, thread_consume, 0);

 // now sleep, the two threads are doing the work
 l4_sleep_forever();

 return 0;
}

```

## 16.15 examples/sys/aliens/main.c

This example shows how system call tracing can be done.

```

/*
 * (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>
 * 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.
 */
/*
 * 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/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");
}

#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;
}

```

## 16.16 examples/sys/ipc/ipc.cfg

Sample configuration file for the IPC example.

```

vim:se ft=lua:

local L4 = require("L4");

L4.default_loader:start({}, "rom/ex_ipc1");

```

## 16.17 examples/sys/ipc/ipc\_example.c

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.

```

/*
 * (c) 2008-2009 Author(s)
 * 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.
 */
#include <l4/sys/ipc.h>

#include <pthread-l4.h>
#include <unistd.h>
#include <stdio.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;
}

```

## 16.18 examples/sys/isr/main.c

Example of an interrupt service routine.

```

/*
 * (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>
 * 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 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. The interrupt number of the virtual key interrupt can be
 * found in the kernel info page.
 */

#include <l4/re/c/util/cap_alloc.h>
#include <l4/re/c/namespace.h>
#include <l4/sys/utcb.h>
#include <l4/sys/irq.h>
#include <l4/sys/factory.h>
#include <l4/sys/icu.h>

#include <stdio.h>

int main(void)
{
 int irqno = 1;
 l4_cap_idx_t irqcap, icucap;
 l4_msgtag_t tag;
 int err;
 icucap = l4re_env_get_cap("icu");

 /* Get a free capability slot for the ICU capability */
 if (l4_is_invalid_cap(icucap))
 {
 printf("Did not find an ICU\n");
 return 1;
 }

 /* Get another free capability slot for the corresponding IRQ object */
 if (l4_is_invalid_cap(irqcap = l4re_util_cap_alloc()))
 return 1;
 /* Create IRQ object */
 if (l4_error(tag = l4_factory_create_irq(l4re_global_env->
 factory, irqcap)))
 {
 printf("Could not create IRQ object: %lx\n", l4_error(tag));
 return 1;
 }

 /*
 * Bind the recently allocated IRQ object to the IRQ number irqno
 * as provided by the ICU.
 */
 if (l4_error(l4_icu_bind(icucap, irqno, irqcap)))
 {
 printf("Binding IRQ%d to the ICU failed\n", irqno);
 return 1;
 }

 /* Attach ourselves to the IRQ */
 tag = l4_irq_attach(irqcap, 0xDEAD, l4re_env()->main_thread);
 if ((err = l4_error(tag)))
 {
 printf("Error attaching to IRQ %d: %d\n", irqno, err);
 return 1;
 }

 printf("Attached to key IRQ %d\nPress keys now, Shift-Q to exit\n", irqno);

 /* IRQ receive loop */
 while (1)
 {
 unsigned long label = 0;
 /* Wait for the interrupt to happen */
 tag = l4_irq_receive(irqcap, L4_IPC_NEVER);
 if ((err = l4_ipc_error(tag, l4_utcb())))

```

```

 printf("Error on IRQ receive: %d\n", err);
 else
 {
 /* Process the interrupt -- may do a 'break' */
 printf("Got IRQ with label 0x%lX\n", label);
 }
}

/* We're done, detach from the interrupt. */
tag = l4_irq_detach(irqcap);
if ((err = l4_error(tag)))
 printf("Error detach from IRQ: %d\n", err);

return 0;
}

```

## 16.19 examples/sys/migrate/thread\_migrate.cc

Thread migration example.

```

/*
 * (c) 2008-2009 Author(s)
 * 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.
 */
#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;
}

```

## 16.20 examples/sys/migrate/thread\_migrate.cfg

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");

```

## 16.21 examples/sys/singlestep/main.c

This example shows how a thread can be single stepped on the x86 architecture.

```

/*
 * (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>
 * 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.
 */
/*
 * 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/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)
 enter_kdebug("Should not be 1");
 }
 else
 {
 tag = l4_msgtag(L4_PROTO_NONE,
 L4_UTCB_EXCEPTION_REGS_SIZE, 0, 0);

 if (!ipc_stat)
 enter_kdebug("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;
}

```



## 16.22 examples/sys/start-with-exc/main.c

This example shows how to start a newly created thread with a defined set of CPU registers.

```

/*
 * (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>
 * 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.
 */
/*
 * 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/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 %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\t"
 "thread:\n\t"
 #ifdef ARCH_x86
 " pusha\n\t"
 " push %esp\n\t"
 " call thread_func\n\t"
 #endif
 #ifdef ARCH_arm
 " push {r0-r7}\n\t"
 " mov r0, sp\n\t"
 " bl thread_func\n\t"
 #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
/* 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(t1, 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;
}

```

## 16.23 examples/sys/utcb-ipc/main.c

This example shows how to send IPC using the UTCB to store payload.

```

/*
 * (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>
 * 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.
 */
#include <l4/sys/ipc.h>
#include <l4/sys/thread.h>
#include <l4/sys/factory.h>
#include <l4/sys/utcb.h>
#include <l4/re/env.h>
#include <l4/re/c/util/cap_alloc.h>
#include <l4/util/thread.h>

#include <stdio.h>
#include <string.h>

static unsigned char stack2[8 << 10];
static l4_cap_idx_t thread1_cap, thread2_cap;

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");
 }
}

L4UTIL_THREAD_STATIC_FUNC(thread2)
{
 l4_msgtag_t tag;
 l4_msgregs_t mr;
 unsigned i;

 printf("Thread2 up (%p)\n", l4_utcb());

 while (1)
 {
 if (l4_msgtag_has_error(tag = l4_ipc_receive(thread1_cap,
 l4_utcb(), L4_IPC_NEVER)))
 printf("IPC receive error\n");
 memcpy(&mr, l4_utcb_mr(), sizeof(mr));
 printf("Thread2 receive (%d): ", l4_msgtag_words(tag));
 for (i = 0; i < l4_msgtag_words(tag); i++)
 printf("%c", (char)mr.mr[i]);
 printf("\n");
 }

 __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))
 return 1;

 tag = l4_factory_create_thread(l4re_env()->factory, thread2_cap);
 if (l4_error(tag))

```

```

 return 1;

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 *)l4re_env()->first_free_utcb,
 L4RE_THIS_TASK_CAP);
tag = l4_thread_control_commit(thread2_cap);
if (l4_error(tag))
 return 2;

tag = l4_thread_ex_regs(thread2_cap,
 (l4_umword_t)thread2,
 (l4_umword_t)(stack2 + sizeof(stack2)), 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, thread2_cap, &sp);
if (l4_error(tag))
 return 4;

thread1();

return 0;
}

```

## 16.24 examples/sys/ux-vhw/main.c

This example shows how to iterate the virtual hardware descriptors under Fiasco-UX.

```

/*
 * (c) 2008-2009 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 General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#include <l4/sys/ipc.h>
#include <l4/sys/vhw.h>
#include <l4/util/util.h>
#include <l4/util/kip.h>
#include <l4/re/env.h>

#include <stdlib.h>
#include <stdio.h>

static void print_entry(struct l4_vhw_entry *e)
{
 printf("type: %d mem start: %08lx end: %08lx\n"
 "irq: %d pid %d\n",
 e->type, e->mem_start, e->mem_size,
 e->irq_no, e->provider_pid);
}

int main(void)
{
 l4_kernel_info_t *kip = l4re_kip();
 struct l4_vhw_descriptor *vhw;
 int i;

 if (!kip)
 {
 printf("KIP not available!\n");
 return 1;
 }

 if (!l4util_kip_kernel_is_ux(kip))
 {
 printf("This example is for Fiasco-UX only.\n");
 return 1;
 }

 vhw = l4_vhw_get(kip);

 printf("kip at %p, vhw at %p\n", kip, vhw);
 printf("magic: %08x, version: %08x, count: %02d\n",
 vhw->magic, vhw->version, vhw->count);
}

```

```

for (i = 0; i < vhw->count; i++)
 print_entry(l4_vhw_get_entry(vhw, i));

return 0;
}

```

## 16.25 hello/server/src/main.c

This is the famous "Hello World!" program.

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 * Frank Mehnert <fm3@os.inf.tu-dresden.de>,
 * Lukas Grützmacher <lg2@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.
 */
#include <stdio.h>
#include <unistd.h>

int
main(void)
{
 for (;;)
 {
 puts("Hello World!");
 sleep(1);
 }
}

```

## 16.26 tmpfs/lib/src/fs.cc

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>

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() throw() { 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(&_amp;_info, 0, sizeof(_info));
 _info.st_mode = mode;
 }

 const char *path() const { return _path; }
 struct stat64 *info() { return &_amp;_info; }

 void add_ref() throw() { ++_ref_cnt; }
 int remove_ref() throw() { 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 &n);

 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) throw()
 : _dir(d), _getdents_state(false) {}
 int get_entry(const char *, int, mode_t, Ref_ptr<File> *) throw();
 ssize_t getdents(char *, size_t) throw();
 int fstat64(struct stat64 *buf) const throw();
 int utime(const struct utimbuf *) throw();
 int fchmod(mode_t) throw();
 int mkdir(const char *, mode_t) throw();
 int unlink(const char *) throw();
 int rename(const char *, const char *) throw();
 int faccessat(const char *, int, int) throw();

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) throw()
 : Be_file_pos(), _file(f) {}

 off64_t size() const throw();
 int fstat64(struct stat64 *buf) const throw();
 int ftruncate64(off64_t p) throw();
 int ioctl(unsigned long, va_list) throw();

```

```

 int utime(const struct utimbuf *) throw();
 int fchmod(mode_t) throw();

private:
 ssize_t preadv(const struct iovec *v, int iovcnt, off64_t p) throw();
 ssize_t pwritev(const struct iovec *v, int iovcnt, off64_t p) throw();
 Ref_ptr<Pers_file> _file;
};

ssize_t Tmpfs_file::preadv(const struct iovec *v, int iovcnt, off64_t p) throw()
{
 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) throw()
{
 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::fstat64(struct stat64 *buf) const throw()
{
 _file->info()->st_size = _file->data().size();
 memcpy(buf, _file->info(), sizeof(*buf));
 return 0;
}

int Tmpfs_file::ftruncate64(off64_t p) throw()
{
 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 throw()
{
 return _file->data().size();
}

int
Tmpfs_file::ioctl(unsigned long v, va_list args) throw()
{
 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) throw()
{
 _file->info()->st_atime = times->actime;
 _file->info()->st_mtime = times->modtime;
 return 0;
}

int
Tmpfs_file::fchmod(mode_t m) throw()
{
 _file->info()->st_mode = m;
 return 0;
}

int

```



```

Tmpfs_dir::faccessat(const char *path, int mode, int) throw()
{
 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) throw()
{
 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) throw()
{
 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) + 1;
 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::fstat64(struct stat64 *buf) const throw()
{
 memcpy(buf, _dir->info(), sizeof(*buf));
 return 0;
}

int
Tmpfs_dir::utime(const struct utimbuf *times) throw()
{
 _dir->info()->st_atime = times->actime;
 _dir->info()->st_mtime = times->modtime;
 return 0;
}

int
Tmpfs_dir::fchmod(mode_t m) throw()
{
 _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) throw()
{
 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) throw()
{
 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) throw()
{
 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) throw()
 {
 (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;
}
```

# Index

- \_\_lface
    - L4::Kobject\_2t, [1064](#)
    - L4::Kobject\_3t, [1068](#)
  - \_\_lface\_list
    - L4::Kobject\_2t, [1064](#)
    - L4::Kobject\_3t, [1068](#)
  - \_\_L4UTIL\_THREAD\_FUNC
    - l4/util/thread.h, [2253](#)
  - \_\_check\_protocols\_\_
    - L4::Kobject\_2t, [1065](#)
    - L4::Kobject\_3t, [1069](#)
  - \_bus
    - L4vbus::Device, [1497](#)
  - \_c
    - L4::Cap\_base, [868](#)
  - ~Auto\_ptr
    - cxx::Auto\_ptr, [681](#)
  - ~Be\_file\_system
    - L4Re::Vfs::Be\_file\_system, [1405](#)
  - ~H\_list\_item\_t
    - cxx::H\_list\_item\_t, [780](#)
- a
  - L4Re::Video::Pixel\_info, [1450](#)
- ARM Virtual Registers (UTCB), [89](#)
- Access\_width
  - L4Re::Mmio\_space, [1309](#)
- acked
  - L4virtio::Svr::Dev\_status, [1589](#)
- acked\_bfm\_t
  - L4virtio::Svr::Dev\_status, [1588](#)
- acquire
  - L4Re::Inhibitor, [1296](#)
- add
  - L4::lpc\_svr::Timeout\_queue, [1026](#)
  - L4::Thread::Modify\_senders, [1153](#)
  - L4vcpu::State, [1535](#)
  - L4virtio::Svr::Driver\_mem\_list\_t, [1599](#)
- add\_ku\_mem
  - L4::Task, [1133](#)
- add\_timeout
  - L4::lpc\_svr::Server\_iface, [1016](#)
  - L4::lpc\_svr::Timeout\_queue\_hooks, [1031](#)
- align\_to
  - L4::lpc::Msg, [664](#), [665](#)
- all
  - L4::Kip::Mem\_desc, [1051](#), [1052](#)
- alloc
  - cxx::Base\_slab\_static, [705](#)
  - cxx::List\_alloc, [788](#)
  - cxx::Slab, [819](#)
  - cxx::Slab\_static, [823](#)
  - L4Re::Cap\_alloc, [1248](#), [1249](#)
  - L4Re::Mem\_alloc, [1305](#)
  - L4Re::Util::Counting\_cap\_alloc, [1348](#)
- alloc\_buffer\_demand
  - L4::lpc\_svr::Br\_manager\_no\_buffers, [1003](#)
  - L4::lpc\_svr::Server\_iface, [1016](#)
  - L4Re::Util::Br\_manager, [1340](#)
- alloc\_descriptor
  - L4virtio::Driver::Virtqueue, [1551](#)
- alloc\_dynamic\_entry
  - L4Re::Util::Names::Name\_space, [1378](#)
- alloc\_fd
  - L4Re::Vfs::Fs, [1417](#)
- alloc\_max
  - cxx::List\_alloc, [789](#)
- allocate
  - L4Re::Dataspace, [1257](#)
  - L4Re::Util::Dataspace\_svr, [1354](#)
- amd64 Virtual Registers (UTCB), [626](#)
- amd64/l4/sys/cache.h, [2000](#), [2001](#)
- amd64/l4/sys/consts.h, [2015](#), [2016](#)
- amd64/l4/sys/l4int.h, [2102](#), [2103](#)
- amd64/l4/sys/linkage.h, [1728](#)
- amd64/l4/sys/segment.h, [1703](#), [1707](#)
  - fiasco\_amd64\_segment\_info, [1705](#)
  - fiasco\_amd64\_set\_fs, [1706](#)
  - fiasco\_amd64\_set\_segment\_base, [1706](#)
  - L4\_task\_ldt\_x86\_consts, [1705](#)
- amd64/l4/sys/utcb.h, [2152](#), [2154](#)
- amd64/l4/util/apic.h, [1657](#), [1658](#)
- amd64/l4/util/bitops\_arch.h, [1734](#)
- amd64/l4/util/cpu.h, [1742](#), [1743](#)
- amd64/l4/util/idt.h, [1668](#), [1669](#)
- amd64/l4/util/irq.h, [1838](#), [1840](#)
  - l4util\_irq\_acknowledge, [1839](#)
- amd64/l4/util/l4\_macros.h, [1747](#)
- amd64/l4/util/mbi\_argv.h, [1750](#), [1751](#)
- amd64/l4/util/perform.h, [1671](#), [1672](#)
- amd64/l4/util/port\_io.h, [1713](#), [1714](#)
- amd64/l4/util/rdtsc.h, [1683](#), [1685](#)
- amd64/l4/util/spin.h, [1693](#)
- amd64/l4/util/stack\_impl.h, [1754](#), [1755](#)
  - l4util\_stack\_get\_sp, [1754](#)
- amd64/l4/util/util.h, [1695](#), [1697](#)
  - l4\_sleep, [1696](#)
  - l4util\_micros2l4to, [1696](#)
- amd64/l4f/l4/sys/segment.h, [1700](#), [1702](#)

- fiasco\_amd64\_set\_fs, 1701
  - fiasco\_amd64\_set\_segment\_base, 1702
- amd64/l4f/l4/util/port\_io.h, 1712, 1713
- amd64/l4f/l4/util/setjmp.h, 1721, 1723
  - l4\_thread\_longjmp, 1722
  - l4\_thread\_setjmp, 1723
- arm/l4/sys/atomic.h, 2185, 2186
- arm/l4/sys/cache.h, 1997, 1999
- arm/l4/sys/consts.h, 2014, 2015
- arm/l4/sys/l4int.h, 2102
- arm/l4/sys/linkage.h, 1727
- arm/l4/sys/mem\_op.h, 1729, 1731
- arm/l4/sys/thread.h, 2264
- arm/l4/sys/utcb.h, 2150, 2151
- arm/l4/sys/vm.h, 1732
- arm/l4/util/bitops\_arch.h, 1733, 1734
- arm/l4/util/cpu.h, 1741, 1742
- arm/l4/util/irq.h, 1837, 1838
- arm/l4/util/l4\_macros.h, 1746
- arm/l4/util/mbi\_argv.h, 1749, 1750
- arm/l4/util/stack\_impl.h, 1753, 1754
  - l4util\_stack\_get\_sp, 1753
- arm/l4f/l4/sys/ipc.h, 2080, 2081
- arm/l4f/l4/sys/syscall\_defs.h, 1756, 1757
- associate
  - L4Re::Dma\_space, 1269
- Atomic Instructions, 90
  - l4util\_add8, 92
  - l4util\_add8\_res, 92
  - l4util\_atomic\_add, 92
  - l4util\_atomic\_inc, 93
  - l4util\_cmpxchg, 93
  - l4util\_cmpxchg16, 94
  - l4util\_cmpxchg32, 94
  - l4util\_cmpxchg64, 96
  - l4util\_cmpxchg8, 96
  - l4util\_inc8, 97
  - l4util\_inc8\_res, 97
  - l4util\_xchg, 97
  - l4util\_xchg16, 98
  - l4util\_xchg32, 98
  - l4util\_xchg8, 99
- attach
  - L4::Irq, 1035
  - L4Re::Rm, 1326, 1327
  - L4Re::Util::Event\_buffer\_t, 1368
- Attach\_flags
  - L4Re::Rm, 1325
- Attr
  - L4::Thread::Attr, 1150
- Attribute
  - L4Re::Dma\_space, 1268
- Attributes
  - L4Re::Dma\_space, 1267
- Auto\_ptr
  - cxx::Auto\_ptr, 681
- auto\_refresh
  - L4Re::Video::Goos::Info, 1447
- Auxiliary data, 100
- avail
  - cxx::List\_alloc, 790
- avail\_align
  - L4virtio::Virtqueue, 1627
- avail\_size
  - L4virtio::Virtqueue, 1627
- Avl\_map
  - cxx::Avl\_map, 686
- ax
  - l4\_vcpu\_regs\_t, 1232
- b
  - L4Re::Video::Pixel\_info, 1451
- backtrace.h
  - l4util\_backtrace, 2189
- Bad\_descriptor
  - L4virtio::Svr::Bad\_descriptor, 1559
- Base API, 101
- Base\_avl\_set
  - cxx::Bits::Base\_avl\_set, 744
- Basic Macros, 104
  - L4\_ALWAYS\_INLINE, 105
  - L4\_HIDDEN, 105
  - L4\_NOTHROW, 105
- Be\_file\_system
  - L4Re::Vfs::Be\_file\_system, 1405
- begin
  - cxx::Bits::Base\_avl\_set, 744, 745
  - cxx::Bits::Bst, 757, 758
- bind
  - L4::lcu, 904
  - L4::lommu, 918
  - L4::Thread::Attr, 1150
- bind\_thread
  - L4::lpc\_gate, 999
- bit
  - cxx::Bitmap\_base, 729
- Bit Manipulation, 107
  - l4util\_bsf, 107
  - l4util\_bsr, 108
  - l4util\_btc, 109
  - l4util\_btr, 109
  - l4util\_bts, 110
  - l4util\_clear\_bit, 111
  - l4util\_complement\_bit, 111
  - l4util\_find\_first\_set\_bit, 112
  - l4util\_find\_first\_zero\_bit, 112
  - l4util\_next\_power2, 112
  - l4util\_set\_bit, 113
  - l4util\_test\_bit, 113
- bit\_index
  - cxx::Bitmap\_base, 730
- Bitmap
  - cxx::Bitmap, 724
- Bitmap graphics and fonts, 115
- bitmap.h
  - pSLIM\_BMAP\_START\_LSB, 1851
- bits\_per\_pixel

- L4Re::Video::Pixel\_info, [1451](#)
- Bits\_type
  - cxx::Bitfield, [708](#)
- blk\_size
  - I4virtio\_block\_config\_t, [1651](#)
- Block\_dev
  - L4virtio::Svr::Block\_dev, [1562](#)
- bp
  - I4\_vcpu\_regs\_t, [1233](#)
- buf
  - L4Re::Util::Event\_buffer\_t, [1369](#)
- buf\_cp\_in
  - L4::lpc, [656](#)
- buf\_cp\_out
  - L4::lpc, [657](#)
- buf\_in
  - L4::lpc, [657](#)
- buffer
  - L4Re::Util::Event\_t, [1371](#)
- Buffer Registers (BRs), [116](#)
  - I4\_buffer\_desc\_consts\_t, [116](#)
- bus\_cap
  - L4vbus::Device, [1491](#)
- bx
  - I4\_vcpu\_regs\_t, [1233](#)
- bytes\_per\_pixel
  - L4Re::Video::Pixel\_info, [1452](#)
- c
  - L4::Kobject\_2t, [1065](#)
  - L4::Kobject\_3t, [1069](#)
- C++ Exceptions, [118](#)
- C++ IPC Interface Definition., [119](#)
- CPU related functions, [120](#)
  - I4util\_cpu\_capabilities, [120](#)
  - I4util\_cpu\_capabilities\_nocheck, [121](#)
  - I4util\_cpu\_has\_cpuid, [122](#)
- Cache Consistency, [123](#)
  - I4\_cache\_clean\_data, [123](#)
  - I4\_cache\_coherent, [124](#)
  - I4\_cache\_dma\_coherent, [124](#)
  - I4\_cache\_flush\_data, [125](#)
  - I4\_cache\_inv\_data, [125](#)
- call
  - L4::lpc::loststream, [942](#)
- Cap
  - L4::Cap, [855](#), [856](#)
  - L4::lpc::Cap, [933](#)
- cap
  - L4::Cap\_base, [861](#)
  - L4::Invalid\_capability, [913](#)
  - L4::Kobject, [1058](#)
- cap\_alloc
  - L4Re Capability API, [382](#)
- Cap\_base
  - L4::Cap\_base, [860](#)
- cap\_cast
  - L4, [647](#), [648](#)
- cap\_dynamic\_cast
  - L4, [649](#)
- cap\_equal
  - L4::Task, [1134](#)
- cap\_has\_child
  - L4::Task, [1134](#)
- cap\_received
  - L4::lpc::Gen\_fpage, [936](#)
- cap\_reinterpret\_cast
  - L4, [650](#)
- Cap\_type
  - L4::Cap\_base, [860](#)
- cap\_valid
  - L4::Task, [1135](#)
- Capabilities, [127](#)
  - I4\_cap\_consts\_t, [128](#)
  - I4\_capability\_equal, [129](#)
  - I4\_default\_caps\_t, [128](#)
  - I4\_is\_invalid\_cap, [129](#)
  - I4\_is\_valid\_cap, [130](#)
- capability
  - L4\_DISABLE\_COPY, [2005](#)
- Capability allocator, [131](#)
  - I4re\_util\_cap\_last, [131](#)
- cast
  - L4vcpu::Vcpu, [1540](#)
- cfg\_read
  - L4vbus::Pci\_dev, [1518](#)
  - L4vbus::Pci\_host\_bridge, [1523](#)
- cfg\_write
  - L4vbus::Pci\_dev, [1518](#)
  - L4vbus::Pci\_host\_bridge, [1524](#)
- chain
  - L4::Irq\_mux, [1047](#)
- change\_queue\_config
  - L4virtio::Svr::Dev\_config, [1575](#)
- chars
  - cxx::Bitmap\_base, [731](#)
- check\_size
  - L4::lpc::Msg, [666](#)
- chkcap
  - L4Re, [672](#)
- chksys
  - L4Re, [672–674](#)
- Chunks, [132](#)
  - I4shmc\_add\_chunk, [132](#)
  - I4shmc\_chunk\_capacity, [133](#)
  - I4shmc\_chunk\_ptr, [133](#)
  - I4shmc\_chunk\_signal, [134](#)
  - I4shmc\_get\_chunk, [134](#)
  - I4shmc\_get\_chunk\_to, [135](#)
  - I4shmc\_iterate\_chunk, [135](#)
- Class
  - L4::Kobject\_2t, [1064](#)
  - L4::Kobject\_3t, [1068](#)
- clear
  - cxx::Bits::Basic\_list, [753](#)
  - L4::Types::Flags, [1190](#)
  - L4Re::Dataspace, [1257](#)

- L4Re::Util::Dataspace\_svr, [1355](#)
  - L4vcpu::State, [1535](#)
- clear\_bit
  - cxx::Bitmap\_base, [731](#)
- cnt\_jobmap\_tlb\_flush
  - l4\_tracebuffer\_status\_t, [1227](#)
- Color\_component
  - L4Re::Video::Color\_component, [1437](#)
- Com\_error
  - L4::Com\_error, [871](#)
- Comfortable Command Line Parsing, [136](#)
  - parse\_cmdline, [136](#)
- config\_get
  - L4vbus::Gpio\_pin, [1508](#)
- config\_pad
  - L4vbus::Gpio\_module, [1501](#)
  - L4vbus::Gpio\_pin, [1509](#)
- config\_pull
  - L4vbus::Gpio\_pin, [1509](#)
- Console API, [139](#)
- consumed
  - L4virtio::Svr::Virtqueue, [1618](#)
- Consumer, [140](#), [144](#)
  - l4shmc\_chunk\_consumed, [140](#)
  - l4shmc\_chunk\_size, [141](#)
  - l4shmc\_enable\_chunk, [141](#)
  - l4shmc\_enable\_signal, [144](#)
  - l4shmc\_is\_chunk\_ready, [142](#)
  - l4shmc\_wait\_any, [145](#)
  - l4shmc\_wait\_any\_to, [145](#)
  - l4shmc\_wait\_any\_try, [145](#)
  - l4shmc\_wait\_chunk, [142](#)
  - l4shmc\_wait\_chunk\_to, [142](#)
  - l4shmc\_wait\_chunk\_try, [143](#)
  - l4shmc\_wait\_signal, [146](#)
  - l4shmc\_wait\_signal\_to, [146](#)
  - l4shmc\_wait\_signal\_try, [147](#)
- contains
  - L4virtio::Svr::Driver\_mem\_region\_t, [1606](#)
- contrib/libio-io/l4/io/io.h, [1757](#), [1760](#)
- control
  - L4::Thread, [1142](#)
- copy
  - L4::Cap, [857](#)
  - L4::Cap\_base, [862](#)
  - L4Re::Util::Dataspace\_svr, [1355](#)
- copy\_in
  - L4Re::Dataspace, [1258](#)
- copy\_receive\_cap
  - L4Re::Util::Names::Name\_space, [1378](#)
- copy\_to
  - L4virtio::Svr::Data\_buffer, [1570](#)
- count
  - L4::Kip::Mem\_desc, [1052](#), [1053](#)
  - l4\_vhw\_descriptor, [1239](#)
- Counting\_cap\_alloc
  - L4Re::Util::Counting\_cap\_alloc, [1348](#)
- cpu\_disable
  - L4::Platform\_control, [1090](#)
- cpu\_enable
  - L4::Platform\_control, [1091](#)
- create
  - L4::Factory, [887](#), [888](#)
- create\_buffer
  - L4Re::Video::Goos, [1443](#)
- create\_factory
  - L4::Factory, [889](#)
- create\_gate
  - L4::Factory, [890](#)
- create\_irq
  - L4::Factory, [891](#)
- create\_task
  - L4::Factory, [892](#)
- create\_thread
  - L4::Factory, [893](#)
- create\_view
  - L4Re::Video::Goos, [1443](#)
- create\_vm
  - L4::Factory, [894](#)
- current\_flags
  - L4virtio::Svr::Request\_processor, [1611](#)
- cx
  - l4\_vcpu\_regs\_t, [1233](#)
- cxx, [641](#)
- cxx::Auto\_ptr
  - ~Auto\_ptr, [681](#)
  - Auto\_ptr, [681](#)
  - get, [681](#)
  - operator Priv\_type \*, [682](#)
  - operator\*, [682](#)
  - operator->, [682](#)
  - operator=, [682](#)
  - Ref\_type, [680](#)
  - release, [683](#)
- cxx::Auto\_ptr< T >, [679](#)
- cxx::Avl\_map
  - Avl\_map, [686](#)
  - insert, [687](#)
  - operator[], [687](#), [688](#)
- cxx::Avl\_map< KEY\_TYPE, DATA\_TYPE, COMPARE, ALLOC >, [684](#)
- cxx::Avl\_set< ITEM\_TYPE, COMPARE, ALLOC >, [688](#)
- cxx::Avl\_tree
  - insert, [695](#)
  - remove, [695](#)
- cxx::Avl\_tree< Node, Get\_key, Compare >, [691](#)
- cxx::Avl\_tree\_node, [696](#)
- cxx::Base\_slab
  - free\_objects, [701](#)
  - total\_objects, [701](#)
- cxx::Base\_slab< Obj\_size, Slab\_size, Max\_free, Alloc >, [699](#)
- cxx::Base\_slab\_static
  - alloc, [705](#)
  - free, [705](#)
  - free\_objects, [706](#)



- total\_objects, [706](#)
- cxx::Base\_slab\_static< Obj\_size, Slab\_size, Max\_free, Alloc >, [702](#)
- cxx::Bitfield
  - Bits\_type, [708](#)
  - get, [710](#)
  - get\_unshifted, [711](#)
  - Masks, [710](#)
  - Ref, [708](#)
  - Ref\_unshifted, [709](#)
  - set, [711](#)
  - set\_dirty, [712](#)
  - set\_unshifted, [713](#)
  - set\_unshifted\_dirty, [714](#)
  - Shift\_type, [709](#)
  - Val, [709](#)
  - val, [715](#)
  - val\_dirty, [715](#)
  - Val\_unshifted, [709](#)
  - val\_unshifted, [716](#)
- cxx::Bitfield< T, LSB, MSB >, [707](#)
- cxx::Bitfield< T, LSB, MSB >::Value< TT >, [717](#)
- cxx::Bitfield< T, LSB, MSB >::Value\_base< TT >, [718](#)
- cxx::Bitfield< T, LSB, MSB >::Value\_unshifted< TT >, [720](#)
- cxx::Bitmap
  - Bitmap, [724](#)
  - scan\_zero, [724](#)
- cxx::Bitmap< BITS >, [721](#)
- cxx::Bitmap\_base, [725](#)
  - bit, [729](#)
  - bit\_index, [730](#)
  - chars, [731](#)
  - clear\_bit, [731](#)
  - operator[], [732](#)
  - scan\_zero, [733](#)
  - set\_bit, [734](#)
  - word\_index, [735](#)
  - words, [735](#)
- cxx::Bitmap\_base::Bit, [736](#)
- cxx::Bitmap\_base::Char< BITS >, [737](#)
- cxx::Bitmap\_base::Word< BITS >, [737](#)
- cxx::Bits, [643](#)
- cxx::Bits::Avl\_map\_get\_key< KEY\_TYPE >, [739](#)
- cxx::Bits::Avl\_set\_get\_key< KEY\_TYPE >, [739](#)
- cxx::Bits::Base\_avl\_set
  - Base\_avl\_set, [744](#)
  - begin, [744](#), [745](#)
  - end, [745](#), [746](#)
  - erase, [746](#)
  - find\_node, [747](#)
  - insert, [747](#)
  - lower\_bound\_node, [748](#)
  - rbegin, [748](#)
  - remove, [748](#)
  - rend, [749](#)
- cxx::Bits::Base\_avl\_set< ITEM\_TYPE, COMPARE, A↔LLOC, GET\_KEY >, [740](#)
- cxx::Bits::Base\_avl\_set< ITEM\_TYPE, COMPARE, A↔LLOC, GET\_KEY >::Node, [750](#)
- cxx::Bits::Base\_avl\_set::Node
  - operator\*, [751](#)
  - operator->, [751](#)
  - valid, [751](#)
- cxx::Bits::Basic\_list
  - clear, [753](#)
  - iter, [753](#)
- cxx::Bits::Basic\_list< POLICY >, [752](#)
- cxx::Bits::Bst
  - begin, [757](#), [758](#)
  - dir, [758](#), [759](#)
  - end, [760](#)
  - find, [761](#)
  - find\_node, [762](#)
  - lower\_bound\_node, [762](#)
  - rbegin, [763](#), [764](#)
  - remove\_all, [764](#)
  - remove\_tree, [765](#)
  - rend, [765](#), [766](#)
- cxx::Bits::Bst< Node, Get\_key, Compare >, [754](#)
- cxx::Bits::Bst\_node, [767](#)
- cxx::Bits::Direction, [769](#)
  - Direction\_e, [770](#)
  - operator!, [770](#)
- cxx::H\_list
  - erase, [774](#)
  - insert, [775](#)
  - insert\_after, [775](#)
  - insert\_before, [776](#)
  - iter, [776](#)
  - pop\_front, [776](#)
  - remove, [777](#)
  - replace, [777](#)
- cxx::H\_list< T, POLICY >, [771](#)
- cxx::H\_list\_item\_t
  - ~H\_list\_item\_t, [780](#)
  - H\_list\_item\_t, [779](#)
- cxx::H\_list\_item\_t< ELEM\_TYPE >, [778](#)
- cxx::H\_list\_t< T >, [780](#)
- cxx::List
  - items, [784](#)
  - operator[], [784](#)
  - push\_back, [784](#)
  - push\_front, [785](#)
  - remove, [785](#)
  - size, [785](#)
- cxx::List< D, Alloc >, [783](#)
- cxx::List< D, Alloc >::Iter, [786](#)
- cxx::List\_alloc, [787](#)
  - alloc, [788](#)
  - alloc\_max, [789](#)
  - avail, [790](#)
  - free, [790](#)
  - List\_alloc, [787](#)
- cxx::List\_item, [791](#)
  - get\_next\_item, [793](#)

- get\_prev\_item, [793](#)
- insert\_next\_item, [793](#)
- insert\_prev\_item, [794](#)
- push\_back, [794](#)
- push\_front, [795](#)
- remove, [795](#)
- remove\_me, [796](#)
- cxx::List\_item::lter, [797](#)
- remove\_me, [799](#)
- cxx::List\_item::T\_iter< T, Poly >, [800](#)
- cxx::Lt\_functor< Obj >, [803](#)
- cxx::New\_allocator< \_Type >, [803](#)
- cxx::Nothrow, [804](#)
- cxx::Pair
  - Pair, [806](#)
- cxx::Pair< First, Second >, [805](#)
- cxx::Pair\_first\_compare
  - operator(), [807](#)
  - Pair\_first\_compare, [807](#)
- cxx::Pair\_first\_compare< Cmp, Typ >, [806](#)
- cxx::Ref\_ptr
  - get, [812](#)
  - ptr, [812](#)
  - Ref\_ptr, [811](#), [812](#)
  - release, [813](#)
- cxx::Ref\_ptr< T, CNT >, [808](#)
- cxx::S\_list
  - pop\_front, [816](#)
- cxx::S\_list< T, POLICY >, [814](#)
- cxx::Slab
  - alloc, [819](#)
  - free, [819](#)
- cxx::Slab< Type, Slab\_size, Max\_free, Alloc >, [816](#)
- cxx::Slab\_static
  - alloc, [823](#)
- cxx::Slab\_static< Type, Slab\_size, Max\_free, Alloc >, [820](#)
- cxx::String, [825](#)
- cxx::Weak\_ref< T >, [826](#)
- cxx::Weak\_ref\_base, [829](#)
- cxx::static\_vector< T, IDX >, [823](#)
- d\_val
  - Elf32\_Dyn, [832](#)
  - Elf64\_Dyn, [839](#)
- DF\_1\_CONFALT
  - ELF binary format, [177](#)
- DF\_1\_DIRECT
  - ELF binary format, [177](#)
- DF\_1\_DISPRELDNE
  - ELF binary format, [178](#)
- DF\_1\_DISPRELPND
  - ELF binary format, [178](#)
- DF\_1\_ENDFILTEE
  - ELF binary format, [178](#)
- DF\_1\_GLOBAL
  - ELF binary format, [178](#)
- DF\_1\_GROUP
  - ELF binary format, [178](#)
- DF\_1\_INTERPOSE
  - ELF binary format, [179](#)
- DF\_1\_LOADFLTR
  - ELF binary format, [179](#)
- DF\_1\_NODEFLIB
  - ELF binary format, [179](#)
- DF\_1\_NODELETE
  - ELF binary format, [179](#)
- DF\_1\_NODUMP
  - ELF binary format, [179](#)
- DF\_1\_NOOPEN
  - ELF binary format, [180](#)
- DF\_1\_NOW
  - ELF binary format, [180](#)
- DF\_1\_ORIGIN
  - ELF binary format, [180](#)
- DF\_P1\_GROUPPERM
  - ELF binary format, [180](#)
- DF\_P1\_LAZYLOAD
  - ELF binary format, [180](#)
- DMA API for L4Re, [148](#)
- DMA Space Interface, [149](#)
  - l4re\_dma\_space\_associate, [150](#)
  - l4re\_dma\_space\_disassociate, [150](#)
  - l4re\_dma\_space\_map, [151](#)
  - l4re\_dma\_space\_t, [149](#)
  - l4re\_dma\_space\_unmap, [152](#)
- DT\_HIPROC
  - ELF binary format, [181](#)
- DT\_LOPROC
  - ELF binary format, [181](#)
- DT\_NULL
  - ELF binary format, [181](#)
- data
  - L4::lpc::Varg, [990](#)
- Data\_buffer
  - L4virtio::Svr::Data\_buffer, [1570](#)
- data\_size
  - L4virtio::Svr::Block\_request, [1567](#)
- data\_space
  - L4Re::Vfs::Be\_file, [1402](#)
  - L4Re::Vfs::Regular\_file, [1430](#)
- Dataspace interface, [153](#)
  - l4re\_ds\_allocate, [154](#)
  - l4re\_ds\_clear, [154](#)
  - l4re\_ds\_copy\_in, [155](#)
  - l4re\_ds\_flags, [155](#)
  - l4re\_ds\_info, [155](#)
  - l4re\_ds\_map\_flags, [154](#)
  - l4re\_ds\_phys, [156](#)
  - l4re\_ds\_size, [156](#)
- debug
  - L4Re::Debug\_obj, [1266](#)
- Debug interface, [158](#)
  - l4re\_debug\_obj\_debug, [158](#)
- debugger.h
  - l4\_debugger\_get\_object\_name, [2050](#)
  - l4\_debugger\_query\_log\_name, [2050](#)

- l4\_debugger\_query\_log\_typeid, [2051](#)
- l4\_debugger\_switch\_log, [2052](#)
- Debugging API, [159](#)
- dec\_refcnt
  - L4::Kobject, [1060](#)
- delete\_buffer
  - L4Re::Video::Goos, [1443](#)
- delete\_obj
  - L4::Task, [1135](#)
- delete\_view
  - L4Re::Video::Goos, [1444](#)
- Demand
  - L4::Kobject\_typeid, [1073](#)
  - L4::Type\_info::Demand, [1160](#)
- demand
  - L4::Kobject\_typeid, [1073](#)
- desc
  - L4virtio::Driver::Virtqueue, [1551](#)
  - L4virtio::Svr::Virtqueue, [1619](#)
  - L4virtio::Svr::Virtqueue::Head\_desc, [1622](#)
- desc\_align
  - L4virtio::Virtqueue, [1628](#)
- desc\_avail
  - L4virtio::Svr::Virtqueue, [1619](#)
- desc\_size
  - L4virtio::Virtqueue, [1628](#)
- descs
  - l4\_vhw\_descriptor, [1239](#)
- detach
  - L4::Irq, [1036](#)
  - L4Re::Rm, [1328–1330](#)
  - L4Re::Util::Event\_buffer\_t, [1369](#)
- Detach\_flags
  - L4Re::Rm, [1325](#)
- Detach\_result
  - L4Re::Rm, [1325](#)
- Dev\_config
  - L4virtio::Svr::Dev\_config, [1574](#)
- dev\_handle
  - L4vbus::Device, [1492](#)
- device
  - L4vbus::Device, [1492](#)
- device\_by\_hid
  - L4vbus::Device, [1494](#)
- device\_error
  - L4virtio::Svr::Device\_t, [1595](#)
- di
  - l4\_vcpu\_regs\_t, [1233](#)
- dir
  - cxx::Bits::Bst, [758, 759](#)
- Direction
  - L4Re::Dma\_space, [1268](#)
- Direction\_e
  - cxx::Bits::Direction, [770](#)
- disable
  - L4virtio::Virtqueue, [1629](#)
- disable\_notify
  - L4virtio::Svr::Virtqueue, [1620](#)
- disassociate
  - L4Re::Dma\_space, [1269](#)
- dispatch
  - L4::Basic\_registry, [849](#)
  - L4::Server\_object, [1119](#)
- dispatch\_meta\_request
  - L4::Server\_object\_t, [1122](#)
- dma\_space.h
  - l4re\_dma\_space\_direction, [1861](#)
  - l4re\_dma\_space\_dma\_addr\_t, [1861](#)
  - l4re\_dma\_space\_space\_attribs, [1862](#)
- done
  - L4virtio::Svr::Data\_buffer, [1571](#)
- down
  - L4::Semaphore, [1113](#)
- drive\_cylinders
  - l4util\_mb\_drive\_t, [1481](#)
- drive\_mode
  - l4util\_mb\_drive\_t, [1481](#)
- drive\_number
  - l4util\_mb\_drive\_t, [1482](#)
- driver
  - L4virtio::Svr::Dev\_status, [1589, 1590](#)
- driver\_bfm\_t
  - L4virtio::Svr::Dev\_status, [1588](#)
- Driver\_mem\_region\_t
  - L4virtio::Svr::Driver\_mem\_region\_t, [1606](#)
- driver\_ok
  - L4virtio::Svr::Dev\_status, [1590](#)
- driver\_ok\_bfm\_t
  - L4virtio::Svr::Dev\_status, [1588](#)
- drv\_base
  - L4virtio::Svr::Driver\_mem\_region\_t, [1607](#)
- ds
  - L4virtio::Svr::Dev\_config, [1576](#)
  - L4virtio::Svr::Driver\_mem\_region\_t, [1607](#)
- ds\_offset
  - L4virtio::Svr::Driver\_mem\_region\_t, [1607](#)
- dump
  - L4Re::Video::Color\_component, [1437](#)
  - L4Re::Video::Pixel\_info, [1452](#)
  - L4virtio::Virtqueue, [1629](#)
- dx
  - l4\_vcpu\_regs\_t, [1234](#)
- e\_phnum
  - Elf32\_Ehdr, [834](#)
  - Elf64\_Ehdr, [840](#)
- e\_shnum
  - Elf32\_Ehdr, [834](#)
  - Elf64\_Ehdr, [841](#)
- EDID parsing functionality, [160](#)
  - libedid\_check\_header, [161](#)
  - libedid\_checksum, [161](#)
  - Libedid\_consts, [160](#)
  - libedid\_dump, [161](#)
  - libedid\_dump\_standard\_timings, [162](#)
  - libedid\_num\_ext\_blocks, [162](#)
  - libedid\_pnp\_id, [162](#)

- libedid\_preferred\_resolution, 163
- libedid\_revision, 163
- libedid\_version, 163
- EI\_CLASS
  - ELF binary format, 181
- EI\_DATA
  - ELF binary format, 182
- EI\_OSABI
  - ELF binary format, 182
- EI\_PAD
  - ELF binary format, 183
- EI\_VERSION
  - ELF binary format, 183
- ELF binary format, 165
  - DF\_1\_CONFALT, 177
  - DF\_1\_DIRECT, 177
  - DF\_1\_DISPRELDNE, 178
  - DF\_1\_DISPRELPND, 178
  - DF\_1\_ENDFILTEE, 178
  - DF\_1\_GLOBAL, 178
  - DF\_1\_GROUP, 178
  - DF\_1\_INTERPOSE, 179
  - DF\_1\_LOADFLTR, 179
  - DF\_1\_NODEFLIB, 179
  - DF\_1\_NODELETE, 179
  - DF\_1\_NODUMP, 179
  - DF\_1\_NOOPEN, 180
  - DF\_1\_NOW, 180
  - DF\_1\_ORIGIN, 180
  - DF\_P1\_GROUPEPERM, 180
  - DF\_P1\_LAZYLOAD, 180
  - DT\_HIPROC, 181
  - DT\_LOPROC, 181
  - DT\_NULL, 181
  - EI\_CLASS, 181
  - EI\_DATA, 182
  - EI\_OSABI, 182
  - EI\_PAD, 183
  - EI\_VERSION, 183
  - ELFCLASSNONE, 184
  - ELFDATA2LSB, 184
  - ELFDATA2MSB, 185
  - ELFDATANONE, 185
  - ELFOSABI\_AIX, 186
  - ELFOSABI\_FREEBSD, 186
  - ELFOSABI\_HPUX, 186
  - ELFOSABI\_IRIX, 186
  - ELFOSABI\_LINUX, 187
  - ELFOSABI\_MODESTO, 187
  - ELFOSABI\_NETBSD, 187
  - ELFOSABI\_OPENBSD, 187
  - ELFOSABI\_SOLARIS, 187
  - ELFOSABI\_SYSV, 188
  - ELFOSABI\_TRU64, 188
  - EM\_ARC, 188
  - NT\_VERSION, 188
  - PT\_GNU\_EH\_FRAME, 189
  - PT\_GNU\_RELRO, 189
  - PT\_GNU\_STACK, 189
  - PT\_HIOS, 189
  - PT\_HIPROC, 189
  - PT\_L4\_AUX, 190
  - PT\_L4\_KIP, 190
  - PT\_L4\_STACK, 190
  - PT\_LOOS, 190
  - PT\_LOPROC, 190
  - SHF\_GROUP, 191
  - SHF\_MASKOS, 191
  - SHF\_TLS, 191
  - SHT\_NUM, 191
  - ELFCLASSNONE
    - ELF binary format, 184
  - ELFDATA2LSB
    - ELF binary format, 184
  - ELFDATA2MSB
    - ELF binary format, 185
  - ELFDATANONE
    - ELF binary format, 185
  - ELFOSABI\_AIX
    - ELF binary format, 186
  - ELFOSABI\_FREEBSD
    - ELF binary format, 186
  - ELFOSABI\_HPUX
    - ELF binary format, 186
  - ELFOSABI\_IRIX
    - ELF binary format, 186
  - ELFOSABI\_LINUX
    - ELF binary format, 187
  - ELFOSABI\_MODESTO
    - ELF binary format, 187
  - ELFOSABI\_NETBSD
    - ELF binary format, 187
  - ELFOSABI\_OPENBSD
    - ELF binary format, 187
  - ELFOSABI\_SOLARIS
    - ELF binary format, 187
  - ELFOSABI\_SYSV
    - ELF binary format, 188
  - ELFOSABI\_TRU64
    - ELF binary format, 188
  - EM\_ARC
    - ELF binary format, 188
  - Elf32\_Dyn, 832
    - d\_val, 832
  - Elf32\_Ehdr, 833
    - e\_phnum, 834
    - e\_shnum, 834
  - Elf32\_Phdr, 835
  - Elf32\_Shdr, 836
  - Elf32\_Sym, 837
  - Elf64\_Dyn, 838
    - d\_val, 839
  - Elf64\_Ehdr, 839
    - e\_phnum, 840
    - e\_shnum, 841
  - Elf64\_Phdr, 841

- Elf64\_Shdr, [842](#)
- Elf64\_Sym, [843](#)
- empty
  - L4virtio::Svr::Driver\_mem\_region\_t, [1607](#)
- enable\_notify
  - L4virtio::Svr::Virtqueue, [1620](#)
- end
  - cxx::Bits::Base\_avl\_set, [745](#), [746](#)
  - cxx::Bits::Bst, [760](#)
  - L4::Kip::Mem\_desc, [1053](#)
- enqueue\_descriptor
  - L4virtio::Driver::Virtqueue, [1552](#)
- entry\_ip
  - L4vcpu::Vcpu, [1540](#)
- entry\_sp
  - L4vcpu::Vcpu, [1541](#)
- env
  - L4Re::Env, [1274](#)
- env.h
  - l4re\_env\_t, [1913](#)
- erase
  - cxx::Bits::Base\_avl\_set, [746](#)
  - cxx::H\_list, [774](#)
- err\_no
  - L4::Runtime\_error, [1104](#)
- Error
  - L4virtio::Svr::Bad\_descriptor, [1559](#)
- Error codes, [198](#)
  - l4\_error\_code\_t, [198](#)
- Error Handling, [192](#)
  - l4\_error, [193](#)
  - l4\_ipc\_error, [194](#)
  - l4\_ipc\_error\_code, [195](#)
  - l4\_ipc\_is\_rcv\_error, [196](#)
  - l4\_ipc\_is\_snd\_error, [196](#)
  - l4\_ipc\_tcr\_error\_t, [193](#)
- Event API, [200](#)
- Event interface, [201](#)
  - l4re\_event\_get\_axis\_info, [201](#)
  - l4re\_event\_get\_buffer, [202](#)
  - l4re\_event\_get\_num\_streams, [202](#)
  - l4re\_event\_get\_stream\_info, [203](#)
  - l4re\_event\_get\_stream\_info\_for\_id, [203](#)
- Event\_buffer\_t
  - L4Re::Event\_buffer\_t, [1290](#)
- ex\_regs
  - L4::Thread, [1142](#), [1143](#)
- exc\_handler
  - L4::Thread::Attr, [1151](#)
- Exception registers, [205](#)
  - l4\_utcb\_exc, [205](#)
  - l4\_utcb\_exc\_is\_ex\_regs\_exception, [205](#)
  - l4\_utcb\_exc\_is\_pf, [206](#)
  - l4\_utcb\_exc\_pc, [206](#)
  - l4\_utcb\_exc\_pc\_set, [207](#)
- execute
  - L4Re::Ned::Cmd\_control, [1318](#)
- expired
  - L4::lpc\_svr::Timeout, [1024](#)
- ext\_alloc
  - L4vcpu::Vcpu, [1541](#)
- Extended vCPU support, [208](#)
  - l4vcpu\_ext\_alloc, [208](#)
- extra\_str
  - L4::Runtime\_error, [1104](#)
- faccessat
  - L4Re::Vfs::Directory, [1408](#)
- Factory, [210](#)
  - l4\_factory\_create\_factory, [211](#)
  - l4\_factory\_create\_factory\_u, [212](#)
  - l4\_factory\_create\_gate, [213](#)
  - l4\_factory\_create\_gate\_u, [214](#)
  - l4\_factory\_create\_irq, [215](#)
  - l4\_factory\_create\_irq\_u, [216](#)
  - l4\_factory\_create\_task, [217](#)
  - l4\_factory\_create\_task\_u, [218](#)
  - l4\_factory\_create\_thread, [219](#)
  - l4\_factory\_create\_thread\_u, [220](#)
  - l4\_factory\_create\_vm, [221](#)
  - l4\_factory\_create\_vm\_u, [222](#)
- factory
  - L4Re::Env, [1274](#), [1275](#)
- failed
  - L4virtio::Svr::Dev\_status, [1590](#), [1591](#)
- failed\_bfm\_t
  - L4virtio::Svr::Dev\_status, [1588](#)
- fchmod
  - L4Re::Vfs::Generic\_file, [1421](#)
- fd
  - l4\_vhw\_entry, [1241](#)
- fdatsync
  - L4Re::Vfs::Regular\_file, [1430](#)
- feature\_ok
  - L4virtio::Svr::Dev\_status, [1591](#)
- feature\_ok\_bfm\_t
  - L4virtio::Svr::Dev\_status, [1588](#)
- features
  - l4\_icu\_info\_t, [1214](#)
- Fiasco extensions, [223](#)
  - fiasco\_gdt\_get\_entry\_offset, [225](#)
  - fiasco\_gdt\_set, [225](#)
  - fiasco\_ldt\_set, [226](#)
  - fiasco\_tbuf\_get\_status, [226](#)
  - fiasco\_tbuf\_get\_status\_phys, [226](#)
  - fiasco\_tbuf\_log, [226](#)
  - fiasco\_tbuf\_log\_3val, [227](#)
  - fiasco\_tbuf\_log\_binary, [228](#)
- Fiasco real time scheduling extensions, [229](#)
- Fiasco-UX Virtual devices, [230](#)
  - l4\_vhw\_entry\_type, [230](#)
- fiasco\_amd64\_segment\_info
  - amd64/l4/sys/segment.h, [1705](#)
- fiasco\_amd64\_set\_fs
  - amd64/l4/sys/segment.h, [1706](#)
  - amd64/l4f/l4/sys/segment.h, [1701](#)
- fiasco\_amd64\_set\_segment\_base

- amd64/l4/sys/segment.h, 1706
- amd64/l4f/l4/sys/segment.h, 1702
- fiasco\_gdt\_get\_entry\_offset
  - Fiasco extensions, 225
- fiasco\_gdt\_set
  - Fiasco extensions, 225
- fiasco\_ldt\_set
  - Fiasco extensions, 226
- fiasco\_tbuf\_get\_status
  - Fiasco extensions, 226
- fiasco\_tbuf\_get\_status\_phys
  - Fiasco extensions, 226
- fiasco\_tbuf\_log
  - Fiasco extensions, 226
- fiasco\_tbuf\_log\_3val
  - Fiasco extensions, 227
- fiasco\_tbuf\_log\_binary
  - Fiasco extensions, 228
- finalize\_request
  - L4virtio::Svr::Block\_dev, 1563
- find
  - cxx::Bits::Bst, 761
  - L4::Basic\_registry, 849
  - L4Re::Rm, 1330
  - L4virtio::Svr::Driver\_mem\_list\_t, 1600
- find\_next\_used
  - L4virtio::Driver::Virtqueue, 1552
- find\_node
  - cxx::Bits::Base\_avl\_set, 747
  - cxx::Bits::Bst, 762
- first
  - L4::Kip::Mem\_desc, 1054
- first\_free\_cap
  - L4Re::Env, 1275
- first\_free\_utcb
  - L4Re::Env, 1276
- Flags
  - L4::Types::Flags, 1189
  - L4Re::Video::Goos, 1442
  - L4Re::Video::View, 1457
- flags
  - l4\_exc\_regs\_t, 1211
  - l4\_msgtag\_t, 1218
  - L4Re::Dataspace, 1258
  - l4re\_env\_cap\_entry\_t, 1468
  - L4virtio::Svr::Driver\_mem\_region\_t, 1608
- Flex pages, 232
  - L4\_cap\_fpage\_rights, 234
  - l4\_fpage, 238
  - l4\_fpage\_all, 238
  - l4\_fpage\_cacheability\_opt\_t, 236
  - l4\_fpage\_consts, 236
  - l4\_fpage\_contains, 239
  - l4\_fpage\_invalid, 239
  - l4\_fpage\_ioport, 240
  - l4\_fpage\_max\_order, 240
  - l4\_fpage\_memaddr, 241
  - l4\_fpage\_obj, 242
  - l4\_fpage\_page, 243
  - L4\_fpage\_rights, 237
  - l4\_fpage\_rights, 243
  - l4\_fpage\_set\_rights, 244
  - l4\_fpage\_size, 244
  - l4\_fpage\_type, 245
  - l4\_iofpage, 245
  - l4\_is\_fpage\_writable, 246
  - l4\_obj\_fpage, 246
  - L4\_obj\_fpage\_ctl, 237
- foreach\_available\_event
  - L4Re::Util::Event\_buffer\_consumer\_t, 1365
- fpage
  - L4::Cap\_base, 863
- free
  - cxx::Base\_slab\_static, 705
  - cxx::List\_alloc, 790
  - cxx::Slab, 819
  - L4Re::Cap\_alloc, 1249
  - L4Re::Mem\_alloc, 1305
  - L4Re::Util::Counting\_cap\_alloc, 1349
- free\_area
  - L4Re::Rm, 1331
- free\_capability
  - L4Re::Util::Names::Name\_space, 1378
- free\_descriptor
  - L4virtio::Driver::Virtqueue, 1552
- free\_dynamic\_entry
  - L4Re::Util::Names::Name\_space, 1379
- free\_epiface
  - L4Re::Util::Names::Name\_space, 1379
- free\_fd
  - L4Re::Vfs::Fs, 1418
- free\_objects
  - cxx::Base\_slab, 701
  - cxx::Base\_slab\_static, 706
- from\_ci
  - L4::lpc::Cap, 934
- from\_raw
  - L4::Types::Flags, 1190
- fstat64
  - L4Re::Vfs::Be\_file, 1402
  - L4Re::Vfs::Generic\_file, 1422
- fsync
  - L4Re::Vfs::Regular\_file, 1430
- ftruncate64
  - L4Re::Vfs::Regular\_file, 1430
- full
  - L4virtio::Svr::Driver\_mem\_list\_t, 1600
- Functions for rendering bitmap data in frame buffers, 248
  - gfxbitmap\_bmap, 249
  - gfxbitmap\_color\_pix\_t, 248
  - gfxbitmap\_color\_t, 249
  - gfxbitmap\_convert\_color, 250
  - gfxbitmap\_copy, 250
  - gfxbitmap\_fill, 250
  - gfxbitmap\_set, 251

- Functions for rendering bitmap fonts to frame buffers, [252](#)
  - [gfxbitmap\\_font\\_data](#), [253](#)
  - [gfxbitmap\\_font\\_get](#), [253](#)
  - [gfxbitmap\\_font\\_height](#), [254](#)
  - [gfxbitmap\\_font\\_init](#), [254](#)
  - [gfxbitmap\\_font\\_text](#), [254](#)
  - [gfxbitmap\\_font\\_text\\_scale](#), [255](#)
  - [gfxbitmap\\_font\\_width](#), [255](#)
- Functions to manipulate the local IDT, [257](#)
- g
  - [L4Re::Video::Pixel\\_info](#), [1453](#)
- get
  - [cxx::Auto\\_ptr](#), [681](#)
  - [cxx::Bitfield](#), [710](#)
  - [cxx::Ref\\_ptr](#), [812](#)
  - [L4::lpc::Istream](#), [949](#), [950](#)
  - [L4Re::Env](#), [1276](#)
  - [L4Re::Video::Color\\_component](#), [1437](#)
  - [L4vbus::Gpio\\_module](#), [1502](#)
  - [L4vbus::Gpio\\_pin](#), [1510](#)
  - [L4virtio::Ptr](#), [1556](#)
- get\_attr
  - [L4::Vcon](#), [1200](#)
- get\_avail\_idx
  - [L4virtio::Virtqueue](#), [1630](#)
- get\_buffer
  - [L4Re::Event](#), [1288](#)
- get\_buffer\_demand
  - [L4::Server\\_object\\_t](#), [1123](#)
- get\_cap
  - [L4Re::Env](#), [1277](#), [1278](#)
- get\_cap\_alloc
  - [L4Re::Cap\\_alloc](#), [1250](#)
- get\_cmd
  - [L4virtio::Svr::Dev\\_config](#), [1577](#)
- get\_epiface
  - [L4Re::Util::Names::Name\\_space](#), [1379](#)
- get\_fb
  - [L4Re::Util::Video::Goos\\_svr](#), [1396](#)
- get\_file
  - [L4Re::Vfs::Fs](#), [1418](#)
- get\_infos
  - [L4::lpc\\_gate](#), [1000](#)
- get\_lock
  - [L4Re::Vfs::Regular\\_file](#), [1431](#)
- get\_next\_item
  - [cxx::List\\_item](#), [793](#)
- get\_object\_name
  - [L4::Debugger](#), [874](#)
- get\_prev\_item
  - [cxx::List\\_item](#), [793](#)
- get\_rcv\_cap
  - [L4::lpc\\_svr::Server\\_iface](#), [1017](#)
  - [L4Re::Util::Br\\_manager](#), [1340](#)
- get\_resource
  - [L4vbus::Device](#), [1495](#)
- get\_static\_buffer
  - [L4Re::Video::Goos](#), [1444](#)
- get\_status\_flags
  - [L4Re::Vfs::Generic\\_file](#), [1422](#)
- get\_tail\_avail\_idx
  - [L4virtio::Virtqueue](#), [1630](#)
- get\_unshifted
  - [cxx::Bitfield](#), [711](#)
- get\_value
  - [L4::lpc::Varg](#), [990](#)
- gfxbitmap\_bmap
  - Functions for rendering bitmap data in frame buffers, [249](#)
- gfxbitmap\_color\_pix\_t
  - Functions for rendering bitmap data in frame buffers, [248](#)
- gfxbitmap\_color\_t
  - Functions for rendering bitmap data in frame buffers, [249](#)
- gfxbitmap\_convert\_color
  - Functions for rendering bitmap data in frame buffers, [250](#)
- gfxbitmap\_copy
  - Functions for rendering bitmap data in frame buffers, [250](#)
- gfxbitmap\_fill
  - Functions for rendering bitmap data in frame buffers, [250](#)
- gfxbitmap\_font\_data
  - Functions for rendering bitmap fonts to frame buffers, [253](#)
- gfxbitmap\_font\_get
  - Functions for rendering bitmap fonts to frame buffers, [253](#)
- gfxbitmap\_font\_height
  - Functions for rendering bitmap fonts to frame buffers, [254](#)
- gfxbitmap\_font\_init
  - Functions for rendering bitmap fonts to frame buffers, [254](#)
- gfxbitmap\_font\_text
  - Functions for rendering bitmap fonts to frame buffers, [254](#)
- gfxbitmap\_font\_text\_scale
  - Functions for rendering bitmap fonts to frame buffers, [255](#)
- gfxbitmap\_font\_width
  - Functions for rendering bitmap fonts to frame buffers, [255](#)
- gfxbitmap\_offset, [844](#)
- gfxbitmap\_set
  - Functions for rendering bitmap data in frame buffers, [251](#)
- global\_id
  - [L4::Debugger](#), [874](#)
- gran\_offset
  - [l4\\_sched\\_cpu\\_set\\_t](#), [1220](#)
- granularity
  - [l4\\_sched\\_cpu\\_set\\_t](#), [1219](#)



- H\_list\_item\_t
  - cxx::H\_list\_item\_t, 779
- handle\_expired\_timeouts
  - L4::lpc\_svr::Timeout\_queue, 1026
- handle\_mem\_cmd\_write
  - L4virtio::Svr::Device\_t, 1595
- has\_alpha
  - L4Re::Video::Pixel\_info, 1454
- has\_more
  - L4virtio::Svr::Request\_processor, 1611
- hdr
  - L4virtio::Svr::Dev\_config, 1578
- i
  - L4vcpu::Vcpu, 1542
- IA32 Port I/O API, 258
  - I4util\_in16, 258
  - I4util\_in32, 260
  - I4util\_in8, 260
  - I4util\_ins16, 261
  - I4util\_ins32, 261
  - I4util\_ins8, 262
  - I4util\_out16, 262
  - I4util\_out32, 262
  - I4util\_out8, 263
  - I4util\_outs16, 263
  - I4util\_outs32, 264
  - I4util\_outs8, 264
- IO interface, 265
  - I4io\_device\_types\_t, 266
  - I4io\_has\_resource, 267
  - I4io\_iomem\_flags\_t, 266
  - I4io\_lookup\_device, 268
  - I4io\_lookup\_resource, 268
  - I4io\_release\_iomem, 269
  - I4io\_release\_ioport, 269
  - I4io\_request\_iomem, 269
  - I4io\_request\_iomem\_region, 270
  - I4io\_request\_ioport, 271
  - I4io\_request\_resource\_iomem, 271
  - I4io\_resource\_t, 266
  - I4io\_resource\_types\_t, 267
  - I4io\_search\_iomem\_region, 272
- IPC-Gate API, 273
  - I4\_ipc\_gate\_bind\_thread, 274
  - I4\_ipc\_gate\_get\_infos, 274
- IRQ handling library, 276
- IRQs, 277
  - I4\_irq\_attach, 279
  - I4\_irq\_attach\_u, 279
  - I4\_irq\_detach, 280
  - I4\_irq\_detach\_u, 281
  - L4\_irq\_mode, 278
  - I4\_irq\_mux\_chain, 282
  - I4\_irq\_mux\_chain\_u, 283
  - I4\_irq\_receive, 284
  - I4\_irq\_receive\_u, 285
  - I4\_irq\_trigger, 286
  - I4\_irq\_trigger\_u, 287
  - I4\_irq\_unmask, 288
  - I4\_irq\_unmask\_u, 288
  - I4\_irq\_wait, 289
  - I4\_irq\_wait\_u, 290
- id
  - L4::Kobject\_typeid, 1073
- id\_received
  - L4::lpc::Gen\_fpage, 936
- idle\_time
  - L4::Scheduler, 1107
- indirect
  - L4virtio::Virtqueue::Desc::Flags, 1644
- indirect\_bfm\_t
  - L4virtio::Virtqueue::Desc::Flags, 1643
- info
  - L4::lcu, 905
  - L4::Scheduler, 1108
  - L4Re::Dataspace, 1260
  - L4Re::Video::Goos, 1445
  - L4Re::Video::View, 1458
- Info\_sub\_type
  - L4::Kip::Mem\_desc, 1050
- init
  - L4Re::Util::Event\_t, 1371
  - L4virtio::Svr::Driver\_mem\_list\_t, 1601
- init\_infos
  - L4Re::Util::Video::Goos\_svr, 1396
- init\_mem\_info
  - L4virtio::Svr::Device\_t, 1595
- init\_poll
  - L4Re::Util::Event\_t, 1373
- init\_queue
  - L4virtio::Driver::Virtqueue, 1553
- Initial Environment, 292
  - I4re\_env, 293
  - I4re\_env\_get\_cap, 293
  - I4re\_env\_get\_cap\_e, 294
  - I4re\_env\_get\_cap\_l, 295
  - I4re\_kip, 296
- initial\_caps
  - L4Re::Env, 1278
- initialize\_rings
  - L4virtio::Driver::Virtqueue, 1554
- insert
  - cxx::Avl\_map, 687
  - cxx::Avl\_tree, 695
  - cxx::Bits::Base\_avl\_set, 747
  - cxx::H\_list, 775
- insert\_after
  - cxx::H\_list, 775
- insert\_before
  - cxx::H\_list, 776
- insert\_next\_item
  - cxx::List\_item, 793
- insert\_prev\_item
  - cxx::List\_item, 794
- Integer Types, 297
  - I4\_int16\_t, 298



- [l4\\_int32\\_t](#), [298](#)
- [l4\\_int64\\_t](#), [299](#)
- [l4\\_int8\\_t](#), [299](#)
- [l4\\_uint16\\_t](#), [299](#)
- [l4\\_uint32\\_t](#), [299](#)
- [l4\\_uint64\\_t](#), [299](#)
- [l4\\_uint8\\_t](#), [299](#)
- interface
  - [L4::Meta](#), [1080](#)
- Interface for asynchronous ISR handlers with a given L4↔RQ capability., [300](#)
- [l4irq\\_request\\_cap](#), [300](#)
- Interface for asynchronous ISR handlers., [302](#)
- [l4irq\\_release](#), [302](#)
- [l4irq\\_request](#), [303](#)
- Interface using direct functionality., [304](#), [310](#)
- [l4irq\\_attach](#), [304](#)
- [l4irq\\_attach\\_cap](#), [310](#)
- [l4irq\\_attach\\_cap\\_ft](#), [310](#)
- [l4irq\\_attach\\_ft](#), [305](#)
- [l4irq\\_attach\\_thread](#), [305](#)
- [l4irq\\_attach\\_thread\\_cap](#), [311](#)
- [l4irq\\_attach\\_thread\\_cap\\_ft](#), [311](#)
- [l4irq\\_attach\\_thread\\_ft](#), [306](#)
- [l4irq\\_detach](#), [306](#)
- [l4irq\\_unmask](#), [307](#)
- [l4irq\\_unmask\\_and\\_wait\\_any](#), [307](#)
- [l4irq\\_wait](#), [307](#)
- [l4irq\\_wait\\_any](#), [309](#)
- Internal constants, [314](#)
- Internal functions, [316](#)
- Internal Helpers, [313](#)
- internal\_loop
  - [L4::Server](#), [1117](#)
- Interrupt controller, [317](#)
- [l4\\_icu\\_bind](#), [319](#)
- [l4\\_icu\\_bind\\_u](#), [320](#)
- [L4\\_icu\\_flags](#), [319](#)
- [l4\\_icu\\_info](#), [321](#)
- [l4\\_icu\\_info\\_t](#), [318](#)
- [l4\\_icu\\_info\\_u](#), [321](#)
- [l4\\_icu\\_mask](#), [322](#)
- [l4\\_icu\\_mask\\_u](#), [323](#)
- [l4\\_icu\\_msi\\_info](#), [324](#)
- [l4\\_icu\\_msi\\_info\\_u](#), [325](#)
- [l4\\_icu\\_set\\_mode](#), [325](#)
- [l4\\_icu\\_set\\_mode\\_u](#), [326](#)
- [l4\\_icu\\_unbind](#), [327](#)
- [l4\\_icu\\_unbind\\_u](#), [328](#)
- [l4\\_icu\\_unmask](#), [329](#)
- [l4\\_icu\\_unmask\\_u](#), [329](#)
- Invalid\_capability
  - [L4::Invalid\\_capability](#), [913](#)
- Invalid\_type
  - [L4virtio::Ptr](#), [1556](#)
- io.h
  - [l4io\\_get\\_root\\_device](#), [1759](#)
  - [l4io\\_iterate\\_devices](#), [1759](#)
  - [l4io\\_request\\_all\\_ioports](#), [1760](#)
  - [l4io\\_request\\_icu](#), [1760](#)
- io\_page\_fault
  - [L4::io\\_pager](#), [915](#)
- ioctl
  - [L4Re::Vfs::Special\\_file](#), [1435](#)
- loststream
  - [L4::lpc::loststream](#), [941](#)
- ipc\_client
  - [L4\\_RPC\\_DEF](#), [2020](#)
- ipc\_iface
  - [L4\\_INLINE\\_RPC\\_NF\\_OP](#), [2024](#)
  - [L4\\_INLINE\\_RPC\\_NF](#), [2024](#)
  - [L4\\_INLINE\\_RPC\\_OP](#), [2025](#)
  - [L4\\_INLINE\\_RPC](#), [2023](#)
  - [L4\\_RPC\\_NF\\_OP](#), [2026](#)
  - [L4\\_RPC\\_NF](#), [2026](#)
  - [L4\\_RPC\\_OP](#), [2027](#)
  - [L4\\_RPC](#), [2025](#)
- ipc\_stream
  - operator<<, [1806–1808](#)
  - operator>>, [1809–1813](#)
- irq
  - [L4Re::Util::Event\\_t](#), [1373](#)
- irq\_disable\_save
  - [L4vcpu::Vcpu](#), [1542](#)
- irq\_enable
  - [L4vbus::Pci\\_dev](#), [1519](#)
  - [L4vbus::Pci\\_host\\_bridge](#), [1525](#)
  - [L4vcpu::Vcpu](#), [1543](#)
- irq\_no
  - [l4\\_vhw\\_entry](#), [1241](#)
- irq\_restore
  - [L4vcpu::Vcpu](#), [1544](#)
- is\_compatible
  - [L4vbus::Device](#), [1495](#)
- is\_compound
  - [L4::lpc::Gen\\_fpage](#), [937](#)
- is\_irq\_entry
  - [L4vcpu::Vcpu](#), [1544](#)
- is\_nil
  - [L4::lpc::Varg](#), [991](#)
- is\_of
  - [L4::lpc::Varg](#), [991](#)
- is\_of\_int
  - [L4::lpc::Varg](#), [991](#)
- is\_online
  - [L4::Scheduler](#), [1108](#)
- is\_page\_fault\_entry
  - [L4vcpu::Vcpu](#), [1545](#)
- is\_static
  - [L4Re::Util::Dataspace\\_svr](#), [1356](#)
- is\_valid
  - [L4::Cap\\_base](#), [864](#)
  - [L4virtio::Ptr](#), [1557](#)
- is\_virtual
  - [L4::Kip::Mem\\_desc](#), [1055](#)
- is\_writable

- L4virtio::Svr::Driver\_mem\_region\_t, 1608
- Istream
  - L4::lpc::Istream, 948
- items
  - cxx::List, 784
- iter
  - cxx::Bits::Basic\_list, 753
  - cxx::H\_list, 776
- Kernel Debugger, 331
  - l4\_debugger\_global\_id, 331
  - l4\_debugger\_kobj\_to\_id, 332
  - l4\_debugger\_set\_object\_name, 332
- Kernel Interface Page, 334
  - l4\_kernel\_info\_version\_offset, 335
  - l4\_kip\_clock, 336
  - l4\_kip\_clock\_lw, 336
  - l4\_kip\_version, 337
  - l4\_kip\_version\_string, 337
- Kernel Interface Page API, 339
  - l4util\_kip\_for\_each\_feature, 339
  - l4util\_kip\_kernel\_abi\_version, 340
  - l4util\_kip\_kernel\_has\_feature, 340
  - l4util\_kip\_kernel\_is\_ux, 340
  - l4util\_memdesc\_vm\_high, 340
- Kernel Objects, 342
- kobj\_to\_id
  - L4::Debugger, 875
- kobject\_typeid
  - L4 kernel object type information, 373
- Kumem allocator utility, 344
  - l4re\_util\_kumem\_alloc, 344
- Kumem utilities, 346
  - kumem\_alloc, 346
- kumem\_alloc
  - Kumem utilities, 346
- L4, 643
  - cap\_cast, 647, 648
  - cap\_dynamic\_cast, 649
  - cap\_reinterpret\_cast, 650
  - throw\_ipc\_exception, 651, 653
- L4 IPC Opcodes, 348
  - L4\_icu\_opcode, 348
  - L4\_ipc\_gate\_ops, 349
  - L4\_platform\_ctl\_ops, 350
  - L4\_task\_ops, 350
  - L4\_thread\_ops, 350
  - L4\_vcon\_ops, 351
- L4 kernel object type information, 373
  - kobject\_typeid, 373
- L4 V-BUS functions, 352
  - l4vbus\_assign\_dma\_domain, 353
  - l4vbus\_dma\_domain\_assign\_flags, 353
  - l4vbus\_get\_device, 354
  - l4vbus\_get\_device\_by\_hid, 355
  - l4vbus\_get\_hid, 356
  - l4vbus\_get\_next\_device, 356
  - l4vbus\_get\_resource, 357
  - l4vbus\_is\_compatible, 357
  - l4vbus\_release\_resource, 358
  - l4vbus\_request\_resource, 359
  - l4vbus\_vicu\_get\_cap, 360
- L4 VIRTIO Block Device, 361
  - L4virtio\_block\_operations, 361
  - L4virtio\_block\_status, 362
- L4 VIRTIO Interface, 363
- L4 VIRTIO Transport Layer, 364
  - L4\_virtio\_cmd, 366
  - L4\_virtio\_irq\_status, 366
  - L4\_virtio\_opcodes, 367
  - l4virtio\_config\_queue, 368
  - l4virtio\_config\_queue\_t, 365
  - l4virtio\_config\_queues, 369
  - l4virtio\_device\_config, 369
  - L4virtio\_device\_ids, 367
  - L4virtio\_device\_status, 368
  - L4virtio\_feature\_bits, 368
  - l4virtio\_register\_ds, 370
  - l4virtio\_register\_iface, 371
  - l4virtio\_set\_status, 371
- l4/cxx/alloc.h, 2172
- l4/cxx/atomic.h, 2186, 2187
- l4/cxx/avl\_map, 1767, 1768
- l4/cxx/avl\_set, 1770, 1771
- l4/cxx/avl\_tree, 1774, 1776
- l4/cxx/basic\_ostream, 1780, 1781
- l4/cxx/basic\_vector.h, 1784, 1785
- l4/cxx/bits/bst.h, 1785, 1787
- l4/cxx/bits/bst\_base.h, 1789, 1791
- l4/cxx/bits/bst\_iter.h, 1792, 1794
- l4/cxx/exceptions, 1795, 1797
- l4/cxx/iostream, 1800, 1801
- l4/cxx/ipc\_helper, 1801, 1803
- l4/cxx/ipc\_server, 2033, 2034
- l4/cxx/ipc\_stream, 1803, 1814
- l4/cxx/l4iostream, 1824, 1825
- l4/cxx/l4types.h, 1826, 1827
- l4/cxx/main\_thread, 1827, 1828
- l4/cxx/pair, 1829, 1830
- l4/cxx/std\_exc\_io, 1831, 1832
- l4/cxx/string.h, 1832, 1834
- l4/cxx/thread, 2141, 2142
- l4/irq/irq.h, 1834, 1836
- l4/libedid/edid.h, 1848, 1849
- l4/libgfxbitmap/bitmap.h, 1849, 1852
- l4/libgfxbitmap/font.h, 1853, 1854
- l4/libgfxbitmap/support, 1855, 1856
- l4/re/c/dataspace.h, 1856, 1858
- l4/re/c/debug.h, 1859
- l4/re/c/dma\_space.h, 1860, 1862
- l4/re/c/event.h, 1863, 1865
- l4/re/c/log.h, 1867, 1868
- l4/re/c/mem\_alloc.h, 1869, 1871
- l4/re/c/namespace.h, 1872, 1873
- l4/re/c/rm.h, 1874, 1875
- l4/re/c/util/cap\_alloc.h, 1878, 1879

l4/re/c/util/kumem\_alloc.h, 1879, 1880  
l4/re/c/util/video/goos\_fb.h, 1881, 1882  
l4/re/c/video/colors.h, 1882, 1884  
l4/re/c/video/goos.h, 1885, 1887  
l4/re/c/video/view.h, 1888, 1890  
l4/re/cap\_alloc, 1891, 1892  
l4/re/consts, 1897, 1898  
l4/re/consts.h, 2017, 2018  
l4/re/dataspace, 1898, 1900  
l4/re/dataspace-sys.h, 1901, 1902  
l4/re/debug, 1902, 1904  
l4/re/dma\_space, 1904, 1905  
l4/re/elf\_aux.h, 1906, 1908  
l4/re/env, 1909, 1910  
l4/re/env.h, 1911, 1913  
l4/re/error\_helper, 1915, 1917  
l4/re/event.h, 1865, 1866  
l4/re/impl/dataspace\_impl.h, 1921, 1922  
l4/re/impl/mem\_alloc\_impl.h, 1923, 1924  
l4/re/impl/namespace\_impl.h, 1925, 1926  
l4/re/impl/rm\_impl.h, 1927, 1928  
l4/re/l4aux.h, 1930, 1931  
l4/re/log, 1931, 1933  
l4/re/log-sys.h, 1933, 1934  
l4/re/mem\_alloc, 1934, 1936  
l4/re/mem\_alloc-sys.h, 1936, 1937  
l4/re/namespace, 1938, 1939  
l4/re/namespace-sys.h, 1940, 1941  
l4/re/parent, 1941, 1943  
l4/re/parent-sys.h, 1943, 1944  
l4/re/rm, 1944, 1946  
l4/re/rm-sys.h, 1949, 1950  
l4/re/util/bitmap\_cap\_alloc, 1950, 1952  
l4/re/util/cap, 1953, 1954  
l4/re/util/cap\_alloc, 1893, 1895  
l4/re/util/cap\_alloc\_impl.h, 1954, 1956  
l4/re/util/counting\_cap\_alloc, 1956, 1958  
l4/re/util/event, 1918, 1919  
l4/re/util/item\_alloc, 1960, 1961  
l4/re/util/kumem\_alloc, 1962, 1963  
l4/re/util/region\_mapping, 1963, 1965  
l4/re/video/goos-sys.h, 1969, 1970  
l4/shmc/shmc.h, 1971, 1973  
l4/sigma/sigma0.h, 1975, 1978  
l4/sys/\_\_kernel\_object\_impl.h, 1979, 1980  
l4/sys/\_\_ktrace-impl.h, 1980, 1982  
l4/sys/\_\_typeinfo.h, 1983, 1986  
l4/sys/assert.h, 2175, 2177  
l4/sys/cache.h, 1996, 1997  
l4/sys/capability, 2003, 2005  
l4/sys/compiler.h, 2007, 2008  
l4/sys/consts.h, 2010, 2012  
l4/sys/cxx/ipc\_client, 2018, 2020  
l4/sys/cxx/ipc\_iface, 2021, 2028  
l4/sys/cxx/ipc\_types, 2035, 2037  
l4/sys/cxx/types, 2044, 2045  
l4/sys/debugger, 2046, 2048  
l4/sys/debugger.h, 2048, 2052  
l4/sys/err.h, 2055, 2057  
l4/sys/factory, 2057, 2059  
l4/sys/factory.h, 2061, 2062  
l4/sys/icu, 2067, 2068  
l4/sys/icu.h, 2068, 2071  
l4/sys/ipc.h, 2075, 2077  
    l4\_ipc\_to\_errno, 2077  
l4/sys/ipc\_gate, 2083, 2085  
l4/sys/ipc\_gate.h, 2085, 2087  
l4/sys/irq, 2088, 2090  
l4/sys/irq.h, 1843, 1845  
l4/sys/kernel\_object.h, 2092, 2093  
l4/sys/kip, 2094, 2096  
l4/sys/kip.h, 2223, 2224  
l4/sys/ktrace.h, 2097, 2099  
l4/sys/l4int.h, 2100, 2101  
l4/sys/memdesc.h, 2104, 2106  
l4/sys/meta, 2108, 2110  
l4/sys/pager, 2110, 2112  
l4/sys/platform\_control, 2112, 2114  
l4/sys/platform\_control.h, 2114, 2116  
l4/sys/scheduler, 2118, 2119  
l4/sys/scheduler.h, 2120, 2121  
l4/sys/semaphore, 2124, 2126  
l4/sys/smart\_capability, 2126, 2128  
l4/sys/task, 2130, 2132  
l4/sys/task.h, 2133, 2134  
l4/sys/thread, 2138, 2139  
l4/sys/thread.h, 2254, 2256  
l4/sys/typeinfo\_svr, 2143, 2145  
l4/sys/types.h, 1761, 1764  
    l4\_capability\_next, 1764  
l4/sys/utcb.h, 2145, 2147  
l4/sys/vcon, 2158, 2160  
l4/sys/vcon.h, 2160, 2163  
l4/sys/vhw.h, 2166, 2168  
l4/sys/vm, 2169, 2170  
l4/util/alloc.h, 2170, 2171  
l4/util/assert.h, 2173, 2174  
l4/util/atomic.h, 2178, 2181  
l4/util/backtrace.h, 2188, 2189  
l4/util/base64.h, 2189, 2191  
l4/util/bitops.h, 2191, 2193  
l4/util/elf.h, 2196, 2209  
l4/util/getopt.h, 2219  
l4/util/keymap.h, 2220, 2221  
l4/util/kip.h, 2221, 2222  
l4/util/kprintf.h, 2226, 2227  
l4/util/l4\_macros.h, 1747, 1748  
l4/util/list\_alloc.h, 2227, 2230  
l4/util/lock.h, 2230, 2231  
l4/util/mb\_info.h, 2232, 2235  
l4/util/parse\_cmd.h, 2239, 2240  
l4/util/rand.h, 2240, 2242  
l4/util/reboot.h, 2242, 2243  
l4/util/sll.h, 2243, 2244  
l4/util/splitlog2.h, 2247, 2248  
l4/util/stack.h, 2249, 2251

- l4/util/thread.h, 2251, 2253
  - \_\_L4UTIL\_THREAD\_FUNC, 2253
- l4/vbus/vbus.h, 2265, 2267
- l4/vbus/vbus\_types.h, 2268, 2269
- L4::Alloc\_list, 845
- L4::Base\_exception, 846
- L4::Basic\_registry, 847
  - dispatch, 849
  - find, 849
- L4::Bounds\_error, 850
- L4::Cap
  - Cap, 855, 856
  - copy, 857
  - move, 857
- L4::Cap< T >, 853
- L4::Cap\_base, 858
  - \_c, 868
  - cap, 861
  - Cap\_base, 860
  - Cap\_type, 860
  - copy, 862
  - fpage, 863
  - is\_valid, 864
  - move, 865
  - No\_init\_type, 860
  - snd\_base, 866
  - validate, 867
- L4::Com\_error, 869
  - Com\_error, 871
- L4::Debugger, 871
  - get\_object\_name, 874
  - global\_id, 874
  - kobj\_to\_id, 875
  - query\_log\_name, 875
  - query\_log\_typeid, 876
  - set\_object\_name, 876
  - switch\_log, 877
- L4::Element\_already\_exists, 878
- L4::Element\_not\_found, 880
- L4::Exception\_tracer, 882
- L4::Factory, 883
  - create, 887, 888
  - create\_factory, 889
  - create\_gate, 890
  - create\_irq, 891
  - create\_task, 892
  - create\_thread, 893
  - create\_vm, 894
- L4::Factory::Lstr, 895
  - Lstr, 896
- L4::Factory::Nil, 896
- L4::Factory::S, 897
  - operator l4\_msgtag\_t, 899
  - operator<<, 899–901
  - S, 898
- L4::IOModifier, 918
- L4::lcu, 901
  - bind, 904
  - info, 905
  - mask, 906
  - msi\_info, 907
  - set\_mode, 907
  - unbind, 908
- L4::lcu::Info, 909
- L4::Invalid\_capability, 910
  - cap, 913
  - Invalid\_capability, 913
- L4::lo\_pager, 914
  - io\_page\_fault, 915
- L4::lommu, 916
  - bind, 918
  - unbind, 918
- L4::lpc, 654
  - buf\_cp\_in, 656
  - buf\_cp\_out, 657
  - buf\_in, 657
  - make\_cap, 658
  - make\_cap\_full, 659
  - make\_cap\_rw, 659
  - make\_cap\_rws, 660
  - msg\_ptr, 661
  - read, 661
  - str\_cp\_in, 662
- L4::lpc::Array< ELEM\_TYPE, LEN\_TYPE >, 919
- L4::lpc::Array\_in\_buf< ELEM\_TYPE, LEN\_TYPE, MAX >, 922
- L4::lpc::Array\_ref< ELEM\_TYPE, LEN\_TYPE >, 923
- L4::lpc::As\_value< T >, 925
- L4::lpc::Buf\_item, 926
- L4::lpc::Call, 927
- L4::lpc::Call\_t< RIGHTS >, 928
- L4::lpc::Call\_zero\_send\_timeout, 930
- L4::lpc::Cap
  - Cap, 933
  - from\_ci, 934
- L4::lpc::Cap< T >, 931
- L4::lpc::Gen\_fpage
  - cap\_received, 936
  - id\_received, 936
  - is\_compound, 937
  - local\_id\_received, 937
- L4::lpc::Gen\_fpage< T >, 934
- L4::lpc::In\_out< T >, 938
- L4::lpc::Istream, 938
  - call, 942
  - Istream, 941
  - reply\_and\_wait, 943
  - reset, 944
- L4::lpc::Istream, 945
  - get, 949, 950
  - Istream, 948
  - receive, 951
  - reset, 952
  - skip, 952
  - tag, 953
  - wait, 954

- L4::lpc::Msg, 662
  - align\_to, 664, 665
  - check\_size, 666
  - msg\_add, 667
  - msg\_get, 668
- L4::lpc::Msg::CInt\_val\_ops< MTYPE, DIR, CLASS >, 955
- L4::lpc::Msg::Cls\_buffer, 956
- L4::lpc::Msg::Cls\_data, 958
- L4::lpc::Msg::Cls\_item, 959
- L4::lpc::Msg::Dir\_in, 960
- L4::lpc::Msg::Dir\_out, 961
- L4::lpc::Msg::Do\_in\_data, 962
- L4::lpc::Msg::Do\_in\_items, 963
- L4::lpc::Msg::Do\_out\_data, 964
- L4::lpc::Msg::Do\_out\_items, 965
- L4::lpc::Msg::Do\_rcv\_buffers, 966
- L4::lpc::Msg::Elem< Array< A, LEN > >, 969
- L4::lpc::Msg::Elem< Array< A, LEN > &>, 968
- L4::lpc::Msg::Elem< Array\_ref< A, LEN > &>, 970
- L4::lpc::Msg::Is\_valid\_rpc\_type< T >, 971
- L4::lpc::Msg::Svr\_arg\_pack< IPC\_TYPE >, 973
- L4::lpc::Msg::Svr\_val\_ops< MTYPE, DIR, CLASS >, 973
- L4::lpc::Msg\_ptr
  - Msg\_ptr, 975
- L4::lpc::Msg\_ptr< T >, 974
- L4::lpc::Opt< T >, 975
- L4::lpc::Ostream, 977
  - put, 980, 981
  - send, 981
  - tag, 982
- L4::lpc::Out< T >, 983
- L4::lpc::Ret\_array< T >, 984
- L4::lpc::Send\_only, 985
- L4::lpc::Small\_buf, 985
  - Small\_buf, 986
- L4::lpc::Snd\_item, 987
- L4::lpc::Str\_cp\_in
  - Str\_cp\_in, 988
- L4::lpc::Str\_cp\_in< T >, 987
- L4::lpc::Varg, 989
  - data, 990
  - get\_value, 990
  - is\_nil, 991
  - is\_of, 991
  - is\_of\_int, 991
  - length, 991
  - tag, 992
  - type, 993
  - value, 993
- L4::lpc::Varg\_list< MAX >, 994
- L4::lpc::Varg\_list\_ref, 995
  - Varg\_list\_ref, 996
- L4::lpc\_gate, 997
  - bind\_thread, 999
  - get\_infos, 1000
- L4::lpc\_svr, 668
  - L4::lpc\_svr::Br\_manager\_no\_buffers, 1000
    - alloc\_buffer\_demand, 1003
  - L4::lpc\_svr::Compound\_reply, 1004
  - L4::lpc\_svr::Default\_loop\_hooks, 1005
  - L4::lpc\_svr::Default\_setup\_wait, 1006
  - L4::lpc\_svr::Default\_timeout, 1007
  - L4::lpc\_svr::Direct\_dispatch< R >, 1008
  - L4::lpc\_svr::Direct\_dispatch< R \* >, 1010
  - L4::lpc\_svr::Exc\_dispatch< R, Exc >, 1011
  - L4::lpc\_svr::Ignore\_errors, 1012
  - L4::lpc\_svr::Server\_iface, 1013
    - add\_timeout, 1016
    - alloc\_buffer\_demand, 1016
    - get\_rcv\_cap, 1017
    - rcv\_cap, 1018, 1019
    - realloc\_rcv\_cap, 1020
    - remove\_timeout, 1021
  - L4::lpc\_svr::Timed\_work< HOOKS >, 1022
  - L4::lpc\_svr::Timeout, 1022
    - expired, 1024
    - timeout, 1024
  - L4::lpc\_svr::Timeout\_queue, 1025
    - add, 1026
    - handle\_expired\_timeouts, 1026
    - next\_timeout, 1027
    - remove, 1027
    - timeout\_expired, 1028
  - L4::lpc\_svr::Timeout\_queue\_hooks
    - add\_timeout, 1031
    - remove\_timeout, 1031
  - L4::lpc\_svr::Timeout\_queue\_hooks< HOOKS, BR\_M↵AN >, 1029
  - L4::lpc::Irq, 1032
    - attach, 1035
    - detach, 1036
    - receive, 1037
    - unmask, 1038
    - wait, 1039
  - L4::lpc::Irq\_eoi, 1040
    - unmask, 1041
  - L4::lpc::Irq\_handler\_object, 1042
  - L4::lpc::Irq\_mux, 1045
    - chain, 1047
  - L4::lpc::Kip::Mem\_desc, 1048
    - all, 1051, 1052
    - count, 1052, 1053
    - end, 1053
    - first, 1054
    - Info\_sub\_type, 1050
    - is\_virtual, 1055
    - Mem\_desc, 1050
    - Mem\_type, 1050
    - set, 1055
    - size, 1055
    - start, 1056
    - sub\_type, 1056
    - type, 1057
  - L4::lpc::Kobject, 1057

- cap, 1058
- dec\_refcnt, 1060
- L4::Kobject\_2t
  - \_\_lface, 1064
  - \_\_lface\_list, 1064
  - \_\_check\_protocols\_\_, 1065
  - c, 1065
  - Class, 1064
- L4::Kobject\_2t< Derived, Base1, Base2, PROTO, S\_↔  
DEMAND >, 1061
- L4::Kobject\_3t
  - \_\_lface, 1068
  - \_\_lface\_list, 1068
  - \_\_check\_protocols\_\_, 1069
  - c, 1069
  - Class, 1068
- L4::Kobject\_3t< Derived, Base1, Base2, Base3, PROTO, S\_↔  
TO, S\_DEMAND >, 1066
- L4::Kobject\_demand< T >, 1069
- L4::Kobject\_t< Derived, Base, PROTO, S\_DEMAND >, 1070
- L4::Kobject\_typeid
  - Demand, 1073
  - demand, 1073
  - id, 1073
  - proto\_dispatch, 1074
- L4::Kobject\_typeid< T >, 1072
- L4::Kobject\_x< Derived, ARGS >, 1075
- L4::Meta, 1077
  - interface, 1080
  - num\_interfaces, 1080
  - supports, 1080
- L4::Out\_of\_memory, 1081
- L4::Pager, 1084
  - page\_fault, 1086
- L4::Platform\_control, 1087
  - cpu\_disable, 1090
  - cpu\_enable, 1091
  - Opcode, 1090
  - system\_shutdown, 1091
  - system\_suspend, 1091
- L4::Poll\_timeout\_kipclock, 1092
  - Poll\_timeout\_kipclock, 1093
  - set, 1093
  - test, 1094
  - timed\_out, 1095
- L4::Proto\_t< P >, 1095
- L4::Registry\_iface, 1097
  - register\_irq\_obj, 1098, 1099
  - register\_obj, 1100
  - unregister\_obj, 1101
- L4::Runtime\_error, 1101
  - err\_no, 1104
  - extra\_str, 1104
  - Runtime\_error, 1104
- L4::Scheduler, 1105
  - idle\_time, 1107
  - info, 1108
  - is\_online, 1108
  - run\_thread, 1109
- L4::Semaphore, 1110
  - down, 1113
  - up, 1114
- L4::Server
  - internal\_loop, 1117
  - loop, 1117
  - Server, 1116
- L4::Server< LOOP\_HOOKS >, 1115
- L4::Server\_object, 1118
  - dispatch, 1119
- L4::Server\_object\_t
  - dispatch\_meta\_request, 1122
  - get\_buffer\_demand, 1123
  - proto\_dispatch, 1123
- L4::Server\_object\_t< IFACE, BASE >, 1120
- L4::Server\_object\_x< Derived, IFACE, BASE >, 1124
- L4::Smart\_cap
  - Smart\_cap, 1129
- L4::Smart\_cap< T, SMART >, 1126
- L4::String, 1130
- L4::Task, 1130
  - add\_ku\_mem, 1133
  - cap\_equal, 1134
  - cap\_has\_child, 1134
  - cap\_valid, 1135
  - delete\_obj, 1135
  - map, 1136
  - release\_cap, 1136
  - unmap, 1137
  - unmap\_batch, 1137
- L4::Thread, 1138
  - control, 1142
  - ex\_regs, 1142, 1143
  - modify\_senders, 1144
  - register\_del\_irq, 1145
  - stats\_time, 1145
  - switch\_to, 1146
  - vcpu\_control, 1146
  - vcpu\_control\_ext, 1147
  - vcpu\_resume\_commit, 1148
  - vcpu\_resume\_start, 1148
- L4::Thread::Attr, 1149
  - Attr, 1150
  - bind, 1150
  - exc\_handler, 1151
  - pager, 1151, 1152
  - ux\_host\_syscall, 1152
- L4::Thread::Modify\_senders, 1152
  - add, 1153
- L4::Triggerable, 1154
  - trigger, 1156
- L4::Type\_info, 1158
- L4::Type\_info::Demand, 1159
  - Demand, 1160
  - no\_demand, 1161

- L4::Type\_info::Demand\_t< CAPS, FLAGS, MEM, PO↵  
RTS >, [1162](#)
- L4::Type\_info::Demand\_union\_t< D1, D2 >, [1164](#)
- L4::Typeid, [669](#)
- L4::Typeid::Detail::\_Rpc< OPCODE, O, Default\_op<  
R > >::Rpc< Y >, [1168](#)
- L4::Typeid::Detail::\_Rpc< OPCODE, O, R, X... >,  
[1169](#)
- L4::Typeid::Detail::\_Rpc< OPCODE, O, R, X... >::↵  
Rpc< Y >, [1170](#)
- L4::Typeid::Detail::\_Rpc< OPCODE, O, X >, [1167](#)
- L4::Typeid::Detail::Rpc\_end, [1171](#)
- L4::Typeid::P\_dispatch< LIST >, [1172](#)
- L4::Typeid::Raw\_ipc< CLASS >, [1173](#)
- L4::Typeid::Rpc\_nocode< OPERATION >, [1173](#)
- L4::Typeid::Rpc< RPCS >, [1176](#)
- L4::Typeid::Rpc\_code< OPCODE\_TYPE >, [1178](#)
- L4::Typeid::Rpc\_code< OPCODE\_TYPE >::F< RP↵  
CS >, [1179](#)
- L4::Typeid::Rpc\_sys< ARG >, [1181](#)
- L4::Types, [670](#)
- L4::Types::Bool< V >, [1183](#)
- L4::Types::False, [1184](#)
- L4::Types::Flags
  - clear, [1190](#)
  - Flags, [1189](#)
  - from\_raw, [1190](#)
  - None\_type, [1189](#)
- L4::Types::Flags< BITS\_ENUM, UNDERLYING >,  
[1186](#)
- L4::Types::Same< A, B >, [1191](#)
- L4::Types::True, [1193](#)
- L4::Unknown\_error, [1195](#)
- L4::Vcon, [1198](#)
  - get\_attr, [1200](#)
  - read, [1201](#)
  - read\_with\_flags, [1202](#)
  - send, [1202](#)
  - set\_attr, [1203](#)
  - write, [1204](#)
- L4::Vm, [1205](#)
- L4\_ALWAYS\_INLINE
  - Basic Macros, [105](#)
- L4\_DISABLE\_COPY
  - capability, [2005](#)
- L4\_HIDDEN
  - Basic Macros, [105](#)
- L4\_INLINE\_RPC\_NF\_OP
  - ipc\_iface, [2024](#)
- L4\_INLINE\_RPC\_NF
  - ipc\_iface, [2024](#)
- L4\_INLINE\_RPC\_OP
  - ipc\_iface, [2025](#)
- L4\_INLINE\_RPC
  - ipc\_iface, [2023](#)
- L4\_IPC\_TIMEOUT\_0
  - Timeouts, [556](#)
- L4\_LOG2\_PAGESIZE
  - Memory related, [428](#)
- L4\_LOG2\_SUPERPAGESIZE
  - Memory related, [428](#)
- L4\_NOTHROW
  - Basic Macros, [105](#)
- L4\_PAGEMASK
  - Memory related, [428](#)
- L4\_RPC\_DEF
  - ipc\_client, [2020](#)
- L4\_RPC\_NF\_OP
  - ipc\_iface, [2026](#)
- L4\_RPC\_NF
  - ipc\_iface, [2026](#)
- L4\_RPC\_OP
  - ipc\_iface, [2027](#)
- L4\_RPC
  - ipc\_iface, [2025](#)
- L4\_SUPERPAGEMASK
  - Memory related, [429](#)
- L4\_SUPERPAGESIZE
  - Memory related, [429](#)
- l4\_addr\_consts\_t
  - Memory related, [429](#)
- l4\_assert
  - sys/assert.h, [2177](#)
- l4\_buf\_regs\_t, [1208](#)
- l4\_buffer\_desc\_consts\_t
  - Buffer Registers (BRs), [116](#)
- l4\_busy\_wait\_ns
  - Timestamp Counter, [565](#)
- l4\_busy\_wait\_us
  - Timestamp Counter, [566](#)
- l4\_cache\_clean\_data
  - Cache Consistency, [123](#)
- l4\_cache\_coherent
  - Cache Consistency, [124](#)
- l4\_cache\_dma\_coherent
  - Cache Consistency, [124](#)
- l4\_cache\_flush\_data
  - Cache Consistency, [125](#)
- l4\_cache\_inv\_data
  - Cache Consistency, [125](#)
- l4\_calibrate\_tsc
  - Timestamp Counter, [566](#)
- l4\_cap\_consts\_t
  - Capabilities, [128](#)
- L4\_cap\_fpage\_rights
  - Flex pages, [234](#)
- l4\_capability\_equal
  - Capabilities, [129](#)
- l4\_capability\_next
  - l4/sys/types.h, [1764](#)
- l4\_debugger\_get\_object\_name
  - debugger.h, [2050](#)
- l4\_debugger\_global\_id
  - Kernel Debugger, [331](#)
- l4\_debugger\_kobj\_to\_id
  - Kernel Debugger, [332](#)



- l4\_debugger\_query\_log\_name
  - debugger.h, [2050](#)
- l4\_debugger\_query\_log\_typeid
  - debugger.h, [2051](#)
- l4\_debugger\_set\_object\_name
  - Kernel Debugger, [332](#)
- l4\_debugger\_switch\_log
  - debugger.h, [2052](#)
- l4\_default\_caps\_t
  - Capabilities, [128](#)
- l4\_error
  - Error Handling, [193](#)
- l4\_error\_code\_t
  - Error codes, [198](#)
- l4\_exc\_regs\_t, [1209](#)
  - flags, [1211](#)
  - ss, [1211](#)
- l4\_factory\_create\_factory
  - Factory, [211](#)
- l4\_factory\_create\_factory\_u
  - Factory, [212](#)
- l4\_factory\_create\_gate
  - Factory, [213](#)
- l4\_factory\_create\_gate\_u
  - Factory, [214](#)
- l4\_factory\_create\_irq
  - Factory, [215](#)
- l4\_factory\_create\_irq\_u
  - Factory, [216](#)
- l4\_factory\_create\_task
  - Factory, [217](#)
- l4\_factory\_create\_task\_u
  - Factory, [218](#)
- l4\_factory\_create\_thread
  - Factory, [219](#)
- l4\_factory\_create\_thread\_u
  - Factory, [220](#)
- l4\_factory\_create\_vm
  - Factory, [221](#)
- l4\_factory\_create\_vm\_u
  - Factory, [222](#)
- l4\_fpage
  - Flex pages, [238](#)
- l4\_fpage\_all
  - Flex pages, [238](#)
- l4\_fpage\_cacheability\_opt\_t
  - Flex pages, [236](#)
- l4\_fpage\_consts
  - Flex pages, [236](#)
- l4\_fpage\_contains
  - Flex pages, [239](#)
- l4\_fpage\_invalid
  - Flex pages, [239](#)
- l4\_fpage\_ioport
  - Flex pages, [240](#)
- l4\_fpage\_max\_order
  - Flex pages, [240](#)
- l4\_fpage\_memaddr
  - Flex pages, [241](#)
- l4\_fpage\_obj
  - Flex pages, [242](#)
- l4\_fpage\_page
  - Flex pages, [243](#)
- l4\_fpage\_rights
  - Flex pages, [237](#)
- l4\_fpage\_rights
  - Flex pages, [243](#)
- l4\_fpage\_set\_rights
  - Flex pages, [244](#)
- l4\_fpage\_size
  - Flex pages, [244](#)
- l4\_fpage\_t, [1212](#)
- l4\_fpage\_type
  - Flex pages, [245](#)
- l4\_get\_hz
  - Timestamp Counter, [567](#)
- l4\_icu\_bind
  - Interrupt controller, [319](#)
- l4\_icu\_bind\_u
  - Interrupt controller, [320](#)
- l4\_icu\_flags
  - Interrupt controller, [319](#)
- l4\_icu\_info
  - Interrupt controller, [321](#)
- l4\_icu\_info\_t, [1212](#)
  - features, [1214](#)
  - Interrupt controller, [318](#)
- l4\_icu\_info\_u
  - Interrupt controller, [321](#)
- l4\_icu\_mask
  - Interrupt controller, [322](#)
- l4\_icu\_mask\_u
  - Interrupt controller, [323](#)
- l4\_icu\_msi\_info
  - Interrupt controller, [324](#)
- l4\_icu\_msi\_info\_t, [1214](#)
  - msi\_data, [1215](#)
- l4\_icu\_msi\_info\_u
  - Interrupt controller, [325](#)
- l4\_icu\_opcode
  - L4 IPC Opcodes, [348](#)
- l4\_icu\_set\_mode
  - Interrupt controller, [325](#)
- l4\_icu\_set\_mode\_u
  - Interrupt controller, [326](#)
- l4\_icu\_unbind
  - Interrupt controller, [327](#)
- l4\_icu\_unbind\_u
  - Interrupt controller, [328](#)
- l4\_icu\_unmask
  - Interrupt controller, [329](#)
- l4\_icu\_unmask\_u
  - Interrupt controller, [329](#)
- l4\_int16\_t
  - Integer Types, [298](#)
- l4\_int32\_t



- Integer Types, [298](#)
- `l4_int64_t`
  - Integer Types, [299](#)
- `l4_int8_t`
  - Integer Types, [299](#)
- `l4_iofpage`
  - Flex pages, [245](#)
- `l4_ipc`
  - Object Invocation, [461](#)
- `l4_ipc_call`
  - Object Invocation, [462](#)
- `l4_ipc_error`
  - Error Handling, [194](#)
- `l4_ipc_error_code`
  - Error Handling, [195](#)
- `l4_ipc_gate_bind_thread`
  - IPC-Gate API, [274](#)
- `l4_ipc_gate_get_infos`
  - IPC-Gate API, [274](#)
- `L4_ipc_gate_ops`
  - L4 IPC Opcodes, [349](#)
- `l4_ipc_is_rcv_error`
  - Error Handling, [196](#)
- `l4_ipc_is_snd_error`
  - Error Handling, [196](#)
- `l4_ipc_receive`
  - Object Invocation, [464](#)
- `l4_ipc_reply_and_wait`
  - Object Invocation, [465](#)
- `l4_ipc_send`
  - Object Invocation, [466](#)
- `l4_ipc_send_and_wait`
  - Object Invocation, [467](#)
- `l4_ipc_sleep`
  - Object Invocation, [469](#)
- `l4_ipc_tcr_error_t`
  - Error Handling, [193](#)
- `l4_ipc_timeout`
  - Timeouts, [557](#)
- `l4_ipc_to_errno`
  - `l4/sys/ipc.h`, [2077](#)
- `l4_ipc_wait`
  - Object Invocation, [469](#)
- `l4_irq_attach`
  - IRQs, [279](#)
- `l4_irq_attach_u`
  - IRQs, [279](#)
- `l4_irq_detach`
  - IRQs, [280](#)
- `l4_irq_detach_u`
  - IRQs, [281](#)
- `L4_irq_mode`
  - IRQs, [278](#)
- `l4_irq_mux_chain`
  - IRQs, [282](#)
- `l4_irq_mux_chain_u`
  - IRQs, [283](#)
- `l4_irq_receive`
  - IRQs, [284](#)
- `l4_irq_receive_u`
  - IRQs, [285](#)
- `l4_irq_trigger`
  - IRQs, [286](#)
- `l4_irq_trigger_u`
  - IRQs, [287](#)
- `l4_irq_unmask`
  - IRQs, [288](#)
- `l4_irq_unmask_u`
  - IRQs, [288](#)
- `l4_irq_wait`
  - IRQs, [289](#)
- `l4_irq_wait_u`
  - IRQs, [290](#)
- `l4_is_fpage_writable`
  - Flex pages, [246](#)
- `l4_is_invalid_cap`
  - Capabilities, [129](#)
- `l4_is_valid_cap`
  - Capabilities, [130](#)
- `l4_kernel_info_get_mem_desc_end`
  - Memory descriptors (C version), [421](#)
- `l4_kernel_info_get_mem_desc_is_virtual`
  - Memory descriptors (C version), [421](#)
- `l4_kernel_info_get_mem_desc_start`
  - Memory descriptors (C version), [421](#)
- `l4_kernel_info_get_mem_desc_subtype`
  - Memory descriptors (C version), [422](#)
- `l4_kernel_info_get_mem_desc_type`
  - Memory descriptors (C version), [422](#)
- `l4_kernel_info_get_num_mem_descs`
  - Memory descriptors (C version), [422](#)
- `l4_kernel_info_mem_desc_t`, [1215](#)
  - Memory descriptors (C version), [420](#)
- `l4_kernel_info_set_mem_desc`
  - Memory descriptors (C version), [423](#)
- `l4_kernel_info_version_offset`
  - Kernel Interface Page, [335](#)
- `l4_kip_clock`
  - Kernel Interface Page, [336](#)
- `l4_kip_clock_lw`
  - Kernel Interface Page, [336](#)
- `l4_kip_version`
  - Kernel Interface Page, [337](#)
- `l4_kip_version_string`
  - Kernel Interface Page, [337](#)
- `l4_map_control`
  - Message Items, [435](#)
- `l4_map_obj_control`
  - Message Items, [436](#)
- `l4_mem_info_sub_type_t`
  - Memory descriptors (C version), [420](#)
- `L4_mem_op_widths`
  - Memory operations., [424](#)
- `l4_mem_read`
  - Memory operations., [425](#)
- `l4_mem_type_t`

- Memory descriptors (C version), [421](#)
- `l4_mem_write`
  - Memory operations., [425](#)
- `l4_msg_item_consts_t`
  - Message Items, [434](#)
- `l4_msg_regs_t`, [1216](#)
- `l4_msgtag`
  - Message Tag, [441](#)
- `l4_msgtag_flags`
  - Message Tag, [440](#), [442](#)
- `l4_msgtag_has_error`
  - Message Tag, [443](#)
- `l4_msgtag_is_exception`
  - Message Tag, [444](#)
- `l4_msgtag_is_io_page_fault`
  - Message Tag, [445](#)
- `l4_msgtag_is_page_fault`
  - Message Tag, [446](#)
- `l4_msgtag_is_preemption`
  - Message Tag, [447](#)
- `l4_msgtag_is_sigma0`
  - Message Tag, [448](#)
- `l4_msgtag_is_sys_exception`
  - Message Tag, [449](#)
- `l4_msgtag_items`
  - Message Tag, [450](#)
- `l4_msgtag_label`
  - Message Tag, [451](#)
- `l4_msgtag_protocol`
  - Message Tag, [440](#)
- `l4_msgtag_t`, [1217](#)
  - flags, [1218](#)
  - Message Tag, [439](#)
- `l4_msgtag_words`
  - Message Tag, [451](#)
- `l4_ns_to_tsc`
  - Timestamp Counter, [567](#)
- `l4_obj_fpage`
  - Flex pages, [246](#)
- `L4_obj_fpage_ctl`
  - Flex pages, [237](#)
- `l4_platform_ctl_cpu_disable`
  - Platform Control C API, [473](#)
- `l4_platform_ctl_cpu_enable`
  - Platform Control C API, [474](#)
- `L4_platform_ctl_ops`
  - L4 IPC Opcodes, [350](#)
- `L4_platform_ctl_proto`
  - Message Tag, [441](#)
- `l4_platform_ctl_system_shutdown`
  - Platform Control C API, [474](#)
- `l4_platform_ctl_system_suspend`
  - Platform Control C API, [475](#)
- `l4_rcv_timeout`
  - Timeouts, [558](#)
- `l4_rdpmc`
  - Timestamp Counter, [568](#)
- `l4_rdpmc_32`
  - Timestamp Counter, [568](#)
- `l4_rdtsc`
  - Timestamp Counter, [569](#)
- `l4_rdtsc_32`
  - Timestamp Counter, [569](#)
- `l4_round_page`
  - Memory related, [430](#)
- `l4_round_size`
  - Memory related, [431](#)
- `l4_sched_cpu_set`
  - Scheduler, [497](#)
- `l4_sched_cpu_set_t`, [1219](#)
  - gran\_offset, [1220](#)
  - granularity, [1219](#)
  - offset, [1220](#)
- `l4_sched_param_t`, [1221](#)
- `l4_scheduler_idle_time`
  - Scheduler, [498](#)
- `l4_scheduler_info`
  - Scheduler, [499](#)
- `l4_scheduler_is_online`
  - Scheduler, [500](#)
- `L4_scheduler_ops`
  - Scheduler, [497](#)
- `l4_scheduler_run_thread`
  - Scheduler, [500](#)
- `l4_sleep`
  - amd64/l4/util/util.h, [1696](#)
  - Utility Functions, [574](#)
  - x86/l4/util/util.h, [1699](#)
- `l4_snd_fpage_t`, [1222](#)
- `l4_snd_timeout`
  - Timeouts, [558](#)
- `l4_sndfpage_add`
  - Object Invocation, [471](#)
- `l4_syscall_flags_t`
  - Object Invocation, [461](#)
- `l4_task_add_ku_mem`
  - Task, [526](#)
- `l4_task_cap_equal`
  - Task, [526](#)
- `l4_task_cap_has_child`
  - Task, [526](#)
- `l4_task_cap_valid`
  - Task, [527](#)
- `l4_task_delete_obj`
  - Task, [527](#)
- `L4_task_ldt_x86_consts`
  - amd64/l4/sys/segment.h, [1705](#)
  - x86/l4/sys/segment.h, [1711](#)
- `l4_task_map`
  - Task, [528](#)
- `L4_task_ops`
  - L4 IPC Opcodes, [350](#)
- `l4_task_release_cap`
  - Task, [529](#)
- `l4_task_unmap`
  - Task, [529](#)

- l4\_task\_unmap\_batch
  - Task, [530](#)
- l4\_thread\_arm\_set\_tpidrro
  - Thread, [536](#)
- l4\_thread\_control\_alien
  - Thread control, [551](#)
- l4\_thread\_control\_bind
  - Thread control, [551](#)
- l4\_thread\_control\_commit
  - Thread control, [552](#)
- l4\_thread\_control\_exc\_handler
  - Thread control, [552](#)
- L4\_thread\_control\_flags
  - Thread, [534](#)
- L4\_thread\_control\_mr\_indices
  - Thread, [534](#)
- l4\_thread\_control\_pager
  - Thread control, [553](#)
- l4\_thread\_control\_start
  - Thread control, [553](#)
- l4\_thread\_control\_ux\_host\_syscall
  - Thread control, [553](#)
- l4\_thread\_ex\_regs
  - Thread, [537](#)
- L4\_thread\_ex\_regs\_flags
  - Thread, [536](#)
- l4\_thread\_ex\_regs\_ret
  - Thread, [538](#)
- l4\_thread\_ex\_regs\_ret\_u
  - Thread, [539](#)
- l4\_thread\_ex\_regs\_u
  - Thread, [540](#)
- l4\_thread\_longjmp
  - amd64/l4f/l4/util/setjmp.h, [1722](#)
  - x86/l4f/l4/util/setjmp.h, [1725](#)
- l4\_thread\_modify\_sender\_add
  - Thread, [541](#)
- l4\_thread\_modify\_sender\_commit
  - Thread, [541](#)
- l4\_thread\_modify\_sender\_start
  - Thread, [542](#)
- L4\_thread\_ops
  - L4 IPC Opcodes, [350](#)
- l4\_thread\_register\_del\_irq
  - Thread, [542](#)
- l4\_thread\_regs\_t, [1223](#)
- l4\_thread\_setjmp
  - amd64/l4f/l4/util/setjmp.h, [1723](#)
  - x86/l4f/l4/util/setjmp.h, [1726](#)
- l4\_thread\_stats\_time
  - Thread, [543](#)
- l4\_thread\_switch
  - Thread, [543](#)
- l4\_thread\_vcpu\_control
  - Thread, [543](#)
- l4\_thread\_vcpu\_control\_ext
  - Thread, [544](#)
- l4\_thread\_vcpu\_control\_ext\_u
  - Thread, [545](#)
- l4\_thread\_vcpu\_control\_u
  - Thread, [546](#)
- l4\_thread\_vcpu\_resume\_commit
  - Thread, [547](#)
- l4\_thread\_vcpu\_resume\_start
  - Thread, [548](#)
- l4\_thread\_yield
  - Thread, [548](#)
- l4\_timeout
  - Timeouts, [559](#)
- l4\_timeout\_abs
  - Timeouts, [559](#)
- l4\_timeout\_abs\_validity
  - Timeouts, [557](#)
- l4\_timeout\_get
  - Timeouts, [560](#)
- l4\_timeout\_is\_absolute
  - Timeouts, [561](#)
- l4\_timeout\_rel
  - Timeouts, [561](#)
- l4\_timeout\_rel\_get
  - Timeouts, [562](#)
- l4\_timeout\_s, [1224](#)
  - Timeouts, [557](#)
- l4\_timeout\_t, [1225](#)
  - Timeouts, [557](#)
- l4\_touch\_ro
  - Utility Functions, [575](#)
- l4\_touch\_rw
  - Utility Functions, [575](#)
- l4\_tracebuffer\_status\_t, [1226](#)
  - cnt\_jobmap\_tlb\_flush, [1227](#)
- l4\_tracebuffer\_status\_window\_t, [1228](#)
- l4\_trunc\_page
  - Memory related, [432](#)
- l4\_trunc\_size
  - Memory related, [432](#)
- l4\_tsc\_init
  - Timestamp Counter, [570](#)
- l4\_tsc\_to\_ns
  - Timestamp Counter, [571](#)
- l4\_tsc\_to\_s\_and\_ns
  - Timestamp Counter, [571](#)
- l4\_tsc\_to\_us
  - Timestamp Counter, [572](#)
- l4\_uint16\_t
  - Integer Types, [299](#)
- l4\_uint32\_t
  - Integer Types, [299](#)
- l4\_uint64\_t
  - Integer Types, [299](#)
- l4\_uint8\_t
  - Integer Types, [299](#)
- l4\_unmap\_flags\_t
  - Task, [525](#)
- l4\_usleep
  - Utility Functions, [576](#)

- l4\_utcb\_br
  - Virtual Registers (UTCBS), [624](#)
- L4\_utcb\_consts\_x86
  - x86 Virtual Registers (UTCB), [638](#)
- l4\_utcb\_exc
  - Exception registers, [205](#)
- l4\_utcb\_exc\_is\_ex\_regs\_exception
  - Exception registers, [205](#)
- l4\_utcb\_exc\_is\_pf
  - Exception registers, [206](#)
- l4\_utcb\_exc\_pc
  - Exception registers, [206](#)
- l4\_utcb\_exc\_pc\_set
  - Exception registers, [207](#)
- l4\_utcb\_mr
  - Virtual Registers (UTCBS), [624](#)
- l4\_utcb\_mr64\_idx
  - Timeouts, [562](#)
- l4\_utcb\_t
  - Virtual Registers (UTCBS), [623](#)
- l4\_utcb\_tcr
  - Virtual Registers (UTCBS), [625](#)
- l4\_vcon\_attr\_t, [1229](#)
- l4\_vcon\_get\_attr
  - Virtual Console, [611](#)
- l4\_vcon\_get\_attr\_u
  - Virtual Console, [611](#)
- L4\_vcon\_i\_flags
  - Virtual Console, [609](#)
- L4\_vcon\_l\_flags
  - Virtual Console, [610](#)
- L4\_vcon\_o\_flags
  - Virtual Console, [610](#)
- L4\_vcon\_ops
  - L4 IPC Opcodes, [351](#)
- l4\_vcon\_read
  - Virtual Console, [612](#)
- L4\_vcon\_read\_flags
  - vcon.h, [2163](#)
- l4\_vcon\_read\_u
  - Virtual Console, [613](#)
- l4\_vcon\_read\_with\_flags
  - Virtual Console, [614](#)
- l4\_vcon\_send
  - Virtual Console, [615](#)
- l4\_vcon\_send\_u
  - Virtual Console, [616](#)
- l4\_vcon\_set\_attr
  - Virtual Console, [617](#)
- l4\_vcon\_set\_attr\_u
  - Virtual Console, [617](#)
- L4\_vcon\_size\_consts
  - Virtual Console, [610](#)
- l4\_vcon\_write
  - Virtual Console, [618](#)
- l4\_vcon\_write\_u
  - Virtual Console, [619](#)
- l4\_vcpu\_ipc\_regs\_t, [1230](#)
- l4\_vcpu\_regs\_t, [1231](#)
  - ax, [1232](#)
  - bp, [1233](#)
  - bx, [1233](#)
  - cx, [1233](#)
  - di, [1233](#)
  - dx, [1234](#)
  - si, [1234](#)
- L4\_vcpu\_state\_flags
  - vCPU API, [628](#)
- L4\_vcpu\_state\_offset
  - vCPU API, [628](#)
- l4\_vcpu\_state\_t, [1235](#)
- L4\_vcpu\_sticky\_flags
  - vCPU API, [628](#)
- l4\_vhw\_descriptor, [1237](#)
  - count, [1239](#)
  - descs, [1239](#)
  - magic, [1239](#)
  - version, [1239](#)
- l4\_vhw\_entry, [1240](#)
  - fd, [1241](#)
  - irq\_no, [1241](#)
  - mem\_size, [1241](#)
  - mem\_start, [1241](#)
  - provider\_pid, [1242](#)
  - type, [1242](#)
- l4\_vhw\_entry\_type
  - Fiasco-UX Virtual devices, [230](#)
- L4\_virtio\_cmd
  - L4 VIRTIO Transport Layer, [366](#)
- L4\_virtio\_irq\_status
  - L4 VIRTIO Transport Layer, [366](#)
- L4\_virtio\_opcodes
  - L4 VIRTIO Transport Layer, [367](#)
- l4\_vm\_svm\_vmcb\_control\_area, [1243](#)
- l4\_vm\_svm\_vmcb\_state\_save\_area, [1243](#)
- l4\_vm\_svm\_vmcb\_state\_save\_area\_seg, [1245](#)
- l4\_vm\_svm\_vmcb\_t, [1245](#)
- l4\_vm\_tz\_state, [1247](#)
- L4\_vm\_vmx\_caps\_regs
  - VM API for VMX, [584](#)
- l4\_vm\_vmx\_clear
  - VM API for VMX, [585](#)
- L4\_vm\_vmx\_dfl1\_regs
  - VM API for VMX, [584](#)
- l4\_vm\_vmx\_field\_len
  - VM API for VMX, [585](#)
- l4\_vm\_vmx\_field\_order
  - VM API for VMX, [586](#)
- l4\_vm\_vmx\_get\_caps
  - VM API for VMX, [587](#)
- l4\_vm\_vmx\_get\_caps\_default1
  - VM API for VMX, [587](#)
- l4\_vm\_vmx\_get\_cr2\_index
  - VM API for VMX, [588](#)
- l4\_vm\_vmx\_ptr\_load
  - VM API for VMX, [588](#)

- l4\_vm\_vmx\_read
  - VM API for VMX, [589](#)
- l4\_vm\_vmx\_read\_16
  - VM API for VMX, [590](#)
- l4\_vm\_vmx\_read\_32
  - VM API for VMX, [591](#)
- l4\_vm\_vmx\_read\_64
  - VM API for VMX, [591](#)
- l4\_vm\_vmx\_read\_nat
  - VM API for VMX, [592](#)
- l4\_vm\_vmx\_write
  - VM API for VMX, [592](#)
- l4\_vm\_vmx\_write\_16
  - VM API for VMX, [593](#)
- l4\_vm\_vmx\_write\_32
  - VM API for VMX, [594](#)
- l4\_vm\_vmx\_write\_64
  - VM API for VMX, [595](#)
- l4\_vm\_vmx\_write\_nat
  - VM API for VMX, [595](#)
- L4RE\_ELF\_AUX\_ELEM\_T
  - L4Re ELF Auxiliary Information, [384](#)
- L4RE\_ELF\_AUX\_ELEM
  - L4Re ELF Auxiliary Information, [384](#)
- L4Re, [670](#)
  - chkcapp, [672](#)
  - chksys, [672–674](#)
- L4Re C Interface, [375](#)
  - l4re\_inhibitor\_acquire, [376](#)
- L4Re C++ Interface, [378](#)
- L4Re Capability API, [380](#)
  - cap\_alloc, [382](#)
  - make\_auto\_cap, [381](#)
  - make\_auto\_del\_cap, [381](#)
  - make\_ref\_cap, [381](#)
  - make\_ref\_del\_cap, [382](#)
- L4Re ELF Auxiliary Information, [383](#)
  - L4RE\_ELF\_AUX\_ELEM\_T, [384](#)
  - L4RE\_ELF\_AUX\_ELEM, [384](#)
- L4Re Protocol identifiers, [386](#)
- L4Re Util C Interface, [387](#)
- L4Re Util C++ Interface, [388](#)
- L4Re::Cap\_alloc, [1247](#)
  - alloc, [1248](#), [1249](#)
  - free, [1249](#)
  - get\_cap\_alloc, [1250](#)
- L4Re::Console, [1251](#)
- L4Re::Dataspace, [1254](#)
  - allocate, [1257](#)
  - clear, [1257](#)
  - copy\_in, [1258](#)
  - flags, [1258](#)
  - info, [1260](#)
  - map, [1260](#)
  - Map\_flags, [1256](#)
  - map\_region, [1261](#)
  - phys, [1262](#)
  - size, [1263](#)
- L4Re::Dataspace::Stats, [1263](#)
- L4Re::Debug\_obj, [1264](#)
  - debug, [1266](#)
- L4Re::Dma\_space, [1266](#)
  - associate, [1269](#)
  - Attribute, [1268](#)
  - Attributes, [1267](#)
  - Direction, [1268](#)
  - disassociate, [1269](#)
  - map, [1270](#)
  - Space\_attrib, [1269](#)
  - unmap, [1270](#)
- L4Re::Env, [1271](#)
  - env, [1274](#)
  - factory, [1274](#), [1275](#)
  - first\_free\_cap, [1275](#)
  - first\_free\_utcb, [1276](#)
  - get, [1276](#)
  - get\_cap, [1277](#), [1278](#)
  - initial\_caps, [1278](#)
  - log, [1279](#)
  - main\_thread, [1279](#), [1280](#)
  - mem\_alloc, [1280](#)
  - parent, [1282](#)
  - rm, [1282](#), [1283](#)
  - scheduler, [1283](#)
  - task, [1284](#)
  - utcb\_area, [1284](#)
- L4Re::Event, [1285](#)
  - get\_buffer, [1288](#)
- L4Re::Event\_buffer\_t
  - Event\_buffer\_t, [1290](#)
  - next, [1291](#)
  - put, [1291](#)
- L4Re::Event\_buffer\_t< PAYLOAD >, [1289](#)
- L4Re::Event\_buffer\_t< PAYLOAD >::Event, [1292](#)
- L4Re::Inhibitor, [1293](#)
  - acquire, [1296](#)
  - next\_lock\_info, [1297](#)
  - release, [1298](#)
- L4Re::Log, [1298](#)
  - print, [1301](#)
  - printn, [1301](#)
- L4Re::Mem\_alloc, [1301](#)
  - alloc, [1305](#)
  - free, [1305](#)
  - Mem\_alloc\_flags, [1304](#)
- L4Re::Mmio\_space, [1306](#)
  - Access\_width, [1309](#)
  - mmio\_read, [1309](#)
  - mmio\_write, [1310](#)
- L4Re::Namespace, [1310](#)
  - query, [1314](#), [1315](#)
  - Query\_result\_flags, [1313](#)
  - Register\_flags, [1313](#)
  - register\_obj, [1316](#)
  - unlink, [1316](#)
- L4Re::Ned::Cmd\_control, [1317](#)

- execute, 1318
- L4Re::Parent, 1319
  - signal, 1321
- L4Re::Rm, 1321
  - attach, 1326, 1327
  - Attach\_flags, 1325
  - detach, 1328–1330
  - Detach\_flags, 1325
  - Detach\_result, 1325
  - find, 1330
  - free\_area, 1331
  - Region\_flags, 1326
  - reserve\_area, 1332, 1333
- L4Re::Smart\_cap\_auto< Unmap\_flags >, 1334
- L4Re::Util::Auto\_cap< T >, 1335
- L4Re::Util::Auto\_del\_cap< T >, 1336
- L4Re::Util::Br\_manager, 1338
  - alloc\_buffer\_demand, 1340
  - get\_rcv\_cap, 1340
  - realloc\_rcv\_cap, 1341
  - set\_rcv\_cap\_flags, 1341
- L4Re::Util::Br\_manager\_hooks, 1342
- L4Re::Util::Br\_manager\_timeout\_hooks, 1343
- L4Re::Util::Cap\_alloc\_base, 1346
- L4Re::Util::Counting\_cap\_alloc
  - alloc, 1348
  - Counting\_cap\_alloc, 1348
  - free, 1349
  - release, 1350
  - setup, 1351
  - take, 1352
- L4Re::Util::Counting\_cap\_alloc< COUNTERTYPE >, 1347
- L4Re::Util::Dataspace\_svr, 1352
  - allocate, 1354
  - clear, 1355
  - copy, 1355
  - is\_static, 1356
  - map, 1357
  - map\_hook, 1358
  - page\_shift, 1359
  - phys, 1360
  - release, 1361
  - take, 1361
- L4Re::Util::Event\_buffer\_consumer\_t
  - foreach\_available\_event, 1365
  - process, 1365
- L4Re::Util::Event\_buffer\_consumer\_t< PAYLOAD >, 1362
- L4Re::Util::Event\_buffer\_t
  - attach, 1368
  - buf, 1369
  - detach, 1369
- L4Re::Util::Event\_buffer\_t< PAYLOAD >, 1366
- L4Re::Util::Event\_t
  - buffer, 1371
  - init, 1371
  - init\_poll, 1373
  - irq, 1373
  - Mode, 1371
- L4Re::Util::Event\_t< PAYLOAD >, 1370
- L4Re::Util::Item\_alloc\_base, 1374
- L4Re::Util::Names::Name, 1375
- L4Re::Util::Names::Name\_space, 1376
  - alloc\_dynamic\_entry, 1378
  - copy\_receive\_cap, 1378
  - free\_capability, 1378
  - free\_dynamic\_entry, 1379
  - free\_epiface, 1379
  - get\_epiface, 1379
- L4Re::Util::Object\_registry, 1380
  - Object\_registry, 1382
  - register\_irq\_obj, 1383, 1384
  - register\_obj, 1384, 1385
  - unregister\_obj, 1385
- L4Re::Util::Ref\_cap< T >, 1386
- L4Re::Util::Ref\_del\_cap< T >, 1387
- L4Re::Util::Registry\_server
  - registry, 1391, 1392
  - Registry\_server, 1391
- L4Re::Util::Registry\_server< LOOP\_HOOKS >, 1388
- L4Re::Util::Smart\_cap\_auto< Unmap\_flags >, 1392
- L4Re::Util::Smart\_count\_cap< Unmap\_flags >, 1393
- L4Re::Util::Vcon\_svr< SVR >, 1394
- L4Re::Util::Video::Goos\_svr, 1395
  - get\_fb, 1396
  - init\_infos, 1396
  - refresh, 1397
  - screen\_info, 1398
  - view\_info, 1398
- L4Re::Vfs, 675
- L4Re::Vfs::Be\_file, 1399
  - data\_space, 1402
  - fstat64, 1402
  - unlock\_all\_locks, 1402
- L4Re::Vfs::Be\_file\_system, 1403
  - ~Be\_file\_system, 1405
  - Be\_file\_system, 1405
  - type, 1405
- L4Re::Vfs::Directory, 1406
  - faccessat, 1408
  - link, 1408
  - mkdir, 1408
  - rename, 1409
  - rmdir, 1409
  - symlink, 1410
  - unlink, 1410
- L4Re::Vfs::File, 1411
- L4Re::Vfs::File\_system, 1413
  - mount, 1414
  - type, 1415
- L4Re::Vfs::Fs, 1415
  - alloc\_fd, 1417
  - free\_fd, 1418
  - get\_file, 1418
  - mount, 1418

- set\_fd, [1419](#)
- L4Re::Vfs::Generic\_file, [1419](#)
  - fchmod, [1421](#)
  - fstat64, [1422](#)
  - get\_status\_flags, [1422](#)
  - set\_status\_flags, [1422](#)
  - unlock\_all\_locks, [1423](#)
- L4Re::Vfs::Mman, [1424](#)
- L4Re::Vfs::Ops, [1425](#)
- L4Re::Vfs::Regular\_file, [1428](#)
  - data\_space, [1430](#)
  - fdasync, [1430](#)
  - fsync, [1430](#)
  - ftruncate64, [1430](#)
  - get\_lock, [1431](#)
  - lseek64, [1431](#)
  - readv, [1431](#)
  - set\_lock, [1432](#)
  - writev, [1432](#)
- L4Re::Vfs::Special\_file, [1433](#)
  - ioctl, [1435](#)
- L4Re::Video::Color\_component, [1436](#)
  - Color\_component, [1437](#)
  - dump, [1437](#)
  - get, [1437](#)
  - operator==, [1438](#)
  - set, [1438](#)
  - shift, [1438](#)
  - size, [1439](#)
- L4Re::Video::Goos, [1440](#)
  - create\_buffer, [1443](#)
  - create\_view, [1443](#)
  - delete\_buffer, [1443](#)
  - delete\_view, [1444](#)
  - Flags, [1442](#)
  - get\_static\_buffer, [1444](#)
  - info, [1445](#)
  - view, [1445](#)
- L4Re::Video::Goos::Info, [1446](#)
  - auto\_refresh, [1447](#)
- L4Re::Video::Pixel\_info, [1448](#)
  - a, [1450](#)
  - b, [1451](#)
  - bits\_per\_pixel, [1451](#)
  - bytes\_per\_pixel, [1452](#)
  - dump, [1452](#)
  - g, [1453](#)
  - has\_alpha, [1454](#)
  - operator==, [1454](#)
  - Pixel\_info, [1449](#), [1450](#)
  - r, [1455](#)
- L4Re::Video::View, [1455](#)
  - Flags, [1457](#)
  - info, [1458](#)
  - refresh, [1458](#)
  - set\_info, [1459](#)
  - set\_viewport, [1459](#)
  - stack, [1460](#)
  - V\_flags, [1457](#)
- L4Re::Video::View::Info, [1460](#)
- L4UTIL\_MB\_MEMORY
  - mb\_info.h, [2234](#)
- l4io\_device\_types\_t
  - IO interface, [266](#)
- l4io\_get\_root\_device
  - io.h, [1759](#)
- l4io\_has\_resource
  - IO interface, [267](#)
- l4io\_iomem\_flags\_t
  - IO interface, [266](#)
- l4io\_iterate\_devices
  - io.h, [1759](#)
- l4io\_lookup\_device
  - IO interface, [268](#)
- l4io\_lookup\_resource
  - IO interface, [268](#)
- l4io\_release\_iomem
  - IO interface, [269](#)
- l4io\_release\_ioport
  - IO interface, [269](#)
- l4io\_request\_all\_ioports
  - io.h, [1760](#)
- l4io\_request\_icu
  - io.h, [1760](#)
- l4io\_request\_iomem
  - IO interface, [269](#)
- l4io\_request\_iomem\_region
  - IO interface, [270](#)
- l4io\_request\_ioport
  - IO interface, [271](#)
- l4io\_request\_resource\_iomem
  - IO interface, [271](#)
- l4io\_resource\_t
  - IO interface, [266](#)
- l4io\_resource\_types\_t
  - IO interface, [267](#)
- l4io\_search\_iomem\_region
  - IO interface, [272](#)
- l4irq\_attach
  - Interface using direct functionality., [304](#)
- l4irq\_attach\_cap
  - Interface using direct functionality., [310](#)
- l4irq\_attach\_cap\_ft
  - Interface using direct functionality., [310](#)
- l4irq\_attach\_ft
  - Interface using direct functionality., [305](#)
- l4irq\_attach\_thread
  - Interface using direct functionality., [305](#)
- l4irq\_attach\_thread\_cap
  - Interface using direct functionality., [311](#)
- l4irq\_attach\_thread\_cap\_ft
  - Interface using direct functionality., [311](#)
- l4irq\_attach\_thread\_ft
  - Interface using direct functionality., [306](#)
- l4irq\_detach
  - Interface using direct functionality., [306](#)

- l4irq\_release
  - Interface for asynchronous ISR handlers., [302](#)
- l4irq\_request
  - Interface for asynchronous ISR handlers., [303](#)
- l4irq\_request\_cap
  - Interface for asynchronous ISR handlers with a given IRQ capability., [300](#)
- l4irq\_unmask
  - Interface using direct functionality., [307](#)
- l4irq\_unmask\_and\_wait\_any
  - Interface using direct functionality., [307](#)
- l4irq\_wait
  - Interface using direct functionality., [307](#)
- l4irq\_wait\_any
  - Interface using direct functionality., [309](#)
- l4la\_alloc
  - list\_alloc.h, [2228](#)
- l4la\_avail
  - list\_alloc.h, [2228](#)
- l4la\_dump
  - list\_alloc.h, [2229](#)
- l4la\_free
  - list\_alloc.h, [2229](#)
- l4la\_init
  - list\_alloc.h, [2229](#)
- l4re\_aux\_t, [1463](#)
- l4re\_debug\_obj\_debug
  - Debug interface, [158](#)
- l4re\_dma\_space\_associate
  - DMA Space Interface, [150](#)
- l4re\_dma\_space\_direction
  - dma\_space.h, [1861](#)
- l4re\_dma\_space\_disassociate
  - DMA Space Interface, [150](#)
- l4re\_dma\_space\_dma\_addr\_t
  - dma\_space.h, [1861](#)
- l4re\_dma\_space\_map
  - DMA Space Interface, [151](#)
- l4re\_dma\_space\_space\_attribs
  - dma\_space.h, [1862](#)
- l4re\_dma\_space\_t
  - DMA Space Interface, [149](#)
- l4re\_dma\_space\_unmap
  - DMA Space Interface, [152](#)
- l4re\_ds\_allocate
  - Dataspace interface, [154](#)
- l4re\_ds\_clear
  - Dataspace interface, [154](#)
- l4re\_ds\_copy\_in
  - Dataspace interface, [155](#)
- l4re\_ds\_flags
  - Dataspace interface, [155](#)
- l4re\_ds\_info
  - Dataspace interface, [155](#)
- l4re\_ds\_map\_flags
  - Dataspace interface, [154](#)
- l4re\_ds\_phys
  - Dataspace interface, [156](#)
- l4re\_ds\_size
  - Dataspace interface, [156](#)
- l4re\_ds\_stats\_t, [1464](#)
- l4re\_elf\_aux\_mword\_t, [1464](#)
- l4re\_elf\_aux\_t, [1465](#)
- l4re\_elf\_aux\_vma\_t, [1466](#)
- l4re\_env
  - Initial Environment, [293](#)
- l4re\_env\_cap\_entry\_t, [1467](#)
  - flags, [1468](#)
  - l4re\_env\_cap\_entry\_t, [1467](#)
- l4re\_env\_get\_cap
  - Initial Environment, [293](#)
- l4re\_env\_get\_cap\_e
  - Initial Environment, [294](#)
- l4re\_env\_get\_cap\_l
  - Initial Environment, [295](#)
- l4re\_env\_t, [1469](#)
  - env.h, [1913](#)
- l4re\_event\_get\_axis\_info
  - Event interface, [201](#)
- l4re\_event\_get\_buffer
  - Event interface, [202](#)
- l4re\_event\_get\_num\_streams
  - Event interface, [202](#)
- l4re\_event\_get\_stream\_info
  - Event interface, [203](#)
- l4re\_event\_get\_stream\_info\_for\_id
  - Event interface, [203](#)
- l4re\_event\_t, [1470](#)
- l4re\_inhibitor\_acquire
  - L4Re C Interface, [376](#)
- l4re\_kip
  - Initial Environment, [296](#)
- l4re\_log\_print
  - Log interface, [406](#)
- l4re\_log\_print\_srv
  - Log interface, [407](#)
- l4re\_log\_printn
  - Log interface, [408](#)
- l4re\_log\_printn\_srv
  - Log interface, [408](#)
- l4re\_ma\_alloc
  - Memory allocator, [414](#)
- l4re\_ma\_alloc\_align
  - Memory allocator, [415](#)
- l4re\_ma\_alloc\_align\_srv
  - Memory allocator, [416](#)
- l4re\_ma\_flags
  - Memory allocator, [413](#)
- l4re\_ma\_free
  - Memory allocator, [417](#)
- l4re\_ma\_free\_srv
  - Memory allocator, [418](#)
- l4re\_ns\_query\_srv
  - Namespace interface, [455](#)
- l4re\_ns\_query\_to\_srv
  - Namespace interface, [455](#)



- [l4re\\_ns\\_register\\_flags](#)
  - Namespace interface, [454](#)
- [l4re\\_ns\\_register\\_obj\\_srv](#)
  - Namespace interface, [457](#)
- [l4re\\_rm\\_attach](#)
  - Region map interface, [485](#)
- [l4re\\_rm\\_attach\\_srv](#)
  - Region map interface, [487](#)
- [l4re\\_rm\\_detach](#)
  - Region map interface, [487](#)
- [l4re\\_rm\\_detach\\_ds](#)
  - Region map interface, [488](#)
- [l4re\\_rm\\_detach\\_ds\\_unmap](#)
  - Region map interface, [489](#)
- [l4re\\_rm\\_detach\\_srv](#)
  - Region map interface, [490](#)
- [l4re\\_rm\\_detach\\_unmap](#)
  - Region map interface, [491](#)
- [l4re\\_rm\\_find](#)
  - Region map interface, [492](#)
- [l4re\\_rm\\_find\\_srv](#)
  - Region map interface, [492](#)
- [l4re\\_rm\\_flags\\_t](#)
  - Region map interface, [485](#)
- [l4re\\_rm\\_free\\_area](#)
  - Region map interface, [493](#)
- [l4re\\_rm\\_free\\_area\\_srv](#)
  - Region map interface, [493](#)
- [l4re\\_rm\\_reserve\\_area](#)
  - Region map interface, [494](#)
- [l4re\\_rm\\_reserve\\_area\\_srv](#)
  - Region map interface, [494](#)
- [l4re\\_rm\\_show\\_lists](#)
  - Region map interface, [495](#)
- [l4re\\_util\\_cap\\_last](#)
  - Capability allocator, [131](#)
- [l4re\\_util\\_kumem\\_alloc](#)
  - Kumem allocator utility, [344](#)
- [l4re\\_video\\_color\\_component\\_t](#), [1471](#)
- [l4re\\_video\\_goos\\_create\\_buffer](#)
  - Video API, [600](#)
- [l4re\\_video\\_goos\\_create\\_view](#)
  - Video API, [601](#)
- [l4re\\_video\\_goos\\_delete\\_buffer](#)
  - Video API, [601](#)
- [l4re\\_video\\_goos\\_delete\\_view](#)
  - Video API, [603](#)
- [l4re\\_video\\_goos\\_get\\_static\\_buffer](#)
  - Video API, [603](#)
- [l4re\\_video\\_goos\\_get\\_view](#)
  - Video API, [603](#)
- [l4re\\_video\\_goos\\_info](#)
  - Video API, [604](#)
- [l4re\\_video\\_goos\\_info\\_flags\\_t](#)
  - Video API, [600](#)
- [l4re\\_video\\_goos\\_info\\_t](#), [1472](#)
- [l4re\\_video\\_goos\\_refresh](#)
  - Video API, [604](#)
- [l4re\\_video\\_pixel\\_info\\_t](#), [1473](#)
- [l4re\\_video\\_view\\_get\\_info](#)
  - Video API, [605](#)
- [l4re\\_video\\_view\\_info\\_flags\\_t](#)
  - Video API, [600](#)
- [l4re\\_video\\_view\\_info\\_t](#), [1474](#)
- [l4re\\_video\\_view\\_refresh](#)
  - Video API, [605](#)
- [l4re\\_video\\_view\\_set\\_info](#)
  - Video API, [605](#)
- [l4re\\_video\\_view\\_set\\_viewport](#)
  - Video API, [606](#)
- [l4re\\_video\\_view\\_stack](#)
  - Video API, [606](#)
- [l4re\\_video\\_view\\_t](#), [1476](#)
  - Video API, [599](#)
- [l4shmc\\_add\\_chunk](#)
  - Chunks, [132](#)
- [l4shmc\\_add\\_signal](#)
  - Signals, [516](#)
- [l4shmc\\_area\\_overhead](#)
  - Shared Memory Library, [505](#)
- [l4shmc\\_area\\_size](#)
  - Shared Memory Library, [505](#)
- [l4shmc\\_area\\_size\\_free](#)
  - Shared Memory Library, [506](#)
- [l4shmc\\_attach](#)
  - Shared Memory Library, [506](#)
- [l4shmc\\_attach\\_signal](#)
  - Signals, [517](#)
- [l4shmc\\_attach\\_signal\\_to](#)
  - Signals, [517](#)
- [l4shmc\\_attach\\_to](#)
  - Shared Memory Library, [507](#)
- [l4shmc\\_check\\_magic](#)
  - Signals, [518](#)
- [l4shmc\\_chunk\\_capacity](#)
  - Chunks, [133](#)
- [l4shmc\\_chunk\\_consumed](#)
  - Consumer, [140](#)
- [l4shmc\\_chunk\\_overhead](#)
  - Shared Memory Library, [507](#)
- [l4shmc\\_chunk\\_ptr](#)
  - Chunks, [133](#)
- [l4shmc\\_chunk\\_ready](#)
  - Producer, [476](#)
- [l4shmc\\_chunk\\_ready\\_sig](#)
  - Producer, [477](#)
- [l4shmc\\_chunk\\_signal](#)
  - Chunks, [134](#)
- [l4shmc\\_chunk\\_size](#)
  - Consumer, [141](#)
- [l4shmc\\_chunk\\_try\\_to\\_take](#)
  - Producer, [477](#)
- [l4shmc\\_connect\\_chunk\\_signal](#)
  - Shared Memory Library, [507](#)
- [l4shmc\\_create](#)
  - Shared Memory Library, [509](#)

- l4shmc\_enable\_chunk
  - Consumer, [141](#)
- l4shmc\_enable\_signal
  - Consumer, [144](#)
- l4shmc\_get\_chunk
  - Chunks, [134](#)
- l4shmc\_get\_chunk\_to
  - Chunks, [135](#)
- l4shmc\_get\_signal\_to
  - Signals, [518](#)
- l4shmc\_is\_chunk\_clear
  - Producer, [477](#)
- l4shmc\_is\_chunk\_ready
  - Consumer, [142](#)
- l4shmc\_iterate\_chunk
  - Chunks, [135](#)
- l4shmc\_signal\_cap
  - Signals, [519](#)
- l4shmc\_trigger
  - Producer, [479](#)
- l4shmc\_wait\_any
  - Consumer, [145](#)
- l4shmc\_wait\_any\_to
  - Consumer, [145](#)
- l4shmc\_wait\_any\_try
  - Consumer, [145](#)
- l4shmc\_wait\_chunk
  - Consumer, [142](#)
- l4shmc\_wait\_chunk\_to
  - Consumer, [142](#)
- l4shmc\_wait\_chunk\_try
  - Consumer, [143](#)
- l4shmc\_wait\_signal
  - Consumer, [146](#)
- l4shmc\_wait\_signal\_to
  - Consumer, [146](#)
- l4shmc\_wait\_signal\_try
  - Consumer, [147](#)
- l4sigma0\_debug\_dump
  - Sigma0 API, [511](#)
- l4sigma0\_map\_anypage
  - Sigma0 API, [511](#)
- l4sigma0\_map\_errstr
  - Sigma0 API, [512](#)
- l4sigma0\_map\_iomem
  - Sigma0 API, [513](#)
- l4sigma0\_map\_kip
  - Sigma0 API, [513](#)
- l4sigma0\_map\_mem
  - Sigma0 API, [514](#)
- l4sigma0\_map\_tbuf
  - Sigma0 API, [514](#)
- l4sigma0\_new\_client
  - Sigma0 API, [515](#)
- l4sigma0\_return\_flags\_t
  - Sigma0 API, [511](#)
- l4util\_add8
  - Atomic Instructions, [92](#)
- l4util\_add8\_res
  - Atomic Instructions, [92](#)
- l4util\_atomic\_add
  - Atomic Instructions, [92](#)
- l4util\_atomic\_inc
  - Atomic Instructions, [93](#)
- l4util\_backtrace
  - backtrace.h, [2189](#)
- l4util\_bsf
  - Bit Manipulation, [107](#)
- l4util\_bsr
  - Bit Manipulation, [108](#)
- l4util\_btc
  - Bit Manipulation, [109](#)
- l4util\_btr
  - Bit Manipulation, [109](#)
- l4util\_bts
  - Bit Manipulation, [110](#)
- l4util\_clear\_bit
  - Bit Manipulation, [111](#)
- l4util\_cmpxchg
  - Atomic Instructions, [93](#)
- l4util\_cmpxchg16
  - Atomic Instructions, [94](#)
- l4util\_cmpxchg32
  - Atomic Instructions, [94](#)
- l4util\_cmpxchg64
  - Atomic Instructions, [96](#)
- l4util\_cmpxchg8
  - Atomic Instructions, [96](#)
- l4util\_complement\_bit
  - Bit Manipulation, [111](#)
- l4util\_cpu\_capabilities
  - CPU related functions, [120](#)
- l4util\_cpu\_capabilities\_nocheck
  - CPU related functions, [121](#)
- l4util\_cpu\_has\_cpuid
  - CPU related functions, [122](#)
- l4util\_find\_first\_set\_bit
  - Bit Manipulation, [112](#)
- l4util\_find\_first\_zero\_bit
  - Bit Manipulation, [112](#)
- l4util\_idt\_desc\_t, [1477](#)
- l4util\_idt\_header\_t, [1478](#)
- l4util\_in16
  - IA32 Port I/O API, [258](#)
- l4util\_in32
  - IA32 Port I/O API, [260](#)
- l4util\_in8
  - IA32 Port I/O API, [260](#)
- l4util\_inc8
  - Atomic Instructions, [97](#)
- l4util\_inc8\_res
  - Atomic Instructions, [97](#)
- l4util\_ins16
  - IA32 Port I/O API, [261](#)
- l4util\_ins32
  - IA32 Port I/O API, [261](#)

- l4util\_ins8
  - IA32 Port I/O API, [262](#)
- l4util\_ioport\_map
  - x86/l4/l4/util/port\_io.h, [1716](#)
- l4util\_irq\_acknowledge
  - amd64/l4/util/irq.h, [1839](#)
  - x86/l4/util/irq.h, [1842](#)
- l4util\_kip\_for\_each\_feature
  - Kernel Interface Page API, [339](#)
- l4util\_kip\_kernel\_abi\_version
  - Kernel Interface Page API, [340](#)
- l4util\_kip\_kernel\_has\_feature
  - Kernel Interface Page API, [340](#)
- l4util\_kip\_kernel\_is\_ux
  - Kernel Interface Page API, [340](#)
- l4util\_mb\_addr\_range\_t, [1479](#)
- l4util\_mb\_apm\_t, [1480](#)
- l4util\_mb\_drive\_t, [1480](#)
  - drive\_cylinders, [1481](#)
  - drive\_mode, [1481](#)
  - drive\_number, [1482](#)
- l4util\_mb\_info\_t, [1482](#)
- l4util\_mb\_mod\_t, [1484](#)
  - mod\_end, [1485](#)
  - mod\_start, [1485](#)
- l4util\_mb\_vbe\_ctrl\_t, [1485](#)
- l4util\_mb\_vbe\_mode\_t, [1486](#)
- l4util\_memdesc\_vm\_high
  - Kernel Interface Page API, [340](#)
- l4util\_micros2l4to
  - amd64/l4/util/util.h, [1696](#)
  - Utility Functions, [577](#)
  - x86/l4/util/util.h, [1699](#)
- l4util\_next\_power2
  - Bit Manipulation, [112](#)
- l4util\_out16
  - IA32 Port I/O API, [262](#)
- l4util\_out32
  - IA32 Port I/O API, [262](#)
- l4util\_out8
  - IA32 Port I/O API, [263](#)
- l4util\_outs16
  - IA32 Port I/O API, [263](#)
- l4util\_outs32
  - IA32 Port I/O API, [264](#)
- l4util\_outs8
  - IA32 Port I/O API, [264](#)
- l4util\_rand
  - Random number support, [480](#)
- l4util\_set\_bit
  - Bit Manipulation, [113](#)
- l4util\_splitlog2\_hdl
  - Utility Functions, [577](#)
- l4util\_splitlog2\_size
  - Utility Functions, [578](#)
- l4util\_srand
  - Random number support, [480](#)
- l4util\_stack\_get\_sp
  - amd64/l4/util/stack\_impl.h, [1754](#)
  - arm/l4/util/stack\_impl.h, [1753](#)
  - stack.h, [2250](#)
  - x86/l4/util/stack\_impl.h, [1756](#)
- l4util\_test\_bit
  - Bit Manipulation, [113](#)
- l4util\_xchg
  - Atomic Instructions, [97](#)
- l4util\_xchg16
  - Atomic Instructions, [98](#)
- l4util\_xchg32
  - Atomic Instructions, [98](#)
- l4util\_xchg8
  - Atomic Instructions, [99](#)
- L4vbus, [676](#)
- L4vbus GPIO functions, [389](#)
  - l4vbus\_gpio\_config\_get, [391](#)
  - l4vbus\_gpio\_config\_pad, [391](#)
  - l4vbus\_gpio\_config\_pull, [392](#)
  - L4vbus\_gpio\_generic\_func, [390](#)
  - l4vbus\_gpio\_get, [393](#)
  - l4vbus\_gpio\_multi\_config\_pad, [393](#)
  - l4vbus\_gpio\_multi\_get, [394](#)
  - l4vbus\_gpio\_multi\_set, [395](#)
  - l4vbus\_gpio\_multi\_setup, [396](#)
  - L4vbus\_gpio\_pull\_modes, [390](#)
  - l4vbus\_gpio\_set, [397](#)
  - l4vbus\_gpio\_setup, [397](#)
  - l4vbus\_gpio\_to\_irq, [398](#)
- L4vbus PCI functions, [400](#)
  - l4vbus\_pci\_cfg\_read, [400](#)
  - l4vbus\_pci\_cfg\_write, [401](#)
  - l4vbus\_pci\_irq\_enable, [402](#)
  - l4vbus\_pciddev\_cfg\_read, [403](#)
  - l4vbus\_pciddev\_cfg\_write, [403](#)
  - l4vbus\_pciddev\_irq\_enable, [404](#)
- L4vbus::Device, [1489](#)
  - \_bus, [1497](#)
  - bus\_cap, [1491](#)
  - dev\_handle, [1492](#)
  - device, [1492](#)
  - device\_by\_hid, [1494](#)
  - get\_resource, [1495](#)
  - is\_compatible, [1495](#)
  - next\_device, [1496](#)
  - operator!=, [1497](#)
  - operator==, [1497](#)
- L4vbus::Gpio\_module, [1498](#)
  - config\_pad, [1501](#)
  - get, [1502](#)
  - pin, [1502](#)
  - set, [1503](#)
  - setup, [1503](#)
- L4vbus::Gpio\_module::Pin\_slice, [1504](#)
- L4vbus::Gpio\_pin, [1505](#)
  - config\_get, [1508](#)
  - config\_pad, [1509](#)
  - config\_pull, [1509](#)

- get, [1510](#)
- pin, [1510](#)
- set, [1511](#)
- setup, [1511](#)
- to\_irq, [1512](#)
- L4vbus::Icu, [1513](#)
- L4vbus::Pci\_dev, [1515](#)
  - cfg\_read, [1518](#)
  - cfg\_write, [1518](#)
  - irq\_enable, [1519](#)
- L4vbus::Pci\_host\_bridge, [1520](#)
  - cfg\_read, [1523](#)
  - cfg\_write, [1524](#)
  - irq\_enable, [1525](#)
- L4vbus::Pm< DEC >, [1526](#)
- L4vbus::Vbus, [1527](#)
  - release\_resource, [1530](#)
  - request\_resource, [1530](#)
  - root, [1531](#)
- l4vbus\_assign\_dma\_domain
  - L4 V-BUS functions, [353](#)
- l4vbus\_device\_t, [1532](#)
- L4vbus\_dma\_domain\_assign\_flags
  - L4 V-BUS functions, [353](#)
- l4vbus\_get\_device
  - L4 V-BUS functions, [354](#)
- l4vbus\_get\_device\_by\_hid
  - L4 V-BUS functions, [355](#)
- l4vbus\_get\_hid
  - L4 V-BUS functions, [356](#)
- l4vbus\_get\_next\_device
  - L4 V-BUS functions, [356](#)
- l4vbus\_get\_resource
  - L4 V-BUS functions, [357](#)
- l4vbus\_gpio\_config\_get
  - L4vbus GPIO functions, [391](#)
- l4vbus\_gpio\_config\_pad
  - L4vbus GPIO functions, [391](#)
- l4vbus\_gpio\_config\_pull
  - L4vbus GPIO functions, [392](#)
- L4vbus\_gpio\_generic\_func
  - L4vbus GPIO functions, [390](#)
- l4vbus\_gpio\_get
  - L4vbus GPIO functions, [393](#)
- l4vbus\_gpio\_multi\_config\_pad
  - L4vbus GPIO functions, [393](#)
- l4vbus\_gpio\_multi\_get
  - L4vbus GPIO functions, [394](#)
- l4vbus\_gpio\_multi\_set
  - L4vbus GPIO functions, [395](#)
- l4vbus\_gpio\_multi\_setup
  - L4vbus GPIO functions, [396](#)
- L4vbus\_gpio\_pull\_modes
  - L4vbus GPIO functions, [390](#)
- l4vbus\_gpio\_set
  - L4vbus GPIO functions, [397](#)
- l4vbus\_gpio\_setup
  - L4vbus GPIO functions, [397](#)
- l4vbus\_gpio\_to\_irq
  - L4vbus GPIO functions, [398](#)
- l4vbus\_is\_compatible
  - L4 V-BUS functions, [357](#)
- l4vbus\_pci\_cfg\_read
  - L4vbus PCI functions, [400](#)
- l4vbus\_pci\_cfg\_write
  - L4vbus PCI functions, [401](#)
- l4vbus\_pci\_irq\_enable
  - L4vbus PCI functions, [402](#)
- l4vbus\_pciddev\_cfg\_read
  - L4vbus PCI functions, [403](#)
- l4vbus\_pciddev\_cfg\_write
  - L4vbus PCI functions, [403](#)
- l4vbus\_pciddev\_irq\_enable
  - L4vbus PCI functions, [404](#)
- l4vbus\_release\_resource
  - L4 V-BUS functions, [358](#)
- l4vbus\_request\_resource
  - L4 V-BUS functions, [359](#)
- l4vbus\_resource\_t, [1533](#)
- l4vbus\_resource\_type\_t
  - vbus\_types.h, [2269](#)
- l4vbus\_vicu\_get\_cap
  - L4 V-BUS functions, [360](#)
- L4vcpu::State, [1534](#)
  - add, [1535](#)
  - clear, [1535](#)
  - set, [1535](#)
  - State, [1534](#)
- L4vcpu::Vcpu, [1536](#)
  - cast, [1540](#)
  - entry\_ip, [1540](#)
  - entry\_sp, [1541](#)
  - ext\_alloc, [1541](#)
  - i, [1542](#)
  - irq\_disable\_save, [1542](#)
  - irq\_enable, [1543](#)
  - irq\_restore, [1544](#)
  - is\_irq\_entry, [1544](#)
  - is\_page\_fault\_entry, [1545](#)
  - r, [1545](#), [1546](#)
  - saved\_state, [1546](#)
  - state, [1547](#)
  - task, [1547](#)
  - wait\_for\_event, [1548](#)
- l4vcpu\_ext\_alloc
  - Extended vCPU support, [208](#)
- l4vcpu\_irq\_disable
  - vCPU Support Library, [631](#)
- l4vcpu\_irq\_disable\_save
  - vCPU Support Library, [631](#)
- l4vcpu\_irq\_enable
  - vCPU Support Library, [632](#)
- l4vcpu\_irq\_restore
  - vCPU Support Library, [633](#)
- l4vcpu\_is\_irq\_entry
  - vCPU Support Library, [634](#)

- l4vcpu\_is\_page\_fault\_entry
  - vCPU Support Library, [635](#)
- l4vcpu\_print\_state
  - vCPU Support Library, [635](#)
- l4vcpu\_wait\_for\_event
  - vCPU Support Library, [636](#)
- L4virtio, [677](#)
- L4virtio::Driver::Virtqueue, [1549](#)
  - alloc\_descriptor, [1551](#)
  - desc, [1551](#)
  - enqueue\_descriptor, [1552](#)
  - find\_next\_used, [1552](#)
  - free\_descriptor, [1552](#)
  - init\_queue, [1553](#)
  - initialize\_rings, [1554](#)
- L4virtio::Ptr
  - get, [1556](#)
  - Invalid\_type, [1556](#)
  - is\_valid, [1557](#)
- L4virtio::Ptr< T >, [1554](#)
- L4virtio::Svr::Bad\_descriptor, [1558](#)
  - Bad\_descriptor, [1559](#)
  - Error, [1559](#)
- L4virtio::Svr::Block\_dev
  - Block\_dev, [1562](#)
  - finalize\_request, [1563](#)
  - process\_request, [1563](#)
  - register\_obj, [1564](#)
  - set\_blk\_size, [1565](#)
  - set\_size\_max, [1565](#)
  - set\_topology, [1565](#)
- L4virtio::Svr::Block\_dev< Ds\_data >, [1560](#)
- L4virtio::Svr::Block\_request
  - data\_size, [1567](#)
  - next\_block, [1568](#)
- L4virtio::Svr::Block\_request< Ds\_data >, [1566](#)
- L4virtio::Svr::Data\_buffer, [1568](#)
  - copy\_to, [1570](#)
  - Data\_buffer, [1570](#)
  - done, [1571](#)
  - set, [1571](#)
  - skip, [1572](#)
- L4virtio::Svr::Dev\_config, [1572](#)
  - change\_queue\_config, [1575](#)
  - Dev\_config, [1574](#)
  - ds, [1576](#)
  - get\_cmd, [1577](#)
  - hdr, [1578](#)
  - qconfig, [1578](#)
  - reset\_cmd, [1579](#)
  - reset\_queue, [1580](#)
  - set\_failed, [1581](#)
  - set\_status, [1582](#)
  - status, [1582](#)
- L4virtio::Svr::Dev\_features, [1583](#)
  - ring\_event\_idx, [1585](#)
  - ring\_event\_idx\_bfm\_t, [1585](#)
  - ring\_indirect\_desc, [1585](#), [1586](#)
  - ring\_indirect\_desc\_bfm\_t, [1585](#)
- L4virtio::Svr::Dev\_status, [1586](#)
  - acked, [1589](#)
  - acked\_bfm\_t, [1588](#)
  - driver, [1589](#), [1590](#)
  - driver\_bfm\_t, [1588](#)
  - driver\_ok, [1590](#)
  - driver\_ok\_bfm\_t, [1588](#)
  - failed, [1590](#), [1591](#)
  - failed\_bfm\_t, [1588](#)
  - feature\_ok, [1591](#)
  - feature\_ok\_bfm\_t, [1588](#)
  - running, [1592](#)
- L4virtio::Svr::Device\_t
  - device\_error, [1595](#)
  - handle\_mem\_cmd\_write, [1595](#)
  - init\_mem\_info, [1595](#)
  - reset\_queue\_config, [1596](#)
  - setup\_queue, [1596](#)
- L4virtio::Svr::Device\_t< DATA >, [1593](#)
- L4virtio::Svr::Driver\_mem\_list\_t
  - add, [1599](#)
  - find, [1600](#)
  - full, [1600](#)
  - init, [1601](#)
  - load\_desc, [1602](#), [1603](#)
  - remove, [1603](#)
- L4virtio::Svr::Driver\_mem\_list\_t< DATA >, [1597](#)
- L4virtio::Svr::Driver\_mem\_region\_t
  - contains, [1606](#)
  - Driver\_mem\_region\_t, [1606](#)
  - drv\_base, [1607](#)
  - ds, [1607](#)
  - ds\_offset, [1607](#)
  - empty, [1607](#)
  - flags, [1608](#)
  - is\_writable, [1608](#)
  - local, [1608](#)
  - local\_base, [1609](#)
  - size, [1609](#)
- L4virtio::Svr::Driver\_mem\_region\_t< DATA >, [1604](#)
- L4virtio::Svr::Request\_processor, [1610](#)
  - current\_flags, [1611](#)
  - has\_more, [1611](#)
  - next, [1612](#)
  - start, [1613](#), [1615](#)
- L4virtio::Svr::Virtqueue, [1615](#)
  - consumed, [1618](#)
  - desc, [1619](#)
  - desc\_avail, [1619](#)
  - disable\_notify, [1620](#)
  - enable\_notify, [1620](#)
  - next\_avail, [1620](#)
- L4virtio::Svr::Virtqueue::Head\_desc, [1622](#)
  - desc, [1622](#)
  - operator Null\_ptr\_check const \*, [1623](#)
  - valid, [1623](#)
- L4virtio::Virtqueue, [1624](#)

- avail\_align, 1627
- avail\_size, 1627
- desc\_align, 1628
- desc\_size, 1628
- disable, 1629
- dump, 1629
- get\_avail\_idx, 1630
- get\_tail\_avail\_idx, 1630
- no\_notify\_guest, 1631
- no\_notify\_host, 1631, 1632
- num, 1632
- ready, 1633
- setup, 1633
- setup\_simple, 1634
- total\_size, 1635
- used\_align, 1636
- used\_size, 1636
- L4virtio::Virtqueue::Avail, 1637
- L4virtio::Virtqueue::Avail::Flags, 1638
  - no\_irq, 1639
  - no\_irq\_bfm\_t, 1639
- L4virtio::Virtqueue::Desc, 1640
- L4virtio::Virtqueue::Desc::Flags, 1642
  - indirect, 1644
  - indirect\_bfm\_t, 1643
  - next, 1645
  - next\_bfm\_t, 1644
  - write, 1645
  - write\_bfm\_t, 1644
- L4virtio::Virtqueue::Used, 1646
- L4virtio::Virtqueue::Used::Flags, 1647
  - no\_notify, 1648, 1649
  - no\_notify\_bfm\_t, 1648
- L4virtio::Virtqueue::Used\_elem, 1649
  - Used\_elem, 1650
- l4virtio\_block\_config\_t, 1650
  - blk\_size, 1651
- l4virtio\_block\_header\_t, 1652
- L4virtio\_block\_operations
  - L4 VIRTIO Block Device, 361
- L4virtio\_block\_status
  - L4 VIRTIO Block Device, 362
- l4virtio\_config\_hdr\_t, 1653
  - magic, 1654
  - status, 1654
- l4virtio\_config\_queue
  - L4 VIRTIO Transport Layer, 368
- l4virtio\_config\_queue\_t, 1655
  - L4 VIRTIO Transport Layer, 365
- l4virtio\_config\_queues
  - L4 VIRTIO Transport Layer, 369
- l4virtio\_device\_config
  - L4 VIRTIO Transport Layer, 369
- L4virtio\_device\_ids
  - L4 VIRTIO Transport Layer, 367
- L4virtio\_device\_status
  - L4 VIRTIO Transport Layer, 368
- L4virtio\_feature\_bits
  - L4 VIRTIO Transport Layer, 368
- l4virtio\_register\_ds
  - L4 VIRTIO Transport Layer, 370
- l4virtio\_register\_iface
  - L4 VIRTIO Transport Layer, 371
- l4virtio\_set\_status
  - L4 VIRTIO Transport Layer, 371
- length
  - L4::lpc::Varg, 991
- libedid\_check\_header
  - EDID parsing functionality, 161
- libedid\_checksum
  - EDID parsing functionality, 161
- Libedid\_consts
  - EDID parsing functionality, 160
- libedid\_dump
  - EDID parsing functionality, 161
- libedid\_dump\_standard\_timings
  - EDID parsing functionality, 162
- libedid\_num\_ext\_blocks
  - EDID parsing functionality, 162
- libedid\_pnp\_id
  - EDID parsing functionality, 162
- libedid\_prefered\_resolution
  - EDID parsing functionality, 163
- libedid\_revision
  - EDID parsing functionality, 163
- libedid\_version
  - EDID parsing functionality, 163
- link
  - L4Re::Vfs::Directory, 1408
- List\_alloc
  - cxx::List\_alloc, 787
- list\_alloc.h
  - l4la\_alloc, 2228
  - l4la\_avail, 2228
  - l4la\_dump, 2229
  - l4la\_free, 2229
  - l4la\_init, 2229
- load\_desc
  - L4virtio::Svr::Driver\_mem\_list\_t, 1602, 1603
- local
  - L4virtio::Svr::Driver\_mem\_region\_t, 1608
- local\_base
  - L4virtio::Svr::Driver\_mem\_region\_t, 1609
- local\_id\_received
  - L4::lpc::Gen\_fpage, 937
- log
  - L4Re::Env, 1279
- Log interface, 406
  - l4re\_log\_print, 406
  - l4re\_log\_print\_srv, 407
  - l4re\_log\_printn, 408
  - l4re\_log\_printn\_srv, 408
- Logging interface, 410
- loop
  - L4::Server, 1117
- Low-Level Thread Functions, 411

- lower\_bound\_node
  - cxx::Bits::Base\_avl\_set, [748](#)
  - cxx::Bits::Bst, [762](#)
- lseek64
  - L4Re::Vfs::Regular\_file, [1431](#)
- Lstr
  - L4::Factory::Lstr, [896](#)
- MB\_ARD\_MEMORY
  - mb\_info.h, [2234](#)
- MB\_ART\_MEMORY
  - mb\_info.h, [2235](#)
- Machine Restarting Function, [412](#)
- magic
  - l4\_vhw\_descriptor, [1239](#)
  - l4virtio\_config\_hdr\_t, [1654](#)
- main\_thread
  - L4Re::Env, [1279](#), [1280](#)
- make\_auto\_cap
  - L4Re Capability API, [381](#)
- make\_auto\_del\_cap
  - L4Re Capability API, [381](#)
- make\_cap
  - L4::lpc, [658](#)
- make\_cap\_full
  - L4::lpc, [659](#)
- make\_cap\_rw
  - L4::lpc, [659](#)
- make\_cap\_rws
  - L4::lpc, [660](#)
- make\_ref\_cap
  - L4Re Capability API, [381](#)
- make\_ref\_del\_cap
  - L4Re Capability API, [382](#)
- map
  - L4::Task, [1136](#)
  - L4Re::Dataspace, [1260](#)
  - L4Re::Dma\_space, [1270](#)
  - L4Re::Util::Dataspace\_svr, [1357](#)
- Map\_flags
  - L4Re::Dataspace, [1256](#)
- map\_hook
  - L4Re::Util::Dataspace\_svr, [1358](#)
- map\_region
  - L4Re::Dataspace, [1261](#)
- mask
  - L4::lcu, [906](#)
- Masks
- cxx::Bitfield, [710](#)
- max
  - Small C++ Template Library, [521](#)
- mb\_info.h
  - L4UTIL\_MB\_MEMORY, [2234](#)
  - MB\_ARD\_MEMORY, [2234](#)
  - MB\_ART\_MEMORY, [2235](#)
- mem\_alloc
  - L4Re::Env, [1280](#)
- Mem\_alloc\_flags
  - L4Re::Mem\_alloc, [1304](#)
- Mem\_desc
  - L4::Kip::Mem\_desc, [1050](#)
- mem\_size
  - l4\_vhw\_entry, [1241](#)
- mem\_start
  - l4\_vhw\_entry, [1241](#)
- Mem\_type
  - L4::Kip::Mem\_desc, [1050](#)
- Memory allocator, [413](#)
  - l4re\_ma\_alloc, [414](#)
  - l4re\_ma\_alloc\_align, [415](#)
  - l4re\_ma\_alloc\_align\_srv, [416](#)
  - l4re\_ma\_flags, [413](#)
  - l4re\_ma\_free, [417](#)
  - l4re\_ma\_free\_srv, [418](#)
- Memory descriptors (C version), [419](#)
  - l4\_kernel\_info\_get\_mem\_desc\_end, [421](#)
  - l4\_kernel\_info\_get\_mem\_desc\_is\_virtual, [421](#)
  - l4\_kernel\_info\_get\_mem\_desc\_start, [421](#)
  - l4\_kernel\_info\_get\_mem\_desc\_subtype, [422](#)
  - l4\_kernel\_info\_get\_mem\_desc\_type, [422](#)
  - l4\_kernel\_info\_get\_num\_mem\_descs, [422](#)
  - l4\_kernel\_info\_mem\_desc\_t, [420](#)
  - l4\_kernel\_info\_set\_mem\_desc, [423](#)
  - l4\_mem\_info\_sub\_type\_t, [420](#)
  - l4\_mem\_type\_t, [421](#)
- Memory operations., [424](#)
  - L4\_mem\_op\_widths, [424](#)
  - l4\_mem\_read, [425](#)
  - l4\_mem\_write, [425](#)
- Memory related, [427](#)
  - L4\_LOG2\_PAGESIZE, [428](#)
  - L4\_LOG2\_SUPERPAGESIZE, [428](#)
  - L4\_PAGEMASK, [428](#)
  - L4\_SUPERPAGEMASK, [429](#)
  - L4\_SUPERPAGESIZE, [429](#)
  - l4\_addr\_consts\_t, [429](#)
  - l4\_round\_page, [430](#)
  - l4\_round\_size, [431](#)
  - l4\_trunc\_page, [432](#)
  - l4\_trunc\_size, [432](#)
- Message Items, [434](#)
  - l4\_map\_control, [435](#)
  - l4\_map\_obj\_control, [436](#)
  - l4\_msg\_item\_consts\_t, [434](#)
- Message Registers (MRs), [437](#)
- Message Tag, [438](#)
  - l4\_msgtag, [441](#)
  - l4\_msgtag\_flags, [440](#), [442](#)
  - l4\_msgtag\_has\_error, [443](#)
  - l4\_msgtag\_is\_exception, [444](#)
  - l4\_msgtag\_is\_io\_page\_fault, [445](#)
  - l4\_msgtag\_is\_page\_fault, [446](#)
  - l4\_msgtag\_is\_preemption, [447](#)
  - l4\_msgtag\_is\_sigma0, [448](#)
  - l4\_msgtag\_is\_sys\_exception, [449](#)
  - l4\_msgtag\_items, [450](#)
  - l4\_msgtag\_label, [451](#)



- l4\_msgtag\_protocol, [440](#)
- l4\_msgtag\_t, [439](#)
- l4\_msgtag\_words, [451](#)
- L4\_platform\_ctl\_proto, [441](#)
- min
  - Small C++ Template Library, [521](#)
- mkdir
  - L4Re::Vfs::Directory, [1408](#)
- mmio\_read
  - L4Re::Mmio\_space, [1309](#)
- mmio\_write
  - L4Re::Mmio\_space, [1310](#)
- mod\_end
  - l4util\_mb\_mod\_t, [1485](#)
- mod\_start
  - l4util\_mb\_mod\_t, [1485](#)
- Mode
  - L4Re::Util::Event\_t, [1371](#)
- modify\_senders
  - L4::Thread, [1144](#)
- mount
  - L4Re::Vfs::File\_system, [1414](#)
  - L4Re::Vfs::Fs, [1418](#)
- move
  - L4::Cap, [857](#)
  - L4::Cap\_base, [865](#)
- msg\_add
  - L4::lpc::Msg, [667](#)
- msg\_get
  - L4::lpc::Msg, [668](#)
- Msg\_ptr
  - L4::lpc::Msg\_ptr, [975](#)
- msg\_ptr
  - L4::lpc, [661](#)
- msi\_data
  - l4\_icu\_msi\_info\_t, [1215](#)
- msi\_info
  - L4::lcu, [907](#)
- NT\_VERSION
  - ELF binary format, [188](#)
- Name-space API, [453](#)
- Namespace interface, [454](#)
  - l4re\_ns\_query\_srv, [455](#)
  - l4re\_ns\_query\_to\_srv, [455](#)
  - l4re\_ns\_register\_flags, [454](#)
  - l4re\_ns\_register\_obj\_srv, [457](#)
- next
  - L4Re::Event\_buffer\_t, [1291](#)
  - L4virtio::Svr::Request\_processor, [1612](#)
  - L4virtio::Virtqueue::Desc::Flags, [1645](#)
- next\_avail
  - L4virtio::Svr::Virtqueue, [1620](#)
- next\_bfm\_t
  - L4virtio::Virtqueue::Desc::Flags, [1644](#)
- next\_block
  - L4virtio::Svr::Block\_request, [1568](#)
- next\_device
  - L4vbus::Device, [1496](#)
- next\_lock\_info
  - L4Re::Inhibitor, [1297](#)
- next\_timeout
  - L4::lpc\_svr::Timeout\_queue, [1027](#)
- no\_demand
  - L4::Type\_info::Demand, [1161](#)
- No\_init\_type
  - L4::Cap\_base, [860](#)
- no\_irq
  - L4virtio::Virtqueue::Avail::Flags, [1639](#)
- no\_irq\_bfm\_t
  - L4virtio::Virtqueue::Avail::Flags, [1639](#)
- no\_notify
  - L4virtio::Virtqueue::Used::Flags, [1648](#), [1649](#)
- no\_notify\_bfm\_t
  - L4virtio::Virtqueue::Used::Flags, [1648](#)
- no\_notify\_guest
  - L4virtio::Virtqueue, [1631](#)
- no\_notify\_host
  - L4virtio::Virtqueue, [1631](#), [1632](#)
- None\_type
  - L4::Types::Flags, [1189](#)
- num
  - L4virtio::Virtqueue, [1632](#)
- num\_interfaces
  - L4::Meta, [1080](#)
- Object Invocation, [459](#)
  - l4\_ipc, [461](#)
  - l4\_ipc\_call, [462](#)
  - l4\_ipc\_receive, [464](#)
  - l4\_ipc\_reply\_and\_wait, [465](#)
  - l4\_ipc\_send, [466](#)
  - l4\_ipc\_send\_and\_wait, [467](#)
  - l4\_ipc\_sleep, [469](#)
  - l4\_ipc\_wait, [469](#)
  - l4\_sndfpage\_add, [471](#)
  - l4\_syscall\_flags\_t, [461](#)
- Object\_registry
  - L4Re::Util::Object\_registry, [1382](#)
- offset
  - l4\_sched\_cpu\_set\_t, [1220](#)
- Opcode
  - L4::Platform\_control, [1090](#)
- operator l4\_msgtag\_t
  - L4::Factory::S, [899](#)
- operator new
  - Small C++ Template Library, [522](#)
- operator Null\_ptr\_check const \*
  - L4virtio::Svr::Virtqueue::Head\_desc, [1623](#)
- operator Priv\_type \*
  - cxx::Auto\_ptr, [682](#)
- operator!
  - cxx::Bits::Direction, [770](#)
- operator!=
  - L4vbus::Device, [1497](#)
- operator<<
  - ipc\_stream, [1806–1808](#)
  - L4::Factory::S, [899–901](#)



operator>>  
     ipc\_stream, [1809–1813](#)  
 operator\*  
     cxx::Auto\_ptr, [682](#)  
     cxx::Bits::Base\_avl\_set::Node, [751](#)  
 operator()  
     cxx::Pair\_first\_compare, [807](#)  
 operator->  
     cxx::Auto\_ptr, [682](#)  
     cxx::Bits::Base\_avl\_set::Node, [751](#)  
 operator=  
     cxx::Auto\_ptr, [682](#)  
 operator==  
     L4Re::Video::Color\_component, [1438](#)  
     L4Re::Video::Pixel\_info, [1454](#)  
     L4vbus::Device, [1497](#)  
 operator[]  
     cxx::Avl\_map, [687](#), [688](#)  
     cxx::Bitmap\_base, [732](#)  
     cxx::List, [784](#)  
  
 pSLIM\_BMAP\_START\_LSB  
     bitmap.h, [1851](#)  
 PT\_GNU\_EH\_FRAME  
     ELF binary format, [189](#)  
 PT\_GNU\_RELRO  
     ELF binary format, [189](#)  
 PT\_GNU\_STACK  
     ELF binary format, [189](#)  
 PT\_HIOS  
     ELF binary format, [189](#)  
 PT\_HIPROC  
     ELF binary format, [189](#)  
 PT\_L4\_AUX  
     ELF binary format, [190](#)  
 PT\_L4\_KIP  
     ELF binary format, [190](#)  
 PT\_L4\_STACK  
     ELF binary format, [190](#)  
 PT\_LOOS  
     ELF binary format, [190](#)  
 PT\_LOPROC  
     ELF binary format, [190](#)  
 page\_fault  
     L4::Pager, [1086](#)  
 page\_shift  
     L4Re::Util::Dataspace\_svr, [1359](#)  
 pager  
     L4::Thread::Attr, [1151](#), [1152](#)  
 Pair  
     cxx::Pair, [806](#)  
 Pair\_first\_compare  
     cxx::Pair\_first\_compare, [807](#)  
 parent  
     L4Re::Env, [1282](#)  
 Parent API, [472](#)  
 parse\_cmdline  
     Comfortable Command Line Parsing, [136](#)  
 phys  
     L4Re::Dataspace, [1262](#)  
     L4Re::Util::Dataspace\_svr, [1360](#)  
 pin  
     L4vbus::Gpio\_module, [1502](#)  
     L4vbus::Gpio\_pin, [1510](#)  
 Pixel\_info  
     L4Re::Video::Pixel\_info, [1449](#), [1450](#)  
 Platform Control C API, [473](#)  
     l4\_platform\_ctl\_cpu\_disable, [473](#)  
     l4\_platform\_ctl\_cpu\_enable, [474](#)  
     l4\_platform\_ctl\_system\_shutdown, [474](#)  
     l4\_platform\_ctl\_system\_suspend, [475](#)  
 Poll\_timeout\_kipclock  
     L4::Poll\_timeout\_kipclock, [1093](#)  
 pop\_front  
     cxx::H\_list, [776](#)  
     cxx::S\_list, [816](#)  
 print  
     L4Re::Log, [1301](#)  
 printn  
     L4Re::Log, [1301](#)  
 process  
     L4Re::Util::Event\_buffer\_consumer\_t, [1365](#)  
 process\_request  
     L4virtio::Svr::Block\_dev, [1563](#)  
 Producer, [476](#), [479](#)  
     l4shmc\_chunk\_ready, [476](#)  
     l4shmc\_chunk\_ready\_sig, [477](#)  
     l4shmc\_chunk\_try\_to\_take, [477](#)  
     l4shmc\_is\_chunk\_clear, [477](#)  
     l4shmc\_trigger, [479](#)  
 proto\_dispatch  
     L4::Kobject\_typeid, [1074](#)  
     L4::Server\_object\_t, [1123](#)  
 provider\_pid  
     l4\_vhw\_entry, [1242](#)  
 ptr  
     cxx::Ref\_ptr, [812](#)  
 push\_back  
     cxx::List, [784](#)  
     cxx::List\_item, [794](#)  
 push\_front  
     cxx::List, [785](#)  
     cxx::List\_item, [795](#)  
 put  
     L4::ipc::Ostream, [980](#), [981](#)  
     L4Re::Event\_buffer\_t, [1291](#)  
  
 qconfig  
     L4virtio::Svr::Dev\_config, [1578](#)  
 query  
     L4Re::Namespace, [1314](#), [1315](#)  
 query\_log\_name  
     L4::Debugger, [875](#)  
 query\_log\_typeid  
     L4::Debugger, [876](#)  
 Query\_result\_flags  
     L4Re::Namespace, [1313](#)

- r
  - L4Re::Video::Pixel\_info, 1455
  - L4vcpu::Vcpu, 1545, 1546
- Random number support, 480
  - l4util\_rand, 480
  - l4util\_srand, 480
- rbegin
  - cxx::Bits::Base\_avl\_set, 748
  - cxx::Bits::Bst, 763, 764
- rcv\_cap
  - L4::lpc\_svr::Server\_iface, 1018, 1019
- read
  - L4::lpc, 661
  - L4::Vcon, 1201
- read\_with\_flags
  - L4::Vcon, 1202
- readv
  - L4Re::Vfs::Regular\_file, 1431
- ready
  - L4virtio::Virtqueue, 1633
- realloc\_rcv\_cap
  - L4::lpc\_svr::Server\_iface, 1020
  - L4Re::Util::Br\_manager, 1341
- Realtime API, 482
- receive
  - L4::lpc::lstream, 951
  - L4::lrcq, 1037
- Ref
  - cxx::Bitfield, 708
- Ref\_ptr
  - cxx::Ref\_ptr, 811, 812
- Ref\_type
  - cxx::Auto\_ptr, 680
- Ref\_unshifted
  - cxx::Bitfield, 709
- refresh
  - L4Re::Util::Video::Goos\_svr, 1397
  - L4Re::Video::View, 1458
- Region map API, 483
- Region map interface, 484
  - l4re\_rm\_attach, 485
  - l4re\_rm\_attach\_srv, 487
  - l4re\_rm\_detach, 487
  - l4re\_rm\_detach\_ds, 488
  - l4re\_rm\_detach\_ds\_unmap, 489
  - l4re\_rm\_detach\_srv, 490
  - l4re\_rm\_detach\_unmap, 491
  - l4re\_rm\_find, 492
  - l4re\_rm\_find\_srv, 492
  - l4re\_rm\_flags\_t, 485
  - l4re\_rm\_free\_area, 493
  - l4re\_rm\_free\_area\_srv, 493
  - l4re\_rm\_reserve\_area, 494
  - l4re\_rm\_reserve\_area\_srv, 494
  - l4re\_rm\_show\_lists, 495
- Region\_flags
  - L4Re::Rm, 1326
- register\_del\_irq
  - L4::Thread, 1145
- Register\_flags
  - L4Re::Namespace, 1313
- register\_irq\_obj
  - L4::Registry\_iface, 1098, 1099
  - L4Re::Util::Object\_registry, 1383, 1384
- register\_obj
  - L4::Registry\_iface, 1100
  - L4Re::Namespace, 1316
  - L4Re::Util::Object\_registry, 1384, 1385
  - L4virtio::Svr::Block\_dev, 1564
- registry
  - L4Re::Util::Registry\_server, 1391, 1392
- Registry\_server
  - L4Re::Util::Registry\_server, 1391
- release
  - cxx::Auto\_ptr, 683
  - cxx::Ref\_ptr, 813
  - L4Re::Inhibitor, 1298
  - L4Re::Util::Counting\_cap\_alloc, 1350
  - L4Re::Util::Dataspace\_svr, 1361
- release\_cap
  - L4::Task, 1136
- release\_resource
  - L4vbus::Vbus, 1530
- remove
  - cxx::Avl\_tree, 695
  - cxx::Bits::Base\_avl\_set, 748
  - cxx::H\_list, 777
  - cxx::List, 785
  - cxx::List\_item, 795
  - L4::lpc\_svr::Timeout\_queue, 1027
  - L4virtio::Svr::Driver\_mem\_list\_t, 1603
- remove\_all
  - cxx::Bits::Bst, 764
- remove\_me
  - cxx::List\_item, 796
  - cxx::List\_item::lter, 799
- remove\_timeout
  - L4::lpc\_svr::Server\_iface, 1021
  - L4::lpc\_svr::Timeout\_queue\_hooks, 1031
- remove\_tree
  - cxx::Bits::Bst, 765
- rename
  - L4Re::Vfs::Directory, 1409
- rend
  - cxx::Bits::Base\_avl\_set, 749
  - cxx::Bits::Bst, 765, 766
- replace
  - cxx::H\_list, 777
- reply\_and\_wait
  - L4::lpc::lostream, 943
- Reply\_mode
  - Server-Side IPC framework, 503
- request\_resource
  - L4vbus::Vbus, 1530
- reserve\_area
  - L4Re::Rm, 1332, 1333

- reset
  - L4::lpc::loststream, [944](#)
  - L4::lpc::lstream, [952](#)
- reset\_cmd
  - L4virtio::Svr::Dev\_config, [1579](#)
- reset\_queue
  - L4virtio::Svr::Dev\_config, [1580](#)
- reset\_queue\_config
  - L4virtio::Svr::Device\_t, [1596](#)
- ring\_event\_idx
  - L4virtio::Svr::Dev\_features, [1585](#)
- ring\_event\_idx\_bfm\_t
  - L4virtio::Svr::Dev\_features, [1585](#)
- ring\_indirect\_desc
  - L4virtio::Svr::Dev\_features, [1585](#), [1586](#)
- ring\_indirect\_desc\_bfm\_t
  - L4virtio::Svr::Dev\_features, [1585](#)
- rm
  - L4Re::Env, [1282](#), [1283](#)
- rmdir
  - L4Re::Vfs::Directory, [1409](#)
- root
  - L4vbus::Vbus, [1531](#)
- run\_thread
  - L4::Scheduler, [1109](#)
- running
  - L4virtio::Svr::Dev\_status, [1592](#)
- Runtime\_error
  - L4::Runtime\_error, [1104](#)
- S
  - L4::Factory::S, [898](#)
- SHF\_GROUP
  - ELF binary format, [191](#)
- SHF\_MASKOS
  - ELF binary format, [191](#)
- SHF\_TLS
  - ELF binary format, [191](#)
- SHT\_NUM
  - ELF binary format, [191](#)
- saved\_state
  - L4vcpu::Vcpu, [1546](#)
- scan\_zero
  - cxx::Bitmap, [724](#)
  - cxx::Bitmap\_base, [733](#)
- Scheduler, [496](#)
  - I4\_sched\_cpu\_set, [497](#)
  - I4\_scheduler\_idle\_time, [498](#)
  - I4\_scheduler\_info, [499](#)
  - I4\_scheduler\_is\_online, [500](#)
  - L4\_scheduler\_ops, [497](#)
  - I4\_scheduler\_run\_thread, [500](#)
- scheduler
  - L4Re::Env, [1283](#)
- screen\_info
  - L4Re::Util::Video::Goos\_svr, [1398](#)
- send
  - L4::lpc::Ostream, [981](#)
  - L4::Vcon, [1202](#)
- Server
  - L4::Server, [1116](#)
- Server-Side IPC framework, [502](#)
  - Reply\_mode, [503](#)
- set
  - cxx::Bitfield, [711](#)
  - L4::Kip::Mem\_desc, [1055](#)
  - L4::Poll\_timeout\_kipclock, [1093](#)
  - L4Re::Video::Color\_component, [1438](#)
  - L4vbus::Gpio\_module, [1503](#)
  - L4vbus::Gpio\_pin, [1511](#)
  - L4vcpu::State, [1535](#)
  - L4virtio::Svr::Data\_buffer, [1571](#)
- set\_attr
  - L4::Vcon, [1203](#)
- set\_bit
  - cxx::Bitmap\_base, [734](#)
- set\_blk\_size
  - L4virtio::Svr::Block\_dev, [1565](#)
- set\_dirty
  - cxx::Bitfield, [712](#)
- set\_failed
  - L4virtio::Svr::Dev\_config, [1581](#)
- set\_fd
  - L4Re::Vfs::Fs, [1419](#)
- set\_info
  - L4Re::Video::View, [1459](#)
- set\_lock
  - L4Re::Vfs::Regular\_file, [1432](#)
- set\_mode
  - L4::lcu, [907](#)
- set\_object\_name
  - L4::Debugger, [876](#)
- set\_rcv\_cap\_flags
  - L4Re::Util::Br\_manager, [1341](#)
- set\_size\_max
  - L4virtio::Svr::Block\_dev, [1565](#)
- set\_status
  - L4virtio::Svr::Dev\_config, [1582](#)
- set\_status\_flags
  - L4Re::Vfs::Generic\_file, [1422](#)
- set\_topology
  - L4virtio::Svr::Block\_dev, [1565](#)
- set\_unshifted
  - cxx::Bitfield, [713](#)
- set\_unshifted\_dirty
  - cxx::Bitfield, [714](#)
- set\_viewport
  - L4Re::Video::View, [1459](#)
- setup
  - L4Re::Util::Counting\_cap\_alloc, [1351](#)
  - L4vbus::Gpio\_module, [1503](#)
  - L4vbus::Gpio\_pin, [1511](#)
  - L4virtio::Virtqueue, [1633](#)
- setup\_queue
  - L4virtio::Svr::Device\_t, [1596](#)
- setup\_simple
  - L4virtio::Virtqueue, [1634](#)

- Shared Memory Library, [504](#)
  - [l4shmc\\_area\\_overhead](#), [505](#)
  - [l4shmc\\_area\\_size](#), [505](#)
  - [l4shmc\\_area\\_size\\_free](#), [506](#)
  - [l4shmc\\_attach](#), [506](#)
  - [l4shmc\\_attach\\_to](#), [507](#)
  - [l4shmc\\_chunk\\_overhead](#), [507](#)
  - [l4shmc\\_connect\\_chunk\\_signal](#), [507](#)
  - [l4shmc\\_create](#), [509](#)
- shift
  - [L4Re::Video::Color\\_component](#), [1438](#)
- Shift\_type
  - [cxx::Bitfield](#), [709](#)
- si
  - [l4\\_vcpu\\_regs\\_t](#), [1234](#)
- Sigma0 API, [510](#)
  - [l4sigma0\\_debug\\_dump](#), [511](#)
  - [l4sigma0\\_map\\_anypage](#), [511](#)
  - [l4sigma0\\_map\\_errstr](#), [512](#)
  - [l4sigma0\\_map\\_iomem](#), [513](#)
  - [l4sigma0\\_map\\_kip](#), [513](#)
  - [l4sigma0\\_map\\_mem](#), [514](#)
  - [l4sigma0\\_map\\_tbuf](#), [514](#)
  - [l4sigma0\\_new\\_client](#), [515](#)
  - [l4sigma0\\_return\\_flags\\_t](#), [511](#)
- signal
  - [L4Re::Parent](#), [1321](#)
- Signals, [516](#)
  - [l4shmc\\_add\\_signal](#), [516](#)
  - [l4shmc\\_attach\\_signal](#), [517](#)
  - [l4shmc\\_attach\\_signal\\_to](#), [517](#)
  - [l4shmc\\_check\\_magic](#), [518](#)
  - [l4shmc\\_get\\_signal\\_to](#), [518](#)
  - [l4shmc\\_signal\\_cap](#), [519](#)
- size
  - [cxx::List](#), [785](#)
  - [L4::Kip::Mem\\_desc](#), [1055](#)
  - [L4Re::Dataspace](#), [1263](#)
  - [L4Re::Video::Color\\_component](#), [1439](#)
  - [L4virtio::Svr::Driver\\_mem\\_region\\_t](#), [1609](#)
- skip
  - [L4::lpc::Istream](#), [952](#)
  - [L4virtio::Svr::Data\\_buffer](#), [1572](#)
- Small C++ Template Library, [520](#)
  - [max](#), [521](#)
  - [min](#), [521](#)
  - [operator new](#), [522](#)
- Small\_buf
  - [L4::lpc::Small\\_buf](#), [986](#)
- Smart\_cap
  - [L4::Smart\\_cap](#), [1129](#)
- snd\_base
  - [L4::Cap\\_base](#), [866](#)
- Space\_attrib
  - [L4Re::Dma\\_space](#), [1269](#)
- ss
  - [l4\\_exc\\_regs\\_t](#), [1211](#)
- stack
  - [L4Re::Video::View](#), [1460](#)
- stack.h
  - [l4util\\_stack\\_get\\_sp](#), [2250](#)
- start
  - [L4::Kip::Mem\\_desc](#), [1056](#)
  - [L4virtio::Svr::Request\\_processor](#), [1613](#), [1615](#)
- State
  - [L4vcpu::State](#), [1534](#)
- state
  - [L4vcpu::Vcpu](#), [1547](#)
- stats\_time
  - [L4::Thread](#), [1145](#)
- status
  - [L4virtio::Svr::Dev\\_config](#), [1582](#)
  - [l4virtio\\_config\\_hdr\\_t](#), [1654](#)
- Str\_cp\_in
  - [L4::lpc::Str\\_cp\\_in](#), [988](#)
- str\_cp\_in
  - [L4::lpc](#), [662](#)
- sub\_type
  - [L4::Kip::Mem\\_desc](#), [1056](#)
- supports
  - [L4::Meta](#), [1080](#)
- switch\_log
  - [L4::Debugger](#), [877](#)
- switch\_to
  - [L4::Thread](#), [1146](#)
- symlink
  - [L4Re::Vfs::Directory](#), [1410](#)
- sys/assert.h
  - [l4\\_assert](#), [2177](#)
- system\_shutdown
  - [L4::Platform\\_control](#), [1091](#)
- system\_suspend
  - [L4::Platform\\_control](#), [1091](#)
- tag
  - [L4::lpc::Istream](#), [953](#)
  - [L4::lpc::Ostream](#), [982](#)
  - [L4::lpc::Varg](#), [992](#)
- take
  - [L4Re::Util::Counting\\_cap\\_alloc](#), [1352](#)
  - [L4Re::Util::Dataspace\\_svr](#), [1361](#)
- Task, [524](#)
  - [l4\\_task\\_add\\_ku\\_mem](#), [526](#)
  - [l4\\_task\\_cap\\_equal](#), [526](#)
  - [l4\\_task\\_cap\\_has\\_child](#), [526](#)
  - [l4\\_task\\_cap\\_valid](#), [527](#)
  - [l4\\_task\\_delete\\_obj](#), [527](#)
  - [l4\\_task\\_map](#), [528](#)
  - [l4\\_task\\_release\\_cap](#), [529](#)
  - [l4\\_task\\_unmap](#), [529](#)
  - [l4\\_task\\_unmap\\_batch](#), [530](#)
  - [l4\\_unmap\\_flags\\_t](#), [525](#)
- task
  - [L4Re::Env](#), [1284](#)
  - [L4vcpu::Vcpu](#), [1547](#)
- test
  - [L4::Poll\\_timeout\\_kipclock](#), [1094](#)

- Thread, [532](#)
  - [l4\\_thread\\_arm\\_set\\_tpidruro](#), [536](#)
  - [L4\\_thread\\_control\\_flags](#), [534](#)
  - [L4\\_thread\\_control\\_mr\\_indices](#), [534](#)
  - [l4\\_thread\\_ex\\_regs](#), [537](#)
  - [L4\\_thread\\_ex\\_regs\\_flags](#), [536](#)
  - [l4\\_thread\\_ex\\_regs\\_ret](#), [538](#)
  - [l4\\_thread\\_ex\\_regs\\_ret\\_u](#), [539](#)
  - [l4\\_thread\\_ex\\_regs\\_u](#), [540](#)
  - [l4\\_thread\\_modify\\_sender\\_add](#), [541](#)
  - [l4\\_thread\\_modify\\_sender\\_commit](#), [541](#)
  - [l4\\_thread\\_modify\\_sender\\_start](#), [542](#)
  - [l4\\_thread\\_register\\_del\\_irq](#), [542](#)
  - [l4\\_thread\\_stats\\_time](#), [543](#)
  - [l4\\_thread\\_switch](#), [543](#)
  - [l4\\_thread\\_vcpu\\_control](#), [543](#)
  - [l4\\_thread\\_vcpu\\_control\\_ext](#), [544](#)
  - [l4\\_thread\\_vcpu\\_control\\_ext\\_u](#), [545](#)
  - [l4\\_thread\\_vcpu\\_control\\_u](#), [546](#)
  - [l4\\_thread\\_vcpu\\_resume\\_commit](#), [547](#)
  - [l4\\_thread\\_vcpu\\_resume\\_start](#), [548](#)
  - [l4\\_thread\\_yield](#), [548](#)
- Thread control, [550](#)
  - [l4\\_thread\\_control\\_alien](#), [551](#)
  - [l4\\_thread\\_control\\_bind](#), [551](#)
  - [l4\\_thread\\_control\\_commit](#), [552](#)
  - [l4\\_thread\\_control\\_exc\\_handler](#), [552](#)
  - [l4\\_thread\\_control\\_pager](#), [553](#)
  - [l4\\_thread\\_control\\_start](#), [553](#)
  - [l4\\_thread\\_control\\_ux\\_host\\_syscall](#), [553](#)
- Thread Control Registers (TCRs), [549](#)
- throw\_ipc\_exception
  - [L4](#), [651](#), [653](#)
- timed\_out
  - [L4::Poll\\_timeout\\_kipclock](#), [1095](#)
- timeout
  - [L4::lpc\\_svr::Timeout](#), [1024](#)
- timeout\_expired
  - [L4::lpc\\_svr::Timeout\\_queue](#), [1028](#)
- Timeouts, [555](#)
  - [L4\\_IPC\\_TIMEOUT\\_0](#), [556](#)
  - [l4\\_ipc\\_timeout](#), [557](#)
  - [l4\\_rcv\\_timeout](#), [558](#)
  - [l4\\_snd\\_timeout](#), [558](#)
  - [l4\\_timeout](#), [559](#)
  - [l4\\_timeout\\_abs](#), [559](#)
  - [l4\\_timeout\\_abs\\_validity](#), [557](#)
  - [l4\\_timeout\\_get](#), [560](#)
  - [l4\\_timeout\\_is\\_absolute](#), [561](#)
  - [l4\\_timeout\\_rel](#), [561](#)
  - [l4\\_timeout\\_rel\\_get](#), [562](#)
  - [l4\\_timeout\\_s](#), [557](#)
  - [l4\\_timeout\\_t](#), [557](#)
  - [l4\\_utcb\\_mr64\\_idx](#), [562](#)
- Timestamp Counter, [564](#)
  - [l4\\_busy\\_wait\\_ns](#), [565](#)
  - [l4\\_busy\\_wait\\_us](#), [566](#)
  - [l4\\_calibrate\\_tsc](#), [566](#)
  - [l4\\_get\\_hz](#), [567](#)
  - [l4\\_ns\\_to\\_tsc](#), [567](#)
  - [l4\\_rdpmc](#), [568](#)
  - [l4\\_rdpmc\\_32](#), [568](#)
  - [l4\\_rdtsc](#), [569](#)
  - [l4\\_rdtsc\\_32](#), [569](#)
  - [l4\\_tsc\\_init](#), [570](#)
  - [l4\\_tsc\\_to\\_ns](#), [571](#)
  - [l4\\_tsc\\_to\\_s\\_and\\_ns](#), [571](#)
  - [l4\\_tsc\\_to\\_us](#), [572](#)
- to\_irq
  - [L4vbus::Gpio\\_pin](#), [1512](#)
- total\_objects
  - [cxx::Base\\_slab](#), [701](#)
  - [cxx::Base\\_slab\\_static](#), [706](#)
- total\_size
  - [L4virtio::Virtqueue](#), [1635](#)
- trigger
  - [L4::Triggerable](#), [1156](#)
- type
  - [L4::lpc::Varg](#), [993](#)
  - [L4::Kip::Mem\\_desc](#), [1057](#)
  - [l4\\_vhw\\_entry](#), [1242](#)
  - [L4Re::Vfs::Be\\_file\\_system](#), [1405](#)
  - [L4Re::Vfs::File\\_system](#), [1415](#)
- unbind
  - [L4::lcu](#), [908](#)
  - [L4::lommu](#), [918](#)
- unlink
  - [L4Re::Namespace](#), [1316](#)
  - [L4Re::Vfs::Directory](#), [1410](#)
- unlock\_all\_locks
  - [L4Re::Vfs::Be\\_file](#), [1402](#)
  - [L4Re::Vfs::Generic\\_file](#), [1423](#)
- unmap
  - [L4::Task](#), [1137](#)
  - [L4Re::Dma\\_space](#), [1270](#)
- unmap\_batch
  - [L4::Task](#), [1137](#)
- unmask
  - [L4::lrc](#), [1038](#)
  - [L4::lrc\\_eoi](#), [1041](#)
- unregister\_obj
  - [L4::Registry\\_iface](#), [1101](#)
  - [L4Re::Util::Object\\_registry](#), [1385](#)
- up
  - [L4::Semaphore](#), [1114](#)
- used\_align
  - [L4virtio::Virtqueue](#), [1636](#)
- Used\_elem
  - [L4virtio::Virtqueue::Used\\_elem](#), [1650](#)
- used\_size
  - [L4virtio::Virtqueue](#), [1636](#)
- utcb\_area
  - [L4Re::Env](#), [1284](#)
- Utility Functions, [573](#)
  - [l4\\_sleep](#), [574](#)
  - [l4\\_touch\\_ro](#), [575](#)

- l4\_touch\_rw, [575](#)
- l4\_usleep, [576](#)
- l4util\_micros2l4to, [577](#)
- l4util\_splitlog2\_hdl, [577](#)
- l4util\_splitlog2\_size, [578](#)
- ux\_host\_syscall
  - L4::Thread::Attr, [1152](#)
- V\_flags
  - L4Re::Video::View, [1457](#)
- vCPU API, [627](#)
  - L4\_vcpu\_state\_flags, [628](#)
  - L4\_vcpu\_state\_offset, [628](#)
  - L4\_vcpu\_sticky\_flags, [628](#)
- vCPU Support Library, [630](#)
  - l4vcpu\_irq\_disable, [631](#)
  - l4vcpu\_irq\_disable\_save, [631](#)
  - l4vcpu\_irq\_enable, [632](#)
  - l4vcpu\_irq\_restore, [633](#)
  - l4vcpu\_is\_irq\_entry, [634](#)
  - l4vcpu\_is\_page\_fault\_entry, [635](#)
  - l4vcpu\_print\_state, [635](#)
  - l4vcpu\_wait\_for\_event, [636](#)
- VM API for SVM, [580](#)
- VM API for TZ, [581](#)
- VM API for VMX, [582](#)
  - L4\_vm\_vmx\_caps\_regs, [584](#)
  - l4\_vm\_vmx\_clear, [585](#)
  - L4\_vm\_vmx\_dfl1\_regs, [584](#)
  - l4\_vm\_vmx\_field\_len, [585](#)
  - l4\_vm\_vmx\_field\_order, [586](#)
  - l4\_vm\_vmx\_get\_caps, [587](#)
  - l4\_vm\_vmx\_get\_caps\_default1, [587](#)
  - l4\_vm\_vmx\_get\_cr2\_index, [588](#)
  - l4\_vm\_vmx\_ptr\_load, [588](#)
  - l4\_vm\_vmx\_read, [589](#)
  - l4\_vm\_vmx\_read\_16, [590](#)
  - l4\_vm\_vmx\_read\_32, [591](#)
  - l4\_vm\_vmx\_read\_64, [591](#)
  - l4\_vm\_vmx\_read\_nat, [592](#)
  - l4\_vm\_vmx\_write, [592](#)
  - l4\_vm\_vmx\_write\_16, [593](#)
  - l4\_vm\_vmx\_write\_32, [594](#)
  - l4\_vm\_vmx\_write\_64, [595](#)
  - l4\_vm\_vmx\_write\_nat, [595](#)
- Val
  - cxx::Bitfield, [709](#)
- val
  - cxx::Bitfield, [715](#)
- val\_dirty
  - cxx::Bitfield, [715](#)
- Val\_unshifted
  - cxx::Bitfield, [709](#)
- val\_unshifted
  - cxx::Bitfield, [716](#)
- valid
  - cxx::Bits::Base\_avl\_set::Node, [751](#)
  - L4virtio::Svr::Virtqueue::Head\_desc, [1623](#)
- validate
  - L4::Cap\_base, [867](#)
- value
  - L4::lpc::Varg, [993](#)
- Varg\_list\_ref
  - L4::lpc::Varg\_list\_ref, [996](#)
- vbus\_types.h
  - l4vbus\_resource\_type\_t, [2269](#)
- vcon.h
  - L4\_vcon\_read\_flags, [2163](#)
- vcpu\_control
  - L4::Thread, [1146](#)
- vcpu\_control\_ext
  - L4::Thread, [1147](#)
- vcpu\_resume\_commit
  - L4::Thread, [1148](#)
- vcpu\_resume\_start
  - L4::Thread, [1148](#)
- version
  - l4\_vhw\_descriptor, [1239](#)
- Video API, [598](#)
  - l4re\_video\_goos\_create\_buffer, [600](#)
  - l4re\_video\_goos\_create\_view, [601](#)
  - l4re\_video\_goos\_delete\_buffer, [601](#)
  - l4re\_video\_goos\_delete\_view, [603](#)
  - l4re\_video\_goos\_get\_static\_buffer, [603](#)
  - l4re\_video\_goos\_get\_view, [603](#)
  - l4re\_video\_goos\_info, [604](#)
  - l4re\_video\_goos\_info\_flags\_t, [600](#)
  - l4re\_video\_goos\_refresh, [604](#)
  - l4re\_video\_view\_get\_info, [605](#)
  - l4re\_video\_view\_info\_flags\_t, [600](#)
  - l4re\_video\_view\_refresh, [605](#)
  - l4re\_video\_view\_set\_info, [605](#)
  - l4re\_video\_view\_set\_viewport, [606](#)
  - l4re\_video\_view\_stack, [606](#)
  - l4re\_video\_view\_t, [599](#)
- view
  - L4Re::Video::Goos, [1445](#)
- view\_info
  - L4Re::Util::Video::Goos\_svr, [1398](#)
- Virtual Console, [608](#)
  - l4\_vcon\_get\_attr, [611](#)
  - l4\_vcon\_get\_attr\_u, [611](#)
  - L4\_vcon\_i\_flags, [609](#)
  - L4\_vcon\_l\_flags, [610](#)
  - L4\_vcon\_o\_flags, [610](#)
  - l4\_vcon\_read, [612](#)
  - l4\_vcon\_read\_u, [613](#)
  - l4\_vcon\_read\_with\_flags, [614](#)
  - l4\_vcon\_send, [615](#)
  - l4\_vcon\_send\_u, [616](#)
  - l4\_vcon\_set\_attr, [617](#)
  - l4\_vcon\_set\_attr\_u, [617](#)
  - L4\_vcon\_size\_consts, [610](#)
  - l4\_vcon\_write, [618](#)
  - l4\_vcon\_write\_u, [619](#)
- Virtual Machines, [621](#)
- Virtual Registers (UTCBs), [622](#)

- [l4\\_utcb\\_br](#), [624](#)
  - [l4\\_utcb\\_mr](#), [624](#)
  - [l4\\_utcb\\_t](#), [623](#)
  - [l4\\_utcb\\_tcr](#), [625](#)
- wait
  - [L4::lpc::lstream](#), [954](#)
  - [L4::lrc](#), [1039](#)
- wait\_for\_event
  - [L4vcpu::Vcpu](#), [1548](#)
- word\_index
  - [cxx::Bitmap\\_base](#), [735](#)
- words
  - [cxx::Bitmap\\_base](#), [735](#)
- write
  - [L4::Vcon](#), [1204](#)
  - [L4virtio::Virtqueue::Desc::Flags](#), [1645](#)
- write\_bfm\_t
  - [L4virtio::Virtqueue::Desc::Flags](#), [1644](#)
- writetev
  - [L4Re::Vfs::Regular\\_file](#), [1432](#)
- x86 Virtual Registers (UTCB), [638](#)
  - [L4\\_utcb\\_consts\\_x86](#), [638](#)
- [x86/l4/sys/cache.h](#), [2002](#)
- [x86/l4/sys/consts.h](#), [2016](#), [2017](#)
- [x86/l4/sys/l4int.h](#), [2103](#), [2104](#)
- [x86/l4/sys/linkage.h](#), [1728](#), [1729](#)
- [x86/l4/sys/segment.h](#), [1710](#), [1711](#)
  - [L4\\_task\\_ldt\\_x86\\_consts](#), [1711](#)
- [x86/l4/sys/utcb.h](#), [2155](#), [2157](#)
- [x86/l4/util/apic.h](#), [1662](#), [1663](#)
- [x86/l4/util/bitops\\_arch.h](#), [1737](#), [1738](#)
- [x86/l4/util/cpu.h](#), [1744](#), [1745](#)
- [x86/l4/util/idt.h](#), [1670](#), [1671](#)
- [x86/l4/util/irq.h](#), [1841](#), [1842](#)
  - [l4util\\_irq\\_acknowledge](#), [1842](#)
- [x86/l4/util/l4\\_macros.h](#), [1748](#), [1749](#)
- [x86/l4/util/mbi\\_argv.h](#), [1751](#), [1752](#)
- [x86/l4/util/perform.h](#), [1677](#), [1678](#)
- [x86/l4/util/port\\_io.h](#), [1717](#), [1719](#)
- [x86/l4/util/rdtsc.h](#), [1688](#), [1689](#)
- [x86/l4/util/spin.h](#), [1694](#)
- [x86/l4/util/stack\\_impl.h](#), [1755](#), [1756](#)
  - [l4util\\_stack\\_get\\_sp](#), [1756](#)
- [x86/l4/util/util.h](#), [1698](#), [1700](#)
  - [l4\\_sleep](#), [1699](#)
  - [l4util\\_micros2l4to](#), [1699](#)
- [x86/l4f/l4/sys/ipc.h](#), [2082](#), [2083](#)
- [x86/l4f/l4/sys/segment.h](#), [1708](#), [1709](#)
- [x86/l4f/l4/util/port\\_io.h](#), [1714](#), [1717](#)
  - [l4util\\_ioport\\_map](#), [1716](#)
- [x86/l4f/l4/util/setjmp.h](#), [1724](#), [1726](#)
  - [l4\\_thread\\_longjmp](#), [1725](#)
  - [l4\\_thread\\_setjmp](#), [1726](#)