Dokumentation des Advantech PCL726-Treibers für Linux

Norman Feske

29. Januar 2002

Inhaltsverzeichnis

1 Einführung

Die Advantech PCL726-Controllerkarte bietet 6 analoge Ausgänge. Für jeden Ausgang kann die Spannung per Software separat eingestellt werden. Dabei kann für jeden Kanal (per Jumper auf der Karte) konfiguriert werden, in welchem Bereich die Spannung liegen soll (z.B. -10V bis 10V). Das Festlegen der eigentlichen Ausgangsspannung per Software erfolgt immer relativ zum gewählten Bereich.

Zusätzlich bietet die Karte die Möglichkeit, digitale Ein/Ausgabe-Kanäle zu nutzen. Der hier beschriebene Treiber beschränkt sich jedoch auf das Ansteuern der analogen Ausgabekanäle. Die digitalen Kanäle werden zur Zeit nicht unterstützt.

Die Hardware-Register zur Programmierung der Karte werden vom PCI-Bus in den physischen Adressraum des Rechners eingeblendet *(mapping)*. Ein Anwendungsprogramm unter Linux läuft im sog. *'userlevel-mode'*. Das heißt, daß ein Programm nicht direkt auf Hardware-Ressourcen zugreifen darf und somit auch nicht die Hardware-Register der Controllerkarte direkt nutzen kann.

1.1 Devices

Für ein Anwendungsprogramm besteht der einzige Weg zur Nutzung von Hardware-Ressourcen in der Nutzung sog. logischer *'devices'*. 'devices' sind eine spezielle Art von Dateien. Solche Dateien können gelesen und geschrieben werden, wie beliebige andere Dateien (es wird weiter unterschieden in 'block'- und 'stream devices' sowie 'memory mapped devices' - im vorliegenden Fall sind nur 'stream devices' von Interesse).

Der große Unterschied ist, daß die Datei Ein- und Ausgaben nicht vom Filesystem behandelt werden, sondern vom Linux-Kernel. Im Kernel gibt es nun Behandlungsroutinen *('handler')*, die den Zugriff auf die verschiedenen 'device'-Typen vornehmen. Verschiedene Typen sind z.Bsp. Festplatten, serielle Schnittstelle, Framebuffer etc.

Damit verschiedene 'devices' unterschieden werden können, besitzt jeder 'device'-Typ eine Kennungsnummer *('major number')*. Für die verschiedenen 'major numbers' besitzt der Kernel jeweils die entsprechenden Behandlungsroutinen. Wird nun auf ein spezielles 'device' zugegriffen, leitet der Kernel den Zugriff an den 'handler' mit der entsprechenden 'major number' weiter.

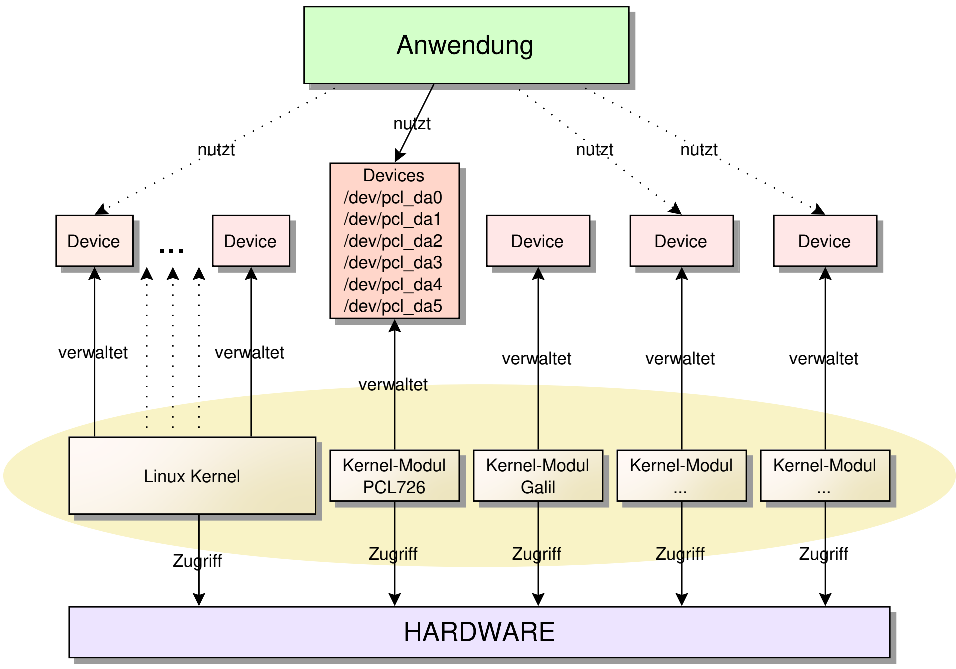
Oft kommt es vor, daß von einem 'device'-Typ mehrere 'devices' existieren (z.B. mehrere serielle Schnittstellen) auf die prinzipiell gleich zugegriffen wird (also nur ein 'handler' für die verschiedenen 'devices' nötig ist). Alle 'devices' von einem Typ bekommen nun die gleiche 'major number'. Damit ein 'handler' die einzelnen 'devices' unterscheiden kann, bekommt jedes 'device' eine zusätzliche Kennung *('minor number')* zugewiesen.

Die 'major' und 'minor' Kennungen von 'devices' können mit dem Befehl 'ls -la' angezeigt werden:

```
ls -la /dev/ttyS*
```

zeigt beispielsweise die Liste der seriellen Schnittstellen-'devices' an. 'Devices' kann man an dem vorangestellten 'c' in der Rechteanzeige erkennen. Die beiden Einträge hinter der Gruppeninformation sind die 'minor' und 'major' Kennung des 'devices'. Man sieht, daß alle seriellen Schnittstellen die gleiche 'major'-Nummer besitzen - also vom gleichen 'handler' verwaltet werden. Sie unterscheiden sich lediglich in ihren 'minor numbers', damit der 'handler' die einzelnen 'devices' auseinander halten kann.

1.2 Linux Kernelmodule



Wie in Kapitel 1.1 beschrieben, besitzt der Linux-Kernel für die verschiedenen 'device'-Typen die jeweiligen Behandlungsroutinen. Damit nicht bei jeder Änderung der Rechnerkonfiguration (Hinzufügen und Entfernen von 'device'-Typen) der Linux-Kernel neu compiliert werden muß, besteht die Möglichkeit, die verschiedenen 'handler' in sog. *Kernelmodule* auszulagern, die dem Kernel in Laufzeit dynamisch hinzugefügt oder entfernt werden können. Dies macht es z.Bsp. auch möglich, Hardware-Treiber auszutauschen, ohne den Rechner neu zu starten. Dieses Konzept ermöglicht es sehr einfach, Linux an eine gegebene Rechnerkonfiguration anzupassen indem nur die entsprechenden Kernel-Module ausgewählt werden.

Der hier implementierte Treiber für die Advantech PCL726 Controllerkarte ist ein solches Kernelmodul.

2 Installation

2.1 Kernelmodule benutzen

2.1.1 Benutzte Kernelmodule anzeigen

Zur Anzeige der momentan vom Kernel verwendeten Kernelmodule dient der Befehl

```
lsmod
```

Das Programm befindet sich unter Linux im Verzeichnis '/sbin' und kann mit normalen Benutzerrechten ausgeführt werden. Die Ausgabe ist im wesentlichen identisch mit dem Inhalt der logischen Datei '/proc/modules'. Für jede, dem Kernel bekannte 'major number' existiert weiterhin jeweils ein logisches Verzeichnis unter '/proc/' mit der 'major number' als Namen.

2.1.2 Kernelmodule hinzufügen

Kernelmodule sind normale Objektdateien im 'ELF'-Format - z.Bsp. erzeugt vom gcc. Sie können dem Kernel mit dem Befehl:

```
/sbin/insmod <objectfile>
```

in Laufzeit hinzugefügt werden. Hierzu werden allerdings 'root'-Rechte benötigt, da ja ansonsten jeder Benutzer auf diesem Weg direkten Zugriff auf die Rechnerhardware hätte. Als Argument wird lediglich der Dateiname des Kernelmodules benötigt. Zum Laden des Kernelmodules für den Advantech PCL726 Treiber dient entsprechend der Befehl:

```
/sbin/insmod pcl726.o
```

'pcl726.o' muß sich im aktuellen Verzeichnis oder in einem Standard-Suchpfad befinden.

2.1.3 Kernelmodule entfernen

Um ein bereits geladenes Kernelmodul wieder aus dem Kernel zu entfernen, dient der Befehl:

```
/sbin/rmmod <modulename>
```

Als Argument muß der Name des zu entfernenden Modules angegeben werden, der z.Bsp. von 'lsmod' angezeigt wird (nicht den Namen der Objektdatei!). Auch für diesen Befehl sind 'root'-Privilegien notwendig.

2.2 Automatisches Starten beim Bootprozeß

Damit die Kernelmodule nicht nach jedem Rechnerstart manuell geladen werden müssen, kann das Laden/Entfernen von kleinen Skripten erledigt werden, die automatisch aufgerufen werden, wenn der Rechner hoch- oder runterfährt. Die Skripte zum Laden und Entfernen des Kernelmodules sind:

```
load_pcl726
remove_pcl726
```

Damit die Skripte automatisch gestartet werden, müssen sie in das Verzeichnis

```
etc/init.d/rc3.d
```

kopiert und in

```
S63pcl726
K63pcl726
```

umbenannt werden. Alle Skripte innerhalb des Verzeichnisses 'rc3.d' werden beim Start bzw. Ende des 'runlevel 3' automatisch gestartet. Dateinamen mit einem beginnenden 'S' kennzeichnen Startskripte (werden beim Erreichen des 'runlevel 3' gestartet). Dateinamen mit einem beginnenden 'K' kennzeichnen Deinitialisierungsskripte (werden beim Verlassen des 'runlevel 3' gestartet). Die Nummer nach dem 'S' oder 'K' gibt die Startreihenfolge der Skripte vor.

2.3 Installation per Makefile

Das beiliegende Makefile kann das Kernelmodul mit dem Aufruf

```
make install
```

automatisch installieren. Hierfür werden 'root'-Rechte benötigt. Desweiteren legt das Makefile die benötigten 'devices' automatisch an. Die 'load'- und 'remove' Scripte werden nicht direkt in das 'rc3.d'-Verzeichnis kopiert, sondern nach '/lib/modules/pcl726/'. Vorn dort werden anschließend symbolische Links in das 'rc3.d'-Verzeichnis gelegt.

Der Treiber kann ebenfalls durch das Makefile deinstalliert werden. Dies kann durch den Aufruf

```
make remove
```

geschehen. Dabei werden alle bei der Installation getätigten Aktionen rückgängig gemacht. Hierfür werden ebenfalls 'root'-Rechte benötigt.

3 Benutzung in Anwendungsprogrammen

Zum Ansteuern der 6 analogen Ausgabekanäle des Controllers stehen 6 'devices' zur Verfügung - für jeden Kanal eines. Die Devices mit ihren Namen, 'major'- und 'minor'-Kennungen sind:

Name	'major'	'minor'
/dev/pcl_da0	253	0
/dev/pcl_da1	253	1
/dev/pcl_da2	253	2
/dev/pcl_da3	253	3
/dev/pcl_da4	253	4
/dev/pcl_da5	253	5

Die Namen sind prinzipiell egal - Beim Zugriff auf den Controller sind die 'major' und 'minor'-Kennungen entscheidend. Das beiliegende Makefile legt beim Aufruf von

```
make install
```

die 'devices' automatisch an. ('root'-Rechte notwendig!)

Alternativ können die 'devices' manuell mit dem Befehl

```
mknod
```

angelegt werden.

Durch das Beschreiben dieser 6 'devices' kann nun ein 'user level'-Programm auf den Controller zugreifen, ohne direkt mit der tatsächlichen Hardware 'in Berührung' zu kommen.

3.1 Shell-Skripte

Es ist sehr einfach möglich, den Controller mit Shell-Skripten zu programmieren. Hierzu kann einfach der gewünschte Ausgabewert (eine Zahl im Intervall [0...4095]) in das 'device' des entsprechenden Controller-Kanals geschrieben werden. Dies kann z.B. durch:

```
echo "2047" > /dev/pcl_da0
```

geschehen. Hier wird der Ausgabekanal #0 auf den mittleren Wert gesetzt (jeweils relativ zum per 'jumper' gewählten Spannungsbereich).

3.2 C-Programme

Das Steuern des Controllers per C-Programm erfolgt analog zur Programmierung des Controllers per Shell-Skript. Zunächt muß das entsprechende 'device' beispielsweise durch:

```
int pcl0_fh = open("/dev/pcl_da0",O_WRONLY);
```

für den schreibenden Zugriff geöffnet werden. Da der neue Ausgabewert als ASCII-String in das 'device' geschrieben werden muß, sollte ein Integer-Wert mittels:

```
/* Groesse des buffers muss fuer */
/* Integer-Zahl ausreichend sein */
/* min. 6 Stellen inkl. Null-    */
/* terminierung.                 */
unsigned char buffer[10];
unsigned int  value;
...
...
value = 2047;
```

```
...
...
sprintf(buffer,"%d",value);
```

in einen String gewandelt werden. Dieser String kann mittels:

```
write(pcl0_fh,buffer,strlen(buffer));
```

in das 'device' geschrieben werden.

4 Sicherheitsaspekte

Typischerweise wird der Controller zum Ansteuern einer Hardware verwendet. Dies macht den Zugriff auf die entsprechenden Controller-'devices' durch beliebige Nutzer zu einem Sicherheitsrisiko (je nach dem, welche Hardware durch den Controller gesteuert wird). Auf jeden Fall sollten die Nutzungsrechte für die Controller-'devices' auf eine Gruppe von berechtigten Personen eingeschränkt werden. Der Zugriff auf 'devices' kann genauso eingestellt werden, wie der Zugriff auf Dateien.

Ein Vorschlag, die Sicherheit des Systems zu wahren ist, eine Gruppe 'pcl726' anzulegen und nur die Personen in die Gruppe aufzunehmen, die den Zugriff auf den Controller tatsächlich benötigen. Die Zugriffsrechte und Gruppe der 'devices' sollte dann beispielsweise durch

```
chgrp pcl726 /dev/pcl_*
chmod 660 /dev/pcl_*
```

gesetzt werden. Wenn die einzelnen Ausgabekanäle für unterschiedliche Aufgaben genutzt werden, sollte die Gruppe pcl726 gegebenenfalls in Untergruppen für die Nutzung der einzelnen Kanäle unterteilt werden.

5 Implementation

Der PCL726-Treiber wurde unter und für Linux-Kernel-Version 2.4 programmiert. Das Kernelmodul stellt die Funktionen 'device_open', 'device_write' und 'device_close' zur Verfügung. Die Devices werden bei der Benutzung durch ein Programm nicht(!) für andere Programme gesperrt. Somit können auch mehrere

Programme 'gleichzeitig' auf die Ausgabekanäle schreiben. Die 'minor' Kennungen der 'devices' werden direkt zum Bestimmen der Hardware-Register-Adressen verwendet. Es sind nur 'minor numbers' zwischen 0 und 5 zulässig.

Die Funktionen 'device_release' und 'device_open' haben keine Funktion, da geöffnete 'devices' ohnehin nicht für andere Prozesse gesperrt werden. Jedoch verlangt der Linux-Kernel diese Funktionen, damit korrekt auf die 'devices' zugegriffen werden kann.

Der schreibende Zugriff auf die Hardware-Register wird mit der Funktion 'outb' vorgenommen. Ein Setzen eines Ausgabewertes besteht aus zwei Schreibzugriffen für 'hi'- und 'lowbyte'.

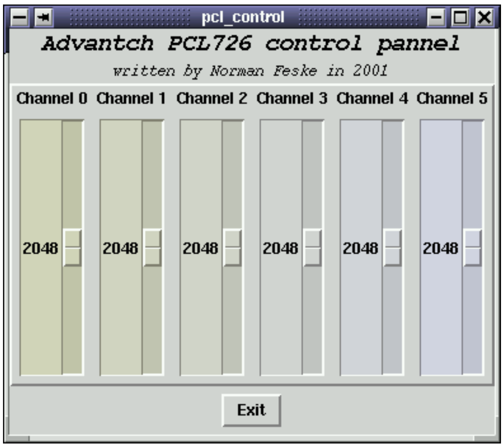
6 Praktische Anwendung - Objektivsteuerung

Für die konkrete Anwendung der Steuerung elektrisch ansteuerbarer Kameraobjektive wurde die Ausgangsspannung des Controllers direkt zum Betreiben der Motoren der Objektive verwendet. Ein Objektiv besitzt jeweils zwei Motoren: einen zum Einstellen der Brennweite und einen zum Einstellen des Zooms. Beide Motoren können mit Spannungen von -10V bis +10V betrieben werden, wobei die Größe des Betrages der Spannung die Geschwindigkeit bestimmt und die Polarität die Richtung. Somit lassen sich die Motoren der Objektive sehr flexibel ansteuern.

Die Advantech Controllerkarte wurde per 'Jumper' so konfiguriert, daß die Ausgangsspannung im Intervall [-10V...10V] liegt. Der 12-Bit-Zahlenwert bei der Ansteuerung eines Kanals liegt im Intervall [0...4095] und wird entsprechend auf das Ausgangsspannungsintervall linear abgebildet. Somit beträgt die Spannung beim Wert '2047' genau 0V.

Da zwei Objektive unabhängig voneinander angesteuert werden sollen, sind vier Ausgabekanäle des Controllers notwendig.

6.1 Testprogramm: pcl_control



Mit Hilfe des Testprogrammes 'pcl_control' können die Ausgabespannungen für jeden Kanal individuell eingestellt werden. Das Programm ist ein Tcl/Tk-Skript, welches wiederum mit den Tcl-Kommandos 'open', 'puts' und 'close' auf die Ausgabe-'devices' zugreift.