Application Note 072

Betrieb von SORCUS-Karten unter GNU/Linux

Autor: CB Datei: `an072b` (6 Seiten)

Inhaltsverzeichnis

1 X-Bus Systeme

1.1 MAX6pci

Für das Ansprechen von MAX6pci Trägerkarten[1] unter GNU/Linux gibt es einen Treiber, der als Kernel-Modul ausgeführt ist. Im Lieferumfang befinden sich Treiber, die speziell für Versionen[2] 2.4.3 bis 2.4.22 des Linux-Kernels kompiliert wurden.

1.1.1 Installation

Entpacken Sie die Datei `maxlindr.tgz` (`tar xvfz maxlindr.tgz`). Beim Entpacken wird das Verzeichnis `maxdrv` angelegt. Wechseln Sie in dieses Verzeichnis und rufen Sie dort `./maxdrvsetup` auf. Das Skript `maxdrvsetup` ermittelt die Versionsnummer des gerade laufenden Kernels und installiert das dazu passende Treibermodul `max6pci.o` bzw. `max6pci_smp.o`. Zum einfachen Laden des Treibers wird das Skript `max_ins` in das Verzeichnis `/usr/sbin` kopiert. Um den Zugriff auf den Treiber zu regeln wird versucht die Benutzergruppe `sorcus` anzulegen. `max_ins` vergibt später entsprechende Zugriffsrechte an diese Gruppe. Der Treiber sollte immer durch Aufruf von `max_ins` geladen werden, da sonst evtl. keine entsprechenden Einträge im Verzeichnis `/dev` angelegt werden. Das Skript `max_ins` ermittelt auch, ob es sich um ein System mit mehreren CPUs handelt und lädt in diesem Falle das Modul `max6pci_smp.o`.

[1] Der Treiber dient nur zum Ansprechen der Trägerkarte. Für die einzelnen Module müssen separate Treiber verwendet werden.
[2] weitere Versionen ($\geq$ 2.4.0) können auf Anfrage unterstützt werden

1.1.2 Konfiguration

Zur Zeit kann keine Konfiguration vorgenommen werden. Die erkannten Karten werden von 0 ab durchnummeriert. Über die SORCUS-Bibliothek werden sie unter dieser Nummer angesprochen.

1.1.3 Verwendung

Zum Starten des Treibers muß das Skript `max_ins` ausgeführt werden. Dieses lädt das Treibermodul und legt entsprechende Dateien im Verzeichnis `/dev` an.

1.2 Seriell angekoppelte Systeme

Seriell angekoppelte Systeme werden direkt, ohne spezielle Treiber, aus der SORCUS-Bibliothek heraus angesprochen. Dies bedeutet allerdings, dass keine konkurrierenden Zugriffe möglich sind.

1.2.1 Konfiguration

Standardmäßig sind bei der SORCUS-Bibliothek die Kartennummern 8 und 9 für serielle Ankopplung über `/dev/ttyS0` bzw. `/dev/ttyS1` vorkonfiguriert.

1.3 Die Bibliothek

Zum Ansprechen von X-Bus Systemen wird eine C-Bibliothek mitgeliefert. Diese wickelt die Zugriffe auf das Kernel-Modul bzw. die serielle Schnittstelle ab.

1.3.1 Installation

Entpacken Sie die Datei `libmax.tgz`. Wechseln Sie in das entstandene Verzeichnis `libmax-VERSIONSNUMMER`. Geben Sie den Befehl `libmaxsetup` ein. Die Bibliotheken werden nun nach `/usr/local/lib` kopiert und das Programm `ldconfig` aufgerufen, damit diese dem System bekannt werden (stellen Sie sicher, dass in der Datei `/etc/ld.so.conf` der Pfad `/usr/local/lib` eingetragen ist). Die zugehörigen C-Header werden nach `/usr/local/include/sorcus` kopiert. Die zum Ansprechen der Module nötigen MDDs[3] werden nach `/usr/local/lib/sorcus/mdd` kopiert.

1.3.2 Konfiguration

Bei der Installation wird die Datei `/etc/sorcus/boards.xml` angelegt. Standardmäßig sind die Kartennummern 0-7 für PCI-Karten (MAX6pci) und die Nummern 8 und 9 für serielle Ankopplung (MAX5DiP, MAX8DiP und X-Kit-3) vorkonfiguriert. Die Datei `boards.xml` sollte nicht von Hand bearbeitet werden. Ein grafisches Konfigurationsprogramm ist in Entwicklung.

1.3.3 Verwendung

Zur Verwendung der im SORCUS-Handbuch beschriebenen Funktionen in eigenen C-Programmen muß die Header-Datei `max_lib.h` in das Programm eingebunden werden. Zusätzlich muß die Bibliothek `libmaxw32`[4] zum Programm hinzugebunden werden. Da die Header-Dateien auch zur Entwicklung unter anderen Betriebssystemen verwendet werden können, muss das Makro `PLATFORM_LINUX` definiert sein. Zur Kompilierung eines C-Programms muss also folgendes Kommando eingegeben werden:

```
gcc -I/usr/local/include/sorcus -DPLATFORM_LINUX -o... -lmaxw32 ...
```

Zur automatischen Bestimmung der Kompiler-Parameter steht das Skript `libmax-config` zur Verfügung. Mit den Optionen `--cflags` und `--libs` können die nötigen Parameter für Präprozessor bzw. Linker abgefragt werden (z.B. zur Verwendung in Makefiles oder mit automake und autoconf). Ein minimales Makefile könnte damit so aussehen:

[3] Z.Z. werden nur Treiber für OsX unterstützt, d.h. es muß ein CPU-Modul auf der Karte vorhanden sein. Treiber für Module auf Karten ohne CPU befinden sich in der Entwicklung.

[4] Der Name wurde von der Windows-Version übernommen, um unter wine (siehe Abschnitt 3) kompatibel zu sein

```
CFLAGS=$(shell libmax-config --cflags)
LDFLAGS=$(shell libmax-config --libs)

example: example.c
```

2 MLx-Karten ('Alte Welt')

2.1 Der Treiber

Der Treiber ist als Kernel-Modul ausgeführt. Zum Einbinden des Treibers in das System ist daher eine Neukompilierung des Linux-Kernels nicht notwendig. Im Lieferumfang befinden sich Treiber, die speziell für Versionen[5] 2.4.3 bis 2.4.22 des Linux-Kernels kompiliert wurden, jeweils in einer Version für Ein-Prozessor- und Multi-Prozessor-Systeme.
Es werden folgende Karten unterstützt:

- "große" MODULAR-4/486 (ML8)
- "kleine" MODULAR-4/486 (ML7)
- Multi-COM (ML6)
- Multi-LAB/2 (ML2)

2.1.1 Installation

Entpacken Sie die Datei `mlxlindr.tgz` (`tar xvfz mlxlindr.tgz`). Beim Entpacken wird das Verzeichnis `mlxdrv` angelegt. Wechseln Sie in dieses Verzeichnis und rufen Sie dort `./mlxdrvsetup` auf. Das Skript `mlxdrvsetup` ermittelt die Versionsnummer des gerade laufenden Kernels und installiert die dazu passenden Treibermodule `mlx.o` (für Systeme mit einer CPU) und `mlx_smp.o` (für Systeme mit mehreren CPUs). Außerdem wird das Verzeichnis `/etc/sorcus` angelegt und die Datei `mlx.ini` dorthin kopiert. Diese Datei wird von dem Skript `mlx_ins`, das nach `/usr/sbin` kopiert wird, benötigt. Um den Zugriff auf den Treiber zu regeln wird versucht die Benutzergruppe `sorcus` anzulegen. `mlx_ins` vergibt später entsprechende Zugriffsrechte an diese Gruppe.

2.1.2 Konfiguration

Die Konfiguration der SORCUS-Karten wird in der Datei `/etc/sorcus/mlx.ini` eingetragen. Jede Definition für eine Karte muß von einer Zeile, die nur das Zeichen '#' als erstes Zeichen enthält eingeleitet werden. Danach folgen die Schlüsselwörter 'card' (laufende Nummer der Karte: 0..7), 'type' (Typ der Karte ML8=8, ML7=7, ML6=6, ML2=2), 'base' (Basisadresse der Karte in Hexadezimal) und 'irq' (IRQ-Nummer der Karte). Die zu den Schlüsselwörtern gehörenden Werte werden durch ein Leerzeichen getrennt angefügt.

Beispiel:

```
#
card 1
type 8
base 380
irq 11
#
card 2
type 7
...
```

Für die einzelnen Kartentypen gilt folgendes:

- "große" MODULAR-4/486 (ML8): Auf dieser Karte werden sowohl IO-Adresse als auch IRQ-Nummer per Jumper eingestellt. Die Kartennummer kann frei gewählt werden.

[5] weitere Versionen ($\geq$ 2.2.13) können auf Anfrage unterstützt werden

- Multi-COM (ML6) und ”kleine” MODULAR-4/486 (ML7): Auf diesen Karten wird die IO-Adresse per Drehschalter eingestellt. Der IRQ und die Kartennummer können frei gewählt werden.
- Multi-LAB/2 (ML2): Auf dieser Karte wird die Kartennummer per Drehschalter eingestellt. IO-Adresse und IRQ können frei gewählt werden.

2.1.3 Verwendung

Um den Treiber in das System einzubinden, muß das Kommando `mlx_ins` aufgerufen werden. Durch dieses Kommando wird der Treiber für die in der Datei `/etc/sorcus/mlx.ini` angegeben Karten geladen und im Verzeichnis `/dev` entsprechende Gerätedateien angelegt. Mit `rmmod mlx` bzw. `rmmod mlx_smp` bei Multi-Prozessor-Systemen kann der Treiber wieder aus dem System entfernt werden.

2.2 Die Bibliothek

2.2.1 Installation

Entpacken Sie die Datei `libmlx.tgz`. Wechseln Sie in das entstandene Verzeichnis `libmlx-VERSIONS-NUMMER`. Geben Sie den Befehl `make` ein. Die Bibliotheken werden nun nach `/usr/local/lib` kopiert und das Programm `ldconfig` aufgerufen, damit diese dem System bekannt werden (stellen Sie sicher, dass in der Datei `/etc/ld.so.conf` der Pfad `/usr/local/lib` eingetragen ist). Die zugehörigen C-Header werden nach `/usr/local/include/sorcus` kopiert.

2.2.2 Verwendung

Zur Verwendung der im SORCUS-Handbuch beschriebenen Funktionen in eigenen C-Programmen muß die Header-Datei mlx.h in das Programm eingebunden werden. Zusätzlich muß die Bibliothek libmlx zum Programm hinzugebunden werden. Zur Kompilierung eines C-Programms ist also folgendes Kommando einzugeben:

```
gcc -I/usr/local/include/sorcus -o... -lmlx ....c
```

Da die Interruptverarbeitung der SORCUS-libmlx-Bibliothek auf `libpthread` aufsetzt und dies wegen Inkompatibilitäten mitunder zu Problemen führen kann, ist im Lieferumfang auch eine Version der SORCUS-Bibliothek (`libmlx_sthr`) enthalten, in der die Interruptverarbeitung mittels eines Signal-Handlers realisiert ist. Bei Verwendung dieser Version der Bibliothek kann allerdings erheblich schlechter auf Interrupts reagiert werden, sie sollte daher nur dann verwendet werden, wenn die `libpthread` nicht verwendet werden kann. Zur Kompilierung eines Programms unter Verwendung der `libmlx_sthr` muß das Kommando
`gcc -I/usr/local/include/sorcus -o... lmlx_sthr ....c` eingegeben werden. Die Bibliotheken `libmlx` und `libmlx_sthr` stehen jeweils in einer Version für statisches (`libmlx.a` und `libmlx_sthr.a`) und für dynamisches (`libmlx.so` und `libmlx_sthr.so`) Linken zur Verfügung.

3 Das Programm SNW32

Das Windows-Programm SNW32 kann mittels wine auch unter GNU/Linux verwendet werden. wine muss dafür so konfiguriert werden, dass Zugriffe auf die DLLs `MAXW32.DLL` und `MLXW32.DLL` nach `libmaxw32.so` bzw. `libmlx.so` umgeleitet werden. Zusätzlich müssen die original Windows-DLLs `comctl32.dll`, `commctrl.dll` und `msvcrt.dll` verwendet werden. Folgende Schritte sind zur Installation von SNW32 unter wine nötig:

1. wine installieren: Die meisten Linux-Distributionen enthalten fertige Binärpakete. Neuere Versionen von wine unterstützen jedoch keine nativen ELF-Bibliotheken mehr (DLL-Typ 'so'), was aber zur Benutzung von SNW32 notwendig ist. Die Version `20030115`[6] unterstützt noch den Typ 'so' und wurde erfolgreich mit SNW32 getestet.

2. wine konfigurieren: Zur Konfiguration von wine empfiehlt sich das Programm WineSetupTk. Bei Verwendung von WineSetupTk sollte mit der Option *'Create a new windows directory'* eine neue 'Windows-Installation' angelegt werden. Hierdurch wird das Verzeichnis `~/.wine/fake_windows` erzeugt. Es sollten keine weiteren Einstellungen vorgenommen werden und WineSetupTk zunächst beendet werden.

3. Einbinden der SORCUS-Bibliotheken: Hierzu müssen zunächst Platzhalter erzeugt werden. Dies kann z.B. einfach durch folgende Befehle in einer Textkonsole erreicht werden:

   ```
   sorcus> touch ~/.wine/fake_windows/Windows/System/maxw32.dll
   sorcus> touch ~/.wine/fake_windows/Windows/System/mlxw32.dll
   sorcus> touch ~/.wine/fake_windows/Windows/System/sorcinfo.dll
   ```

   Die original Windows-DLLs `comctl32.dll`, `commctrl.dll` und `msvcrt.dll` müssen nun nach `~/.wine/fake_windows` kopiert werden. Danach kann WineSetupTk erneut gestartet und die zusätzlichen DLLs konfiguriert werden (siehe Abb. 1).

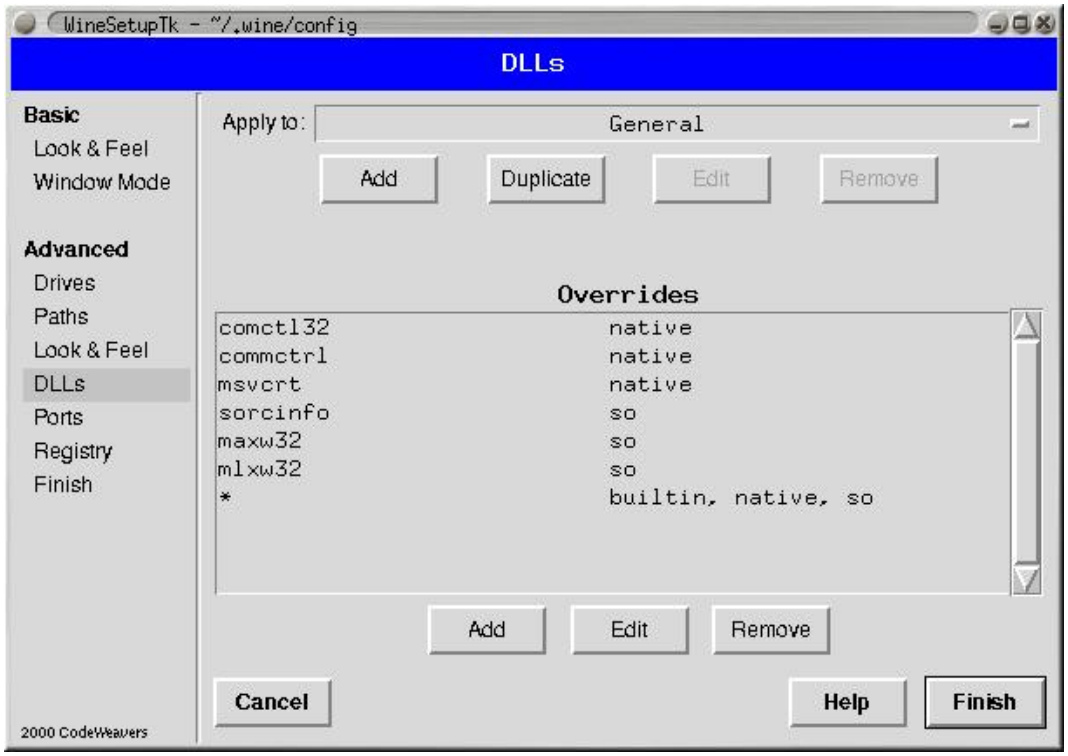

Abb. 1: Einstellungen zu den von wine verwendeten DLLs

[6] laut Auskunft eines Wine-Entwicklers ist die letzte Version, die 'so' unterstützt `20030219`

Historie des Dokuments

Datum	Autor	Bemerkung
05.12.03	CB	Löschen des _REENTRANT Flags bei libmlx
13.11.03	CB	Überarbeitung, Einfügen von X-Bus Systemen
06.04.00	RW	Erste Version, nur MLx-Karten