Dieses Dokument enthält Hinweise zur Treiberinstallation und zur Erstellung eigener Programme für die MAX6pci bzw. X-MAX-1 unter Verwendung der mitgelieferten Bibliotheken.

Inhaltsverzeichnis

1. Treiberinstallation unter Windows

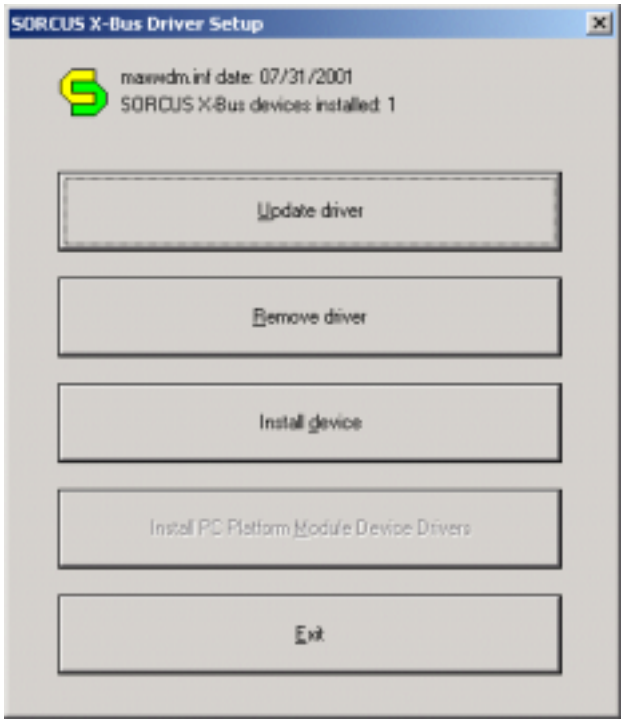
1.1. Allgemeines

Bevor Sie eine MAX-Karte unter Windows nutzen können, müssen Sie einen Treiber installieren. Der Treiber befindet sich in der Datei *maxsetup.zip* (diese kann von der SORCUS Hompage *www.sorcus.com* unter *Software\X-Bus* heruntergeladen werden) bzw. auf der SORCUS-CD im Verzeichnis *files\instwin*. Momentan werden die Windows Betriebssysteme 98, ME, NT 4.0 und 2000 unterstützt. Der Installationsvorgang unterscheidet sich dabei zwischen NT 4.0 und den anderen Betriebssystemen.

1.2. Treiberinstallation Windows NT 4.0

Um den Treiber unter Windows NT 4.0 zu installieren, führen Sie die folgenden Schritte aus:

- Starten Sie das Programm *setup.exe*.
- Drücken Sie nun die Schaltfläche *Install driver files*, um die Treiberdateien auf Ihren Rechner zu kopieren.
- Drücken Sie nun die Schaltfläche *Install device* um eine Karte zu installieren. Dabei wird der Karte eine Nummer zugewiesen, die bei der Programmierung zur Identifizierung der Karte verwendet wird.
- Wenn Sie PC basierte Modul-Device-Treiber (siehe Handbuch Kapitel *Modul-Device-Treiber*) verwenden möchten, dann drücken Sie die Schaltfläche *Install PC Platform Module Device Drivers*.



1.3. Treiberinstallation Windows 98, ME, 2000

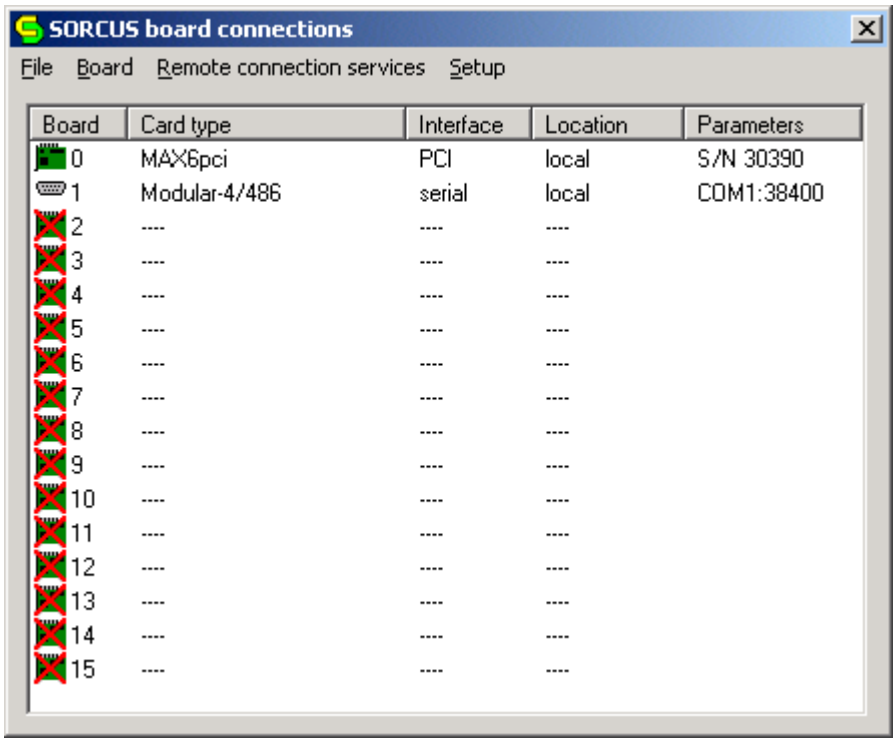
Beim Starten des Rechners erkennt Windows die eingesteckte SORCUS Karte selbstständig und fordert Sie auf, einen Datenträger mit dem Treiber anzugeben. Wählen Sie dazu das Verzeichnis *files\instwin* von der SORCUS-CD aus oder entpacken Sie die Datei *maxsetup.zip* und wählen Sie dieses Verzeichnis aus. Nun wird der Treiber samt aller PC basierten Modul-Device-Treiber (siehe Handbuch Kapitel *Modul-Device-Treiber*) installiert.

1.4. Treiber-Updates

Wenn Sie den SORCUS Treiber updaten möchten, dann starten Sie das Programm *setup.exe* und drücken die Schaltfläche *Update drivers*. Es werden nun alle installierten SORCUS Treiber (inclusive eventuell installierter PC basierter Modul-Device-Treiber) aktuallisiert.

1.5. Systemsteuerung

Nach der Treiberinstallation befindet sich in der Windows-Systemsteuerung ein *SORCUS boards* Symbol. Wenn Sie das Programm starten, wird Ihnen eine Übersicht aller installierten SORCUS-Karten angezeigt. Das Programm dient auch zur Einrichtung von Remote-Verbindungen zu Karten (seriell, Netzwerk).



2. Remote-Verbindungen

2.1. Allgemeines

Die SORCUS Remote-Verbindungen werden über einen Windows-DCOM-Server (mlxserv.exe) ausgeführt. Dieser erlaubt es momentan auf Karten über eine serielle Schnittstelle oder auf Karten in einem anderen PC über ein Netzwerk zuzugreifen. Soll über ein Netzwerk zugegriffen werden, so muss auf beiden Rechnern der Server installiert sein. Ist die Remote-Verbindung eingerichtet, stellt sich für den Programmierer/Anweder kein Unterschied zu einer lokalen Karte mehr dar.

2.2. Einrichten von Remote Verbindungen

Wenn Sie eine Remote-Verbindung (seriell, Netzwerk) zu einer SORCUS-Karte (z. B. zu einem X-Kit-3) einrichten möchten, gehen Sie folgenmaßen vor:

- Installieren Sie zuerst den Windows Treiber siehe *Treiberinstallation unter Windows*.
- Starten Sie das *SORCUS boards* Symbol in der Windows-Systemsteuerung.
- Installieren Sie nun den SORCUS DCOM-Server indem Sie aus dem Menü *Remote connection services* den Menüpunkt *Install mlxserv.exe* auswählen.
- Jetzt können Sie in der Kartenliste mit der rechten Maustaste auf eine freie Karte klicken und aus dem Kontextmenü *New remote connection* auswählen.
- Die nachfolgenden Dialoge ermöglichen Ihnen dann eine entsprechende Verbindung einzurichten.
- Nach diesem Vorgang erscheint die neu eingerichtet Karte dann in der Kartenliste.

2.3. Entfernen von Remote Verbindungen

Zum Entfernen einer Remote-Verbindung klicken Sie mit der rechten Maustaste auf die zu löschende Verbindung und wählen aus dem Kontextmenü *Delete*. Die Verbindung wird nun aus der Kartenliste gelöscht. Falls keine Remote-Verbindungen mehr existieren, können Sie auch den DCOM-Server aus ihrem System entfernen. Wählen Sie dazu aus dem Menü *Remote connection services* den Menüpunkt *Remove mlxserv.exe* aus.

3. PC-Programmierung

Die für das Erstellen eigener Programme benötigten Dateien stehen in der Datei maxlib.zip. In den darunter liegenden Unterverzeichnissen PC und COMMON sind alle Dateien für die Programmierung eigener Anwendungen in den Programmiersprachen C, Delphi und Visual Basic enthalten.

3.1. Programmiersprache C/C++

Für die Programmierung in C/C++ werden die Compiler *Microsoft Visual C* (ab Version 4.0), Borland C++ (ab Version 5.0) und Borland C++ Builder 5.5 direkt unterstützt.

Für diese Compiler werden Importbibliotheken und Header-Dateien als Schnittstelle zur Bibliothek *MAXW32.DLL* (wird bei der Treiberinstallation automatisch in das Windows/System32 Verzeichnis kopiert) mitgeliefert.

Die Importbibliothek-Datei *MAXW32.LIB* aus dem Verzeichnis *PC\WIN32\BC5* (Borland C++), *PC\WIN32\MSVC* (Microsoft Visual C) bzw. PC\WIN32\BCB\BCB55 (Borland C++ Builder) ist zum Projekt hinzuzulinken.

Andere Versionen des Borland C++ Builders erfordern andere Importbibliotheken. Diese können Sie sich mit dem Tool IMPLIB.EXE im BIN Verzeichnis des Borland Builders selber erzeugen:

implib maxw32.lib maxw32.dll

Dieser Vorgang ist nur einmalig für jede Version der MAXW32.DLL erforderlich.

Die Header-Datei *MAX_LIB.H* mit den Prototypen der Bibliotheksfunktionen aus dem Verzeichnis *COMMON\C* ist in jedem Fall per #include einzubinden. Diese bindet automatisch die Header-Dateien *X_DEFS.H* und *X_TYPES.H* ein. Diese enthalten Konstantendefinitionen bzw. Definitionen von in den Bibliotheksfunktionen verwendeten Datentypen.

Bei Verwendung der Modul-Device-Treiber ist zusätzlich die Datei *X_MDD.H* per #include einzubinden.

Möchten Sie in Ihrem Programm die für Fehlermeldungen definierten Konstanten verwenden, ist zusätzlich die Datei *X_ERROR.H* einzubinden.

Da die MAXW32.DLL aus mehreren Threads besteht, ist es erforderlich, dass ein Programm, das auf die Bibliothek zugreift, Multithreading unterstützt. Diese Eigenschaft wird in den Compiler-Optionen angegeben.

3.2. Programmiersprache Delphi

Für die Programmierung in Borland-Delphi werden die Compiler ab Version 2.0 unterstützt.

Es werden keine compilierten DCU-Dateien mitgeliefert. Statt dessen müssen die .PAS-Sourcefiles aus dem Verzeichnis *COMMON\PAS* per uses-Anweisung in das Projekt eingebunden werden.

Die Dateien *MAXLIB.PAS*, *MAXDEFS.PAS* und *MAXTYPE.PAS* müssen in jedem Fall eingebunden werden, die Dateien *MAXMDD.PAS* nur bei Verwendung der Modul-Device-

Treiber-Funktionen und die Datei *X_ERROR.PAS* nur bei Verwendung der für Fehlermeldungen definierten Konstanten.

3.3. Programmiersprache Visual Basic

Für das Ansprechen der X-Bus-Bibliothek *MAXW32.DLL* aus Visual Basic Programmen stehen folgende Importbibliotheken zur Verfügung:

- *max_lib.bas* enthält die Deklaration von Funktionen und Typen,
- *max_defs.bas* enthält die verwendeten Konstanten
- *max_mdd.bas* enthält die Konstanten und CPS-Typ-Deklarationen für die Verwendung der Modul-Device-Treiber
- *max_err.bas* enthält die definierten Fehler-Konstanten

Diese Dateien müssen als *Module* einem Visual Basic Projekt hinzugefügt werden.

Einige Datentypen wie z.B. *MAX_ERROR*, *MAX_VERSION* sowie alle MAX-Handle-Typen wurden in Visual Basic nicht eigens definiert. Dafür werden die Standard Typen *INTEGER* bzw. *LONG* verwendet. Im einzelnen geht das aus den mitgelieferten Basic-Bibliotheken hervor.

Bei Konstanten, die in den anderen Sprachen und im Handbuch mit einem Unterstrich beginnen, entfällt der Unterstrich in der Visual Basic Definition, da dieser nicht zulässig ist.

Einige in der DLL verwendete Datentypen sind auf Word-Alignment angewiesen. Wenn eine Variable eines solchen Typs an eine DLL-Funktion übergeben wird, erwartet die DLL alle Elemente der Struktur im Speicher direkt hintereinander stehend. Visual Basic richtet dagegen die Struktur-Elemente an 4-Byte-Adressen aus. Wenn z.B. ein Integer und ein Long Element hintereinander in einer Datenstruktur stehen, fügt Visual Basic zwischen beiden Elementen einen Integer Dummy-Wert ein.

Aus diesem Grund sind die Datenstrukturen *MAX_OS_INFO*, *MAX_TI_TYPE*, *MAX_BOARD_TYPE*, *MAX_CHANNEL_TRANSFER_TYPE* und *MAX_FLASH_TYPE* sowie einige CPS-Strukturen in Basic anders definiert als im Handbuch. Der Typ Long wurde in diesen Strukturen durch MAX_LONG ersetzt. Dieser Typ ist nur in Visual Basic definiert. Zur Umwandlung von MAX_LONG in Long ist im Modul Max_lib.bas folgende Funktion enthalten:

```
Public Function ConvertMaxLongToLong (ByRef prcMaxLong As MAX_LONG) As Long
```

Zur Umwandlung von Long- nach MAX_LONG steht die folgende Subroutine zur Verfügung:

```
Public Sub ConvertLongToMaxLong (ByVal ulLongValue As Long,
                                 ByRef prcMaxLong As MAX_LONG)
```

Die Funktion *max_set_service* ist unter Visual Basic nicht implementiert, da der Aufruf von Callback-Funktionen unter den derzeitigen Visual Basic Versionen (inkl. 6.0) nicht funktioniert, sofern der Aufruf aus einem anderen Thread heraus erfolgt. Dieses ist im Fall der Callback-Funktion für Service-Requests der Fall. Aus dem selben Grund muß der Parameter *pCbFunc* beim Aufruf der Funktion *max_open_channel* in Visual Basic-Programmen =NULL gesetzt werden.

Die Visual Basic Versionen 4 bis 6 werden unterstützt.

3.4. Verwendung einer MAX6pci ohne aufgestecktes CPU-Modul

Die Programmierung einer Karte ohne aufgestecktes X-MAX-1 Modul ist prinzipiell identisch zum vorherigen. Dadurch, dass viele der im Handbuch beschriebenen Funktionen sich direkt auf ein X-MAX-1 bzw. das darauf laufende OsX-Betriebssystem beziehen, steht aber natürlich nur ein eingeschränkter Funktionsumfang zur Verfügung. Er umfaßt im wesentlichen die Modul-Device-Treiber-Zugriffsfunktionen für das Ansprechen der aufsteckenden I/O-Module.

Folgende Bibliotheksfunktionen stehen zur Verfügung:

Initialisierung:

- max_init_lib und max_exit_lib
- max_reset_board
- max_board_reacting
- max_set_timeout
- max_set_board_led
- max_connect_cpu (dabei ist slot=layer=0 zu setzen)
- max_exit_cpu

Version und andere Informationen abfragen

- max_get_drv_version, max_get_lib_version, max_get_version_string
- max_get_board_info, max_get_module_info und max_scan_boards

Zugriff auf Modul-EEPROMs

- max_read_eeprom und max_write_eeprom

Modul-Device-Treiber-Zugriffe

- alle Funktionen aus Kapitel 6.7

Fehlerbehandlung

- alle Funktionen aus Kapitel 6.18

4. Echtzeitprogrammierung unter OsX

Die für das Erstellen eigener Programme benötigten Dateien stehen in der Datei maxlib.zip. In den darunter liegenden Unterverzeichnissen RT und COMMON sind alle Dateien für die Programmierung eigener Anwendungen in den Programmiersprachen C, und Pascal enthalten.

In allen Projekten zur Erstellung eines Echtzeitprogrammes muß immer die Konstante *_RTLIB_* definiert sein.

4.1. Programmiersprache C/C++

Für die Programmierung in C/C++ werden die Borland C++ Compiler (Version 3.1, 4.5 und 5.0) unterstützt.

Die Bibliotheks-Datei *MAXRT.LIB* aus dem entsprechenden Verzeichnis *RT\BC31* (für Borland C 3.1 ohne Gleitkomma-Unterstützung), *RT\BC31_EMU* (für Borland C 3.1 mit Gleitkomma-Emulation) bzw. *RT\BC45* (für Borland C 4.5 und 5.0) ist zum Projekt hinzuzulinken.

Zusätzlich stehen in den *RT\BCxx*-Verzeichnissen die für das OsX-Betriebssystem angepassten Startup-Code-Objektdateien. Damit tatsächlich diese Startup-Codes anstelle der im Borland C enthaltenen in das Projekt eingebunden werden, muss unter *Option/Directories/Library-Directories* die Pfadangabe *RT\BCxx* vor derjenigen zu den Original Borland Bibliotheken stehen.

Die Header-Datei *MAX_LIB.H* mit den Prototypen der Funktionen aus dem Verzeichnis *COMMON\C* ist in jedem Fall per #include einzubinden. Es bindet automatisch die Header-Files *X_DEFS.H* und *X_TYPES.H* ein. Diese enthalten Konstantendefinitionen bzw. Definitionen von in den Bibliotheksfunktionen verwendeten Datentypen.

Bei Verwendung der Modul-Device-Treiber ist zusätzlich die Datei *X_MDD.H* per #include einzubinden.

Möchten Sie in Ihrem Programm die für Fehlermeldungen definierten Konstanten verwenden, ist zusätzlich die Datei *X_ERROR.H* per #include einzubinden.

4.2. Programmiersprache Pascal

Für die Programmierung in Pascal wird der Borland-Pascal 7.0 Compiler unterstützt.

Im Verzeichnis *RT\BP7* befinden sich alle benötigten Unit-Dateien (TPUs). Diese wurden aus den Quellcode-Dateien im Verzeichnis *COMMON\PAS* erzeugt. Sie müssen per uses-Anweisung in das Projekt eingebunden werden. Die Datei *MAXMDD.TPU* braucht nur bei Verwendung der Modul-Device-Treiber-Funktionen und die Datei *X_ERROR.TPU* nur bei Verwendung der für Fehlermeldungen definierten Konstanten eingebunden werden.

Zusätzlich steht im Verzeichnis *RT\BP7* die für das OsX-Betriebssystem angepasste *SYSTEM.TPU*. Damit tatsächlich diese TPU anstelle der im Borland Pascal enthaltenen in das Projekt eingebunden wird, muß unter *Option/Verzeichnisse/Unit-Verzeichnisse* die Pfadangabe *RT\BP7* vor derjenigen zu den Borland TPUs stehen.

4.3. Historie

Datum	Bemerkung
10.03.2003	Ergänzender Hinweis zum Borland C++ Builder Historie hinzugefügt