*X-Bus Intensivseminar*

# Das Modul-Device-Treiber (MDD) Konzept

SORCUS Computer GmbH
Dipl.-Ing. Jürgen Schäfer


SORCUS

# Inhalt

- Begriffserklärung
- Grundlagen
- Dienste eines Kanals
- Zugriff über MDDs auf ein Modul
- Laden eines MDDs
- Öffnen eines Kanals
- Zugriff auf Kanal-Dienste
- Beispiel
- Callback-Mechanismus
- Vorteile der MDDs
- Weitere Informationen

# Begriffserklärung

- MDD
  - Module-Device-Treiber. Treiber-Programm (für OsX/Windows), das den Zugriff auf Module/Karten ermöglicht
- MDD-Handle
  - wird von der Bibliothek beim Laden eines MDDs vergeben. Eindeutiger 32 Bit Wert, mit dem auf den MDD zugegriffen werden kann (z. B. um den MDD zu resetten oder Kanäle zu öffnen)
- Kanal-Handle
  - wird von der Bibliothek beim Öffnen eines Kanals vergeben. Eindeutiger 32 Bit Wert, mit dem auf den Kanal zugegriffen werden kann.
  - Handles sind Prozessspezifisch (dürfen nicht an andere Prozesse weitergegeben werden).

# Begriffserklärung

- Device
  - Funktionseinheit eines Moduls (z. B. digitaler Eingang, analoger Ausgang, ...)
- CPS
  - Channel Property Structure. Enthält Informationen über den zu öffnenden Kanal. Für jeden MDD ist eine eigene CPS definiert.
- Kanal-Dienste
  - verfügbare Funktionen eines Kanals

# Grundlagen

- Zugriff auf Funktionseinheiten der I/O-Module erfolgt (ausschließlich) über Treiberprogramme, die sogenannten Modul-Device-Treiber (MDDs)
- MDDs sind verfügbar für:
  - den MAX-PC unter OsX (OsX-MDD)
  - den Host-PC unter Windows (PC-MDD) (als Kernel-Mode-Treiber)
- Die Kommunikation mit den Treibern erfolgt mit Hilfe von Bibliotheksfunktionen

# Grundlagen

- Die MDDs gehen von einem *Kanalorientierten* Ansatz aus
- Die Verbindung zu einem Device (Funktionseinheit eines Moduls z. B. analoger Eingang) wird als *Kanal* bezeichnet
- Der Vorgang des Verbindungsaufbaus wird als das *Öffnen des Kanals* bezeichnet
- Der Vorgang des Verbindungsabbaus wird als das *Schließen des Kanals* bezeichnet
- Nach dem Öffnen ist der Kanal in der Lage *Dienste* auszuführen

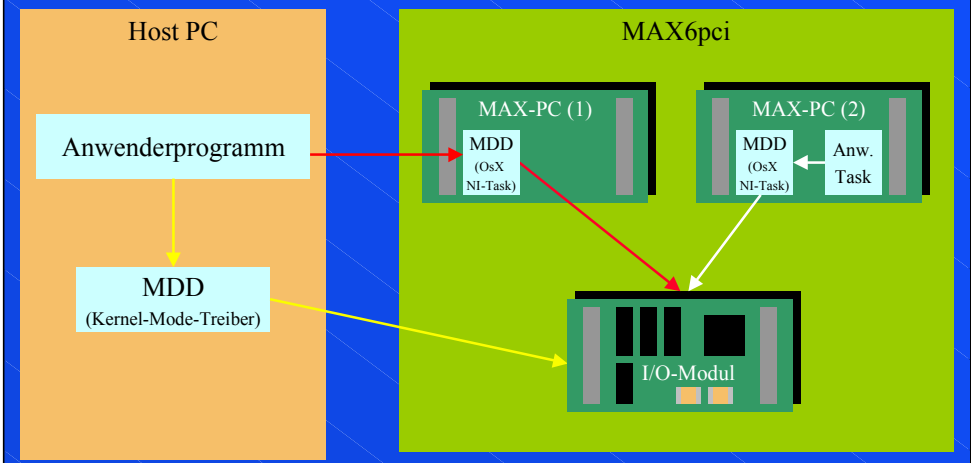
# Dienste eines Kanals

- Je nach geöffnetem Kanal, stellt dieser verschiedene Dienste (Funktionen) zur Verfügung:
  - Dateneingabedienst
    - wurde ein Kanal z. B. auf einen analogen Eingang geöffnet, stellt dieser einen Eingabedienst zum Einlesen der Werte zur Verfügung
  - Datenausgabedienst
    - wurde ein Kanal z. B. auf einen digitalen Ausgang geöffnet, stellt dieser einen Ausgabedienst zum Setzten des Ausgangs zur Verfügung
  - Sonderdienste
    - mit Hilfe der Sonderdienste kann ein Kanal gesteuert, parametriert oder diagnostiziert werden (es kann z. B. einem Kanal ein Name zugewiesen werden, oder die Richtung eines DIOs umgeschaltet werden)

# Zugriff über MDDs auf ein Modul

- Es können mehrere MDDs auf ein Modul zugreifen (z. B. PC MDD und OsX MDD)
- Vom Host-PC kann sowohl über einen OsX-MDD (sofern ein MAX-PC vorhanden ist) als auch über einen PC-MDD auf die Module zugegriffen werden
- Um Zugriffskonflikte zu vermeiden, werden die Zugriffe über Semaphoren (die sich auf dem Modul befinden) gesteuert
- Devices können beim Öffnen des Kanals auch als *exklusiv* gekennzeichnet werden (Dann kann kein weiterer Kanal mehr auf dieses Device geöffnet werden)

# Zugriff über MDDs auf ein Modul




SORCUS

# Laden eines MDDs

- MDDs können auf dem MAX-PC (unter OsX) oder dem Host-PC (unter Windows) laufen
- MDD soll auf dem MAX-PC laufen
  - Ein Aufruf von *max_load_mdd* vom PC aus, installiert den MDD als NI-Task auf das gewählte CPU-Modul
  - Ein Echtzeitprogramm muss ebenfalls *max_load_mdd* aufrufen um sich mit dem MDD zu verbinden. Der MDD muss in diesem Fall schon auf dem MAX-PC installiert sein (vom PC oder Flash).
- MDD soll auf dem Host-PC laufen:
  - der MDD muss als Kernel-Mode-Treiber installiert werden:
    - Windows 98/2000/XP: bei der Treiberinstallation/Reset der Karte werden die MDDs automatisch vom PnP-Manager installiert
    - Windows NT 4.0: die MDDs müssen mit Hilfe des Treiber-Installations-Programms installiert werden

# Laden eines MDDs

- mit *max_load_mdd* kann jetzt eine Verbindung zum MDD aufgebaut werden

- Die Funktion *max_load_mdd* liefert ein *Handle* zurück, das bei jedem MDD-Zugriff angegeben werden muss
- Beim Laden des MDDs wird eine Versionsprüfung vorgenommen (passt der MDD zu dem Modul?).

# Öffnen eines Kanals

- Die Funktion *max_open_channel* öffnet einen Kanal zu einem Device
- Der Funktion müssen u. a. das Handle des MDDs und die Eigenschaften des Kanals übergeben werden
- Die Eigenschaften des Kanals werden in der sogenannten CPS (Channel Property Structure) festgelegt und hängen vom jeweiligen MDD ab (jedes Modul besitzt eine eigene Struktur)
- Konnte der Kanal fehlerfrei geöffnet werden, erhält man ein *Handle* auf den Kanal, mit dem man auf diesen zugreifen kann

# Zugriff auf Kanal-Dienste

- Wurde der Kanal erfolgreich geöffnet, kann man auf den Kanal zugreifen:
  - *max_read_channel*: (Eingabedienst) liest Daten eines Kanals (z. B. analoger Eingang)
  - *max_write_channel*: (Ausgabedienst) schreibt Daten zu einem Kanal (z. B. digitaler Ausgang)
  - *max_channel_control*: (Sonderdienst) sendet eine Steuerkommando an einen Kanal (z. B. Zähler starten)
  - *max_channel_info*: (Sonderdienst) liest Informationen eines Kanals
- Welche Dienste die Kanäle zur Verfügung stellen ist abhängig vom jeweiligen MDD

# Zugriff auf Kanal-Dienste

- Wird ein Kanal nicht mehr benötigt, sollte dieser mit *max_close_channel* geschlossen werden

# Beispiel

- Es soll mit dem Modul X-AD14-20 ein analoges Signal erfasst werden
- X-AD14-20:
  - 10 analoge Differenz-Eingänge
  - 20 analoge Massebezogene-Eingänge
  - verschiedene Eingangsbereiche
- Vorgehensweise:
  - Laden des MDDs
  - Ausfüllen der CPS (Channel Property Structure)
  - Öffnen des Kanals
  - Lesen von Werten

# Beispiel

```
// mit PC-CPU verbinden
MAXMODHND hCpuModule;
usCpuSlot = 0;
usCpuLayer = 0;
max_connect_cpu(usCard, usCpuSlot, usCpuLayer, &rcOsInfo, &hCpuModule);

// MDD auf dem PC laden
MAXMDDHND hMdd;
max_load_mdd(hCpuModule, usModuleSlot, usModuleLayer, 0, MDD_X_AD14_20, NULL,
    &hMdd);

// CPS für einen massebezogenen analogen Eingang ausfüllen
CPS_XAD1420 rcAin;
rcAin.usDevice = DEVICE_AIN_SE;     // massebezogener analoger Eingang
rcAin.usIndexFirst = 0;             // Eingang 0 des Moduls soll erfasst werden
rcAin.usIndexLast = 0;              // ist usIndexLast != usIndexFirst werden
                                    // mehrere Eingänge des Moduls erfasst
rcAin.usFlags = _CP_EXCLUSIVE;      // das Device wird exklusiv genutzt
rcAin.usReadMode = IO_MODE_DIRECT; // Lesemodus = direkt
rcAin.usRange = RANGE_BIP_10V;      // Messbereich: -10 ... +10V
rcAin.usSettleTime = 1000;          // Settle Time in ns
```

SORCUS

# Beispiel

```
// Kanal öffnen
MAXCHLHND hAin;
max_open_channel(hMdd, sizeof(rcAin), (void*)&rcAin, NULL, 0, &hAin);

// Werte aus dem Kanal lesen = Eingangssignal erfassen
max_read_channel_long(hAin, &lData);
...

// Kanal schließen
max_close_channel(&hAin);
```

# Callback-Mechanismus

- Kanäle können das Anwenderprogramm über aufgetretene Ereignisse (z. B. Zählerüberlauf des Zählermoduls) aktiv informieren
- Hierzu kann eine vom Anwender definierte Callback-Funktion beim Öffnen des Kanals angegeben werden
- Tritt das Ereignis ein, wird diese von der Bibliothek aufgerufen
- Welche MDD-Kanäle diese Funktionalität aufweisen steht in der MDD Dokumentation (ebenso welche Daten die aufgerufene Callback-Funktion übergeben bekommt)

# Vorteile der MDDs

- Alle Module werden auf die gleiche Art behandelt
- Zugriffssynchronisation
- Bei Änderungen der Hardware bleibt MDD kompatibel
- MDDs sind auf verschiedenen Plattformen verfügbar
- Zugriff auf allen Plattformen identisch
- weniger Speicherbedarf als Modul-Bibliotheken
- Versionskontrolle
- Normiertes Datenformat für analoge Ein- und Ausgänge

# Weitere Informationen

- MAX6pci Handbuch:
  - Kapitel: Bibliotheken
    - Funktionen für Modul-Device-Treiber-Kommunikation
    - Funktionen für den MDD Kanalzugriff
  - Kapitel: Modul-Device-Treiber
  - Kapitel: Modulbeschreibungen

# Ende