# Application Note AN095

# Verwendung der Modul-Device-Treiber in Assembler-Programmen

Autor: JD Datei: An095_B (4 Seiten)

Assembler-Programme für einen MAX-PC benötigen Kenntnis der Assembler-Schnittstelle der Modul-Device-Treiber (MDD), wenn auf andere Module zugegriffen wird, denn:

Zugriffe auf die Modul-Hardware erfolgen auch in Assembler **immer** über die MDDs!

## Öffnen von Kanälen in Assembler

Das Öffnen von MDD-Kanälen geschieht über den Aufruf der Funktion 8 der entsprechenden MDD-Task. Die Konvention zum Aufruf einer Funktion mit **call_func** ist in Application Note AN083 beschrieben. Die Anzahl der zur Funktion übergebenen Parameter entspricht der Größe der jeweiligen CPS, die Funktion soll 4 Byte zurückliefern (das ist die erhaltene Kennung, mit der anschließend auf den geöffneten Kanal zugegriffen wird). Zu beachten ist, dass die Adressen der Hin- und Rück-Parameter jeweils physikalisch anzugeben sind.

## Verwendung von Kanälen, die mit max_open_channel geöffnet wurden

In Programmen, die aus Hochsprachen und Assembler-Teilen bestehen, gelten einige Besonderheiten.

Assembler-Aufrufe werden in der Regel zur Beschleunigung der Programm-Ausführung eingesetzt. Dazu werden die zeitkritischen Teile in Assembler bzw. inline-Assembler geschrieben. Zeitkritische Teile sind z.B. das Einlesen von Daten von einem Modul bzw. das Ausgeben von Daten an ein Modul sowie die damit in Zusammenhang stehenden Speicher- und Rechenoperationen. In den meisten Fällen ist nur Prozedur 0 einer Task zeitkritisch.

Alle Bibliotheksfunktion enthalten zwangsläufig einen Overhead gegenüber direkten Assembler-Aufrufen. Dieser ist im Verhältnis zur im MDD für die Zugriffe auf die Modul-Hardware benötigten Zeit klein, kann aber in sehr zeitkritischen Anwendungen durchaus eine Rolle spielen. In diesen Fällen ist der Einsatz von Assembler für den Zugriff auf die MDD-Kanäle sinnvoll. Die Funktionen für Speicherzugriffe und Taskbefehle in Assembler sind in der Application Note AN083 beschrieben.

Das Öffnen und Schließen von Kanälen ist demgegenüber nicht zeitkritisch (und in den Bibliotheken auch nicht optimiert) und kann mit der Bibliotheksfunktion gemacht werden. max_open_channel liefert ein Handle zurück, mit dem in Assembler-Programmen nicht direkt auf die MDD-Kanäle zugegriffen werden kann. Um das Handle in ein für Assembler-Aufrufe

verwendbares Format zu konvertieren, stellt die Echtzeitbibliothek die Funktion max_get_channel_id zur Verfügung:

## max_get_channel_id

Pascal FUNCTION max_get_channel_id (hChannel: MAXCHLHND;
VAR pulID: LONGWORD) : MAX_ERROR;

C MAX_ERROR max_get_channel_id (MAXCHLHND hChannel,
ULONG *pulID);

Funktion Die Funktion liefert zu einem mit max_open_channel erhaltenen MDD-Kanal-Handle eine Kennung zurück, mit der ein direkter Assembler-Aufruf der Dienste des Kanals möglich wird.

Parameter *hChannel* Kanal-Handle

*pulID* Zeiger auf eine Variable, in die die Funktion die Kennung des Kanals einträgt.

Die erhaltene Kennung darf nur für die Aufrufe der Kanal-Dienste und Sonderdienste verwendet werden (also als Ersatz der Bibliotheksfunktionen max_read_channel_xxx, max_write_channel_xxx, max_channel_info und max_channel_control). Der Aufruf führt aus Performance-Gründen keine Überprüfung der Übergabeparameter durch (entspricht einem Aufruf der entsprechenden Bibliotheksfunktion mit Error-Check-Level = 0).

### Aufrufkonvention in Assembler

Zunächst muss die Kanal-Kennung in es:bx geladen werden. Ggf. müssen weitere Register initialisiert werden. Anschließend wird die MDD Funktion über es:bx aufgerufen (die Unterscheidung zwischen Eingabe-, Ausgabe- und Sonderdienst erfolgt durch einen Offset, der zu bx zu addieren ist, s.u.).

Nach dem Abruf eines Dienstes sind die Register **eax**, **ecx** und **edx** geändert. Ebenfalls geändert ist das Flagregister, allerdings nicht das **Direction** Flag.

Tritt während der Ausführung eines Dienstes ein Fehler auf, so wird der Dienst mit **CY** = 1 und **ax** = Fehlercode beendet. Für den Aufrufenden bedeutet die Fehlermeldung, dass der Dienst keine Daten übernommen bzw. übergeben hat.

## Eingabedienst:

```
les     bx, [DWORD handle]      ; Laden der erhaltenen ID in es:bx
call    [DWORD FAR es:bx + 4]   ;Aufruf des Eingabedienstes
```

Für den Aufruf der Einzelwert-Dienste sind vor dem Aufruf keine weiteren Register zu setzen, vor dem Aufruf des Block-Eingabedienstes müssen folgende Register initialisiert werden:

cx = Anzahl zu lesender Byte

eax = Physikalischer Zeiger auf Ziel-Datenpuffer

Ist der Aufruf erfolgreich (CY=0), steht das Ergebnis in AL (UCHAR), AX (SHORT bzw. USHORT), EAX (LONG bzw. ULONG) oder beim Blockdienst in der Speicheradresse, die zuvor in EAX angegeben wurde. In diesem Fall liefert der Aufruf in AX zurück, wie viele Bytes tatsächlich gelesen wurden

## Ausgabedienst:

```
les     bx, [DWORD handle]      ; Laden der erhaltenen ID in es:bx
call    [DWORD FAR es:bx]       ;Aufruf des Ausgabedienstes
```

Vor dem Aufruf müssen folgende Register initialisiert werden:

eax = Einzelwert: in AL (UCHAR), AX (SHORT bzw. USHORT) bzw. EAX (LONG bzw. ULONG) steht der auszugebende Wert

Blockdienst: EAX = Physikalischer Zeiger auf Quell-Datenpuffer

cx = Anzahl zu lesender Byte (nur für Blockdienst)

Ist der Aufruf erfolgreich (CY=0), steht beim Blockdienst in AX anschließend, wie viele Bytes nicht übernommen wurden.

## Sonderdienst:

```
les     bx, [DWORD handle]      ; Laden der erhaltenen ID in es:bx
call    [DWORD FAR es:bx + 8]   ;Aufruf des Sonderdienstes
```

Vor dem Aufruf müssen folgende Register initialisiert werden:

dx = Kennung des Sonderdienstes

cx = Größe des Übergabeparameterblocks

eax = Physikalischer Zeiger auf Übergabe- bzw. Rückgabeparameter (falls cx != 0)

Die unterstützten Kennungen stehen in der jeweiligen Modul-Beschreibung bzw. im Handbuch unter den Funktion max_channel_control und max_channel_info.

## Weitere MDD-Funktionen

Alle weiteren MDD-Funktionen werden durch call_func Anweisungen in die MDD-Task durchgeführt. Die Tasknummer ist jeweils in dx einzutragen. Die Bedeutung der einzelnen Parameter ist der Beschreibung der entsprechenden Bibliotheksfunktion im Handbuch zu entnehmen. Die folgende Tabelle enthält die beim Aufruf der Funktionen benötigten Parameter.

| **Entsprechende Bibliotheksfunktion** | **Funk. Nr. (bx)** | **Eingangsdaten (Adresse in eax)** | **Anzahl Eingangs daten (in di)** | **Ausgangsdaten (Adresse in ecx)** | **Anzahl Ausgangs-daten (in si)** |
|---|---|---|---|---|---|
| max_open_channel | 8 | CPS | sizeof(CPS) | Kanal-Kennung | 4 |
| max_read_info_text | 9 | High Byte = page<br>Low Byte = line | 2 | text | size |
| max_reset_mdd | 3 | - | 0 | - | 0 |
| max_get_mdd_ version | 6 | - | 0 | version<br>date | 2 * ULONG |
| max_device_status | 12 | device | sizeof (MAX_ DEV_SPEC_TYP E) | status | 2 |

## Einschränkungen in Assembler

Eine Unterstützung von Callback-Kanälen (d.h. der Parameter *pCbFunc* bei **max_open_channel** ist 1= 0) ist in Assembler nicht möglich, da der verwendete Mechanismus in den Bibliotheken implementiert ist.

## Historie dieses Dokumentes

| **Datum** | **Autor** | **Änderung** |
|---|---|---|
| 24.06.05 | jd | Sonderdienst korrigiert: Kennung in dx |
| 11.11.02 | jd | neu |