Application Note AN075

Einsatz des Message-Modul-Device-Treibers

Autor: JD Datei: AN075.DOC (3 Seiten)

Eine SORCUS-Karte kann einen Host-PC nur durch das Verschicken eines Service-Requests aktiv benachrichtigen. Dabei wird nur ein Wort (wovon nur 14 Bit effektiv nutzbar sind) an den PC übertragen. Soll der Service-Request z.B. signalisieren, daß auf der Karte Messwerte bereitliegen, muß sich der PC diese im Anschluß an den Empfang des SRQs selber abholen. Maßnahmen zur Sicherstellung der Datenkonsistenz erfordern u.U. einen hohen Programmieraufwand auf Karten- und PC-Seite.

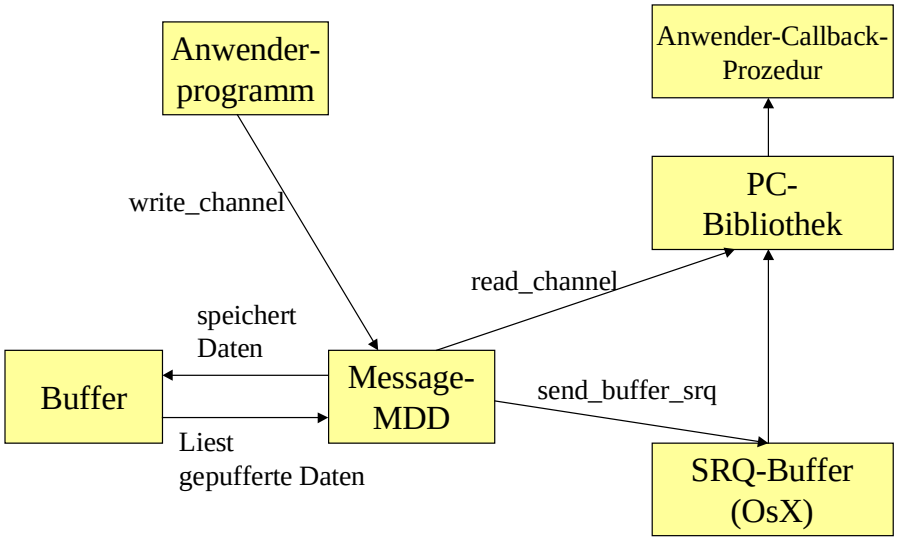
Der Message-Modul-Device-Treiber (Message-MDD) ist ein auf dem Modul-Device-Treiber-Konzept basierendes Treiberprogramm, welches auf der SORCUS-Karte läuft. In Zusammenarbeit mit den Bibliotheken vereinfacht er die von der Karte initiierte Übertragung von Daten zum PC. Mit dem Aufruf des Schreibdienstes eines Message-MDD-Kanals auf der SORCUS-Karte kann der Aufruf einer Anwender-(Callback-)Prozedur auf der Host-PC-Seite ausgelöst werden. Die Prozedur auf der PC-Seite bekommt den beim Aufruf des Schreibdienstes übergebenen Block von Anwenderdaten als Parameter übergeben - das Anwenderprogramm braucht sich in der Prozedur also nicht um die Abholung der Daten zu kümmern.

Aus der Sicht des Anwenders reduziert sich der Programmieraufwand auf folgende Schritte:

1. Schreiben einer Callback-Routine im Programm auf der Host-PC-Seite
2. Öffnen eines Kanals zum Message-MDD aus dem PC-Programm heraus
3. Schreiben einer Funktion auf der Karte für den Transfer eines Handles vom PC zur Karte
4. Um Daten von der SORCUS-Karte zu einer anderen CPU zu verschicken, muß das Anwenderprogramm auf der Karte den Schreibdienst eines zum Message-MDD geöffneten Kanals aufrufen (*mddx_write_channel_block*) und ihm die entsprechenden Daten übergeben.

Intern geschieht beim Aufruf eines Message-MDD-Kanal-Schreibdienstes folgendes:

1. Der Message-MDD puffert die Daten auf der Karte zwischen und sendet einen (gepufferten) Service-Request an den Host-PC. Das Karten-Betriebssystem OsX erledigt anschließend das Absetzen des Service-Requests.
2. In der PC-Bibliothek wird der SRQ entgegen genommen und ausgewertet. Die Bibliothek liest über den Lesedienst des Message-MDD-Kanals automatisch die gepufferten Daten aus.
3. Die Bibliothek speichert beim Öffnen des Kanals einige Informationen intern. Mit Hilfe dieser Informationen ruft die Bibliothek die Anwender-Callback-Routine auf und übergibt ihr direkt die vom Message-MDD-Kanal erhaltenen Daten.



Installation

Parameter	Wert
Dateiname	MSG_MDD.EXE
Programmnummer	440h
Tasknummer	beliebig, (default = 205 = CDh)
Interruptnummer	0
Länge des Datenbereiches	0
Flags	980h (NI-Task)

Befehl in **INS-Datei**: M8INST MSG_MDD.EXE 0440 00CD 00 000000 00000980

Auf der PC-Seite versucht die Bibliothek den MDD automatisch zu installieren, sobald der erste Kanal zu ihm geöffnet wird.

Der Suchpfad für das Programm ist unter Windows im folgenden Registry-Eintrag festgelegt: *HKEY_LOCAL_MACHINE\SOFTWARE\SORCUS\MDD\mdd_path*. Betriebssysteme ohne zentrale Datenbank entnehmen den Suchpfad dem Eintrag *mdd_path* aus der Datei SORCUS.INI, die entweder im aktuellen Verzeichnis oder im Verzeichnis SORCUS stehen muß. Existiert dieser Eintrag nicht, wird versucht, das Programm aus dem Verzeichnis C:\SORCUS\ML8\MDD zu laden.

Die für die Installation zu verwendende Tasknummer steht unter Windows im Registry-Eintrag *HKEY_LOCAL_MACHINE\SOFTWARE\SORCUS\MDD\MsgMddTaskNo*. Unter DOS und LINUX steht die Tasknummer in der SORCUS.INI Datei unter dem Eintrag *msg_mdd*. Existiert dieser Eintrag nicht, wird die Tasknummer CDh verwendet.

Verwendung:

Der Kanal, über den eine Nachricht von einer CPU zu einer anderen verschickt wird, ist von dem Programm aus, das das Ziel der Nachricht darstellt, mit der folgenden Funktion zu öffnen (der Message-MDD muß sich auf der CPU befinden, auf der die Nachrichten abgeschickt werden):

```
HMDD mdd_open_channel_callback (USHORT slot,          = 0
                                 USHORT size,          = 4
                                 void   *pRcvHandlFunc,= Task und Funktionsnr.
                                                         einer RT-Funktion
                                 ULONG  param,         = 0
                                 void   *pCbFunc,      = Adresse der Callback-
                                                         Funktion
                                 USHORT mode)          = 0
```

Die Parameter *slot, size, param* und *mode* müssen die angegebenen Werte enthalten.

Der Parameter *pRcvHandlFunc* muß die Adresse einer Struktur folgenden Aufbaus enthalten:

```
struct { USHORT   task;       USHORT func;}
```

Darin wird eine Funktion auf der CPU spezifiziert, auf der auch der Message-MDD installiert ist, in der Regel aus dem Programm, das Absender der Nachrichten sein soll. Diese Funktion wird in *mddx_open_channel_callback* automatisch einmal aufgerufen, um das Handle des geöffneten Message-MDD-Kanals zum Absender zu transferieren. Die Funktion muß folgenden Aufbau haben:

```
void far pascal ReceiveHandle(USHORT task, USHORT *pInsize, HMDD8 *pHandle,
                              USHORT *pOutsize, void *pDummy);
```

Im Parameter *pHandle* übergibt *mddx_open_channel_callback* dem Echtzeit-Anwenderprogramm das Handle des zum Message-MDD geöffneten Kanals. Das Anwenderprogramm hat die Aufgabe, das erhaltene Handle (z.B. im Parameterbereich) zu speichern, um damit später Nachrichten an die Callback-Funktion zu versenden.

Im Parameter *pCbFunc* gibt der Anwender seine Callback-Prozedur an. Sie muß folgenden Aufbau haben:

```
void MyCallbackFunction (HMDD8 handle, ULONG param, ULONG size, void *pData);
```

Die beim Aufruf an die Prozedur übergebenen Parameter haben folgende Bedeutung:

1. handle reserviert (=0, wird nur verwendet wenn der Absender ein MDD-Kanal ist)
2. param reserviert (=0)
3. size Länge der übergebenen Nutzdaten (Anzahl Bytes, max. 256)
4. pData Zeiger auf die übergebenen Nutzdaten

mddx_open_channel_callback liefert das Handle des geöffneten Message-MDD Kanals zurück.

Das Abschicken der Daten auf der Echtzeit-Seite geschieht über den Aufruf von

```
mddx_write_channel_block (handle, size, data);
```

Dabei ist in *handle* das Handle des Message-MDD-Kanals zu übergeben, in *size* die Anzahl der übergebenen Bytes und in *data* ein Zeiger auf die Nutzdaten, die auf der Ziel-CPU der Callback-Funktion übergeben werden sollen.

Die direkte Verwendung des Message-MDD ohne die zugehörige Bibliotheksfunktion *mddx_open_channel_callback* ist nicht sinnvoll, da ein hoher Programmieraufwand erforderlich wäre. Daher ist der Aufbau der CPS des Message-MDD nicht offengelegt.