Application Note AN067

DMA-Programmierung mit der Modular-4/486 unter Windows NT

Autor: MS an067.doc (6 Seiten)

Einleitung

Seit der Version 7B.003 des NT-Treibers für SORCUS-Karten (mlxdrv.sys) kann die DMA-Funktionalität der Modular-4/486 genutzt werden. Das Hauptargument für die Nutzung des DMA-Betriebs ist die Entlastung des NT-Rechners, denn der Handshake mit der Modular-4/486 braucht relativ viel Rechenzeit. Vor allem Anwendungen, die hohe Datenmengen zwischen SORCUS- Karte und PC austauschen, wirkten sich bisher sehr negativ auf die Gesamtperformance von Windows NT aus. Der Einsatz des DMA-Controllers bringt Vorteile beim Übertragen langer Datenpakete, jedoch nur auf der Windows-NT-Seite. Bei kurzen Datenpaketen macht der Rechenaufwand zur Initialisierung des DMA-Transports den Gewinn bei der eigentlichen Datenübertragung zunichte. Auf dem SORCUS-Board wird durch den DMA-Betrieb kein Performance-Gewinn erzielt.

Zusätzlicher Datenkanal

Für lange Datenpakete sowohl zum PC als auch zur SORCUS-Karte kann ein 16-bit DMA-Kanal benutzt werden. Dazu muß auf der SORCUS-Karte eine Task installiert werden, die den DMA-Transport verwaltet und durchführt. Die Kommunikation mit dem Betriebssystem OsX über die bisher üblichen Bibliotheksfunktionen bleibt davon unberührt, abgesehen davon, daß sie dieselbe Schnittstelle benutzt. Die Bibliotheksfunktionen, die mit Makrobefehlen mit der Karte kommunizieren, haben auf der Schnittstelle die höhere Priorität, sie werden vom Treiber in die laufende DMA-Übertragung eingefügt. Ebenso werden Service Requests aus OsX-Programmen in eine laufende Datenübertragung eingefügt. Der DMA-Betrieb stellt also parallel zu den Makrobefehlen einen zusätzlichen Kommunikationskanal zur Verfügung.

DMA-Task

Um den DMA-Betrieb auf der SORCUS-Karte zu aktivieren, muß das Programm DMATASK.LIB installiert werden. Für die Installation dieser Task ist der Anwender selbst verantwortlich. Falls die Karte über Flash-Eeprom verfügt empfiehlt es sich, die DMATASK.LIB automatisch nach dem Reset der Karte laden zu lassen, ansonsten kann sie über eine Installationsdatei oder vom Anwenderprogramm selbst über *ml8_transfer_and_install* geladen werden.
Man kann DMATASK.LIB als Erweiterung des Betriebssystems verstehen, welche parallel zu den Kommunikationsmechanismen des OsX den DMA-Transport verwaltet und durchführt. DMATASK.LIB wird als NI-Task unter der Tasknummer 240 (0xF0) installiert. Ab

Treiberversion 7B.004 kann die Tasknummer durch einen Registry-Eintrag variiert werden. Die Kommunikationsschnittstelle zu den Anwenderprogrammen auf der Karte besteht aus 2 Funktionen: Beide Funktionen – *REQUEST_TOPC* (Funktionsnummer 2) für die Übertragung von Daten zum PC und *REQUEST_FROMPC* (Funktionsnummer 3) für die Übertragung vom PC zur Karte – erhalten als Argument eine Datenstruktur vom Typ *DMAREQUEST*. Diese Struktur ist folgendermaßen aufgebaut:

```
typedef struct
{
     ABSOLUTE_ADDR  pBufList;
     USHORT         usCallbTask;
     USHORT         usCallbFunc;
     ULONG          ulCallbParam;
}
DMAREQUEST;
```

`pBufList` Zeiger auf eine verkettete Liste von Puffern, die zum PC übertragen werden sollen. Der Zeiger ist als absolute Adresse zu verstehen, d.h. er muß u.U. mit der Funktion *ml8rt_phys_adr* konvertiert werden. Die Elemente dieser Liste sind wie folgt aufgebaut:

```
typedef struct
{
      ABSOLUTE_ADDR  pNext;
      ABSOLUTE_ADDR  pData;
      ULONG          ulSize;
      ULONG          ulCount;
}
BUFFERLIST;
```

`pNext` Absolute Adresse des nächsten Elements oder 0, falls kein Listenelement mehr folgt.

`pData` Zeiger auf die Daten, die übertragen werden sollen.

`ulSize` Größe des Datenpuffers, auf den `pData` zeigt.

`ulCount` enthält nach der Übertragung die Anzahl der tatsächlich übertragenen Bytes.

`ulCallbParam` User-Parameter, der nach Abschluß der DMA-Übertragung der Callback-Funktion übergeben wird.

`usCallbTask` Tasknummer einer Callback-Funktion, die nach erfolgter Übertragung aufgerufen wird.

`usCallbFunc` Funktionsnummer der Callback-Funktion. Diese Funktion, die im Anwenderprogramm implementiert werden muß, erhält als Argument die folgende Datenstruktur eine Datenstruktur mit folgendem Inhalt:

```
typedef struct
{
      ABSOLUTE_ADDR  pBufList;
      ULONG          ulNumBytes;
      USHORT         usReason;
      ULONG          ulCallbParam;
}
DMARESULT;
```

`pBufList` Zeiger auf die *BUFFERLIST*-Struktur, die an *REQUEST_FROMPC* bzw. *REQUEST_TOPC* übergeben wurde.

`ulNumBytes` Anzahl der übertragenen Bytes insgesamt (Summe über alle Elemente der Pufferliste)

`usReason` Grund für das Ende der Übertragung:

DMASRQ_COMPLETE (0x01CE)
Inhalt der gesamten Pufferliste ordnungsgemäß übertragen

DMASRQ_NOMOREBUFFERS (0x02CE)
Datenübertragung konnte nicht vollständig durchgeführt werden.
Bei Übertragung zum PC: PC hat am Ende der Pufferliste noch Daten angefordert -> ok
Bei Übertragung vom PC: PC hat am Ende der Pufferliste noch Daten zu schicken versucht -> PC-seitiges Fehlerhandling erforderlich, da der Datenblock nicht komplett übertragen wurde.

DMASRQ_TIMEOUT (0x03CE)
Abbruch der DMA-Übertragung durch Timeout

DMASRQ_SYNCERROR (0x04CE)
Fehler bei der Datensynchronisierung zwischen PC und Karte

`ulCallbParam`
dieser Wert wird beim Start der DMA-Übertragung der Funktion *REQUEST_TOPC/REQUEST_FROMPC* übergeben. DMATASK.LIB speichert diesen Parameter und übergibt ihn am Ende der Übertragung an die Callback-Funktion.

DMA-Betrieb in NT-Anwenderprogrammen

Die DMA-Kommunikation mit der SORCUS-Karte wird mit den beiden neuen Bibliotheksfunktionen *ml8_read_dma* und *ml8_write_dma* (ab Bibliotheksversion 7.B.004) programmiert.

ml8_read_dma

Funktion	**ml8_read_dma** liest einen Datenblock über DMA von der SORCUS-Karte ein	
PASCAL	FUNCTION ml8_read_dma (VAR pbuffer; ulBufferSize : ULONG; VAR pulBytesRead : ULONG) : USHORT;	
C	USHORT ml8_read_dma (VOID *pbuffer, ULONG ulBufferSize, ULONG *pulBytesRead);	
Parameter	*pbuffer*	Zeiger auf einen Puffer, in den die empfangenen Daten geschrieben werden
	ulBufferSize	Größe des Puffers = max. Anzahl gelesener Bytes
	pulBytesRead	Enthält nach dem Aufruf die Anzahl der tatsächlich gelesenen Bytes. Sollte unbedingt ausgewertet werden!
Rückgabe	0 oder Fehlercode, siehe Handbuch der Modular-4/486, Anhang E. Das High-Byte des Rückgabewertes enthält die Fehlerklasse, das Low-Byte den Fehlercode.	

ml8_write_dma

Funktion	**ml8_write_dma** schreibt einen Datenblock über DMA zur SORCUS-Karte	
PASCAL	FUNCTION ml8_write_dma (VAR pbuffer; ulBufferSize : ULONG; VAR pulBytesWritten : ULONG) : USHORT;	
C	USHORT ml8_write_dma (VOID *pbuffer, ULONG ulBufferSize, ULONG *pulBytesWritten);	
Parameter	*pbuffer*	Zeiger auf einen Puffer, der die zu übertragenden Daten enthält
	ulBufferSize	Größe des Puffers = Anzahl zu schreibender Bytes
	pulBytesWritten	Enthält nach dem Aufruf die Anzahl der tatsächlich geschriebenen Bytes. Sollte unbedingt ausgewertet werden!
Rückgabe	0 oder Fehlercode, siehe Handbuch der Modular-4/486, Anhang E. Das High-Byte des Rückgabewertes enthält die Fehlerklasse, das Low-Byte den Fehlercode.	

Neue Registry-Einträge für den DMA-Betrieb

Die beschriebenen Einträge müssen unter dem Schlüssel
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\mlxdrv\Mlx?
vorgenommen werden. Das Fragezeichen in *Mlx?* steht für die Boardnummer der betreffenden Modular-4/486.

DMAChannel Nummer des benutzten DMA-Kanals. Datentyp: DWORD.
Die Treiber unterstützen momentan nur die 16-bit DMA-Kanäle (5-7).
Falls dieser Registry-Eintrag fehlt oder den Wert 0 hat, wird der DMA-Betrieb nicht aktiviert.
Der hier ausgewählte Kanal muß zusätzlich auf der Modular-4/486 über J3 eingestellt werden. Die Anleitung hierzu finden Sie im Handbuch, Kapitel 2.
Hinweis: Die Änderung wird erst nach dem Booten des Systems wirksam!

DMATaskNum Tasknummer der Treibertask DMATASK.LIB. Datentyp: DWORD.
Falls dieser Eintrag fehlt, geht der NT-Treiber davon aus, daß die DMA-Treibertask auf der Karte unter Tasknummer 240 (0xF0) installiert ist.

Anwendungsbeispiel

Das beigefügte Anwendungsbeispiel (messtask.c) tastet in einer TI-Task mit der Abtastrate 1 kHz alle 16 Eingangskanäle eines M-AD12-16-Moduls ab. Die Abtastwerte werden in einen Wechselpuffer geschrieben. Wenn dieser Puffer voll ist, wird die Funktion *REQUEST_TOPC* von DMATASK.LIB aufgerufen. Nachdem DMATASK.LIB den Puffer übertragen hat, wird die Callback-Routine *dma_end_callback* aufgerufen. In dieser Routine wird der gerade übertragene Wechselpuffer wieder als frei markiert. Falls nicht alle Daten korrekt übertragen werden konnten, wird ein Service Request zum PC-Programm geschickt.

Das Anwendungsbeispiel auf der PC-Seite (dmasamp.c) installiert zunächst die Treibertask DMATASK.LIB, den Modul Device Treiber für das Modul M-AD12-16 und die TI-Task MESSTASK.EXE. Dann wird in einer Schleife *ml8_read_dma* aufgerufen. Aus den empfangenen Daten werden in diesem Beispiel Mittelwerte, Maxima und Minima errechnet. Das Beispielprogramm dmasamp.c installiert außerdem eine User-Serviceprozedur, die aufgerufen wird, wenn in MESSTASK.EXE ein Pufferüberlauf stattgefunden hat oder die Daten nicht korrekt übertragen werden konnten.

Benötigte Dateien:

Messtask.c	Quellcode des OsX-Programms Messtask.exe
Messtask.h	Definitionen für Messtask.c
Messtask.exe	Anwendungsprogramm auf der SORCUS-Karte
Dmatask.lib	DMA-Steuertask
Dmatask.h	Headerdatei zu DMATASK.LIB
ML8D2700.LIB	Modul Device Treiber für M-AD12-16
Dmasamp.c	Quellcode des WIN32-Programms Dmasamp.c
Dmasamp.exe	PC-Anwendungsprogramm, als Konsolenapplikation übersetzt

Weitere Systemvoraussetzungen:

- NT-Treiber für Modular-4/486, Version 7B.003 oder höher. Hinweis: Der Registry-Eintrag für den benutzten DMA-Kanal muß wie oben beschrieben von Hand eingefügt werden!
- SORCUS PC-Bibliothek (wird mit dem NT-Treiber installiert), Version 7B.003 oder höher und zugehörige Importbibliothek mlxw32.lib + Headerdateien, falls Sie dmasamp.exe selbst erstellen wollen.
- Modular-4/486 sollte unter Kartennummer 0 installiert werden, ansonsten muß der Aufruf *ml8_reset* in dmasamp.c angepaßt werden.
- Modul M-AD12-16 sollte auf Steckplatz 1 gesteckt werden, ansonsten muß die Konstante *M_AD12_16_SLOT* in messtask.h angepaßt werden.