5. Software

PC-Zusatzkarten von SORCUS sind 'intelligent', das heißt, sie sind eigenständige Computer, die durch ihre verschiedenen Schnittstellen und ihre 'on-board' Software für eine Vielzahl von Meß-, Steuer-, Regelungs- und Kommunikationsaufgaben eingesetzt werden können. Durch ihre freie Programmierbarkeit und bei den MODULAR-4 Karten durch die aufsteckbaren Module bieten sie eine hohe Anpassungsfähigkeit. Hinzu kommt die sehr hohe Verarbeitungsgeschwindigkeit der CPU. Es gibt Multi-LAB/2 Karten mit 16 MHz und MODULAR-4 Karten, deren CPUs mit bis zu 33 MHz extern getaktet werden (intern dann mit 33, 66 oder 133 MHz, je nach Version).

Dieser hohen Leistungsfähigkeit der Hardware ist auch die Software auf der Karte angepaßt: Ein eigenes Echtzeit-Multi-Tasking-Betriebssystem, das auf der Karte läuft, verwaltet alle Meß-, Steuerungs- und Regelungsprogramme und auch die Kommunikation mit dem Host-Rechner, also dem PC. Einige der Karten können mit minimaler Änderung der Software auch als 'Stand-alone'-Systeme eingesetzt werden. In diesem Fall kann die Kommunikation mit einem (auch räumlich weiter entfernten) Host-Rechner über die serielle Schnittstelle der Basiskarte oder über eine andere Schnittstelle auf den Modulen erfolgen, z.B. über IEC-Bus, RS-232, RS-422, RS-485 oder Lichtwellenleiter. Die Multi-LAB/2 Karte kann aber nur in einem PC betrieben werden. Die Beschreibung in diesem Kapitel bezieht sich hierauf.

Das Betriebssystem muß nach einem Reset der Karte immer ins RAM der Karte geladen werden. Der PC kommuniziert mit der Karte auf unterer Ebene in einer Makrobefehlssprache. Darauf aufgebaut sind PC-Bibliotheken, die es ermöglichen, die Karte bequem in Hochsprachen anzusprechen. Diese PC-Bibliotheken sind in Kapitel 6 beschrieben, die zugrundeliegenden Makrobefehle in Kapitel 12.

Das Betriebssystem unterstützt alle Schnittstellen und Funktionseinheiten der Karte und ermöglicht dadurch, die Karte komplett vom PC aus zu betreiben. Für Anwendungen, bei denen es nicht auf hohe Geschwindigkeit ankommt, kann allein mit diesen Befehlen die Karte durchaus erfolgreich eingesetzt werden.

Die Möglichkeiten der Karte als Meß-, Steuerungs- und Regelungssystem kommen aber erst bei Ausnutzung der Multi-Tasking-Fähigkeiten des Betriebssystems voll zum Tragen. Damit ist es möglich, die gesamte Echtzeitprogrammierung auf die Karte zu verlagern. Der PC muß dann lediglich noch die Ansteuerung und Parameterübergabe vornehmen.

Das Echtzeit-Multi-Tasking-Betriebssystem unterstützt 1024 Tasks (task, engl. = Aufgabe, Arbeit), die Interrupt-gesteuert (sog. I-Task) oder Nicht-Interrupt-gesteuert (sog. NI-Task) sein können. Jeder Task kann ein Anwendungsprogramm zugeordnet werden. Diese Zuordnung wird im folgenden als "Installieren" bezeichnet. Es gibt viele Anwendungsprogramme, die mehr als eine Task benötigen und die deshalb aus mehreren Teilprogrammen bestehen. Jedes dieser Teilprogramme muß dann auch unter einer eigenen Task installiert werden. Die Anwendungsprogramme werden vom PC aus ins RAM der Karte geladen. Hierzu steht das Hilfsprogramm MLC zur Verfügung, mit dem die Karte z.B. beim Starten des Systems automatisch wie gewünscht konfiguriert werden kann. Daneben bieten die PC-Bibliotheken die Möglichkeit, Anwendungsprogramme aus dem eigenen PC-Programm heraus auf die Karte zu laden und zu installieren.

5.1. Programmierebenen

5.1.1. Verwendung fertiger Softwarepakete

Der Anwender braucht keinerlei eigene Programmierarbeit zu leisten. Auf dem Markt erhältlich bzw. in Vorbereitung sind fertige Programmpakete für Meßdatenerfassung und Auswertung, Prüfstandssteuerung, Regelung, etc. Sie enthalten spezielle Treiber für die Multi-LAB/2 Karte und nutzen die Fähigkeiten der Karte optimal aus, z.B.:

Produkt	Hersteller	Lieferant
ARGUS	SORCUS	SORCUS
DIA*dem*	GfS, Aachen	GfS
DIA*dact*	GfS, Aachen	GfS
DIA/DAGO	GfS, Aachen	GfS
PDES	GIF, Aachen	GIF
DASY*Lab*	DASYTEC	DATALOG

In Kapitel 1 finden Sie eine kurze Beschreibung von ARGUS, DIA*dact* und DIA/DAGO.

5.1.2. Verwendung fertiger Echtzeitprogramme

Da die gesamte Echtzeitprogrammierung ja bereits durch die für die Karte vorhandenen Programme zur Verfügung gestellt wird (vorausgesetzt, die Programme decken die Problemstellung ab), beschränkt sich der noch zu leistende Programmieraufwand auf die Ansteuerung der Karte. Da dies im allgemeinen nicht mehr zeitkritisch ist, spielt es auch keine große Rolle, in welcher Programmiersprache die PC-Programme geschrieben sind.

Da ständig neue Echtzeit-Anwendungsprogramme entwickelt werden, empfehlen wir Ihnen, sich über den aktuellen Stand bei Ihrem Händler oder direkt bei SORCUS zu informieren, wenn das für Ihren Anwendungsfall erforderliche Programm nicht bereits auf Diskette mitgeliefert wurde.

5.1.3. Entwickeln eigener Echtzeitprogramme

Hierfür sind in Kapitel 7 Beispiele in Borland Pascal und Borland C++ angegeben. Für darüber hinausgehende Informationen, z.B. zu den auf der Karte eingesetzten ICs und ihrer Programmierung, wird auf die allgemein erhältliche Literatur verwiesen. In den Kapiteln 7 bis 10 wird ausführlich auf die Programmierung von eigenen on-board Echtzeitprogrammen eingegangen.

Außerdem wird auf die Seminare, die SORCUS anbietet, hingewiesen.

5.2. Das Multi-Tasking-Betriebssystem "OsX"

Jede Multi-LAB/2 Karte wird mit einem eigenen Echtzeit-Multi-Tasking- Betriebssystem geliefert, das alle Schnittstellen und Funktionseinheiten der Karte unterstützt. Es wird vom PC aus auf die Karte geladen und anschließend vom PC aus über Makrobefehle und von Tasks auf der Karte über System-Subroutinen angesprochen. Die Makrobefehle und System-Subroutinen liegen den Bibliotheken zugrunde, mit denen Sie das Betriebssystem direkt in Hochsprache ansprechen können.

5.2.1. Das Prinzip

Das Echtzeit-Multi-Tasking-Betriebssystem OsX der Multi-LAB/2 Karte geht von einem objektorientierten Ansatz aus. Es wurde bereits 1986 von SORCUS Computer für Z80-Systeme und 1990 für Intel i486-Prozessoren entwickelt und hat bis heute in Tausenden von Anwendungen seine hohe Effizienz und Benutzerfreundlichkeit unter Beweis gestellt. Es erlaubt dem Anwender, mehrere unabhängige Programme (theoretisch bis zu 1024) gleichzeitig auf einer Multi-LAB/2 Karte laufen zu lassen. Zur Zeit können Programme Interrupt-gesteuert (sog. DI-Task oder II-Task) und Nicht-Interrupt-gesteuert (sog. NI-Task) sein.

Um ein Programm auf der Multi-LAB/2 Karte laufen zu lassen, muß es zunächst im Speicher der Karte vorhanden sein. Dann muß das Programm unter einer Task installiert werden, z.B. mit einem Bibliotheksaufruf vom PC aus. Das Programm läuft aber erst dann los, wenn die Task aktiviert wird bzw. der zugehörige Interrupt demaskiert wird. Auch dies kann vom PC aus gemacht werden. Bereits nach dem Installieren besitzt die Task folgende Strukturen:

Programm (bestehend aus Prozeduren)

Datenbereich

Parameterbereich

Die Wahl der Tasknummer bleibt dem Benutzer überlassen, sie hat keinen Einfluß auf die Priorität, mit der die Programme auf der Karte bearbeitet werden (siehe unten). Lediglich Task 0 ist zur Zeit für das Betriebssystem reserviert, das immer unter Task 0 installiert ist und auch unter Tasknummer 0 angesprochen wird.

5.2.2. Die Echtzeitprogramme

Ein Echtzeitprogramm besteht aus einer Anzahl von Prozeduren (im folgenden wird einfach der Begriff Prozedur für eine Subroutine verwendet, unabhängig von der Zahl der hin- oder zurückgegebenen Parameter). Dabei wird unterschieden zwischen

Haupt-Prozedur

Auto-Init-Prozedur

Globalen Prozeduren

Die **Haupt-Prozedur** ist die Prozedur, die vom Betriebssystem aufgerufen wird, wenn die Task an der Reihe ist. Sie stellt also das eigentliche Programm der Task dar.

Beispiel: Ein Programm zur Meßdatenerfassung wird unter einer interruptgesteuerten Task installiert. Der Interrupt wird durch einen Timer ausgelöst, der den Abtast-Trigger darstellt. In der Haupt-Prozedur, die bei jedem Interrupt aufgerufen wird, wird nun der A/D-Wandler angesteuert, die gewandelten Meßdaten vom A/D-Wandler in den Speicher geschrieben und ein Daten-Pointer weiterbewegt. Außerdem wird die Anzahl der Meßdaten mitgezählt und das Programm gegebenenfalls bei Ende der Messung abgebrochen.

Die **Auto-Init-Prozedur** eines Programmes wird (optional), unmittelbar nachdem das Programm unter einer Task installiert wurde, einmalig vom Betriebssystem aufgerufen. Der Aufruf kann aber mit einem Flag bei der Installierung auch unterbunden werden.

Beispiel: Bei einem PID-Regler-Programm wird nach der Installierung der analoge Ausgang auf einen definierten Anfangswert gesetzt.

Globale Prozeduren sind Subroutinen, die von allen Tasks aus aufgerufen werden können, auch z.B. vom PC aus. Sie "gehören" zu einem Programm und stellen anderen Programmen bzw. Tasks bestimmte Funktionen zur Verfügung, die üblicherweise im Zusammenhang mit dem Programm stehen, aber nicht müssen.

Beispiel: Ein interruptgesteuertes Programm holt Daten von den digitalen Eingängen ab und legt sie in einen Puffer. Das Programm stellt eine Prozedur zur Verfügung, mit dem ein anderes Programm ein oder mehrere Zeichen aus dem Puffer auslesen kann.

Unbedingt erforderlich für ein Programm ist lediglich die Haupt-Prozedur, alle übrigen Prozeduren sind optional. Falls die Task, unter der das Programm installiert wurde, nie aktiviert wird, kann man sogar die Haupt-Prozedur weglassen. Eine sol-

che Task könnte dann z.B. nur einen Daten- und/oder einen Parameterbereich haben, in dem globale Parameter für viele andere Tasks zur Verfügung gestellt werden. Aus Gründen der Übersichtlichkeit kann es in größeren Projekten auch durchaus sinnvoll sein, bestimmte Prozeduren, die von mehreren Tasks aus aufrufbar sein sollen, nur einmal auf der Karte zu halten und einer Task zuzuordnen, die selbst gar keine Hauptprozedur hat.

5.2.3. Die Tasktypen

Wann eine Task an der Reihe ist, also deren Haupt-Prozedur aufgerufen wird, hängt zunächst einmal vom Tasktyp ab. Dieser wird durch das Programm selbst vorgegeben oder bei der Installierung der Task festgelegt. Es gibt zwei Tasktypen:

Interrupt-Tasks (DI- bzw. II-Task)

Nicht-Interrupt-Tasks (NI-Task)

5.2.3.1. Interrupt-Tasks (DI- bzw. II-Tasks)

Sie haben die höchste Priorität. Wenn die entsprechende Interrupt-Bedingung auftritt, wird die Haupt-Prozedur des unter der Interrupt-Task installierten Programmes einmal aufgerufen. Das Programm, also die Haupt-Prozedur, bestimmt selbst, wann es die Kontrolle an das Betriebssystem zurückgibt.

Der Unterschied zwischen einer DI-Task (Direkte Interrupt-Task) und einer II-Task (Indirekte Interrupt-Task) besteht nur im Service, den das Betriebssystem dem Programmierer für die Haupt-Prozedur zur Verfügung stellt. Die Haupt-Prozedur einer II-Task ist eine ganz gewöhnliche Prozedur und hat den gleichen Aufbau wie alle anderen Prozeduren, auch wie die Hauptprozeduren der anderen Tasktypen. Die Haupt-Prozedur einer DI-Task dagegen muß als Interrupt-Service-Routine programmiert werden, d.h. der Programmierer muß sich um das Retten der Register und das korrekte Verlassen der Prozedur selbst kümmern. Dies kann für sehr zeitkritische Anwendungen ausgenutzt werden. Es ist aus vielen Gründen sinnvoll, die CPU durch interruptgesteuerte Programme so wenig wie möglich zu belasten.

Die Priorität der DI- bzw. II-Tasks untereinander ist durch Definition im Betriebssystem (also nicht durch Hardware) festgelegt. Es besteht durch entsprechende Programmierung die Möglichkeit, die Priorität zu ändern, was bisher aber nicht genutzt wird. Da einige Interrupt-Quellen, z.B. die Timer und die externen Interrupt-Eingänge IP-0 und IP-1, funktionell im übrigen gleichwertig sind, kann die Priorität auch durch die Wahl des Timers oder des Interrupt-Eingangs in gewissem Umfang beeinflußt werden.

Bei der Installierung einer Interrupt-Task kann die Interrupt-Nummer mit angegeben werden, wenn sie nicht durch das Programm fest vorgegeben ist. Sie entspricht nicht der Interrupt-Vektor-Nummer (siehe Tabelle 5-1). Im folgenden wird einfach von Interrupt xxx gesprochen, wenn die Interrupt-Nummer gemeint ist. In Tabelle 5-2 sind alle Interrupts der Multi-LAB/2 Karte zusammengefaßt. Die ersten 32 Interrupts (0 bis 31) sind den von Intel bzw. NEC reservierten Traps, Errors, etc. fest zugeordnet (Anmerkung: beim IBM-AT unter DOS ist die Belegung anders, hier in OsX hingegen wurde eine saubere Zuordnung gewählt). Nummer 128 bis 134 sind die Hardware-Interrupts der Karte, Nummer 135 bis 255 sind für SORCUS reserviert und dürfen nicht verwendet werden.

Die weiteren Interrupts von 32 bis 95 sind Soft-Interrupts und für SORCUS reserviert, die Interrupts 96 bis 127 können frei vom Anwender verwendet werden.

Priorität	Interrupt-Nr.		Quelle	Beschreibung
	Dez.	Hex.		
Hoch	128	80h	IP-0	Ext. Interrupt-Eingang IP-0
	129	81h	IP-1	Ext. Interrupt-Eingang IP-1
	131	83h	TA	Timer A
	132	84h	TB/Up	Timer B, Aufwärtszähler
	133	85h	TB/Down	Timer B, Abwärtszähler
	134	86h	SIO-RBF	Serielle Schnittstelle Receive
Niedrig	130	82h	PC-RBF	Befehl vom PC

Tab. 5-1: Die Hardware-Interrupts der Multi-LAB/2 Karte

Interrupt-Nummer		Typ	Quelle: Erklärung
0	(00h)	Fault	CPU: Divide Error (z.B. / 0)
1	(01h)	Fault	CPU: Debug Except. (Trap/Fault)
2	(02h)	NMI	ext.: INT 2 oder NMI
3	(03h)	Trap	CPU: Soft-Interrupt (INT 3) (1-Byte Opcode)
4 - 31	(04h - 1fh)	-	Reserviert
32 - 95	(20h - 5fh)	Trap	Soft-Interrupt (2-Byte Opcode), reserviert für SORCUS
96 - 127	(69h - 7fh)	Trap	Soft-Interrupt (2-Byte Opcode), frei für Anwender
128 - 134	(80h - 86h)	Hardware	Interrupt-Controller auf der Karte (siehe Tab. 5-1)
135 - 255	(87h - ffH)	Hardware	Reserviert

Tab. 5-2: Übersicht aller Interrupts der Multi-LAB/2 Karte

5.2.3.2. Nicht-Interrupt-Tasks (NI-Tasks)

Sie haben die niedrigste Priorität. Sie können von DI- bzw. II-Tasks unterbrochen werden. NI-Tasks werden in einer vom Anwender festgelegten Reihenfolge aufgerufen. Die Häufigkeit, mit der eine NI-Task aufgerufen wird, kann dadurch erhöht werden, daß sie mehrfach aktiviert wird. Dadurch ändert sich aber nur ihre Aufruffrequenz relativ zu anderen NI-Tasks. Auch NI-Tasks bestimmen selbst, wann sie die Kontrolle an das Betriebssystem zurückgeben.

Eine Priorität im herkömmlichen Sinne von NI-Tasks untereinander existiert eigentlich nicht, weil NI-Tasks vom Betriebssystem einfach in einer bestimmten Reihenfolge aufgerufen werden, die durch die Reihenfolge ihrer Aktivierung gegeben ist. Da aber NI-Tasks auch mehrfach aktiviert werden können, kann man sagen, daß eine häufig aktivierte NI-Task statistisch auch häufiger aufgerufen wird und damit eine höhere Bedeutung hat.

Auch bei NI-Tasks besteht die Möglichkeit, eine bestimmte Task vorübergehend als die nächste aufzurufende in der Reihenfolge ganz nach vorne zu schieben, so daß sie in jedem Fall als nächste NI-Task drankommt. Falls gerade eine NI-Task läuft, so wird damit gewartet, bis diese die Kontrolle an das Betriebssystem zurückgegeben hat.

5.2.4. Daten- und Parameterbereich

Bei der Installierung eines Programmes unter einer Task bekommt diese Task einen Datenbereich und einen Parameterbereich zugewiesen. Die Größe dieser beiden Bereiche wird vom Programm selbst, die des Datenbereichs kann auch beim Installieren festgelegt werden. Beide Bereiche sind später nicht mehr änderbar. Sie können auch die Länge 0 haben. Sie sind voneinander unabhängige, lineare, durchgängige Bereiche. Sie unterscheiden sich durch ihre maximal erlaubte Größe und durch die Art, wie auf ihre Inhalte zugegriffen wird:

- Der Zugriff auf den Datenbereich geschieht über Betriebssystemaufrufe. Dazu existiert für jede Task ein Schreib- und ein Lese-Pointer, die von den vom Betriebssystem zur Verfügung gestellten Subroutinen und von Makrobefehlen verwendet werden. Der jeweilige Pointer wird nach jedem Zugriff um die Anzahl Byte inkrementiert, die gelesen bzw. geschrieben wurden. Der Datenbereich eignet sich also besonders für fortlaufende Daten, wie sie zum Beispiel bei einem Meßprogramm anfallen.
- Auf den Parameterbereich wird direkt durch Angabe der Nummer des Parameters zugegriffen. Die Nummer ist die relative Adresse des ersten Byte des Parameters, bezogen auf den Anfang des Parameterbereichs.

	Datenbereich	Parameterbereich
Länge	0 bis max. verfügbarer Speicher	0 bis 65280 Byte
Zugriff	indirekt über Pointer	direkt

5.2.5. Einige interne Task-Tabellen des Betriebssystems

Bei der Installierung eines Programmes unter einer Task werden außer Daten- und Parameterbereich auch einige Tabellen angelegt, die für die Verwaltung notwendig sind. Die beiden wichtigsten sind:

Programm-Deskriptor-Tabelle (PDT)

Task-Deskriptor-Tabelle (TDT)

Die Programm-Deskriptor-Tabelle (PDT) ist Teil des Programmes, das auf die Karte geladen und unter einer Task installiert wird. Sie wird durch die Installierung nicht verändert, auch das Programm selbst darf sie zur Laufzeit nicht mehr verändern. Es könnte ja z.B. sein, daß dasselbe Programm nochmals unter einer anderen Task installiert wird, dann werden die Informationen auch nochmals benötigt. Außerdem greift das Betriebssystem auch zur Laufzeit auf die Tabelle zu, allerdings nur lesend. Der Aufbau und eine genaue Erklärung der PDT finden sich im Anhang.

Die Task-Deskriptor-Tabelle (TDT) ist der jeweiligen Task zugeordnet. Sie enthält praktisch alle Informationen über die Task, über das unter der Task installierte Programm, z.B. auch die Adresse der PDT, und weitere Parameter, die vorwiegend zur Laufzeit benutzt werden. Hier stehen auch die oben erwähnten Schreib- und Lese-Pointer für Zugriffe auf den Datenbereich. Der Aufbau und eine genaue Erklärung der TDT finden sich im Anhang.

5.2.6. Fehlerbehandlung

Fehler und Service-Requests (sog. SRQs) werden als Wort (2 Byte) über die Schnittstelle zum PC gesendet. Eine Aufstellung dieser Fehler- und SRQ-Codes finden Sie im Anhang F.

Bei Fehlern und SRQs wird wie folgt unterschieden:

Typ	Beispiel
Systemfehler	CPU-Defekt nach Selbsttest
Spontaner Fehler	Nach Abschluß eines Makrobefehls
Provozierter Fehler	während Ausführung eines Makrobefehls
System-SRQ	Überlauf eines Systempuffers
Anwender-SRQ	SRQ aus Anwendungsprogramm

Innerhalb des Betriebssystems auftretende Fehler werden direkt über die PC-Schnittstelle zum PC gesendet, z.B. solche Fehler, die nach dem Einschalten des Systems und beim anschließenden Selbsttest der Karte auftreten.

5.2.7. Installierung von Programmen

Ein Anwendungsprogramm, das von SORCUS auf Diskette geliefert wurde, muß zunächst in das RAM der Karte geladen und anschließend installiert werden. Der Name solcher Programmdateien ist:

M2Pxxxx.LIB oder

M2Pxxxx.EXE

Dabei bedeutet xxxx die Programm-Nummer zwischen 1 und ffffh (hexadezimal).

Zur komfortablen Installierung der Programme finden Sie auf einer der Originaldisketten, die mit der Multi-LAB/2 Karte geliefert wurden, das Hilfsprogramm

"MLC".

Das Programm "MLC" (**M**ulti-**L**AB-**C**ommander) kann auch von der "AUTOEXEC.BAT“-Datei des PC automatisch aufgerufen werden. Es sorgt dann für eine komplette Installierung der gewünschten Programme auf der Karte, z.B. beim Einschalten des Systems, ohne daß der Anwender eingreifen muß. Eine ausführliche Beschreibung des PC-Hilfsprogrammes finden Sie in Kapitel 4.

Wenn Sie die Programme aus Ihrem eigenen PC-Programm heraus installieren wollen, stehen dafür Funktionen in der PC-Bibliothek "ML2BIB" zur Verfügung. Mehr darüber erfahren Sie in Kapitel 6.

Echtzeitprogrammdateien mit der Erweiterung ".EXE" dürfen nicht auf dem PC gestartet werden und umgekehrt für den PC geschriebene Programme nicht auf die Karte geladen werden. In den Kapiteln 7, 8, 9 und 10 finden Sie Angaben dazu, wie Echtzeitprogramme für die Multi-LAB/2 erstellt werden. **!**

5.3. Installieren eines neuen Betriebssystems

Nach jedem Reset der Karte wird ein Betriebssystem geladen. In der Regel wird der Name des zu ladenden Betriebssystems im Resetbefehl der PC-Bibliothek bzw. der Installationsdatei direkt angegeben und kann somit jederzeit geändert werden. Der Name der Betriebssystemdatei enthält Versionsinformationen und ist wie folgt aufgebaut:

ML2-za.nnR Neues Betriebssystem

(z = Zahl 2 bis 9, a = Buchstabe A bis Z,

nn = lfd. Nr. des Betriebssystems)

Das Programm MLC bietet die Möglichkeit, bei einem Reset in einer Installationsdatei (in der M2DEVICE-Zeile) oder einem interaktiven Reset (z.B. [Strg]+[F10]) das jeweils aktuelle Betriebssystem zu laden. Dazu muß als Name "ML2-??.??R" angegeben werden. MLC durchsucht dann beim Reset das "OSX"-Verzeichnis (wird beim Installieren der Originaldisketten erstellt) und das Verzeichnis, in dem "MLC.EXE" steht, nach dem aktuellen Betriebssystem. Ein neues Betriebssystem muß also nur in eines dieser Verzeichnisse kopiert werden (Vorzugsweise in das "OSX"-Verzeichnis).

5.4. Aufbau des RAM-Bereichs der Karte

Das Betriebssystem residiert immer in den unteren 64 KByte. Die Interrupt-Vektor-Tabelle beginnt ab Adresse 0 und belegt 1 KByte. Nach einem Hardware-Reset und der Aktivierung des Betriebssystems können Sie die Grenzen des noch freien RAM von der Karte lesen. Die Grenzen stehen im Parameterbereich der Task 0, also des Betriebssystems (siehe Kapitel 5.5. oder Anhang K). Es werden immer physikalische 32 Bit-Adressen geliefert, als untere Grenze wird zur Zeit 0001 0000h gemeldet. Wenn ein Programm auf die Karte geladen und installiert wird, wird die untere bzw. obere Grenze des freien RAM entsprechend der Größe des Programmes und gegebenenfalls der Größe des Parameter- und Datenbereichs der gerade installierten Task verändert, also nach oben verschoben. Auch bestimmte Aktionen des Betriebssystems belegen zusätzlich etwas Speicherplatz.

Alle Anwendungsprogramme müssen im Real-Mode arbeiten.

5.5. Parameterbereich des Betriebssystems

Das Betriebssystem selbst ist ebenfalls eine Task auf der Karte: Task 0. Damit stehen alle Taskbefehle auch für den Zugriff auf die Strukturen des Betriebssystems zur Verfügung. Das Betriebssystem hat dementsprechend auch einen eigenen Parameterbereich, der teilweise auch für den Anwender von Interesse sein kann. Alle Parameter dürfen nur gelesen werden. Auch hier nicht erklärte bzw. reservierte Parameter dürfen auf keinen Fall verändert werden.

In der folgenden Tabelle bedeutet in der Spalte "Typ":

B = Ein-Byte-Parameter
nB = n-Byte-Parameter
W = Wort-Parameter (2 Byte)
D = Doppelwort-Parameter (4 Byte)
nS = n-Byte-String (ASCII-Zeichen)
P = Physikalische Adresse (4 Byte)

Parameter	Typ	Bedeutung
0 (=00h)	B	Version dieser Parameterdefinition: z.Zt. = 01h
1 (=01h)	W	Anzahl gültiger Parameter
4 (=04h)	B	Kartentyp (2=Multi-LAB/2)
5 (=05h)	B	Kartenrevision (1=A, 2=B)
10 (=0ah)	2S	Jahrhundert der Herstellung des Betriebssystems
12 (=0ch)	10S	Betriebssystem Name, Version und Revision, z.B.: "ML2-1A.00x" x = "R" für RAM-Version, x = "B" für Beta-Test-Version
22 (=16h)	8S	Datum der Herstellung des Betriebssystems, z.B.: "10/08/94" (tt/mm/jj)
30 (=1eh)	8S	Uhrzeit der Herstellung des Betriebssystems, z.B.: "12:42:59" (hh:mm:ss)
38 (=26h)	B	CPU-Typ: 01h = V20, 04h = i486, 05h = Pentium
39 (=27h)	B	CPU-Revision (siehe Intel-/NEC-Spezifikation)
40 (=28h)	B	CPU-Modell (z.Zt. nur gültig für Intel-Pentium)
41 (=29h)	B	CPU-Hersteller: 1 = Intel, 2 = NEC, 255 = unbek.
42 (=2ah)	D	Ergebnis des Hardware-Selbsttests der CPU (0 = ok, <> 0 = Fehler)
46 (=2eh)	B	Co-Proz.-Typ: 0 = keiner, 4 = 487, 5 = Pentium
47 (=2fh)	B	Co-Proz.-Hersteller: 1 = Intel, 255 = unbekannt
48 (=30h)	D	CPU-Features (z.Zt. nur gültig für Intel-Pentium und Nexgen 586)

Parameter	Typ	Bedeutung
56 (=38h)	W	Betriebssystem-Features: Bit 0: 0 = Datenzugriffe auf 1 MB beschränkt 1 = Datenzugriffe bis 4 GB erlaubt Bit 1: 0 = Datenbereiche werden von unten nach oben im Speicher reserviert 1 = von oben nach unten Bit 2: 0 = RAM-Größen-Erkennung automatisch 1 = nicht automatisch, fix Bit 3: 0 = freies RAM wird nicht initialisiert 1 = freies RAM wird mit 0 initialisiert
64 (=40h)	P	Physikal. RAM-Anfang (physikal. Adresse)
68 (=44h)	P	Physikal. RAM-Ende (letzte Stelle + 1)
72 (=48h)	P	Freies RAM-Anfang (erste freie Stelle)
76 (=4ch)	P	Freies RAM-Ende (letzte Stelle + 1)
100 (=64h)	W	EEPROM der Karte: Länge (Anzahl Wörter) (wenn = 0, dann Fehler beim Lesen des EEPROMs, Hinweis auf Ursache steht in Parameter 102, siehe Anhang F)
102 (=66h)	W	Parameter-Nr., ab der die EEPROM-Kopie steht
150 (=96h)	W	Timer-Quarzfrequenz (in Vielfachen von 10 kHz)
168 (=a8h)	W	Max. Anzahl Aktivierungen für NI-Tasks
170 (=aah)	W	Max. Anzahl Tasks (alle Typen)
176 (=b0h)	W	Max. Anzahl Puffer
200 (=c8h)	B	SRQ-Mode (PC-Schnittstelle): z.Zt. = 1
201 (=c9h)	B	SRQ-Delay (PC-Schnittstelle): z.Zt. = 64
214 (=d6h)	B	Watchdog: 0=Auto-Retrigger off, 1=on
226 (=e2h)	D	Größe des Stack
232 (=e8h)	D	Anfangsadresse Stack (physikal.)
240 (=f0h)	B	LED-Zustand (0=off, 1=on)
258 (=102h)	B	I/O-Adresse der LED