12. Makrobefehle

Einzelheiten der Hardware der PC-Schnittstelle für die Kommunikation zwischen PC und Multi-LAB/2 sind am Anfang von Kapitel 11 beschrieben. Alle im folgenden aufgeführten Makrobefehle sind im Betriebssystem der Karte enthalten. Sie sind nur für diejenigen interessant, die eigene PC-Treiber schreiben wollen.

PC-Programmbibliotheken in Borland Pascal und Borland C++ für alle Makrobefehle sind im Lieferumfang der Karte enthalten (auf Diskette). Sie müssen nur noch die entsprechenden Prozeduren aufrufen und können damit die benötigten Makrobefehle sehr einfach in Ihr PC-Programm einbinden.

Makrobefehle können auch gesendet werden, während Anwenderprogramme auf der Karte laufen. Allerdings ist Vorsicht geboten, wenn mit den Befehlen die gleichen Funktionseinheiten auf der Karte angesprochen werden, die auch von einem gerade auf der Karte laufenden Anwenderprogramm benutzt werden.

Die Kommunikation zwischen PC und Karte ist so organisiert, daß es nur die beiden folgenden Fälle geben kann:

Fall 1: Der PC fordert per Makrobefehl bestimmte Daten oder Aktionen der Karte an. Die Karte führt das aus und sendet ggf. eine Antwort zurück. Wenn eine Antwort gefordert ist, sendet der PC erst dann den nächsten Befehl, wenn er die Antwort komplett empfangen hat. Das erste Byte des Makrobefehls und das erste Byte der Antwort sind identisch, wenn keine Fehler aufgetreten sind.

Fall 2: Die Karte kann unabhängig davon jederzeit selbst aktiv werden und per Service-Request (SRQ) den PC zu bestimmten Aktionen auffordern. Dies tut sie, indem sie ein Wort zum PC sendet (eventuell mit Interrupt-Auslösung auf dem PC). Am gesetzten DLP-Bit im Status der Schnittstelle kann der PC unterscheiden, ob es sich um die Antwort auf einen zuvor gesendeten Makrobefehl oder um eine spontane Aktivität handelt. Das gleiche gilt, wenn Fehler auf der Karte auftreten.

Eine Kurzzusammenfassung aller Makrobefehle findet sich im Anhang G.

12.1. Das Format der Makrobefehle

Jeder Makrobefehl besteht mindestens aus 2 Byte. Das erste Byte enthält den Befehlscode, das zweite Byte enthält zwei Angaben: in den unteren 4 Bit steht ein Formatcode für den Befehl bzw. die Länge des Befehls (FCB), in den oberen 4 Bit steht ein Formatcode für die Antwort bzw. die Länge der Antwort (FCA).

Beim Formatcode für den Befehl bedeutet eine Angabe zwischen 0 und11 die Anzahl Byte, die nach den ersten beiden Byte noch folgen. Angaben >= 12 sind Codierungen für das Format des Befehls (s.u.).

Beim Formatcode für die Antwort bedeutet eine Angabe zwischen 0 und11 direkt die Länge der Antwort, und zwar die Gesamtzahl Byte, aus der die Antwort besteht. Bei 0 wird keine Antwort erwartet. Angaben >= 12 sind Codierungen für die Antwortlänge und das Format der Antwort. Dabei ist es auch möglich, die Länge der Antwort zunächst offenzulassen, also nicht vorzugeben, sondern sie statt dessen von der Karte selbst ermitteln zu lassen. In einem solchen Fall wird die Länge der Antwort als Teil der Antwort zurückgegeben.

Wenn eine Antwort erwartet wird, hat sie folgenden Aufbau: Das erste Byte enthält wieder den Befehlscode, das zweite Byte kann schon das erste Byte der Nutzdaten oder die Längenangabe der Antwort sein.

Fehlermeldungen und spontane Aktivitäten der Karte (z.B. Service-Requests (SRQs) und Traps) bestehen immer nur aus zwei Byte. Eine Übersicht aller Fehlermeldungen (Fehlergruppe und Fehlertyp) finden Sie in Anhang F. Beide Byte geben einen Hinweis auf den Grund der Aktivität, z.B. bei einem SRQ die Nummer der anfragenden Task.

Das erste Byte eines Makrobefehls wird immer im Low-Byte, das zweite im High-Byte übertragen, etc.

Das allgemeine Format der Makrobefehle (1 Kästchen = 1 Byte):

Befehl	CB	FCA	FCB	nL	nH	mL	mH	b1	:::	bn
Antwort	CA	s	wL	wH	vL	vH	a1	:::	ax	

Erklärung der Abkürzungen

CB	Makrobefehls-Code
FCA	Formatcode Antwort
FCB	Formatcode Befehl
n	Anzahl Nutzdatenbyte beim Befehl (nL, nH = Low-Byte, High-Byte von n)
m	Anzahl Nutzdatenbyte der Antwort (Vorgabe) (mL, mH = Low-Byte, High-Byte von m)
b1...bn	Nutzdatenbyte beim Befehl (optional)
CA	Makrobefehls-Code der Antwort
s	Statusmeldung
w	Anzahl verworfener Nutzdatenbyte des Befehls (wL, wH = Low-Byte, High-Byte von w)
v	Anzahl Nutzdatenbyte der Antwort (vL, vH = Low-Byte, High-Byte von v)
a1...ax	Nutzdatenbyte der Antwort

Hinweise zum Format:

1. Wenn bei der Ausführung des Makrobefehls kein Fehler aufgetreten ist und FCA ungleich 0 ist (also eine Antwort gefordert ist), dann ist CA = CB.
2. Wenn FCB = 0 ist, dann entfallen nL und nH und b1...bn.
3. Wenn 0 < FCB < 12 ist, dann entfallen nL und nH, n = FCB.
4. Wenn FCB = 0fh ist, dann kann n max. 65280 sein.
5. Wenn FCA = 0 ist, dann entfallen mL, mH und die gesamte Antwort (CA, s, wL, wH, vL, vH und a1...ax).
6. Wenn FCA = 1 ist, dann entfallen mL, mH, wL, wH, vL, vH und a1...ax.
7. Wenn 1 < FCA < 12 ist, dann entfallen mL, mH, s, wL, wH, vL und vH, x = FCA - 1.
8. Wenn FCA = 0fh ist, dann entfallen s, wL, wH, vL, vH, x = m - 1.
9. Wenn FCA = 0eh ist, dann entfallen mL, mH, s, wL, wH, x = v.

10. Wenn FCB = 0dh sind, kann der Makrobefehl selbst festlegen, wie vile Nutzdatenbyte des Befehls er verwendet und wie viele er verwirft. Die Antwort enthält in jedem Fall die Felder wH, wL und s.
11. Wenn FCA = 0dh ist, entscheidet der Makrobefehl, wie vile Nutzdaten er in der Antwort zuückgibt. Die Antwort enthält in jedem Fall die Felder vL, vH und s.
12. Die übrigen Werte für FCA und FCB sind reserviert.

Beispiel 1: Befehl ohne Nutzdatenbyte und ohne Antwort

CMD	0	0

Beispiel 2: Befehl mit n Nutzdatenbyte (0 < n < 12) ohne Antwort

CMD	0	n	b1	:::	bn

Beispiel 3: Befehl mit n Nutzdatenbyte (0 < n < 65280) ohne Antwort

CMD	0	0fh	nL	nH	b1	:::	bn

Beispiel 4: Befehl mit 2 Nutzdatenbyte, Antwort mit m Nutzdatenbyte (0 < m < 65280)

Befehl	CMD	0fh	2	mL	mH	b1	b2

Antwort	CMD	a1	:::	am

Beispiel 5: Befehl ohne Nutzdatenbyte, die Karte bestimmt die Länge der Antwort mit x Nutzdatenbyte selbst (-1 < x < 65280)

Befehl	CMD	0eh	0

Antwort	CMD	vL	vH	a1	:::	av

Beispiel 6: Befehl mit n Nutzdatenbyte (0 < n < 12),
Antwort mit m Nutzdatenbyte (0 < m < 11)

Befehl	CMD	m+1	n	b1	:::	bn

Antwort	CMD	a1	:::	am

Beispiel 7: Befehl mit n Nutzdatenbyte (-1 < n < 65280), die Karte bestimmt die Länge der Antwort mit v Nutzdatenbyte selbst (-1 < v < 65280)

Befehl	CMD	0eh	0eh	nL	nH	b1	:::	bn

Antwort	CMD	vL	vH	a1	:::	av

Beispiel 8: Befehl mit n Nutzdatenbyte (-1 < n < 260). Die Karte bestimmt selbst, wieviel sie davon nicht verwendet hat und meldet dies in w zurück. Bezüglich der Antwort wird im Befehl vorgegeben, wieviel Byte die Karte maximal zurückliefern darf (-1 < m < 256). Sie legt diese Anzahl selbst fest und liefert die Anzahl der tatsächlich zurückgelieferten Byte in v (v <= m). Zusätzlich erfolgt in der Antwort eine Statusmeldung (s), die dem Fehlertyp entspricht.

Befehl	CMD	0dh	0dh	nL	nH	mL	mH	b1	:::	bn
Antwort	CMD	s	wL	wH	vL	vH	a1	:::	av	

Die Befehlscodes von 00h bis 7fh sind für die Makrobefehle vorgesehen. Sie dienen dazu, das Multi-Tasking-Betriebssystem bzw. die Strukturen der einzelnen Tasks anzusprechen.

Alle Parameter, wie z.B. Tasknummer, Parameter-Nummer, etc. werden in der maximal erforderlichen Länge angegeben. Bei einem Zugriff auf Parameter wird also z.B. die Parameter-Nummer als 16 Bit Zahl (2 Byte) angegeben, weil eine Task maximal 65280 Parameterbyte) haben kann

12.2. Kommunikationsbefehle

Bei "Code/Data" sind die einzelnen Byte in der Reihenfolge angegeben, wie sie zur Karte geschickt werden. Das gilt auch für solche Angaben, die aus zwei Byte bestehen, wie z.B. die Tasknummer (tL und tH). Das erste Byte des Befehls wird im Low-Byte, das zweite im High-Byte, das dritte dann wieder im Low-Byte, etc. gesendet.

Code/Data	Befehl
00h 20h	**Anforderung: Überprüfen, ob Multi-LAB/2 Karte bereit ist** Dieser Befehl sollte so oft gesendet werden, bis als Antwort 00h 00h zurückkommt.
00h 00h	Antwort: Multi-LAB/2 Karte ist bereit für Kommunikation
01h 20h	**Anforderung: Typ der Karte und Betriebssystem melden**
01h 02h	Antwort (02h für Multi-LAB/2, andere für andere Karten)

12.3. Konfigurationsbefehle

Mit diesem Makrobefehl können verschiedene Optionen auf der Karte eingestellt werden, die den Programmablauf in Bezug auf PC-Makrobefehle regeln.

Code/Data	Funktion
1bh 02h 10h 10h	**PC-Makrobefehle unterbrechbar durch II-/DI-Tasks**
1bh 02h 00h 10h	**PC-Makrobefehle nicht unterbrechbar** (= Einstellung nach Reset)

12.4. Systemzugriffe: Speicher (privilegierte Befehle)

Code/Data	Funktion
20h 04h aE aF aG aH	**Pointer P setzen** (absolute physikalische Adresse)
25h f0h mL mH	**Anforderung: Block von (Pointer) lesen, P = P + m** m = Anzahl Byte (1 bis 65520)
25h b1...bm	Antwort
21h 0fh nL nH b1...bn	**Block an (Pointer) schreiben, P = P + n** n = Anzahl Byte (1 bis 65520)
24h 2 55h aah	**Starte Programm ab Adresse des Pointers**
22h 52h 00h	**Anforderung: Freien Speicherplatz melden**
z	Alignment: Der Anfang des freien Bereichs kann auf eine Grenze gelegt werden, deren Adresse durch eine 2er Potenz teilbar ist: z=0: Byte, z=1: Wort, z=2: Doppelwort, ...
22h aE aF aG aH	Antwort: Größe des freien Speichers in Anzahl Byte

Code/Data	Funktion
22h 9fh 0ah 00h	**Anforderung: Speicherplatz reservieren**
s	Strategie: s=0: nur Größe des freien Speichers melden s=1: von unten nach oben, so viel wie angegeben s=2: von unten nach oben, so viel wie möglich, aber nicht mehr als angegeben s=3: von oben nach unten, so viel wie angegeben s=4: von oben nach unten, so viel wie möglich, aber nicht mehr als angegeben
z	Alignment: Der Anfang des zu reservierenden Bereichs kann auf eine Grenze gelegt werden, deren Adresse durch eine 2er Potenz teilbar ist: z=0: Byte, z=1: Wort, z=2: Doppelwort, ...
tL tH hL hH	t = Tasknummer (nur für statistische Zwecke) h = Speichernutzung (nur für statistische Zwecke) (dyn., stat., Code, Daten, Parameter, etc.)
nE nF nG nH	Größe des zu reservierenden Bereichs (in Anzahl Byte)
22h	Antwort:
gE gF gG gH	Größe des reservierten Bereichs (s=1, 2, 3, 4) bzw. des freien Speichers (s=0)
aE aF aG aH	Anfangsadresse des reservierten Bereichs (physikal. Adresse)

12.5. Systemzugriffe: I/O (privilegierte Befehle)

Code/Data	Funktion
27h 22h pL pH	**Anforderung: Byte von 8 Bit breitem Port lesen** p = Port (I/O-Adresse)
27h b	Antwort
27h 32h pL pH	**Anforderung: Wort von 16 Bit breitem Port lesen** p = Port (I/O-Adresse)
27h wL wH	Antwort
23h 03h pL pH b	**Byte an 8 Bit breiten Port schreiben** p = Port (I/O-Adresse), b = Byte
23h 04h pL pH wL wH	**Wort an 16 Bit breiten Port schreiben** p = Port (I/O-Adresse), w = Wort

12.6. Die Taskbefehle

Erläuterung zu den Abkürzungen:

n	= Anzahl (0 bis 255)
nL nH	= Anzahl, Low-Byte, High-Byte (0 bis 65280)
aE aF aG aH	= 4 Byte-Adresse, E = Low-Byte, H = High-Byte
b	= Byte
b1..bn	= n Byte (Byte 1 bis Byte n)
c	= Code für Makrobefehl
dE dF dG dH	= Datenbereich Länge, E = Low-Byte, H = High-Byte
f	= Flag
pL pH	= Programm-Nummer
(p)	= Parameternummer (variables Format): Langform (2 Byte): pL pH (0 bis 65280) Kurzform (1 Byte): p (0 bis 255) Taskanwahl (1 Byte): p (0 bis 255)
rL rH	= relative Adresse, Low-Byte, High-Byte
rE rF rG rH	= relative Adresse (4 Byte, E = Low-Byte, H = High-Byte)
(t)	= Tasknummer (optional, variables Format): Langform (2 Byte): tL tH (0 bis 1023) Kurzform (1 Byte): t (0 bis 255) Taskanwahl (0 Byte): Angabe der Tasknr. entfällt
vL vH	= Wort: Low-Byte, High-Byte
wL wH	= Wort: Low-Byte, High-Byte
z	= Prozedur-Nr. (0 bis 255)
sL sH	= Stations-Nr. (0 bis 65280)

12.6.1. Das Prinzip

Das Betriebssystem ordnet jeder Task, wenn ein Programm installiert wird, einen Parameter- und einen Datenbereich zu. Die Länge beider Bereiche kann auch = 0 sein. Die Länge des Parameterbereichs wird vom Programm selbst vorgegeben und ist damit konstant. Für die Angabe der Länge des Datenbereichs gibt es verschiedene Möglichkeiten, sie kann auch vom Anwender beim Installieren angegeben werden.

Das Programm selbst besteht aus allgemein aufrufbaren Prozeduren bzw. Funktionen, z.B. vom PC oder von einer anderen Task aus.

Vom PC aus kann mit Makrobefehlen auf die Strukturen (Prozeduren, Funktionen, Parameter, Daten) jeder Task zugegriffen werden: Es können Funktionen aufgerufen werden, Parameter gesetzt und gelesen werden und Daten geschrieben oder gelesen werden.

12.6.2. Datenbereich

Der Datenbereich einer Task ist immer ein durchgängiger Bereich ohne Lücken oder Überlappungen. Der Zugriff erfolgt immer indirekt über Pointer: einen Schreib-Pointer (W-Pointer) und einen Lese-Pointer (R-Pointer). Beide Pointer sind der Task, zu der der Datenbereich gehört, fest zugeordnet. Alle Tasks können Anfang und Ende des Datenbereichs jeder Task aber ermitteln und natürlich auch eigene Pointer halten.

Für das Lesen und Schreiben von Daten von anderen Tasks oder vom jeweiligen Host aus stehen Funktionen zur Verfügung:

- R-Pointer auf Anfang setzen
- R-Pointer verschieben (vorwärts/rückwärts)
- W-Pointer auf Anfang setzen
- W-Pointer verschieben (vorwärts/rückwärts)
- Block aus Bereich lesen und R-Pointer um n inkrementieren
- Block in Bereich schreiben und W-Pointer um n inkrementieren

Bei einem linearen Puffer erfolgt nur beim Schreiben eine Bereichsüberprüfung.

12.6.3. Befehle zur Installierung und Taskverwaltung

12.6.3.1. Makrobefehl 40h zum Installieren

Vom PC aus muß ein Programm mit diesem Makrobefehl unter einer Task installiert werden. Auf der Karte von einer anderen Task aus geschieht das durch Aufruf der System-Routine INSTALL. Hierbei sind verschiedene Optionen möglich.

Code/Data	Funktion
40h 0fh	**Installiere Programm** (Erklärung siehe unten)
12h 00h	
tL tH	t = Tasknummer
pL pH	p = Programm-Nummer
iL 00h	i = Interrupt-Nummer
fE fF fG fH	f = Flag
dE dF dG dH	d = Größe des zu reservierenden Datenbereichs
aE aF aG aH	a = Adresse

Erläuterungen zum Makrobefehl 40h zum Installieren

t = Tasknummer (Wort)

Dies ist die Tasknummer, unter der das Programm installiert werden soll. Erlaubt sind Angaben von 1 bis 1023. Task 0 ist immer für das Betriebssystem reserviert. Task 1 bis 15 sollten nicht für Anwendungsprogramme benutzt werden. Aus Geschwindigkeitsgründen ist es sinnvoll, zunächst Tasknummern kleiner 256 zu verwenden.

p = Programm-Nummer (Wort)

Die Programm-Nummer kann zwischen 1 und 65535 betragen. Die im Makrobefehl angegebene Nummer muß mit der Nummer in der PDT übereinstimmen. Programm-Nummer 0 ist reserviert für das Betriebssystem, Programm-Nummer 0ffffh (= -1) ist ein Dummy-Programm ohne Funktion. Sie wird gemeldet, wenn kein Programm unter einer Task installiert ist. Die Programm-Nummer erscheint auch als 4stellige hexadezimale Zahl im Namen von Programmdateien und in dazugehörigen Dateien:

M2Pnnnn.LIB = Relokiertes Anwenderprogramm
M2Pnnnn.SEX = Sorcus EXe-Datei (identisch mit .EXE)
M2Pnnnn.SRX = Sorcus Relokierte eXe-Datei
M2Pnnnn.SHX = Sorcus HyperteXt-Datei

i = Interrupt-Nummer (Byte), siehe Anhang C

Bei NI-Tasks wird diese Angabe nicht ausgewertet, nur unter bestimmten Voraussetzungen bei Interrupt-Tasks (II- und DI-Tasks). Die Interrupt-Nummer legt fest, welcher Interrupt der Task, die gerade installiert wird, zugeordnet wird. Wenn dieser Interrupt auftritt, wird die Hauptprozedur des Programmes aufgerufen, das unter der Task installiert ist.

Für die Festlegung des Interrupts gibt es zwei Möglichkeiten: durch die PDT (= Programm-Deskriptor-Tabelle) oder mit Makrobefehl 40h zum Installieren (siehe auch bei "Tasktyp"). Hierfür sind zwei Bit zuständig: Bit 3 im Flag in der PDT und Bit 3 im Flag im Makrobefehl. Dies sind die gleichen Bit, die auch für die Festlegung des Tasktyps verwendet werden.

f = Flag (Doppelwort = 32 Bit)

Die folgende Tabelle gibt eine Kurzerklärung für die Verwendung der 32 Bit in Flag.

Bit-Nr.	Anzahl Bit	Erklärung
0 - 2	3	Tasktyp (NI-, II-, DI -Task)
3	1	Wer legt Tasktyp und Interrupt-Nr. fest?
4, 5	2	Privilegstufe des Programmes
6 - 8	3	Programm im RAM oder ROM? In welchem Format?
9, 10	2	Wie wird Größe des Datenbereichs festgelegt?
11	1	Auto-Init-Prozedur nach Installieren aufrufen?
12	1	Task sofort nach Installieren aktivieren?
13	1	Debug-Informationen enthalten?
14	1	Typ des Debuggers
15	1	Tasknr. vom Befehl oder vom OS vergeben?
16 - 20	5	Reserviert
21	1	Zwangs-Caching von Parameterbereich
22	1	Zwangs-Caching von TDT
23	1	Zwangs-Caching von Datenbereich
24 - 31	8	Zwangs-Caching von Code: nur Prozedur 0 - 7

Tasktyp: NI-, DI-, II-Task (Bit 0 bis 2)

Hiermit wird angegeben, welchen Typ die installierte Task haben soll:

0 (=000b): NI-Task (Nicht-Interrupt-Task)
1 (=001b): II-Task (Indirekte Interrupt-Task)
2 (=010b): DI-Task (Direkte Interrupt-Task)

Für die Festlegung des Tasktyps (und der Interrupt-Nummer) gibt es zwei Möglichkeiten: entweder durch das zu installierende Programm selbst oder mit Makrobefehl 40h zum Installieren. Hierfür sind zwei Bit zuständig: Bit 3 im Flag in der PDT und Bit 3 im Flag im Makrobefehl. Diese Bit werden auch für die Festlegung der Interrupt-Nummer verwendet.

Wer legt Tasktyp und Interrupt-Nummer fest? (Bit 3)

Wenn Bit 3 im Flag der PDT = 1 gesetzt ist, wird dieses Bit ignoriert und Tasktyp und Interrupt-Nummer werden in jedem Fall aus der PDT verwendet. Die Angaben "Tasktyp" und "Interrupt-Nummer" im Makrobefehl werden verworfen.

Wenn Bit 3 im Flag der PDT = 0 gesetzt ist, erlaubt das Programm, beide Angaben auch per Makrobefehl festzulegen. Hierzu muß dieses Bit = 1 gesetzt werden. Natürlich müssen die Angaben von Tasktyp und Interrupt-Nummer gültige Werte aufweisen. Wenn dieses Bit = 0 ist, werden wiederum die Angaben für Tasktyp und Interrupt-Nummer aus der PDT verwendet.

Privilegstufe des Programmes (Bit 4 und 5)

0 (=00b) ist die höchste Privilegstufe, 3 (= 11b) die niedrigste. Anwendungsprogramme haben immer die Privilegstufe 3, Systemprogramme die höchste, also 0. Diese Flags werden bei Multi-LAB/2 nicht ausgewertet und können immer = 0 gesetzt werden.

Programm im ROM oder RAM und Programmformat (Bit 6 bis 8)

Wenn das zu installierende Programm im RAM steht, wird die Adresse a als absolute physikalische Adresse angesehen, ihre Bedeutung richtet sich danach, in welchem Format das Programm im RAM der Karte steht. Dies wird dem Betriebssystem über Bit 6 bis 8 im Flag des Makrobefehls mitgeteilt:

Wenn das zu installierende Programm im ROM vorhanden ist, wird die im Makrobefehl angegebene Adresse a ignoriert, entscheidend ist dann die im Makrobefehl angegebene Programm-Nummer.

Wo?	Programmformat	Adresse a	Format Adresse	Flag-Bit *) 8 7 6
RAM	PDT tiny	Anfang PDT	physikal.	0 0 0
RAM	PDT large	Anfang PDT	physikal.	1 0 0
RAM	EXE nicht relocated	Anfang EXE-Header	physikal.	1 1 0
RAM	EXE relocated	PREPARE-Code	Seg:Offs.	0 1 0
ROM	PDT	wird ignoriert	-	0 0 1

*) alle anderen Kombinationen sind zur Zeit nicht erlaubt

Wie wird die Größe des Datenbereichs festgelegt? (Bit 9 und 10)

Hierfür gibt es mehrere Möglichkeiten (siehe folgende Tabelle). Wenn Bit 9 im Flag der PDT = 1 gesetzt ist, kann die Größe durch den Makrobefehl nicht verändert werden. Der Normalfall ist der, daß die Größe durch d im Makrobefehl beim Installieren angegeben wird. Die maximal und minimal mögliche Größe kann durch die Vorgaben in der PDT eingeschränkt werden, falls ein Programm z.B. nur einen max. 64 KByte großen Datenbereich unterstützt.

fix/ variabel	Größe richtet sich nach	Flag in PDT Bit 9	Flag in Makrobefehl Bit 10	Bit 9
fix	"Größe" in PDT	1	x	x
fix	"Größe" in PDT	0	1	1
fix	"Minimum" in PDT	0	1	0
fix	"Maximum" in PDT	0	0	1
variabel	d in Makrobefehl	0	0	0

Auto-Init aufrufen? (Bit 11)

Wenn dieses Bit = 1 gesetzt ist, wird unmittelbar nach dem Installieren sofort die Auto-Init Prozedur (Prozedur 1) aufgerufen. Wenn das Bit = 0 gesetzt ist, nicht.

Task nach Installieren sofort aktivieren? (Bit 12)

Wenn dieses Bit = 1 gesetzt ist, wird nach dem Installieren und ggf. nach dem Aufruf der Auto-Init Prozedur die Task auch sofort aktiviert. Wenn das Bit = 0 gesetzt ist, nicht.

Reserviert (Bit 13 bis 31)

d = Größe Datenbereich (Doppelwort)

Die Größe wird in Anzahl Byte angegeben. Weitere Erklärungen finden sich beim Flag zu Bit 9 und 10 (s.o.).

a = Adresse (Doppelwort)

Die Adresse wird als physikalische oder Segment:Offset-Adresse angegeben, siehe Flag, Bit 6 bis 8. Bei Verwendung der mitgelieferten PC-Bibliothek wird die Formatanpassung automatisch übernommen.

12.6.3.2. Makrobefehle zur Taskverwaltung

Code/Data	Funktion
3ah 31h i	**Anforderung: Melde die Nummer der Task, die den Interrupt i nutzt** i = Interrupt-Nr.
3ah tL tH	Antwort: Befehls-Code Tasknummer (wenn t = -1, dann ist der Interrupt i nicht benutzt)
4eh 64h nL nH pL pH	**Melde die Task, unter der Programm p installiert ist** n = Ordnungszahl der gesuchten Installierung p = Programmnummer
4eh s mL mH tL tH	Antwort Status: s = 0: alles ok, s = -1 (ffh) Programm ist nicht installiert m = Ordnungszahl einer gefundenen Installierung (m ≤ n) t = Tasknummer der m.-ten Installierung
41h 02h tL tH	**Aktivieren einer Task** tL, tH = Tasknr. Eine NI-Task kann auch mehrfach aktiviert werden. Dadurch wird das installierte Programm häufiger aufgerufen. Bei II-Tasks und DI-Tasks hat der Makrobefehl keine Wirkung.
42h 02h tL tH	**Deaktivieren einer Task** tL, tH = Tasknr. (Task 0 ist nicht erlaubt) Eine mehrfach aktivierte NI-Task muß auch mehrfach wieder deaktiviert werden. Ein laufendes Programm wird abgebrochen, ein gerade laufender Aufruf wird ordnungsgemäß beendet. Bei II-Tasks und DI-Tasks hat der Makrobefehl keine Wirkung.
43h 22h tL tH	**Anforderung: Melde, ob Task t aktiviert ist** tL, tH = Tasknr.
43h n	Antwort: n = Anzahl Aktivierungen (n = 0: Task ist nicht aktiviert)

12.6.4. Zugriff auf Parameter

Während des Zugriffs mit den Blockbefehlen sind die Parameter nicht vor dem gleichzeitigen Zugriff z.B. durch eine andere Interrupt-Task geschützt. Umgekehrt kann der PC-Makrobefehl eine andere Task unterbrechen, die ebenfalls gerade auf dieselben Parameter zugreift, sofern diese Task das nicht unterbindet, z.B. durch vorübergehendes Sperren des maskierbaren Interrupts.

Code/Data	Funktion
58h 24h tL tH pL pH	**Anforderung: Parameter Byte lesen** t = Task-Nummer p = Nummer des zu lesenden Parameters
58h b	Antwort: b = gelesenes Parameter Byte
59h 34h tL tH pL pH	**Anforderung: Parameter Wort lesen** t = Task-Nummer p = Nummer des zu lesenden Parameters
59h wL wH	Antwort: w = gelesenes Parameter Wort
5ah 54h tL tH pL pH	**Anforderung: Parameter Doppelwort lesen** t = Task-Nummer p = Nummer des zu lesenden Parameters
5ah dE dF dG dH	Antwort: d = gelesenes Parameter Doppelwort
4ch f4h mL mH tL tH pL pH	**Anforderung: Parameter Block lesen** m = Anzahl zu lesender Parameterbyte t = Task-Nummer p = Nummer des ersten zu lesenden Parameters
4ch b1...bm	Antwort: Data
5ch 05h tL tH pL pH b	**Parameter Byte schreiben** t = Task-Nummer p = Nummer des zu schreibenden Parameters b = Parameter Byte

Code/Data	**Funktion**
5dh 06h tL tH pL pH wL wH	**Parameter Wort schreiben** t = Task-Nummer p = Nummer des zu schreibenden Parameters w = Parameter Wort
5eh 08h tL tH pL pH dE dF dG dH	**Parameter Doppelwort schreiben** t = Task-Nummer p = Nummer des zu schreibenden Parameters d = Parameter Doppelwort
4dh 0fh nL nH tL tH pL pH b1...bn-4	**Parameter Block schreiben** n = Anzahl zu schreibender Parameterbyte + 4 t = Task-Nummer p = Nummer des ersten zu schreibenden Parameters Data

12.6.5. Aufruf einer Prozedur bzw. Funktion

Beim Aufruf einer **Prozedur** einer Task werden keine Parameter an sie übergeben und keine von ihr zurückgeliefert.

Beim Aufruf einer **Funktion** können Parameter hin und zurück übergeben werden (Anzahl Hin und Rück werden immer als Anzahl Byte angegeben und können auch = 0 sein):

Bezüglich der an die Funktion übergebenen Parameter meldet die Funktion zurück, wie viele Byte sie davon verwerten konnte. Sie beginnt mit der Verwertung sequentiell ab dem zuerst übergebenen Parameterbyte.

Beim Aufruf kann außerdem angegeben werden, wie viele Parameter (Anzahl Byte) von der Funktion maximal zurückerwartet werden. Die Funktion meldet immer zurück, wie viele Byte sie tatsächlich liefern konnte.

Code/Data	Erklärung
45h 04h tL tH fL fH	**Aufruf der Prozedur f einer Task** (es werden keine Parameter an die Prozedur übergeben oder von ihr zurückerwartet)
44h ddh nL nH mL mH tL tH fL fH b1...bn-4	**Aufruf der Funktion f einer Task** Anzahl Byte Hin (= Anzahl der an die Funktion übergebenen Parameterbyte + 4), max. 256 + 4 Anzahl der max. zurückerwarteten Byte, max. 256 Tasknr. Funktions-Nr. Parameter, die an die Funktion übergeben werden
44h error wL wH vL vH b1...bv	Antwort: error = 0: kein Fehler[1] Anzahl der verworfenen Parameterbyte Hin Anzahl der tatsächlich zurückgelieferten Byte Parameter, die von der Funktion zurückgeliefert werden

[1] Die in error übergebenen Fehlercodes entsprechen den übrigen Fehlermeldungen, allerdings wird hier nur der Fehlertyp gemeldet (siehe Anhang F), sofern die von der aufgerufenen Funktion gelieferte Fehlermeldung der Gruppe e0h ("Fehler bei Aufruf der Funktion") angehört, andernfalls wird error = 1ah ("unbekannter Fehler") gemeldet.

12.6.6. Zugriff auf Datenbereich

Der Zugriff auf den Datenbereich einer Task ist nur indirekt über Daten-Pointer möglich. Für den Zugriff vom PC aus werden die taskeigenen Pointer (je ein R-Pointer und W-Pointer pro Task) der angesprochenen Task verwendet. Dabei ist zu beachten, daß die gleichen Pointer auch von den lokalen System-Subroutinen auf der Karte verwendet werden (siehe Kapitel 9 und 10).

Der Datenbereich ist während des Zugriffs mit den Blockbefehlen nicht vor dem gleichzeitigen Zugriff durch eine andere Task geschützt, z.B. durch eine Interrupt-Task. Umgekehrt kann der PC-Makrobefehl eine andere Task unterbrechen, die ebenfalls gerade auf denselben Datenbereich oder einen der Pointer zugreift. Eine einfache Methode, solche Konflikte zu verhindern, besteht darin, den maskierbaren Interrupteingang der CPU vorübergehend zu maskieren.

Code/Data	Funktion
46h 02h tL tH	**Setze R-Pointer auf den Anfang des Datenbereichs** t = Task-Nummer
47h 06h tL tH nE nF nG nH	**Verschiebe R-Pointer um n Byte** t = Task-Nummer n = Anzahl Byte (2er Komplement)
48h 02h tL tH	**Setze W-Pointer auf den Anfang des Datenbereichs** t = Task-Nummer
49h 06h tL tH nE nF nG nH	**Verschiebe W-Pointer um n Byte** t = Task-Nummer n = Anzahl Byte (2er Komplement)
50h 22h tL tH	**Anforderung: Lies Datenbyte** von (R-Pointer) und inkrementiere danach R-Pointer um 1 t = Task-Nummer
50h b	Antwort: b = Datenbyte
51h 32h tL tH	**Anforderung: Lies Datenwort** von (R-Pointer) und inkrementiere danach R-Pointer um 2 t = Task-Nummer
51h wL wH	Antwort: w = Datenwort

Code/Data	Funktion
52h 52h tL tH	**Anforderung: Lies Daten-Doppelwort** von (R-Pointer) und inkrementiere danach R-Pointer um 4 t = Task-Nummer
52h dE dF dG dH	Antwort: d = Daten-Doppelwort
4ah f2h mL mH tL tH	**Anforderung: Lies Datenblock** von (R-Pointer) und inkrementiere danach R-Pointer um m t = Task-Nummer
4ah b1...bm	Antwort: Datenbyte (Blockgröße = m Byte)
54h 03h tL tH b	**Schreibe Datenbyte** an (W-Pointer) und inkrementiere danach W-Pointer um 1 t = Task-Nummer b = Datenbyte
55h 04h tL tH wL wH	**Schreibe Datenwort** an (W-Pointer) und inkrementiere danach W-Pointer um 2 t = Task-Nummer w = Datenwort
56h 06h tL tH dE dF dG dH	**Schreibe Daten-Doppelwort** an (W-Pointer) und inkrementiere danach W-Pointer um 4 t = Task-Nummer d = Daten-Doppelwort
4bh 0fh nL nH tL tH b1...bn-2	**Schreibe Datenblock** an (W-Pointer) und inkrementiere danach W-Pointer um (n - 2) t = Task-Nummer (Blockgröße = n - 2 Byte)

12.7. System-Call: Taskinformationen (privilegierter Befehl)

Code/Data	Befehl
2fh 54h tL tH nL nH	**Anforderung: Task-Info** Tasknr. (t = 0: Betriebssystem) Call-Nr. (Erklärung siehe Anhang H)
2fh aE aF aG aH	Antwort: 4 Byte Ergebnis, z.B. Adresse

12.8. EEPROM und Kopie davon

Wenn das EEPROM mit den unten angegebenen Makrobefehlen direkt angesprochen wird, dann darf keine Task auf der Karte aktiv sein. Deshalb muß vor einem direkten Zugriff auf das EEPROM ein Reset der Karte gemacht werden. Wenn das EEPROM neu beschrieben wurde, wird die neue Information erst nach einem Hardware-Reset der Karte vom Betriebssystem verwendet bzw. steht im Parameterbereich des Betriebssystems zur Verfügung, so daß nach dem Schreiben noch einmal ein Reset der Karte ausgeführt werden sollte.

Code/Data	Funktion
28h 32h 00h a	**Lesen eines Wortes direkt aus dem EEPROM** Rel. Adresse des zu lesenden Wortes im EEPROM (a = 0 bis 63)
28h wL wH	Antwort: EEPROM-Wort
29h 04h 00h a wL wH	**Schreiben eines Wortes direkt in das EEPROM** Rel. Adresse des zu schreibenden Wortes im EEPROM (a = 0 bis 63) Zu schreibendes Wort

Die gesamte EEPROM-Information steht nach einem Reset im Parameterbereich des Betriebssystems zur Verfügung, sofern das Lesen des EEPROMs ohne Probleme erfolgte. Dafür ist ab Parameter 100 ein Block von 4 Byte vorgesehen, der die EEPROM-Information beschreibt. Die ersten beiden Byte enthalten die Anzahl der gültigen Wörter, die vom Betriebssystem aus dem EEPROM gelesen wurden, die letzten beiden enthalten die Parameternummer, ab der der EEPROM-Inhalt selbst steht. Wenn die Anzahl = 0 ist, konnte das EEPROM nicht gelesen werden, statt der Parameternummer wird dann eine Fehlerinformation geliefert (siehe Übersicht Fehlermeldungen im Anhang F).

12.9. Kontroll-LED

Code/Data	Funktion
30h (02h)	**LED ein**
31h (02h)	**LED aus**
33h 20h	**Anforderung: Zustand der LED**
33h b	Antwort: Zustand der LED (Bit 0: 0 = aus, 1 = ein)