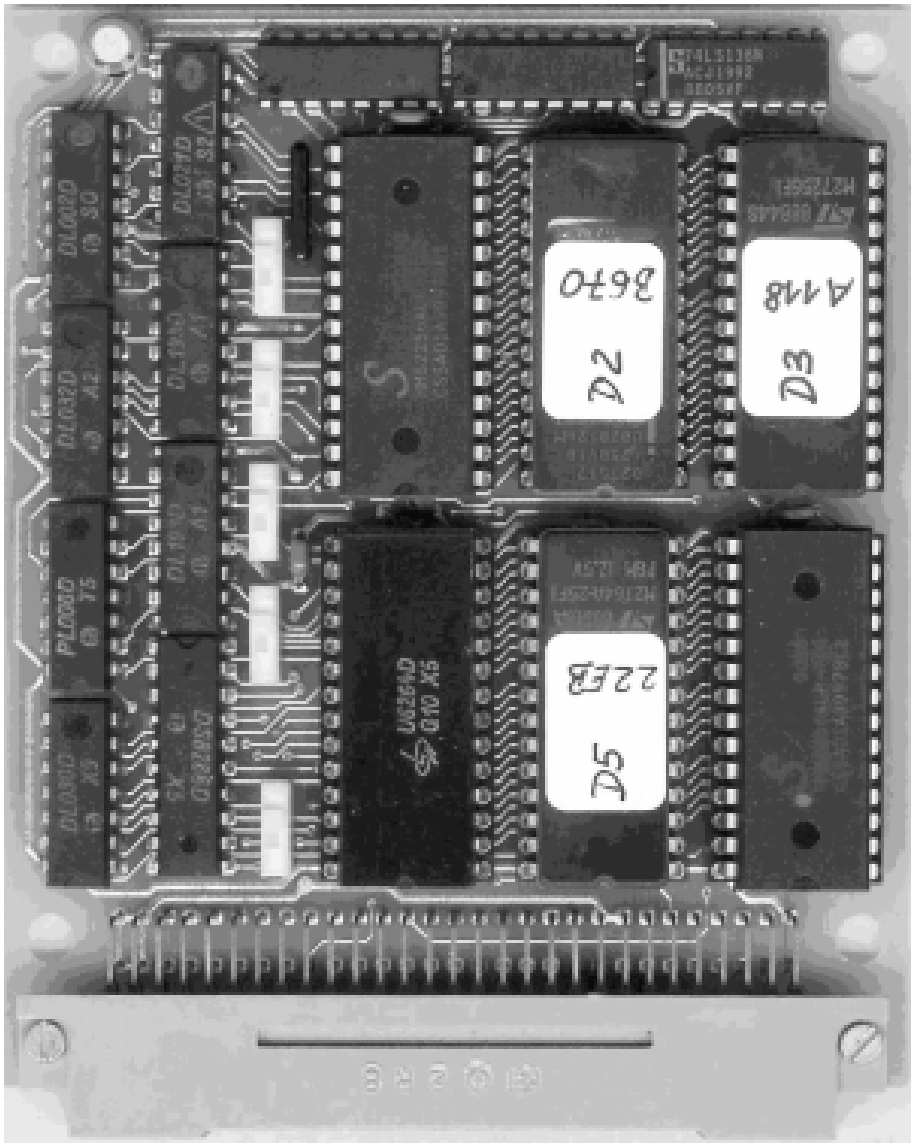
192 KByte RAM/EPROM-Modul

Beschreibung aller Softwarekomponenten



digitalisiert : Ulrich Zander 6/2013 < zande@felix.sax.de>

VORWORT

Wir schreiben März 1991 und befinden uns im Zeitalter der "Abwicklung". In der Öffentlichkeit ist der Begriff 8-Bit-Rechner bereits seit langer Zeit abgewickelt. Die zahlreichen Personalcomputer oder CAD-Stationen wie sie einmal liebevoll genannt wurden verschenkt man fast, damit westliche Investoren beim Anblick dieser Altlasten keinen Schock bekommen. Den Arbeitnehmern, die diese Schlüsseltechnologien besser beherrscht haben als sich die zugereisten Arbeitgeber vorstellen können, hat man die Arbeit genommen.

Neue Computer kommen auf uns zu, aber keiner kann sie bezahlen. Und wer das Geld dafür besorgt hat, findet sich im Dschungel der Grafikkarten, Monitore, Mäuse usw. nicht zurecht. Dabei hilft zum Glück der Händler, den man auch Experte nennen kann. Er kann die unzähligen neuen Fachbegriffe bei der Kundenberatung auch treffsicher einsetzen. Der Software, für die vor kurzem noch 64 KByte reichten, gelingt es immer besser Megabyte und Farbgrafik vorauszusetzen. Es ist "in" staunend vor dem Rechner zu sitzen und irgendwelche bunten Bilder oder Sternenhimmel über sich ergehen zu lassen. Wie armselig fühlen sich da die obengenannten Arbeitnehmer. Sie besuchen eine vom Arbeitsamt bezuschußte Bildungsmaßnahme (Computerkurs) um ihr Selbstbewußtsein wiederzufinden.

Es gibt einige Menschen, für die die Computertechnik der Lebensinhalt war und bei denen es die "Umstände" bisher nicht geschafft haben sich zu "wenden". Diese Mitmenschen haben sich über viele Jahre unter gigantischem persönlichen Einsatz ihren eigenen Heimcomputer zusammengebaut oder versucht aus einem durch Beziehungen erworbenen Gerät irgendwann einmal etwas sinnvolles zu machen. Jeder hat sich seine eigene Suppe gekocht. Die Hersteller wurden mit anderen Dingen wie der "von oben" verordneten Umbenennung des Heimcomputers Z 9001 in Kleincomputer KC 85/1 beschäftigt. In allen Beschreibungen mußten kurzfristig diese Zeichenketten ausgetauscht werden. Das wäre ja ganz einfach gewesen, wenn ein Computer mit Textverarbeitungssoftware zur Verfügung gestanden hatte.

Ich war auch von dieser Misere betroffen, denn ich habe anläßlich des 35. Jahrestages der DDR am 12. Oktober 1984 einen von den allerersten Z 9001 erkämpft. Das entwickelte sich in meinem Leben zu einem wahrhaft historischen Ereignis. Ich lernte BASIC und Assembler programmieren, denn sonst gab es ja nichts. Ich erfand auch zahlreiche Zusatzmodule, weil sich davon 4 Stück an den Z 9001 anschließen lassen und die Stromversorgung unerschöpflich ist. Mit Grafik und Farbe war nicht so viel los, also interessierte ich mich vorsichtshalber gar nicht dafür. Mein Heimcomputer hat meine Kreativität ganz erheblich angeregt und sehr viel von meiner Zeit verbraucht.

Der Computer und ein 1985 für unwahrscheinlich viel Geld angeschaffter Drucker waren auch der Grund, daß ich auf Wunsch des Herstellers VEB Robotron Meßelektronik "Otto Schön" meinen Arbeitsplatz dorthin verlegte. Endlich konnte ich mit allen Entwicklern zusammentreffen und brauchte nicht auf irgendwelche Veröffentlichungen zu warten. Gemeinsam gingen wir dann zum Staatsplanthema Bildungscomputer A 5105 über. Meine intensive Hobbytätigkeit am Z 9001 versetzte mich in die Lage, dort nicht nur mitreden zu können sondern auch viele eigene Ideen einzubringen. Bei dieser Arbeit lernte ich das Betriebssystem SCP von innen heraus genauestens kennen.

Die Entwicklung wurde im Sommer 1989 erfolgreich abgeschlossen und als wir gerade den Dank der Ministerin erwarteten, wurde die Serienproduktion begonnen, um kurz danach eingestellt zu werden. War die Entwicklung nun sinnlos? Für mich nicht, denn ich habe dabei außerordentlich viel gelernt und eine interessante, anspruchsvolle Arbeit gemacht. Allerdings kann ich mit diesem Spezialwissen zur Zeit kein Geld verdienen. Das Leben ist hart aber ungerecht.

Bei den Arbeiten am Z 9001 kamen mir immer Ideen, für die ich Bauelemente gebraucht hätte, die es bei uns nicht gab. So entstand die Idee zu dem heute vorliegenden 192 KByte RAM/EPROM-Modul schon lange vor der "Wende“.

Nachdem dann alle Bauelemente der Welt verfügbar waren, habe ich das Modul kurzfristig produzieren lassen, um mein Lebenswerk auch einmal in den Händen halten zu können.

Die von mir dafür entwickelte Systemsoftware hat mich so stark gefesselt, daß ich einfach nicht davon lassen kann. Während der gesamten Entwicklung habe ich mich von nichts und niemandem, auch nicht durch Computerclubs beeinflussen lassen. Das Ziel war CP/M-Kompatibilität, ohne die Kompatibilität zu den Z9001-Programmen aufzugeben und die uneingeschränkte Verwirklichung meiner Ideen. Dabei habe ich niemals unter irgendwelchem Zeitdruck gestanden und jede Einzelheit solange verbessert, bis sie mir gefiel. Es ist kein Geheimnis, daß die gesamte Software selbstverständlich mit dem Z 9001 und dem sich schrittweise entwickelnden 192 KByte RAM/EPROM-Modul entstanden ist.

Meine Entwicklungen habe ich erstmals 1990 im "Funkamateur" veröffentlicht und daraufhin eine wahre Postflut ins Haus bekommen. Zur Zeit beantworte ich meistens Briefe, anstatt an der Software weiterzuarbeiten. Das ist auch der Grund, daß sich CP/M noch nicht in einem Zustand befindet, der verbreitet werden kann. Die technischen Probleme sind gelöst, jetzt ist es nur noch eine Zeitfrage, bis CP/M auch bedienbar ist.

Mit den vorhandenen Softwarekomponenten läßt sich aber auch schon sehr viel anfangen. Die dafür unerläßliche Beschreibung liegt hiermit vor. Das Ziel dieser Beschreibung ist, die Handhabung aller Kommandos zu vermitteln und einen Überblick über das Gesamtsystem zu geben. Für eigene Programmierung unter Nutzung aller Systemunterprogramme und sonstiger Spezialitäten ist die vorliegende Beschreibung sicherlich nicht ganz ausreichend.

Die Softwarekomponenten sind etwa in der Reihenfolge ihrer Entstehung bzw. Bedeutung beschrieben worden. Zwischendurch bin ich auf alles eingegangen was mir nebenbei noch einfiel. Manchmal sind Hardwareinformationen oder Begriffserklärungen eingefügt worden. Einige Bemerkungen traten auch wiederholt auf, wenn sie mir besonders wichtig erschienen.

Ich hoffe, daß jeder Leser nach einer Gewöhnungsphase die gewünschten Informationen schnell findet und mit dem 192 KByte RAM/EPROM-Modul gut zurechtkommt. An Hinweisen und Meinungsäußerungen bin ich interessiert, auch wenn dieses Thema keine große Zukunft mehr hat.

Freital, im März 1991
Lutz Elßner

Inhaltsverzeichnis

Softwarekomponente 1 BO - Systemanlauf (BOOT)

PROGRAMMBESCHREIBUNG

Zweckbestimmung

Diese Softwarekomponente dient zum Start des speziellen Betriebssystems für das 192 KByte RAM/EPROM-Modul. Dieses Modul ist so aufgebaut, daß die Speicherbereiche C000-DFFF und E000-E7FF jeweils aus 16 möglichen Speicherbanken beliebig miteinander kombiniert werden können. In beiden Adreßbereichen befinden sich Bestandteile der Komponente BO, die teilweise die gleichen Funktionen ausführen. Das ist erforderlich, um einen Kommandonamen oder eine Speicherzelle in allen Bänken des jeweils anderen Adreßbereiches aufzufinden. Solche Aktionen laufen bei gesperrtem Interrupt (DI) ab, weil gegebenenfalls der Stack und die Interrupttabelle dabei weggeschaltet werden.

Außerdem wird das Kommando HELP zur Anzeige aller Kommandonamen realisiert.

Systemanlauf

Bei RESET oder Einschalten des Computers wird das Speicherbankregister auf dem 192 KByte RAM/EPROM-Modul hardwaremäßig auf 00 gelöscht. Das Betriebssystem Z 9001 sucht nach der eigenen Initialisierung den Kommandonamen '# ' im Speicher und verzweigt dorthin, anstatt zur eigenen Kommandoeingabe zu springen. Auf Adresse E000H in Bank 0 stehen der genannte Kommandoname und die Startroutine.
Zuerst wird der andere Teil der Komponente BO im Adreßbereich C000-DFFF anhand des Namens '## in allen 16 Bänken gesucht und dort hingesprungen. Dort werden wichtige Systemzellen in den Speicherbereichen:

0000H - 003AH Restart-Adressen (je 3 Bytes ab 00H, 08H, 10H usw.)
0040H - 005BH Bankrückschaltungsprogramme
02C0H - 02FFH Programmanfänge der Restarts
EFF1H - EFFFH Merkzellen für Bankumschaltung, Interruptrückkehr

initialisiert. Anschließend werden die unbedingt erforderlichen Softwarekomponenten UP und BD anhand der Namen '#UP ' und '#BD ' initialisiert. Ab diesem Zeitpunkt sind die darin enthaltenen Routinen über RST 20H und RST 08H verfügbar. Genaue Angaben sind den Beschreibungen UP und BD zu entnehmen.

Wird ein bisher gesuchter Name nicht gefunden, erfolgt Systemabbruch. Dazu wird zwangsweise eine bestimmte Bank (meistens die mit dem BASIC-Interpreter) eingeschaltet und zur Adresse F003H (Warmstart Z 9001 im ROM) gesprungen. Die automatische Startroutine kann nicht wieder gefunden werden, weil sie in der Bank 0 steht. In diesem Falle kann so gearbeitet werden, als wären gar keine Speicherbänke vorhanden und nur das BASIC-Modul gesteckt. Mit Hilfe des Kommandos '^' aus der Komponente CCP kann ein solcher Zustand auch absichtlich eingenommen werden.

Automatische Kaltstartinitialisierung

Nach erfolgreicher Initialisierung der Komponenten UP und BD werden automatische Kaltstartinitialisierungen durchgeführt. Dazu werden alle Kommandonamen, die mit einem '$' beginnen, gesucht und abgearbeitet. Die Namen werden am Bildschirm angezeigt. Wieviele solcher Routinen vorhanden sind, hängt von den anderen im gesamten 192 KByte RAM/EPROM-Modul enthaltenen Programmen ab. Diese können vom Anwender beliebig erweitert werden, ohne das Programm BO zu verändern. Innerhalb dieser Routinen erfolgt im Allgemeinen die Initialisierung zusätzlicher Hardware (Floppy, V24, BIOS) und die Eintragung in die Adreßtabellen der Komponenten UP und BD. Bei Fehlern während der Initialisierung (z.B. Zusatzmodul nicht gesteckt) wird die Routine mit gesetztem CARRY-Flag verlassen und hinter dem Kommandonamen 'nicht bereit' angezeigt.

Warmstart

Nach der Regenerierung des Stackpointers und einiger konstanter RAM-Bereiche werden ähnlich wie beim Kaltstart alle Kommandonamen, die mit '%' beginnen gesucht und abgearbeitet. Hier werden die Namen nur im Fehlerfalle mit den Zusatz 'nicht bereit' angezeigt. Welche Warmstartinitialisierungsroutinen vorhanden sind und in welcher Reihenfolge sie abgearbeitet werden, kann mit dem Kommando 'HELP %*' angezeigt werden.

Sprung zum Kommandointerpreter CCP

Nach Abschluß aller Initialisierungen wird abhängig davon, ob es sich um einen Kaltstart oder Warmstart handelt, einer der Kommandonamen '$CCPK ' oder '$CCPW ' gesucht und dorthin gesprungen. Damit ist der Systemanlauf beendet. Falls einer dieser Namen nicht gefunden wird, erfolgt eine Fehlermeldung und Systemabbruch.

Kommando HELP

Syntax: HELP [*name*]

Funktion:

Anzeige aller oder einer bestimmten Gruppe von Kommandonamen. Bei Weglassen von *name* werden alle Kommandos angezeigt. Es können auch, wie vom Betriebssystem CP/M bekannt, die Zeichen '?' und '*' benutzt werden, um eine bestimmte Gruppe von Namen auszuwählen.

Bildschirmanzeige: (gekürztes Beispiel)

```
77 E400 E204 BASIC
77 E40C CO8C WBASIC
```

Angezeigt werden:
- Bank, wo der Kommandoname gefunden wurde
- Adresse, wo der Kommandoname steht
- Anfangsadresse des eigentlichen Programmes
- Kommandoname

Die Anzeige kann mit der Taste PAUSE angehalten, mit CONT fortgesetzt und mit STOP abgebrochen werden.
Die Suche beginnt auf der Adresse E700 in der Bank FF und wird in Richtung niedrigerer Adressen und Banknummern fortgesetzt. Die genaue Reihenfolge ist aus der Bildschirmanzeige deutlich zu erkennen.

Bei Eingabe eines Kommandonamens zum Zweck des Programmstarts wird vom CCP die angezeigte Bank eingeschaltet und zur Programmanfangsadresse gesprungen.

Aufbau der Kommandonamen

An dieser Stelle soll noch einmal erklärt werden, was unter Kommandonamen zu verstehen ist und wie sie im Speicher aussehen müssen.
Dieser Sachverhalt ist aber bereits vom Z 9001 bekannt und unverändert übernommen worden. Die Namenssuche bildet jetzt die Grundlage zur Funktion des gesamten Betriebssystems.

Alle Programme werden anhand ihres Namens aufgerufen. Daran können sich wahlweise Parameter anschließen. Die Parameterauswertung ist Sache des gestarteten Programmes unter Mithilfe das CCP und wird bei der Softwarekomponente CCP beschrieben.

Im Speicher muß ab einer Adresse, die ein Vielfaches von 100H darstellt, ein Sprungbefehl zum Programmanfang stehen (3 Bytes, beginnend mit C3H). Die nächsten 8 Bytes sind für die ASCII-Zeichen des Namens reserviert. Kürzere Namen sind mit Leerzeichen (20H) aufzufüllen. Danach folgt ein Byte 00H. Das ergibt 12 Bytes, das erste ist immer C3H und das letzte 00H. Anschließend kann der nächste Kommandoname mit weiteren 12 Bytes folgen usw. Das Ende ist gekennzeichnet durch einen anderen Code anstatt C3H.
Auf dem nächsten Speicherplatz kann das entsprechende Programm beginnen.

Beispiel:

```
      ORG    7300H              ;Adresse muß mit ..00H enden (.. ist beliebig )
      JMP    PROG1              ;Sprungbefehl 3 Byte
      DB     'NAME1 '           ;Kommandoname 8 Byte
      DB     0                  ;Endekennzeichen immer 0
;
      JMP    PROG2              ;nächstes Kommando
      DB     'NAME2 '
      DB     0
;
      DB     0                  ;Ende der Namenstabelle (kann entfallen, wenn
                                ;Programm nicht mit C3H beginnt)
PROG1: ...
       ...
      RET
PROG2: ...
       ...
      RET
```

04.12.90 Lutz Elßner, Postfach 127-14, 0-8210 Freital

Softwarekomponente 2 UP - zentrale Unterprogramme

PROGRAMMBESCHREIBUNG

Das Programmsystem UP ist der zentrale Kernpunkt des Betriebssystems. Es ermöglicht den völlig problemlosen Zugriff auf Speicherplatze in anderen Speicherbänken und den Aufruf von Unterprogrammen in anderen Speicherbänken.
Maximal 256 Unterprogramme sind erreichbar, die teilweise selbst Bestandteil des Programmsystems UP sind oder von externen Programmen in beliebigen Banken zur Verfügung gestellt werden können.

Der Aufruf erfolgt mit dem Assemblerbefehl RST 20H und einem nachfolgenden Byte, das die Unterprogrammnummer angibt.

```
RST 20H
DB nummer
```

Die Nummer ist eine Zahl im Bereich 0 bis 255.
Bei diesem Verfahren werden sämtliche CPU-Registerinhalte beibehalten und können Parameter an das Unterprogramm übergeben. Das gleiche gilt auch für die Rückgabe der Registerinhalte an das rufende Programm. Außerdem gilt die Regel, daß grundsätzlich nur die CPU-Register vom Unterprogramm verändert werden dürfen, die Parameter zurückgeben (außer AF). Dadurch lassen sich die Anwenderprogramme im wesentlichen ohne PUSH und POP Befehle schreiben.

Der Aufruf darf von allen Stellen und Systemzuständen erfolgen, d.h. ein UP-Ruf kann aus dem Anwenderprogramm oder einem Unterprogramm genauso erfolgen, wie aus einem Gerätetreiber, der Bestandteil des BIOS ist, oder einer Interruptserviceroutine. Dabei ist zu beachten, daß beim Aufruf der Befehl EI abgearbeitet wird. Auch Programme, die selbst als Erweiterung dem UP-System zugeordnet werden, dürfen ihrerseits UP-Rufe benutzen.
Möchte der Anwender ein eigenes Unterprogramm einbinden oder ein vorhandenes ersetzen, erfolgt das ebenfalls durch einen RST-Ruf mit der Rufnummer 255. Die Anfangsadresse und Bank des Programmes sind dabei in den Registern zu übergeben.

Sollte einmal beim Aufruf eine Programmnummer angegeben werden, für die noch kein Unterprogramm existiert, dann wird im Register A der Fehlercode 7 und im Register B die Rufnummer zurückgegeben, das Carry-Flag ist gesetzt. Diese Form der Fehlermeldung gilt auch generell für alle Unterprogramme.

```
Carry-Flag  = 1
Register A  = Fehlercode
Register B  = Zusatzinformation zum Fehler (nur in Ausnahmefällen
              verwenden)
```

Daraus folgt, daß bei erfolgreicher Unterprogrammbeendigung das Carry-Flag grundsätzlich zu löschen ist. Im rufenden Programm sollte nach der Befehlsfolge RST und DB bei gesetztem Carry-Flag grundsätzlich ein Sprung zu einer Fehlerbehandlung oder eine Rückkehr in das Betriebssystem (zur Systemfehlermeldung) erfolgen. Bei Bedarf kann die Systemfehleranzeige auch selbst über RST UP angesprochen werden.
Wie zu erkennen ist, wird das Programmsystem UP zur Kommunikation zwischen verschiedenen unabhängigen Softwarekomponenten benutzt, die in beliebigen Speicherbanken und auf verschiedenen Adressen stehen. Die einzelnen Komponenten, z.B. Treiber für Kassetteninterface oder Kommandointerpreter, werden nicht mit dem gesamten Betriebssystem zu einem Ganzen gelinkt, sondern bleiben einzeln für sich stehen und sind dadurch viel übersichtlicher zu handhaben. Teile können weggelassen oder ergänzt werden. Treiberprogramme z.B. für Bildschirm oder Tastatur können ausgetauscht werden, wodurch sich das Betriebssystem problemlos an verschiedene Rechner und Hardwarekonfigurationen anpassen läßt. Dabei ist es, wie gesagt, unwichtig, in welcher Bank die neuen Programme stehen und auf welchen Adressen.
In der Initialisierungsphase des Systems (siehe Beschreibung Systemanlauf) werden alle Softwarekomponenten, die sich aktuell im Speicher befinden, an ihren Namen erkannt und können sich bei Bedarf in die Liste der Unterprogramme eintragen.

Liste der Unterprogramme

Nr.	Name	gehört zu	Funktion
			Ausgaben zum Bildschirm (CON-Gerät)
00	PRNA	UP	Ausgabe Register A zum Bildschirm
01	PRNZ	UP	Ausgabe Register A als ASCII-Zeichen zum Bildschirm (00 -1F als '.')
02	PRNSP	UP	Ausgabe ein Leerzeichen
03			
04	PRNDE	UP	Ausgabe von 2 Zeichen. aus D und E
05	PCRLF	UP	Ausgabe Zeilenschaltung (0DH, 0AH)
06	PRNBY	UP	Ausgabe Register A hexadezimal (2 Zeichen)
07	PRNAD	UP	Ausgabe Register HL hexadezimal (4 Zeichen)
08	PRNST	UP	Ausgabe Zeichenkette ab (DE) bis zum Code 00
09	PRST	UP	Ausgabe Zeichenkette ab (HL), Zeichenzahl in B
0A	PRCST	UP	Ausgabe Zeichenkette ab (DE) bis zum Code 00 (für Fehlermeldungen)
0B	DISPO	UP	Anzeige eines Speicherbereiches hexadezimal und ASCII von (HL) bis (DE)
0C	PRNER	UP	Systemfehleranzeige, A=Fehlercode, B=Zusatzinformation
0D	PRDEZ	UP	Ausgabe Register HL Dezimal (5 Zeichen)
0E			
0F			
			Eingaben von Tastatur (CON-Gerät)
10	INBY	UP	Eingabe 1 Byte hexadezimal in E R:A=1 leer / A=2 falsch / A=3 OK
11	INAD	UP	Eingabe Adresse hexadezimal in DE R:A=1 leer / A=2 falsch / A=3 OK
12	INSTR	UP	Eingabe Zeichenkette ab (HL), B=Länge, C=Bank
13	PAUSE	UP	Tastatureingabe PAUSE, CONT, STOP auswerten
14	INZIF	UP	Eingabe 1 Byte mit Aufforderungstext
15	INZIA	UP	Eingabe Adresse mit Aufforderungstext
16	INPUT	UP	Eingabe Zeichenkette mit Aufforderungstext
17	INPDA	UP	Eingabe 1 Byte mit Aufforderungstext und Anzeige des vorgegebenen Wertes
18	STENT	UP	Tastatureingabe STOP, ENTER auswerten
			Auswertung von Parametern aus der Kommandozeile
19	PARH2	UP	1 Byte hexadezimal aus Kommandozeile in Register E holen
1A	PARB2	UP	2 Byte hex. aus Kommandozeile holen, 1.Byte in L, 2.Byte in H u. E
1B	PARDB	UP	1 Byte dezimal aus Kommandozeile in E holen
1C	PARDA	UP	1 Adresse dezimal aus Kommandozeile in DE holen
1D	PARH4	UP	1 Adresse hexadezimal aus Kommandozeile in DE holen
1E	PARA2	UP	2 Adressen hexadezimal aus Kommandozeile in HL und DE holen
1F	PARA3	UP	3 Adressen hexadezimal aus Kommandozeile in HL, DE und BC holen
			Konvertierung / Rechnung
20	ASHEX	UP	'0' ... 'F' in 30...3F umwandeln
21	NUMP	UP	Byte aus A in 2 ASCII-Zeichen in DEumwandeln
22			
23	DADBC	UP	BCD-Addition 8 Tetraden DEHL:=DEHL+BC
24			
25			
26			
27			
28	NPAR	CCP	2 Parameter aus Kommandozeile ob 80H als Dateibezeichnung in FCB ab (5C)
29	PARA	CCP	Parameter ab (DE) als Dateibezeichnung interpretieren und ab (HL) eintr.
2A	KMOVE	CCP	1. Parameter der Kommandozeile (DE) wegschieben
2B	HEXDU	UP	Umwandlung Zeichenkette (z.B.'F123H') in HEX -Zahl in DE
2C	DEZDU	UP	Umwandlung Zeichenkette (z.B.'0123') in Dezimalzahl in DE
2D			
2E			
2F			

*** UP - zentrale Unterprogramme ***

Nr.	Name	gehört zu	Funktion
			Bedienung der E/A-Kanäle CDN, ROR, PUN und LST
30	CONSI	UP	Eingabe von Console (Tastatur)
31	STCON	UP	Statusabfrage Console (Tastaturstatus)
32	CONO	UP	Ausgabe zu Console (Bildschirm)
33	CONSO	UP	Ausgabe zu Console und evtl. zum Drucker
34	READI	UP	Eingabe vom Reader
35	STRDR	UP	Statusabfrage Reader
36	OPRDR	UP	OPEN Reader
37	CLRDR	UP	CLOSE Reader
38	PUNO	UP	Ausgabe zu Punch
39	STPUN	UP	Statusabfrage Punch
3A	OPPUN	UP	OPEN Punch
			Nummer Name gehört zu Funktion
3B	CLPUN	UP	CLOSE Punch
3C	LISTO	UP	Ausgabe zu List
3D	STLST	UP	Statusabfrage List
3E	OPLST	UP	OFEN List
3F	CLLST	UP	CLOSE List
			Speicherinhalt in anderen Bänken lesen / schreiben
40	R1	UP	1 Byte von (HL) Bank (A) in A lesen
41	R3	UP	3 Byte von (HL) (A) in E, D, H lesen
42	W1	UP	1 Byte aus E auf (HL) in Bank (A) schreiben
43	W2	UP	2 Byte aus E, D auf (HL) (A) schreiben
44	LDIRU	UP	Blocktransport zwischen Bänken von (HL) (B) nach (DE) (C), Länge = IX
45	CPROM	BOOT	Kommandoname in allen Bänken suchen
46	LDDRU	UP	Blocktransport rückwärts (entspricht DDR), Register wie LDDR
47			
48			
49			
4A			
4B			
4C			
4D			
4E			
4F			
50 - 6F	zur Zeit unbelegt		

Nr.	Name	gehört zu	Funktion
70	CTTY	UP	CON - TTY - Gerät
71	CCRT	UP	CON - CRT - Gerät (Bildschirm und Tastatur)
72	CBAT	UP	CON - BAT - Gerät
73	CUC1	UP	CON – UC1
74	RTTY	UP	RDR – TTY
75	RPTR	UP	RDR - PTR
76	RUR1	UP	ROR – UR1
77	RUR2	UP	RDR – UR2
78	PTTY	UP	PUN – TTY
79	PPTP	UP	PUN – PTP
7A	PUP1	UP	PUN - UP1
7B	PUP2	UP	PUN - UP2
7C	LTTY	UP	LST – TTY
7D	LCRT	UP	LST – CRT
7E	LLPT	UP	LST – LPT - Gerät (Drucker)
7F	LUL1	UP	LST - UL1

physische Geräte im BIOS, die über das I/O-Byte ausgewählt werden

Original BIOS-Rufe

80	BOOT	UP	BIOS-Ruf 0 Kaltstart
81	WBOOT	UP	BIOS-Ruf 1 Warmstart
82	STCON	UP	BIOS-Ruf 2 Status Console Eingabe
83	CONSI	UP	BIOS-Ruf 3 Eingabe Console
84	CONSO	UP	BIOS-Ruf 4 Ausgabe Console
85	LISTO	UP	BIOS-Ruf 5 Ausgabe List
86	PUNO	UP	BIOS-Ruf 6 Ausgabe Punch
87	READI	UP	BIOS-Ruf 7 Eingabe Reader
88	HOME	DBIOS	BIOS-Ruf 8 Rücksetzen Diskettensystem
89	SELDS	DBIOS	BIOS-Ruf 9 Auswahl Laufwerk
8A	SETTR	DBIOS	BIOS-Ruf 10 Auswahl Spur
8B	SETSE	DBIOS	BIOS-Ruf 11 Auswahl Sektor
8C	SETDM	DBIOS	BIOS-Ruf 12 Festlegen DMA-Adresse, die Bank wird auf 0FFH gesetzt
8D	READ	DBIOS	BIOS-Ruf 13 Lesen eines 128-Byte-Sektors von Diskette auf die DMA-Adresse
8E	WRITE	DBIOS	BIOS-Ruf 14 Schreiben eines 128 Byte Sektors auf Diskette
8F	STLST	UP	BIOS-Ruf 15 Status List
90	SECT	DBIOS	BIOS-Ruf 16 Sektortranslate

Nummer Name gehört zu Funktion

91	STCON	UP	BIOS-Ruf 17 Status Console Ausgabe
92	STRDR	UP	BIOS-Ruf 18 Status Reader
93	STPUN	UP	BIOS-Ruf 19 Status Punch
94	SETDB	DBIOS	Festlegen der Bank zur DMA-Adresse
95 – FE			zur Zeit unbelegt
FF	EXTUP	UP	Einbindung eines weiteren Unterprogrammes in das System UP

31.12.90 Lutz Elßner, Postfach 127-14, 0-8210 Freital

Softwarekomponente 3 CCP – Kommandointerpreter

PROGRAMMBESCHREIBUNG

Zweckbestimmung

Der Console Command Processor (CCP) dient zur Kommunikation mit dem Bediener. Diese Softwarekomponente ist vom Betriebssystem CP/M gut bekannt und auch die Bedienerkommunikation am Z 9001 funktioniert entsprechend. Es gibt also keine großen Probleme, beide Funktionen in einem CCP zu vereinigen. Im vorliegenden Fall realisiert der CCP die Kommandoeingabe und Auswertung, sowie das Aufsuchen eines Namens im Speicher, auf Diskette (wenn angeschlossen) und auf der Magnetbandkassette. Die Kommandos DIR, ERA, TYPE, SAVE, REN, USER usw. sind nicht Bestandteil des CCP und werden gesondert im EPROM bereitgestellt.

Kommandoeingabe

Hinter dem sogenannten Prompt-Zeichen wird vom Bediener eine Tastatureingabe erwartet. Die eingegebenen Zeichen werden im Speicherbereich 0081H bis 00FFH abgelegt. Ein Kommando kann aus dem Kommandonamen und Parametern bestehen. Zwischen einzelnen Parametern oder Teilen davon stehen ein oder mehrere Trennzeichen. Das sind zumeist Leerzeichen, aber auch die folgenden 6 Zeichen =_.: <> gelten als Trennzeichen. Diese dürfen nicht Bestandteil von Kommandonamen sein. Die Trennzeichen Doppelpunkt und Punkt werden auch vom CCP zur Kennzeichnung von Laufwerk, Dateiname und Dateityp ausgewertet.

Parameterauswertung

Nach dem Drücken der Taste ENTER beginnt die Interpretation des Kommandos. Zuerst werden alle Kleinbuchstaben in Großbuchstaben umgewandelt und die tatsächliche Länge des Kommandos ermittelt.
CCP versucht, die ersten zwei Parameter als Dateibezeichnungen zu interpretieren. Dafür wird die Grundform

```
lw:name.typ
```

angenommen. Alle Teile können auch einzeln weggelassen werden. Die ermittelten Werte für Laufwerk, Dateiname und Typ werden im Standard-File-Control-Block (FCB) ob Adresse 005CH (2. Parameter ab 006CH) formatgerecht eingetragen und stehen für die weitere Verwendung zur Verfügung.

Ein Kommando (das erste Wort einer Kommandozeile) besteht oft nur aus einem Wort und enthält keine n Doppelpunkt oder Punkt. Diese Zeichenkette (z.B. BASIC) steht dann als Dateiname im FCB ab Adresse 005DH zur Verfügung.

Anschließend wird dieser Name im Speicher in allen Bänken gesucht. Wenn er dort nicht vorhanden ist, wird er auch noch auf Diskette(wenn angeschlossen) und Kassette gesucht. Auch dafür ist ein aufbereiteter Dateiname im FCB erforderlich. Wird der eingegebene Name nirgends gefunden, muß die Kassetteneingabe abgebrochen werden, um wieder in den Kommandomodus zurückzukehren. Dieser Abbruch kann im Gegensatz zum Z 9001 jederzeit durch die Taste GRAPHIC hervorgerufen werden (siehe Softwarekomponente TAPE).

Nach dem Auffinden im Speicher oder Laden von Diskette oder Kassette ist der erste Parameter der Eingabezeile abgearbeitet.

Parameterübergabe an das Programm

Jetzt wird die gesamte Eingabezeile so verschoben, daß der erste Parameter (Kommandoname) überschrieben wird und der nächste an erster Stelle steht. Anschließend werden wie bereits beschrieben die ersten 2 Parameter als Dateinamen interpretiert und in den FCB eingetragen. Wenn das alles passiert ist, wird endlich unter Beachtung von Bankumschaltungen und Rückkehrmöglichkeiten zum aufgerufenen Programm gesprungen. Falls es sich

um ein CP/M-Programm handelt, muß zuvor noch in den CP/M-Modus umgeschaltet werden.

Beispiel:

%TEST 123.abc B:5.DE fgh

Nachdem der Name TEST gefunden und weggeschoben wurde, wird im FCB abgelegt:

```
005C: 00                         _        kein Laufwerk: mit Doppelpunkt angegeben
005D: 31 32 33 20 20 20 20 20 123         Dateiname
                               -----
0065: 41 42 43                   ABC      Dateityp in Großbuchstaben

006C: 02                         _        Laufwerkscode für B:
006D: 35 20 20 20 20 20 20 20 5           Dateiname ist die Zahl 5
                               -------
0075: 44 45 20                   DE_      Dateityp
```

auf Adresse 0080H steht die Länge vom Rest der eingegebenen Zeile (hexadezimal)

```
0080: 13
```

ab Adresse 0081H folgen die Zeichen der eingegebenen Kommandozeile

```
0081: 20 31 32 33 2E 41 42 43 20 42 3A 35 2E 44 45 20 46 47 48
```

Parameterauswertung im Programm

Das gestartete Programm findet die ersten zwei gültigen Parameter ab 005DH und 006DH aufbereitet vor. Das sind oft Hexadezimalzahlen. Wenn sich auf diesen Speicherplätzen Leerzeichen befinden, sind keine Parameter eingegeben worden. Die Umwandlung dieser ASCII -Zeichenketten in echte 8 oder 16 Bit-Werte, die in CPU-Registern stehen, kann sehr einfach mit den Funktionen 19H bis 1FH der Softwarekomponente UP erfolgen.

Beispiel:

DEZ E70C eingegebenes Kommando zur Zahlenumwandlung

dazugehöriges Programm:

```
DEZ:    LD A,(005DH)
        CMP 20H             ;wenn Leerzeichen, dann kein Parameter eingegeben
        RZ                  ;und Rückkehr (oder Sprung zur Eingabeaufforderung)
;
        RST UP              ;Aufruf der Softwarekomponente UP
        DB PARH4            ;mit der Funktion PARH4
                            ;jetzt enthält das Registerpaar DE den Wert 0E70CH
        RC                  ;Rückkehr bei Fehlern z.B. Eingabefehler mit
                            ;Fehleranzeige im Betriebssystem
        EX DE,HL            ;übertragen des Wertes in HL
        RST UP              ;Aufruf der Softwarekomponente UP
        DB PRDEZ            ;HL als Dezimalzahl am Bildschirm anzeigen
        RET
```

Bei Verwendung dieser UP-Funktionen werden die abgeholten Parameter auch automatisch weggeschoben, d.h., die nächsten Parameter rücken wie oben schon beschrieben noch vorn.
Diese Verschiebefunktion und die Auswertung und Eintragung in den FCB können auch einzeln als UP-Funktion aufgerufen werden. CCP trägt diese Funktionen unter den Nummern 28H und 29 H in die Adreßtabelle des UP ein.

Falls keine geeignete UP-Funktion zur Auswertung eines bestimmten Parameters zur Verfügung steht, wird der Rest der Kommandozeile ab Adresse 0081H zur Weiterverarbeitung auch noch i m Originalzustand vorgefunden.

Rückkehr aus einem Programm

Programme bzw. Kommandos können mit RET beendet werden, um wieder in den CCP-Kommandomodus zurückzukehren. Bei gesetztem CY-Flag wird vom CCP ein Fehler mit der Fehlernummer aus Register A angezeigt. In diesem Fall dürfen vom Programm bestimmte Systemzustände (Stack usw.) nicht verändert werden. Alle zum Betriebssystem gehörenden und hier beschriebenen Kommandos enden mit RET.

Größere Programmkomplexe, wie Programmiersprachen, enden mit einem Sprung zur Adresse 0, um einen Warmstart auszulösen. Nach Abarbeitung des Warmstarts (siehe Softwarekomponente BO) wird wieder der CCP-Kommandomodus eingenommen.

Diese Rückkehrmöglichkeiten sind beim CP/M und Z 9001 identisch.

Kommando ^

Syntax:

^[*bank*]

Funktion:

Dieses Kommando ist für ganz schwierige Kompatibilitätsprobleme gedacht und im Normalfall nicht erforderlich. Es dient zum Verlassen des Betriebssystems für das 192 KByte RAM/EPROM-Modul und führt einen Warmstart des Original Z 9001 Betriebssystems durch Sprung auf die Adresse F003H aus. Zuvor wird im 192 KByte RAM/EPROM-Modul die angegebene Speicherbank zugeschaltet. Wird keine Bank angegeben, dann wird im Allgemeinen der BASIC - Interpreter eingeschaltet.

Mit diesem Kommando besteht die Möglichkeit, ein beliebiges 10-KByte-ROM-Modul softwaremäßig zu "stecken" und hundertprozentig kompatibel mit dem Original-Betriebssystem Z 9001 zu arbeiten. Die Speicherbereiche in den nicht aktiven Bänken sind praktisch nicht vorhanden und nicht nutzbar, d.h. sie können auch nicht stören.

Durch RESET oder durch Sprung zur Adresse E000 nach Zuschalten der Bank 00 kann wieder zum Bankbetrieb umgeschaltet werden, wobei der Inhalt des Anwenderspeichers erhalten bleibt.

TRICK: Falls sich der BASIC-Interpreter in der Speicherbank 77H befindet (mit HELP zu ermitteln), kann diese Umschaltung zum Bankbetrieb aus BASIC heraus mit der Befehlsfolge: OUT 255,112:CALL* E000 durchgeführt werden. Durch den OUT-Befehl wird ein Teil des BASIC-Interpreters weggeschaltet!

Hinweis:

Unter dem Original-Betriebssystem Z 9001 sind nur unveränderte Programme, die vorher schon auf 10-KByte-ROM-Modulen existierten, und nicht für den Bankbetrieb angepaßt wurden, betriebsfähig. Die zum Betriebssystem gehörenden Kommandos sind, auch wenn die richtige Bank eingeschaltet wurde, allein nicht funktionstüchtig und können zum Programmabsturz führen.

Softwarekomponente 4 KO - Monitorkommandos

PROGRAMMBESCHREIBUNG

Zweckbestimmung

Die Monitorkommandos dienen zum Anzeigen, Ändern, Vergleichen, Transportieren und Füllen von Speicherbereichen in beliebigen Bänken, sowie zur Prüfsummenberechnung und Ein- und Ausgabe über Portadressen.

Hinweis: Die hier gemachten Angaben zur Schreibweise, Parameterübergabe und zum Ablauf gelten auch für Kommandos, die zu bestimmten Softwarekomponenten gehören (z.B. Kassetteninterface oder Diskettentreiber). Prinzipiell kann jedes beliebige Programm auf diese Weise mit Parametern aufgerufen werden.

Alle Kommandos sind aus dem Kommandomodus des Betriebssystems als residente Kommandos jederzeit aufrufbar und benutzen selbst keinen Anwenderspeicher, sofern das nicht mit dem Kommando beabsichtigt ist.
Die Parameterübergabe erfolgt, wie das beim Betriebssystem CP/M (SCP) üblich ist. Der Kommandointerpreter CCP legt die Eingabezeile im Eingabepuffer (Adreßbereich 0080H - 00FFH) ab und versucht gleichzeitig die ersten zwei Parameter als Dateinamen zu interpretieren. Diese werden im Standard FCB (File-Control-Block) ab Adresse 005CH bzw. 006CH abgelegt. Von dort werden sie mit Hilfe von Funktionen der Softwarekomponente UP ausgewertet. Für die Programme der eigentlichen Monitorkommandos bleibt somit nicht mehr viel zu tun und es ist für den Anwender kein Problem, weitere Kommandos hinzuzufügen.

Erläuterungen zur Schreibweise

Alle Kommandos beginnen mit ihrem Namen, der auch nur aus einem Buchstabe bestehen kann. Danach folgen durch Trennzeichen getrennt wahlweise Parameter, die standardmäßig Hexadezimalzahlen darstellen. Führende Nullen oder der Suffix H am Ende können geschrieben oder weggelassen werden. Wenn als Parameter eine Zeichenkette oder Dezimalzahl eingegeben werden soll, so ist das bei der Kommandobeschreibung extra angegeben.

Als Trennzeichen sind ein oder mehrere Leerzeichen zu verwenden, wenn nicht anders gefordert.
Parameter in eckigen Klammern können weggelassen werden.
Wenn Parameter, die die Bank zu einer Speicheradresse bezeichnen, weggelassen werden und die Speicheradresse befindet sich im Bereich 0000H - E7FFH, fordert das Programm die Eingabe an.

Ablauf der Kommandos und Beenden

Die Kommandos können noch einmaliger Ausführung beendet sein oder z.B. mit der nächsten Adresse fortgesetzt werden. Fortlaufende Bildschirmausgaben können mit der Taste PAUSE angehalten und mit CONT fortgesetzt werden. Ein Abbruch aus jedem beliebigen Zustand kann mit der Taste STOP erfolgen.
Aufgetretene Fehler werden nach Rückkehr vom Betriebssystem angezeigt. Besonders häufig treten die folgenden vom Betriebssystem Z 9001 bekannten Fehler auf:

1 zu wenige Parameter eingegeben

2 Parameter falsch geschrieben, Anfangsadresse größer als Endadresse u.a.

3 Parameter außerhalb zulässiger Grenzen (z.B. Dezimalzahl > 255)

Anzeigen und Ändern von Speicherzellen

S *aadr* [*bank*]

angezeigt wird:

bank adresse hex ascii:

Das angezeigte Byte kann unverändert gelassen (ENTER) oder auch Eingabe einer Hexadezimalzahl geändert werden. Bei Fehleingaben wird dieselbe Zeile erneut angezeigt.

Anzeigen von Speicherbereichen

D *aadr eadr* [*bank*]

Pro Zeile werden 8 Bytes hexadezimal und ASCII angezeigt.

Transportieren von Speicherbereichen

T *aadr eadr zadr* [*qbank*] [*zbank*]

Vergleichen von Speicherbereichen

V *aadr eadr zadr* [*qbank*] [*zbank*]

Füllen von Speicherbereichen

F *aadr eadr byte* [*bank*]

Berechnen der Prüfsumme eines Speicherbereiches

CS *aadr eadr* [*bank*]

Angezeigt wird die 16 Bit Summe, die durch Addition aller Bytes entstanden ist und das 16 Bit Ergebnis der CRC-Berechnung.

Sprung zu einer Speicheradresse

J *adr* [*bank*]

Dieses Kommando kann benutzt werden, um in Spezialfällen ein anderes Kommando, das über seinen Namen nicht erreichbar ist, aufzurufen. Die Parameter für das Kommando können einfach hinter der Angabe der Bank angefügt werden.

Eingabe von einer Portadresse

IN *port* [*byte*]

Anzeige des von der Adresse *port* eingelesenen Bytes. Wenn *byte* angegeben, wird es vor der Eingabe auf *port* ausgegeben. Diese Zusatzfunktion dient hauptsächlich zum Lesen der Register des SIO U 856.

Ausgabe auf eine Portadresse

OUT *port byte* [*byte ...*]

Ausgabe des *byte* auf die Adresse *port*. Es können beliebig viele Bytes angegeben werden, die dann nacheinander auf *port* ausgegeben werden. Die

ersten zwei Bytes werden bei verbotenem Interrupt (DI) ausgegeben. Das ist zum Schreiben auf interruptbetriebene Peripherieschaltkreise erforderlich.

Ausgabe eines Speicherbereiches auf den logischen Kanal PUNCH

PM *aadr eadr* [*sadr*] [*bank*]

Ausgabe des Speicherinhaltes von *aadr* bis einschließlich *eadr* aus der angegebenen Bank auf den logischen Kanal PUNCH. Die Angabe *sadr* ist erforderlich, falls ein Maschinenprogramm mit Startadresse ausgegeben werden soll und das dem PUNCH-Kanal zugewiesene Treiberprogramm diese Angabe auswertet. Soll das Maschinenprogramm nach dem Laden nicht gestartet werden, ist *sadr* wegzulassen oder FFFF einzugeben. Die Funktionen OPEN und CLOSE werden automatisch ausgeführt.

Eingabe vom logischen Kanal READER in den Speicher

RM *aadr eadr* [*bank*]

Eingabe vom logischen Kanal READER in den Speicher von *aadr* bis *eadr* in die angegebene Speicherbank. Die Eingabe ist beendet, wenn die Endadresse erreicht ist. Ein vorzeitiger Abbruch ist möglich, wenn das vom dem READER-Kanal zugewiesenen Treiberprogramm unterstützt wird. Die Funktionen OPEN und CLDSE werden automatisch ausgeführt.

Eingabe vom logischen Kanal READER und Ausgabe auf den logischen Kanal PUNCH

PR

Dieses Kommando dient zum "Durchreichen" von Bytes von einem peripheren Gerät zu einem anderen. Aufgrund der großen Anzahl von verfügbaren Gerätetreiberprogrammen kann hiermit sehr einfach eine Schnittstellenkonvertierung (seriell - parallel usw.) realisiert werden. Die übertragenen Zeichen werden gezählt und am Ende als 8 stellige Dezimalzahl angezeigt. Die Funktionen OPEN und CLOSE werden automatisch ausgeführt.

Die Anzahl der verfügbaren Kommandos wird bei Bedarf erweitert.

Softwarekomponente 5 TAPE - Kassetteninterface

PROGRAMMBESCHREIBUNG

Zweckbestimmung

Das Treiberprogramm TAPE ersetzt die Treibersoftware für das Kassetteninterface aus dem Betriebssystem Z 9001. Das ist eine Voraussetzung, um das Betriebssystem des 192 KByte RAM/EPROM-Moduls selbständig auch außerhalb dieses Rechners zu betreiben. Vom Rechner werden lediglich die Interruptserviceroutinen und die Hardware (PIO, CTC, Impulsformerstufen und Diodenbuchse) weiterhin benutzt.

Die neue Treibersoftware realisiert vollständig das alte Aufzeichnungsverfahren, wurde aber an die neuen Bedingungen wie z.B. Bankbetrieb angepaßt. Auch der verwendete Arbeitsspeicher wurde aus dem Adreßbereich 005CH - 00FFH in eine RAM-Speicherbank verlegt, um nicht mit dem Betriebssystem CP/M in Konflikt zu kommen.

Zur wesentlichen Verbesserung des Bedienkomforts ist eine Abbruchsmöglichkeit bei Kassetteneingabe realisiert worden, die zu jedem Zeitpunkt (auch wenn gar kein Kassettengerät angeschlossen ist) funktioniert. Während vom Programm auf Impulse von der Kassette gewartet wird, leuchtet die grüne GRAPHIC-Leuchtdiode auf der Tastatur. Damit wird dieser scheinbare Verklemmungszustand sichtbar gemacht. Der Bediener kann durch Drücken der Taste GRAPHIC einen Abbruch erzwingen. Diese Anzeige- und Abbruchsmöglichkeit wird auch bei anderen Treibern z.B. Druckertreiber angewandt.

Schnittstelle

Mehrere Unterprogramme der Softwarekomponente TAPE können über die Softwarekomponente BD mit Rufnummern ab 80H von anderen Programmen aufgerufen werden. (Siehe Beschreibung BD) Die nutzbaren Unterprogramme sind:

OPENW	OPEN WRITE (Schreiben Block Nr. 0 mit Dateiname/typ)
WRITE	Schreiben Datenblock
CLOSW	CLOSE WRITE (Schreiben Endeblock Nr. FF)
OPENR	OPEN READ (Lesen Block 0 und vergleichen Dateiname/typ)
READ	Lesen Datenblock
CLOSR	CLOSE READ (Lesen Endeblock FF)
RRAND	Block mit vorgegebener Blocknummer lesen
KARAM	Block mit vorgegebener Blocknummer schreiben
MAREK	Lesen des ersten gefundenen Blockes (Blocknummer ermitteln)
LOAD1	wird vom CCP zum Laden von Programmen gerufen LOAD1 darf im CP/M-Modus nicht benutzt werden

Außerdem existieren folgende residente Kommandos zur direkten Arbeit mit dem Kassetteninterface:

Kommando CLOAD

Syntax:

CLOAD *name.typ aadr eadr bank ladr*

Funktion:

Laden einer Datei mit dem angegebenen *name* und *typ* von Kassette in den Speicher. *aadr* ist die Speicheradresse für das erste Byte der Datei. Auf *eadr* wird das letzte Byte abgespeichert, wenn die Datei länger ist, werden nachfolgende Daten ignoriert. *bank* gibt die Speicherbank an, wo die Daten abgelegt werden. Mit dem Parameter *ladr* können auch am Anfang der Datei Daten übersprungen werden. Der Adreßbereich von *aadr* bis *ladr*-1 wird nicht beschrieben, aber die entsprechende Anzahl gelesener Datenbytes wird übersprungen.

Mit diesem leistungsfähigen Kommando kann eine beliebig kleine Anzahl von Bytes (auch 1 Byte) "aus der Mitte" einer Datei von Kassette in jeden Speicherbereich und in jede Speicherbank eingelesen werden.
Diesen komplizierten Fall wird man selten brauchen. Durch Weglassen der Parameter von hinten wird das Kommando sehr einfach und verständlich.

CLOAD *name*

entspricht genau dem CLOAD-Kommando des Z 9001 Betriebssystems. Die Datei wird auf die im File-Control-Block (das ist der erste Block auf der Kassette) angegebene Adresse geladen. Wenn der Typ nicht angegeben wird, setzt das Programm den Typ COM ein.
Manchmal ist diese Adresse gerade nicht frei oder nicht bekannt. Bei Dateien, die keine Maschinenprogramme sind, ist die Angabe im FCB evtl. undefiniert. Um jede beliebige Datei laden zu können, muß man dem Rechner die Speicheradresse mitteilen, wohin die Datei geladen werden soll:

CLOAD *name*[.*typ*] *aadr*

Damit wird die Datei ab *aadr* bis zum Dateiende eingelesen. Die Endadresse und weitere Angaben kann man sich anschließend mit dem residenten Kommando FCB anzeigen lassen.
Soll nur ein Teil der Datei eingelesen oder der Speicherbereich begrenzt werden, kann auch noch die Endadresse mit angegeben werden:

CLOAD *name*[.*typ*] *aadr eadr*

Dateien können in jeden beliebigen RAM-Speicherbereich eingelesen werden. Im Adreßbereich 0000H bis E7FFH stellt das 192 KByte RAM/EPROM-Modul mehrere Speicherbänke zur Verfügung. Die letzte Speicherbank mit der Nummer 15 ist für Anwenderprogramme vorgesehen. Dorthin werden normalerweise auch die Dateien von der Kassette geladen. Um auch in die anderen Bänke zu laden, ist der nächste Parameterbank anzugeben:

CLOAD *name*[.*typ*] *aadr eadr bank*

Dabei muß unbedingt beachtet werden, daß die Bänke mit dem Systemspeicher nicht überschrieben werden.

Jetzt folgt noch ein Beispiel:

Die Datei DATEN.MAX hat die Länge 1 KByte, das entspricht 400H. Diese Datei soll auf die Adresse C800H in die Bank 13 (0DH) geladen werden:

CLOAD DATEN.MAX C800 FFFF DD

Die Endadresse wäre CBFF. Diese muß man aber nicht erst ausrechnen, weil das Laden am Dateiende automatisch beendet wird. Wichtig ist nur, daß der Parameter nicht kleiner als die wirkliche Endadresse ist. Beim Parameter Bank gelten die höherwertigen 4 Bits (eine Tetrade) für den Adreßbereich C000H bis DFFFH, die niederwertigen für den Adreßbereich E000H bis E7FFH. Das muß man sich aber nicht merken, wenn die gewünschte Banknummer in beide Tetraden eingetragen wird.

Jetzt sollen von derselben Datei nur die Speicheradressen C923H bis C925H beschrieben werden. Auf diese Adressen sollen dieselben Bytes geschrieben werden, wie im vorhergehenden Beispiel. Beim Lesen sind also vorher und nachher Daten zu überspringen:

CLOAD DATEN.MAX C800 C925 DD C923

Kommando CSAVE

Syntax:

CSAVE *name.typ aadr eadr bank sadr aadf eadf*

Funktion:

Schreiben eines Speicherbereiches als Datei mit dem angegebenen *name* und *typ* auf Kassette. Die Parameter *aadr*, *eadr* und *bank* kennzeichnen den Speicherbereich, aus dem die Daten gelesen werden sollen. Die Parameter *sadr*, *aadf* und *eadf* werden in den FCB der Datei eingetragen und mit auf der Kassette abgespeichert.
Sie sind beim Wiedereinlesen der Datei von Bedeutung. Das einfachste Kommando sieht so aus:

CSAVE *name aadr eadr*

Als Typ wird COM eingetragen. Der Speicherbereich von *aadr* bis einschließlich *eadr* aus der Bank 15 wird auf Kassette abgespeichert. Die angegebenen Adressen werden auch als Anfangs- und Endadresse in den FCB übernommen. Als Startadresse wird FFFFH eingetragen.

Die Angaben Anfangs-, End- und Startadresse werden benötigt, wenn die Datei im CCP-Kommandomodus als Programm geladen wird. Nach Angabe des Datei- (bzw. Programm-) namens und ENTER wird die Datei ab der Anfangsadresse bis zum Dateiende in den Speicher geladen und auf der Startadresse gestartet. Der Wert FFFFH als Startadresse ist die Ausnahme, weil die Datei damit nicht gestartet wird. Diese Vorgänge sind jedem Nutzer bereits bestens vertraut, denn genauso wurden schon immer alle Maschinenprogramme geladen und gestartet. Die abgespeicherten Dateien sind selbstverständlich auch an allen Computern einlesbar, die nicht mit dem 192 KByte RAM/EPROM-Modul und dem hier beschriebenen Kassettentreiberprogramm ausgerüstet sind.

Um Programme auch beim Laden auf andere Speicherbereiche zu bekommen, als beim Abspeichern, kann man die Parameter *aadf* und *eadf* angeben.

Ein Vergleichslesen (VERIFY) nach dem Abspeichern erfolgt nicht automatisch, dazu dient das folgende Kommando.

Kommando VERIFY

Syntax:

VERIFY

Funktion:

Nach dem Start des Kommandos wird um Bildschirm ständig die gerade auf der Kassette gelesene Blocknummer angezeigt. Im Wartezustand leuchtet die grüne GRAPHIC-Leuchtdiode auf der Tastatur. Ein Beenden des Kommandos ist nur durch Abbruch mit der Taste GRAPHIC möglich.

Die Kassette kann beliebig hin und her gespult werden. Die Anzeige der Blocknummer erleichtert das Auffinden von Dateianfängen wesentlich. Bei fehlerhaft gelesenen Blocken wird anstatt der Blocknummer ** angezeigt.

Beim Auffinden eines Dateianfanges werden der Dateiname und alle Angaben zur Datei angezeigt. Dabei wird zwischen standardgerechten Dateien und Dateien, die der BASIC -Interpreter auf die Kassette geschrieben hat, unterschieden.

Die BASIC-Dateien beginnen nicht mit der Blocknummer 0 (sondern mit 1) und enden nicht mit der Nummer FF. Stattdessen steht der Dateiname am Anfang von Block 1. Dort sind auch der Typ und die Länge verschlüsselt. Diese Angaben werden entschlüsselt und angezeigt.

Bei standardgerechten Dateien werden Name, Typ, Anfangs-, End-, und Startadresse sowie der Dateischutz angezeigt.

Beim Kontrollesen gesamter Dateien werden defekte Blöcke registriert und zum Zurückspulen aufgefordert. Am Ende werden vollständig fehlerfrei gelesene Dateien mit der Ausschrift "FILE COMPLETE" gemeldet.

Kommando FCB

Syntax:

```
FCB
```

Funktion:

Anzeige des im Speicher stehenden Kassetten-File-Control-Blockes in folgender Form:

```
NAME:xxxxxxxx
TYP :xxx
AADR:yyyy
EADR:yyyy
SADR:yyyy
SBY :yy
```

x=Buchstabe, y=Hexadezimalziffer, SBY bedeutet Schutzbyte

Kommando READS

Syntax:

READS *nummer*

Funktion:

Lesen eines 128 Byte langen Blockes von Kassette und Anzeigen auf dem Bildschirm. Als Parameter ist die zu lesende Blocknummer (hexadezimal) einzugeben. Die Anzeige erfolgt hexadezimal und als ASCII -Zeichen.
Der gelesene Block steht anschließend im Adreßbereich E580-E5FF in der Bank 14.

Softwarekomponente 6 FDT - Floppy-Disk-Treiber

PROGRAMMBESCHREIBUNG

Das Treiberprogramm Floppy-Disk, dient zur direkten Ansteuerung des Floppy-Disk-Controller-Schaltkreises U 8272. Zur Ansteuerung von maximal 4 Disketten-tenlaufwerken beliebiger Typen (8 Zoll, 5.25 Zoll und 3.5 Zoll) wurde ein spezielles Floppy-Disk-Modul entwickelt, aber leider nicht produziert. Als Ersatz kann ein FD-Modul zur Ansteuerung von max. 2 Laufwerken des Typs 1.6 (80 Spuren, 2 Seiten) angeboten werden. Damit ist hardwarebedingt nur die Betriebsart MFM bei 4 MHz möglich. Die Software FDT ist aber trotzdem für das komfortablere Modul ausgelegt. Diese Einschränkungen sind bei der folgenden Beschreibung zu beachten.

Das Treiberprogramm und der Arbeitsspeicher (Diskettenpuffer usw.) befinden sich in einer Speicherbank des 192 KByte RAM/EPROM-Moduls. Damit wird eine wichtige Voraussetzung für ein CP/M-kompatibles Betriebssystem erfüllt.

Wichtigster Bestandteil des Floppy-Disk-Treibers ist das Steuerprogramm zur zeitoptimalen Ansteuerung des Schaltkreises U 8272. Mit diesem einen Programm können alle 15 FDC-Befehle ausgeführt werden. Abhängig vom Laufwerkstyp und vom Aufzeichnungsverfahren gelten verschiedene Zeitbedingungen für das Lesen oder Schreiben. Im Extremfall muß aller 13 µs ein Datenbyte übertragen werden. Um diese Zeit einzuhalten, muß einerseits der Rechner mit einer hohen Taktfrequenz arbeiten und andererseits die Software mit möglichst wenig Taktzyklen auskommen. Es ist gelungen für einen Schreib - oder Lesezyklus mit 47 Takten auszukommen, das dürfte das absolute Minimum darstellen! Damit sind bereits bei 2 MHz Rechnertaktfrequenz 5.25 Zoll Disketten bis 800 KByte (MFM) und 8-Zoll Disketten bis 308 KByte (FM) ansteuerbar. Für die Ansteuerung von 8-Zoll-Disketten im MFM-Modus oder 5.25 Zoll Disketten bis 1.2 MByte ist eine Rechnertaktfrequenz von mindestens 4 MHz erforderlich. Diese Formate sind aber bei CP/M nicht üblich und ein Verzicht wegen zu geringer Taktfrequenz stellt keine große Einschränkung dar.

Weitere Bestandteile des Floppy-Disk-Treibers dienen zur Vorbereitung und Ausführung einzelner Befehle wie Kopfpositionierung, lesen und schreiben. Es sind alle Programme enthalten, die vom DISK-BIOS für das Betriebssystem CP/M benötigt werden.

Andere Programme z.B. das DISK-BIOS und die residenten Kommandos zur Floppy-Steuerung (FDK) können Unterprogramme des Floppy-Disk-Treibers über eine Sprungtabelle erreichen. Das ist die einzige Stelle im gesamten Betriebssystem, wo Unterprogramme aus Echtzeitgründen nicht über die zentralen Schnittstellen UP oder BDOS erreicht werden.

Aufbau der Sprungtabelle

Die Softwarekomponente FDT beginnt auf der Adresse 0000H in einer beliebigen Bank. Diese Bank kann bei Bedarf mit dem Kommando HELP ermittelt werden.
Auf den folgenden Adressen stehen Sprungbefehle zu den genannten Unterprogrammen.

C119	U8272	beliebiges FDC-Kommando
C11C	U82ND	FDC-Kommando ohne Datenübertragung
C11F	U82RW	READ/WRITE DATA
C122	PRD	Diskettenpuffer lesen
C125	PWR	Diskettenpuffer schreiben
C128	RECAL	RECALIBRATE (mit neuem Steuerblock)
C12B	SEEK	SEEK (mit neuem Steuerblock)
C12E	IDZY	ID-Feld lesen
C131	ANZ	Anzeigen der Befehls- und Ergebnisphase
C134	ERROR	Fehleranzeige FDC
C137	RECAU	RECALIBRATE-Unterprogramm

```
C13A  SEEKU  SEEK-Unterprogramm
C13D  MOREC  Motor ein, RECALIBRATE, Motor aus
C140  MORDY  Motor ein, warten bis FDC READY
C143  READY  warten bis FDC READY
C146  STAT   Statusabfrage FDC, Rückkehrparameter A=ST3
C149  FDRES  Kommando FDR
C14C  FDM    Kommando FDM
```

Die residenten Kommandos zur Ansteuerung des U 8272 sind kein Bestandteil des Treiberprogrammes. Sie sind in der Softwarekomponente FDK enthalten und müssen gesondert, aber in der gleichen Speicherbank im 192 KByte RAM/EPROM-Modul bereitgestellt werden.
In dieser Komponente ist nur das Kommando für die Kaltstartinitialisierunq enthalten. Bei Bedarf kann diese Initialisierung auch von Hand mit folgendem Kommando ausgeführt werden.

Kommando FDI

Syntax:

```
FDI
```

Funktion:

Initialisierung des Floppy-Disk-Moduls. Es werden alle Motoren ausgeschaltet, die Präkompensation ausgeschaltet, die FDC-Taktfrequenz 4 MHz eingestellt und der Befehl FDC-SPEZIFY zur Einstellung der Schrittzeit, Kopfentladezeit, Kopfladezeit und zur Einstellung des Nicht-DMA-Modus abgearbeitet.

31.12.90 Lutz Elßner, Postfach 127-14, 0-8210 Freital 23

Softwarekomponente 7 FDK - Floppy-Disk-Kommandos

PROGRAMMBESCHREIBUNG

Die Softwarekomponente FDK mit den anschließend beschriebenen residenten Kommandos zur Ansteuerung des U 8272 muß in der gleichen Speicherbank wie die Softwarekomponente FDT (Floppy-Disk-Treiber) im 192 KByte RAM/EPROM-Modul bereitgestellt werden.

Kommando FDC

Syntax:

FDC *p1 p2 p3 p4 p5 p6 p7 p8 p9*

Funktion:

Ausführung eines der 15 Befehle des FDC. Die 9 Parameter entsprechen den 9 Bytes, die der FDC in der Befehlsphase maximal liest. Abhängig vom Befehl können auch weniger Parameter ausreichen. Jeder Befehl besteht aus den 3 Phasen Befehlsphase, Ausführungsphase und Ergebnisphase. In der Befehlsphase werden die vom FDC tatsächlich gelesenen Bytes, in der Ausführungsphase die tatsächlich übertragenen Datenbytes und in der Ergebnisphase die tatsächlich zurückgegebenen Bytes gezählt.

Anhand dieser Zahlen werden nach dem Befehl die Anfangs- und Endadresse des Datenbereiches, die gelesenen Bytes der Befehlsphase und die zurückgegebenen Bytes der Ergebnisphase in folgender Form angezeigt:

```
xxxx xxxx                    (Datenbereich)
xx xx xx xx xx xx xx xxxx    (Befehlsphase)
xx xx xx xx xx xx xx         (Ergebnisphase)
```

Beispiel:

```
%FDC 4 0          Befehl SENSE DRIVE STATUS und Laufwerksnummer
9000 9000         keine Datenübertragung
04 00             2 Bytes in Befehlsphase gelesen
30                1 Byte in Ergebnisphase zurückgegeben
```

Der Befehlsbeschreibung kann entnommen werden, daß das Statusregister 3 zurückgegeben wurde. Der Wert 30 bedeutet, daß das Laufwerk bereit ist und der Kopf sich in Spur 0 befindet.

Kommando FDR

Syntax:

FDR

Funktion:

Auslösung eines Hardware-Reset für den FDC U 8272. Dieses Kommando ist bei Verklemmungen des FDC einzugeben.

Kommando FDM

Syntax:

FDM [*nummer*]

Funktion:

Ein- und Ausschalten des Motors eines Diskettenlaufwerkes. Zum Einschalten ist als *nummer* die physische Laufwerksnummer 0-3 einzugeben. Die Motoren

der anderen drei Laufwerke werden dabei ausgeschaltet. Bei Weglassen der *nummer* werden alle Motoren ausgeschaltet.

Kommando RECALIBRATE

Syntax:

FRE *nummer*

Funktion:

Im angegebenen Laufwerk wird der Kopf auf Spur 0 positioniert.

Kommando SEEK

Syntax:

FSE *nummer spur*

Funktion:

Im angegebenen Laufwerk wird der Kopf auf die angegebene Spur positioniert.

Kommando READ ID

Syntax:

FID *befehl nummer anzahl*

Funktion:

Lesen mehrerer aufeinanderfolgender Identifikationsfelder von der aktuellen Spur der Diskette. Als *befehl* ist 0A für FM und 4A für MFM anzugeben. Die *nummer* entspricht wieder der Laufwerksnummer 0-3, wobei zusätzlich Bit 3 die Seite angibt. Als *anzahl* wird angegeben, wieviele ID-Felder gelesen werden sollen. Mit diesem Kommando können die Sektorgröße, der physische Sektorversatz und weitere Angaben zum Diskettenformat ermittelt werden.

Kommando FORMAT

Syntax:

FORMAT *befehl nummer N SC GPL D tab H0vH1b*

Funktion:

Physisches Formatieren von einzelnen Spuren oder gesamten Disketten. Die ersten sechs Parameter entsprechen dem FDC-Befehl FORMAT A TRACK.

befehl	0D für FM und 4D für MFM
nummer	Laufwerksnummer
N	Sektorlängencode 0 für 128, 1 für 256, 2 für 512 und 3 für 1024
SC	Sektoranzahl pro Spur
GPL	Lange der Lücke zwischen 2 Sektoren
D	Fülldatenbyte, damit werden alle Sektoren beschrieben
tab	Nummer einer Sektornummerntabelle für physischen Sektorversatz
H0v	Bit 7=1 Seite 0 formatieren Bit 0-6 Nummer der ersten zu formatierenden Spur
H1b	Bit 7=1 Seite 1 formatieren Bit 0-6 Nummer der letzten zu formatierenden Spur

Beispiele zum Formatieren von Disketten:

1. 2 Seiten, 80 Spuren, 5 Sektoren je 1024 Byte MFM (800K)

```
FORMAT 4D 0 3 5 74 E5 3 80 CF
```

2. 1 Seite, 40 Spuren, 16 Sektoren je 25 6 Byte MFM (160K)

```
FORMAT 4D 1 1 10 32 E5 1 80 27
```

3. 1 Seite, 77 Spuren, 26 Sektoren je 128 Byte FM (250K)

```
FORMAT 0D 3 0 1A 1B E5 0 80 4C
```

Hinweis zum angebotenen Floppy-Disk-Modul

Alle beschriebenen Funktionen sind nur mit einem speziell ausgerüsteten Floppy-Disk-Modul realisierbar. Dieses Modul wurde entwickelt und erfolgreich erprobt, aber nicht produziert. Interessenten können einen Schaltplan erhalten. Von einem Nachbau dieser sehr komplexen Schaltung wird abgeraten.

Das als Ersatz angebotene Floppy-Disk-Modul hat folgende technische Daten, die nicht veränderbar sind.

Taktfrequenz 4 MHz
MFM-Modus
Ansteuerung von maximal 2 Laufwerken und 2 Motoren
keine Präkompensation
nur die Steuersignale des Laufwerkstypes 1.6 werden verarbeitet

Die Beschränkung auf Laufwerke des Typs 1.6 (80 Spuren, 2 Seiten) stellt zumeist keinen praktischen Nachteil dar, weil die Laufwerke heute preisgünstig erhältlich sind. Damit ist auch die beim CP/M übliche maximale Speicherkapazität 800 KByte möglich.

Die folgenden Kommandos dienen zur sinnvollen Benutzung der Diskette, solange kein vollständiges Disketten-Betriebssystem zur Verfügung steht. Vor dem Aufruf eines Kommandos ist mit FDM der Motor des Diskettenlaufwerkes einzuschalten und nachher wieder auszuschalten.

Kommando FSAVE

Syntax:

FSAVE *zahl*

Funktion:

Abspeichern des Speicherinhaltes von Adresse 300H bis A2FFH (40 KByte) auf 4 Spuren (Vorder- und Rückseite) der Diskette. Die Spuren müssen mit 5 * 1024 Byte formatiert sein. Die gesamte Diskette ist in 20 Bereiche von je 4 Spuren (40 KByte) eingeteilt. Die zahl (0 -13H) gibt an, auf welchen Bereich abgespeichert werden soll.

Die wichtigsten Programme z.B. der BASIC -Interpreter und der EDITOR-ASSEMBLER beginnen im Speicher ab der Adresse 300H mit dem Arbeitsspeicher. Das Ende des verwendeten Speicherbereiches läßt sich beliebig festlegen. Mit dem Kommando FSAVE kann innerhalb von 1-2 Sekunden der gesamte wichtige Speicherinhalt auf Diskette ausgelagert werden.

Kommando FLOAD

Syntax:

FLOAD *zahl*

Funktion:

Laden eines mit FSAVE abgespeicherten 40 KByte Bereiches in den Speicher-Adreßbereich 300H bis A2FFH. Die *zahl* (0-13H) gibt die Nummer des Bereiches an.

Danach kann das vor dem Abspeichern aktive Programm mit Warmstart wieder aktiviert werden. Das lästige Abspeichern und Laden von Kassette entfällt.

Mit diesen Kommandos und dem EDITOR-ASSEMBLER entstand das gesamte Betriebssystem für das 192 KByte RAM/EPROM-Modul. Es war dabei oft erforderlich in kurzen Zeitabständen zwischen mehreren großen Assembler-Quellprogrammen zu wechseln. Mit diesem Verfahren ist aufgrund der seltenen und unübertrefflich schnellen Diskettenzugriffe eine deutlich schnellere Programmentwicklung (Editieren, Übersetzen, Testen) möglich, als mit dem Assembler im Betriebssystem CP/M.

31.12.90 Lutz Elßner, Postfach 127-14, 0-8210 Freital

Softwarekomponente 8 DBIOS - DISK-BIOS

PROGRAMMBESCHREIBUNG

Zweckbestimmung

Das BIOS ist der hardwareabhängige Teil des Betriebssystems CP/M. Es ist bei jedem CP/M-kompatiblen Betriebssystem vorhanden und stellt eine genau definierte Schnittstelle zum Zugriff auf die zeichenweise arbeitenden E/A-Geräte (wie z.B. Tastatur, Bildschirm oder Drucker) sowie auf die Diskette bereit.
Die BIOS-Schnittstelle besteht aus dem sogenannten Sprungvektor und wird hauptsächlich vom BDOS, aber auch von Anwenderprogrammen genutzt. Der Aufbau der BIOS-Schnittstelle wird als bekannt vorausgesetzt, bzw. ist der umfangreichen CP/M-Literatur zu entnehmen.
In der vorliegenden CP/M-Version ist das traditionelle BIOS auf mehrere Softwarekomponenten aufgeteilt worden. Der Aufruf einer BIOS-Funktion ist (zusätzlich zum Sprungvektor) von jeder Speicherbank aus über die Schnittstelle der Softwarekomponente UP möglich.
Die hier beschriebene Softwarekomponente DISK-BIOS beinhaltet nur die Funktionen für den Diskettenzugriff, die im folgenden erklärt werden. Die Adressen der einzelnen Unterprogramme werden bei der Kaltstart-Initialisierung in die Adreßtabelle der Softwarekomponente UP eingetragen. Außerdem werden alle Steuertabellen für die Diskettenarbeit initialisiert.
Die Softwarekomponente DBIOS ruft für physische Diskettenzugriffe die Softwarekomponente FDT (Floppy-Disk-Treiber) auf, welche sich in der selben Speicherbank befinden muß.

Aufrufmöglichkeiten für BIOS-Funktionen

Die BIOS-Funktionen haben die Nummern 0 bis 19 dezimal. Der Aufruf kann von CP/M-Programmen über die standardisierte Schnittstelle (BIOS-Sprungvektor) erfolgen. Beim hier beschriebenen Betriebssystem, das sich in mehreren parallelliegenden Speicherbänken befindet, kann die Bank mit dem BIOS-Sprungvektor auch weggeschaltet sein. Deshalb gibt es für selbstgeschriebene Assemblerprogramme die Möglichkeit, das CP/M-kompatible BIOS über die Schnittstelle der Softwarekomponente UP, mit den Rufnummern 80H - 93H von jeder beliebigen Speicherbank aus, aufzurufen. Zur Ermittlung der Rufnummer ist zur Nummer der BIOS-Funktion jeweils 80H zu addieren.

BIOS-Funktion 8: HOME

Aufrufparameter : -
Rückkehrparameter : -

Einstellen der Spur 0. Im Gegensatz zu früheren CP/M-Versionen erfolgt hier kein Zugriff auf ein Diskettenlaufwerk. Fehlpositionierungen bei READ und WRITE-Zugriffen werden durch die BIOS-interne Fehlerbehandlung beseitigt.

BIOS-Funktion 9: SELDSK

Aufrufparameter : C – Laufwerksnummer 0 für A, 1 für B, 2 für C, ...
E - LOGIN-Bit (s. Text)
Rückkehrparameter : HL - Adresse des DPH des gewählten Laufwerkes oder 0000

Auswahl des (logischen) Diskettenlaufwerkes entsprechend Register C.
Wenn Bit 0 des Registers E rückgesetzt ist (z.B. E=0), dann kann eine automatische Erkennung des Formates der Diskette im ausgewählten Laufwerk erfolgen. Die automatische Formaterkennung ist aber noch nicht programmiert. Bei gesetztem Bit 0 im Register E erfolgt nur die Laufwerksauswahl.
Im Registerpaar HL wird die Adresse des zum (logischen) Laufwerk gehörenden Disk-Parameter-Headers zurückgegeben. Der DPH wird im Anschluß beschrieben.
Falls das mit Register C ausgewählte (logische)
Diskettenlaufwerk im BIOS nicht definiert ist, wird in HL der Wert 0 zurückgegeben.

BIOS-Funktion 10: SETTRK

Aufrufparameter : C - Spurnummer (ab 0)
Rückkehrparameter : -

Einstellen der (logischen) Spur entsprechend Register C.
Diese Spurnummer muß nicht der physischen Spur des Lesekopfes im Diskettenlaufwerk entsprechen. Bei 80 spurigen, zweiseitigen Laufwerken kann die Spurnummer 0 - 79 oder 0 - 159 betragen, je nachdem, ob die Spuren der Rückseite extra gezählt werden oder als Verlängerung der Vorderseite gewertet werden. Wenn die Parameter in den (anschließend beschriebenen) Parameterblöcken stimmen, berechnet das BDOS daraus vor dem Aufruf einer BIOS-Funktion die richtigen Werte und der Anwender braucht sich nicht darum zu kümmern.

BIOS-Funktion 11: SETSEC

Aufrufparameter : C - Sektornummer (ab 1)
Rückkehrparameter : -

Einstellen der (logischen) Sektornummer entsprechend Register C.
Die Zählung der Sektornummern beginnt immer bei 1. Die logische Sektornummer kennzeichnet (logische) Sektoren von 128 Byte Länge, die mit einem READ oder WRITE BIOS-RUF übertragen werden. Auf der Diskette gibt es je nach Format physische Sektorlängen von 128, 256, 512 und 1024 Bytes. Die Umrechnung übernimmt das BIOS unter Verwendung der Parameterblöcke.

BIOS-Funktion 12: SETDMA

Aufrufparameter : BC - Speicher-Adresse für die Übertragung von 128 Byte
Rückkehrparameter : -

Einstellen der Anfangsadresse eines 128 Byte langen Speicherbereiches, der bei den Funktionen READ und WRITE zum Datenaustausch benötigt wird. Zu dieser Speicheradresse wird zusätzlich die Systembank 0FFH definiert.

BIOS-Funktion 13: READ

Aufrufparameter : -
Rückkehrparameter : A - Fehlercode 0 wenn erfolgreich
1 bei Fehler
Lesen des 128 Byte langen (logischen) Sektors, der mit SETSEC eingestellt wurde von der mit SETTRK definierten (logischen) Spur der mit SELDSK ausgewählten (logischen) Diskette auf den mit SET DMA festgelegten Speicherbereich.
Falls sich der gewünschte Sektor bereits im BIOS-internen Diskettenpuffer befindet, erfolgt kein Zugriff auf das Diskettenlaufwerk, anderenfalls kann es vorkommen, daß der Inhalt des Diskettenpuffers erst zurückgeschrieben werden muß, bevor der gewünschte Sektor gelesen wird.

BIOS-Funktion 14: WRITE

Aufrufparameter : C = 0 normales Schreiben
1 Schreiben in Verzeichnis
2 Schreiben in neuen Block
Rückkehrparameter : A – Fehlercode 0 wenn erfolgreich
1 bei Fehler

Schreiben des mit SETDMA festgelegten Speicherbereiches auf den mit SETSEC und SETTRK festgelegten (logischen) Sektor der mit SELDSK ausgewählten Diskette.
Falls sich der gewählte Sektor bereits im BIOS-internen Diskettenpuffer befindet, wird er überschrieben. Ein sofortiges Abspeichern auf Diskette

erfolgt nur, wenn C=1 war oder der letzte Sektor des Diskettenpuffers beschrieben wurde.
Falls sich der gewählte Sektor nicht im BIOSS-internen Diskettenpuffer befindet, muß der aktuelle Inhalt des Puffers gegebenenfalls zuerst auf seinen ursprünglichen Platz einer Diskette zurückgeschrieben werden und anschließend von der ausgewählten Diskette neu geladen werden. Dieses vorhergehende Lesen (preread) unterbleibt, wenn der Aufrufparameter C=2 war. Auch dieser Aufrufparameter wird vom BDOS sinnvoll eingestellt und ist für den Anwender von untergeordneter Bedeutung.

BIOS-Funktion 16: SECTRAN

```
Aufrufparameter   :     BC - umzuwandelnde Sektornummer (ab 0)
                        DE - Adresse der Umwandlungstabelle (unbenutzt!)
Rückkehrparameter :     HL - umgewandelte Sektornummer(ab 1)
```

Diese historische Funktion diente zur Realisierung von Sektorversatz auf der Diskette, um die Zugriffe zu beschleunigen. Der Sektorversatz wird heute bereits beim Formatieren von Disketten realisiert. Für ältere Disketten mit (logischem) Sektorversatz stellt das BIOS intern eine Umrechnungsmöglichkeit bereit (erstmalig beim CP/A angewendet).
Weil diese Funktion aber vom BDOS weiterhin aufgerufen wird, muß ein nicht vorhandener Sektorversatz vorgetäuscht werden. Die Sektornummer aus Register BC wird um 1 erhöht und in HL zurückgegeben. Der in DE übergebene Wert wird ignoriert.

Behandlung von Fehlern bei Diskettenzugriffen

Bei den BIOS-Rufen READ und WRITE auftretende Fehler werden durch Rückgabe des Wertes 1 im Register A gemeldet. Grobe logische Fehler (unerlaubte Parameter bei SELDSK, SETTRK, SETSEC) werden programmtechnisch erkannt und sofort gemeldet. Schreib- und Lesefehler, die vom Floppy-Disk-Controller erkannt werden, führen zu mehreren Wiederholungen mit neuer Kopfpositionierung. Ist eine Wiederholung erfolgreich, merkt der Nutzer davon nichts. Erst bei mehreren Mißerfolgen wird ein Fehler gemeldet. Dazu wird die komplette Befehls- und Ergebnisphase des FDC U 8272 angezeigt, um die Fehlerursache exakt ermitteln zu können. Zur Entschlüsselung der vielen Bytes ist aber die genaue Kenntnis der Funktionsweise des FDC erforderlich.
Bei geöffnetem Laufwerkshebel wird eine längere Zeit auf das READY-Signal gewartet, um dem Nutzer das Schließen des Hebels noch zu ermöglichen. Passiert das nicht in der zur Verfügung stehenden Zeit, wird auch hier ein Fehler gemeldet, der zum Abbruch führt.
Ein Vergleichslesen nach dem Schreiben wird nicht durchgeführt, dadurch können also auch keine Fehlermeldungen hervorgerufen werden.

Realisierung verschiedener Diskettenformate

Die soeben beschriebene BIOS-Schnittstelle kennt nur 128 Byte lange (logische) Sektoren, (logische) Spuren und (logische) Diskettenlaufwerke.
Die Realität ist aber wesentlich verschiedener und gerade beim CP/M katastrophal vielfältig:

- es gibt ein- und zweiseitige Disketten
- es gibt Disketten mit 40, 77, 80, ... Spuren
- es gibt physische Sektorlängen von 128, 256, 512 und 1024 Byte
- es gibt manchmal Systemspuren
- das Verzeichnis ist unterschiedlich groß
- es gibt Speicherkapazitäten von etwa 100 KByte bis über 1 MByte
- ein physisches Laufwerk kann unter mehreren logischen Bezeichnungen angesprochen werden
- es gibt RAM-Floppys und andere virtuelle Disketten

Unabhängig von den genannten physischen Unterschieden gibt es weitere Probleme bei der logischen Verwaltung.

Block und Extent

Zur Adressierung eines 128 Byte langen Bereiches oder sogar nur eines Bytes daraus wurden sehr große Zahlen benötigt. Um den Speicherplatz für die Verwaltung zu reduzieren erfand man den Block.
Ein Block ist die kleinste logische Einheit für die Abspeicherung einer Datei. Die Blockgröße betrug zunächst 1 KByte. Das bedeutet, daß Dateien mit einer Lange von 1 bis 1024 Bytes immer einen Platz von 1 KByte auf der Diskette belegen. Bei 1025 Byte werden dann 2 Blöcke (2 KByte) belegt usw.
Wenn man für die Abspeicherung der Blocknummer im Verzeichnis 1 Byte einplant, kann eine Diskette maximal 256 Blöcke (256 KByte) enthalten. Die Diskettenkapazität ist aber über 256 KByte hinausgewachsen. Demzufolge wurde die Blockgröße auf 2 KByte erhöht und man kam auf 256 * 2 KByte = 512 KByte.
Das reichte aber immer noch nicht. Für die Abspeicherung der Blocknummer wurden dann 2 Byte im Verzeichnis vorgesehen. Bei 2 KByte Blockgröße kommt man jetzt bis 131072 KByte bzw. 128 MByte theoretische Diskettenkapazität. Das reicht für CP/M aus. (Man kann aber die Blockgröße weiter erhöhen.)

In einem Verzeichnis-Eintrag von 32 Bytes sind die letzten 16 Bytes für die Blocknummern vorgesehen. Bei 1-Byte-Blocknummern passen 16 Stück in einen Eintrag. Das ergibt 16 KByte bei 1 KByte Blockgröße und 32 KByte bei 2 KByte Blockgröße. Von den 2-Byte-Blocknummern passen nur 8 Stück in einen Eintrag. Das ergibt bei 2 KByte Blockgröße ebenfalls 16 KByte.
Diese 16 KByte im Verzeichniseintrag heißen Extent. Es können also 1 oder 2 Extents in einem DIR-Eintrag vorhanden sein. Die Extentnummer steht in jedem DIR-Eintrag gleich hinter dem Dateityp. Bei längeren Dateien werden dann weitere Verzeichnis-Eintragungen mit den selben Dateinamen, nur anderen Extent-Nummern angelegt. Bei Direktzugriff auf Dateien ist es auch erlaubt, daß zwischendurch Blöcke oder Extents gar nicht belegt sind.

DPH und DPB

Um dieses gigantische Durcheinander etwas zu ordnen gibt es für jede logische Diskette einen Disk-Parameter-Header (DPH), der hauptsächlich Adressen von anderen Speicherbereichen enthält. Diese Bereiche werden vom BIOS zur Verfügung gestellt, aber vom BDOS benutzt. Dadurch ist es möglich, das BDOS völlig rechnerunabhängig zu gestalten, auch die Anzahl der logischen Laufwerke wird nur im BIOS festgelegt. Im CP/M des 192 KByte RAM/EPROM-Moduls sind die 5 logischen Laufwerke A-E definiert.

Die Adresse des Disk-Parameter-Blockes (DPB) ist auch im DPH zu finden. Darin wird es nun endlich konkret. Hier befinden sich Angaben über die soeben beschriebenen Blöcke, Extents, Verzeichnis-Eintragungen und System-spuren.

Die beiden Bereiche DPH und DPB sind im CP/M standardisiert und bei jedem CP/M System einheitlich. Sie müssen vom BIOS mit Daten gefüllt werden, während die anderen im DPH adressierten Speicherbereiche XLT, DIRBUF, CSV und ALV nur speicherplatzmäßig vom BIOS bereitzustellen sind. Der Inhalt wird vom BDOS verwaltet.

Leider reichen die Angaben für ein flexibles CP/M -System immer noch nicht ganz aus. Es fehlen Informationen zum Laufwerkstyp (8 Zoll, 5.25 Zoll , SS, DS, SD, DD, Spuranzahl usw.), zum physischen Diskettenformat (Sektorlänge, Sektoren pro Spur), zur Anordnung und Numerierung der Spuren (die Firmen haben ihre eigenen Hausformate erfunden wegen Kopierschutz oder aus anderen konkurrenztechnischen Gründen). Diese nicht standardisierten Angaben befinden sich beim vorliegenden BIOS in weiteren Tabellen, belegen aber keinen Anwenderspeicher weil sie in der Speicherbank für Diskettenzugriffe (Bank 14) stehen. Die Anfangsadresse dieser nicht standardisierten Erweiterung wurde an den jeweiligen DPH ab Position DPH+16 (dez.) angefügt.

DISK - PARAMETER - HEADER DPH

Jedes logische Laufwerk hat einen eigenen DPH. Die Adresse des DPH ist mit der BIOS-Funktion 9 (SELDSK) zu ermitteln. Der DPH befindet sich in der Systembank 15. Die Bytes 00-15 (dez.) sind im CP/M standardisiert.

Byte	Name	Bedeutung
00/015	XLT	Adresse einer Sektornummern-Umrechnungstabelle, bei 0000 ist keine Tabelle vorhanden.
02/03		Arbeitsspeicher für BDOS, mit 0000 initialisiert.
04/05		-
06/07		-
08/09	DIRBUF	Adresse eines 128 Byte langen Speicherbereiches, wo BDOS 4 Einträge des aktuellen Verzeichnis zwischenspeichert. Diese Adresse ist in allen DPH gleich, d.h., DIRBUF ist nur einmal vorhanden.
10/11	DPB	Adresse des zugehörigen DISK-PARAMETER-BLOCKes (DPB). Jedes (logische) Laufwerk hat einen eigenen DPB.
12/13	CSV	Adresse eines Puffers für die Prüfsummen von Verzeichnis - Eintragen. Die Prüfsummen dienen zur Erkennung eines (unerlaubten) Diskettenwechsels. Die Länge des Puffers ist im DPB unter der Bezeichnung CKS angegeben und von der Größe des Verzeichnis abhängig. Jedes Laufwerk hat einen eigenen CSV.
14/15	ALV	Anfangsadresse des DISK-ALLOCATION-VEKTORs. Dieser Vektor hat 1 Bit pro Block zur Kennzeichnung ob dieser Block belegt oder frei ist. Die Länge des ALV ergibt sich aus der Anzahl der Blöcke pro Diskette durch 8. Diese Länge steht im DPB unter DSM. Das Verzeichnis selbst beginnt bei Block 0 und belegt deshalb die ersten Bits. Die 16 Bit AL0 und AL1 aus dem DPB werden vom BDOS zur Kennzeichnung der Verzeichnis-Blöcke in den ALV kopiert. Jedes Laufwerk hat einen eigenen ALV. Ende des standardisierten DPH.
16/17	DPBF	Adresse des zusätzlichen Disk-Parameter-Blocks. Er enthält interne Parameter zu Diskettenlaufwerken und Formaten. Dieser Speicherbereich befindet sich nicht in der Systembank 15. Jedes Laufwerk hat einen eigenen DPBF.

Die Speicherbereiche DIRBUF, DPB, CSV und ALV befinden sich in der System-bank 15 und sind vom BDOS über ihre Anfangsadresse im DPH auffindbar. Der Bereich DPBF liegt in der "Disketten-Bank" 14.

DISK - PARAMETER - BLOCK DPH

Jedes logische Laufwerk hat einen eigenen DPB. Die Adresse des DPB steht im DPH oder ist mit der BDOS-Funktion 31 zu ermitteln. Der DPB befindet sich in der Systembank 15. Er hat eine Länge von 15 (dez.) Bytes und ist im CP/M standardisiert.

Byte	Name	Bedeutung
00/01	SPT	Anzahl der (logischen) 128 Byte langen Sektoren pro (logische) Spur.
02	BSH	Blockverschiebungsfaktor
03	BLM	Blockmaske Der Begriff Block wurde erklärt. Abhängig von der Blockgröße werden hier folgende Werte eingetragen: Blockgröße 1K 2K 4K usw. BSH 3 4 5 BLM 7 15 31
04	EXM	Extentmaske Der Wert ist von der Blockgröße und davon abhängig, ob 1 oder 2 Byte große Blocknummern verwendet werden. Blockgröße 1K 2K 2K 4K 4K u.s.w. Blocknummer 1 Byte 1 Byte 2 Byte 1 Byte 2 Byte EXM 0 1 0 3 1 EXM = 0: 16K (1 Extent) pro Verzeichnis-Eintrag EXM = 1: 32K (2 Extents) pro Verzeichnis-Eintrag usw.
05/06	DSM	Anzahl der Blöcke pro Diskette minus 1. Hieraus ist die Diskettenspeicherkapazität abzuleiten.
07/08	DRM	Anzahl der Verzeichnis-Eintragungen minus 1. Ein Verzeichnis-Eintrag ist 32 Byte lang.
09 10	AL0 AL1	16 Bit-Vektor, in dem die vom Verzeichnis belegten Blöcke vermerkt sind. Diese 16 Bit werden vom BDOS an den Anfang des DISK-ALLOCATION-VEKTORs kopiert und dienen zur Reservierung der Blöcke, die vom Verzeichnis selbst belegt werden. Die Zählung beginnt beim höchsten Bit. Bei 2 belegten Blöcken wäre AL 0 = 11000000B und AL1 = 00000000B.
11/12	CKS	Größe des Puffers für die Prüfsummen der Verzeichnis-Einträge. Die Anfangsadresse des Puffer CSV steht im DPH. Für je 4 DIR -Einträge zu 32 Byte (= 128 Byte entsprechend 1 log. Sektor) wird eine Prüfsumme von 1 Byte Länge gebildet. Diese Maßnahme dient zur Erkennung von (unerlaubtem) Diskettenwechsel. Bei nichtwechselbaren Disketten (RAM-Floppy) wird hier 0000 eingetragen.
13/14	OFF	Anzahl der Systemspuren (OFFSET). Die Systemspuren beginnen immer bei der niedrigsten Spur der Diskette (Spur 0). Sie werden in die gesamte logische Gliederung der Diskette in Blöcke, Extents usw. nicht mit einbezogen. Weil die Spurnumerierung mit 0 beginnt, beinhaltet der Wert OFF die (logische) Spurnummer, wo Block 0 mit dem Verzeichnis in Sektor 1 beginnt. OFF ist also der Offset des logischen Diskettenanfangs (Block 0) zum physischen Diskettenanfang (Spur 0). Wenn keine Systemspuren vorhanden sind, enthält OFF den Wert 0000. In den Systemspuren wird normalerweise das Betriebssystem CP/M selbst abgespeichert. Das vorliegende Betriebssystem befindet sich vollständig in EPROMs auf dem 192 KByte RAM/EPROM-Modul und muß nicht von Diskette geladen werden. Die Systemspuren sind unbenutzt bzw. können das Betriebssystem für einen beliebigen anderen Rechner enthalten.

Zusätzlicher, nicht standardisierter Diskettenparameterblock DPBF

Jedes logische Laufwerk hat außer den im CP/M standardisierten DPH und DPB im hier beschriebenen Betriebssystem noch einen zusätzlichen Bereich DPBF, wo weitere Angaben zum Diskettenzugriff enthalten sind. Die Adresse dieses Speicherbereiches steht im DPH im Anschluß an die standardisierten Bytes auf Position DPH Der DPBF belegt keinen Anwenderspeicher, denn er befindet sich in der RAM-Bank für Diskettenoperationen (Bank 14).

Byte	Name	Bedeutung
00	DPBFL	DPB-Flag. Jedes Bit hat eine eigene Bedeutung: Bit 1-0 Zahl der (System-) Spuren mit Format 26*128 (0-3) Bit 2 =1, wenn FM zu erzwingen Bit 3 =1, wenn DS u. Vorder- u. Rückseite von außen nach innen =0, wenn D5 und Rückseite von innen noch außen Bit 4 =1, --> Bit 3 benutzen =0, wenn DS und ungerade log. Spuren auf Rückseite Bit 5 =1, wenn DS (double sided) --> Bit 4 benutzen =0, wenn SS (single sided) --> Bit 3, 4 wirkungslos Bit 6 =1, wenn keine automatische Formaterkennung in SELDSK Bit 7 =1, wenn die folgenden Bytes nicht benutzt werden (RAM-Floppy)
01	DPBSL	Sektorlängencode für physische Sektoren auf Diskette. Sektorlänge: 128 256 512 1024
	DPBSL	: 0 1 2 3
02	DPBBM	Anzahl der 128 Byte langen Sektoren im BIOS-internen Diskettenpuffer minus 1. Der Diskettenpuffer ist 1 KByte groß, DPBBM hat deshalb den Wert 7.
03	DPBST	Stepimpulse pro Spur (1 oder 2). Bei 40 spurigen Disketten in 80 spurigen Laufwerken müssen pro Spur vom Schreib-/Lesekopf 2 Schritte zurückgelegt werden.
04	DPBSN	Verschiebung der Sektornummern auf der Rückseite wird verwendet, wenn die physische Sektornummer auf der Rückseite nicht mit 1 beginnt, sondern als "Verlängerung" der Vorderseite gezählt wird. Sonst ist DPBSN =0.
05	DPBTR	logische Anzahl der Spuren Wenn die Rückseite einzeln gezählt wird, kann die Diskette z.B. 160 logische Spuren haben.
06	DPBDN	physische Laufwerksnummer (0 -3) Der DPH, DPB usw. existiert für jedes logische Laufwerk. Es ist möglich mit mehreren logischen (z.B. A, B und D) auf dasselbe physische Laufwerk zuzugreifen (wenn nur ein Laufwerk vorhanden). Dabei kann jeweils ein anderes Diskettenformat realisiert werden, z.B. A:780K, B:800K und D:148K. Es dürfen 16 logische Laufwerke (A -P) definiert werden. Im Gegensatz dazu darf DPBDN nur die Werte 0-3 für die 4 vom Floppy-Disk-Controller ansprechbaren physischen Laufwerke annehmen. Jeder der 4 physischen Laufwerksnummern ist wieder eine Tabelle der physischen Laufwerksparameter zugeordnet. Darin stehen Angaben zum Typ, Spuranzahl, Präkompensation und Zeitangaben für die Kopfpositionierung.
07	DPBGP	Länge der GAP 3. GAP 3 ist die Lücke, die noch dem Schreiben eines physischen Sektors noch aufgezeichnet wird. Wenn GAP 3 zu groß ist, wird der Anfang des nächsten Sektors überschrieben.
08/09	DPBTT	Adresse der Sektortranslatetabelle. Die Tabelle enthält die Reihenfolge in der die Sektoren von der Diskette logisch zusammengehören. Normalerweise besteht die Tabelle aus den Zahlen 1 bis 26 in geordneter Reihenfolge. Andere Reihenfolgen sind zur Zeit nicht erforderlich. Die Tabelle kann im ROM stehen und für alle logischen Laufwerke benutzt werden.

Physischer Sektorversatz

Der logische Sektorversatz darf nicht mit den physischen Sektorversatz beim Formatieren von Disketten verwechselt werden, der große Bedeutung hat. Hierbei befinden sich die Sektoren tatsächlich in ungeordneter Reihenfolge auf der Diskette.
Der Sektorversatz hat entscheidenden Einfluß auf die Zugriffsgeschwindigkeit der Diskette und ist von der Reaktionszeit der Soft- und Hardware des jeweiligen Rechners abhängig. Mit der gleichen Diskette können deshalb auf verschiedenen Rechnern erhebliche Zeitunterschiede beobachtet werden, wahrend andere Disketten wieder woanders schneller sind.

Beispiel: Disketten mit 5 Sektoren je 1024 Byte können folgende Sektorreihenfolgen haben:

```
1 2 3 4 5   bei SCPX 1715
1 5 4 3 2   bei CP/A
1 4 2 5 3   bei SCPX 5105
```

Wie in den anderen Softwarekomponenten des 192 KByte RAM/EPROM-Moduls gibt es auch in der Komponente DBIOS ein residentes Kommando:

Kommando RDT

Syntax:

RDT *lw spur sektor*

Funktion:

Ausführung einer Disketten-Leseoperation über die BIOS-Schnittstelle (READ TEST). Die Parameter *lw*, *spur* und *sektor* entsprechen den bei den BIOS-Funktionen SELDSK, SETTRK und SETSEC im Register C übergebenen Parametern. Sie müssen hier als Hexadezimalzahl eingegeben werden.

lw : 0=A 1=B 2=C
spur : logische Spurnummer ab 0
sektor : logische Sektornummer ab 1

Der Inhalt der gelesenen Sektors (128 Byte) wird am Bildschirm angezeigt. Mit diesem Kommando kann schnell ein Blick auf eine Diskette geworfen werden.

Nachwort

Die Beschreibung des DISK-BIOS erfolgte aufgrund der Erfahrungen, die beim Programmieren entstanden sind. Besonders wichtige oder besonders verwirrende Sachverhalte wurden mehrmals, an verschiedenen Stellen erläutert. Die Beschreibung soll eine Grundlage bilden, noch der erfahrene Nutzer den Inhalt der Disketten-Parameterblöcke verstehen und an eigene Anwendungsfälle anpassen können. Es wurde versucht, Begriffe wie Block und Extent praxisnah zu erläutern, was bekannten CP/M-Beschreibungen noch nie gelungen ist. Auf überall beschriebene Einzelheiten, wie die BIOS-Schnittstelle wurde verzichtet.

Die Softwarekomponente DISK-BIOS wurde nicht von einem bestehenden CP/M-System übernommen, sondern grundlegend neu konzipiert. Dadurch konnte sehr viel RAM-Speicher eingespart werden und die Programme wurden bis hinunter in die tiefste Hardware (Floppy-Disk-Controller U 8272) effektiv gestaltet. Entgegen der "gelehrten" Verfahrensweise wurde die gesamte Softwareentwicklung "auf der Leiterplatte" mit der Hardwaresteuerung in der Softwarekomponente FDT begonnen. Darauf aufbauend entstanden immer höhere Schichten, die ihre Parameter sofort in die Befehlsphase des U 8272 eintragen konnten ohne sie im RAM mehrmals umzuschichten.

Natürlich wurden, um Kompatibilität auch in den Feinheiten zu erreichen, mehrere bestehende CP/M -Systeme zu Rate gezogen, aber der schon mehrfach geänderte und total aufgeblähte Quelltext dieser Vorbilder wurde für das BIOS in keinem einzigen Fall übernommen.

31.12.90 Lutz Elßner, Postfach 127-14, 0-8210 Freital

Softwarekomponente 9 BD – BDOS-Schnittstelle

PROGRAMMBESCHREIBUNG

Zweckbestimmung

Die Softwarekomponente BD dient ähnlich wie die Softwarekomponente UP zum Aufruf von maximal 256 Unterprogrammen in beliebigen Speicherbänken. Hier wurde allerdings besonderer Wert auf die vollständige Kompatibilität zur CP/M-BDOS-Schnittstelle gelegt. Diese Schnittstelle kann hier noch wesentlich erweitert werden.

Aufruf

Weil die Adresse 5, die beim CP/M zum Aufruf des BDOS benutzt wird, im Betriebssystem Z 9001 belegt ist, wird hier auf die Adresse 8 ausgewichen. Der Aufruf kann mit dem 1 Byte Befehl RST 08H erfolgen. Im CP/M-Modus wird die vollständige Kompatibilität des Aufrufes hergestellt. Über RST 08H ist die Nutzung der BDOS-Funktionen von beiden Systemzuständen aus möglich.

Aufrufparameter:	C = Rufnummer
	DE = Übergabeparameter
Rückkehrparameter:	A = 8 Bit Werte
	HL = 16 Bit Werte

Funktionsumfang

Die Softwarekomponente BD enthält zur Zeit nur den Rahmen zum Aufruf von Unterprogrammen in beliebigen Bänken. Die Funktionen selbst müssen von weiteren Softwarekomponenten bereitgestellt werden, die sich in die Adreß-tabelle von BD eintragen (z.B. TAPE).

CP/M-BDOS

Das komplette CP/M-BDOS mit allen Funktionen zur Ein- und Ausgabe über Schnittstellen und Disketten steht ebenfalls als externe Softwarekomponente zur Verfügung. Diese Software muß in einer eigenen Speicherbank in einem EPROM des 192 KByte RAM/EPROM-Moduls untergebracht werden und vor der Benutzung in den RAM in Bank 15 umgeladen werden. Wenn der RAM vorhanden ist, trägt sich dieses BDOS in die Adreßtabelle der Softwarekomponente BD ein und alle 40 Funktionen sind verfügbar. Ob ein Aufruf dieser Funktionen dann Sinn hat, hängt auch von der zur Verfügung stehenden Hardware ab. Für die Diskettenfunktionen ist außer den Treiberprogrammen im BIOS logischer-weise auch ein Diskettenlaufwerk mit passendem Modul oder ein RAM-Floppy erforderlich.

09.03.91 Lutz Elßner, Postfach 127-14, D-8210 Freital

Softwarekomponente 10 SIO - serielle Treiber

PROGRAMMBESCHREIBUNG

Zweckbestimmung

Dieses Treiberprogramm dient im Zusammenhang mit einem beliebigen Drucker - bzw. Schreibmaschinenmodul zur Realisierung serieller Schnittstellen. Darüberhinaus können auch selbstgebaute V24 oder IFSS-Module mit SIO U856 benutzt werden. Die Portadressen und alle Parameter können bei der Initialisierung angegeben werden. Diese Softwarekomponente gilt nicht für parallele Datenübertragung (wie CENTRONICS), dafür ist die Softwarekomponente PIO zuständig.

Hardwarevoraussetzungen

Standardmäßig ist der Einsatz eines Drucker- oder Schreibmaschinenmoduls vorgesehen. Anschließend wird erklärt, daß für den hier beschriebenen Einsatzfall alle jemals hergestellten Typen dieser Module gleichwertig sind. Im folgenden wird daher nur noch der Begriff Druckermodul verwendet.
Die Unterschiede bestehen im EPROM oder ROM auf dem Modul und im Steckverbinder. Allen Modulen ist gemeinsam:

Portadresse CTC	: A8H
Portadressen SIO	: B0H (Daten) B2H (Steuerung)
Interface	: V 24 (nur senden)
Protokoll	: DTR (Hardware-Handshake)
Interfacesignale	: TxD (Sendedaten, Ausgang) CTS (Senderfreigabe, Eingang)
Speicherbereich EPROM	: B800H-BFFFH (2 KByte) abschaltbar

Der EPROM ist auf allen Modulen mittels DIL-Schalter abzuschalten (beide Schalter aus!), weil die hier beschriebene Softwarekomponente den Treiber im EPROM ersetzt. Somit ist es unbedeutend, welche Software sich im EPROM befindet.
Falls in Spezialfallen doch mit dem eingebauten EPROM gearbeitet werden soll, darf der Speicherbereich nicht gleichzeitig durch andere Module belegt werden. Auf dem 192 KByte RAM/EPROM-Modul ist in diesem Fall mindestens der Adreßbereich A000H-BFFFH durch DIL-Schalter S4 abzuschalten.

Falls zufällig der am Modul angebrachte Stecker zum vorhandenen V24 - Drucker paßt, kann das Modul auch mit dieser Softwarekomponente zum Drucken benutzt werden. Die Initialisierung des Treiberprogrammes erfolgt mit den Kommandos, die am Ende dieser Beschreibung erklärt sind.

Alle Module sind herstellerseitig nur für die Richtung Ausgabe zum Drucker vorbereitet. Eine bidirektionale Arbeitsweise (Ausgabe und Eingabe)ist erst nach einem kleinen Umbau und Auswechseln des Kabels möglich.

Begriffserklärungen, Definitionen der seriellen Schnittstellen

V.24 Definition der Schnittstellenleitungen
Am Druckermodul werden nur die folgenden Leitungen verwendet:

TxD - Transmitter Data (Sendedaten, Ausgang)
CTS - Clear To Send (Senderfreigabe, Eingang)
SG - Signal Ground (gemeinsame Masse)

nach dem Umbau werden folgende Empfängerleitungen ergänzt:

RxD - Receiver Data (Empfangsdaten, Eingang)
DTR - Data Terminal Ready (Empfänger bereit, Ausgang)

Bei der Verbindung zweier Geräte sind die Leitungen über Kreuz, also TxD mit RxD und CTS mit DTR und umgekehrt zu verbinden. Die Abschirmung kann an SG angeschlossen werden.

RS 232 Definition der elektrischen Bedingungen der Schnittstelle

logische 1 -3 bis -15 Volt
logische 0 +3 bis +15 Volt max. Kabellänge: 15 Meter

Diese Definitionen wurden von internationalen Organisationen festgelegt, werden aber in der Praxis oft nicht so genau auseinandergehalten.

Die folgende Definition für IFSS ist hier zum Vergleich wiedergegeben. Die Treibersoftware kann mit dem XON/XOFF-Protokoll auch die IFSS-Hardware ansteuern, aber die dafür erforderlichen Module sind nicht so leicht erhältlich.

IFSS serielles Interface, das nicht mit Spannungspegeln (wie RS 232) sondern mit potentialfreien Stromschleifen arbeitet.

Für das IFSS-Interface werden grundsätzlich nur die Steuerleitungen Sendedaten (TxD) und Empfangsdaten (RxD) verwendet. Für jedes dieser beiden Signale ist 1 Leitungspaar erforderlich, wo im Zustand logisch 0 ein Strom von 0-3 mA und im Zustand logisch 1 ein Strom von 15-25 mA fließt. Die Stromeinspeisung kann an einer beliebigen Seite (Sender oder Empfänger) erfolgen. Die andere Seite ist grundsätzlich über Optokoppler potentialfrei angeschlossen. Dieses Verfahren ist sehr störsicher und erlaubt Leitungslängen bis 500 m.

Aufgrund des Verzichts auf die Steuerleitungen CTS und DTR ist eine Verständigung zwischen Empfänger und Sender (Handshake) nur durch Übertragung von Steuerzeichen über die andere Datenleitung möglich (Software-Handshake). Diese Funktionen erfüllt das XON/XOFF-Protokoll.

Definitionen der Datenübertragungsprotokolle

DTR-Protokoll (Hardware-Handshake)

Außer den Datenleitungen TxD und RxD sind die Steuerleitungen DTR (des Empfängers) und CTS (des Senders) zu verbinden. Der Empfänger teilt dem Sender durch ein statisches Signal mit, ob er bereit ist Daten aufzunehmen. Dadurch kann der sendende Rechner zum Warten veranlaßt werden, während z.B. der Drucker die zuvor übertragenen Daten ausdruckt.
Die Steuersignale DTR und CTS werden von der Treibersoftware bedient. Dies ist nur bei V24-Interface möglich. Bei IFSS fehlen diese Signale, weshalb das DTR-Protokoll nicht anwendbar ist.

XON/XOFF-Protokoll (Software-Handshake)

Die Bereitschaft oder Nichtbereitschaft teilt der Daten-Empfänger durch Senden eines Steuerzeichens über die Datenleitung in der entgegengesetzten Richtung dem Sender mit. Der Sender empfängt das Steuerzeichen und stoppt die Übertragung oder setzt sie fort. Folgende Steuerzeichen werden verwendet:

Code	Bezeichnungen	Bedeutung
11H	DC1 oder XON	Freigabe für Zeichenübertragung
13H	DC3 oder XOFF	Sperre für Zeichenübertragung

außer diesen einheitlich definierten Steuerzeichen kann es noch zahlreiche andere geben z.B.:

14H	DC4	Fehlermeldung

Die Kommunikation bei diesem Protokoll ist ziemlich mehrdeutig und ein ständiges Wechseln der Geräte ohne sorgfältige Vorbereitung führt meistens zu großen Problemen. Deshalb hat das einfachere DTR-Protokoll eine größere Verbreitung gefunden.

Das XON/XOFF-Protokoll muß für IFSS-Hardware immer benutzt werden. bei V24-Hardware kann wahlweise das DTR- oder XON/XOFF-Protokoll benutzt werden. Die zur Verfügung stehenden Druckermodule sind nur für V24 ausgelegt. Die Treibersoftware erlaubt wahlweise beide Protokollarten. Die Kommandos zur Auswahl des Prototrolls und Initialisierung werden später beschrieben.

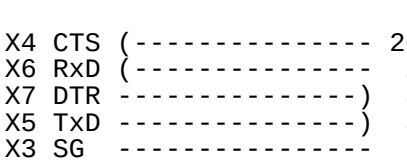
Hinweis: Die genannten Protokolle sind allgemeingültig für Standardhardware wie Drucker, Plotter, Datenübertragung usw. entwickelt worden. Besondere Hardware wie Meßgeräte, Kassettengeräte usw. können auch mit V24 oder IFSS-Interface arbeiten, benötigen aber eigene gerätespezifische Treiberprogramme.

Umbau des Druckermoduls auf bidirektionalen Betrieb

Alle handelsüblichen Druckermodule sind nur für Senden vorbereitet. Diese Betriebsart reicht aus, wenn man nur drucken will. Falls an die V24-Schnittstelle aber ein anderes Gerät z.B. ein zweiter Rechner zur Datenübertragung oder das EPROM-Programmiergerät EPROG 27011 (siehe Softwarekomponente EPROG) angeschlossen werden soll, muß mit dem Modul auch empfangen werden können. Folgender Umbau ist durchzuführen:

1. Auf der Rockseite der Leiterplatte am SI0 (U 8560) den Leiterzug vom Anschlug 17 (RTSA) trennen und mit einer kleinen Drahtbrücke an 16 (DTRA) anlöten. (Dieses Signal wurde im Originalzustand nicht benutzt.)

2. Auswechseln des Kabels. Es wird vieradriges abgeschirmtes Kabel benötigt. Für kurze Entfernungen ist auch mindestens 5 adriges Kabel ohne Schirm verwendbar. Auf der Leiterplatte befinden sich links oben 4 Anschlußpunkte (X4-X7). Davon sind im Originalzustand aber nur die äußeren (X4, X5) benutzt. An die zwei dazwischenliegenden Lötpunkte (X6, X7) werden die zusätzlichen Leitungen angeschlossen. Die Masseleitung SG (Abschirmung) wird unterhalb der Zugentlastung an die Lötöse angeschlossen. Die Lötösen X3-X7 sind auf dem Druckermodul so angeordnet, wie sie hier gezeichnet sind.



Die Anschlußbelegung der Signale CTS, TxD, SG entspricht der Originalbelegung mit 25 poligem Stecker. An der rechten Seite sind die Signalbezeichnungen angegeben, die für die Buchse an einem angeschlossenen Gerät gelten. Die Signale sind also bereits gekreuzt. Der Stecker kann unmittelbar an einen Drucker oder das EPROM-Programmiergerät angesteckt werden. Falls ein anderer Stecker erforderlich ist, kann anhand der Signalnamen die richtige Belegung ermittelt werden. Auf alle Falle müssen die drei Verbindungen CTS, TxD und SG wieder so angeschlossen werden, wie vor dem Auswechseln des Kabels.

Überblick über die Treiberprogramme

Nach diesem ausführlichen, aber notwendigen Ausflug in die Hardware soll jetzt die Initialisierung und Funktion der Treiberprogramme beschrieben werden.

Die Initialisierung erfolgt durch Eingabe des Treibernamens mit Parametern und nicht mit der ASGN-Anweisung des Betriebssystems Z 9001.
Außerdem ist eine automatisch e Kaltstartinitialisierung vorgesehen. Damit wird die V24 -Schnittstelle bereits bei der Systeminitialisierung aktiviert und der Nutzer braucht sich nicht darum zu kümmern. Zwei Kommandos stehen für die Initialisierung "von Hand" zur Verfügung:

Kommando DTR

Syntax:

DTR *p1 p2* ...

Kommando XON/XOFF

Syntax:

XON/XOFF *p1 p2* ...

Funktion:

Wie aus dem Namen zu erkennen, dienen die Kommandos zur Auswahl des DTR- und des XON/XOFF-Protokolls. Die Parameter sind bei beiden Kommandos identisch, deshalb werden sie gemeinsam beschrieben.
Die Parameter *p1* und *p2* geben an, welchem Gerät und welchem Kanal das Treiberprogramm zugewiesen werden soll. Die Bezeichnungen und die Funktionsweise sind CP/M kompatibel. Zum Aufruf der logischen Geräte existieren im Betriebssystem CP/M definierte Systemschnittstellen.
Für beide Parameter dürfen jeweils nur die Ziffern 0, 1, 2, oder 3 verwendet werden.

P1 gibt das logische Gerät an:

0 CON: 1 RDR: 2 PUN: 3 LST:

p2 gibt den zugehörigen Kanal an. Die Bezeichnung hängt vom mit Parameter *p1* gewählten log. Gerät ab:

0 TTY:	0 TTY:	0 TTY:	0 TTY:
1 CRT:	1 PTR:	1 PTP:	1 CRT:
2 BAT:	2 UR1:	2 UP1:	2 LPT:
3 UC1:	3 UR2:	3 UP2:	3 UL1:

Wie im CP/M üblich, erfolgt die Auswahl des zutreffenden Kanals durch Auswertung von je 2 Bits im I/O-Byte. Die betreffenden Bits im I/O-Byte werden bei der Initialisierung automatisch entsprechend *p2* gesetzt.

Beispiel 1:

Soll ein V24-Drucker mit DTR-Protokoll angeschlossen werden, dann ist mit *p1*=3 der LIST-Kanal auszuwählen. Als 2. Parameter kann normalerweise jede Ziffer von 0 bis 3 gewählt werden, aber Kanal 2 (LPT:) ist für Drucker allgemein üblich.

```
DTR 3 2
```

Der 2. Parameter erhält dann eine große Bedeutung, wenn mehrere Drucker-treiberprogramme gleichzeitig zugewiesen werden sollen und die Umschaltung softwaremäßig über das I/O-Byte erfolgt. (CP/M)

Beispiel 2:

Bei Kopplung mit einem zweiten Rechner oder dem EPROM -Programmiergerät werden zur Datenübertragung beide Richtungen (senden und empfangen) benötigt. Dafür eignen sich die beiden log. Geräte READER (empfangen) und PUNCH (senden), weil damit der Druckerkanal nicht beeinflußt wird. Das Treiberprogramm muß für beide Gerate getrennt initialisiert werden.

```
DTR 1 1     für RDR:=PTR:
DTR 2 1     für PUN:=PTP:
```

Ablauf der Parametereingabe

Nach der Eingabe des Kommandonamens DTR oder XON/X OFF und der zwei Parameter *p1* und *p2* kann die Taste ENTER gedrückt oder weitere Parameter geschrieben werden. Für jeden weiteren Parameter entfällt eine der folgenden Eingabeaufforderungen unter der Bedingung, daß der Parameter an dieser Stelle zugelassen ist. Werden alle 5 weiteren Parameter (also insgesamt 7) hingeschrieben und sind sie sinnvoll, dann wird die Initialisierung ohne weitere Eingabeaufforderungen durchgeführt und das Kommando beendet.

Die Kommandozeile konnte z.B. so aussehen:

```
DTR 3 2 A8 1 B0 2 8
```

Einfacher ist es, die letzten 5 Parameter wegzulassen. Das Programm erfragt dann diese Angaben. Der Wert in den eckigen Klammern kann mit ENTER übernommen oder ein neuer Wert eingegeben und mit ENTER bestätigt werden. Falls der Parameter einen unzulässigen Wert hat, wird die Eingabeaufforderung wiederholt.

```
CTC-Adresse        [A8]:
```

Die Adresse A8 ist für das Druckermodul zu verwenden.

```
Zeitkonstante      [01]:
```

Mit der Zeitkonstante wird die Datenübertragungsgeschwindigkeit (Baudrate) festgelegt. Folgende Werte sind üblich:

```
01 = 9600 Baud      10 = 600 Baud
02 - 4800 Baud      20 = 300 Baud
04 = 2400 Baud      40 = 150 Baud
08 = 1200 Baud      80 = 75 Baud
```

```
SIO-Adresse Daten  [B0]:
```

Die Adresse B0 ist für das Druckermodul zu verwenden. Die Adresse für die Steuerung ergibt sich aus der angegebenen Adresse plus 2.

```
PARITY 0:NO/1:ODD/2:EVEN [02]:
```

Hier sind nur die Ziffern 0, 1 oder 2 für die Paritätsauswahl zulässig.

```
BIT/ZEICHEN        [08]:
```

Hier sind nur die Ziffern 7 und 8 zulässig.

Nach der Eingabe dieser Parameter oder der Bestätigung mit ENTER wird das serielle Treiberprogramm initialisiert. Zwischendurch ist mit der STOP-Taste jederzeit ein Abbruch möglich.

Ablauf der Initialisierung

- CTC und SIO initialisieren, Merkzellen im RAM vorbereiten.
- Eintragung der Treiberadresse in die Adreßtabelle der Softwarekomponente UP. Der Aufruf erfolgt grundsätzlich über UP-Rufe.
- Die betreffenden 2 Bits im I/O-Byte modifizieren.

Hinweis: Als Arbeitsspeicher für die 16 möglichen Treiber stehen im Adreßbereich 0280H bis 02BFH für jeden Treiber 4 Byte zur Verfügung. Die jeweils gültige Adresse wird intern ermittelt. Die Verwendung dieses Bereiches für Treiberprogramme wurde bereits im Heft "Betriebssystem Z 9001 " vorgeschlagen. Im CP/M-Modus wird dieser Bereich verlagert.

Initialisierung für mehrere Schnittstellen gleichzeitig

Sind mehrere Module mit seriellen Schnittstellen (SIO) gleichzeitig angeschlossen oder werden auf einen Modul beide SIO-Kanäle benutzt, dann sind bei den Initialisierungskommandos verschiedene Kanäle zu wählen und die jeweiligen unterschiedlichen Portadressen für CTC und SIO einzugeben. Die Treiberprogramme beeinflussen sich in diesem Fall nicht gegenseitig, weil für jeden der 16 möglichen Kanäle ein eigener Arbeitsspeicher zur Verfügung steht, wo auch die Portadressen abgelegt werden.

Automatische Kaltstartinitialisierung

Es besteht die Möglichkeit, während der Systeminitialisierung nach Kaltstart bereits die serielle Schnittstelle automatisch zu aktivieren. Der Nutzer braucht dann die soeben erklärten Initialisierungskommandos nicht einzugeben, ist aber an die Standardparameter gebunden.
Wie die Kaltstartinitialisierung funktioniert, ist bei der Softwarekomponente BO (Systemanlauf) erklärt. Die SIO-Initialisierung wird um Namen '$SIO ' erkannt. Standardmäßig ist dieser Kommandoname vorhanden und bewirkt bei jedem Kaltstart die Abarbeitung der folgenden Schritte. Auf besonderen Wunsch kann dieses Kommando auch entfernt bzw. geändert werden.

- Initialisierung CTC und SIO.
- Test, ob Initialisierung erfolgreich war, d.h. ob Druckermodul gesteckt ist.
- Wenn Modul nicht gesteckt, dann Rückkehr mit Fehlercode und Anzeige 'nicht bereit' durch die Softwarekomponente BO.
- Wenn Modul gesteckt, dann Arbeitsspeicher für READER und PUNCH initialisieren.
- Treiberadressen an Softwarekomponente UP übergeben.
- I/O-Byte modifizieren.
- Die Zuweisung erfolgt zu den Kanälen RDR:=PTR: und PUN:=PTP:
- Der Druckerkanal LST: wird nicht zugewiesen.
- DTR-Protokoll
- Portodressen entsprechend Druckermodul: CTC=A8 und SIO=B0/B2.
- 9600 Baud
- gerade Parität
- 8 Datenbits

Falls diese Daten unerwünscht sind, kann nachträglich mit Hilfe der beschriebenen Kommandos ein anderer Wert eingestellt werden.

Fehlerbehandlung

Bei seriellen Datenübertragungen können sehr vielfältige Fehler auftreten, die von diesem Treiberprogramm erkannt und so konkret wie möglich angezeigt werden. In der Praxis treten diese Fehler gehäuft auf, wenn die Parameter der verbundenen Geräte nicht übereinstimmen (PARITY, FRAME). Es sind dann so viele Versuche mit veränderten Parametern (z.B. DIL-Schalter am Drucker) durchzuführen, bis keine Fehler mehr auftreten.
Nach einer Fehlermeldung kann man sich durch Tastendruck entscheiden, ob fortgesetzt oder abgebrochen werden soll.

Fehlermeldung mögliche Ursache

TxC-ERROR	Sendetakt fehlt, CTC nicht initialisiert oder Hardwarefehler
RxD-BREAK	Unterbrechung der Verbindungsleitung z.B. Wackelkontakt
PARITY	Paritätsfehler, Parameter falsch gewählt
FRAME	Stopbit falsch erkannt, Parameter falsch gewählt
OVERFLOW	Empfängerüberlauf, Rückmeldung zum Sender funktioniert nicht

Das Treiberprogramm kann noch weitere Mitteilungen und Aufforderungen auf den Bildschirm schreiben, die aber ohne weitere Erläuterungen verständlich sind.

Abbruchmöglichkeiten

Immer wenn sich der Rechner im Treiberprogramm befindet, wird die grüne Leuchtdiode "GRAPHIC" auf der Tastatur eingeschaltet und bei Verlassen wieder ausgeschaltet. Der Bediener kann also die Arbeit des Treibers durch "Flackern" oder eine Verklemmung durch ständiges Leuchten optisch erkennen. Verklemmungen treten besonders dann auf, wenn gar kein Gerät an die serielle Schnittstelle angeschlossen ist. Bei anderen Treiberprogrammen hilft in diesem Fall oft nur die RESET-Taste weiter. Hier ist ein Abbruch jederzeit (nicht nur bei Verklemmungen) möglich, indem die leuchtende GRAPHIC-LED mit der Taste GRAPHIC ausgeschaltet wird.
Der Treiber meldet den Abbruch durch die Ausschrift

```
BREAK BYGRAPHIC-KEY
```

09.03.91 Lutz Elßner, Postfach 127-14, 0-8210 Freital

Softwarekomponente 11 EPROG - EPROM-Programmierung

PROGRAMMBESCHREIBUNG

Zweckbestimmung, benötigte Hardware

Die Softwarekomponente dient zur Bedienung des EPROM-Programmiergerätes EPROG 27011, welches vom Elektronikversandhaus CONRAD, Klaus-Conrad-Straße 1, W-8452 Hirschau, Telefon (09622) 30111 bezogen werden kann. Die Daten und Bestellnummern werden in dieser Beschreibung genannt.
Die Voraussetzung zum Betrieb des Programmiergerätes ist das Vorhandensein eines Druckermoduls und der Softwarekomponente SIO. Das Druckermodul muß entsprechend der Beschreibung SIO umgebaut sein und das serielle Treiberprogramm DTR muß den log. Geräten READER (für Eingabe) und PUNCH (für Ausgabe) zugewiesen sein (z.B. durch Kaltstartinitialisierung).
Zur Ausführung der wichtigsten Funktionen des Programmiergerätes stehen eigenständige Kommandos zur Verfügung. Die Kommandos sind mit zahlreichen Parametern einzugeben, es erfolgt keine Menuführung.

EPROG 27011 - Technische Daten

Bedien- und Anzeigeelemente:
2 Tasten RESET und START
2 Wahlschalter EPROM-TYP und MODE
2 Leuchtdioden WORKING und PROGRAMMING
2 28 polige Fassungen MASTER und SLAVE

programmierbare EPROMs:
2516, 2716, 2532, 2732, 2732A, 2764, 2764A, 27128, 27128A, 27256, 27512, 27513, 27011

4 Programmieralgorithmen:
STANDARD, FAST INTEL, FAST AMD, FAST FUJITSU

Betriebsspannung: 12V 900 mA

intern erzeugte Programmierspannungen: 12.5V, 21V, 25V

Interface: V24, DTR-Protokoll, Parameter durch DIL-Schalter einstellbar

Programmiergerät enthält eigenen Mikrocontroller, deshalb ist Leertest, Vergleichen und Duplizieren der EPROMs in beiden Fassungen ohne angekoppelten Rechner möglich.

EPROG 27011 - Lieferumfang, Bestellangaben

Für den Betrieb am Z 9001, KC 85/1, KC 87 entsprechend dieser Beschreibung genügt die Leiterplatte und eine geeignete 12V Stromversorgung.

1. Programmiergerät EPROG 27011
   Best.-Nr.: 97 58 00-22 Preis: 198.-- DM

Dieses Gerät besteht nur aus einer bestückten Leiterplatte und einer sehr guten Beschreibung. Anhand dieser Beschreibung ist es gelungen, dem Gerät alle notwendigen Funktionen zu entlocken. Darin ist auch ein BASIC-Programm zur Bedienung abgedruckt. Für den Anschlug der Betriebsspannung und V24 - Interface stehen 7 Steckkontakte zur Verfügung. Alle Bauelemente sind auf der Leiterplatte beschriftet.

2. Passendes Gehäuse zu EPROG 27011
   Best.-Nr.: 97 58 69-22 Preis: 149.-- DM

Dieser Bausatz enthält ein komplettes Gehäuse mit fertig bearbeiteter und beschrifteter Frontplatte und Rückwand sowie allen Knöpfen, Steck-verbindern, Schrauben und Schwenkhebelfassungen. Beim Zusammenbau hat man

allerdings noch die Durchbrüche für Steckverbinder in der Plast-Rückwand anzufertigen. Die Schwenkhebelfassungen sollten vor dem Einstecken in die Frontplatte gründlich verzinnt werden, denn beim Anlöten der Zwischensteckverbinder geht es sehr eng zu. Zuletzt sind noch die Verbindungsleitungen anzulöten und alles zusammenzuschrauben. Die richtige Anschlußbelegung der V24-Buchse ist im folgenden Abschnitt wiedergegeben.
folgendes weiteres Zubehör ist erhältlich (nur zur Information):

3. ATARIST Steuersoftware für EPROG 27011
   Best.-Nr.: 97 54 19-22 Preis: 29.90 DM

4. Professionelle PC-Steuersoftware für EPROG 27011
   Best.-Nr.: 97 54 27-22 Preis: 39.90 DM

5. Passendes Netzteil NG III (12V, 1A)
   Best.-Nr.: 97 61 30-22 Preis: 19.50 DM

6. Passender Anschlußkabel-Bausatz (für 12V-Stromversorgung)
   Best.-Nr.: 97 54 00-22 Preis: 5.90 DM

Alle Angaben sind im Hauptkatalog Electronic ’91 auf Seite 911 nachzulesen. Der Katalog kann auch bestellt werden.

7. Hauptkatalog E91
   Best.-Nr.: 90 00 01-20 Preis: 5.-- DM + 3.50 DM Porto

Bei einer relativ umfangreichen Bestellung erhält man den Katalog kostenlos.

V24 Verbindungsplan

Anschließend ist die Verbindung der V24-Signale vom Druckermodul zum Programmiergerät wiedergegeben. Die Lötösen X3-X7 sind auf dem Druckermodul so angeordnet wie sie hier gezeichnet sind. Weitere Informationen sind in der Beschreibung der Softwarekomponente SIO nachzulesen.
In der Mitte ist die Belegung eines 25poligen D-SUB-Steckverbinders angegeben. Die 25polige Buchse sollte nach folgendem Schema und nicht nach dem auf dem EPROG-Gehäuse aufgedruckten Plan belegt werden. Damit wird erreicht, daß der Stecker am Druckermodul auch an einen Drucker angeschlossen werden kann.

```
Druckermodul  Stecker D-SUB Buchse  EPROG 27011

X4       CTS (-------- 20 --------- RTS (4)
  X6     RxD (--------  2 --------- TxD (3)
    X7   DTR ---------  5 --------) CTS (2)
      X5 TxD ---------  3 --------) RxD (1)
     X3  SG  ---------  7 --------- GND (5)
```

Einstellung der DIL-Schalter

Wenn die Softwarekomponente SIO beim Kaltstart initialisiert wird, müssen die DIL-Schalter auf der EPROG-Leiterplatte wie folgt eingestellt werden, um einen fehlerfreien Betrieb der V24-Schnittstelle zu gewährleisten:

```
S1 - S3   off   9600 Baud
S4 off          1 Stopbit
S5 on           8 Datenbits
56 on S7  off   Gerade Parität (in EPROG-Beschreibung falsch angegeben)
S8              siehe EPROG-Beschreibung
```

Kommandos

Für die Bedienung des Programmiergerätes stehen die folgenden Kommandos zur Verfügung. In allen Fällen muß sich der angesprochene EPROM in der "SLAVE"-Fassung befinden und der richtige Typ am Wahlschalter eingestellt sein! Der

EPROM-Typ wird grundsätzlich vom Wahlschalter entnommen und nicht softwaremäßig verändert. Der andere Wahlschalter (MODE) kann beliebig stehen, es wird aber Linksanschlag empfohlen. Die folgenden Kommandos sind nicht für die Behandlung der "gebankten" EPROMs 27513 und 27011 ausgelegt.

Wichtiger Hinweis:

> Beim Einschalten der 12V Betriebsspannung für das Programmiergerät darf sich kein EPROM in der Fassung befinden! Es ist möglich, daß ein undefinierter Programmierimpuls erzeugt wird.

Systemschnittstelle für alle Kommandos:

Zur Bedienung des Programmiergerätes benutzen alle Kommandos ausschließlich die systeminterne READER- und PUNCH-Schnittstelle der Softwarekomponente UP. Für die V24-Schnittstelle ist allein die Softwarekomponente SIO zuständig, die auch initialisiert sein muß.

Kommando EL

Syntax:

EL *aepr eepr*

Funktion:

Leertest eines EPROM-Bereiches. Der Adreßbereich kann angegeben werden. Die Anfangs- und Endadresse müssen sich im Adreßbereich des am Wahlschalter eingestellten EPROM-Typs befinden. Beim Weglassen der Parameter wird der gesamte EPROM getestet. Als Ergebnis werden die Codierungen der drei Antwortbytes angezeigt:

```
30 0D 0A                          bedeutet Bereich ist leer
31 00 0A EPROM ist nicht leer!    dieser Text wird angezeigt
```

Bei diesem Kommando wird im Rechner kein RAM benutzt.

Kommando EK

Syntax:

EK

Funktion:

Ermittlung der Prüfsumme des gesamten EPROMs durch einfache Addition aller Bytes. Diese Funktion wird vom Programmiergerät selbständig ausgeführt, nur das Ergebnis wird dem Rechner zurückgeliefert.
Als Ergebnis wird eine 8 stellige Hexadezimalzahl angezeigt. Die letzten 4 Stellen (16 Bit) entsprechen der Prüfsumme, die mit dem Kommando CS (der Softwarekomponente KO) für einen Speicherbereich ermittelt werden kann. So ist eine einfache, aber nicht eindeutige Vergleichsmöglichkeit zwischen EPROM und Speicher gegeben. Bei diesem Kommando wird im Rechner kein RAM benutzt.

Kommando ED

Syntax:

ED *aepr eepr*

Funktion:

Anzeigen eines EPROM-Bereiches (DISPLAY). Der Adreßbereich kann angegeben werden. Die Anfangs- und Endadresse müssen sich im Adreßbereich des am

Wahlschalter eingestellten EPROM-Typs befinden. Bei Weglassen der Parameter wird der gesamte EPROM am Bildschirm angezeigt.
Auf dem Bildschirm werden in jeder Zeile die Adresse relativ zum EPROM - Anfang und anschließend 8 Bytes hexadezimal angezeigt.
Die Anzeige kann mit PAUSE angehalten, mit CONT fortgesetzt und mit STOP abgebrochen werden. Nach dem Abbruch mit STOP holt der Rechner noch die restlichen Daten vom Programmiergerät, um einen definierten Zustand herzustellen. Wenn das zu lange dauert, kann um Rechner die GRAPHIC-Taste zum gewaltsamen Abbruch (des V24-Treibers) benutzt werden. Anschließend ist auch das Programmiergerät mit der RESET-Taste in Grundstellung zu bringen.
Bei diesem Kommando wird im Rechner kein RAM benutzt.

Kommando ER

Syntax:

ER *zadr* [*bank*]
ER *aepr eepr zadr* [*bank*]

Funktion:

Einlesen eines EPROM-Bereiches in den Speicher des Rechners. Zur Kommandoeingabe gibt es zwei verschiedene Schreibweisen.

1. ER mit 1 oder 2 Parametern
Es wird grundsätzlich der gesamte EPROM in den Speicher ob Adresse *zadr* (Zieladresse) eingelesen. Der Parameter *bank* ist nur erforderlich, wenn die Zieladresse in einer Bank des 192 KByte RAM/EPROM-Moduls liegt.

2. ER mit 3 oder 4 Parametern
Hier muß der Adreßbereich des EPROMs mit Anfangs- und Endadresse hingeschrieben werden. Es ist darauf zu achten, daß sich diese Adressen innerhalb des EPROMs befinden (Wahlschalter). Für die Parameter *zadr* und *bank* gilt das unter 1. gesagte. Auf *zadr* wird das Byte von *aepr* übertragen usw.
Mit diesem Kommando ist das Lesen jedes beliebigen Adreßbereiches aus dem EPROM möglich.

Beispiel:

Einlesen der "zweiten Hälfte" eines 8-KByte-EPROMs 2764 in den Speicher ab Adresse 3000H.

ER 1000 1FFF 3000

Der gesamte EPROM hat einen Adreßbereich von 0000H bis 1FFFH. Der zu lesende Bereich beginnt bei Adresse 1000H und endet bei der EPROM-Endadresse 1FFFH. Diese 4 KByte werden im Speicher von 3000H - 3FFFH abgelegt.
Die Angabe der Bank kann entfallen, weil es in diesem Adreßbereich keine Bänke gibt.

Kommando EP

Syntax:

EP *aadr eadr* [*bank* [*aepr* [*alg*]]]

Funktion:

Programmieren eines EPROM-Bereiches mit Daten aus dem Speicher des Rechners. *aadr*, *eadr* und *bank* geben den Adreßbereich und die Bank der Datenquelle im Rechner an. Somit können alle Speicherbereiche, die im Rechner überhaupt adressierbar sind auch auf EPROM übertragen werden. Der Parameter *bank* hat nur für Speicherbereiche aus dem 192 KByte RAM/EPROM-Modul Bedeutung.
aepr ist die Anfangsadresse im EPROM, wohin das erste Byte (von *aadr*) programmiert wird. Fehlt *aepr*, dann wird 0000H angenommen. Jeder EPROM

beginnt (wenn er im Programmiergerät steckt) mit der Adresse 0000H. Die höchste Adresse ist von der Speicherkapazität des EPROMs abhängig. Es ist darauf zu achten, daß nicht nur *aepr* im Adreßbereich des EPROMs liegt, sondern auch die zu programmierenden Daten noch Platz haben.

Der letzte Parameter *alg* sollte aus Sicherheitsgründen nicht hingeschrieben werden. Wenn *alg* weggelassen wurde, erscheint zur Kontrolle noch einmal die Anzeige von Anfangs- und Endadresse sowie der Bank des Speicherbereiches sowie die Anfangs- und Endadresse des zu programmierenden EPROM-Bereiches. Hier ist eine letzte Kontrolle vor der Programmierung zu empfehlen. Außerdem kann man noch einmal nachsehen, ob der richtige EPROM in der Fassung steckt und der Typ am Wahlschalter eingestellt wurde. Bei negativem Kontrollergebnis ist jetzt noch ein Abbruch mit STOP möglich.

Auf dem Bildschirm werden außer den Adressen auch die 4 möglichen Programmieralgorithmen angezeigt. Durch Eingabe einer Ziffer kann man sich entscheiden und die Programmierung starten. Wenn die Entscheidung zu schwer fällt, kann auch nur die Taste ENTER gedrückt werden. Der so ausgewählte Algorithmus FAST INTEL hat sich allgemein bewährt.

Die Anzeige von Adressen und Programmieralgorithmen kann unterdrückt werden, wenn die Kennzahl für den Algorithmus mit in die Kommandozeile geschrieben wird.

Ablauf der Programmierung

Der genaue Ablauf ist in der EPROG-Beschreibung wiedergegeben. Nachdem ein Byte programmiert wurde, wird es zu Kontrolle gelesen und verglichen. Der Programmieralgorithmus entscheidet, wie oft es noch nachprogrammiert werden kann. Wird beim Vergleich ein Fehler festgestellt, meldet das Programmier-gerät diesen sofort an den Rechner und beendet die Programmierung. Am Rechner wird der Fehlercode angezeigt:

```
31 0D 0A Programmierfehler!
```

Die Fehlerstelle kann nachträglich z.B. mit dem Kommando ED ermittelt werden. Die Fehlerursache kann ein defekter oder ein nicht gelöschter EPROM sein. Um solche bösen Überraschungen zu vermeiden, sollte der zu programmierende EPROM-Bereich vorher mit dem Kommando EL (Leertest) überprüft werden. Eine weitere Ursache kann die Wahl einer zu niedrigen Programmierspannung (am Wahlschalter) sein. Im Zweifelsfalle sollte der erste Versuch immer mit der niedrigeren Spannung durchgeführt werden (Typ mit A) und bei einer Fehlermeldung erst danach umgeschaltet werden.

EPROMs im 192 KByte RAM/EPROM-Modul

Die in diesem Modul eingesetzten EPROMs werden durch besondere Hardware in einem begrenzten Speicherbereich mehrfach parallel abgebildet. Dadurch wird die Zuordnung der EPROM-Adreßbereiche zu den Speicher-Adreßbereichen und Banken unübersichtlich. Die folgenden Tabellen sollen helfen, den richtigen Adreßbereich im EPROM zu programmieren, damit dieser anschließend im Modul an der richtigen Stelle steht. Tabelle 1 gilt für die EPROMs in den Fassungen D2 und D3, die grundsätzlich im Adreßbereich C000H-DFFFH (8K) wiedergefunden werden. Tabelle 2 gilt für D5 im Adreßbereich E000H-E7FFH (2K).
Links ist der EPROM-Typ mit dem jeweils zusammenhängenden Adreßbereich angegeben. In den rechten Spalten steht jeweils die Bank, wo dieser Bereich im Modul zu finden ist. Das x bedeutet, daß diese Tetrade (4 Bit) der Bankkennzahl für den betrachteten Adreßbereich keine Bedeutung hat. Ein Strich bedeutet, daß dieser Adreßbereich des EPROMs im Modul nicht adressierbar ist. EPROMs mit geringen Speicherkapazitäten, die nicht alle Bänke ausfüllen, sind teilweise in den unbelegten Bänken zusätzlich adressierbar. Diese Erscheinung ist in den Tabellen nicht berücksichtigt.

Tabelle 1

Typ	EPROM-Bereich	D2 (0000-DFFF) Bank	D3 (0000-DFFF) Bank
2764	0000-1FFF	7x	Bx
27128	0000-1FFF	6x	Ax
	2000-3FFF	7x	Bx
27256	0000-1FFF	4x	8x
	2000-3FFF	5x	8x
	4000-5FFF	6x	Ax
	6000-7FFF	7x	Bx
27512	0000-1FFF	0x	--
	2000-3FFF	1x	--
	4000-5FFF	2x	--
	6000-7FFF	3x	--
	8000-9FFF	4x	8x
	A000-BFFF	5x	9x
	C000-DFFF	6x	Ax
	E000-FFFF	7x	Bx

Tabelle 2

Typ	EPROM-Bereich	D5 (E000-E7FF) Bank
2764	0000-07FF	x4
	0800-0FFF	x5
	1000-17FF	x6
	1800-1FFF	x7
27128	0000-07FF	x0
	0800-0FFF	x1
	1000-17FF	x2
	1800-IFFF	x3
	2000-27FF	x4
	2800-2FFF	x5
	3000-37FF	x6
	3800-3FFF	x7
27256	0000-3FFF	nicht adressierbar
	4000-47FF	x0
	4800-4FFF	x1
	5000-57FF	x2
	5800-5FFF	x3
	6000-67FF	x4
	6800-6FFF	x5
	7000-77FF	x6
	7800-7FFF	x7

Beispiel:

Im 192 KByte RAM/EPROM-Modul soll der Adreßbereich C000H-DFFFH (8K) in Bank 6 programmiert werden. Dazu ist der EPROM aus Fassung D2 in das Programmiergerät zu stecken. In Tabelle 1 ist neben dem richtigen EPROM-Typ die gewünschte Bank 6 aufzusuchen und links der Adreßbereich für die Programmierung abzulesen. Dieser Bereich hat wie im Modul die Länge 8 KByte (2000H). Für die Programmierung von Teilbereichen können die entsprechenden Adressen jetzt leicht berechnet werden.

08.03.91 Lutz Elßner, Postfach 127-14, D-8210 Freital 50

Alle Kommandos auf einen Blick Seite

Kommandos für Magnetbandkassette

Kommandos für physische Diskettenarbeit Seite

Kommandos für serielle Treiber

Kommandos für EPROG 27011