Wolfgang Kanis

Der Z80-EMUF

Preiswerter Einplatinen-Computer

Der mc-6504-EMUF fand eine weite Verbreitung, weil er erstens sehr preiswert ist und sich zweitens Programme für ihn mit preiswerten Tischcomputern entwickeln lassen. Beides gilt auch für den hier vorgestellten Z80-EMUF; er läßt sich mit einer Taktfrequenz bis zu 4 MHz betreiben, und die Platine kann mit 2 KByte RAM, 8 KByte EPROM und I/O-Ports mit 32 Leitungen bestückt werden.

Der Z80-EMUF (Einplatinen-Mikrocomputer für universelle Festprogramm-Anwendung) soll es allen Besitzern von Computern wie TRS-80, Video-Genie, Nascom usw. ermöglichen, ihre spezielle Computerlösung für eine bestimmte Aufgabe zu finden. Wie sein Verwandter, der 6504-EMUF [1], ist der Z80-EMUF nicht dazu gedacht, auf ihm Programme zu entwickeln (wenn das mit einem geeigneten Monitorprogramm auch prinzipiell möglich wäre), ihn groß auszubauen oder Basic-Programme laufen zu lassen. Vielmehr ist er als eine Art softwaregesteuerte Logikschaltung zu betrachten.

Maximales Z80-Minimalsystem

Diese Einschränkungen ermöglichten es, ein sehr preiswertes Minimalsystem für weniger als 100 DM zu konzipieren. Der Z80-EMUF besteht aus CPU, RAM, EPROM, PIO, Oszillator, Reset und Adressendecoder. Die genaue Verschaltung der Bauelemente ist aus *Bild 1* ersichtlich. Das Schaltbild ist für den Betrieb mit 2-KByte-Speicherbausteinen gezeichnet. Die Verwendung von 4- oder 8-KByte-Speichern ist auf der Platine schon vorgesehen. Dabei müssen die Lötbrücken entsprechend Schaltbild und Bestückungsplan (*Bild 2*) eingelötet werden. *Bild 3* und *Bild 4* zeigen das Layout der doppelseitigen Platine.

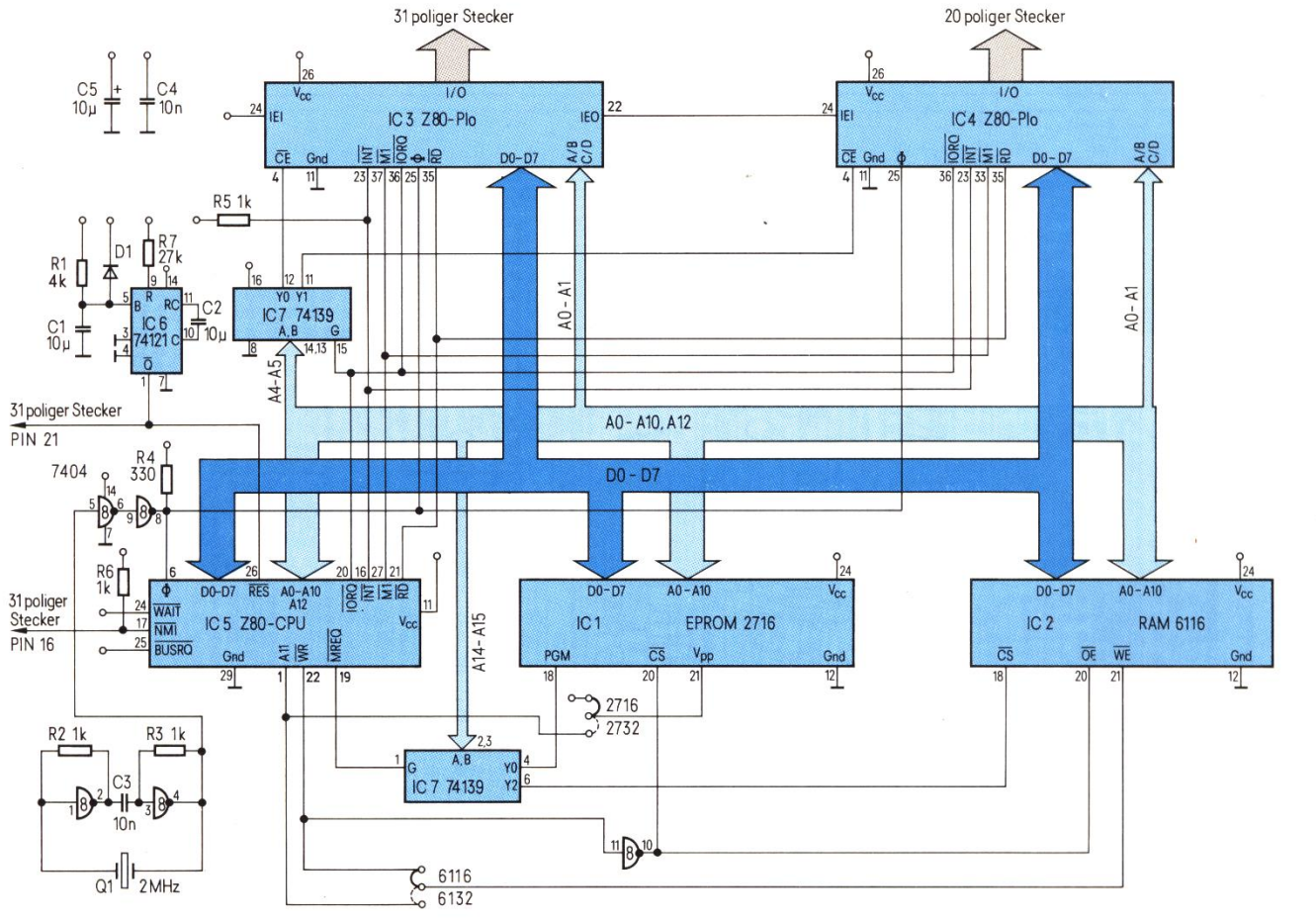


Bild 1. Gesamtschaltbild des Z80-EMUF. Er kann wahlweise mit ein oder zwei PIO-Bausteinen sowie mit RAM- und EPROM-Bausteinen unterschiedlicher Kapazität bestückt werden

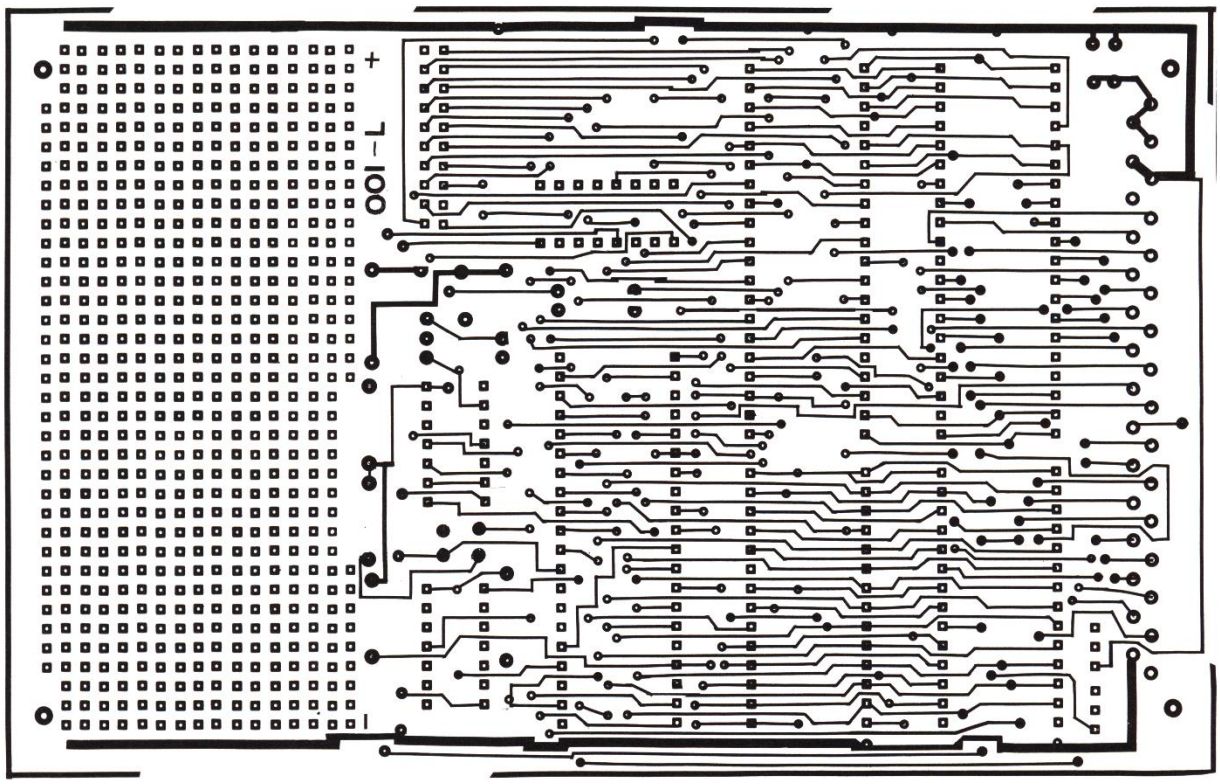


Bild 3. Lötseite des Platinenlayouts

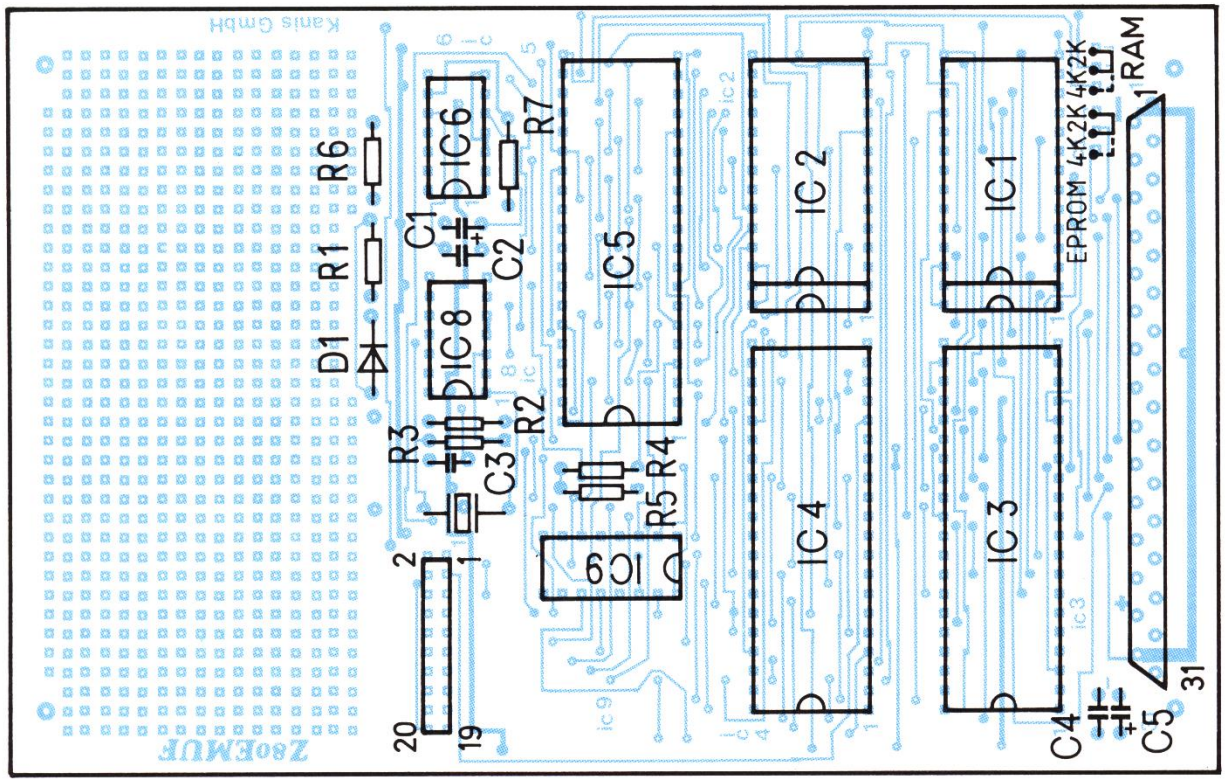


Bild 2. Bestückungsplan. Je nach verwendeter RAM- und EPROM-Kapazität sind Brücken erforderlich

```
0000                    ORG 0000H
                        *Testprogramm fuer Z80 EMUF
                        *auf den Ausgaengen der PIO's
                        *wird nacheinander gezaehlt
8000                    RAM EQU 8000H
0000                    PIO0 EQU 0
0010                    PIO1 EQU 10H
0000 3E CF              EMUT1 LD A,11001111B *Initiali-
                        *sierung aller Ports
0002 D3 02              OUT (PIO0+2),A *auf Einzelbit Ein-
                        *Ausgabe
0004 3E 00              LD A,0 *Maske alle Bits Ausgaenge
0006 D3 02              OUT (PIO0+2),A
0008 3E CF              LD A,11001111B
000A D3 03              OUT (PIO0+3),A

000C     3E 00                LD A,0
000E     D3 03                OUT (PIO0+3),A
0010     3E CF                LD A,11001111B
0012     D3 12                OUT (PIO1+2),A
0014     3E 00                LD A,0
0016     D3 12                OUT (PIO1+2),A
0018     3E CF                LD A,11001111B
001A     D3 13                OUT (PIO1+3),A
001C     3E 00                LD A,0
001E     D3 13                OUT (PIO1+3),A
0020     06 00         LOOP   LD B,0
0022     78            L1     LD A,B
0023     32 00 80             LD (RAM),A *RAM WIRD ANGESPROCHEN
0026     3A 00 80             LD A,(RAM)
0029     D3 00                OUT (PIO0+0),A
002B     05                   DEC B
002C     C2 22 00             JP NZ,L1
002F     06 00                LD B,0
0031     78            L2     LD A,B
0032     32 00 80             LD (RAM),A
0035     3A 00 80             LD A,(RAM)
0038     D3 01                OUT (PIO0+1),A
003A     05                   DEC B
003B     C2 31 00             JP NZ,L2
003E     06 00                LD B,0
0040     78            L3     LD A,B
0041     32 00 80             LD (RAM),A
0044     3A 00 80             LD A,(RAM)
0047     D3 10                OUT (PIO1+0),A

0049     05                   DEC B
004A     C2 40 00             JP NZ,L3
004D     06 00                LD B,0
004F     78            L4     LD A,B
0050     32 00 80             LD (RAM),A
0053     3A 00 80             LD A,(RAM)
0056     D3 11                OUT (PIO1+1),A
0058     05                   DEC B
0059     C2 4F 00             JP NZ,L4
005C     F2 20 00             JP LOOP
005F                          DS 1000H
105F     00                   END 2280H
```

Bild 5. Ein kleines Testprogramm für erste Versuche

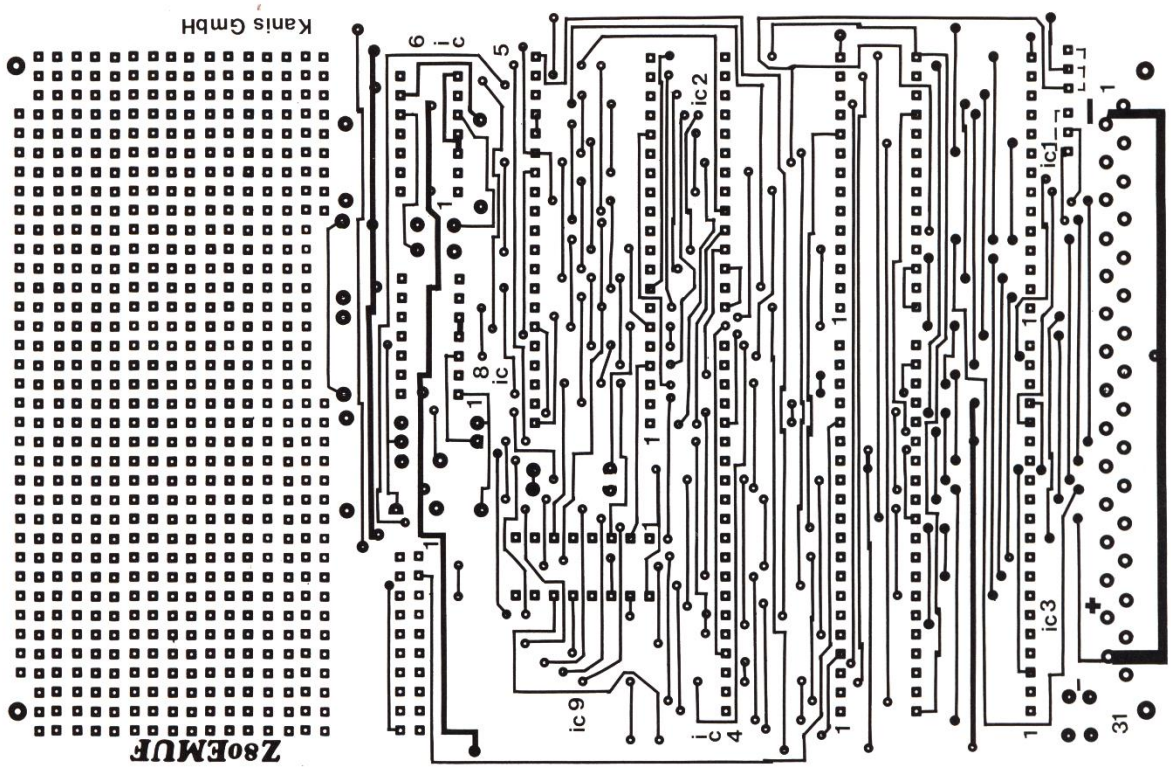


Bild 4. Die bestückungsseitigen Leiterbahnen der Platine

Das System ist über $\overline{NMI}$ und $\overline{INT}$ voll interruptfähig. PIO 0 (IC3) hat über die Daisy-Chain-Logik höchste Priorität. Wird nur eine PIO verwendet, sollte diese IC3 sein. Höhere Taktfrequenzen als 2 MHz sind bei Verwendung entsprechender Bauteile erzielbar [2, 3].

Adressenbelegung

Folgende Adressen werden durch die Adressendecoder-IC7 festgelegt:

EPROM (IC1)	ab	0000	H
RAM (IC2)	ab	8000	H
PIO0 (IC3)	A Data	00	H
	Control	02	H
	B Data	01	H
	Control	03	H
PIO1 (IC4)	A Data	10	H
	Control	12	H
	B Data	11	H
	Control	13	H

Für spezielle Anpassungen steht ein relativ großes Verdrahtungsfeld zur freien Verfügung. Auf diesem lassen sich z. B. A/D-, D/A-Wandler, Taktgenerator, Opto-Koppler o. ä. leicht aufbauen.
Um eine Verwendung der bisher erschienenen Peripherie möglichst einfach zu gestalten, wurde auf eine Kompatibilität am 31poligen Stecker mit dem 6504-EMUF geachtet. Wie aus *Tabelle 2* ersichtlich, sind alle Steckerpunkte gleich. Nur die freien Steckerpunkte wurden für zusätzliche Funktionen verwendet. Eine weitere 20polige Steckleiste dient als Anschluß für IC4 als zusätzliche PIO (*Tabelle 3*).

Die Inbetriebnahme

Um eventuelle Fehlfunktionen leichter ermitteln zu können, ist es zu empfehlen, die Platine mit Sockeln zu bestükken, mindestens für die LSI-Bauteile. Auf das richtige Einsetzen (siehe Bild 2) von IC1 und IC2 ist zu achten. Sie sollten in 28polige Sockel eingesetzt werden, dadurch ist eine spätere Erweiterung mit 4-KByte- und 8-KByte-Speicherbausteinen möglich. Vor der Inbetriebnahme ist die Platine zur Kontrolle optisch auf Lötbrücken und vergessene Lötungen abzusuchen. Nun steht einem Start nichts mehr im Wege. Das bereits programmierte EPROM einsetzen, +5 V anschließen, und schon sollte Ihr Programm abgearbeitet werden. Der Stromverbrauch beträgt in der Regel 0,3 A. Um einige Reserven zu haben, sollte ein Netzteil mit 5 V/0,5 A verwendet werden.
Sollte Ihr Programm nicht auf Anhieb laufen, empfiehlt es sich, erst den EMUF zu testen. Dies kann sehr gut mit einem kleinen Testprogramm (*Bild 5*) geschehen. Alle Ports, RAM und EPROM werden damit angesprochen. Zur Fehlersuche verwendet man am besten ein Oszilloskop. Alle Signale an EPROM, RAM, Adressendecoder und PIO werden damit untersucht. Fehlende Signale deuten auf Leiterbahnunterbrechungen, deformierte Pegel auf Kurzschlüsse hin. *Bild 6* zeigt den fertig aufgebauten Z80-EMUF.
Zum Schluß sei noch für die tatkräftige Unterstützung Herrn Rolf-Dieter Klein gedankt.

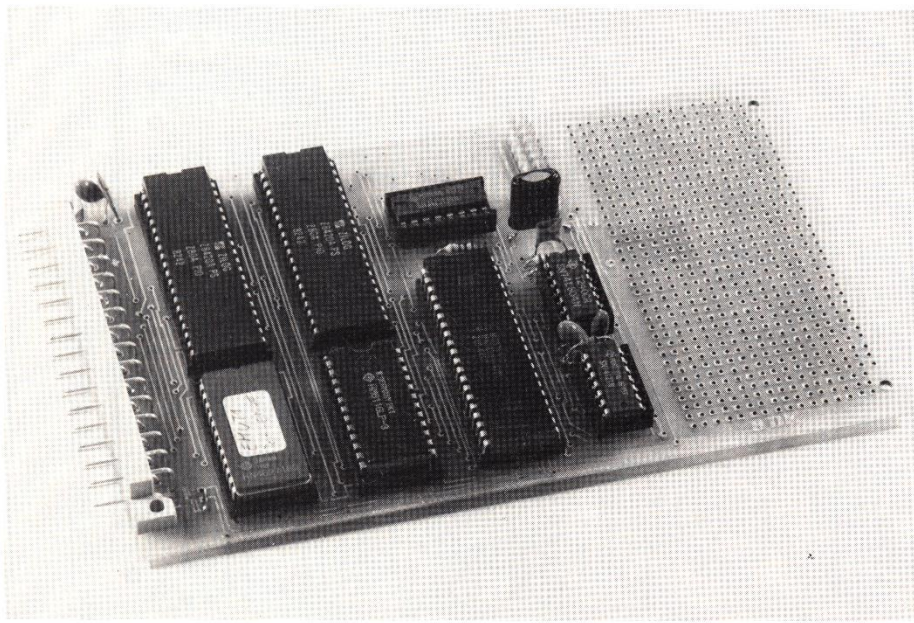


Bild 6. So sieht der Z80-EMUF fertig aus. Die große Lochrasterfläche bietet viel Platz für individuelle Erweiterungen

Platinen, Bausätze und Fertiggeräte sind beim Ing.-Büro W. Kanis, Lindenberg 113, 8134 Pöcking, erhältlich.

Literatur

[1] *Feichtinger, H.*: Mädchen für alles (6504-EMUF). mc 1981, Heft 2, und EMUF-Sonderheft, Franzis-Verlag.
[2] Zilog (Hrsg.): Z80-CPU Technical Manual.
[3] Zilog (Hrsg.): Z80-PIO Technical Manual.
[4] *Klein, M.*: Z80-Applikationsbuch, Franzis-Verlag.
[5] *Klein, R. D.*: Mikrocomputer-Hard- und Software-Praxis, Franzis-Verlag.

Tabelle 1: Stückliste

IC1 EPROM 2716 (oder entsprechende 4- bzw. 8-KByte-Speicher)
IC2 RAM 6116 (oder entsprechende 4- bzw. 8-KByte-Speicher)
IC3 PIO Z80
IC4 PIO Z80
IC5 CPU Z80
IC6 TTL 74121
IC7 TTL 74LS139
IC8 TTL 74LS04
R1 Widerstand 47 kΩ
R2 Widerstand 1 kΩ
R3 Widerstand 1 kΩ
R4 Widerstand 330 Ω
R5 Widerstand 1 kΩ
R6 Widerstand 1 kΩ
R7 Widerstand 27 kΩ
C1 Elko 1 µF
C2 Elko 10 µF
C3 Keramik-Kond. 10 nF
C4 Kermaik-Kond. 10 nF
C5 Elko 10 µF
Q1 Quarz 2 MHz (Standard)

Tabelle 2: Belegung des 31poligen Steckers (PIO – IC3)

1 – Masse	16 – $\overline{NMI}$
2 – Masse	17 – B1
3 – ARDY	18 – B2
4 – BRDY	19 – B3
5 – $\overline{ASTB}$	20 – $\overline{BSTB}$
6 – A0	21 – $\overline{RES}$
7 – A1	22 – B7
8 – A2	23 – B6
9 – A7	24 – B5
10 – A6	25 – B4
11 – A5	26 – NC
12 – A4	27 – +5 V
13 – A3	28 – +5 V
14 – Masse	29 – Masse
15 – B0	30 – Masse
	31 – Masse

Tabelle 3: Belegung des 20poligen Steckers (PIO – IC4)

1 – ARDY	11 – A3
2 – BRDY	12 – B3
3 – $\overline{ASTB}$	13 – A4
4 – $\overline{BSTB}$	14 – B4
5 – A0	15 – A5
6 – B0	16 – B5
7 – A1	17 – A6
8 – B1	18 – B6
9 – A2	19 – A7
10 – B2	20 – B7

Andreas Zilker, Stefan Wurdack

Z80-EMUF mit Komfort

Der Z80-EMUF ist ein kompletter Mikrocomputer und eigentlich zu schade, um nur zu Steuerungsaufgaben eingesetzt zu werden. Deshalb soll in diesem Artikel eine Möglichkeit vorgestellt werden, den EMUF mit einer einfachen Tastatur und einer Anzeige auszurüsten und ihn durch die entsprechende Software zum Leben zu erwecken. So ausgerüstet eignet er sich besonders für Entwicklungs- und Meßaufgaben oder einfach als Lehrsystem für diejenigen, die ein erstes Mal ernsthaften Kontakt zu einem Mikrocomputer suchen.

Um die CPU nicht mit der Kontrolle des Displays und der Tastatur zu belasten, wurden zwei „intelligente" Peripherietreiber verwendet. Der 7218 von Intersil steuert das 8stellige LED-Display (Typ 7218 A mit gem. Anode, B mit gemeinsamer Katode) und der 74C923 von National Semiconductor die Tastatur. Beide Bausteine werden über PIO 2 von der CPU mit Daten (PIO2A) und Steuersignalen (PIO2B) versorgt. Ein direkter Anschluß an den CPU-Bus, der prinzipiell möglich wäre, wurde nicht vorgenommen, damit die Peripherie z. B. auch für den 6504-EMUF ohne Hardwareänderungen verwendbar ist.

Der Aufbau der Hardware

Der Aufbau auf eine Einfach-Europakarte in Fädeltechnik ist problemlos möglich (*Bild 1*), wobei nicht mit Entkopplungskondensatoren gespart werden sollte (Multiplex-Störungen des Anzeigentreibers). Beide ICs, speziell der Anzeigentreiber, sind angesichts der gepflegten Preise mit großer Vorsicht zu behandeln. Für die Tastatur wurden Digitast-Mini-Keys verwendet; der 74C923 verdaut sicher auch einfachere Tasten. Außerdem ist noch Platz auf der Platine für ein Kassettenrecorder-Interface [1] und eine einfache V.24-Schnittstelle (*Bild 2*).
Der Anschluß an die CPU-Karte erfolgt mit einem 26poligen Flachbandkabel mit Pfostenstecker. Dazu wurde der „intelligenterweise" nur 20polige PIO-Anschluß der EMUF-Platine ins Lochrasterfeld herübergezogen und auf 26 Pole erweitert (zusätzlich Versorgungsspannung, RESET und NMI von der CPU). Auf keinen Fall darf man vergessen, die Handshake-Leitung STRB des A-Ports auf GND zu legen. Der EMUF selbst muß mit 4 MHz laufen, sonst stimmen die Zeitkonstanten für die serielle Schnittstelle nicht.

Schon recht komfortable Software

Das Monitor-Programm in *Bild 3* versorgt Tastatur, Anzeigen und Kassettenrecorder. Außerdem wurden einige Utilities wie z. B. MOVE und Displacement-Berechnung angefügt. Als besonderes „Zuckerl" gibt es noch einen Downloader für Host-Rechner-Kopplung und eine DCF-77-Decodierung für angeschlossenen Empfänger. Der Monitor wurde so modular wie möglich angelegt, wobei alles außer der kurzen Monitor-Hauptschleife als Unterprogramm verwendbar ist. In den Unterprogrammen werden meist nur die parametertragenden Register verändert, die zweite Registerbank der CPU bleibt frei für den Anwender. Stack und Monitor-RAM belegen die Adressen 8000H-80FFH, ab 8100H aufwärts (mögliche RAM-Erweiterung) ist Platz für „Selbstgestricktes".
Die Interrupts NMI und IRQ werden vom Monitor über RAM-Vektoren versorgt. Bei RESET wird in beiden Bereichen (NMIVE Adr. 8000H IRQVE Adr. 8003H) ein kompletter Jump-Befehl (C3...) zum Breakpoint-Entry des Monitor geladen. Die beiden Adreß-Bytes können nun vom Anwender auf seine Routinen gerichtet werden. So wird der fehlende JP (nn) des Z80 simuliert, wobei alle Register unverändert bleiben. Um die Kommando-Decodierung im Monitor für Erweiterungen zu öffnen, erfolgt am Ende der Monitor-Hauptschleife ein indirekter Sprung nach obigem Schema über den Monitor-Erweiterungsvektor (MONEVE Adr. 8006H). Er wird beim RESET auf die „Error"-Routine des Monitors gerichtet und bewirkt bei einem unbekannten Kommando eine Fehlermeldung auf dem Display. Der Benutzer kann nun diesen Vektor auf eine selbst definierte Fortsetzung des Kommando-Decoders richten und so z. B. Monitor-Erweiterungen testen, bevor sie ins EPROM kommen.

Ein kommentiertes Source-Listing ist beim Franzis-Software-Service erhältlich.

Funktionen und Kommandos

Die Belegung der Tastatur ist in *Bild 4* gezeigt, die möglichen Monitor-Kommandos sind in der *Tabelle* aufgelistet. Die meisten Kommandos sind dialogorientiert gestaltet und benutzen das 7-Segment-Display zur Textausgabe so gut es eben geht. In der Monitor-Hauptschleife stellen auch die Hex-Tasten Kommandotasten dar, zur Zifferneingabe werden sie erst innerhalb der Kommandos benutzt. Mit Hilfe der Shift-Taste (∧) stehen 38 Kommandotasten zur Verfügung. Die Verwendung der Shift-Funktion erfolgt wie beim Taschenrechner (zuerst Shift-, dann Kommandotaste). Die Shift-Funktion wird durch den

Tabelle 1: Die Monitor-Kommandos auf einen Blick

Folgende Tasten sind mit Funktionen belegt:

0	Speicher durchblättern
1	Speicherausschnitt verschieben
2	Speicher füllen
3	Programm seriell senden bzw. auf Kassette speichern
4	Programm von Kassette bzw. serieller Schnittstelle laden
5	Verify von Kassette
6	CPU-Register nach Break anschauen evtl. ändern
7	Continue nach Break
8	Relative Sprungweite berechnen
9	DCF 77 decodieren (St. Min. Sek.)
GO	Start des USER-Programms
MEM	Memory anzeigen

Dezimalpunkt ganz rechts im Display angezeigt. Ein nicht definiertes Kommando bringt „Error" aufs Display (siehe auch Monitorerweiterung).

Bei der Hex-Eingabe werden die Ziffern von rechts nach links im Display durchgeschoben, Korrektur erfolgt durch Überschreiben. Innerhalb der einzelnen Kommandos wirkt die „+"-Taste wie eine Return-Taste zum Beenden der Zifferneingabe. Der Abschluß des Kommandos und die Rückkehr zum Monitor wird durch „Mon 3.2" angezeigt. Mittels Reset kann jedes Kommando abgebrochen werden.

Die Monitor-Kommandos im einzelnen

□ Memory-Display: Taste Mem
Hiermit können Daten im RAM abgelegt werden.

Taste	Anzeige	
Mem	M.0000	Adreßeingabe
Mem	M.aaaa dd	Dateneingabe
∧Mem	Mon	Rückkehr zur Monitor-Schleife

Mit der Mem-Taste kann zwischen Adreß- und Dateneingabe hin- und hergeschaltet werden.

□ Go-Kommando: Taste Go

Go	G 0000	Eingabe der Startadresse

Nochmaliges Betätigen der Go-Taste schickt die CPU auf die Reise. Jede andere Kommando-Taste bewirkt Rückkehr zum Monitor. Beim Start des User-Programms werden alle CPU-Register (Bank 1) aus dem Puffer im RAM geladen (siehe Break-Kommando). Ein offenes RET am Ende des Anwenderprogramms führt zurück zum Monitor.

□ Speicher durchblättern: Taste 0

0	Sta 0000	Eingabe der Startadresse
+	M.aaaa dd	
+	M.aaaa+1 dd	vorwärts blättern
Go	M.aaaa–1 dd	rückwärts blättern
Mem	M.aaaa dd	Einsprung ins normale Mem-Kommando
∧Mem	Mon	Rückkehr zum Monitor

Hier kann man z. B. nach einem bestimmten Byte im Speicher suchen, da die „+"- und die „Go"(„–")-Taste mit Autorepeat ausgestattet sind.

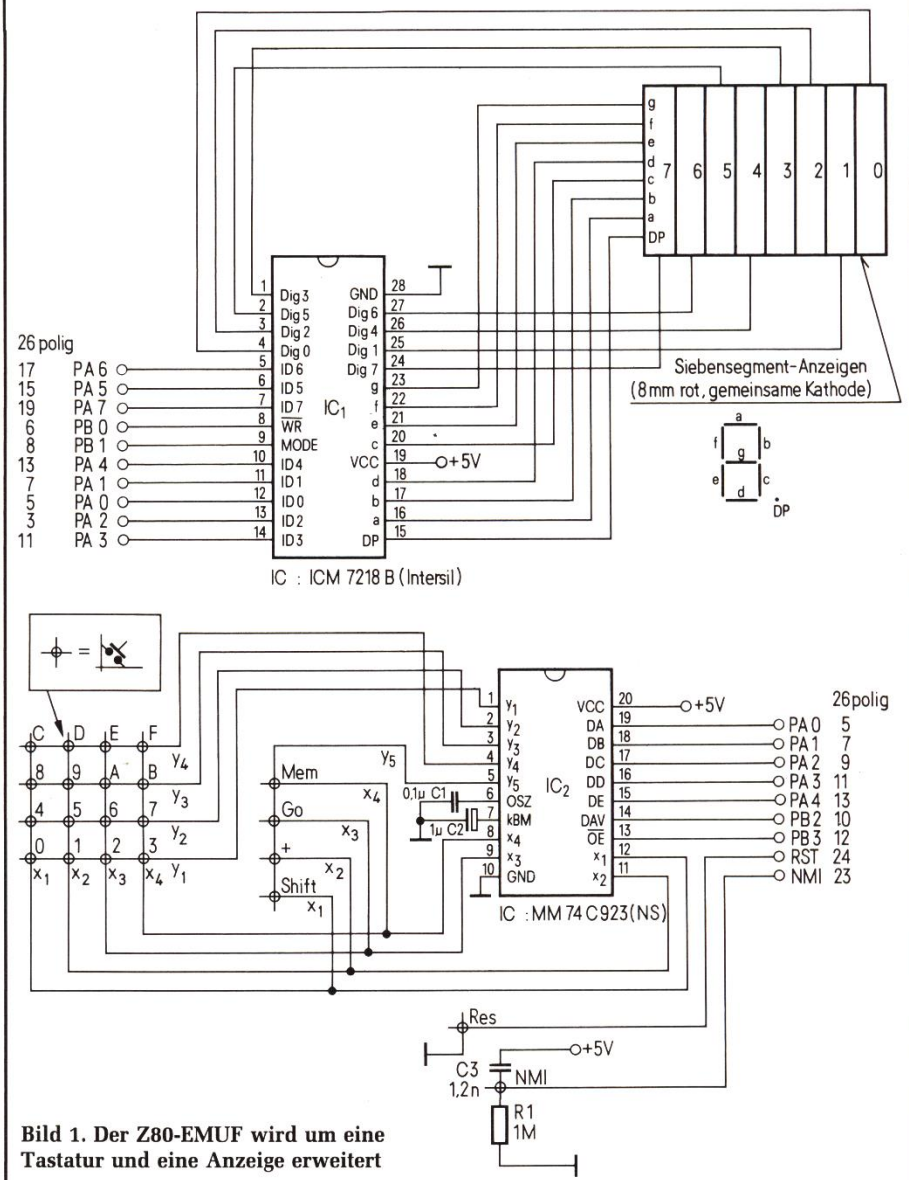


Bild 1. Der Z80-EMUF wird um eine Tastatur und eine Anzeige erweitert

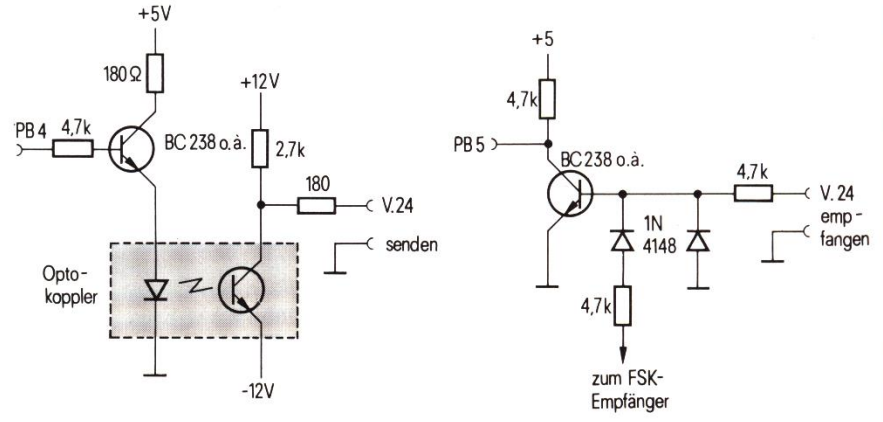
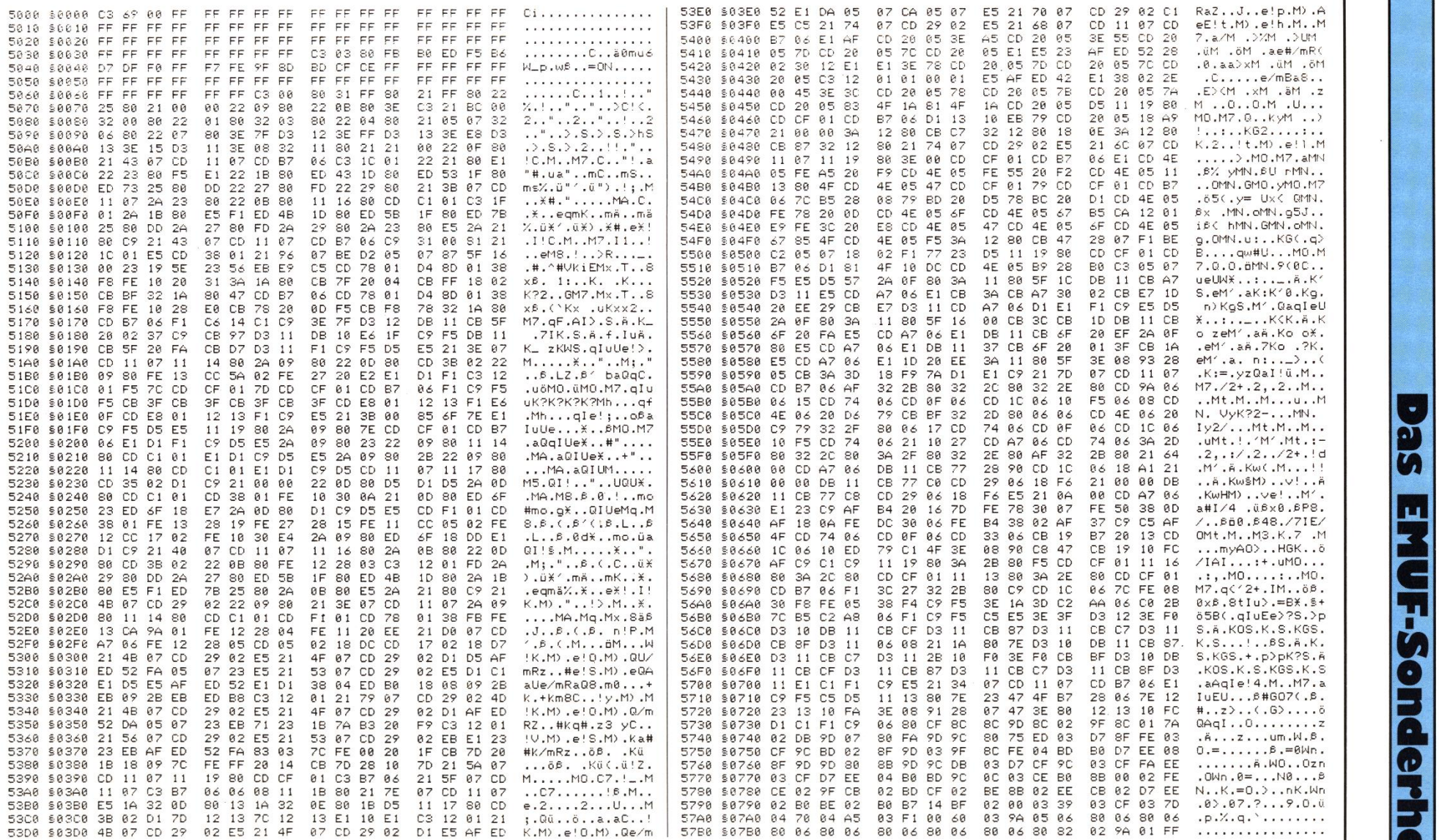


Bild 2. Die Hardware zur V.24-Schnittstelle



```
5000 $0000 C3 69 00 FF  FF FF FF FF  FF FF FF FF  FF FF FF FF  Ci..............
5010 $0010 FF FF FF FF  FF FF FF FF  FF FF FF FF  FF FF FF FF  ................
5020 $0020 FF FF FF FF  FF FF FF FF  FF FF FF FF  FF FF FF FF  ................
5030 $0030 FF FF FF FF  FF FF FF FF  C3 03 80 FB  B0 ED F5 B6  ........C..à0muö
5040 $0040 D7 DF F0 FF  F7 FE 9F 8D  BD CF CE FF  FF FF FF FF  W_p.wß..=ON.....
5050 $0050 FF FF FF FF  FF FF FF FF  FF FF FF FF  FF FF FF FF  ................
5060 $0060 FF FF FF FF  FF FF C3 00  80 31 FF 80  21 FF 80 22  ......C..1..!.."
5070 $0070 25 80 21 00  00 22 09 80  22 0B 80 3E  C3 21 BC 00  %.!..".."..>C!<.
5080 $0080 32 00 80 22  01 80 32 03  80 22 04 80  21 05 07 32  2.."..2.."..!..2
5090 $0090 06 80 22 07  80 3E 7F D3  12 3E FF D3  13 3E E8 D3  .."..>.>.S.>.S.>hS
50A0 $00A0 13 3E 15 D3  11 3E 08 32  11 80 21 21  00 22 0F 80  .>.S.>.2..!!."..
50B0 $00B0 21 43 07 CD  11 07 CD B7  06 C3 1C 01  22 21 80 E1  !C.M..M7.C.."!.a
50C0 $00C0 22 23 80 F5  E1 22 1B 80  ED 43 1D 80  ED 53 1F 80  "#.uä"..mC..mS..
50D0 $00D0 ED 73 25 80  DD 22 27 80  FD 22 29 80  21 3B 07 CD  ms%.ü"'.ü").!;.M
50E0 $00E0 11 07 2A 23  80 22 0B 80  11 16 80 CD  C1 01 C3 1F  ..*#."......MA.C.
50F0 $00F0 01 2A 1B 80  E5 F1 ED 4B  1D 80 ED 5B  1F 80 ED 7B  .*..eqmK..mä..mä
5100 $0100 25 80 DD 2A  27 80 FD 2A  29 80 2A 23  80 E5 2A 21  %.ü*'.ü*).*#.e*!
5110 $0110 80 C9 21 43  07 CD 11 07  CD B7 06 C9  31 00 81 21  .I!C.M..M7.I1..!
5120 $0120 1C 01 E5 CD  38 01 21 96  07 BE D2 05  07 87 5F 16  ..eM8.!..>R...._.
5130 $0130 00 23 19 5E  23 56 EB E9  C5 CD 78 01  D4 8D 01 38  .#.^#VkiEMx.T..8
5140 $0140 F8 FE 10 20  31 3A 1A 80  CB 7F 20 04  CB FF 18 02  x8. 1:..K. .K....
5150 $0150 CB BF 32 1A  80 47 CD B7  06 CD 78 01  D4 8D 01 38  K?2..GM7.Mx.T..8
5160 $0160 F8 FE 10 28  E0 CB 78 20  0D F5 CB F8  78 32 1A 80  x8.(`Kx .uKxx2..
5170 $0170 CD B7 06 F1  C6 14 C1 C9  3E 7F D3 12  DB 11 CB 5F  M7.qF.AI>.S.ä.K_
5180 $0180 20 02 37 C9  CB 97 D3 11  DB 10 E6 1F  C9 F5 DB 11  .7IK.S.ä.f.IuÄ.
5190 $0190 CB 5F 20 FA  CB D7 D3 11  F1 C9 F5 D5  E5 21 3E 07  K_ zKWS.qIuUe!>.
51A0 $01A0 CD 11 07 11  14 80 2A 09  80 22 0D 80  CD 3B 02 22  M.....*.."..M;."
51B0 $01B0 09 80 FE 13  CC 5A 02 FE  27 20 E2 E1  D1 F1 C3 12  ..ß.LZ.ß' baQqC.
51C0 $01C0 01 F5 7C CD  CF 01 7D CD  CF 01 CD B7  06 F1 C9 F5  .uöMO.üMO.M7.qIu
51D0 $01D0 F5 CB 3F CB  3F CB 3F CB  3F CD E8 01  12 13 F1 E6  uK?K?K?K?Mh...q+
51E0 $01E0 0F CD E8 01  12 13 F1 C9  E5 21 3B 00  85 6F 7E E1  .Mh...qIe!;..oß~a
51F0 $01F0 C9 F5 D5 E5  11 19 80 2A  09 80 7E CD  CF 01 CD B7  IuUe...*..~MO.M7
5200 $0200 06 E1 D1 F1  C9 D5 E5 2A  09 80 23 22  09 80 11 14  .aQqIUe*..#"....
5210 $0210 80 CD C1 01  E1 D1 C9 D5  E5 2A 09 80  2B 22 09 80  .MA.aQIUe*..+"..
5220 $0220 11 14 80 CD  C1 01 E1 D1  C9 D5 CD 11  07 11 17 80  ...MA.aQIUM.....
5230 $0230 CD 35 02 D1  C9 21 00 00  22 0D 80 D5  D1 D5 2A 0D  M5.QI!.."..UQU*.
5240 $0240 80 CD C1 01  CD 38 01 FE  10 30 0A 21  0D 80 ED 6F  .MA.M8.ß.0.!..mo
5250 $0250 23 ED 6F 18  E7 2A 0D 80  D1 C9 D5 E5  CD F1 01 CD  #mo.g*..QIUeMq.M
5260 $0260 38 01 FE 13  28 19 FE 27  28 15 FE 11  CC 05 02 FE  8.ß.(.ß'(.ß.L..ß
5270 $0270 12 CC 17 02  FE 10 30 E4  2A 09 80 ED  6F 18 DD E1  .L..ß.0d*..mo.üa
5280 $0280 D1 C9 21 40  07 CD 11 07  11 16 80 2A  0B 80 22 0D  QI!§.M.....*.."..
5290 $0290 80 CD 3B 02  22 0B 80 FE  12 28 03 C3  12 01 FD 2A  .M;."..ß.(.C..ü*
52A0 $02A0 29 80 DD 2A  27 80 ED 5B  1F 80 ED 4B  1D 80 2A 1B  ).ü*'.mä..mK..*.
52B0 $02B0 80 E5 F1 ED  7B 25 80 2A  0B 80 E5 2A  21 80 C9 21  .eqmä%.*..e*!.I!
52C0 $02C0 4B 07 CD 29  02 22 09 80  21 3E 07 CD  11 07 2A 09  K.M).."..!>.M..*.
52D0 $02D0 80 11 14 80  CD C1 01 CD  F1 01 CD 78  01 38 FB FE  ....MA.Mq.Mx.8ä8
52E0 $02E0 13 CA 9A 01  FE 12 28 04  FE 11 20 EE  21 D0 07 CD  .J..ß.(.ß. n!P.M
52F0 $02F0 A7 06 FE 12  28 05 CD 05  02 18 DC CD  17 02 18 D7  '.ß.(.M...öM...W
5300 $0300 21 4B 07 CD  29 02 E5 21  4F 07 CD 29  02 D1 D5 AF  !K.M).e!O.M).QU/
5310 $0310 ED 52 FA 05  07 23 E5 21  53 07 CD 29  02 E5 D1 C1  mRz..#e!S.M).eQA
5320 $0320 E1 D5 E5 AF  ED 52 E1 D1  38 04 ED B0  18 08 09 2B  aUe/mRaQ8.m0...+
5330 $0330 EB 09 2B EB  ED B8 C3 12  01 21 79 07  CD 29 02 4D  k.+km8C..!y.M).M
5340 $0340 21 4B 07 CD  29 02 E5 21  4F 07 CD 29  02 D1 AF ED  !K.M).e!O.M).Q/m
5350 $0350 52 DA 05 07  23 EB 71 23  1B 7A B3 20  F9 C3 12 01  RZ..#kq#.z3 yC..
5360 $0360 23 E5 21 56  07 CD 29 02  E5 21 53 07  CD 29 02 EB  #e!V.M).e!S.M).ka#
5370 $0370 23 EB AF ED  52 FA 83 03  7C FE 00 20  1F CB 7D 20  #k/mRz..öß. .Kü
5380 $0380 1B 18 09 7C  FE FF 20 14  CB 7D 28 10  7D 21 5A 07  ...öß. .Kü(.ü!Z.
5390 $0390 CD 11 07 11  19 80 CD CF  01 C3 B7 06  21 5F 07 CD  M.....MO.C7.!_.M
53A0 $03A0 11 07 C3 B7  06 06 08 11  1B 80 21 7E  07 CD 11 07  ..C7......!ß.M..
53B0 $03B0 E5 1A 32 0D  80 13 1A 32  0E 80 1B D5  11 17 80 CD  e.2....2....U...M
53C0 $03C0 3B 02 D1 7D  12 13 7C 12  13 E1 10 E1  C3 12 01 21  ;.Qü..ö..a.aC..!
53D0 $03D0 4B 07 CD 29  02 E5 21 4F  07 CD 29 02  D1 E5 AF ED  K.M).e!O.M).Qe/m
53E0 $03E0 52 E1 DA 05  07 CA 05 07  E5 21 70 07  CD 29 02 C1  Ra2..J..e!p.M).A
53F0 $03F0 E5 C5 21 74  07 CD 29 02  E5 21 68 07  CD 11 07 CD  eE!t.M).e!h.M..M
5400 $0400 B7 06 E1 AF  CD 20 05 3E  A5 CD 20 05  3E 55 CD 20  7.a/M .>%M .>UM
5410 $0410 05 7D CD 20  05 7C CD 20  05 E1 E5 23  AF ED 52 28  .üM .öM .ae#/mR(
5420 $0420 02 30 12 E1  E1 3E 78 CD  20 05 7D CD  20 05 7C CD  .0.aa>xM .üM .öM
5430 $0430 20 05 C3 12  01 01 00 01  E5 AF ED 42  E1 38 02 2E  .C.....e/mBa8..
5440 $0440 00 45 3E 3C  CD 20 05 78  CD 20 05 7B  CD 20 05 7A  .E>M .xM .äM .z
5450 $0450 CD 20 05 83  4F 1A 81 4F  1A CD 20 05  D5 11 19 80  M ..O..O.M .U...
5460 $0460 CD CF 01 CD  B7 06 D1 13  10 EB 79 CD  20 05 18 A9  MO.M7.Q..kyM ..)
5470 $0470 21 00 00 3A  12 80 CB C7  32 12 80 18  0E 3A 12 80  !..:..KG2....:..
5480 $0480 CB 87 32 12  80 21 74 07  CD 29 02 E5  21 6C 07 CD  K.2..!t.M).e!l.M
5490 $0490 11 07 11 19  80 3E 00 CD  CF 01 CD B7  06 E1 CD 4E  .....>.MO.M7.aMN
54A0 $04A0 05 FE A5 20  F9 CD 4E 05  FE 55 20 F2  CD 4E 05 11  .%. yMN.%U rMN..
54B0 $04B0 13 80 4F CD  4E 05 47 CD  CF 01 79 CD  CF 01 CD B7  ..OMN.GMO.yMO.M7
54C0 $04C0 06 7C B5 28  08 79 BD 20  D5 78 BC 20  D1 CD 4E 05  .ö5(.y= Ux< QMN.
54D0 $04D0 FE 78 20 0D  CD 4E 05 6F  CD 4E 05 67  B5 CA 12 01  ßx .MN.oMN.g5J..
54E0 $04E0 E9 FE 3C 20  E8 CD 4E 05  47 CD 4E 05  6F CD 4E 05  iß< hMN.GMN.oMN.
54F0 $04F0 67 85 4F CD  4E 05 F5 3A  12 80 CB 47  28 07 F1 BE  g.OMN.u:..KG(.q>
5500 $0500 C2 05 07 18  02 F1 77 23  D5 11 19 80  CD CF 01 CD  B....qw#U...MO.M
5510 $0510 B7 06 D1 81  4F 10 DC CD  4E 05 B9 28  B0 C3 05 07  7.Q.O.ÜMN.9(0C..
5520 $0520 F5 E5 D5 57  2A 0F 80 3A  11 80 5F 1C  DB 11 CB A7  ueUW*..:.._.Ä.K'
5530 $0530 D3 11 E5 CD  A7 06 E1 CB  3A CB A7 30  02 CB E7 1D  S.eM'.aK:K'0.Kg.
5540 $0540 20 EE 29 CB  E7 D3 11 CD  A7 06 D1 E1  F1 C9 E5 D5  n)KgS.M'.QaqIeU
5550 $0550 2A 0F 80 3A  11 80 5F 16  00 CB 3C CB  1D DB 11 CB  *..:.._..K<K.Ä.K
5560 $0560 6F 20 FA E5  CD A7 06 E1  DB 11 CB 6F  20 EF 2A 0F  o zeM'.aÄ.Ko o*.
5570 $0570 80 E5 CD A7  06 E1 DB 11  37 CB 6F 20  01 3F CB 1A  .eM'.aÄ.7Ko .?K.
5580 $0580 E5 CD A7 06  E1 1D 20 EE  3A 11 80 5F  3E 08 93 28  eM'.a. n:.._>..(
5590 $0590 05 CB 3A 3D  18 F9 7A D1  E1 C9 21 7D  07 CD 11 07  .K:=.yzQaI!ü.M..
55A0 $05A0 CD B7 06 AF  32 2B 80 32  2C 80 32 2E  80 CD 9A 06  M7./2+.2,.2..M..
55B0 $05B0 06 15 CD 74  06 CD 0F 06  CD 1C 06 10  F5 06 08 CD  ..Mt.M..M....u..M
55C0 $05C0 4E 06 20 D6  79 CB BF 32  2D 80 06 06  CD 4E 06 20  N. VyK?2-...MN.
55D0 $05D0 C9 79 32 2F  80 06 17 CD  74 06 CD 0F  06 CD 1C 06  Iy2/...Mt.M..M..
55E0 $05E0 10 F5 CD 74  06 21 10 27  CD A7 06 CD  74 06 3A 2D  .uMt.!.'M'.Mt.:-
55F0 $05F0 80 32 2C 80  3A 2F 80 32  2E 80 AF 32  2B 80 21 64  .2,.:/.2../2+.!d
5600 $0600 00 CD A7 06  DB 11 CB 77  28 90 CD 1C  06 18 A1 21  .M'.Ä.Kw(.M...!!
5610 $0610 00 00 DB 11  CB 77 C0 CD  29 06 18 F6  21 00 00 DB  ..Ä.Kw§M)..v!..Ä
5620 $0620 11 CB 77 C8  CD 29 06 18  F6 E5 21 0A  00 CD A7 06  .KwHM)..ve!..M'.
5630 $0630 E1 23 C9 AF  B4 20 16 7D  FE 78 30 07  FE 50 38 0D  a#I/4 .ü8x0.ß P8.
5640 $0640 AF 18 0A FE  DC 30 06 FE  B4 38 02 AF  37 C9 C5 AF  /..ßö0.ß48./7IE/
5650 $0650 4F CD 74 06  CD 33 06 CB  19 B7 20 13  CD 0F 01 CD  OMt.M..M3.K.7 .M
5660 $0660 1C 06 10 ED  79 C1 4F 3E  08 90 C8 47  CB 19 10 FC  ...myAO>..HGK..ö
5670 $0670 AF C9 C1 C9  11 19 80 3A  2B 80 F5 CD  CF 01 11 16  /IAI...:+.uMO...
5680 $0680 80 3A 2C 80  CD CF 01 11  13 80 3A 2E  80 CD CF 01  .:,.MO....:..MO.
5690 $0690 CD B7 06 F1  3C 27 32 2B  80 C9 CD 1C  06 7C FE 08  M7.q<'2+.IM..öß.
56A0 $06A0 30 F8 FE 05  38 F4 C9 F5  3E 1A 3D C2  AA 06 C0 2B  0x8.8tIu>.=B*.§+
56B0 $06B0 7C B5 C2 A8  06 F1 C9 F5  C5 E5 3E 3F  D3 12 3E F0  ö5B(.qIuEe>?S.>p
56C0 $06C0 D3 10 DB 11  CB CF D3 11  CB 87 D3 11  CB C7 D3 11  S.Ä.KOS.K.S.KGS.
56D0 $06D0 CB 8F D3 11  06 08 21 1A  80 7E D3 10  DB 11 CB 87  K.S...!..ß§.Ä.K.
56E0 $06E0 D3 11 CB C7  D3 11 2B 10  F0 3E F0 CB  BF D3 10 DB  S.KGS.+.p>pK?S.Ä
56F0 $06F0 11 CB CF D3  11 CB 87 D3  11 CB C7 D3  11 CB 8F D3  .KOS.K.S.KGS.K.S
5700 $0700 11 E1 C1 F1  C9 E5 21 34  07 CD 11 07  CD B7 06 E1  .aAqIe!4.M..M7.a
5710 $0710 C9 F5 C5 D5  11 13 80 7E  23 47 4F B7  28 06 7E 12  IuEU...ß#GO7(.ß.
5720 $0720 23 13 10 FA  3E 08 91 28  07 47 3E 80  12 13 10 FC  #..z>..(.G>....ö
5730 $0730 D1 C1 F1 C9  06 80 CF 8C  8C 9D 8C 02  9F 8C 01 7A  QAqI..O........z
5740 $0740 02 DB 9D 07  80 FA 9D 9C  80 75 ED 03  D7 8F FE 03  .Ä...z....um.W.ß.
5750 $0750 CF 9C BD 02  8F 9D 03 9F  8C FE 04 BD  B0 D7 EE 08  O.=.....ß.=0Wn.
5760 $0760 8F 9D 9D 80  8B 9D 9C DB  03 D7 CF 9C  03 CF FA EE  .......Ä.WO..Ozn
5770 $0770 03 CF D7 EE  04 B0 BD 9C  0C 03 CE B0  8B 00 02 FE  .OWn.0=...N0...ß
5780 $0780 CE 02 9F CB  02 BD CF 02  BE 8B 02 EE  CB 02 D7 EE  N..K.=O.>..nK.Wn
5790 $0790 02 B0 BE 02  B0 B7 14 BF  02 00 03 39  03 CF 03 7D  .0>.07.?...9.O.ü
57A0 $07A0 04 70 04 A5  03 F1 00 60  03 9A 05 06  80 06 80 06  .p.%.q.`........
57B0 $07B0 80 06 80 06  80 06 80 06  80 06 80 82  02 9A 01 FF  ................
```

Bild 3. Das Monitorprogramm erweckt den Z80-EMUF zum Leben. Es ist vom Franzis-Software-Service auch im EPROM erhältlich

□ MOVE-Kommando: Taste 1

1	Sta	0000	Startadresse des Blocks eingeben
+	End	0000	Endadresse
+	to	0000	Zieladresse
+	Mon		

Bei Startadresse ≦ Endadresse erfolgt eine Fehlermeldung

□ FILL-Kommando: Taste 2

2	FIL	0000	Eingabe des „Fill-Bytes" (nur die beiden rechten Ziffern sind gültig)
+	Sta	0000	Startadresse
+	End	0000	Endadresse
+	Mon		

□ SAVE-Kommando: Taste 3

3	Sta	0000	Start des Speicherblocks
+	End	0000	Ende des Speicherblocks
+	Esp	0000	Einsprungadresse (siehe unten)
+	ldnr	0000	Eingabe einer vierstelligen Kennummer des Programmblocks auf Band
+	Sen	XX	XX „flimmernde" Sendedaten

Die Einsprungadresse wird mit aufgezeichnet und nach dem Laden des Datenblocks startet der Prozessor an dieser Stelle. Dadurch wird ein Autostart des Programms möglich. Wird als Einsprungadresse 0000 eingegeben bzw. „+" gedrückt, dann kehrt der Rechner nach dem Laden zum Monitor zurück.

□ LOAD-Kommando: Taste 4

4	Idnr	0000	Eingabe der Kennummer des zu ladenden Programms. Wird 0000 eingegeben bzw. „+" gedrückt, dann wird das nächste vollständige Programm vom Band geladen.
+	Emp	00	Empfangsbereitschaft des Prozessors

Sobald Daten empfangen werden, verändert sich die Anzeige:

iiii	xx	iiii Id-Nummer des empfangenen Programms xx empfangene Daten

nach abgeschlossenem Ladevorgang siehe LOAD-Kommando.

□ Verify-Kommando: Taste 5

5	EMP	00 siehe LOAD

Das auf Band geschriebene Programm kann mit dem Speicherinhalt verglichen werden. Wenn keine Fehler aufgetreten sind, erfolgt Rückkehr zum Monitor, andernfalls wird „Error" gemeldet und die Operation abgebrochen.

□ Anzeige der CPU-Register: Taste 6
Diese Funktion ist nur sinnvoll nach einem NMI oder Breakpoint. Nur dann sind die Puffer-Zellen im RAM auf aktuellem Stand.

6	AF	aaff	Anzeige von Akku und Flags, zu ändern wie Memory-Zellen
+	BC	bbcc	
...			
+	IY	yyyy	
+	Mon		

□ Continue-Kommando: Taste 7
Auch dieses Kommando sollte nur nach NMI oder Break verwendet werden. Andernfalls kann es zum Absturz des Rechners führen.
Die eventuell modifizierten RAM-Zellen werden in die CPU-Register geladen und das Programm beim augenblicklichen Programmzählerstand fortgesetzt.

□ Displacement-Rechner: Taste 8
Hiermit kann der Offset für die relativen Sprünge der Z80-CPU berechnet werden.

8	bra	0000	Adresse des JR-Befehls (nicht des Displacement-Bytes) eingeben
+	to	0000	Adresse des Ziel-Labels
+	disp	xx	Displacement

Wurde die maximale Sprungweite überschritten, so erscheint die Meldung „too long" im Display.

□ DCF-77-Decodierung: Taste 9
Mit diesem Kommando kann der „Kleine" nach Anschluß eines einfachen DCF-77-Empfängers an PIO2, PB 6, in eine hochgenaue Uhr verwandelt werden. Der Empfänger sollte TTL-Pegel (Ruhezustand high) liefern. Der EMUF verdaut für log. 0 80...120 ms Absenkung, für log. 1 180...220 ms.

Um den Programmaufwand gering zu halten, decodiert der Rechner nur Stunden, Minuten und Sekunden und zeigt sie auf dem Display an. Nach Start des Programms wartet der EMUF die 59-Sekunden-Marke ab und zeigt 00 00 an. Nach einer weiteren Minute werden auch die Stunden und Minuten angezeigt, die maximale Synchronisationszeit beträgt also 1 Minute und 59 Sekunden. Treten während des Empfangs Störungen auf, so erlischt das Display, um sinnlose Zeitanzeigen zu vermeiden; der Rechner versucht erneut zu synchronisieren. Aus diesem Programm kann der EMUF nur mit Reset zurückgeholt werden.

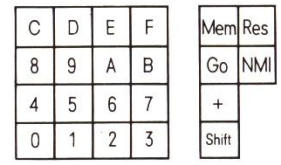


Bild 4. Die Belegung der kleinen Tastatur

BREAK- und STOP-Befehl

Als BREAK-Befehl wird, wie allgemein üblich, der RST 38H (0FFH) verwendet. Taucht er im Programm auf, so werden die CPU-Register ins RAM gerettet und man landet im Monitor (siehe auch Kommando 6 und 7).

Der NMI der CPU wird als STOP-Befehl interpretiert, um die CPU definiert aus endlosen Schleifen herauszuholen. Die Register werden ebenfalls gerettet. Dieses Kommando funktioniert nur, solange der NMI-Vektor im RAM nicht verändert wird. Zum Entprellen der NMI-Taste wird ein einfaches RC-Glied verwendet (siehe Bild 1).

Im folgenden soll noch ausführlich auf die Möglichkeiten des seriellen Empfangs bzw. Sendens mittels des softwaremäßig implementierten USART's eingegangen werden.

Die Tasten 3 + 4 zum Abspeichern und Laden von Programmen bedienen sich des Datenformats in *Tabelle 2.*

Der Markierzustand der Schnittstelle ist high. Die Daten haben seriell folgende Form:

8 Bit breite Worte, 1 Startbit, 2 Stopbits. Die Übertragungsrate beträgt 300 Baud. Durch Änderung der entsprechenden RAM-Adressen (BITNR 8011H, BAUDCT 800FH) lassen sich Baudraten bis zu 2400 Baud und beliebige (bis 8 Bit) Wortbreiten einstellen. Entsprechende Versuche sind bereits erfolgreich durchgeführt worden. Die Sendung von

einem Startbit und 2 Stopbits ist fest programmiert.

Hierdurch ergibt sich die interessante Möglichkeit, Programme, die dieses Format einhalten, von einem Host-Rechner zu laden und automatisch ausführen zu lassen. Wer dieses Schema nicht einhalten möchte, kann direkt auf die Schnittstellentreiber zugreifen und sein eigenes Übertragungsprotokoll ablaufen lassen.

Der Rechner könnte durch Nutzung der seriellen Schnittstelle und entsprechender Programme völlig ferngesteuert werden. Außerdem gilt es noch zu beachten, daß er nur entweder senden oder empfangen kann. Beides gleichzeitig ist nicht möglich.

Die Aufzeichnung der Programme mittels des (Funkschau-)FSK-Modems funktionierte auf Anhieb. Die Anforderungen in bezug auf Kassetten- und Recorderqualität sind gering. Da keine Umschaltung von Recorderinterface auf die serielle Schnittstelle notwendig ist, versteht es sich von selbst, daß beim Empfang nur eine der beiden Quellen in Betrieb sein darf. Die Anschlüsse der nicht benutzten Schnittstelle müssen offen bleiben.

Tabelle 2: Das Datenformat beim Speichern auf Kassette

gesendete Bytes	Anzahl	
0	1	
A5H	1	Marken für Programmbeginn
55H	1	
Kennummer	2	Reihenfolge low Byte, high Byte
3CH	1	Startbyte für 1. Datenfeld
Länge	1	Länge des Datenfeldes, max. 256 Bytes/ Block
Quelladresse	2	low Byte, high Byte
Datenfeld	Länge	
Checksum	1	(Summe aller gesendeten Bytes pro Block) modulo 256
2. Datenblock		
n-ter Datenblock		
letzter Datenblock		
78H	1	Marke Fileende
Einsprungadresse	2	low Byte, high Byte

Wichtige RAM-Adressen

8000H NMIVE

Sprungvektor für NMI 'C3 xx xx'

8003H IRQVE

Sprungvektor für Interrupt Mode 1 bzw. RST 38H.
Analog NMIVE

8006H MONEVE

Monitorerweiterungsvektor. Zeigt nach der Initialisierung auf die Error-Routine. Hier kann der Benutzer Fehlermeldungen abfangen, die nach einer nicht definierten Taste in der Monitorschleife ausgegeben werden. Der Akku enthält zu diesem Zeitpunkt 'Tastencode * 2'. 'SLR A' ergibt den Tastencode.

800FH BAUDCT

Zeitkonstante für Baudrate. Durch entsprechende Werte lassen sich die Baudraten einstellen. Folgende Beziehung gilt:

1/Baudrate = (BAUDCT)*100 Mikrosekunden

Folgende Baudraten wurden erprobt:

Baudrate	BAUDCT
75	83H
150	42H
300	21H
600	10H
1200	8H
2400	4H

Nach einem Reset sind 300 Baud eingestellt.

8011H BITNR

Anzahl der Bits pro seriellem Datenwort.

8013H DISBUF

Displaypuffer. Hier stehen die Bitmuster für den aktuellen Displayinhalt (8 Zeichen).

Segment – Bit Zuordnung (s. Bild 1):

Bit	7	6	5	4	3	2	1	0
Segment	$\overline{DP}$	a	b	c	e	g	f	d

Wichtige ROM-Adressen und Unterprogramme

011FH		Monitorwarmstart
0796H		Kommandotabelle
0138H	KEYIN	Wartet auf Taste. Tastenwert im Akku, berücksichtigt Shift-Taste
0178H	GETKEY	Tastenfeldabfrage Taste gedrückt → Carry = 0, A = Tastencode Taste nicht gedrückt → Carry = 1, A = xx
01C1H	HLDISP	HL als 4-Digit-Zahl ausgeben. DE ist Pointer auf 1. Digit
01CFH	TO-DIBUF	Hexzahl im Akku wird als Bitmuster im Displaypuffer abgelegt. DE ist Pointer auf 1. Stelle
0235H	HLINCL	Eingabe einer 4stelligen Hexzahl in HL. Durchschieben und Echo auf Display (DE als Pointer). Abbruch der Eingabe durch Kommandotaste (> 0FH)
05200	SERSEN	Byte in A seriell senden (siehe BAUDCT und BITNR)
054E	SEREMP	Byte seriell empfangen (siehe oben). Empfangenes Byte im Akku
06B7H	DIS-POUT	Ausgabe des Displaypuffers aufs Display (siehe DISBUF)
0711H	FILBUF	Zeichenkette in den Displaypuffer schreiben; Format: n CC...C $0 \leqq n \leqq 8$ n = Anzahl der Textzeichen (Bitmuster) Register HL zeigt auf „n“

Literatur

[1] *Feichtinger, Herwig:* FSK-Demodulator/-Oszillator. Funkschau 1978, Heft 14, Seite 698 und Heft 15, Seite 742.

Dr. Dieter Götz

Z80-EMUF mit Display und Tastatur

In mc 1983, Heft 4, wurde der äußerst preiswerte Z80-EMUF vorgestellt [1]. Im vollausgebauten Zustand verfügt er über 40 programmierbare Ein-/Ausgabeleitungen und zusätzlich über eine Reihe von Steuer- und Interruptleitungen. Ziel dieses Beitrages ist es, seinen Einsatz durch den Anschluß einer einfachen Tastatur und Anzeige sowie durch einen entsprechenden Monitor noch vielseitiger zu machen.

Das Grundkonzept war, mit möglichst wenig Hardware auszukommen, andererseits aber die Zahl der vorhandenen Ein-/Ausgabeleitungen für den späteren Anschluß von A/D- bzw. D/A-Wandlern zu erhalten.
Es wurde deshalb eine zusätzliche Schnittstelle, und zwar der Parallelbaustein 8255, an den EMUF angeschlossen. Er findet neben der Stromversorgung auf dem reichlich bemessenen Lochrasterteil der EMUF-Platine Platz. Zur Adreßdecodierung wird ein noch freier Anschluß des 2-Bit-Binärdecoders (74139) des EMUF verwandt (Pin 10). Die Adressen für den 8255 sind 20H...23H.
Außer der Schnittstelle werden nur noch sechs 7-Segment-Anzeigen, 13 Transistoren, 16 Tasten und einige Widerstände benötigt. Das Multiplexen der sechs Digits und die Codierung der 7-Segment-Ziffern wird dem EMUF übertragen.

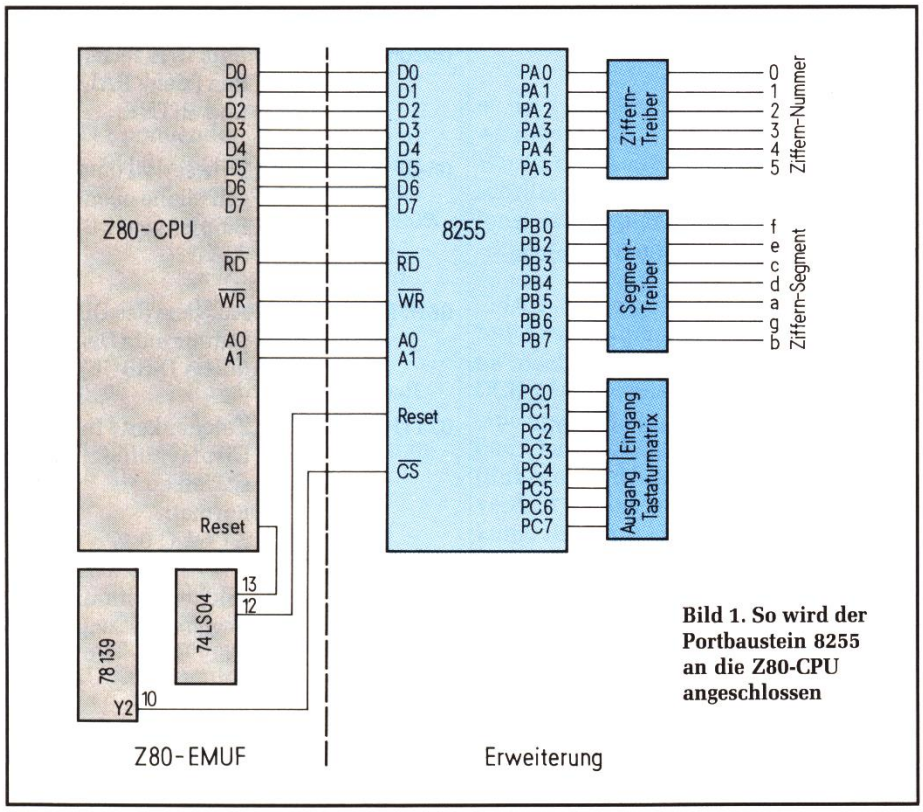


Bild 1. So wird der Portbaustein 8255 an die Z80-CPU angeschlossen

Von den Ein-/Ausgabeleitungen des EMUF wird nur eine einzige benötigt und zwar Bit 7 von Port B des Z80-PIO 0. Sie dient zum seriellen Datenaustausch mit anderen Computern. Im vorliegenden Fall war dies ein TRS-80, M1, auf dem z. B. größere Programme erstellt und getestet wurden, die dann anschließend zum EMUF transferiert wurden. Andererseits können auch vom EMUF gesammelte Daten zur Weiterverarbeitung, z. B. für eine graphische Auswertung, überspielt werden.

Der Anschluß der Siebensegment-Anzeigen

Bild 1 zeigt den Anschluß der Schnittstelle an die Z80-CPU. Das Reset-Signal des Z80 muß noch invertiert werden. Dies geschieht über die noch freien Pins 13 und 12 des sich auf der Platine befindenden Inverterbausteins. Von Port A des 8255 werden die Ausgänge 0...5, von Port B die Ausgänge 0 und 2...7 benötigt. Die Tastatur wird an Port C angeschlossen.

Da die Stromaufnahme der verwendeten Siebensegment-Anzeigen (HA 1077y von Siemens, gemeinsame Kathode) relativ gering ist, genügen als Treiber Transistoren vom Typ BC107. Der Anschluß der Transistoren als Segment- bzw. Zifferntreiber ist aus *Bild 2* ersichtlich.

Die Zuordnung der Matrixelemente der Tastatur

Matrix-element	Bedeutung ohne Shift	mit Shift
07	Ziffer: 0	Ziffer: 8
17	1	9
27	2	A
37	3	B
06	4	C
16	5	D
26	6	E
36	7	F
04	Shift (S)	Shift (S)
14	Clear (CR)	Out (O)
24	Break (BK)	In (IN)
34	Display (DP)	–
05	Enter (EN)	–
15	Insert (I)	–
25	Breakpoint (BP)	–
35	Go (G)	–

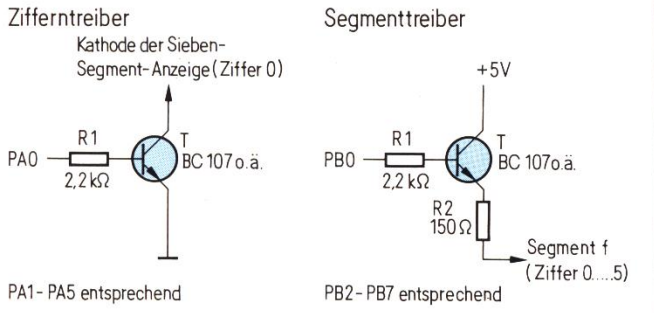


Bild 2. Die Siebensegment-Anzeigen werden mittels Transistoren gesteuert

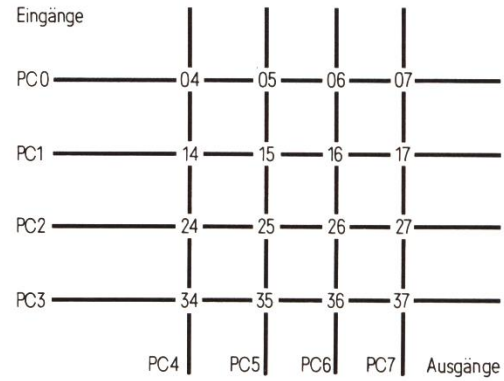


Bild 3. Der Aufbau der Tastaturmatrix

Zum Eingeben braucht man eine Tastatur

Auch bei der Tastatur wurde das eingangs erwähnte Konzept einer Minimierung der Hardware eingehalten. Es wurden lediglich 16 Tasten verwendet. Um neben den 16 Ziffern noch zusätzliche Funktionstasten zu haben, wurden die Tasten zum größten Teil doppelt belegt. *Bild 3* zeigt den Anschluß der Tastaturmatrix. Die Bedeutung der Matrixelemente zeigt die *Tabelle*.

Der Monitor mit seinen Funktionen

Das Listing des Monitors ist aus *Bild 4* ersichtlich.
Beim Einschalten des EMUF meldet sich der Monitor mit 'r u n'. Mit Ausnahme des IN-, O- und CR-Befehls ist jetzt eine entsprechende Speicherstelle einzugeben (zur Erinnerung: ROM:0000-07FF; RAM:8000-87FF). Die Speicherplätze 87D4-87FF werden durch den Monitor belegt. Der Stack beginnt bei 87D3. Nachfolgend nun eine Aufstellung der möglichen Befehle:
Display (DP)
Format: nnnnDP
(nnnn = 4stellige Adresse)
Ausgabe: Inhalt der Speicherstelle nnnn wird auf Ziffer 5 und 6 angezeigt.

```
oooo: 31 d3 87 3e 81 d3 23 dd 21 d6 87 dd 36 oo ef dd
oo1o: 36 o1 5f dd 36 o2 dc dd 36 o3 cd c3 23 oo oo oo
oo2o: c3 5d o3 dd 36 o4 49 dd 36 o5 d9 dd 36 o6 e7 af
oo3o: 32 f1 87 32 df 87 32 eo 87 32 dd 87 3e o3 32 de
oo4o: 87 oo oo oo oo oo oo cd ec oo cd 83 o3 3e oo d3
oo5o: 22 cd 97 oo 7b fe o2 ca cf o1 fe o3 ca d7 o1 fe
oo6o: o4 ca 7b o2 fe o5 ca e3 o1 fe o6 ca 7e o2 fe o7
oo7o: ca be o2 fe o8 ca 4a o3 fe o9 ca 43 o3 fe oa ca
oo8o: fe o3 fe ob ca oo o5 c2 91 oo 21 eo 87 34 c3 4d
oo9o: oo cd d3 oo c3 8a oo cd bd oo db 22 b7 e2 a8 oo
ooao: 21 eo 87 af 77 c3 97 oo 21 eo 87 7e b7 c2 97 oo
oobo: cd bd oo db 22 b7 ea 97 oo cd oa o1 c9 af 47 57
ooco: 5f 21 e1 87 13 cd fa oo 23 eb 29 eb 7b fe 4o c2
oodo: c5 oo c9 o6 oo 21 de 87 4e 21 e1 87 o9 73 od 79
ooeo: fe ff c2 e7 oo 3e o3 21 de 87 77 c9 o6 o6 21 e1
oofo: 87 3e oo 77 o5 c8 23 c3 f1 oo 7b d3 2o 7e d3 21
o1oo: oe 44 od 2o fd oo oo oo oo c9 c5 e5 21 9f o1 3a
o11o: df 87 b7 ca 1b o1 o6 oo oe 1o o9 3e oo 32 df 87
o12o: o6 da 3e eo d3 22 db 22 o5 c2 22 o1 e6 of fe of
o13o: c4 7c o1 23 23 23 23 o6 oa 3e do d3 22 db 22 o5
o14o: c2 39 o1 e6 of fe of c4 7c o1 23 23 23 23 o6 oa
o15o: 3e bo d3 22 db 22 o5 c2 5o o1 e6 of fe of c4 7c
o16o: o1 23 23 23 23 o6 oa 3e 7o d3 22 db 22 o5 c2 67
o17o: o1 e6 of fe of c4 7c o1 7b e1 c1 c9 fe oe c2 85
o18o: o1 7e c3 9d o1 23 fe od c2 8f o1 7e c3 9d o1 23
o19o: fe ob c2 99 o1 7e c3 9d o1 23 fe o7 7e 5f c9 o2
o1ao: o3 o4 o5 o6 o7 o8 o9 cb 7b 7f a8 bf 88 f6 fa o2
o1bo: oa ob oo oo oo oo oo 35 dc 77 67 ff fb ef 5f bf
o1co: 88 f6 fa cb 7b 7f a8 ff fb ef 5f 35 dc 77 67 3e
o1do: o2 32 df 87 c3 8a oo cd ec oo 21 de 87 3e o3 77
o1eo: c3 8a oo cd ef o1 cd oc o2 cd 27 o2 c3 8a oo 16
o1fo: o6 fd 21 e7 87 dd 21 e1 87 21 bf o1 cd 52 o2 78
o2oo: fd 77 oo dd 23 fd 23 15 c2 f9 o1 c9 fd 21 e7 87
o21o: fd 7e oo 47 fd 7e o1 cd 6o o2 6f fd 7e o2 47 fd
o22o: 7e o3 cd 6o o2 67 c9 7e 47 cd 6a o2 fd 21 bf o1
o23o: dd 21 e1 87 o6 oo 4d fd o9 fd 7e oo dd 77 o4 fd
o24o: 21 bf o1 4c fd o9 fd 7e oo dd 77 o5 3e o1 32 dd
o25o: 87 c9 oo o6 oo dd 7e oo 4e b9 c8 o4 23 c3 58 o2
o26o: cb 27 cb 27 cb 27 cb 27 8o c9 47 cb 2f cb 2f cb
o27o: 2f cb 2f e6 of 67 78 e6 of 6f c9 c3 oo oo 3a dd
o28o: 87 fe o1 c2 8a oo cd 8c o2 c3 e3 o1 cd oc o2 23
o29o: c5 7d 54 47 cd 29 o2 dd 21 e1 87 dd 7e o4 dd 77
o2ao: oo dd 7e o5 dd 77 o1 7a 47 cd 29 o2 dd 21 e1 87
o2bo: dd 7e o4 dd 77 o2 dd 7e o5 dd 77 o3 c1 c9 3e oo
o2co: d3 22 21 eo 87 34 cd 97 oo 7b fe o2 c2 d7 o2 3e
o2do: o2 32 df 87 c3 be o2 fe o6 ca 19 o3 fe o4 ca 7b
o2eo: o2 oe o5 21 de 87 71 cd d3 oo 3e oo d3 22 21 eo
o2fo: 87 34 cd 97 oo oe o4 21 de 87 71 7b fe o2 c2 o9
o3oo: o3 3e o2 32 df 87 c3 ea o2 fe o4 ca 7b o2 fe o6
o31o: ca 19 o3 cd d3 oo c3 be o2 cd ef o1 cd oc o2 54
o32o: fd 21 e7 87 fd 23 fd 23 cd 1b o2 7c 62 77 3e o3
o33o: 11 de 87 12 cd 8c o2 cd ef o1 cd oc o2 cd 27 o2
o34o: c3 be o2 cd ef o1 cd oc o2 e9 cd ef o1 cd oc o2
o35o: 7e 32 ed 87 22 ee 87 3e e7 77 c3 8a oo 33 33 ed
o36o: 73 fc 87 3b 3b e5 f5 3a ed 87 2a ee 87 77 f1 e1
o37o: cd 94 o3 cd ec oo af 32 ee 87 32 ef 87 32 ed 87
o38o: c3 oo oo dd 21 e1 87 dd 36 oo 4c dd 36 o1 1c dd
o39o: 36 o2 44 c9 ed 43 f2 87 ed 53 f4 87 22 f6 87 dd
o3ao: 22 f8 87 fd 22 fa 87 32 fo 87 16 o7 dd 21 fo 87
o3bo: o1 d6 87 dd 6e oo dd 66 o1 dd e5 d5 cd 9o o2 af
o3co: 32 e5 87 d3 22 oa 32 e6 87 c5 cd 97 oo 7b fe o6
o3do: c2 ca o3 21 eo 87 34 c1 d1 dd e1 o3 dd 23 dd 23
o3eo: 15 c2 b3 o3 cd ec oo 3a fo 87 ed 4b f2 87 ed 5b
o3fo: f4 87 2a f6 87 dd 2a f8 87 fd 2a fa 87 c9 3e cf
o4oo: d3 o3 3e 7f d3 o3 cd 5o o4 7e 4f cd 1a o4 23 1b
o41o: 7b b2 2o f5 c3 oo oo oo oo oo f5 c5 e5 79 f5 cd
o42o: 27 o4 f1 e1 c1 f1 c9 37 o6 o9 f5 d4 3c o4 dc 41
o43o: o4 f1 1f 1o f5 cd 3c o4 cd 3c o4 c9 af d3 o1 18
o44o: o6 3e ff d3 o1 18 oo 21 16 oo 2b 7c b5 2o fb c9
o45o: 21 e1 87 af 77 23 77 23 77 23 77 23 3e af 77 23
o46o: 3e ef 77 cd bd oo af d3 22 cd 97 oo 7b fe o2 ca
o47o: 9o o4 fe o9 ca 7f o4 fe o6 ca 98 o4 c2 86 o4 21
o48o: eo 87 34 c3 66 o4 cd d3 oo 21 eo 87 34 c3 66 o4
o49o: 3e o2 32 df 87 c3 7f o4 cd ef o1 cd oc o2 eb d5
o4ao: 21 e1 87 af 77 23 77 23 77 23 77 23 3e af 77 23
o4bo: 3e 77 77 cd bd oo oo oo oo af d3 22 cd 97 oo
o4co: 7b fe o2 ca e7 o4 fe o6 ca d6 o4 fe o9 ca fo o4
o4do: oo oo oo c2 dd o4 21 eo 87 34 c3 ba o4 cd d3 oo
o4eo: 21 eo 87 34 c3 ba o4 3e o2 32 df 87 c3 d6 o4 oo
o4fo: cd ef o1 cd oc o2 d1 d5 ed 52 d1 eb c9 oo oo oo
o5oo: 3e cf d3 o3 3e ff d3 o3 oo cd 5o o4 cd 2o o5 77
o51o: 23 1b 7b b2 ca oo oo o1 of oo cd 44 o5 c3 oc o5
o52o: d9 db o1 17 3o fb o6 o8 11 ob oo cd 3e o5 11 15
o53o: oo cd 3e o5 db o1 17 cb 19 1o f3 79 d9 c9 1b 7b
o54o: b2 2o fb c9 ob 79 bo 2o fb c9 oo oo oo oo oo oo
```

Bild 4. Das Monitorprogramm ist nur knapp 1,5 KByte lang

Enter (EN)
Format: EN
Ausgabe: Inhalt der Speicherstelle nnnn+1 auf Ziffer 5 und 6. Speicherstelle nnnn+1 auf Ziffern 0...3. Abbruch durch Drücken der Break-Taste.

Insert (I)
Format: nnnnImmEN
(mm = Hexadezimalzahl)
Ausgabe: mm wird in die nnnn Speicherzelle übernommen. Anzeige der Speicherstelle nnnn+1 auf den Ziffern 0...3. Inhalt der Speicherstelle nnnn+1 auf Ziffer 5 und 6. Wenn keine Break-Taste gedrückt ist, erwartet der EMUF die Eingabe der nächsten Hexadezimalzahl für die angezeigte Speicherstelle.

Clear (CR)
Format: CR
Ausgabe: Display wird gelöscht.

Break (BK)
Format: BK
Ausgabe: EMUF meldet sich mit 'r u n'.

Go (G)
Format: nnnnG
Ausgabe: Programm wird bei Adresse nnnn gestartet.

Breakpoint (BP)
Format: nnnnBP
Ausgabe: Bei Adresse nnnn wird ein Breakpoint gesetzt (RST20). Beim Erreichen des Breakpoint wird eine Routine angesprungen, die die Register A, BC, DE, HL, IX, IY und SP anzeigt. Dabei erscheint auf Ziffer 6 eine Abkürzung für das entsprechende Register. Durch Drücken der Enter-Taste wird das nächste Register angezeigt. Nach dem letzten

Register wird der Breakpoint wieder gelöscht, und der Monitor meldet sich mit 'r u n'.

In (IN)
Serielle Eingabe von Daten über Bit 7 des PIO 0 Port B

Format: IN
Ausgabe: Auf Ziffer 4 und 5 erscheint AN. Monitor erwartet Anfangsadresse für die Ablage der aufzunehmenden Daten.

Format: nnnnEN
Ausgabe: Auf Ziffer 4 und 5 erscheint EN. Monitor erwartet Endadresse für die Ablage der aufzunehmenden Daten.
Format: nnnnG
Bis alle Daten eingelesen sind, erscheint auf Ziffer 5 ein E.
Voraussetzung für eine ordnungsgemäße Eingabe ist ein entsprechendes Programm beim die Daten sendenden Computer. Die Übertragungsrate beträgt etwa 2400 Baud.

Out (O)
Serielle Ausgabe von Daten über Bit 7 des PIO 0, Port B

Format: 0
Ausgabe: Analog zur Eingabe von Daten.

Erweiterungen sind möglich

Der Monitor belegt ungefähr 1,5 KByte des 2-KByte-EPROMs (2716). Es ist durchaus denkbar, die fünf noch freien Tasten mit weiteren Funktionen zu belegen. Aber auch auf der vorgestellten Stufe ermöglicht der Monitor zusammen mit der Tastatur und dem Display einen recht flexiblen Einsatz des Z80-EMUF.

Literatur

[1] *Kanis, Wolfgang:* Der Z80-EMUF. mc 1983, Heft 4, S. 112.

Joachim König

Z80-EMUF als universelle Fernbedienung

Fertig zu kaufende Fernsteuerungen sind zwar heute schon recht preiswert zu haben, lassen sich aber kaum an individuelle Bedürfnisse anpassen. Wir zeigen Ihnen, wie man mit modernen Bausteinen eine Fernsteuerung aufbaut, die – dank Z80-EMUF – auf Tastendruck auch komplexe Abläufe bewältigt.

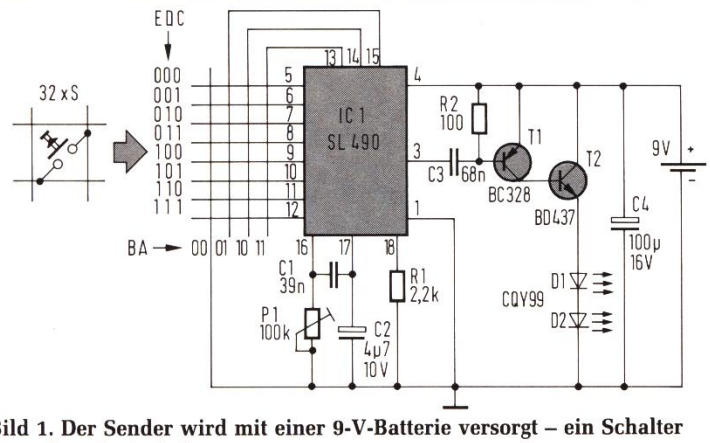


Bild 1. Der Sender wird mit einer 9-V-Batterie versorgt – ein Schalter ist nicht erforderlich

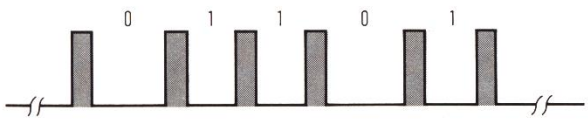


Bild 2. Das Datenwort besteht aus sechs kurzen Impulsen, deren Zwischenräume die Information darstellen. Ein kurzer Zwischenraum bedeutet „1", ein langer „0". Hier ist das Signal für das Datenwort 01101 (hex 0D) dargestellt

Das Kernstück des Senders ist die integrierte Schaltung SL 490 (*Bild 1*). In ihr ist die ganze Logik enthalten, die zum Betrieb des Senders nötig ist. Sie decodiert das Tastenfeld und erzeugt ein PPM-Signal (Puls-Pausen-Modulation).

Es können maximal 32 Tasten angeschlossen werden. Jede Taste entspricht einem fünfstelligen Binärcode. Drückt man eine Taste, so erscheinen am Ausgang des SL 490 sechs kurze Impulse.

Die Pause zwischen diesen Impulsen enthält die Information. Eine kurze Pause bedeutet, „logisch 1", eine lange „logisch 0" (*Bild 2*). Die beiden als Darlington geschalteten Transistoren T1 und T2 dienen als Verstärker und steuern direkt die Infrarotsender D1 und D2. Der SL 490 besitzt eine Abschaltautomatik, so daß nur wenige µA Strom fließen, wenn keine Taste gedrückt ist. Deswegen ist kein Schalter in der Stromversorgung nötig. Als Stromversorgung genügt eine einfache 9-V-Batterie. Mit P1 kann man die Zeit zwischen den Impulsen justieren. Das Verhältnis zwischen „1" und „0" beträgt 1:1,5.

Der Empfänger

Das mit der Empfängerdiode D1 gewonnene Signal wird mit dem Operationsverstärker SL 480 verstärkt (*Bild 3*).

Dann wird es über T2 dem Schmitt-Trigger IC2 zugeleitet. Der Schmitt-Trigger macht saubere Rechteckimpulse aus dem Signal, und das Monoflop IC3 verlängert diese Impulse auf 100 µs. Da das Monoflop nicht nachgetriggert werden kann, werden Störimpulse ausgeblendet. Am Ausgang von IC3 steht eine Impulsfolge, wie in Bild 2 gezeigt, bereit. Diese Impulse werden an Port A, Bit 0 geführt. Mit dem Potentiometer P1 kann die Empfindlichkeit eingestellt werden.

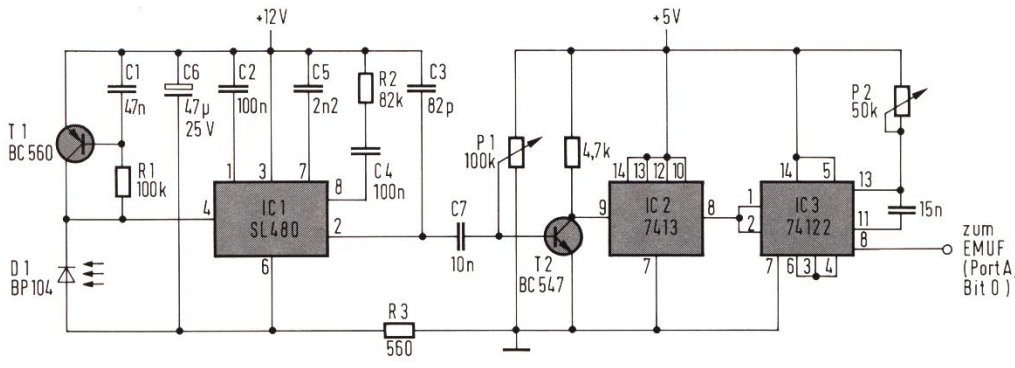


Bild 3. Der Empfänger liefert am Ausgang saubere Signale gemäß Bild 2. Störimpulse unterdrückt er

```
                        .z80
0002                    portic  equ     02h             ; port 1 control
0000                    portid  equ     00h             ; port 1 data
0000'   31 87FF         start:  ld      sp,087ffh
0003'   21 0147'                ld      hl,inittab      ; ports initialisieren
0006'   06 04                   ld      b,4             ; anzahl der ports
0008'   4E              st1:    ld      c,(hl)          ; hole portadresse (control)
0009'   23                      inc     hl
000A'   7E                      ld      a,(hl)
000B'   ED 79                   out     (c),a           ; setze mode
000D'   23                      inc     hl
000E'   7E                      ld      a,(hl)
000F'   ED 79                   out     (c),a           ; bestimme input und output bits
0011'   3E 03                   ld      a,3
0013'   ED 79                   out     (c),a           ; disable interupts
0015'   0D                      dec     c
0016'   0D                      dec     c               ; mache datenadressse
0017'   3E 00                   ld      a,0
0019'   ED 79                   out     (c),a           ; setze ausgaenge auf 0
001B'   23                      inc     hl
001C'   10 EA                   djnz    st1
001E'   3E 00                   ld      a,0
0020'   32 0004"                ld      (port1m),a      ; reset memoryimage
0023'   32 0005"                ld      (port2m),a      ; reset memoryimage
0026'   32 0006"                ld      (port3m),a      ; reset memoryimage
0029'   32 0007"                ld      (port4m),a      ; reset memoryimage
002C'   21 0193'                ld      hl,lampe
002F'   22 0000"                ld      (table),hl      ; kontext = lampe
0032'   CD 00A9'        main1:  call    getword         ; hole nummer
0035'   1E 00           main:   ld      e,0             ; clear zaehler
0037'   50                      ld      d,b             ; save nummer
0038'   1C                      inc     e               ; erhoehe zaehler
0039'   CD 00A9'        m1:     call    getword         ; hole naechste nummer
003C'   7A                      ld      a,d
003D'   B8                      cp      b               ; neue nummer gleich der ersten
003E'   C2 0035'                jp      nz,main         ; wenn ungleich dann beginne neu
0041'   1C                      inc     e               ; erhoehe zaehler
0042'   7B                      ld      a,e
0043'   FE 05                   cp      5               ; 5 gleiche nummern eingetroffen?
0045'   C2 0039'                jp      nz,m1           ; wenn nicht hole neachste nummer
0048'   CD 004E'                call    schalt          ; fuehre funktion aus
004B'   C3 0032'                jp      main1
004E'   CD 007A'        schalt: call    convert         ; wandle physische nummer in logische um
0051'   7B                      ld      a,e
0052'   FE 10                   cp      16              ; if nummer>16 goto switch
0054'   D2 0064'                jp      nc,switch
0057'   2A 0000"                ld      hl,(table)      ; get basis fuer verzweigung
005A'   CB 23                   sla     e               ; e = e * 2
005C'   16 00                   ld      d,0
005E'   19                      add     hl,de           ; hl = basis + index
005F'   5E                      ld      e,(hl)          ; hole verzweigungadresse
0060'   23                      inc     hl
0061'   56                      ld      d,(hl)
0062'   EB                      ex      de,hl
0063'   E9                      jp      (hl)            ; springe zur routine
0064'   21 0173'        switch: ld      hl,basis        ; basisadressse der kontexttabellenadress
0067'   7B                      ld      a,e             ; a = tastennummer
0068'   E6 0F                   and     0fh             ; a = tastennummer-16
006A'   5F                      ld      e,a
006B'   CB 23                   sla     e
006D'   16 00                   ld      d,0             ; de = tastennummer*2
006F'   19                      add     hl,de           ; hl = basis+tastenummer*2
0070'   5E                      ld      e,(hl)          ; get low byte
0071'   23                      inc     hl
0072'   56                      ld      d,(hl)          ; get high byte
0073'   ED 53 0000"            ld      (table),de      ; set new kontextbasis
0077'   C3 0083'                jp      endret
007A'   21 0153'        convert:ld      hl,contab       ; adresse der converttabelle
007D'   16 00                   ld      d,0
007F'   58                      ld      e,b             ; pyhsische tastennummer nach e
0080'   19                      add     hl,de           ; hl = contab + physische tastennummer
0081'   5E                      ld      e,(hl)          ; log tastennummer nach e
0082'   C9                      ret
0083'   21 01F4        endret: ld      hl,0500         ; startwert fuer time-out counter
0086'   50                      ld      d,b             ; save taste
0087'   CD 0091'      er1:    call    getwo1          ; hole naechste taste
008A'   C8                      ret     z               ; time-out
008B'   7A                      ld      a,d
008C'   B8                      cp      b               ; gleiche taste?
008D'   C0                      ret     nz              ; andere taste gibt auch time-out
008E'   C3 0083'                jp      endret
0091'   E5              getwo1: push    hl
0092'   21 0091'                ld      hl,getwo1
0095'   22 0002"                ld      (retadr),hl     ; setze return adresse fuer fehlerfall
0098'   E1                      pop     hl
0099'   06 00                   ld      b,0             ; clear nummer
009B'   DB 00           g10:    in      a,(port1d)      ; warte auf eine 1
009D'   E6 01                   and     1
009F'   C2 00B8'                jp      nz,getw2
00A2'   2B                      dec     hl
00A3'   7C                      ld      a,h
00A4'   B5                      or      l
00A5'   C8                      ret     z               ; time-out
00A6'   C3 009B'                jp      g10
00A9'   21 00A9'        getword:ld      hl,getword
00AC'   22 0002"                ld      (retadr),hl     ; setze returnadresse fuer fehlerfall
00AF'   06 00                   ld      b,0             ; clear nummer
00B1'   DB 00           getw1:  in      a,(port1d)      ; warte auf 1
00B3'   E6 01                   and     1               ; check ob bit = 1
00B5'   CA 00B1'                jp      z,getw1
00B8'   CD 00D2'        getw2:  call getbit             ; hole 1.bit
00BB'   CB 10                   rl b                    ; schiebe bit nach b
00BD'   CD 00D2'                call getbit
00C0'   CB 10                   rl b                    ; 2.bit
00C2'   CD 00D2'                call getbit
00C5'   CB 10                   rl b                    ; 3.bit
00C7'   CD 00D2'                call getbit
00CA'   CB 10                   rl b                    ; 4.bit
00CC'   CD 00D2'                call getbit
00CF'   CB 10                   rl b                    ; 5.bit
00D1'   C9                      ret
00D2'   E5              getbit: push    hl
00D3'   C5                      push    bc
00D4'   01 01F4                 ld      bc,500          ; init timeout counter
00D7'   21 0000                 ld      hl,0            ; clear counter
00DA'   DB 00           g1:     in      a,(port1d)      ; warte auf low flanke
00DC'   E6 01                   and     1
00DE'   C2 00DA'                jp      nz,g1
00E1'   DB 00           getb3:  in      a,(port1d)      ; warte auf high flanke
00E3'   E6 01                   and     1
00E5'   C2 00F2'                jp      nz,getb2
00E8'   23                      inc     hl              ; erhoehe zaehler
00E9'   0B                      dec     bc              ; decrement time out counter
00EA'   78                      ld      a,b             ; wenn timeout counter = 0
00EB'   B1                      or      c               ; dann springe auf fehlerroutine
```

Bild 4. Das Auswerteprogramm: Wer keinen Linker besitzt, muß die mit doppelten Anführungszeichen gekennzeichneten Adressen von Hand verändern

```
00EC' CA 011A'          jp      z,getb7
00EF' C3 00E1'          jp      getb3
00F2' 7D                LD      A,l             ; if l in [130..200]
00F3'           getb2:  LD      200+1           ; dann ist es eine 0
00F5' FE C9             CP      NC,.L1
00F7' D2 0100'          JP      130
00FA' FE 82             CP      C,.L1
00FD' DA 0100'          JP      getb4
0100' C3 0115'  .L2:    jp      getb4
0103' 7D        .L1:    ld      a,l             ; if l in [080..130]
0104' FE 83             CP      130+1           ; dann ist es eine 1
0106' D2 010E'          JP      NC,.L3
0109' FE 50             CP      080
010B' DA 010E'          JP      C,.L3
010E' C3 0111'  .L4:    jp      getb6
0111' C3 011A'  .L3:    jp      getb7           ; error in bitlaenge
0114' 37        getb6:  scf                     ; es war eine 1
0115' C1                pop     bc
0116' E1                pop     hl
0117' C9                ret
0118' C1        getb7:  pop     bc
0119' E1                pop     hl
011A' FD E1             pop     iy              ; dummy pop der ret addresse
011C' 2A 0002"          ld      hl,(retadr)
011F' E9                jp      (hl)            ; error return
                ; Routinen fuer lampe
                ; Beispiel fuer die Realisation einer TASTE
0122' 3A 0004"  lampe1: ld      a,(portim)      ; get memoryimage
0125' CB CF             set     1,a             ; set bit-1
0127' D3 00             out     (portid),a      ;
0129' 32 0004"          ld      (portim),a
012C' CD 0083'          call    endret          ; wait auf ende tastendruck
012F' 3A 0004"          ld      a,(portim)
0132' CB 8F             res     1,a             ; reset bit-1
0134' D3 00             out     (portid),a      ;
0136' 32 0004"          ld      (portim),a
0139' C9                ret
                ; Beispiel fuer die Realisatin eines SCHALTERS
013A' 3A 0004"  lampe2: ld      a,(portim)      ; get memoryimage
013D' EE 04                     xor     4h              ; toggle bit-2
013F' D3 00                     out     (portid),a
0141' 32 0004"                  ld      (portim),a      ; set memoryimage
0144' C3 0083'                  jp      endret          ; warte auf ende tastendruck und ret nach main
                ; daten im eprom
0147' 03 FF 01 13       inittab:db      3,0ffh,01,13h,0ffh,0,2,0ffh,0,12h,0ffh,0
014B' FF 00 02 FF
014F' 00 12 FF 00
                ; umrechnungtabelle pyhsische taste im log taste
0153' 00 01 02 03       contab: db      0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21, 22,23
0157' 04 05 06 07
015B' 08 09 0A 0B
015F' 0C 0D 0E 0F
0163' 10 11 12 13
0167' 14 15 16 17
016B' 18 19 1A 1B               db      24,25,26,27,28,29,30,31
016F' 1C 1D 1E 1F
0173' 0193' 0193'       basis:  dw      lampe,lampe,lampe,lampe,lampe,lampe,lampe,lampe
0177' 0193' 0193'
017B' 0193' 0193'
017F' 0193' 0193'               dw      lampe,lampe,lampe,lampe,lampe,lampe,lampe,lampe
0183' 0193' 0193'
0187' 0193' 0193'
018B' 0193' 0193'       ; kontextadressen fuer lampen
018F' 0193' 0193'       lampe:  dw      lampe1,lampe2,endret,endret,endret,endret,endret,endret
0193' 0122' 013A'               dw      lampe1,lampe2,endret,endret,endret,endret,endret,endret
0197' 0083' 0083'
019B' 0083' 0083'
019F' 0083' 0083'
01A3' 0083' 0083'               dw      endret,endret,endret,endret,endret,endret,endret,endret
01A7' 0083' 0083'
01AB' 0083' 0083'
01AF' 0083' 0083'
01B3'                   dseg
                        ; daten im ram
0000"                   table:  ds      2
0002"                   retadr: ds      2
0004"                   portim: ds      1       ; memory abbildung der ports
0005"                   port2m: ds      1
0006"                   port3m: ds      1
0007"                   port4m: ds      1
                                end
```

Mit P2 wird die Impulsbreite auf 100 µs eingestellt. Wer kein Oszilloskop besitzt, stellt P2 in Mittenstellung. P1 muß so eingestellt werden, daß in Ruhestellung (keine Impulse vom Sender) Low-Pegel am Eingang von IC2 anliegt.

Das Programm

Bei der Entwicklung des Programmes (*Bild 4*) wurde auf eine möglichst einfache Erweiterungsmöglichkeit der Funktion Wert gelegt. Die Software stellt nur die Auswertung der empfangenen Impulsfolgen und die davon abhängigen Verzweigungen dar. Die Routinen, die nötig sind, um die einzelnen Funktionen auszuführen, sind abhängig von der jeweiligen Funktion. Als Beispiel sind jedoch zwei Routinen abgedruckt, die eine Taste bzw. einen Schalter emulieren. Der Programmteil ENDRET wartet auf das Loslassen der gedrückten Taste, so daß im Programm eine Funktion nicht mehrmals ausgeführt wird, wenn man die Taste festhält.

Von den 32 zur Verfügung stehenden Tastencodes werden 16 benutzt, um die Aufgabe der Fernbedienung umzuschalten. So kann man 16 Aufgaben mit jeweils 16 Funktionen, also 256 Funktionen, ausführen. In der Tabelle BASIS stehen die 16 Adressen der jeweiligen Funktionsbasisadressen.

Ein Beispiel:
Wenn in der Variablen TABLE die Adresse der Tabelle LAMPE steht, werden beim Druck auf eine der Tasten 0...15 die jeweiligen Adressen aus der Tabelle LAMPE geholt, und das Programm wird dort ausgeführt. Eine Erweiterung der Funktionen ist also dadurch möglich, daß man die Adresse der neuen Funktion in die entsprechende Tabelle einbaut, ohne das Kernprogramm zu ändern.

Die Routine CONVERT dient zum Umwandeln des physikalischen Tastencodes in den logischen. Je nach Verdrahtung des Tastenfeldes ergeben sich die übertragenen Tastencodes. Um nun die Codes in eine einheitliche Reihenfolge zu bringen, müssen sie umgewandelt werden. Das abgedruckte Programm nimmt eine 1:1-Umwandlung vor.

Wem die 256 Funktionen zu wenig sind, der kann das Programm auch so umschreiben, daß eine der 32 Tasten die Umschaltung einleitet und die nächste Taste die Aufgabe festlegt. Dann sind insgesamt 992 Funktionen möglich.

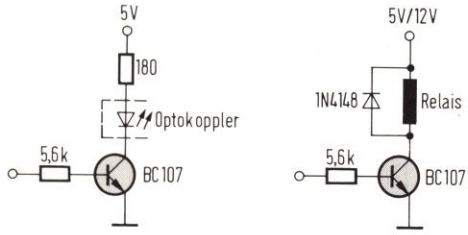


Bild 5. So lassen sich angesteuerte Geräte oder Schaltungen galvanisch vom EMUF-Ausgang trennen

In *Bild 5* sind einige Möglichkeiten der Steuerung gezeigt. Man kann mit den PIOs über einen Transistor direkt ein Relais ansteuern und damit 220-V-Geräte bedienen. Wichtig: Das gesteuerte Gerät und der EMUF müssen galvanisch getrennt sein. Falls die Reichweite der Fernbedienung (ca. 6 m) unbefriedigend ist, kann man die Sendedioden in Reflektoren setzen.

Linken des Programmes

Das Listing ist so geschrieben, daß die Adressen des Programmes erst beim Linken festgelegt werden. Dazu ist ein Linker nötig, der das Programm nicht ablauffähig in den Speicher legt. Der Linker von Pascal MT+ ist dazu in der Lage. Um das Objektfile damit zu linken, bekommt es den Namen FERNB.ERL; dann lautet der Aufruf LINK FERNB/P:0/D:8000. Damit wird das Programm auf Adresse 0 gelegt, die Daten auf Adresse 8000h. Wer keinen Linker besitzt, muß alle Adressen, die mit einem doppelten Anführungszeichen gekennzeichnet sind, entsprechend verändern.

Literatur

[1] Remote Control Data, Plessey Semiconductors.
[2] IR-Fernsteuerung. Elektor, Heft 139, Seite 97.

Erich Gaulke

Z80-EMUF als Spooler

Manche Ausgabegeräte, wie z. B. Typenraddrucker, sind recht langsam und halten den Computer unnötig auf. Ein Spooler (simultaneous peripheral output on-line) kann den auszugebenden Text schnell vom Computer aufnehmen, zwischenspeichern und mit der für den Drucker erforderlichen geringeren Geschwindigkeit weitergeben. Währenddessen kann sich der Computer schon wieder anderen Aufgaben zuwenden.

Ein Spooler ist, wie bereits in [1] beschrieben, im wesentlichen ein Speicher nach dem FIFO-Prinzip (first in, first out). Ein übliches RAM muß also im Sinne eines FIFOs verwaltet werden. Mit Hilfe des Z80-EMUF läßt sich das einfach und zugleich effizient erledigen und wurde hier für eine Centronics-Schnittstelle realisiert. Wenn man den Spooler im Gegensatz zu [1] als eigenständiges Mikrocomputersystem aufbaut, hat das drei Vorteile:

1. Unabhängig von der Druckertreiberroutine bzw. vom Anwenderprogramm (Textverarbeitung, Basic, Assembler, DOS usw.) bedient der Spooler den Drucker.
2. Der Speicherplatz des Hauptrechners wird nicht durch den Spooler belastet.
3. Durch Interruptsteuerung des Programms ist der Z80-EMUF mit der Druckersteuerung kaum ausgelastet und könnte daher auch noch viele andere Aufgaben als Co-Prozessor übernehmen.

Beschaltung des Z80-EMUF

Die PIOs [2] des Z80-EMUF [3] lassen sich im Input- bzw. Output-Mode betreiben. Dann stehen neben den acht Ein- bzw. Ausgabeleitungen zwei für die Abwicklung des Handshakes zur Verfügung. Damit entfällt die Abwicklung der Datenübergabe vom Prozessor aus; er wird für andere Aufgaben freigestellt. Allerdings sind die Handshake-Leitungen den Anforderungen der jeweiligen Druckerschnittstelle anzupassen. Im vorliegendem Fall galt es, den Druckerport eines TRS-80 und eine Typenradschreibmaschine Olympia ES 100 anzuschließen. *Bild 1* zeigt die Beschaltung: Außer den Invertern zur Anpassung der Busy- und Ready-Signale wird ein Monoflop benötigt, um die Daten auszugeben. Als Gatter-Logik kommen vorzugsweise CMOS-ICs in Betracht, da Gatterlaufzeiten hier nicht kritisch sind.

Das Spoolerprogramm

Bei der Entwicklung des Programms wurde auf einen möglichst effizienten Umgang mit Rechenzeit und Speicherplatzbedarf geachtet. Für die Realisierung des FIFOs (2036 Byte Kapazität) wird mit nur zwei 16-Bit-Zeigern in DE und HL gearbeitet; die weiteren Informationen über den Zustand des Systems sind platzsparend und mit kleiner Zugriffszeit als Flags im C-Register untergebracht. Durch die Nutzung der Vektor-Interrupts konnte vollständig auf die Verwendung von Warteschleifen verzichtet werden. Damit benötigt das System für die Bearbeitung der Ein- oder

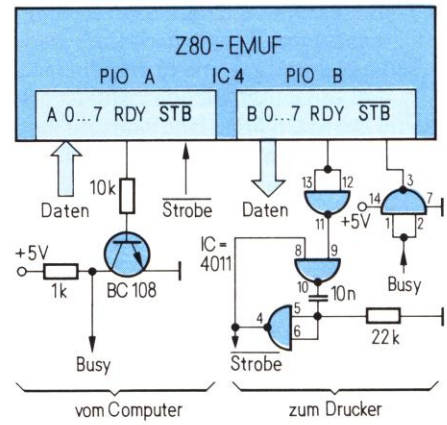


Bild 1. Beschaltung der Z80-PIO zur Verbindung mit Computer und Drucker am Beispiel des TRS-80

Ausgabe eines Zeichens unabhängig vom Tempo der angeschlossenen Geräte nur noch 100 µs bei einer Taktfrequenz von 2,5 MHz. Das führt bei Verwendung eines schnellen Matrixdruckers mit 100 Zeichen pro Sekunde zu einer Auslastung des Z80-EMUF von nur 2 %! Er könnte also noch viele weitere Aufgaben z. B. eines Kommunikationsrechners übernehmen.

Das vollständige Assembler-Listing ist in *Bild 2* abgedruckt: *Tabelle 1* zeigt die Steuerflags in übersichtlicher Darstellung. Nach dem Power-On-Reset beginnt der Prozessor bei Speicherstelle 0000 in dem ROM. Zunächst werden sämtliche PORTs mit den Daten aus der Tabelle TAB1 und unter Nutzung des Blockausgabebefehls OTIR initialisiert. Dann werden die Anfangswerte der Zeiger und Flags im C-Register geladen. Der nicht maskierbare Interrupt (NMI) am Prozessor gestattet es, den Druckvorgang zu unterbrechen und die Anfangswerte neu einzustellen.

Infolge der Interruptsteuerung besteht das Hauptprogramm HAUP aus einer leeren Schleife; hier ist noch Platz für Erweiterungen. Da das System nur durch Interrupts angestoßen wird, jedoch kein periodisches Interruptsignal

Tabelle: Die Steuerflags im C-Register

	Fall	EIN	AUS	Fertig	Zeichen da
Bit-Nr.	0	1	2	3	4
Bit = 1	a	Enable	Enable	ja	ja
Bit = 0	b	Disable	Disable	nein	nein
Anfangs-stellung	1	1	0	1	0

```
0000            00030           ORG     0
0000 C30001     00040           JP      INIT
0008            00050           ORG     08H     ;ANFANG DER INTERRUPT-
0008 1F01       00060           DEFW    EIN     ;VEKTOREN
000A 6F01       00070           DEFW    AUS
0066            00080           ORG     66H     ;START NICHT MASKIERBARER
0066 C30001     00090           JP      INIT    ;INTERRUPT
                00100 ;
0010            00110 PI10      EQU     10H
8000            00120 ANF       EQU     8000H   ;RAM-ANFANG
87F3            00130 END       EQU     87FFH-12        ;PLATZ FUER STACK (=12)
                00140 ;
0100            00150           ORG     100H
0100 21C901     00160 INIT      LD      HL,TAB1 ;PORTS INITIALISIEREN
0103 4E         00170 INIA      LD      C,(HL)  ;PORT-ADRESSE LADEN
0104 23         00180           INC     HL
0105 46         00190           LD      B,(HL)  ;ZAHL DER BYTES LADEN
0106 23         00200           INC     HL
0107 EDB3       00210           OTIR            ;BYTES AUF PORT AUSGEBEN
0109 7E         00220           LD      A,(HL)
010A B7         00230           OR      A       ;TABELLENENDE WIRD DURCH NULL
010B 20F6       00240           JR      NZ,INIA ;ERKANNT
010D 31FF87     00250           LD      SP,87FFH
0110 FB         00260           EI
0111 ED5E       00270           IM      2
0113 0E0B       00280           LD      C,11    ;INITIALISIERUNG DER FLAGS
0115 210080     00290           LD      HL,ANF  ;INITIALISIERUNG DER ZEIGER
0118 110080     00300           LD      DE,ANF
011B DB10       00310           IN      A,(PI10)  ;EING.-INT. ERMOEGLICHEN
011D 18FE       00320 HAUP      JR      HAUP    ;LEERES HAUPTPROGRAMM
                00330 ;
011F CB49       00350 EIN       BIT     1,C     ;EIN DISABLE ?
0121 280D       00360           JR      Z,EINE
0123 CD3501     00370           CALL    EIN1    ;ZEICHEN LESEN
0126 CB59       00380           BIT     3,C     ;FERTIG FLAG ?
0128 C48501     00390           CALL    NZ,AUS1 ;ZEICHEN AUSGEBEN WENN GESETZT
012B CB99       00400           RES     3,C     ;FERTIG FLAG ZURUECKSETZEN
012D FB         00410           EI              ;INTERRUPTS WIEDER ERMOEGLICHEN
012E ED4D       00420           RETI
0130 CBE1       00430 EINE      SET     4,C     ;ZEICHEN DA FLAG SETZEN
0132 FB         00440           EI
0133 ED4D       00450           RETI
0135 CD3C01     00460 EIN1      CALL    ABF1    ;ZEIGER U. FLAGS VERAENDERN
0138 CDBF01     00470           CALL    EIN3    ;EIN ZEICHEN LESEN
013B C9         00480           RET
013C CBD1       00490 ABF1      SET     2,C     ;ENABLE AUSGANG
013E 37         00500           SCF
013F CB41       00510           BIT     0,C     ;FALL A) ODER B)
0141 2822       00520           JR      Z,BFAL
0143 C5         00530           PUSH    BC
0144 E5         00540           PUSH    HL
0145 01F387     00550           LD      BC,END
0148 ED42       00560           SBC     HL,BC   ;HL-1=END ?
014A E1         00570           POP     HL
014B C1         00580           POP     BC
014C C0         00590           RET     NZ      ;ZURUECK WENN ENDE NICHT ERREICHT
014D CB81       00600           RES     0,C     ;FALL B) EIN
014F B7         00610           OR      A
0150 210080     00620           LD      HL,ANF
0153 ED52       00630           SBC     HL,DE   ;DE=ANF ?
0155 2002       00640           JR      NZ,BFA1
0157 CB89       00650           RES     1,C     ;DISABLE EIN, WENN DE=ANF
0159 21F487     00660 BFA1      LD      HL,END+1    ;LETZTES ZEICHEN IN END+1
015C CDBF01     00670           CALL    EIN3        ;ABLEGEN
015F 210080     00680           LD      HL,ANF      ;ZEIGER AUF ANFANG SETZEN
0162 DDE1       00690           POP     IX      ;ZURUECK ZUR INTERRUPT-ROUTINE
0164 C9         00700           RET
0165 EB         00710 BFAL      EX      DE,HL
0166 E5         00720           PUSH    HL
0167 ED52       00730           SBC     HL,DE   ;HL=DE-1 ?
0169 E1         00740           POP     HL
016A EB         00750           EX      DE,HL
016B C0         00760           RET     NZ      ;ZURUECK, WENN PLAETZE FREI
016C CB89       00770           RES     1,C     ;SONST DISABLE EIN
016E C9         00780           RET
                00790 ;
016F CB51       00800 AUS       BIT     2,C     ;AUS DISABLE ?
0171 280D       00810           JR      Z,AUSE
0173 CD8501     00820           CALL    AUS1    ;EIN ZEICHEN AUSGEBEN
0176 CB61       00830           BIT     4,C     ;ZEICHEN-DA-FLAG ?
0178 C43501     00840           CALL    NZ,EIN1 ;ZEICHEN LESEN, WENN GESETZT
017B CBA1       00850           RES     4,C     ;ZEICHEN-DA-FLAG ZURUECKSETZEN
017D FB         00860           EI
017E ED4D       00870           RETI
0180 CBD9       00880 AUSE      SET     3,C     ;FERTIG FLAG SETZEN
0182 FB         00890           EI
0183 ED4D       00900           RETI
0185 CD8C01     00910 AUS1      CALL    ABF2    ;FLAGS U. ZEIGER VERAENDERN
0188 CDC401     00920           CALL    AUS2    ;EIN ZEICHEN AUSGEBEN
018B C9         00930           RET
018C CBC9       00940 ABF2      SET     1,C     ;ENABLE EIN
018E 37         00950           SCF
018F CB41       00960           BIT     0,C     ;FALL A) ODER B)
0191 2808       00970           JR      Z,BFBL
0193 E5         00980           PUSH    HL
0194 ED52       00990           SBC     HL,DE   ;DE=HL-1 ?
0196 E1         01000           POP     HL
0197 C0         01010           RET     NZ      ;ZURUECK, WENN PLAETZE FREI
0198 CB91       01020           RES     2,C     ;DISABLE AUS
019A C9         01030           RET
019B D5         01040 BFBL      PUSH    DE      ;FALL B)
019C D9         01050           EXX
019D E1         01060           POP     HL
019E 01F387     01070           LD      BC,END
01A1 ED42       01080           SBC     HL,BC   ;DE-1=END ?
01A3 D9         01090           EXX
01A4 C0         01100           RET     NZ      ;ZURUECK, WENN ENDE NICHT ERREICHT
01A5 CBC1       01110           SET     0,C     ;FALL A) EIN
01A7 E5         01120           PUSH    HL
01A8 B7         01130           OR      A
01A9 110080     01140           LD      DE,ANF
01AC ED52       01150           SBC     HL,DE   ;HL=ANF ?
01AE 2002       01160           JR      NZ,BFB1
01B0 CB91       01170           RES     2,C     ;DISABLE AUS, WENN NICHTS FREI
01B2 11F487     01180 BFB1      LD      DE,END+1        ;ZEICHEN BEI END+1
01B5 CDC401     01190           CALL    AUS2            ;AUSGEBEN
01B8 110080     01200           LD      DE,ANF          ;ZEIGER AUF ANF SETZEN
01BB E1         01210           POP     HL
01BC DDE1       01220           POP     IX      ;ZURUECK ZUR INTERRUPT-ROUTINE
01BE C9         01230           RET
                01240 ;
01BF DB10       01250 EIN3      IN      A,(PI10)        ; PORT LESEN
01C1 77         01260           LD      (HL),A          ;IN BUFFER SPEICHERN
01C2 23         01270           INC     HL              ;ZEIGER WEITERZAEHLEN
01C3 C9         01280           RET
01C4 1A         01290 AUS2      LD      A,(DE)          ;ZEICHEN AUS BUFFER
01C5 D311       01300           OUT     (PI10+1),A      ;ZEICHEN AUSGEBEN
01C7 13         01310           INC     DE              ;ZAEHLER WEITER
01C8 C9         01320           RET
01C9 12         01330 TAB1      DEFB    PI10+2  ;INITIALISIERUNGSTABELLE
01CA 03         01340           DEFB    3       ;FUER DIE PORTS
01CB 08         01350           DEFB    8
01CC 4F         01360           DEFB    4FH
01CD 87         01370           DEFB    87H
01CE 13         01380           DEFB    PI10+3
01CF 03         01390           DEFB    3
01D0 0A         01400           DEFB    0AH
01D1 0F         01410           DEFB    0FH
01D2 87         01420           DEFB    87H
01D3 00         01430 ZEND      DEFB    0       ;ENDE WIRD DURCH NULL ERKANNT
0000            01440           END
```

Bild 2. Das vollständige Assembler-Listing mit Kommentaren

generiert wird, ist durch sorgfältige Programmierung dafür zu sorgen, daß es nicht stehenbleibt. Die Interrupt-Service-Routinen sind symmetrisch für Ein- und Ausgabe und werden von den jeweils zuständigen PIOs gestartet. Nachdem ein Zeichen in den PIO eingelesen ist, beginnt das Programm bei EIN.
Wenn noch Plätze im Puffer frei sind, gilt EIN-enable. In EIN1 wird ein Zeichen eingelesen und abgespeichert, die Zeiger und Flags werden verwaltet. Das FERTIG-Flag ist nur gesetzt, wenn der Drucker sämtliche Zeichen im Buffer bereits ausgegeben hat und ein AUS-Interrupt kein weiteres Zeichen mehr vorfand. Der AUS-Kanal bleibt dann vollständig abgeschaltet, bis er nach Eingabe eines neuen Zeichens und Aufrufen von AUS1 in EIN vermittels des FERTIG-Flags reaktiviert wird.
Entsprechend wird bei vollem Puffer zunächst EIN-disable gesetzt und beim nächsten Interrupt auf der Eingangsseite, ohne daß inzwischen Platz frei wurde, das Zeichen-da-Flag gesetzt. Dadurch ist der Eingangskanal vollständig abgeschaltet, bis er nach Ausgabe eines Zeichens reaktiviert wird.

Erweiterungen möglich

Da das System nicht ausgelastet ist, bietet es sich an, Erweiterungen anzubringen. Dabei wäre z. B. denkbar, Umcodierungen der zu druckenden Zeichen vorzunehmen, Umlaute darzustellen, Matrix-Drucker mit Einzelpunktansteuerung zum Zeichnen von Kurven mit einem leistungsfähigen Befehlssatz zu versehen, Aufgaben eines Kommunikationsrechners in einem komplexen System zu übernehmen (einfaches Beispiel: bidirektionales Interface mit einer Schreibmaschine), oder an den zweiten PIO eine Tastatur mit Sonderfunktionen anzubringen.
(Herrn Hermann Wacker aus Braunschweig sei für Tips gedankt, die die schnelle Realisierung des Projekts ermöglichten.)

Literatur

[1] *Breymann, U.*: Druckerausgabe nebenbei. mc 1982, Heft 6. Seite 64

[2] Z80-PIO, Product Specification. Zilog

[3] *Kanis, W.*: Der Z80-EMUF. mc 1983, Heft 4, Seite 112

Dr. Sieghard Schicktanz

Multitasking mit dem Z80-EMUF

Mit einigen Einschränkungen im Komfort läßt sich ein Systemkern für Multitasking bereits mit einem Einplatinencomputer minimalen Ausbaus realisieren, wie ihn z. B. der Z80-EMUF darstellt. Damit kann ein vollständiges Anwendungsprogramm in einem EPROM-Bereich von etwa 2 KByte Platz finden. Der Beschreibung des Systemkerns ist eine kurze Besprechung der Grundlagen für ein Multitaskingsystem vorangestellt.

Multitasking, gelegentlich – nicht ganz korrekt – auch Mehrprogrammbetrieb genannt, wird häufig in der Prozeßsteuerungstechnik angewandt. Man versteht darunter die Bearbeitung einer komplexen Aufgabe durch mehrere, quasi parallel ablaufende Teilprogramme, die weitgehend unabhängig voneinander jeweils einen überschaubaren Teil der Gesamtaufgabe ausführen.
Durch die Aufteilung erreicht man eine wesentliche Vereinfachung der Programmerstellung. Eine geschickte Verteilung der Aufgaben ermöglicht es, die Teilprogramme sehr einfach aufzubauen und – wichtig bei industriellem Einsatz – unabhängig voneinander zu erstellen, da die Schnittstellen einfach gehalten werden können.
Außerdem können alle Koordinationsaufgaben in einem zentralen Teilprogramm (Scheduler) zusammengefaßt werden, das unabhängig vom jeweiligen Einsatzfall ausgeführt ist und darüber hinaus weitgehend die Unabhängigkeit der einzelnen anwendungsbezogenen Teilprogramme (Tasks) unterstützt. Der Systemkern stellt alle benötigten Hilfsroutinen zur Verfügung, die zum (quasi) parallelen Bearbeiten der Teilprogramme gebraucht werden, ohne daß der Programmersteller dies besonders berücksichtigen muß.
Bei großen Prozeßrechensystemen stellt diese Funktionen bereits das dort eingesetzte Betriebssystem zur Verfügung. Der Scheduler macht allerdings nur einen kleinen Teil des Betriebssystems aus; mit einigen Einschränkungen im Komfort kann ein Multitaskingsystem, samt Anwendungsprogramm, bereits auf einem Minimalsystem wie dem Z80-EMUF [1] in nur 2 KByte EPROM (und 2 KByte RAM) realisiert werden.

Was ist Multitasking?

Die Grundfunktionen eines Multitaskingsystems sind im folgenden kurz erläutert. Jede Task, d. i. jedes Teilprogramm, das eine abgeschlossene Teilaufgabe bearbeitet, kann verschiedene Zustände einnehmen:

1. Nicht vorhanden
2. Angehalten
3. Bereit (aktiv)
4. Wartend

Der Zustand 1, nicht vorhanden, ist hier trivial, da in einem EPROM-System Tasks nicht neu hinzukommen oder gelöscht werden können. In diesem Fall sind alle möglichen Tasks also immer im System vorhanden, der Zustand 1 kann nicht auftreten.
Eine Task, die den Zustand 3, bereit, hat, wird vom Scheduler in Bearbeitung genommen, sobald dies möglich ist. Im einfachsten Fall wird die Bearbeitung solange durchgeführt, bis aus irgendeinem Grund eine Unterbrechung eintritt (kein Interrupt), z. B. auf Daten aus einer anderen Task gewartet werden muß oder die Bearbeitung beendet ist.
Der Zustand 2, angehalten, bedeutet, daß die Task nicht bearbeitet werden darf. Dies ist z. B. der Fall, wenn die letzte Bearbeitung beendet ist und neue Daten abzuwarten sind.
Der Zustand 4, wartend, schließlich wird dazu benutzt, definierte Zeitverzögerungen zu erreichen. Die Task wird erst dann aufgerufen, wenn ein vom System verwalteter Zähler leergezählt ist. Dieser Zähler wird durch eine zum System gehörige Task in festen Zeitintervallen dekrementiert.
Damit eine Task bearbeitet werden kann, muß sie vom Koordinationsprogramm, dem Scheduler, aufgerufen werden. Dies erfolgt nach einer bestimmten Strategie, die im Schedulerprogramm „eingebaut" ist. Die einfachste Aufrufstrategie ist der zyklische Aufruf, auch (amerikanisch) „Round Robin"-Strategie genannt. Dabei werden die Tasks in einer festen Reihenfolge nacheinander abgefragt und aktive Tasks sofort bearbeitet. Bei Unterbrechung einer Task wird jeweils die nächste aktive Task aufgerufen. Nach der letzten Task in der Reihenfolge wird wieder von vorn begonnen.
Der Scheduler hat dabei folgende Aufgaben:
1. Neustart/Wiederstart von aktiven Tasks
2. Umschalten zwischen den Tasks
3. Anhalten von Tasks

Eine wesentliche Vereinfachung des Systemaufbaus ist erreichbar durch einen „kooperativen" Aufbau der einzelnen Tasks. Das heißt, die Tasks sind so aufgebaut, daß sie nie durch langwierige Bearbeitungsfolgen oder durch Warteschleifen die Bearbeitung der anderen Tasks aufhalten. Dafür stellt der Systemkern die nötigen Hilfsmittel zur Verfügung.

Koordination von Tasks

Zusammen mit diesen sind im Systemkern die folgenden Koordinationsfunktionen verfügbar:

1. PAUSE	Unterbrechen einer Task (Rückkehr zum Scheduler)
2. DELAY	Vorübergehendes Stoppen einer Task (Wartend)
3. STOP	Anhalten einer Task
4. ENDE	Beenden einer Task (mit Reinitialisierung)
5. RESTART	Wiederholung einer Task (Neustart von Beginn an)
6. START	Start einer Task

Lediglich die Funktion 6 (Start einer Task) wirkt auf eine andere als die aktuell bearbeitete Task. Alle anderen Funktionen verändern nur den Zustand der aktuellen Task. Ein Aufruf dieser Funk-

tionen (1 bis 5) bewirkt außerdem jeweils die Rückkehr zum Scheduler und den Aufruf aller anderen momentan aktiven Tasks, bevor die aufrufende Task weiter bearbeitet wird.
Wie ist nun ein solches Koordinationsprogramm zu realisieren?
Es ist klar, daß ein direkter Aufruf der Tasks, z. B. über CALL-Befehle, nicht möglich ist, da die Anzahl der aufzurufenden Tasks unbekannt ist und die einzelnen Tasks so wenig wie möglich von der Existenz der übrigen Tasks beeinflußt werden sollen.
Der einfachste und üblicherweise benutzte Weg zum Feststellen der Task-Anzahl besteht darin, die Information, die das System von jeder Task braucht, in einer Tabelle oder Liste, der Task-Zustandsliste, zusammenzufassen.
Der einfachste Weg, den Tasks möglichst viel Unabhängigkeit zu geben, ist, dafür zu sorgen, daß beliebige Unterprogrammaufrufe, also auch Systemaufrufe, „bunt gemischt" ohne Rücksicht auf die Schachtelungstiefe vorkommen dürfen. Das verlangt aber, jeder Task einen eigenen „privaten" Stack zuzuordnen. Dort kann eine Task Daten zwischenspeichern, ohne andere Tasks oder den Scheduler zu beeinflussen.

Systementwurf

Im ersten Entwurfsschritt ist nun festzulegen, welche Daten der Scheduler für seine Arbeit benötigt und wie sie organisiert werden. Die größte Flexibilität gestattet die Auslegung als Liste von Datenblöcken, die jeweils die Daten für eine einzelne Task enthalten. Jedes Listenelement enthält einen Zeiger auf das nachfolgende Element, somit können die Taskdaten beliebig im Speicher verteilt sein. Ein Listenelement wird häufig als Task-Leitblock bezeichnet.
Zu jeder Task muß sicher die Startadresse in der Liste eingetragen sein, damit die Task überhaupt gestartet werden kann. Des weiteren muß zu jeder Task der Stackpointer abgelegt werden, um dem Scheduler die Umschaltung der Taskstacks zu ermöglichen.

Die Task-Zustandsliste

Natürlich muß der Scheduler den Zustand der Task kennen. Dafür ist ein Eintrag in der Liste nötig, der im hier beschriebenen System wie folgt aussieht:
Für jede Task ist ein sieben Bit breites Identifikationsfeld vorhanden, das den Start einer Task erlaubt, ohne daß etwas anderes als deren Identifikation bekannt sein muß.
Das Identifikationsfeld ist mit einem Flag zusammengefaßt, das die Zustände „angehalten" (Flag = 0) und „bereit" bzw. „wartend" (Flag = 1) unterscheidet. Die Zustände „bereit" und „wartend" werden unterschieden durch den Inhalt eines weiteren Bytes, das gleichzeitig als Wartezeitzähler (DELAY.) dient. Die Wartezeitfelder aller Tasks werden von einer zum System gehörenden Task bearbeitet, die regelmäßig per Interrupt gestartet wird und damit eine Systemzeiteinheit festlegt. Ohne weitere Maßnahmen ist damit eine Verzögerung um max. 255 Systemzeiteinheiten möglich.
Für unser Minimalsystem sieht ein Element der Taskzustandsliste damit also folgendermaßen aus:

1. Wort: (Kalt-)Startadresse der Task
2. Wort: aktueller Stackpointer der Task, bei Initialisierung: Endadresse des Taskstacks
3. Wort: 1. Byte: Task-Identifikation in Bits 0...6, aktiv/inaktiv-Flag in Bit 7
   2. Byte: Verzögerung in Systemzeiteinheiten
4. Wort: Zeiger auf nächstes Taskzustandselement (zyklisch!)

Der „Scheduler"

Nachdem nun die zu bearbeitende Datenstruktur festgelegt ist, kann das Programm entworfen werden, das damit arbeitet. Es gliedert sich in zwei Hauptteile: die Initialisierung und die Task-Aufruf-Schleife. Als Grundstock für ein funktionierendes Steuerprogramm muß unser Scheduler für die Voraussetzungen zum einwandfreien Arbeiten des Systems sorgen.
Als erstes (*Bild 1*) wird deshalb der Stackpointer mit der Anfangsadresse des Scheduler-Stackbereiches geladen. Dann wird die Task-Zustandsliste ins RAM kopiert. Dies ist nötig, weil ihre Elemente während des Programmablaufes verändert werden, was im EPROM nicht möglich wäre. Der ausgeführte Scheduler verwendet das Z80-Register IX als Basisregister für alle Operationen mit den Taskzuständen, deshalb muß es in dieser Programmumgebung (Kontext) immer auf einen gültigen Eintrag der Task-Zustandsliste zeigen. Es wird deshalb zunächst mit der Anfangsadresse

```
                    .MAIN. - Multitasking Scheduler fuer kleine Stand-Alone - Anwendungen
                    Initialisierung, .MAIN.-Modul!

                                .EXTERN TASKLISTE, TASKAREA, .TASKLISTE
                                .EXTERN USR.INIT, IR.INIT, SCHEDULER
                                .INTERN RESET, S.INIT

                                ; Initialisierung der Taskzustands-Information
                                ; transportiert alle Taskzustandselemente ins RAM
                                ; (die Werte im ROM muessen bereits die richtigen Links
                                ;  enthalten und bei AKTIV. und DELAY. richtig gesetzt sein!),
                                ; initialisiert die Taskstartadressen und
                                ; setzt den Scheduler-Stackpointer

0000'                   S.INIT:
0000'  31 0040"                 LXI     SP, SCH.STACK           ; Stackpointer setzen
0003'  21 0000:04               LXI     H, TASKLISTE            ; Liste der vorbesetzten
0006'  11 0000:05               LXI     D, TASKAREA             ; Taskzustandselemente
0009'  01 0000:06               LXI     B, .TASKLISTE           ; nach TASKAREA im RAM
000C'  EDB0                     LDIR                            ; kopieren
000E'  DD21 0000:05             LXI     X, TASKAREA             ; Basisadresse fuer SCHEDULER!
0012'  DD4604                   MOV     B, TASK.ID (X)          ; Id der 1. Task merken
0015'                   ..NEXT:
0015'  DD5E02                   MOV     E, TASK.SP (X)          ; Stackpointer in DE laden
0018'  DD5603                   MOV     D, TASK.SP+ 1 (X)       ; DE laden
001B'  CD 003A'                 CALL    RESET                   ; Startadresse setzen
001E'  DD5E06                   MOV     E, NEXT.T (X)           ; Taskzustandselement-
0021'  DD5607                   MOV     D, NEXT.T+ 1 (X)        ; adresse ...
0024'  D5                       PUSH    D
0025'  DDE1                     POP     X                       ; ... weiterschalten
0027'  DD7E04                   MOV     A, TASK.ID (X)
002A'  B8                       CMP     B                       ; schon einmal rum?
002B'  20E8                     JR      NZ, ..NEXT              ; nochmal
002D'  DDE5                     PUSH    X                       ; Task-Zustands-Zeiger aufheben
002F'  CD 0000:07               CALL    USR.INIT                ; Anwendungs-Initialisierung
0032'  CD 0000:08               CALL    IR.INIT                 ; InterRupt-INITialisierung
0035'  DDE1                     POP     X                       ; Zeiger zurueck!
0037'  C3 0000:09               JP      SCHEDULER               ; und los geht's

                                ; Hilfsroutine zum Setzen der (Kalt) Startadresse der aktuellen Task
                                ; als Aufrufadresse fuer den SCHEDULER am Taskstack

003A'                           RESET:  ; zerstoert IY und Akku!
                                ; PARAMETER:    Task-Stackpointer in DE
003A'  D5                       PUSH    D                       ; Task-Stackpointer
003B'  FDE1                     POP     Y                       ; nach IY
003D'  DD7E00                   MOV     A, 0 (X)                ; Task-Startadresse
0040'  FD7700                   MOV     0 (Y), A                ; auf Task-Stack
0043'  DD7E01                   MOV     A, 1 (X)                ; als RETurnadresse
0046'  FD7701                   MOV     1 (Y), A                ; ablegen
0049'  C9                       RET
```

Bild 1. Initialisierung einer Task-Verwaltung

des Listenspeicherbereichs (im RAM) geladen. Bevor jedoch der erste Aufruf einer Task ausgeführt werden kann, müssen erst noch alle „privaten" Stacks der Tasks mit deren Kaltstartadressen vorbesetzt werden, da der eigentliche Aufruf per RET geschieht. Die Programmschleife, die das macht, hangelt sich mit Hilfe der Zeiger durch die ganze Taskzustandsliste. In der Schleife wird die Task-Identifikation jedesmal mit der Identifikation der ersten bearbeiteten Task verglichen; weil die Liste zyklisch aufgebaut ist, muß das Programm einmal wieder auf diesen Eintrag stoßen. Damit sind alle Einträge in der Liste bearbeitet. Die Scheduler-Initialisierung ist damit beendet. Normalerweise sind noch weitere vom jeweiligen Einsatzfall abhängige Vorbereitungen auszuführen, die von der Routine USR.INIT ausgeführt werden.

Bevor wir schließlich voll einsteigen können, müssen noch die Ein-/Ausgabeports eingestellt werden und die Interruptbearbeitung muß vorbereitet und freigegeben werden (*Bild 2*). Nach diesen Vorbereitungen ist jetzt endlich alles

Bild 2. Initialisierung der Ports und Interrupt-Vektoren

```
SCHEDU - Multitasking Scheduler fuer kleine Stand-Alone - Anwendungen
InterRupt-INITialisierung

                        .EXTERN IRVTABLE

0000'                   IR.INIT:

                                ; Interrupt-Vektoren und Ports initialisieren

0000'  21 0000:04               LXI     H, IRVTABLE             ; Adresse der Vektortabelle laden
0003'  7C                       MOV     A, H
0004'  ED47                     STAI                            ; High Byte -> IR-Basis-register
0006'                   ..SETUP:
0006'  21 0019'                 LXI     H, INITABLE             ; Tabelle der Portinitialisierungen


0009'  46                       MOV     B, M                    ; Anzahl der Eintraege
000A'  23                       INX     H
000B'                   ..SCHLEIFE:
000B'  C5                       PUSH    B                       ; Anzahl festhalten
000C'  4E                       MOV     C, M                    ; Port und ...
000D'  23                       INX     H
000E'  46                       MOV     B, M                    ; ... Byteanzahl laden
000F'  23                       INX     H
0010'  EDB3                     OUTIR                           ; Rest geht nach draussen
0012'  C1                       POP     B                       ; Zaehler zurueck
0013'  10F6                     DJNZ    ..SCHLEIFE              ; so oft wie angegeben
0015'  ED5E                     IM2                             ; Interrupt spezifizieren
0017'  FB                       EI                              ; und freigeben
0018'  C9                       RET                             ; fertig

                        ; Festdaten:    Port-Initialisierungswerte (Beispielwerte)

0019'                   INITABLE:
0019'  09               .BYTE   9                               ; Anzahl der SETUP-Definitionen

001A'  020508CFFFB7     .BYTE   ANZEIGE+ CTRL,  5, 8, 11001111B, 11111111B, 10110111B, 00000000B
0021'  030508CFFFB7     .BYTE   PROGRAMM+ CTRL, 5, 8, 11001111B, 11111111B, 10110111B, 01111111B
0028'  0C030A4F83       .BYTE   LOWTEMP+ CTRL,  3,10, 01001111B, 10000011B
002D'  0D030A4F03       .BYTE   HIGHTEMP+ CTRL, 3,10, 01001111B, 00000011B
0032'  1702A90D         .BYTE   PPICTRL,        2, 10101001B, (6 < 1 ! 1)
0036'  1E03000770       .BYTE   CTC0,           3, 0, 00000111B, 125.
003B'  1F02C7FA         .BYTE   CTC0+ 1,        2, 11000111B, 250.
003F'  2002C7F0         .BYTE   CTC0+ 2,        2, 11000111B, 240.
0043'  210143           .BYTE   CTC0+ 3,        1, 01000011B    ; wird nicht gebraucht
```

```
SCHEDU - Multitasking Scheduler fuer kleine Stand-Alone - Anwendungen
Strategie: Round Robin

                        .EXTERN TASKAREA                ; RAM-Bereich der Taskzustandsliste
                        .EXTERN RESET
                        .INTERN SAVESP
                        .INTERN PAUSE, DELAY, STOP, END, RESTART, START

                        ; Datenstruktur fuer Taskzustandselemente

0007                    AKTIV.  = 7                     ; Flag-Bit
007F                    ID.MASK = 7FH                   ; Maske fuer Taskidentifikation
0002                    TASK.SP = 2                     ; Offset in Taskzustand
0004                    TASK.ID = 4                     ;       "
0005                    DELAY.  = 5                     ;       "
0006                    NEXT.T  = 6                     ;       "

                        ; Ein Taskzustandselement hat folgenden Aufbau:
                        ; 1. Wort: (Kalt-) Startadresse der Task
                        ; 2. Wort: aktueller Stackpointer der Task,
                        ;         bei Initialisierung: Endadresse des Taskstacks
                        ; 3. Wort:
                        ;       1. Byte: Taskidentifikation in Bits 0..6,
                        ;                aktiv/inaktiv-Flag in Bit 7
                        ;       2. Byte: Verzoegerung in Systemzeiteinheiten
                        ; 4. Wort: Zeiger auf naechstes Taskzustandselement (zyklisch!)


0000'                   SCHEDULER:
                        ; Eingang: IX mit Adresse eines Taskzustandselementes besetzt
                        ;         (Taskzustandselement muss richtig besetzt sein!)

                        ; Testen, ob Task aktiv
0000'  DDCB047E                 BIT     AKTIV., TASK.ID (X)
0004'  2814                     JR      Z, NEXTTASK

                        ; Testen, ob Task wartend
0006'  DD7E05                   MOV     A, DELAY. (X)
0009'  B7                       ORA     A
000A'  200E                     JR      NZ, NEXTTASK

                        ; Task aufrufen
000C'  DDE5                     PUSH    X
000E'  ED73 0000"               SSPD    SAVESP                  ; Speicherplatz fuer Stackpointer
0012'  DD6E02                   MOV     L, TASK.SP (X)
0015'  DD6603                   MOV     H, TASK.SP+ 1 (X)       ; Task-Stackpointer laden
0018'  F9                       SPHL                            ; Stack umschalten
0019'  C9                       RET                             ; Bearbeitung aufnehmen

                        ; Rueckkehr aus der Task:       je nach Bearbeitungszustand ueber
                        ;                               verschiedene Eintrittspunkte

                        ; Fortschalten auf naechste Task
001A'                   NEXTTASK:       ; durch Pointer weiterhangeln
001A'  DD5E06                   MOV     E, NEXT.T (X)           ; Taskzustandselement-
001D'  DD5607                   MOV     D, NEXT.T+ 1 (X)        ; adresse nach
0020'  D5                       PUSH    D
0021'  DDE1                     POP     X                       ; Indexregister
0023'  18DB                     JR      SCHEDULER               ; weiter mit diesem

                        ; Routine zur Kontext-Umschaltung
0025'                   SWITCH:
0025'  FDE1                     POP     Y                       ; Rueckkehradresse holen
0027'  2A 0000"                 LHLD    SAVESP                  ; Scheduler-Stackpointer retten
002A'  ED73 0000"               SSPD    SAVESP                  ; Task-Stackpointer
002E'  ED5B 0000"               LDED    SAVESP                  ; zwischenspeichern
0032'  F9                       SPHL                            ; Stack umschalten
0033'  DDE1                     POP     X                       ; Taskelementadresse zurueck
0035'  DD7302                   MOV     TASK.SP (X), E          ; Task-Stackpointer
0038'  DD7203                   MOV     TASK.SP+ 1 (X), D       ; ablegen
003B'  FDE9                     PCIY                            ; fertig, a la RETurn weiter

                        ; Von Anwendungsprogramm aufrufbare Eintrittspunkte:

003D'                   PAUSE:  ; Rueckgabe an Scheduler, wenn Warten noetig
                                ; keine Parameter, Rueckkehr wie aus Subroutine
                                ; alle benoetigten Register sind zu retten!
003D'  CD 0025'                 CALL    SWITCH
0040'  18D8                     JR      NEXTTASK

0042'                   DELAY:  ; wie PAUSE, aber fuer spezifizierte Zeit
                        ; PARAMETER:    Verzoegerung in Akku
0042'  CD 0025'                 CALL    SWITCH
0045'  DD7705                   MOV     DELAY. (X), A           ; Verzoegerung ablegen
0048'  18D0                     JR      NEXTTASK

004A'                   STOP:   ; Task inaktivieren, z.B. fuer Intertask-kommunikation
                                ; sonst wie PAUSE
004A'  CD 0025'                 CALL    SWITCH
004D'  DDCB04BE                 RES     AKTIV., TASK.ID (X)     ; Task inaktiv setzen
0051'  18C7                     JR      NEXTTASK

0053'                   END:    ; Beenden einer Task mit Reinitialisierung der Startadresse
                                ; Stack  m u s s  leer sein !!!!!
0053'  CD 0025'                 CALL    SWITCH                  ; Task-Stackpointer noch in DE
0056'  DDCB04BE                 RES     AKTIV., TASK.ID (X)     ; Task inaktiv setzen
005A'  CD 0000:05               CALL    RESET                   ; Eintrittsadresse reinitialisieren
005D'  18BB                     JR      NEXTTASK

005F'                   RESTART:; Wiederstart einer Task von Anfang an, evtl. mit Verzoegerung
                                ; Stack  m u s s  leer sein !!!!!
                        ; PARAMETER:    Verzoegerung in Akku
005F'  CD 0025'                 CALL    SWITCH
0062'  DD7705                   MOV     DELAY. (X), A           ; Verzoegerung ablegen
0065'  CD 0000:05               CALL    RESET                   ; Eintrittsadresse reinitialisieren
0068'  18B0                     JR      NEXTTASK
```

Bild 3. Routinen zur Verwaltung der Tasks

getan, um die reguläre Programmbearbeitung aufnehmen zu können (*Bild 3*). Das Basiszeigerregister IX zeigt auf das erste Element der Task-Zustandsliste. Im ersten Schritt wird nun geprüft, ob die aktuelle Task angehalten ist. Wenn dies der Fall ist, wird sofort zur nächsten Task weitergegangen. Ansonsten wird im zweiten Schritt geprüft, ob der Wartezeitzähler läuft. Nur wenn dieser auf Null steht, ist die Task aktiv und kann aufgerufen werden.

Dazu wird zuerst das Basiszeigerregister auf den Scheduler-Stack gerettet. Der momentane Stackpointerinhalt wird dann an einer dafür reservierten Stelle abgespeichert. Jetzt kann das Stackpointerregister mit dem Task-Stackpointer geladen werden. Im Task-Stack steht als letzter Eintrag die Adresse des nächsten zu bearbeitenden Befehls der Task (am Anfang steht dort die Startadresse der Task). Ein einfaches RET startet jetzt die Bearbeitung der Task.
Eine Rückkehr zum Scheduler findet mit jedem Aufruf einer Systemroutine in der laufenden Task (mit Ausnahme der Start-Routine) statt. Dabei kommt ein Trick zur Auswirkung: der CALL-Befehl beim Aufruf der Systemroutine hat automatisch die nächste Befehlsadresse der Task auf deren Stack abgelegt. Die Schedulerroutine SWITCH, die bei der Rückkehr immer als erstes aufgerufen wird, „friert" diesen Zustand des Taskstacks ein und führt eine Kontextumschaltung in den Scheduler-Kontext durch (d. h. die gesamte Programmumgebung des Schedulers, die auch den Stack beinhaltet, wird zugänglich gemacht). Dazu holt SWITCH zunächst die überflüssige eigene Rückkehradresse vom Stack in IY. Dann lädt sie – einigermaßen umständlich – den Task-Stackpointerwert in das Registerpaar DE und schaltet das Stackpointer-Register auf den Schedulerstack um. Von diesem wird das Basiszeigerregister zurückgeholt und damit ist der letzte Task-Zustand wieder zugänglich. Dort wird nun der aktuelle Task-Stackpointerwert abgelegt. Über die in IY stehende Rückkehradresse wird danach die Bearbeitung im Systemkontext fortgesetzt.
Das damit erfolgte „Einfrieren" des Taskstacks erlaubt, jede beliebige andere Aktivität zwischendurch auszuführen, ohne den Kontext (die Umgebung) der gerade verlassenen Task zu stören. In unserem Fall werden nun erst einmal alle anderen Tasks geprüft und, wenn aktiv, aufgerufen, bis nach einem vollständigen Umlauf die eben verlassene Task wieder an die Reihe kommt.

Doch weiter der Reihe nach:
Nach der Rückkehr von SWITCH wird evtl. noch der Zustand der letzten Task verändert, z. B. eine Wartezeit eingetragen, dann schaltet NEXTTASK das IX-Register zum nächsten Element der Task-Zustandsliste weiter. Damit wird dann genauso verfahren wie gerade beschrieben.

Der geschilderte Ablauf zeigt auch, warum die einzelnen Tasks „kooperativ" programmiert werden müssen. Da der Scheduler keine Eingriffsmöglichkeit in eine laufende Task hat, kann eine Schleife, in der keine Systemroutine aufgerufen wird, den ganzen Ablauf anhalten. Deshalb sind Warteschleifen nicht erlaubt!

Neustart einer Task

Der bis hier behandelte Teil des Systems erlaubt bereits den quasi-parallelen Ablauf mehrerer Programme durch verschachtelte Abarbeitung. Jedoch gibt es damit keine Möglichkeit, eine angehaltene Task zum Laufen zu bringen.

Dazu ist ein Zugriff auf den Taskzustand einer anderen als der aktuellen Task nötig. Dies leistet die Prozedur START (*Bild 4*). Ein direkter Zugriff auf einen bestimmten Eintrag der Task-Zustandsliste ist normalerweise nicht möglich, weil die Reihenfolge der Einträge und evtl. sogar ihre Lage im Speicher nicht bekannt sind. Wir müssen also mit einem bekannten Eintrag anfangen und die Liste anhand der Zeigerverkettung verfolgen, bis der gesuchte Eintrag gefunden ist. Die Vorgehensweise ist bereits bekannt: sie ist dieselbe wie bei der Stack-Initialisierung. Lediglich der Vergleich der Task-Identifikation muß verfeinert werden: die Identifikation muß unabhängig vom Zustand des höchstwertigen (Flag-)Bits gefunden werden (es ist aber auch die Beschränkung der Suche nur auf inaktive Tasks möglich.

Wenn die gesuchte Task bereits aktiv oder wartend ist, wird sie nicht beeinflußt; die Suchstrategie ist dann entsprechend anzupassen). Zusätzlich ist in der gezeigten Routine noch eine Sicherung eingebaut: wenn die gesuchte Task-Identifikation nicht vorhanden sein sollte, bricht die einfache Suchschleife nicht ab und das System „hängt sich auf". Das wird verhindert, indem geprüft wird, ob die Liste bereits einmal ganz abgesucht worden ist. In diesem Fall wird die Suche abgebrochen und über das Carry-Flag Fehlanzeige zurückgemeldet. Außerdem zeigt das Zero-Flag an, ob die gestartete Task wirklich angehalten war. Mit dieser Routine können wir nun eine inaktive Task „aufwecken", und eine Task kann sich am Ende ihrer Aktivitä-

```
                SCHEDU - Multitasking Scheduler fuer kleine Stand-Alone - Anwendungen
                Strategie: Round Robin

 006A'                  START:  ; Start einer Task, evtl. mit Verzoegerung
                        ; PARAMETER:    Verzoegerung in Akku
                        ;               Taskidentifikation in C-Register
                        ; ERGEBNIS:     Z  gesetzt, wenn Task inaktiv war
                        ;               CY gesetzt, wenn Task nicht gefunden
 006A'  FDE5                    PUSH    Y
 006C'  D5                      PUSH    D
 006D'  C5                      PUSH    B
 006E'  F5                      PUSH    PSW                     ; Register retten
 006F'  FD21 0000:04            LXI     Y, TASKAREA             ; Scheduler-Kontext verfuegbar machen
 0073'  3E7F                    MVI     A, ID.MASK
 0075'  FDA604                  ANA     TASK.ID (Y)             ; Taskidentifikation holen
 0078'  47                      MOV     B, A                    ; erste merken
 0079'                  ..SUCHE:
 0079'  B9                      CMP     C                       ; mit gesuchter vergleichen
 007A'  2818                    JR      Z, ..DIESE              ; gefunden?
 007C'  FD5E06                  MOV     E, NEXT.T (Y)           ; naechstes Taskzustands-
 007F'  FD5607                  MOV     D, NEXT.T+ 1 (Y)        ; element ...
 0082'  D5                      PUSH    D
 0083'  FDE1                    POP     Y                       ; .... ansteuern
 0085'  3E7F                    MVI     A, ID.MASK
 0087'  FDA604                  ANA     TASK.ID (Y)             ; naechste Taskid holen
 008A'  B8                      CMP     B                       ; wieder am Anfang?
 008B'  20EC                    JR      NZ, ..SUCHE             ; weitersuchen
 008D'  F1                      POP     PSW
 008E'  C1                      POP     B
 008F'  D1                      POP     D
 0090'  FDE1                    POP     Y
 0092'  37                      SCF                             ; melde nicht gefunden
 0093'  C9                      RET

 0094'                  ..DIESE:
 0094'  F1                      POP     PSW
 0095'  FDCB047E                BIT     AKTIV., TASK.ID (Y)     ; melde (in)aktiv/wartend
 0099'  FDCB04FE                SET     AKTIV., TASK.ID (Y)     ; setze aktiv/wartend
 009D'  FD7705                  MOV     DELAY. (Y), A           ; setze Verzoegerung
 00A0'  C1                      POP     B                       ; hole restliche Register zurueck
 00A1'  D1                      POP     D
 00A2'  FDE1                    POP     Y
 00A4'  C9                      RET
```

Bild 4. Routine zum Start einer inaktiven Task

ten ruhig „schlafenlegen" (anhalten), bis sie wieder gebraucht wird.

Das Aktivieren einer Task kann von beliebiger Stelle aus geschehen. Es muß lediglich die Adresse von START und die Identifikation der zu startenden Task bekannt sein. Der Start kann auch von einer Interruptroutine ausgeführt werden. Die Interruptroutine bleibt dadurch kurz und belastet das System nicht. Die Hauptarbeit – die für eine Interruptroutine zu komplex sein kann – wird dann im normalen Taskmodus ausgeführt. Ein Beispiel für eine solche Task ist die zum System gehörige Wartezeitbearbeitung (TICK, *Bild 5*).

Die Interruptroutine IR.TICK startet lediglich die Task TICK; erst diese dekrementiert die Wartezeitzähler aller Tasks. Die Bearbeitung läuft wieder nach dem Schema der Stackinitialisierung ab. Wie dort wird die gesamte Liste der Taskzustände einmal durchlaufen, und bei jeder Task wird der DELAY-Zähler dekrementiert, wenn er nicht bereits den Wert Null hat. Da das nicht im Interruptmodus geschieht, kann es keinen Konflikt mit dem Setzen oder Löschen des Zählers von anderer Stelle (START) geben. TICK zeigt auch den Aufruf einer Systemroutine. Der Aufruf von ENDE gibt die Bearbeitung wieder an den Scheduler zurück und setzt als Rückkehradresse für den nächsten Taskaufruf die Startadresse von TICK ein. Damit fängt TICK bei jedem Aufruf von vorn an und bearbeitet die gesamte Task-Zustandsliste.

```
SCHEDU - Multitasking Scheduler fuer kleine Stand-Alone - Anwendungen
Strategie: Round Robin - Beispiel-Task DRUCK
0003                    $READY  == 3                    ; READY-Bit Drucker-Steuerung

                        .EXTERN DRUCKZEIGER
0000'                   DRUCK:
0000'  DB22                     IN      READY
0002'  CB5F                     BIT     $READY, A
0004'  280F                     JR      Z, ..WARTE      ; Drucker nicht bereit?
0006'  2A 0000"                 LHLD    DRUCKZEIGER
0009'  7E                       MOV     A, M
000A'  FEFF                     CPI     $END            ; Textende testen
000C'  CC 0000:04               CALL    Z, ENDE         ; fertig?
000F'  23                       INX     H
0010'  22 0000"                 SHLD    DRUCKZEIGER     ; Druckzeiger weiterstellen
0013'  D320                     OUT     DRUCKER         ; Zeichen ausgeben
0015'                   ..WARTE:
0015'  AF                       CALL    RESTART         ; eine Runde aussetzen
0016'  CD 0000:05
                        ; Hilfsroutine, die signalisiert, ob Drucktask verfuegbar ist
0019'                   BEREIT: ; gibt erst zurueck, wenn Drucktask frei ist
                                ; zerstoert alle Register!
0019'  3EFF                     MVI     A, $END
001B'  2A 0000"                 LHLD    DRUCKZEIGER
001E'  BE                       CMP     M
001F'  C8                       RET     Z
0020'  CD 0000:06               CALL    PAUSE           ; eine Runde aussetzen
0023'  18F4                     JR      BEREIT          ; wieder probieren
```

Bild 6. Beispiel aus einem Protokolldrucker

Ein Anwendungsbeispiel

Schließlich soll noch ein Beispiel aus einer Anwendung des vorgestellten Systems gezeigt werden (*Bild 6*, vgl. [2]). Es handelt sich hier um eine Task, die einen von einer anderen Task vorbereiteten Text auf einem Drucker ausgibt. Der Grundaufbau der Task wird zumindest denen nicht fremd sein, die sich schon einmal mit Schnittstellen und deren Programmierung befaßt haben. Hier sind jedoch zwei Stellen besonders zu beachten:
Es gibt keine echte Warteschleife für den Fall, daß der Port noch keine neuen Daten aufnehmen kann, da das System nicht blockiert werden darf. Statt dessen wird die Sytemfunktion RESTART aufgerufen, die zum Scheduler – und damit zu den anderen Tasks – zurückschaltet und außerdem dafür sorgt, daß beim nächsten Aufruf die Task bei ihrer Startadresse wieder aufgenommen wird. Damit kommen bei jeder vergeblichen Portabfrage alle anderen aktiven Tasks zur Bearbeitung.
Dasselbe geschieht, wenn ein Zeichen ausgegeben wurde, da dann der Drucker erst einmal beschäftigt ist bzw. das Zeichen übertragen werden muß.
Wenn jedoch das letzte Zeichen des Textes ausgegeben ist und der Druckzeiger auf eine Endemarke zeigt, beendet sich das Druckprogramm durch den Aufruf von ENDE und wartet inaktiv auf den nächsten Start (wie TICK). So, wie die letzte Abfrage hier programmiert ist, würde in einem Einprogrammsystem Zeit verschenkt. Im vorliegenden Fall ist dies aber bedeutungslos, da ja ein Warten auf den Port die anderen Tasks nicht am Ablaufen hindert. Die gezeigte Form spart dagegen etwas Programmspeicher, da der Vergleich gleich nach dem sowieso nötigen Holen des Ausgabezeichens steht.

```
SCHEDU - Multitasking Scheduler fuer kleine Stand-Alone - Anwendungen
Strategie: Round Robin
                        ; Task-Identifikation:
0054                    $TICK   == 'T'

0034'                   IR.DELAY:       ; Verzoegerungsbearbeitung (Systemzeit)
0034'  08                       EXAF
0035'  D9                       EXX
0036'  0E54                     MVI     C, $TICK
0038'  AF                       XRA     A
0039'  CD 0000:05               CALL    START
003C'  D9                       EXX
003D'  08                       EXAF
003E'  FB                       EI
003F'  ED4D                     RETI
SCHEDU - Multitasking Scheduler fuer kleine Stand-Alone - Anwendungen
Strategie: Round Robin

                        .EXTERN TASKAREA                ; RAM-Bereich der Taskzustandsliste

0000'                   TICK:   ; dekrementiert Verzoegerungszaehler <> 0 aller Tasks
0000'  FD21 0000:04             LXI     Y, TASKAREA     ; Scheduler-Kontext verfuegbar machen
0004'  3E7F                     MVI     A, ID.MASK
0006'  FDA604                   ANA     TASK.ID (Y)     ; Taskid holen
0009'  47                       MOV     B, A            ; erste merken
000A'                   ..SCAN:
000A'  FD7E05                   MOV     A, DELAY. (Y)   ; Verzoegerung holen
000D'  B7                       ORA     A
000E'  2803                     JR      Z, ..NULL       ; = 0 ?
0010'  FD3505                   DCR     DELAY. (Y)      ; dekrementieren
0013'                   ..NULL:
0013'  FD5E06                   MOV     E, NEXT.T (Y)   ; naechstes Taskzustands-
0016'  FD5607                   MOV     D, NEXT.T+ 1 (Y); element ...
0019'  D5                       PUSH    D
001A'  FDE1                     POP     Y               ; ... ansteuern
001C'  3E7F                     MVI     A, ID.MASK
001E'  FDA604                   ANA     TASK.ID (Y)     ; naechste Taskid holen
0021'  B8                       CMP     B               ; einmal rum?
0022'  CC 0000:05               CALL    Z, ENDE         ; dann fertig
0025'  18E3                     JR      ..SCAN          ; sonst weiterarbeiten
```

Bild 5. Wartzeit-Bearbeitung mit der Task TICK

In Bild 6 ist zusätzlich zur eigentlichen Drucktask noch eine Dienstprozedur (BEREIT) angegeben, die anderen Tasks zur Verfügung steht. Durch einen Aufruf von BEREIT kann eine Task ihren Ablauf mit der Druckausgabe synchronisieren. Ein solcher „Export" von Funktionen fördert ebenfalls die Übersichtlichkeit eines Programms, da alle Funktionen, die ein Modul betreffen, in diesem selbst ausgeführt werden und ein „fremdes" Modul keine Kenntnis über deren Aufbau braucht. Die hier gezeigte Routine BEREIT entspricht in ihrem Aufbau völlig einer einfachen Warteschleife in einem Einprogrammsystem. Da aber einfache Warteschleifen in unserem Multitaskingsystem verboten sind – sie sind nicht kooperativ – muß innerhalb der Schleife mindestens einmal eine Systemfunktion aufgerufen werden, die den Ablauf anderer Tasks erlaubt. Im Beispiel leistet dies die Prozedur PAUSE. Diese Routine ist, im Unterschied zu RESTART und ENDE, genauso zu verwenden wie eine beliebige andere Subroutine. D. h. die Bearbeitung der Task geht wie bei einer solchen mit dem dem Aufruf folgenden Befehl weiter, ohne daß die Programmumgebung verändert ist. Die Wirkung der Systemroutinen ist im Bild 4 bei den einzelnen Routinen selbst angegeben.

Für viele Zwecke ausreichend

Mit dem vorgestellten „Micro-Multitasking-Scheduler" können bereits mit kleinsten Konfigurationen die Programmierverfahren der Prozeßrechentechnik verwendet werden. Es fehlen natürlich noch viele Möglichkeiten der großen Systeme. So ist hier keine Unterstützung von Ein-/Ausgabevorgängen auf Systemebene vorhanden, und auch der Datenaustausch zwischen den Tasks wird nicht unterstützt. Aber bei Computern der Ausbaustufe des Z80-EMUF sind die Programme wohl noch einfach genug, daß zur Organisation dieser Vorgänge keine Klimmzüge nötig sind. Außerdem kann man den hier vorgestellten Kern ja noch um die zusätzlich gebrauchten Funktionen erweitern.

Mit dem vorhandenen Systemkern wurde ein Protokolldrucker ausgeführt, der interruptgesteuert ein Dutzend Eingänge überwacht, ihre Zustände – teils in festen Zeitintervallen – zusammen mit einem Meßwert protokolliert und das Protokoll mit Datum, Uhrzeit und diversen anderen Zusatzinformationen versieht. Datum und Uhrzeit werden ebenfalls interruptgesteuert intern geführt und können mit einer eingebauten Tastatur überprüft und eingestellt werden. Insgesamt enthält das Betriebsprogramm sechs Tasks und ist in knapp als 2 KByte Programmspeicher untergebracht. Der Systemkern benötigt davon etwa 300 Byte.

Die Listings wurden mit einem Z80-Assembler erstellt, dessen Befehlssyntax weitgehend der von Intel für den 8080 eingeführten entlehnt ist. Lediglich die Erweiterungen des Z80-Befehlssatzes ohne 8080-Entsprechungen sind der Zilog-Spezifikation entnommen.

Der Assembler erstellt relokatiblen (verschiebbaren) und linkfähigen (bindefähigen) Code. Er ermöglicht damit ein einfaches Aufspalten des Quellenprogramms in Module durch Angabe von internen Symbolen (die in anderen Modulen verwendbar sein sollen) und von externen Symbolen, die in anderen Modulen definiert sind.

Literatur

z1] *Kanis, W.*: Der Z80-EMUF. mc 1983, Heft 4, Seite 112.

[2] *Gaulke, E.*: Z80-EMUF als Spooler. mc 1983, Heft 10, Seite 98.

Dr. Dieter Götz

Z80-EMUF mißt Spannung und pH-Wert

Ein erweiterter Z80-EMUF [1] wird mit Hilfe eines zusätzlichen A/D-Wandlers und etwas Software zu einem Millivolt- bzw. pH-Meter.

Als A/D-Wandler wird ein 12-Bit-CMOS-A/D-Wandler von Intersil (ICL-7109) verwendet.

Es handelt sich hierbei um einen ausgesprochenen komfortablen Wandler. Er arbeitet nach der „Dual-Slope"-Integrationsmethode und besitzt TTL-kompatible Tri-State-Ausgänge. Zur Steuerung der A/D-Wandlung gibt es einen RUN/$\overline{\text{HOLD}}$-Eingang. Liegt dieser Ausgang auf logisch 1 bzw. bleibt er unbeschaltet, führt der Baustein eine Wandlung nach der anderen aus. Wird er auf logisch 0 gelegt, so beginnt die Wandlung erst beim Umschalten auf logisch 1. Eine zweite Möglichkeit zur Steuerung bietet der Zustand des Statusbits. Während der Wandlung liegt es auf logisch 1, am Ende geht es auf logisch 0. Im vorliegenden Fall wird der RUN/$\overline{\text{HOLD}}$-Eingang unbeschaltet gelassen und die Wandlung erfolgt softwaremäßig mit Hilfe des Statusbits.

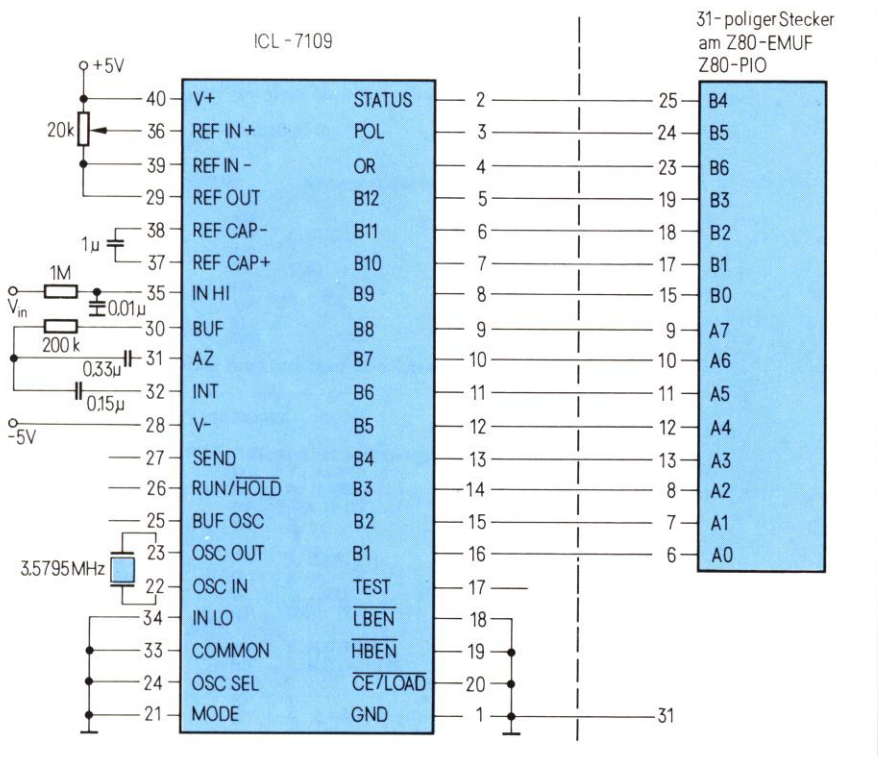


Bild 1. Die A/D-Wandler-Hardware und deren Anschluß am erweiterten Z80-EMUF

Außerdem besitzt der Wandler noch einen Ausgang zur Anzeige der Meßbereichsüberschreitung und der Polarität.

Der ICL 7109 hat die Möglichkeit zur Wahl zweier Betriebsarten: Handshakemodus oder Direktmodus. Es wurde der Direktmodus gewählt. Als Versorgungsspannungen werden ± 5 V benötigt. Die positive Versorgungsspannung kann z. B. an Pin 27 der 31poligen Leiste des Z80-EMUF abgegriffen werden.

Wenig Hardware mit viel Leistung

Wie man aus *Bild 1* sieht, sind neben einem Schwingquarz lediglich einige Widerstände und Kondensatoren notwendig. Die Schaltung wurde so bemessen, daß sie einen Meßbereich von ± 5 V umfaßt. Die genaue Einstellung erfolgt mit dem 20-kΩ-Potentiometer.

Die digitalen Ausgänge (Bit 1...Bit 12, Status, Overrange, Polarität) werden direkt auf die entsprechenden Pins des 31poligen Leistensteckers des EMUF gelegt.

EMUF als Digital-Millivoltmeter

Bild 2 zeigt das Maschinenprogramm, das den EMUF zum Millivoltmeter macht. Es beginnt mit der RAM-Adresse 8200H und umfaßt etwa 1 KByte. Von dem Monitorprogramm [1] werden zwei Unterprogramme verwendet, nämlich Tastaturabfrage/Eingabe (0097H) und Zeichenausgabe auf dem Display (00BDH). Das Programm umfaßt die Lösung folgender Aufgaben:

- ☐ Auswertung der A/D-Wandlung
- ☐ Hexadezimal/Dezimal-Umwandlung
- ☐ Umrechnen der eingelesenen Werte auf den Meßbereich von −5 V bis +5 V
- ☐ Ausgabe der angelegten Meßspannung in mV auf dem Display

Das Programm wird durch Eingabe der Adresse 8200H und anschließendes Drücken der Go-Taste gestartet. Durch Drücken der Display-Taste kann der in diesem Augenblick im Display erscheinende Wert fixiert werden. Durch Drükken der Enter-Taste wird der Programmlauf fortgesetzt. Mit Break gelangt man zurück ins Monitorprogramm.

EMUF als pH-Meter

Zur pH-Messung wird als Elektrode die heute nahezu ausschließlich verwendete

```
8200: af 32 f9 87 11 fa 87 cd 0d 82 c3 00 00 d5 3e cf
8210: d3 02 3e ff d3 02 3e cf d3 03 3e ff d3 03 db 01
8220: e6 40 28 10 21 e1 87 3e 40 06 06 77 23 10 fc cd
8230: bd 00 18 52 db 01 e6 10 20 05 cd bd 00 18 f5 db
8240: 00 6f db 01 e6 0f 67 54 5d 06 02 cd d9 82 19 06
8250: 03 cd d9 82 af ed 52 06 04 cd d9 82 19 d1 eb 73
8260: 23 72 2b eb d5 3a f9 87 b7 28 02 d1 c9 cd a3 82
8270: cd 90 02 db 01 e6 20 21 e5 87 20 06 3e 40 77 af
8280: 18 02 af 77 23 77 cd bd 00 cd 97 00 7b fe 04 20
8290: 02 d1 c9 fe 05 cc e0 82 db 01 e6 10 28 80 cd bd
82a0: 00 18 f5 11 e8 03 cd c3 82 41 11 64 00 cd c3 82
82b0: 78 cd ce 82 11 0a 00 cd c3 82 60 41 4d 78 cd ce
82c0: 82 68 c9 af 4f ed 52 38 03 0c 18 f9 19 c9 cb 27
82d0: cb 27 cb 27 cb 27 b1 47 c9 cb 2a cb 1b 10 fa c9
82e0: cd bd 00 cd 97 00 7b fe 06 20 f5 c9 00 00 00 00
82f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Bild 2. Dieses Programm macht den EMUF zu einem Digital-Millivoltmeter

Glaselektrode eingesetzt (z. B. bei der Firma Ingolf oder der Firma Schott erhältlich). Außerdem benötigt man zum Kalibrieren zwei Lösungen mit genau definierten pH-Werten. Dazu verwendet man zwei Pufferlösungen mit dem pH-Wert 7 und dem pH-Wert 9 (Pufferlösungen sind z. B. bei der Firma Merck, Darmstadt, erhältlich). Da alle späteren pH-Berechnungen sich auf diese beiden Eichpuffer beziehen, sollte die Glaselektrode vor dem Wechseln der Puffer sorgfältig mit destilliertem Wasser abgespült und die Pufferlösungen von Zeit zu Zeit gewechselt werden. *Bild 3* zeigt das Programm zur pH-Messung.

Die Kalibrierung zur pH-Messung

Soll der EMUF als pH-Meter eingesetzt werden, muß auch das Programm nach Bild 2 im RAM stehen, da das pH-Programm auf einige Routinen zurückgreift. Das Programm Kalibrierung beginnt bei der Adresse 8300H und geht bis 834FH. Es wird gestartet durch Eingabe der Adresse 8300H und anschließendes Drücken der Go-Taste. Im Display erscheint dann: PUF1= 7. Die Glaselektrode sollte jetzt am A/D-Wandler angeschlossen sein und in der Pufferlösung mit pH=7 stehen. Nach Drücken der Taste Enter beginnt die A/D-Wandlung und im Display erscheint der entsprechende Meßwert des Eichpuffers 7 in mV. Er liegt in der Nähe von 0 V. Zu beachten ist, daß sich an der Glasmembran der Glaselektrode relativ komplexe chemische Ionenaustauschvorgänge abspielen, so daß die Einstellung des chemischen Gleichgewichts einige Zeit benötigt. Ändert sich der Meßwert im Display nicht mehr, wird die Taste Break gedrückt; dadurch wird der augenblickliche Wert für pH 7 in der Speicherzelle 87FAH abgelegt und im Display erscheint: PUF2= 9. Nach Abspülen der Elektrode wird diese jetzt in den Puffer mit pH 9 gestellt und der Meßvorgang analog zum ersten Puffer wiederholt. Durch Drücken der Taste Break ist der Kalibriervorgang beendet. Die aus den beiden Puffern berechnete Empfindlichkeit der Glaselektrode (Δ mV/pH-Einheit) ist in Speicherzelle 87FEH abgelegt.

An sich genügt es, einmal zu Beginn der pH-Messungen die Kalibrierung durchzuführen. Bei längeren Messungen ändert sich jedoch manchmal die Empfindlichkeit der Elektrode. Eine gelegentliche Nachkalibrierung ist deshalb empfehlenswert. Ist die Empfindlichkeit der Elektrode bekannt, kann sie natürlich auch direkt in 87FEH abgelegt werden. In 87FFH ist dann eine Null einzugeben.

Die pH-Messung

Das Programm für die pH-Messung beginnt ab 8350H. Im Display erscheint zunächst: nESSEn. Nach Drücken von Enter beginnt die pH-Messung. Das Ergebnis erscheint im Display. Es erfolgt eine kontinuierliche pH-Messung; der Rücksprung ins Monitorprogramm erfolgt wieder durch die Break-Taste. Die Bedienung erfolgt analog der Spannungsmessung.

Eingabe des Programms

Zur Eingabe des Programms gibt es drei Möglichkeiten:

- ☐ Per Hand. Da die vorliegenden Programme schon relativ umfangreich sind, ist dies ein wenig mühselig.
- ☐ Wird der EMUF oft als Millivoltmeter und zur pH-Messung eingesetzt, können die Programme neben dem Monitor noch im EPROM 2716 abgespeichert werden.
- ☐ Wie in [1] beschrieben, können über Bit 7 der PIO 0, Port B (Pin 22 des 31poligen Leistensteckers), Daten aus- bzw. eingegeben werden.

```
8300: 21 06 83 c3 df 83 3e a8 12 cd bd 00 cd 97 00 7b
8310: fe 06 20 f5 11 fa 87 af 32 f9 87 cd 0d 82 21 24
8320: 83 c3 df 83 3e f9 12 cd bd 00 cd 97 00 7b fe 06
8330: 20 f5 11 fc 87 cd 0d 82 ed 5b fa 87 2a fc 87 af
8340: ed 52 cb 3c cb 1d 22 fe 87 00 00 00 00 00 00 00

8350 - 849f: pH-Messung

8350: 21 e6 87 3e 6c 77 2b 3e 75 77 2b 3e 7b 77 2b 77
8360: 3e 75 2b 77 3e 4c 2b 77 cd bd 00 cd 97 00 7b fe
8370: 06 20 f5 cd 81 83 cd 97 00 7b fe 04 20 f5 c3 00
8380: 00 3e 01 32 f9 87 11 fc 87 cd 0d 82 cd 8c 84 db
8390: 01 e6 20 20 2b 2a fc 87 ed 5b fa 87 af ed 52 fa
83a0: ac 83 cd f6 83 7c c6 07 67 c3 53 84 11 00 00 eb
83b0: af ed 52 cd f6 83 af 3e 64 9d 6f 26 06 c3 53 84
83c0: ed 5b fa 87 2a fc 87 19 cd f6 83 7d b7 28 0a 3e
83d0: 06 94 67 3e 64 95 6f 18 7a 3e 07 94 67 18 74 11
83e0: e6 87 3e e5 12 1b 3e 9d 12 1b 3e 65 12 1b 3e 60
83f0: 12 1b af 12 1b e9 cd 19 84 cd 8c 84 dd 22 f7 87
8400: cd 42 84 cd 8c 84 cd 19 84 cd 8c 84 3a f7 87 67
8410: dd 22 f7 87 3a f7 87 6f c9 3a fe 87 57 7d 6c 26
8420: 00 1e 00 06 10 dd 21 00 00 29 17 d2 2f 84 2c dd
8430: 29 dd 23 b7 ed 52 d2 3c 84 19 dd 2b 10 eb af 5c
8440: 57 c9 06 64 21 00 00 cb 38 30 01 19 c8 eb 29 eb
8450: 18 f5 c9 e5 cd 8c 84 6c af 67 cd a3 82 eb e1 d5
8460: af 67 cd a3 82 d1 63 cd 90 02 21 e4 87 7e 23 77
8470: 21 e3 87 7e 23 77 3e 40 2b 77 af 21 e6 87 77 cd
8480: bd 00 cd 97 00 7b fe 05 cc e0 82 c9 c5 d5 e5 cd
8490: bd 00 e1 d1 c1 c9 00 00 00 00 00 00 00 00 00 00
```

Bild 3. Die zwei Programme zur Kalibrierung und zur eigentlichen pH-Messung. Die Software zur Spannungsmessung muß sich ebenfalls im Speicher befinden

```
7eoo:   cd c9 o1 21 54 7f 11 oo 3c o1 oe oo ed bo cd 49
7e1o:   oo fe 31 28 2c fe 32 28 o2 18 f3 cd c9 o3 11 oo
7e2o:   3c o1 1a oo ed bo 3e 93 d3 9f cd 49 oo fe 4a 2o
7e3o:   f9 cd 81 7e 1a cd fd 7e 13 2b 7c b5 ca 66 oo 18
7e4o:   f3 3e 9b d3 9f 21 95 7f 11 oo 3c o1 2d oo ed bo
7e5o:   cd 49 oo fe 4a 2o f9 cd 81 7e d5 e5 21 c2 7f 11
7e6o:   4o 3c o1 oc oo ed bo e1 d1 cd 3o 7f 12 13 2b 7c
7e7o:   b5 ca 66 oo d5 11 od oo cd 4e 7f d1 18 eb c3 66
7e8o:   oo cd c9 o1 23 7c 7f 11 oo 3c o1 oc oo ed bo cd
7e9o:   bo 7e cd c9 7e d5 cd c9 o1 21 88 7f 11 oo 3c o1
7eao:   od oo ed bo cd bo 7e cd c9 7e eb d1 af ed 52 c9
7ebo:   o6 o4 11 1o 3c d5 cd 49 oo d1 fe od c8 12 13 1o
7eco:   f4 cd 49 oo fe od c8 18 e7 21 1o 3c cd f2 7e cb
7edo:   27 cb 27 cb 27 cb 27 57 23 cd f2 7e b2 57 23 cd
7eeo:   f2 7e cb 27 cb 27 cb 27 cb 27 5f 23 cd f2 7e b3
7efo:   5f c9 7e fe 3a fa fa 7e d6 o7 d6 3o c9 f5 c5 e5
7foo:   cd o7 7f e1 c1 f1 c9 37 o6 o9 f5 d4 1c 7f dc 21
7f1o:   7f f1 1f 1o f5 cd 1c 7f cd 1c 7f c9 af d3 9e 18
7f2o:   o6 3e ff d3 9e 18 oo 21 11 oo 2b 7c b5 2o fb c9
7f3o:   d9 db 9e 17 3o fb o6 o8 11 o9 oo cd 4e 7f 11 14
7f4o:   oo cd 4e 7f db 9e 17 cb 19 1o f3 79 d9 c9 1b 7b
7f5o:   b2 2o fb c9 45 49 4e 28 31 29 2d 41 55 53 28 32
7f6o:   29 3f 45 4d 55 46 2o 41 55 46 2o 45 49 4e 47 41
7f7o:   42 45 21 2o 4f 4b 41 59 28 4a 29 3f 53 5o 45 49
7f8o:   43 48 45 52 41 4e 46 2e 53 5o 45 45 49 43 48 45
7f9o:   52 45 4e 44 45 45 4d 55 46 2o 41 55 46 2o 41 55
7fao:   53 47 41 42 45 21 4e 4f 43 48 2o 4e 49 43 48 54
7fbo:   2o 53 54 41 52 54 45 4e 21 2o 4f 4b 41 4b 28 4a
7fco:   29 3f 45 4d 55 46 2o 53 54 41 52 54 45 4e 21 oo
```

Bild 4. Ein Programm zum Datenaustausch zwischen EMUF und TRS-80

Bild 4 zeigt ein Maschinenprogramm, geschrieben für den TRS-80 Model I, das den Datenaustausch mit dem EMUF abwickelt. Die Aus- und Eingabe erfolgt über Bit 7 des Port B am 8255 (Adresse 9EH des TRS-80). Das kleine Programm ist dialogorientiert und erklärt sich von selbst. Es beginnt ab 32 256 (dezimal). Es wurden lediglich drei Routinen des TRS-80 verwendet, nämlich Zeicheneingabe (0049H), Zeichenausgabe (0033H) und Löschen des Bildschirms (01C9H). Durch Ändern dieser Routinen und des Bildschirmspeichers (3C00H), kann deshalb dieses Maschinenprogramm auf andere Z80-Systeme angepaßt werden. Zu beachten ist noch, daß die Datenübertragung ohne irgendwelche Handshakesignale arbeitet und deshalb die Zeitschleifen (im Listing unterstrichen) beim Einsatz eines anderen Computers ebenfalls angepaßt werden müssen.

Literatur

[1] *Götz, D.:* Z80-EMUF mit Display und Tastatur. mc 1984, Heft 9.
[2] Datenblatt ICL-7109, Intersil.

Helmut Paulo

Z80-EMUF steuert Selbstbau-Plotter

Für viele mc-Leser, die an dem Selbstbau-Plotter aus Heft 8/1983 interessiert sind, kam ein Bau des Gerätes deshalb nicht in Frage, weil die Steuersoftware nur für den TRS-80 vorhanden war. Da der Plotter über keinerlei „Eigenintelligenz" verfügte, konnte er natürlich ohne entsprechende Software nicht benutzt werden. Im folgenden Beitrag wird gezeigt, wie man sehr kostengünstig mit Hilfe eines Z80-EMUF (mc 4/1983) über eine Centronics-Schnittstelle dem Plotter zu eigener Intelligenz verhelfen kann. Dann kann man den Plotter mit einfachen PRINT-Befehlen von Basic aus ansprechen, ohne daß ein spezieller Rechnertyp vorausgesetzt wird.

Benötigt wird ein Z80-EMUF mit 2 PIOs sowie eine geeignete 5-V-Spannungsquelle. Über PIO-0-Port A werden die Zeichen von der Centronics-Schnittstelle aufgenommen, über PIO-0-Port B werden die Schrittmotoren gesteuert. PIO 1, Port A wird zur Steuerung des Haltemagneten für den Schreibstift verwendet. Damit das Ready-Signal von PIO-0 als Busy-Signal für die Druckerschnittstelle benutzt werden kann, muß es mit Hilfe eines Transistors invertiert werden [1]. Hierzu trennt man auf der Platine des EMUF die Leiterbahn, die zu Stift 3 des 31pol. Steckers führt, auf, und fügt die Schaltung nach *Bild 1* ein.

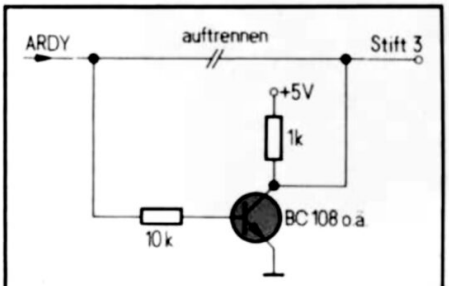


Bild 1. Auf der Z80-EMUF-Platine ist eine Leiterbahn aufzutrennen und durch einen Transistor als Inverter zu überbrücken

Es kann u. U. erforderlich sein, weitere Leitungen der Centronics-Schnittstelle zu verschalten, um zu verhindern, daß sich der Computer nach dem LPRINT- bzw. PRINT-Befehl „aufhängt". Dies betrifft z. B. die Leitungen $\overline{\text{Fault}}$, SLCT (Unit select) und PE (Paper empty). Um einen empfangsbereiten Drucker zu simulieren, müssen $\overline{\text{Fault}}$ und SLCT auf 1 (+5 V) und PE auf 0 gelegt werden. Eine andere Möglichkeit ist, den Druckertreiber des Computers so abzuändern, daß zum Datenverkehr zwischen Rechner und Plotter nur der Busy-Status abgefragt wird. In diesem Falle kann man auch auf den zusätzlichen Transistor zur Invertierung des Ready-Signals verzichten und die Busy-Leitung auf 1 testen; wenn Busy-1 ist, bedeutet dies nun (im Gegensatz zur üblichen Konvention): Daten können gesendet werden. Den freien Stift 26 der 31pol. Leiste des EMUF verbindet man mit Stift 5 des 20pol. Steckers, der zu PIO-1 gehört und das Bit A0 für die Schreibstiftsteuerung liefert. Nun kann man sämtliche Verbindungen über die 31pol. Steckerleiste herstellen (*Bild 2*).

Die Steuersoftware

Die gesamte Software befindet sich einschließlich des ASCII-Zeichensatzes in einem EPROM 2732 (beim Franzis-Software-Service oder beim Autor (H.-A.-Bühler-Str. 22, 7853 Steinen) erhältlich). Der Hexdump (*Bild 3*) beginnt zwar bei Adresse 8000H, jedoch ist das Programm nur mit Anfangsadresse 0000H lauffähig. Aus technischen Gründen (ROM-Bereich des TRS-80) war es nicht möglich, einen Ausdruck im Bereich 0000H bis 0EFFH herzustellen. Man muß einfach die 8 an jedem Zeilenanfang als 0 lesen, am Programm selber sind jedoch keine Änderungen vorzunehmen!
In Anlehnung an [1] wird per Interrupt von der Druckerschnittstelle eine Zeichenkette eingelesen und in einem Puffer gespeichert. Wenn das Ende der Zeichenkette durch CR (CHR$(13)) erkannt ist, wird das erste Zeichen der Zeichenkette als Befehlscode interpretiert und ggfls. unter Verwendung nachfolgender Parameter ausgeführt. Während der Ausführungszeit wird ein weiteres Einlesen von Zeichen mit Hilfe des Busy-Signals unterbunden, so daß der Benutzer sich (wie bei einem Drucker) um den Datenverkehr zwischen Rechner und Plotter nicht zu kümmern braucht. Befehlszei-

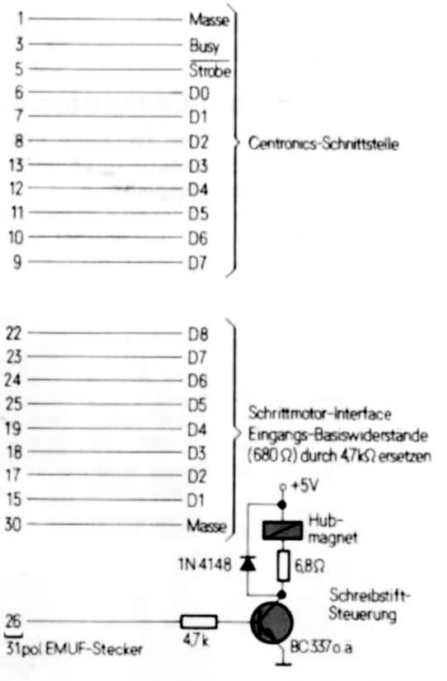


Bild 2. Verbindung des Z80-EMUF mit dem Plotter aus mc 8/1983

```
8000: F3 C3 0E 01 00 00 00 00  89 01 00 00 00 00 00 00
8010: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
8020: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
8030: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
8040: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
8050: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
8060: 00 00 00 00 00 00 C3 0E  01 00 00 00 00 00 00 00
8070: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
8080: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
8090: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
80A0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
80B0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
80C0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
80D0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
80E0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
80F0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00,
8100: 29 0B 00 02 10 27 05 33  33 60 22 00 19 02 06 FF
8110: 23 2B 10 FC 31 FF B7 21  00 01 11 00 B0 01 0E 00
8120: ED B0 AF 06 50 12 13 10  FC 3E 05 32 1B B0 3E 0F
8130: D3 03 D3 12 AF CD E9 03  CD F9 01 AF ED 47 ED 5E
8140: 3E 4F D3 02 3E 08 D3 02  DD 21 80 B0 3E B7 D3 02
8150: DB 00 FB 76 C3 53 01 DD  21 80 B0 DD 7E 00 DD 23
8160: FE 4C CA B4 05 FE 54 CA  D8 04 FE 50 CA A3 07 FE
8170: 44 CA BE 07 FE 46 CA D1  07 FE 52 CA 0B 02 FE 48
8180: CA F9 01 FE 53 CA 04 02  C9 DB 00 FE 0D CA 98 01
8190: DD 77 00 DD 23 FB ED 4D  3E 2C DD 77 00 DD 23 AF
81A0: DD 77 00 CD 57 01 DD 21  B0 B0 FB ED 4D DD 23 DD
81B0: 7E 00 FE 2C C2 AD 01 DD  22 3F B0 11 00 00 0E 00
81C0: DD 2B DD 7E 00 FE 20 CA  C0 01 FE 2C C8 D6 30 CD
81D0: D8 01 19 EB 0C C3 C0 01  D5 C5 26 00 6F 79 B7 CA
81E0: F5 01 CB 25 CB 14 E5 CB  25 CB 14 CB 25 CB 14 D1
81F0: 19 0D C3 DD 01 C1 D1 C9  C9 CD D1 03 CD D1 03 3E,
8200: FF D3 10 C9 AF D3 10 CD  96 07 C9 CD AD 01 3E 09
8210: 32 42 B0 7B CD 1F 08 32  17 B0 DD 2A 3F B0 CD AD
8220: 01 ED 53 19 B0 2A 19 B0  7C B5 C8 DD 21 BB 07 DD
8230: E5 3A 17 B0 FE 01 CA 86  02 FE 02 CA 5D 02 FE 03
8240: CA AF 02 FE 04 CA D8 02  FE 05 CA 01 03 FE 06 CA
8250: 35 03 FE 07 CA 69 03 FE  08 CA 9D 03 C9 E5 CD 21
8260: 05 E1 B7 C0 3A 0D B0 47  3A 07 B0 0F 32 07 B0 E6
8270: 0F 5F 3A 08 B0 E6 F0 B3  CD E9 03 CD D1 03 10 E8
8280: CD E3 03 20 D8 C9 E5 CD  38 05 E1 B7 C0 3A 0D B0
8290: 47 3A 07 B0 07 32 07 B0  E6 0F 5F 3A 08 B0 E6 F0
82A0: B3 CD E9 03 CD D1 03 10  E8 CD E3 03 20 D8 C9 E5
82B0: CD 68 05 E1 B7 C0 3A 0D  B0 47 3A 08 B0 07 32 08
82C0: B0 E6 F0 5F 3A 07 B0 E6  0F B3 CD E9 03 CD D1 03
82D0: 10 E8 CD E3 03 20 D8 C9  E5 CD 54 05 E1 B7 C0 3A
82E0: 0D B0 47 3A 08 B0 0F 32  08 B0 E6 F0 5F 3A 07 B0
82F0: E6 0F B3 CD E9 03 CD D1  03 10 E8 CD E3 03 20 D8,
8300: C9 E5 CD 38 05 E1 B7 C0  E5 CD 68 05 E1 B7 C0 3A
8310: 0D B0 47 3A 07 B0 07 32  07 B0 E6 0F 5F 3A 08 B0
8320: 07 32 08 B0 E6 F0 B3 CD  E9 03 CD D1 03 10 E4 CD
8330: E3 03 20 CD C9 E5 CD 38  05 E1 B7 C0 E5 CD 54 05
8340: E1 B7 C0 3A 0D B0 47 3A  07 B0 07 32 07 B0 E6 0F
8350: 5F 3A 08 B0 0F 32 08 B0  E6 F0 B3 CD E9 03 CD D1
8360: 03 10 E4 CD E3 03 20 CD  C9 E5 CD 21 05 E1 B7 C0
8370: E5 CD 68 05 E1 B7 C0 3A  0D B0 47 3A 07 B0 0F 32
8380: 07 B0 E6 0F 5F 3A 08 B0  07 32 08 B0 E6 F0 B3 CD
8390: E9 03 CD D1 03 10 E4 CD  E3 03 20 CD C9 E5 CD 21
83A0: 05 E1 B7 C0 E5 CD 54 05  E1 B7 C0 3A 0D B0 47 3A
83B0: 07 B0 0F 32 07 B0 E6 0F  5F 3A 08 B0 0F 32 08 B0
83C0: E6 F0 B3 CD E9 03 CD D1  03 10 E4 CD E3 03 20 CD
83D0: C9 C5 57 3A 02 B0 4F 3A  03 B0 47 0B 78 B1 20 FB
83E0: 7A C1 C9 57 2B 7C B5 7A  C9 D3 01 C9 E5 CD 88 04
83F0: E1 45 2A 00 B0 7E 23 FE  41 C2 F5 03 10 F7 E5 DD,
8400: E1 DD 7E 01 D6 41 6F 26  00 DD 7E 00 D6 41 DD 23
8410: DD 23 B7 CA 4B 04 FE 09  CA 98 04 FE 0A CA A0 04
8420: CD EC 07 FE 01 CA AB 04  FE 02 CA AE 04 FE 03 CA
8430: B4 04 FE 04 CA BA 04 FE  05 CA C0 04 FE 06 CA C6
8440: 04 FE 07 CA CC 04 FE 08  CA D2 04 26 00 3A 1B B0
8450: 6F 3A 42 B0 01 81 04 C5  FE 01 CA 86 02 FE 02 CA
8460: 01 03 FE 03 CA AF 02 FE  04 CA 69 03 FE 05 CA 5D
8470: 02 FE 06 CA 9D 03 FE 07  CA D8 02 FE 08 CA 35 03
84B0: C1 3A 1F B0 32 0D B0 C9  CD F9 01 3A 0D B0 32 1F
8490: B0 3A 41 B0 32 0D B0 C9  DD 2B CD F9 01 C3 01 04
84A0: DD 2B CD 04 02 C3 01 04  CD 86 02 C3 01 04 CD 5D
84B0: 02 C3 01 04 CD AF 02 C3  01 04 CD D8 02 C3 01 04
84C0: CD 01 03 C3 01 04 CD 35  03 C3 01 04 CD 69 03 C3
84D0: 01 04 CD 9D 03 C3 01 04  CD AD 01 7B 32 41 B0 DD
84E0: 2A 3F B0 CD AD 01 7B 32  42 B0 2A 3F B0 23 22 21
84F0: B0 06 00 7E B7 CA FD 04  04 23 C3 F3 04 7B 3D 32,
8500: 20 B0 2A 21 B0 E5 FD E1  FD 7E 00 D6 1F 6F CD EC
8510: 03 FD 23 3A 20 B0 3D 32  20 B0 B7 C2 08 05 C3 BB
8520: 07 2A 0E B0 3A 0D B0 47  2B 7C B5 CA 35 05 10 F8
8530: 22 0E B0 AF C9 3E 01 C9  AF 2A 09 B0 ED 5B 0E B0
8540: ED 52 EB 3A 0D B0 47 23  1B 7A B3 28 E8 10 FB 22
8550: 0E B0 AF C9 2A 10 B0 3A  0D B0 47 2B 7C B5 CA 35

8560: 05 10 FB 22 10 B0 AF C9  AF 2A 0B B0 ED 5B 10 B0
8570: ED 52 EB 3A 0D B0 47 23  1B 7A B3 28 B8 10 FB 22
8580: 10 B0 AF C9 CD AD 01 ED  53 2B B0 DD 2A 3F B0 CD
8590: AD 01 ED 53 2D B0 2A 0E  B0 AF CB 1C CB 1D 22 2F
85A0: B0 2A 10 B0 AF CB 1C CB  1D 22 31 B0 2A 2B B0 ED
85B0: 5B 2F B0 AF ED 52 C2 C4  05 2A 2D B0 ED 5B 31 B0
85C0: AF ED 52 C8 21 01 00 22  35 B0 2A 2B B0 ED 5B 2F
85D0: B0 AF ED 52 FA EE 05 CA  EE 05 3E 01 32 33 B0 2A
85E0: 2F B0 22 23 B0 2A 2B B0  22 25 B0 C3 FF 05 3E 02
85F0: 32 33 B0 2A 2B B0 22 23  B0 2A 2F B0 22 25 B0 2A,
8600: 2D B0 ED 5B 31 B0 AF ED  52 FA 23 06 CA 23 06 3E
8610: 03 32 34 B0 2A 31 B0 22  27 B0 2A 2D B0 22 29 B0
8620: C3 34 06 3E 04 32 34 B0  2A 2D B0 22 27 B0 2A 31
8630: B0 22 29 B0 AF 2A 2B B0  ED 5B 2F B0 ED 52 CA 11
8640: 07 AF 2A 2D B0 ED 5B 31  B0 ED 52 CA 39 07 AF 2A
8650: 25 B0 ED 5B 23 B0 ED 52  23 22 37 B0 22 3B B0 AF
8660: 2A 29 B0 ED 5B 27 B0 ED  52 23 22 39 B0 22 3D B0
8670: AF 2A 37 B0 ED 5B 39 B0  ED 52 FA C7 06 AF 2A 3D
8680: B0 ED 5B 39 B0 19 22 3D  B0 AF 2A 3B B0 ED 5B 3D
8690: B0 ED 52 F2 AD 06 2A 27  B0 23 22 27 B0 AF 2A 3D
86A0: B0 ED 5B 37 B0 ED 52 22  3D B0 CD 78 07 2A 23 B0
86B0: 23 22 23 B0 CD 65 07 AF  2A 23 B0 ED 5B 25 B0 ED
86C0: 52 CA BB 07 C3 7D 06 2A  3B B0 ED 5B 37 B0 AF 19
86D0: 22 3B B0 2A 3D B0 ED 5B  3B B0 AF ED 52 F2 F7 06
86E0: 2A 23 B0 23 22 23 B0 AF  2A 3B B0 ED 5B 39 B0 ED
86F0: 52 22 3B B0 CD 65 07 2A  27 B0 23 22 27 B0 CD 78,
8700: 07 AF 2A 27 B0 ED 5B 29  B0 ED 52 CA BB 07 C3 C7
8710: 06 3A 34 B0 FE 03 CA 26  07 2A 31 B0 ED 5B 2D B0
8720: AF ED 52 C3 30 07 2A 2D  B0 ED 5B 31 B0 AF ED 52
8730: 22 35 B0 CD 78 07 C3 BB  07 3A 33 B0 FE 01 CA 4E
8740: 07 2A 2F B0 ED 5B 2B B0  AF ED 52 C3 58 07 2A 2B
8750: B0 ED 5B 2F B0 AF ED 52  22 35 B0 CD 65 07 AF CD
8760: E9 03 C3 BB 07 2A 35 B0  3A 33 B0 FE 01 CA 74 07
8770: CD 5D 02 C9 CD B6 02 C9  2A 35 B0 3A 34 B0 FE 03
8780: CA B7 07 CD D8 02 C9 CD  AF 02 C9 CD D1 03 CD D1
8790: 03 AF CD E9 03 C9 E5 2A  04 B0 E5 C1 E1 0B 78 B1
87A0: 20 FB C9 CD AD 01 CB 23  CB 12 ED 53 0E B0 DD 2A
87B0: 3F B0 CD AD 01 CB 23 CB  12 ED 53 10 B0 C9 CD AD
87C0: 01 ED 53 02 B0 DD 2A 3F  B0 CD AD 01 ED 53 04 B0
87D0: C9 CD AD 01 CB 23 CB 12  ED 53 09 B0 DD 2A 3F B0
87E0: CD AD 01 CB 23 CB 12 ED  53 0B B0 C9 E5 D5 C5 47
87F0: 21 0F 0B 7E B8 23 C2 F3  07 2B 3A 42 B0 3D 5F 16,
8800: 00 19 7E C1 D1 E1 C9 01  02 03 04 05 06 07 08 01
8810: 05 03 07 02 0B 04 06 01  05 03 07 02 0B 04 06 E5
8820: D5 C5 47 21 07 0B C3 F3  07 41 42 55 41 42 4D 4B
8830: 44 43 43 43 45 43 42 43  4A 43 43 44 4B 4B 44 55
8840: 4A 42 4B 45 5F 41 42 46  44 50 4B 46 46 44 4B 4A
8850: 42 46 4B 45 4B 49 46 4A  42 4B 45 50 41 42 46 4B
8860: 44 5F 4A 42 4B 4B 45 5F  4A 42 46 44 4B 4B 43 55
8870: 4A 44 4B 4B 42 55 4A 45  55 41 4B 42 50 46 46 44
8880: 46 48 46 43 4B 48 46 44  46 46 46 42 50 4A 43 4B
8890: 44 46 4B 45 69 4A 42 4B  44 46 41 4B 44 46 46 55
88A0: 44 46 4A 43 4B 4B 43 46  45 46 42 46 44 46 4A 45
88B0: 5F 4B 42 46 44 46 43 46  45 46 4A 42 4B 41 42 55
88C0: 4B 48 55 44 46 46 46 42  46 47 46 49 50 45 46 47
88D0: 46 42 46 46 4B 4A 45 4B  41 42 46 44 50 4B 46 46
88E0: 44 4B 4A 42 4B 45 5F 41  42 50 4B 48 46 44 55 46
88F0: 46 4A 42 46 45 5F 41 42  46 4B 46 46 44 55 48 46,
8900: 4A 42 50 45 5F 41 42 4B  44 48 4B 44 51 4A 49 49
8910: 4B 42 51 4A 44 49 4B 49  51 4A 44 51 4B 47 51 4A
8920: 45 48 42 43 41 42 4B 44  48 4B 44 51 4A 49 49 4B
8930: 42 51 4A 42 43 45 50 41  47 46 4B 46 46 44 4B 4A
8940: 42 4B 45 4B 41 44 50 42  43 4B 42 51 4A 42 43 45
8950: 50 41 42 4B 4B 44 44 43  44 45 44 42 44 4A 42 4B
8960: 41 4B 44 46 46 55 44 46  4A 45 5F 41 44 46 4B 44
8970: 55 46 46 42 4B 47 46 45  55 49 46 43 4B 48 46 46
8980: 55 4A 45 5A 41 44 50 4B  46 50 45 5F 4A 42 46 41
8990: 44 5A 4B 46 46 42 4B 47  46 45 46 49 55 42 55 44
89A0: 46 4A 45 46 41 44 5A 4B  46 46 42 4B 47 46 45 46
89B0: 49 46 43 4B 42 4B 47 46  45 46 49 46 43 4B 48 46
89C0: 4A 42 55 45 46 41 42 50  4B 44 5F 49 50 42 55 4A
89D0: 45 50 41 44 46 4B 47 46  42 4B 46 46 44 46 48 46
89E0: 43 50 44 50 42 55 4A 45  5F 41 44 4B 46 55 4B 49
89F0: 55 45 46 47 46 42 4B 46  46 44 46 48 46 43 4B 4A,
8A00: 42 50 45 50 41 4B 46 55  44 4B 43 55 45 46 4A 45
8A10: 46 47 55 41 44 50 42 46  4B 42 4B 47 46 45 46 49
8A20: 46 43 4B 48 46 44 46 46  46 48 46 44 46 46 46 42
8A30: 4B 47 46 45 46 49 46 4A  42 46 45 50 41 4B 46 55
8A40: 44 46 4B 46 43 4B 49 46  45 46 47 46 42 4B 4A 42
8A50: 46 45 50 41 42 4B 44 46  4B 44 44 43 43 44 45 44 42
8A60: 44 4A 44 4B 4B 44 44 43  44 45 44 42 44 4A 42 4B
8A70: 45 50 41 42 46 45 46 4B  46 46 44 4B 4A 44 46 4B
8A80: 44 44 43 44 45 44 42 44  4A 42 4B 45 50 41 42 50
8A90: 4B 48 50 46 50 4A 42 46  45 5F 41 42 43 44 4D 4B
8AA0: 42 51 4A 44 47 4B 43 51  4A 42 53 45 53 41 4B 46
8AB0: 50 48 50 4A 42 55 45 5F  41 44 5A 4B 46 46 42 4B
```

Bild 3. Hex-Dump des EPROMs. Ein Assembler-Listing liefert der Franzis-Software-Service. Das EPROM enthält auch den Zeichengenerator zur Darstellung von Schrift

chen und Parameterwerte sind jeweils durch Komma voneinander zu trennen. Parameter dürfen nicht negativ sein und müssen ganzzahlige Werte haben! Die Verwendung von Integer-Variablen ist empfehlenswert. Nun zu den einzelnen Steuerbefehlen (Variablennamen sind beliebig):

H
Stift anheben
Beispiel: LPRINT„H"

S
Stift absenken
Beispiel: LPRINT„S"

L
Linie zeichnen
Beispiele: LPRINT„L, 100,200" oder
X=100:Y=200:LPRINT„L,";X;„,";Y
Es wird eine gerade Linie von der momentanen Position zum angegebenen Zielpunkt gezeichnet.

R
Relativbewegung in angegebener Richtung
Beispiele: LPRINT„R,3,200" oder
D=3:S=200:LPRINT„R,";D;„,";S
Es wird in der angegebenen Richtung (3) um die angegebene Schrittzahl (200) weitergefahren. Die 8 möglichen Richtungen sind: 1 = rechts, 2...8 = im Gegenuhrzeigersinn jeweils 45° weiter, z. B. 6 = links unten.

T
Text schreiben.
Beispiel: LPRINT„T,4,1,TEXT" oder
G=4: D=1: A$=„TEXT":LPRINT„T,";G;„,";D;„,";A$. Die erste Zahl bestimmt die Schriftgröße, die zweite die Richtung, in der geschrieben werden soll (hier sind nur die Werte 1, 3, 5 oder 7 zulässig).
P
Position definieren.
Beispiel: LPRINT „P,1500,800". Falls man beim Ausschalten des Gerätes vergessen hat, den Stift in die Home-Position (linke untere Ecke) zurückzuschikken, kann man mit diesem Befehl die (ungefähre) momentane Position eingeben und dann mit LPRINT„L,0,0" die Home-Position erreichen.

F
Format setzen.
Beispiele: LPRINT„F,4400,3200" (für DIN A4), LPRINT„F,6200,4500" (für DIN A3).
Es steht nun der in diesem Befehl festgelegte Bereich für X und Y zur Verfügung, und der Plotter fährt nie über die angegebenen Grenzen hinaus. Hierdurch wird eine mechanische Beschädigung des Gerätes weitgehend ausgeschlossen.

D
Delays (Verzögerungswerte) definieren.
Beispiel: LPRINT„D,500,10 000".
Die erste Zahl ist für die optimale Schrittfrequenz der Motoren erforderlich, die zweite bewirkt ein kurzes Anhalten nach jeder Stiftabsenkung, damit der Stift die Papieroberfläche erreicht, bevor gezeichnet wird. Die angegebenen Werte beziehen sich auf die Taktfrequenz von 2 MHz und den in [3] angegebenen Motortyp.

Ein Beispiel in Basic

Der Plotter wird, wie bereits erwähnt, über den LPRINT-Befehl angesprochen. Da in der Steuersoftware keine Fehler-Routinen enthalten sind, ist der Benutzer dafür verantwortlich, daß die Syntax genau eingehalten wird und keine unzulässigen Zahlenwerte gesendet werden. Außerdem darf der LPRINT-Befehl am Ende einer auszudruckenden Zeichenkette nur das Zeichen CR (Carriage Return, hex 13) generieren und nicht noch zusätzlich ein LF (Line Feed, hex 10). Sollte dies nicht der Fall sein, muß der Drucker-Treiber des Computers entsprechend geändert werden. Das Basic-Programm in *Bild 4* demonstriert alle implementierten Plotter-Befehle und dient zum Zeichnen mathematischer Kurven. Es wurde für den TRS-80 geschrieben, läßt sich aber leicht an andere Rechner anpassen. Die Fehlerroutine (ON ERROR GOTO ..) bewirkt, daß bei Bereichsüberschreitungen (oder Definitionslücken) trotzdem weitergerechnet wird, so daß

```
8AC0: 47 46 45 46 49 46 43 46  45 4B 4A 45 46 4B 44 44
8AD0: 43 44 45 44 42 44 4A 42  4B 41 42 46 4B 44 55 49
8AE0: 46 46 46 42 4B 46 46 49  46 45 50 47 46 4A 41 4B
8AF0: 44 5A 46 46 42 4B 47 46  45 5A 4A 44 50 4B 43 55,
8B00: 4A 42 55 45 50 41 4B 44  5F 42 50 47 46 45 46 49
8B10: 46 43 50 42 50 47 46 45  46 49 46 43 50 4A 42 55
8B20: 41 42 55 44 46 4B 49 46  43 4B 48 46 44 55 46 46
8B30: 42 4B 47 46 4A 45 5A 41  4B 44 5F 42 50 47 46 45
8B40: 55 49 46 43 50 4A 42 55  41 42 55 4B 43 55 44 5F
8B50: 42 55 4A 43 55 45 50 4B  42 50 4A 42 46 45 50 41
8B60: 4B 44 5F 42 55 4A 45 50  43 46 4B 43 50 4A 42 55
8B70: 45 50 41 44 5F 42 55 45  46 4B 4B 48 46 43 4B 49
8B80: 46 45 55 47 46 42 4B 46  46 44 4B 43 4B 4A 42 4B
8B90: 45 50 41 4B 44 5F 4A 42  55 4B 45 5F 4A 44 50 4B
8BA0: 43 55 4A 42 55 45 50 41  42 46 4B 42 4B 43 46 44
8BB0: 5F 43 46 42 4B 4A 42 46  45 5F 41 44 46 4B 47 46
8BC0: 42 4B 46 46 44 5A 43 55  45 46 4A 45 5A 42 55 41
8BD0: 4B 44 5F 4A 42 55 4B 49  55 46 46 47 50 4A 41 42
8BE0: 55 4B 43 55 44 5F 4A 42  55 45 5F 41 4B 44 5F 47
8BF0: 4B 46 4B 45 5F 4A 41 4B  44 5F 45 46 47 55 45 46,
8C00: 44 5F 4A 45 5F 41 42 46  4B 4B 46 44 55 46 46 42
8C10: 4B 47 46 45 55 49 46 43  4B 4A 42 50 41 4B 44 5F
8C20: 42 50 47 46 45 46 49 46  43 50 4A 42 55 45 50 41
8C30: 42 46 4B 4B 46 44 55 46  46 42 4B 47 46 45 55 49
8C40: 46 43 4B 4A 42 50 4B 4B  4B 4A 47 4B 41 4B 44 5F
8C50: 42 50 47 46 45 46 49 46  43 50 42 46 47 50 4A 41
8C60: 44 46 4B 47 46 42 4B 46  46 44 46 48 46 43 4B 48
8C70: 46 44 46 46 46 42 4B 47  46 4A 45 5A 41 42 4B 4B
8C80: 44 5F 43 4B 42 55 4A 45  5F 41 44 5F 4B 45 5A 47
8C90: 46 42 4B 46 46 44 5A 4A  45 5F 41 44 5F 4B 45 55
8CA0: 47 4B 46 4B 44 55 4A 45  5F 41 44 5F 4B 45 5F 46
8CB0: 4B 47 4B 44 5F 4A 45 5F  41 4B 44 46 46 55 44 46
8CC0: 4A 43 55 4B 45 46 47 55  45 46 4A 41 42 4B 4B 44
8CD0: 55 48 4B 47 4B 46 4B 4A  45 5F 41 44 5F 4B 42 55

8CE0: 45 46 49 55 45 46 42 55  4A 41 44 55 4B 46 4B 47
8CF0: 4B 48 4B 45 5F 4A 42 4B  41 44 4B 4B 47 4B 46 4B,
8D00: 49 4B 44 5F 4A 42 4B 45  5F 41 42 4B 44 46 4B 48
8D10: 4B 46 4B 49 4B 42 55 4A  45 50 41 44 50 4B 42 55
8D20: 48 4B 47 4B 49 4B 4A 45  46 42 4B 41 45 46 4B 42
8D30: 55 4A 44 46 41 42 46 4B  44 55 49 46 46 46 42 4B
8D40: 46 46 49 46 45 50 47 46  4A 41 42 55 4B 43 44 48
8D50: 4A 43 47 49 43 45 47 47  43 42 47 46 4D 4A 45 4E
8D60: 41 42 46 45 4B 4B 44 64  46 46 42 46 47 46 45 46
8D70: 49 46 43 4B 42 4B 47 46  45 46 49 46 43 4B 4A 42
8D80: 50 41 44 55 4B 47 50 45  4B 49 46 4B 46 44 4B 46
8D90: 4B 42 44 47 43 4A 45 4E  41 42 46 44 50 4B 45 4B
8DA0: 47 46 42 46 46 46 44 46  48 46 43 44 49 43 45 56
8DB0: 49 43 43 44 4A 44 4B 42  50 41 42 55 44 46 4B 45
8DC0: 46 43 55 46 50 4B 50 42  55 45 46 4A 45 5A 41 42
8DD0: 46 44 5F 4B 44 44 46 43  42 47 47 43 45 47 49 4B
8DE0: 42 4B 44 44 4A 42 46 45  55 41 42 46 44 5F 4B 44
8DF0: 44 46 43 42 47 47 43 45  47 49 43 43 47 42 47 47,
8E00: 43 45 47 49 43 43 47 4B  43 44 44 4A 42 50 45 55
8E10: 41 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
8E20: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
8E30: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
8E40: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
8E50: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
8E60: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
8E70: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
8E80: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
8E90: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
8EA0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
8EB0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
8EC0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
8ED0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
8EE0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
8EF0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00,
```

```
50 GOTO250
100 Y=SIN(X)+SIN(2*X)+SIN(3*X)
150 RETURN
250 CLEAR 1000
300 ON ERROR GOTO 4650
350 DEFINT E-S,U-W,Z
400 CLS
450 GOSUB 4900
500 PRINT"STIFT EINSETZEN, ENTER DRUECKEN"
550 INPUT A$
600 G=4:E=160:X0%=2080:Y0%=1440
650 CLS
700 GOSUB 4900
750 PRINT"1 - EINHEIT EINGEBEN"
800 PRINT"2 - KOORDINATENACHSEN ZEICHNEN"
850 PRINT"3 - FUNKTION PLOTTEN"
900 PRINT"4 - TEXT PLOTTEN"
950 PRINT"5 - KURVEN IN PARAMETERDARSTELLUNG ZEICHNEN"
1000 PRINT"6 - BELIEBIGES PLOTTER-KOMMANDO AUSFUEHREN"
1050 PRINT"7 - BEENDEN"
1100 INPUT W
1150 ON W GOTO 1250,1400,3200,4100,6400,5200,6300
1200 GOTO 650
1250 INPUT"EINHEIT (1 CM ENTSPRICHT 160)";E
1300 INPUT" KOORDINATENURSPRUNG, MITTE = (2200,1500)";X0%,Y0%
1350 GOTO 650
1400 SX=INT(4400/E):SY=INT(3000/E)
1450 Y2%=Y0%
1500 Y2%=Y2%-E:IFY2%>0GOTO1500
1550 Y2%=Y2%+E:X2%=X0%
1600 X2%=X2%-E:IFX2%>0THEN1600
1650 X2%=X2%+E
1700 X%=0:Y%=Y0%:GOSUB4900:GOSUB 5000
1750 GOSUB 4950
1800 X%=X2%:Y%=Y0%:GOSUB5000
1850 FOR I=1 TO SX
1900 R=3:S=16:GOSUB5050
1950 R=7:S=32:GOSUB5050
2000 R=3:S=16:GOSUB5050
2050 R=1:S=E:GOSUB 5050
2100 NEXT
2150 X%=4400:Y%=Y0%:GOSUB5000
2200 X%=4360:Y%=Y0%+40:GOSUB5000:GOSUB4900
2250 Y%=Y0%-40:GOSUB5000:GOSUB4950
2300 X%=4400:Y%=Y0%:GOSUB5000:GOSUB4900
2350 X%=4340:Y%=Y0%-120:IF Y%<2 THEN 2450 ELSEGOSUB5000
2400 Z$="X":GOSUB 5100:GOSUB4900
2450 X%=X0%:Y%=0:GOSUB5000:GOSUB4950
2500 Y%=Y2%:GOSUB5000
2550 FOR I=1 TO SY
2600 R=1:S=16:GOSUB5050
2650 R=5:S=32:GOSUB5050
2700 R=1:S=16:GOSUB5050
2750 R=3:S=E:GOSUB5050
2800 NEXT
2850 Y%=3000:GOSUB5000
2900 X%=X0%+40:Y%=2960:GOSUB5000:GOSUB4900
2950 X%=X0%-40:GOSUB5000:GOSUB4950
3000 X%=X0%:Y%=3000:GOSUB5000:GOSUB4900
3050 X%=X0%+60:Y%=2940:GOSUB5000
3100 Z$="Y":GOSUB5100:GOSUB4900
3150 GOTO 650
3200 CLS:GOSUB4900
3250 PRINT:PRINT"STEHT FUNKTION IN ZEILE 100 ?? J/N EINGEBEN!  ";:LINEINPUT F$: I
F F$="N" THEN PRINT"GIB FUNKTION IN ZEILE 100 EIN STARTE NEU!":STOP
3300 A=-X0%/E:B=(4400-X0%)/E
3350 PRINT:PRINT"INTERVALL EINGEBEN, MAXIMAL VON";A;"BIS";B
3400 INPUT A1,B1
3450 X%=INT((A1-A)*E):B=(B1-A)*E
3500 PRINT:INPUT"GENAUIGKEIT (1=GENAU ... 40=GROB)";GN
3550 FZ=1:Y%=Y0%:GOSUB4900:GOSUB5000
3600 D=GN/E:X=A1:T1=X%:T2=INT(B)
3650 FOR X%=T1 TO T2 STEP GN
3700 IF X%>T2 GOTO 4000
3750 GOSUB 100
3800 Y%=INT(Y*E+.5)+Y0%:IF Y%<0 OR Y%>3000 THEN Y%=40000:FL=1
3850 IF FL=1 THEN FL=0:GOSUB 4900: GOSUB 5000: GOTO 3950
3900 IF FZ=0 THEN GOSUB4950:GOSUB5000 ELSE GOSUB4900:GOSUB5000:FZ=0
3950 X=X+D
4000 NEXT
4050 GOTO 650
4100 CLS: GOSUB 4900
4150 INPUT"GROESSE ";G
4200 PRINT:PRINT"POSITION (0<X<4400, 0<Y<3000) ODER FUER START BEI"
4250 PRINT"MOMENTANER POSITION (0,0) EINGEBEN"
4300 INPUT X%,Y%
4350 "RICHTUNG (1,3,5,7)";DI
4400 LINEINPUT"TEXT  ";Z$
4450 IF X%=0 AND Y%=0 THEN GOTO 4500 ELSE GOSUB5000
4500 GOSUB 4950
4550 GOSUB 5150
4600 GOTO 650
4650 GOSUB 4900
4700 RESUME 4750
4750 R=1:S=GN:GOSUB5050:FL=1
4800 X=X+D
4850 T1=X%+GN:GOTO 3650
4900 LPRINT"H";:RETURN
4950 LPRINT"S";:RETURN
5000 LPRINT"L,";X%;",";Y%:RETURN
5050 LPRINT"R,";R;",";S:RETURN
5100 LPRINT"T,";G;",";1;",";Z$:RETURN
5150 LPRINT"T,";G;",";DI;",";Z$:RETURN
5200 CLS:GOSUB 4900
5250 PRINT"H ==> STIFT ANHEBEN":PRINT"S ==> STIFT ABSENKEN"
5300 PRINT"L ==> LINIE ZEICHNEN":PRINT"R ==> RELATIVBEWEGUNG"
5350 PRINT"P ==> POSITION DEFINIEREN"
5400 PRINT"T ==> TEXT PLOTTEN":PRINT"F ==> FORMAT SETZEN"
5450 PRINT"D ==> DELAYS SETZEN"
5500 PRINT"Z ==> BEENDEN"
5550 PRINT:PRINT
5600 PRINT$704," ";
5650 LINEINPUT"KOMMANDO  EINGEBEN  ";A$
5700 IF A$="H" THEN LPRINT"H":GOTO6250
5750 IF A$="S" THEN LPRINT"S":GOTO6250
5800 IF A$="L"THEN INPUT"ZIELPUNKT (X/Y)";X%,Y%:LPRINT"L,";X%;",";Y%:GOTO6250
5850 IF A$="R"THEN INPUT"RICHTUNG";R:INPUT"SCHRITTZAHL";S:LPRINT"R,";R;",";S:GOT
O6250
5900 IF A$="P" THEN INPUT"POSITION (X/Y)";X%,Y%:LPRINT"P,";X%;",";Y%:GOTO6250
5950 IF A$="T" THEN INPUT"GOESSE";G:INPUT"RICHTUNG";R:INPUT"TEXT    ";A$:LPRINT "
T,";G;",";R;",";A$:GOTO 6250
6000 IF A$="P" THEN INPUT"POSITION (X/Y)";X%,Y%:LPRINT"P,";X%;",";Y%:GOTO6250
6050 IF A$="F"THEN INPUT"FORMAT (X/Y)";X%,Y%:LPRINT"F,";X%;",";Y%:GOTO6250
6100 IF A$="D" THEN INPUT"VERZÖGERUNGSWERTE (M/S)";X%,Y%:LPRINT"D,";X%;",";Y%:GO
TO6250
6150 IF A$="Z" THEN GOTO 650
6200 GOTO 5200
6250 PRINT$704,STRING$(255,32);:GOTO5600
6300 LPRINT"L,0,0"
6350 END
6400 CLS:GOSUB4900
6450 INPUT"STEHT IN ZEILE 100:  X=F(T):Y=G(T) ?? (J/N)";B$
6500 IF B$="N" THEN PRINT"FUNKTIONEN EINGEBEN, NEU STARTEN!":STOP
6550 ON ERROR GOTO 7200
6600 INPUT"GIB BEREICH FUER T EIN";TA,TB
6650 INPUT"SCHRITTWEITE";TC
6700 T=TA:GOSUB 100: X%=X*E+X0%:Y%=Y*E+Y0%:GOSUB5000
6750 TA=TA+TC
6800 LPRINT"S";:FOR T=TA TO TB STEP TC
6850 GOSUB 100
6900 X%=X*E+X0%:Y%=Y*E+Y0%
6950 GOSUB 5000
7000 NEXT
7050 GOSUB 4900
7100 ON ERROR GOTO 4650
7150 GOTO 650
7200 RESUME 7250
7250 TA=TA+TC:GOSUB 4900
7300 GOTO 6800
```

Bild 4. Demonstrationsprogramm in Basic, geschrieben auf einem TRS-80 im Level-II-Basic

kein Programmabbruch erfolgt, sondern die Kurve vollständig, soweit sie auf das Format paßt, wiedergegeben wird. *Bild 5* gibt schließlich einen Eindruck von den grafischen Möglichkeiten.

Modifikationen

Selbstverständlich läßt sich das Z80-EMUF-Steuerprogramm noch erheblich erweitern. Insbesondere kann man von dem nicht benutzten PIO-1-Port B Gebrauch machen, um z. B. mit vier Tasten eine Steuermöglichkeit von Hand vorzusehen. Ferner wäre es denkbar, durch LEDs verschiedene Betriebszustände zu signalisieren, die man über die freien Bits von Port A (PIO-1) anspricht. Oder man sieht mit Hilfe von Sensoren eine elektrische Absicherung gegen Überfahren der Begrenzungen vor. Das benötigte Source-Listing für das in diesem Beitrag vorgestellte Programm kann ebenso wie das EPROM vom Franzis-Software-Service oder vom Autor bezogen werden. Abschließend sei den Firmen W. Kanis GmbH, Pöcking, sowie Haller & Tietze, Stuttgart, für ihre freundliche Unterstützung bei Fehlersuche und EPROM-Programmierung gedankt.

Literatur

[1] Gaulke, E: Z80-EMUF als Spooler, mc 1983, Heft 10.

[2] *Kanis, W.*: Der Z80-EMUF, mc 1983, Heft 4.

[3] *Paulo, H.*: Der Selbstbau-Plotter. mc 1983, Heft 8.

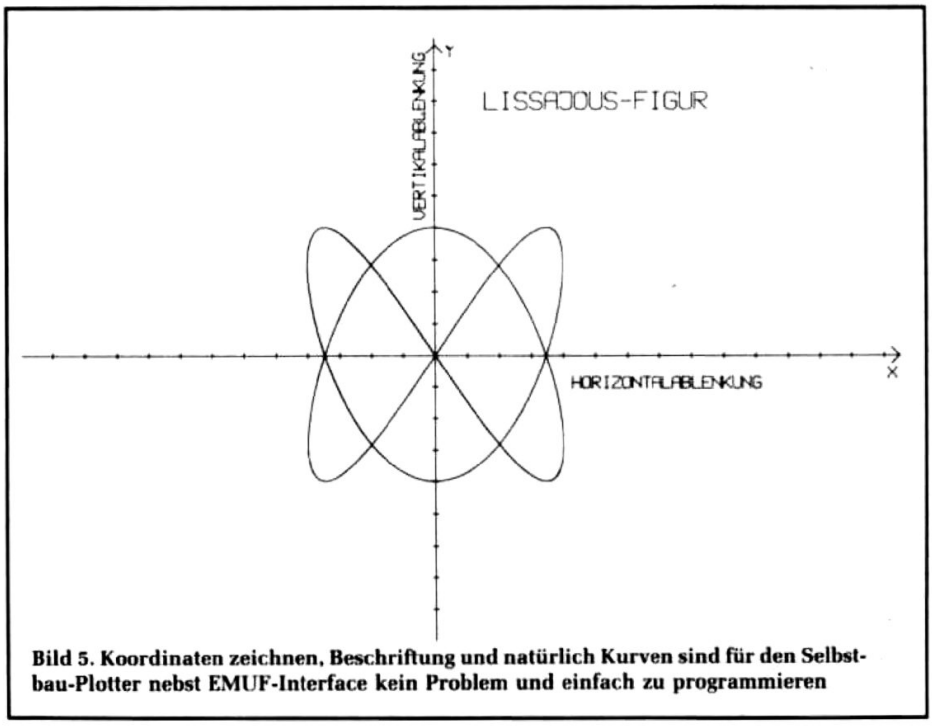


Bild 5. Koordinaten zeichnen, Beschriftung und natürlich Kurven sind für den Selbstbau-Plotter nebst EMUF-Interface kein Problem und einfach zu programmieren

Hermann Josef Middeke

VC-1541 für Z80-Systeme

Das Diskettenlaufwerk VC-1541 von Commodore ist zur Zeit wahrscheinlich das preiswerteste Laufwerk. Es enthält ein eigenes DOS und ist zum Anschluß an den C-64 konzipiert. Zum Datentransfer wird ein serieller IEC-Bus benutzt. Eine Verbindung mit anderen Systemen ist deshalb nur über eine besondere Schnittstelle möglich.

Das Laufwerk VC-1541 arbeitet mit 5¼-Zoll-Disketten mit einer Speicherkapazität von 170 KBytes je Disk. Das DOS (CBM DOS V2.6) kennt die Befehle NEW (Formatieren), INITIALIZE (Initialisieren einer Diskette), $ (Direktory lesen), VALIDATE (Rekonstruktion der „Block availability map", BAM), COPY (Kopieren von Files), RENAME (Umbenennung) und SCRATCH (Löschen). Die Verbindung mit dem C-64 erfolgt über eine serielle IEC-Schnittstelle, die aus nur drei Leitungen besteht, von denen zwei bidirektional sind. Die Leitung ATN (attention) meldet an das Laufwerk, daß ein Datentransfer erfolgen soll. Die Leitungen CLK (clock) und DAT (data) sorgen dann für die Übertragung und die erforderliche Synchronisation. Alle Leitungen sind in Richtung zum Laufwerk invertiert. Zum Anschluß läßt sich der Port-Baustein 8255 verwenden. In *Bild 1* ist die Beschaltung eines Bausteins als serielle IEC-Schnittstelle gezeigt und die Belegung des Steckers des Floppy-Laufwerks.

Zur seriellen Ein- und Ausgabe wurde Port C des 8255-Port-Bausteins verwendet, da gleichzeitig die eine Hälfte Eingang und die andere Ausgang sein kann. Der Portbaustein wird mit dem Befehl 1001 1010 (= 9AH) initialisiert. Im verwendeten Z80-System war die Befehladresse des Port-Bausteins 1BH. In einem System mit anderer Adresse müssen die im Hexdump *Bild 2* unterstrichenen Stellen entsprechend geändert werden.

Die Einbindung in das Betriebssystem

Zwei Routinen dienen zum Anknüpfen der Floppy-Routinen. Die Subroutine TEX bei 36A0H übergibt ASCII-Zeichen im Register E.TEX muß vom Benutzer definiert werden und darf kein Register verändern. TESTSTP bei 36A3H prüft die Stoptaste und unterbricht bei Bedarf LOAD oder SAVE. Dazu muß das Zero-Flag gesetzt sein. Soll von TESTSTP kein Gebrauch gemacht werden, dann muß mit LD A, FF OR A und RET bei 36A3H „kurzgeschlossen" werden. Für eine ausführliche Routine TESTSTP steht bei 32 C0H Platz zur Verfügung. Die Software zur Steuerung des Laufwerks ist der Original-Software des C-64 nachempfunden [1]. Sie belegt die Adressen 3200H bis 3800H und benutzt den Bereich von 4090H bis 40FFH für Puffer und Flags (*Tabelle 1*). In diesem Bereich wurden die Adressen so gewählt, daß das kleinere Byte mit der ursprünglichen, beim C-64 verwendeten Adresse identisch ist. 40A5H entspricht also $A5. *Tabelle 2* enthält eine Liste der verwendeten Subroutinen. Mit den Adressen der Subroutinen und der Bedeutung der Speicherstellen bei 4090H bis 40FFH in Tabelle 1 ist es möglich, das Programm an ein anderes Z80-System anzupassen.

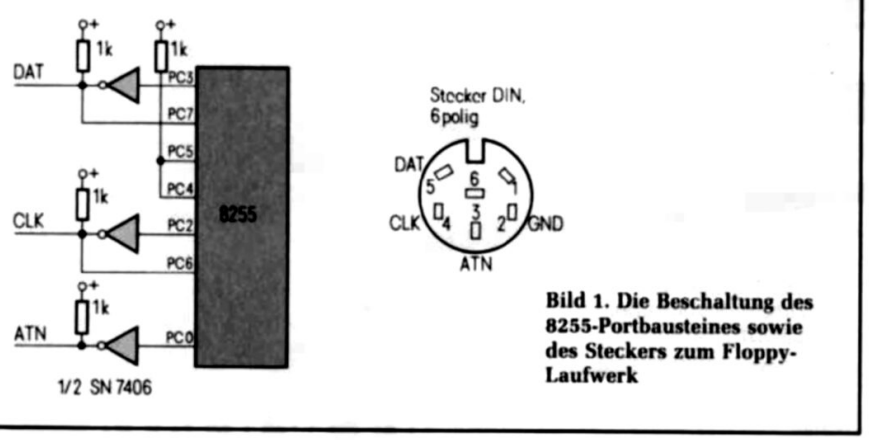


Bild 1. Die Beschaltung des 8255-Portbausteines sowie des Steckers zum Floppy-Laufwerk

Die Subroutinen aus der Nähe betrachtet

Die Subroutinen SERCON, SERCOF, BIT1OUT, BIT0OUT und COMPO bedienen den Port-Baustein. SERCON und SERCOF schalten die CLK-Leitung zum Laufwerk ein oder aus (Low oder High, negative Logik), BIT1OUT und BIT0OUT bestimmen den logischen Pegel der DAT-Leitung, COMPO fragt den Bus vom Laufwerk ab (CLKin und DATin werden Bit 7 des Akku und Carry-Flag); diese Routinen enthalten alle eine Verzögerungsschleife von etwa 20 µs Dauer, um dem Laufwerk genügend Zeit zum Erkennen zu geben (der Systemtakt des verwendeten Z80-Systems beträgt 4 MHz). Der logische Pegel der ATN-Leitung zum Laufwerk wird von den Routinen SENDUNTA, SENDUNLI, TALKS, LISTS, SEKATLI und BYTOUT bestimmt.

Die Routinen SENDUNTA und SENDUNLI (Untalk bzw. Unlisten senden) beenden den Datentransfer zwischen der Floppy. Sie werden unter anderem von CLRCH und von IECOFF gerufen.

Eröffnet wird ein Datenkanal mit der Floppy durch die Routine FOPIEC (File open in IEC). Nach der Prüfung auf korrekte Fileparameter. (Sekundäradresse in 40B9H, Länge des Filenamens in 40B7H) werden die Gerätenummer in 40BAH über die Subroutine LISTS und die Sekundäradresse in 40B9H mit SEKATLI sowie der Filename ab 40E0H mit IECOUT gesendet. Die Gerätenummer eines Laufwerks ist normalerweise 08; diese Nummer kann per Hardware (Trennen von Brücken auf einer Platine im Laufwerk) oder per Software (im Bedienungshandbuch beschrieben) geändert werden.

Nachdem ein Datenkanal eröffnet worden ist, kann mit IECOUT ein Byte zum Laufwerk gesendet werden. Das zu sendende Byte muß dazu im Akku (Register A) sein. IECIN empfängt ein Byte vom Laufwerk, es steht nach der Rückkehr zum rufenden Programm im Akku. Die Routinen IECOUT, IECIN, IECOFF sowie FOPIEC können also zur Datenübertragung von oder zur Floppy benutzt werden. Sie sind in den Routinen SAVE und LOAD jeweils zu einer Einheit zusammengefaßt. Von diesen beiden Routinen werden auch die anderen Subroutinen (LOVEOUT, LFCROUT, SFINAM, SAVOUT, BSOUT und STROUT) benutzt, die alle dazu dienen, Systemmeldungen wie „LOADING", „SAVING" und andere auszugeben.

Tabelle 1: Positionen und Funktionen der verwendeten Puffer und Flags im Bereich 4090H bis 40FFH.

Adresse	Name
4090	Status-Flag
4093	Load/Verify-Flag (00:Load; 01:Verify)
4094	IEC-Ausgabe-Flag
4095	IEC-Ausgabe-Puffer
409D	Flag f. Direkt-Modus (üblich 80H)
40A3	Bitzähler f. serielle Ausgabe
40A4	IEC-Eingangs-Puffer
40A5	Bitzähler f. seriellen Empfang
40AC/D	Startadresse bei Empfang
40AE/F	Zeiger auf Programm-Ende bei Load/Save
40B7	Länge des Filenamens
40BB	Logische Filenummer
40B9	Sekundäradresse
40BA	Gerätenummer (hier immer 08)
40BB/C	Zeiger auf Adresse des Filenamens (400E0/F)
40C0	Hilfspuffer
40C1/2	Endadresse
40C3/4	Startadresse bei Load
40E0/F	Filename

Speichern und Laden von Programmen

SAVE lädt Daten auf eine Diskette. Die Startadresse, ab der Daten zur Diskette übertragen werden, muß bei 40F0/1H stehen, die Endadresse im Registerpaar BC (zuerst das Low-Byte, dann das High-Byte). Die Startadresse wird ebenfalls auf der Diskette abgespeichert, so daß die gespeicherten Daten später wieder an dieselbe Stelle geladen werden können.

LOAD lädt Daten von einer Diskette. Hier steht die Startadresse im Registerpaar BC.Ob BC als Startadresse verwendet wird, oder die auf der Disk vorhan-

```
3200  db 1a e6 fb d3 1a c5 06 05 10 fe c1 c9 db 1a f6  0916
3210  04 d3 1a c5 06 05 10 fe c1 c9 db 1a e6 f7 d3 1a  0818
3220  c5 06 05 10 fe c1 c9 db 1a f6 08 d3 1a c5 06 05  0718
3230  10 fe c1 c9 c5 db 1a 5f 06 05 10 fe db 1a bb 20  079a
3240  f4 c1 cb 27 c9 3e 80 01 3e 03 21 90 40 b6 77 fb  0789
3250  a7 d2 76 32 3a 95 40 cd 91 33 db 1a e6 fe d3 1a  0887
3260  c9 00 00 00 f3 cd 0d 32 db 1a f6 01 d3 1a 3e 5f  063e
3270  01 3e 3f cd 69 33 db 1a e6 fe d3 1a 06 10 10 fe  06d1
3280  cd 00 32 c3 1a 32 00 00 06 ff 10 fe 06 1b 10 fe  0550
3290  1b 7a b7 20 f3 7b b7 20 ef c9 00 00 00 00 00 00  0569
32a0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  0000
32b0  06 be 3a 93 40 b7 28 02 06 cd c3 e8 32 00 00 00  0562
32c0  c3 a3 36 00 00 00 00 00 00 00 00 00 00 00 00 00  019c
32d0  00 00 00 00 5f cd a0 36 c9 00 00 00 00 00 00 00  02cb
32e0  00 00 00 00 00 00 00 00 21 9d 40 cb 7e c8 48 06  035d
32f0  00 21 00 37 09 7e cb 7f 20 06 cd d4 32 23 18 f5  0552
3300  cb bf cd d4 32 3e 20 c3 d4 32 00 00 cd 71 32 c3  07b7
3310  64 32 00 00 3a 9d 40 cb 7f c8 06 c6 cd ee 32 3a  06b2
3320  b7 40 47 77 c8 0e 00 2a bb 40 7e cd d4 32 0c 23  0630
3330  3a b7 40 b9 20 f4 c9 00 3a 9d 40 b7 f0 06 b0 cd  0808
3340  e8 32 3a b7 40 b7 c8 06 bb cd e8 32 3a b7 40 b7  085a
3350  47 c8 2a bb 40 7e e5 c5 cd d4 32 c1 e1 23 10 f5  08f9
3360  c3 60 36 00 f6 40 01 f6 20 f5 3a 94 40 cb 7f 28  071b
3370  11 37 21 a3 40 cb 1e e5 cd f0 33 e1 cb 26 2e 94  079e
3380  cb 26 f1 32 95 40 f3 cd 1a 32 fe 3f 20 03 cd 00  0722
3390  32 db 1a f6 01 d3 1a c5 06 05 10 fe c1 f3 cd 0d  0777
33a0  32 cd 1a 32 11 01 00 cd 88 32 c3 f0 33 32 95 40  05d1
33b0  cd 9d 33 db 1a e6 fe d3 1a c9 32 95 40 cd 9d 33  08d0
33c0  f3 cd 27 32 cd b3 33 cd 00 32 cd 34 32 fa ca 33  07f5
33d0  fb c9 00 00 21 94 40 cb 7e 20 05 37 cb 1e 20 07  056e
33e0  e5 f5 cd f0 33 f1 e1 23 77 a7 c9 00 00 00 00 00  07a6
33f0  f3 cd 1a 32 cd 34 32 da 45 32 cd 00 32 21 a3 40  0693
3400  cb 7e 28 05 cd 34 32 30 fb cd 34 32 38 fb cd 34  073b
3410  32 30 fb cd 0d 32 06 08 cd 34 32 d2 48 32 21 95  05ac
3420  40 cb 0e 38 06 cd 27 32 b7 20 03 cd 1a 32 cd 00  053d
3430  32 00 00 00 00 00 db 1a e6 f7 f6 04 d3 1a 10 d8  05d3
3440  06 10 00 00 00 00 00 00 00 05 ca 48 32 cd 34 32  0292
3450  38 f3 fb c9 00 00 00 00 21 b9 40 7e b7 f8 2b 2b  068c
3460  46 78 b7 c8 2e 90 36 00 3a ba 40 cd 67 33 3a b9  06bf
3470  40 f6 f0 cd ad 33 3a 90 40 b7 f2 82 34 f1 f1 c3  09e1
3480  77 36 3a b7 40 b7 28 0f 2a bb 40 47 7e e5 c5 cd  072d
3490  d4 33 c1 e1 23 10 f5 cd 71 32 a7 c9 00 00 00 00  06b1
34a0  ed 43 c3 40 32 93 40 af 32 90 40 3a ba 40 fe 08  0723
34b0  c2 83 36 3a b7 40 b7 ca 80 36 3a b9 40 32 c0 40  0748
34c0  cd 38 33 3e 60 32 b9 40 cd 58 34 3a ba 40 cd 64  06bf
34d0  33 3a b9 40 cd ba 33 cd 50 35 32 ae 40 21 90 40  0683
34e0  cb 4e c2 74 36 cd 50 35 32 af 40 3a c0 40 b7 20  0709
34f0  06 2a c3 40 22 ae 40 cd b0 32 21 90 40 cb 8e cd  0709

3500  a3 36 ca 27 36 3a 93 40 b7 20 24 2a ae 40 e5 cd  06d2
3510  50 35 e1 77 23 22 ae 40 3a 90 40 cb 77 ca fa 34  0754
3520  cd 64 32 cd 3c 36 da 74 36 a7 ed 4b ae 40 c9 2a  07e6
3530  ae 40 e5 cd 50 35 e1 be 28 09 3e 10 e5 21 90 40  0719
3540  b6 77 e1 23 22 ae 40 3a 90 40 cb 77 28 e1 18 d0  077e
3550  f3 af 21 a5 40 77 cd 00 32 cd 34 32 f2 59 35 06  06d7
3560  40 cd 1a 32 05 28 0d 00 00 00 00 cd 34 32 fa 64  0424
3570  35 f2 95 35 3a a5 40 b7 28 05 3e 02 c3 4a 32 cd  0640
3580  27 32 cd 00 32 3e 40 e5 21 90 40 b6 77 e1 21 a5  0680
3590  40 34 c2 5e 35 3e 08 77 cd 34 32 f2 98 35 2b cb  066e
35a0  1e cd 34 32 fa a1 35 23 35 20 ed cd 27 32 21 90  065d
35b0  40 cb 76 28 03 cd 7c 32 3a a4 40 fb a7 c9 00 00  06b0
35c0  00 00 00 00 ed 43 ae 40 4f 21 00 40 06 00 09 5e  033b
35d0  23 56 ed 53 c1 40 3a ba 40 fe 08 c2 83 36 3e 61  070e
35e0  32 b9 40 3a b7 40 b7 ca 80 36 cd 58 34 cd 14 33  0700
35f0  3a ba 40 cd 67 33 3a b9 40 cd ad 33 06 00 2a c1  066c
3600  40 22 ac 40 3a ac 40 cd d4 33 3a ad 40 cd d4 33  0743
3610  a7 ed 4b ae 40 2a ac 40 ed 42 30 20 2a ac 40 7e  06f6
3620  e5 cd d4 33 cd a3 36 e1 20 06 cd 3f 36 af 37 a9  0837
3630  23 22 ac 40 01 00 00 a7 ed 42 30 d4 cd 71 32 3a  05b6
3640  b9 40 cb 7f 20 13 3a ba 40 cd 67 33 3a b9 40 e6  072a
3650  ef f6 e0 cd ad 33 cd 71 32 a7 c9 00 00 00 00 00  0752
3660  1e 0a cd a0 36 1e 0d c3 a0 36 00 06 a8 21 06 0f  0473
3670  21 06 19 21 06 27 21 06 36 21 06 49 21 06 58 21  01fb
3680  06 67 21 06 78 00 c5 cd 0a 33 c1 21 9d 40 cb 76  05db
3690  20 03 cd e8 32 37 c9 00 00 00 00 00 00 00 00 00  030a
36a0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  0000
36b0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  0000
36c0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  0000
36d0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  0000
36e0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  0000
36f0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  0000
3700  0d 54 4f 4f 20 4d 41 4e 59 20 46 49 4c 45 d3 0d  0474
3710  46 49 4c 45 20 4f 50 45 ce 0d 46 49 4c 45 20 4e  048d
3720  4f 54 20 4f 50 45 ce 0d 46 49 4c 45 20 4e 4f 54  04b3
3730  20 46 4f 55 4e c4 0d 44 45 56 49 43 45 20 4e 4f  0496
3740  54 20 50 52 45 53 45 4e d4 0d 4e 4f 54 20 49 4e  04ca
3750  50 55 54 20 46 49 4c c5 0d 4e 4f 54 20 4f 55 54  04cf
3760  50 54 20 46 49 4c c5 0d 4d 49 53 53 49 4e 47 20  04ab
3770  46 49 4c 45 4e 41 4d c5 0d 49 4c 4c 45 47 41 4c  04c8
3780  20 44 45 56 49 43 45 20 4e 55 4d 42 45 d2 0d 46  048c
3790  49 4c 45 20 44 41 54 c1 0d 56 45 52 49 46 d9 0d  0503
37a0  4c 4f 41 c4 0d 49 2f 4f 2d 45 52 52 4f 52 20 c8  050b
37b0  0d 53 45 41 52 43 48 49 4e c7 0d 46 4f d2 0d 4c  04ee
37c0  4f 41 44 49 4e c7 0d 53 41 56 49 4e c6 0d 56 45  052f
37d0  52 49 46 59 49 4e c7 0d 46 4f 55 4e c4 0d 4f cb  05c8
37e0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  0000
37f0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  0000
```

Bild 2. Der Hexdump des Programmes. Bei 3700H beginnt die Liste der Systemmeldungen

dene Adresse von der ursprünglich die gespeicherten Daten stammen, hängt von der Sekundäradresse in 40B9H ab. Ist sie Null, wird die Adresse in BC verwendet. Wenn LOAD beendet ist, steht die Endadresse, bis zu der geladen wurde in 40AE/FH. Bei der Anpassung der Floppy-Routinen zum Beispiel an ein vorhandenes Basic-Programm wird dieser Wert benötigt.
Selbstverständlich können die Routinen nur funktionieren, wenn zuvor die zum Betrieb nötigen Parameter geladen worden sind: Das Load-Verify-Flag in 4093H (statt Load wird ein Prüflesen „Verify" durchgeführt wenn 4093H nicht Null ist), das Flag in 409AH (die Ausgabe wird vom Display auf den IEC-Bus umgeleitet, wenn 409AH= 08 ist), das Flag in 409DH (im ursprünglichen Commodore-Basic das Direkt-Modus-Flag, muß hier 80H sein) und die Puffer in 40B7H, 40B8H, 40B9H und 40BAH. Die zugehörigen Werte sind Tabelle 1 zu entnehmen. In 40BB/CH muß die Adresse 40E0H abgelegt sein und bei 40E0H muß der Filename in ASCII stehen.

Nicht nur SAVE und LOAD

Zum Abschluß folgen hier vier Routinen zur Bedienung des Laufwerks:

```
RECH  LD A,1A
      OUT Control,A
      LD A,08
      CALL TALKS
      LD A,6F
      CALL SEKATTA
LOOP  CALL IECIN
      LD E,A
      CALL USDF
      LD A,E
      CP 0D
      JRNZ , LOOP
      CALL IECOFF
      EI
      RET
```

Diese Routine liest den Fehlerkanal des Floppy-Laufwerks.
Zunächst wird der Port-Baustein aktiviert, dann 08 (Gerätenummer) mit TALKS gesendet und 6FH (60+„15") als Sekundäradresse mit SEKATTA.

Danach wird Byte für Byte von der Floppy empfangen und mit einer vom Benutzer zu schreibenden Routine ausgegeben, bis Carriage Return (0D) gefunden wird.

Mit IECOFF wird dann der Empfang beendet und das Interruptsystem wieder eingeschaltet.

Tabelle 2: Zusammenstellung der Adressen und der Bedeutung der verwendeten Routinen. Wenn das Programm an einer anderen Speicherstelle laufen soll, müssen diese Adressen geändert werden.

Adresse	Name	Funktion	
3200	SERCON	serial clock on	(Pegel an PC 2)
320D	SERCOF	serial clock off	(Pegel an PC 2)
321A	BIT1OUT	serial data on	(Pegel an PC 3)
3227	BIT0OUT	serial data off	(Pegel an PC 3)
3234	COMPO	compare port	prüft Bit 7 des Ports
3245	STATUS	set status	Statusflag in 4090H setzen
3264	SENDUNTA	send untalk	Abbruch des Datentransfers
3271	SENDUNLI	send unlisten	Abbruch des Datenempfangs
3288	Wait		wartet DE mal 1 ms
32B0	LOVEOUT	load/verify out	gibt „LOAD" od. „VERIFY" aus
32D4	BSOUT	bas. sign out	gibt ein ASCII-Zeichen aus
32E8	STROUT	string out	gibt eine Meldung aus
330C	CLRCH	clear channel	beendet eine Sendung
3314	SAVEOUT	save output	gibt „SAVING nn" aus
3338	SFINAM	searching f. filen.	gibt „SEARCHING FOR nn" aus
3364	TALKS	send talk	startet einen Datentransfer
3367	LISTS	send listen	startet den Datenempfang
33AD	SEKATLI	sec. adr. after list.	sendet die Sekundäradresse
33BA	SEKATTA	sec. adr. after talk	sendet die Sekundäradresse
33D4	IECOUT	iec-bus output	sendet über den IEC-Bus
33F0	BYTOUT	byte output	sendet ein Byte an den IEC-Bus
3458	FOPIEC	file open on iec	eröffnet File auf IEC-Bus
34A0	LOAD	load	holt Daten vom IEC-Bus
3550	IECIN	iec-bus input	holt ein Byte vom IEC-Bus
35C4	SAVE	save	sendet Daten an IEC-Bus
3660	LFCROUT	linefead/car. ret.	gibt CR und LF aus
366B	ERROUT	error output	gibt Fehlermeldung aus
36A0	TEX	output	Zeichenausgabe an Display
36A3	TESTSTP	break	Prüft eine Stop-Taste

```
CMOS  LD A,1A
      OUT Control,A
      LD A,nn
      LD (40B7),A
      CALL FOPIEC
      EI
      RET
```

Hier wird ein Befehl an das Laufwerk gesendet (z. B. N:TEST,88 d. h. eine Disk wird neu formatiert und mit dem Namen TEST und der Identifikationsnummer 88 versehen). Dazu wird wieder der Port aktiviert, die Länge des Filenamens (in 40E0H) nach 40B7H geladen und FOPIEC gerufen. In 40BBH muß E0H und in 40BCH muß 40H stehen.

```
LD A,1A
OUT Control,A
XOR A
LD BC,nnnn
CALL LOAD
EI
RET
```

Laden von einer Disk: Port initialisieren, Akku = 00 setzen (load statt verify), BC mit der Startadresse nnnn laden und LOAD rufen, nachdem alle anderen Parameter in 409AH, 409DH, 40B7H, 40B8H, 40B9H, 40BAH, 40BBH, 40BCH und 40E0H feststehen (s. Tabelle 1).

```
LD A,1A
OUT Control,A
LD A,F0
LD BC,nnnn
CALL SAVE
EI
RET
```

Abspeichern auf Diskette: Parameter wie im vorigen Beispiel. Bevor SAVE gerufen wird, muß der Akku auf die Stelle im Bereich 40XXH zeigen, die die Startadresse enthält und BC muß die Adresse des letzten zu übertragenden Datums enthalten.

Damit die Anpassung an andere Z80-Systeme leichter durchzuführen ist, kann beim Franzis-Software-Service ein Source-Listing des beschriebenen Programms bestellt werden.

Literatur

[1] *Angerhausen, Becker, Englisch, Gerits: 64 intern. Data Becker, Düsseldorf.*
[2] VC-1541 Floppy-Disk-Bedienungshandbuch, Commodore GmbH, Frankfurt.
[3] MCS-85 User's Manual. Seiten 4-29ff, Intel.