

U883



Der U883 ist ein spezieller Vertreter der [U881](#)-Familie der Einchipmikrorechner (EMR) der DDR. Der U883 ist maskenprogrammierter U881, der interne PROM enthält ein einfaches TINY-MPBASIC.



Von Zilog gibt es den [Z8671](#), einen Z8601 mit integriertem TINY-BASIC. **Diese Variante entspricht überhaupt nicht dem U883!!** Weder die BASIC-Beschreibung noch der PROM-Hexdump entsprechen denen des U883. Damit ist der Z8671 auch kein Ersatz für den U883.

Der U883 wird im Heimcomputer [JU+TE TINY](#) genutzt.

Downloads

- [u883bas.zip](#) ROM des U883 incl. Assemblerquellcode

Damit kann man den U883 durch einen U882 + 2K-EPROM ersetzen. Ebenso kann ein moderner Z8-kompatibler Prozessor genutzt werden, wie z.B. [Zilog Z86C93](#) statt U883.

Literatur

- Claßen/Oefler, Wissenspeicher Mikrorechner-Programmierung, VEB Verlag Technik Berlin, 2. Auflage 1986 (als Auszug:
u88x-emr_cty.pdf
)
- TINY-MPBASIC. Kundeninformation, VEB Mikroelektronik Erfurt, 1984
- ein paar Hinweise stehen auch in Kieser/Bankel, Einchipmikrorechner, VEB Verlag Technik Berlin, 1986

TINY-MPBASIC

Der Einchipmikrorechner U883 enthält in seinem 2 KByte großen internen ROM einen einfachen BASIC-Interpreter.

...

Beschreibung s.a. TINY

Auszug aus Claßen/Oefler, Wissenspeicher Mikrorechner-Programmierung, S. 172-180

8. U883 TINY-MPBASIC

Der Einchipmikrorechner U883 enthält in seinem 2K Byte großen internen ROM einen einfachen BASIC-Interpreter. Daneben ist ein dazugehöriger Editor- /Debuggerteil in Entwicklung, der alle Funktionen, die für die Programmentwicklung notwendig sind, enthält (→ BM200). Diese Komponenten liegen im externen ROM, sie können nach vollendeter Entwicklungsarbeit entfallen.

Damit wird vielen potentiellen Anwendern des Einchipmikrorechners, die über keine Entwicklungstechnik verfügen, eine Möglichkeit gegeben, Programme zu entwickeln (z.B.: für Steuerungs- oder Regelungsaufgaben im Rationalisierungsmittelbau). Der Anwender kann somit seine Programme im Zusammenspiel mit der von ihm erstellten Hardware austesten. Nach der Erprobung ist das fertige Programm in einen EPROM zu laden, und das Gerät kann eingesetzt werden.

Da die Problemlösung in Form von BASIC-Programmen erarbeitet wird, ist ein schnelles Erstellen und Modifizieren der Anwendersoftware möglich, ohne daß ein großer Aufwand für die sonst notwendige Entwicklungstechnik auftritt.

8.1. Sprachkonzept und Anwendung

Das in TINY-MPBASIC geschriebene Anwenderprogramm wird vom im internen ROM- Bereich des U883 befindlichen BASIC-Interpreters abgearbeitet.

Neben dem BASIC-Programm sind Initialisierungsteile und, falls notwendig, die Prozeduren GET_CHAR (Einzelzeicheneingabe) und PUT_CHAR (Einzelzeichenausgabe) zu erstellen. Diese sehr stark vom Einsatzfall abhängigen Teile sind in Assemblersprache zu realisieren.

Nach durchgeführter Initialisierung kann das BASIC-Anwenderprogramm aufgerufen werden. Dies erfolgt durch einen CALL-Befehl zur Adresse %7FD. Vorher sind in Register 6 der höherwertige Teil und in Register 7 der niederwertige Teil der Startadresse zu laden. Der Anwender kann für seine Problemlösung zusätzlich noch externe Prozeduren und Funktionen in Assemblersprache realisieren, die vom BASIC aus aufrufbar sind. Das Vorhandensein externer Prozeduren und Funktionen wird dem Interpreter dadurch bekannt gemacht, daß in den Registern 8 und 9 die Adresse einer Prozedurnamentabelle übergeben wird. Falls keine externen Prozeduren verwendet werden, so sind die Register 8 und 9 vor Aufruf des BASIC- Interpreters Null zu setzen.

Das BASIC-Anwenderprogramm muß syntaktisch fehlerfrei sein. Es wird in verdichteter Form abgespeichert. Unter dem Betriebssystem UDOS existiert ein Konvertierungsprogramm (mit dem Namen COMPRIMIERE), das die verdichtete Form aus einer BASIC-Quelldatei herstellt. Dabei werden automatisch Standardprogrammteile zur Initialisierung und zur Einzelzeichenein- bzw. -ausgabe hinzugefügt.

Der TINY-MPBASIC-Interpreter verarbeitet intern 16 Bit breite Daten, die als Integergrößen (Zweierkomplementdarstellung: -32768 ... +32768), Wortgrößen oder Bytewerte (niederwertige 8 Bit) interpretiert werden können. Konstanten können in Dezimal- oder Hexadezimalschreibweise (durch das Prozentzeichen gekennzeichnet: %0 ... %FFFF) angegeben werden. Negative Dezimalzahlen müssen in Klammern gesetzt werden, damit das Vorzeichen nicht als Operator wirkt. Für Variablenbezeichnungen sind die Buchstaben A bis Z verwendbar. (Sie belegen im Registersatz die Adressen ab %30.)

Bild 8.1. Zusammenspiel des U883-BASIC-Interpreters mit Anwenderteilen

Ausdrücke werden durch Verknüpfung von Konstanten, Variablen- oder Funktionswerten durch logische und arithmetische Operatoren gebildet.

Operatoren	
+	Addition
-	Subtraktion
*	Multiplikation
/	Division
\$AND	logisches UND (bitweise)
\$OR	logisches ODER „
\$XOR	exklusives ODER “

Ausdrücke werden von links nach rechts ausgewertet. Es besteht die Möglichkeit der Klammerung. Die Verschachtelungstiefe hängt dabei von der verfügbaren Stackgröße ab.

Eine Prozedur ist ein in Assemblersprache geschriebenes Programm, das einen Satz von BASIC übergebenen Eingabeparametern verarbeitet und Ausgabeparameter an den BASIC-Interpreter zurückgibt. Die Parameterübergabe erfolgt im Stackbereich. Der Aufruf erfolgt über Prozedurnamen.

Bild 8.2. Parameterübergabeschema für externe Prozeduren

Funktionen sind Prozeduren, die genau einen Wert an den Interpreter übergeben. Sie können deshalb in Ausdrücken verwendet werden.

Neben einer Reihe von fest integrierten Standardprozeduren und -funktionen hat der Anwender die Möglichkeit, eigene, seiner Hardwarekonfiguration angepaßte Prozeduren oder Funktionen hinzuzufügen. Die Verbindung zu TINY-MPBASIC erfolgt über die schon erwähnte Prozedurnamentabelle.

Bild 8.3. Aufbau der externen Prozedurnamentabelle

Das BASIC-Anwenderprogramm ist zeilenorientiert. Jede Zeile beginnt mit einer Nummer. Pro Zeile muß wenigstens eine Anweisung vorhanden sein. Mehrere Anweisungen auf einer Zeile sind möglich, wenn sie durch Semikolons getrennt sind. Die Programmzeilen müssen in aufsteigender Folge markiert sein. Die Anweisungsnamen werden in abgekürzter Form abgelegt, um möglichst wenig Speicherplatz zu belegen. Leerzeichen werden außer in Kommentaren und Texten entfernt. Die Sortierung und das Herstellen der verdichteten Form übernimmt normalerweise der Editorteil.

8.2. Anweisungen

Die Wertzuweisung für eine Variable ist die LET-Anweisung.

```
LET Variablenname = Ausdruck
```

Zur Programmverzweigung dient die GOTO-Anweisung.

```
GOTO Ausdruck
```

Unterprogramme können mit Hilfe der GOSUB-Anweisung aufgerufen werden. Das Ende eines Unterprogramms wird durch die RETURN-Anweisung markiert. Sie bewirkt die Programmfortsetzung bei der nach GOSUB folgenden Anweisung.

```
GOSUB Ausdruck  
RETURN
```

Unterprogramme können weitere Unterprogramme aufrufen. Die Verschachtelungstiefe hängt vom verfügbaren Stackbereich ab.

Programmverzweigungen werden mit der IF/THEN-Anweisung realisiert. Sie hat folgende Form:

```
IF Ausdruck Vergleichsoperator Ausdruck THEN Anweisung
```

Für Vergleichsoperator kann dabei stehen:

```
= , < , > , <= , >= oder <> (für ungleich)
```

Falls die nach IF folgende Bedingung wahr ist, so wird die folgende Anweisung ausgeführt. Andernfalls wird mit der nächstfolgenden Programmzeile fortgesetzt. Bei mehreren Anweisungen innerhalb einer Zeile wird der Rest der Programmzeile übersprungen.

Mit der PROC-Anweisung kann innerhalb des BASIC-Programms eine externe Prozedur aufgerufen werden.

```
PROC [Variablenliste] = Prozedurname [Parameterliste]
```

Die Variablen innerhalb der optionalen Variablenliste (Rückgabeparameter) und die Parameter innerhalb der ebenfalls optionalen Parameterliste (Eingabeparameter) sind durch Kommas zu trennen. Als Parameter sind Variablen und Konstanten zulässig.

Zur Ein-/Ausgabe dienen die Anweisungen PRINT, PRINTHEX und INPUT. Die Festlegung, über welche Geräte die Ein-/Ausgabeströme laufen, erfolgt durch die Programme GET_CHAR und PUT_CHAR. Dies kann je nach Anwendungsfall sehr unterschiedliche Hardware sein (z.B.: Terminal, Fernschreiber, LED- oder LCD- Anzeigen und verschiedene Tastaturen o.ä.).

```
PRINT ["beliebiger_Text"] [Ausdruck]  
PRINTHEX ["beliebiger_Text" ] [Ausdruck]
```

Die PRINT-Anweisungen geben einen Zahlenwert (bei PRINT: dezimal und bei PRINTHEX: hexadezimal) aus. Diesem Wert kann eine beliebige Textkette vorangestellt sein. Falls sowohl der Text als auch der Ausdruck fehlen, so wird lediglich ein Zeilenvorschub (%0D) ausgegeben.

```
INPUT ["beliebiger_Text"] Variablenname
```

Die INPUT-Anweisung gibt, falls vorhanden, erst den angegebenen Text aus und weist den eingegebenen Wert der spezifizierten Variable zu.

Zur Programmsteuerung dienen die STOP- und END-Anweisung.

STOP
END

Die Programmzeile, in der STOP auftritt, wird zu Ende abgearbeitet. Danach wird der Interpreter verlassen. Sie dient in Verbindung mit dem Editor/Debuggerteil zum Programmtest (Setzen von Unterbrechungspunkten). Durch Aufruf des Interpreters mit dem Eintrittspunkt %7FA kann der Programmablauf fortgesetzt werden.

Die END-Anweisung kennzeichnet das Programmende. Sie bewirkt ebenfalls das Verlassen des BASIC-Interpreters.

Zur Kommentierung der Programme dient die REM-Anweisung.

REM [Kommentartext]

Bei der Anwendung dieser Anweisung sollte der zur Verfügung stehende Gesamtspeicherplatz für das Anwenderprogramm bedacht werden.

Zur Erzeugung von Warteschleifen kann die WAIT-Anweisung verwendet werden.

WAIT Ausdruck

Sie bewirkt das Durchlaufen einer Softwarewarteschleife. Der Ausdruckswert spezifiziert dabei die Anzahl der Durchläufe. Beim Wert 1 wird die Schleife einmal und beim Wert Null 65536-mal durchlaufen. Bei einer Taktfrequenz von 4 MHz dauert ein Durchlauf eine Millisekunde.

Mit Hilfe der CALL-Anweisung kann ein in Assemblersprache geschriebenes Programm aufgerufen werden, ohne daß Parameter übermittelt werden.

CALL Ausdruck

Der Ausdruckswert ist die Programmadresse. Das Maschinenprogramm muß mit einem RET-Befehl enden.

Fest integriert im TINY-MPBASIC-Interpreter sind folgende Prozeduren:

NEG [Parameter]	arithmetische Negation
ABS [Parameter]	absoluter Betrag
NOT [Parameter]	logische Negation (bitweise)
GETR [Register]	liefert den Inhalt des angegebenen Registers (die höherwertigen 8 Bit sind Null)
GETRR [Register]	liefert den Inhalt des spezifizierten Registers (höherwertige 8 Bit) und des nachfolgenden Registers (niederwertige 8 Bit)
GETEB [Adresse]	holt Bytewert aus externem Speicher
GETEW [Adresse]	holt Wortwert aus externem Speicher

Analog können auch Register und Bytes (bzw. Wörter) im externen Datenspeicher gesetzt werden:

SETR [Register,Wert]	Register setzen
SETRR [Register,Wert]	Doppelregister setzen
SETEB [Adresse,Wert]	externes Byte setzen
SETEW [Adresse,Wert]	externes Wort setzen

Innerhalb der verdichteten Form des BASIC-Programms werden für die Anweisungen folgende Abkürzungen verwendet:

Anweisung	Abkürzung
LET	L
GOTO	G
IF..THEN	F....
GOSUB	S
RETURN	R
PROC	O
INPUT	I
PRINT	P
PRINTHEX	H
STOP	T
END	E
REM	M
WAIT	W
CALL	C

Der restliche Programmtext wird ohne Leerzeichen in ASCII-Zeichen (die Zeilennummer als 2 Byte große Hexadezimalzahl) abgespeichert. Das Zeilenende wird durch %0D gekennzeichnet.

8.3. Programmbeispiel

Das folgende Demonstrationsbeispiel zur Anwendung von TINY-MPBASIC wurde aus /46/ übernommen. Das Bild 8.4. zeigt das zur Initialisierung und Programmstart notwendige Assemblerprogramm.

Die Bilder 8.5. und 8.6. zeigen ein BASIC-Demonstrationsprogramm in Quellform und in der verdichteten Form.

```

0  REM      BASIC DEMONSTRATION
10  PRINT  "WAEHLN SIE BITTE EIN PROGRAMMBEISPIEL !"
20  PRINT
30  PRINT  "1 PRIMFAKTORZERLEGUNG"
40  PRINT  "2 UMRECHNUNG HEX-DEZIMAL"
50  PRINT  "3 UMRECHNUNG DEZIMAL-HEX"
60  PRINT  "4 REGISTERINHALT MODIFIZIEREN"
70  PRINT  "5 LANGSAM ALPHABET DRUCKEN"
80  PRINT  "6 NEU BEGINNEN"
90  PRINT
100  INPUT "PROGRAMM NR ? : " A
110  GOTO 150*A
150  REM PRIMFAKTORZERLEGUNG
160  INPUT " ZAHL=? " A

```

```

170 LET B = 2
180 IF A < 2, GOTO 240
190 LET C = A/B*B
200 IF C <> A, LET B = B+1; GOTO 190
220 PRINT B
230 LET A = A/B; GOTO 180
240 PRINT "FERTIG"
250 GOTO 100
300 REM UMRECHNUNGEN
310 INPUT "HEXZAHL=? " A
320 PRINT "DEZIMAL = " A
330 GOTO 100
450 INPUT "DEZIMALZAHL=? " A
460 PRINTHEX "HEX = " A
470 GOTO 100
600 REM REGISTERINH. MODIFIZ.
610 INPUT "REGISTER NR.: " A
620 PRINTHEX "INHALT = " GETR[A]
630 INPUT "NEUER INHALT: " B
640 PROC SETR[A,B]
650 GOTO 100
750 REM ALPHABET DRUCKEN
760 INPUT "WARTEZEIT ZWISCHEN ZWEI BUCHSTABEN [MSEC]:" A
770 LET B = 26; LET C = %41
780 LET Z = C; GOSUB 1000
790 LET Z = %20; GOSUB 1000; REM SPACE DAZWISCHEN
800 WAIT A
810 LET C = C+1; LET B = B-1
820 IF B <> 0, GOTO 780
830 LET Z = %D; GOSUB 1000; REM CR & LF ANHAENGEN
840 LET Z = %A; GOSUB 1000
850 GOTO 100
900 REM PROGRAMM VERLASSEN
910 END
1000 IF GETR[%FA] $A %10 <> %10 GOTO 1000
1010 PROC SETR[%FA,0]; REM REQUEST RUECKSETZEN
1020 PROC SETR[%F0,Z]
1030 RETURN
2000 ENDE

```

Bild 8.5. TINY-MPBASIC-Programm (Quellform)

```

0000h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF ; yyyyyyyyyyyyyyyyy
0010h: FF FF 8D 08 1B 8D 08 41 8D 08 51 8C 96 E6 FF 80 ; yÿ . . . A . QE-æÿ€
0020h: E6 F4 40 E6 F7 41 E6 F5 0D E6 F1 43 B0 FB 9F E6 ; æô@æ÷Aæö.æñC°ûÿæ
0030h: 06 08 E6 07 69 B0 08 B0 09 E6 F0 23 D6 07 FD 8B ; ..æ.i°.°.æð#Ö.ý<
0040h: DA 56 FA F7 76 FA 08 6B FB 56 FA F7 88 F0 56 E8 ; ÚVú÷vú.kûVú÷^ðVè
0050h: 7F 76 FA 10 6B FB 56 FA EF 89 F0 A6 E8 0D 6B 01 ; ùvú.kûVúïïð!è.k.
0060h: AF 8C 0A D6 08 51 8C 0D AF 00 00 4D 42 41 53 49 ; ù.Ö.QE. . . MBASI
0070h: 43 44 45 4D 4F 4E 53 54 52 41 54 49 4F 4E 0D 00 ; CDEMONSTRATION..
0080h: 0A 50 22 57 41 45 48 4C 45 4E 20 53 49 45 20 42 ; .P"WAEHLEN SIE B

```

```

0090h: 49 54 54 45 20 45 49 4E 20 50 52 4F 47 52 41 4D ; ITTE EIN PROGRAM
00a0h: 4D 42 45 49 53 50 49 45 4C 20 21 22 0D 00 14 50 ; MBEISPIEL !"...P
00b0h: 0D 00 1E 50 22 31 20 50 52 49 4D 46 41 4B 54 4F ; ...P"1 PRIMFAKTO
00c0h: 52 5A 45 52 4C 45 47 55 4E 47 22 0D 00 28 50 22 ; RZERLEGUNG"..(P"
00d0h: 32 20 55 4D 52 45 43 48 4E 55 4E 47 20 48 45 58 ; 2 UMRECHNUNG HEX
00e0h: 2D 44 45 5A 49 4D 41 4C 22 0D 00 32 50 22 33 20 ; -DEZIMAL"..2P"3
00f0h: 55 4D 52 45 43 48 4E 55 4E 47 20 44 45 5A 49 4D ; UMRECHNUNG DEZIM
0100h: 41 4C 2D 48 45 58 22 0D 00 3C 50 22 34 20 52 45 ; AL-HEX"..<P"4 RE
0110h: 47 49 53 54 45 52 49 4E 48 41 4C 54 20 4D 4F 44 ; GISTERINHALT MOD
0120h: 49 46 49 5A 49 45 52 45 4E 22 0D 00 46 50 22 35 ; IFIZIEREN"..FP"5
0130h: 20 4C 41 4E 47 53 41 4D 20 41 4C 50 48 41 42 45 ; LANGSAM ALPHABE
0140h: 54 20 44 52 55 43 4B 45 4E 22 0D 00 50 50 22 36 ; T DRUCKEN"..PP"6
0150h: 20 4E 45 55 20 42 45 47 49 4E 4E 45 4E 22 0D 00 ; NEU BEGINNEN"..
0160h: 5A 50 0D 00 64 49 22 50 52 4F 47 52 41 4D 4D 20 ; ZP..dI"PROGRAMM
0170h: 4E 52 20 3F 3A 20 22 41 0D 00 6E 47 31 35 30 2A ; NR?: "A..nG150*
0180h: 41 0D 00 96 4D 50 52 49 4D 46 41 4B 54 4F 52 5A ; A..-MPRIMFAKTORZ
0190h: 45 52 4C 45 47 55 4E 47 0D 00 A0 49 22 5A 41 48 ; ERLEGUNG.. I"ZAH
01a0h: 4C 3D 3F 20 22 41 0D 00 AA 4C 42 3D 32 0D 00 B4 ; L=? "A..ªLB=2..´
01b0h: 46 41 3C 32 3B 47 32 34 30 0D 00 BE 4C 43 3D 41 ; FA<2;G240..¾LC=A
01c0h: 2F 42 2A 42 0D 00 C8 46 43 3C 3E 41 3B 4C 42 3D ; /B*B..ÈFC<>A;LB=
01d0h: 42 2B 31 3B 47 31 39 30 0D 00 DC 50 42 0D 00 E6 ; B+1;G190..ÛPB..æ
01e0h: 4C 41 3D 41 2F 42 3B 47 31 38 30 0D 00 F0 50 22 ; LA=A/B;G180..øP"
01f0h: 46 45 52 54 49 47 22 0D 00 FA 47 31 30 30 0D 01 ; FERTIG"..úG100..
0200h: 2C 4D 55 4D 52 45 43 48 4E 55 4E 47 45 4E 0D 01 ; ,MUMRECHNUNGEN..
0210h: 36 49 22 48 45 58 5A 41 48 4C 3D 3F 20 22 41 0D ; 6I"HEXZAHL=? "A.
0220h: 01 40 50 22 44 45 5A 49 4D 41 4C 20 3D 20 22 41 ; .@P"DEZIMAL = "A
0230h: 0D 01 4A 47 31 30 30 0D 01 C2 49 22 44 45 5A 49 ; ..JG100..ÂI"DEZI
0240h: 4D 41 4C 5A 41 48 4C 3D 3F 20 22 41 0D 01 CC 48 ; MALZAHL=? "A..ÏH
0250h: 22 48 45 58 20 3D 20 22 41 0D 01 D6 47 31 30 30 ; "HEX = "A..ÖG100
0260h: 0D 02 58 4D 52 45 47 49 53 54 45 52 49 4E 48 2E ; ..XMREGISTERINH.
0270h: 20 4D 4F 44 49 46 49 5A 2E 0D 02 62 49 22 52 45 ; MODIFIZ...bI"RE
0280h: 47 49 53 54 45 52 20 4E 52 2E 3A 20 22 41 0D 02 ; GISTER NR.: "A..
0290h: 6C 48 22 49 4E 48 41 4C 54 20 3D 20 22 47 45 54 ; \H"INHALT = "GET
02a0h: 52 5B 41 5D 0D 02 76 49 22 4E 45 55 45 52 20 49 ; R[A]..vI"NEUER I
02b0h: 4E 48 41 4C 54 3A 20 22 42 0D 02 00 4F 53 45 54 ; NHALT: "B...OSET
02c0h: 52 5B 41 2C 42 5D 0D 02 8A 47 31 30 30 0D 02 EE ; R[A,B]..ŠG100..î
02d0h: 4D 41 4C 50 48 41 42 45 54 20 44 52 55 43 4B 45 ; MALPHABET DRUCKE
02e0h: 4E 0D 02 F8 49 22 57 41 52 54 45 5A 45 49 54 20 ; N..øI"WARTEZEIT
02f0h: 5A 57 49 53 43 48 45 4E 20 5A 57 45 49 20 42 55 ; ZWISCHEN ZWEI BU
0300h: 43 48 53 54 41 42 45 4E 20 5B 4D 53 45 43 5D 3A ; CHSTABEN [MSEC]:
0310h: 22 41 0D 03 02 4C 42 3D 32 36 3B 4C 43 3D 25 34 ; "A...LB=26;LC=%4
0320h: 31 0D 03 0C 4C 5A 3D 43 3B 53 31 30 30 30 0D 03 ; 1...LZ=C;S1000..
0330h: 16 4C 5A 3D 25 32 30 3B 53 31 30 30 30 3B 4D 53 ; .LZ=%20;S1000;MS
0340h: 50 41 43 45 20 44 41 5A 57 49 53 43 48 45 4E 0D ; PACE DAZWISCHEN.
0350h: 03 20 57 41 0D 03 2A 4C 43 3D 43 2B 31 3B 4C 42 ; . WA..*LC=C+1;LB
0360h: 3D 42 2D 31 0D 03 34 46 42 3C 3E 30 3B 47 37 38 ; =B-1..4FB<>0;G78
0370h: 30 0D 83 3E 4C 5A 3D 25 44 3B 53 31 30 30 30 3B ; 0.f>LZ=%D;S1000;
0380h: 4D 43 52 20 26 20 4C 46 20 41 4E 48 41 45 4E 47 ; MCR & LF ANHAENG
0390h: 45 4E 0D 03 48 4C 5A 3D 25 41 3B 53 31 30 30 30 ; EN..HLZ=%A;S1000
03a0h: 0D 83 52 47 31 30 30 0D 03 84 4D 50 52 4F 47 52 ; .fRG100...MPROGR
03b0h: 41 4D 4D 20 56 45 52 4C 41 53 53 45 4E 0D 03 8E ; AMM VERLASSEN..Ž

```



```
03c0h: 45 0D 03 E8 46 47 45 54 52 5B 25 46 41 5D 24 41 ; E..èFGETR[%FA]$A
03d0h: 25 31 30 3C 3E 25 31 30 3B 47 31 30 30 30 0D 03 ; %10<>%10;G1000..
03e0h: F2 4F 53 45 54 52 5B 25 46 41 2C 30 5D 3B 4D 52 ; ò0SETR[%FA,0];MR
03f0h: 45 51 55 45 53 54 20 52 55 45 43 4B 53 45 54 5A ; EQUEST RUECKSETZ
0400h: 45 4E 0D 03 FC 4F 53 45 54 52 5B 25 46 30 2C 5A ; EN..ü0SETR[%F0,Z
0410h: 5D 0D 04 06 52 0D 07 D0 45 0D FF FF FF FF FF FF ; ]...R..ĐE.ÿÿÿÿÿÿ
```

Bild 8.6. TINY-MPBASIC-Programm (verdichtete Form)

JU+TE-Hinweise

Entgegen dem obigen Beispiel aus dem Buch kodiert der JU+TE ein wenig anders:

- Zeilennummern, die in ihrer Hexadezimaldarstellung %0D (Zeilenende) oder %00 (Programmende) enthalten, sind nicht zulässig (z.B. Zeilennummer 0, 13, 256)
- nach 0D (Zeilenende) wird das 7. Bit des nachfolgenden Bytes (Zeilennummer) gesetzt. Die höchstmögliche Zeilennummer ist daher 32767.
- AND, OR, XOR werden mit \$A, \$O, \$X abgekürzt, vgl. a. Quellform Zeile 1000.

From:

<https://hc-ddr.hucki.net/wiki/> - **Homecomputer DDR**

Permanent link:

<https://hc-ddr.hucki.net/wiki/doku.php/elektronik/u883?rev=1512991181>

Last update: **2017/12/11 11:19**

