:

MPBASIC2

Auszug aus Claßen/Oefler, Wissensspeicher Mikrorechner-Programmierung, 4.Auflage, S. 204-212

Hinweis: In früheren Auflagen des Wissensspeichers Mikrorechner-Programmierung wurde nicht die finale Version des TINY-BASIC beschrieben, sondern die des Vorläufers UB881D-004. Dieser maskenprogammierte Einchipmikrorechner enthält die Version 2.0 des Tiny-MPBASIC, der U883 dagegen Version 3.0!

8. U883 TINY-MPBASIC

Der Einchipmikrorechner U883 enthält in seinem 2K Byte großen internen ROM einen einfachen BASIC-Interpreter. Daneben existiert ein dazugehöriger Editor- /Debuggerteil, der alle Funktionen, die für die Programmentwicklung notwendig sind, enthält. Diese Komponenten liegen entweder im externen ROM, sie können nach vollendeter Entwicklungsarbeit entfallen, oder sie existieren auf einem Wirtsrechner.

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 (Registerpointer auf %10: SRP #%10). 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 Setzen der Register 6 bis 9 erfolgt mit dem Registerpointerwert %00.

Das BASIC-Anwenderprogramm muss syntaktisch fehlerfrei sein. Es wird in verdichteter Form abgespeichert. Die Erstellung eines syntaktisch fehlerfreien Programms und der abarbeitbaren verdichteten Form erfolgt mit Hilfe des Editor/Debugger-Programmpakets.

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 %20.)

```
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		**Abkürzung**	**ASCII-Kode**
+	Addition	+	2B
-	Subtraktion	-	2D
*	Multiplikation	*	2A
/	Division	/	2F
$MOD	Modulo	$M	24 4D
$AND	logisches UND (bitweise)	$A	24 41
$OR	logisches ODER „	$O	24 4F
$XOR	exklusives ODER “	$X	24 58

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

```
je Procedur
  1 Byte Namenslänge
  1..5 Byte Procedurenamen
  2 Byte Adresse
```

```
Ende mit Namenslänge = FFh
```

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. Soll eine Anweisung mehrmals hintereinander mit verschiedenen Argumenten ausgeführt werden, dann genügt es, den Namen einmal aufzuschreiben und die verschiedenen Argumente durch Kommas zu trennen. 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 wurde zu Verhinderung eines Stacküberlaufs auf 15 beschränkt.

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

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

Für Vergleichsoperator kann dabei stehen:

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

Falls die nach IF folgende Bedingung wahr ist. so wird die folgende Anweisung ausgeführt und der Rest übersprungen. Andernfalls wird mit die mit ELSE beginnende Zeile abgearbeitet.

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. Wird die PRINT-Anweisung (bzw. PRINTHEX) mit einem Komma abgeschlossen, unterbleibt die Ausgabe des Zeilenvorschubs (%0D).

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

Die INPUT-Anweisung gibt, falls vorhanden, erst den angegebenen Text aus und weist den eingegebenen Wert der spezifizierten Variable zu. Die Eingabe kann sowohl dezimal als auch hexadezimal (% vor dem Wert) erfolgen. Bei fehlerhafter Eingabe wird ein Fragezeichen ausgegeben und eine neue Eingabe erwartet. INPUT ist auch als Funktion verfügbar.

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 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 -1 (entspricht %FFFF) 65536-mal durchlaufen. Bei einer Taktfrequenz von 8 MHz dauert ein Durchlauf eine Millisekunde. Die Zeitangabe ist nicht ganz exakt, da für den Aufruf der Zeitschleife eine gewisse Zeit benötigt wird.

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.

Mit Hilfe der TRAP-Anweisung kann laufend das Erfülltsein einer Bedingung getestet werden.

```
TRAP Bedingung TO Ausdruck
```

Nach dieser Anweisung prüft der Interpreter vor der Abarbeitung jeder neuen Zeile, ob die angegebene Bedingung erfüllt ist. Im positiven Fall wird zur TRAP-Routine verzweigt, deren Anfangsadresse durch Ausdruck bestimmt wird. Eine TRAP-Routine ist ein Unterprogramm, das mit Return endet. Sobald eine TRAP-Bedingung eingeschaltet ist, verlangsamt sich die weitere Programmabarbeitung. Durch die Anweisung

```
CLTRP
```

wird eine aktive TRAP-Bedingung gelöscht.

Fest integriert im TINY-MPBASIC-Interpreter sind folgende Standard-Prozeduren und -Funktionen:

ABS [Parameter]	absoluter Betrag
NOT [Parameter]	logische Negation (bitweise)
GTC	Zeichen mittels GET_CHAR holen
INPUT	Zahl mittels GET_CHAR holen
PTC [Parameter]	Parameter mittels PUT_CHAR ausgeben
RL [Parameter]	links rotieren
RR [Parameter]	rechts rotieren
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**	**ASCII-Code**
LET	L	4C
GOTO	G	47

Anweisung	Abkürzung	ASCII-Code
GOSUB	S	53
RETURN	R	52
IF..THEN	F…;	46…3B
ELSE…	>; …	3E 3B
PROC	O	4F
INPUT	I	49
PRINT	P	50
PRINTHEX	H	48
STOP	T	54
END	E	45
REM	M	4D
WAIT	W	57
CALL	C	43
TRAP…TO	!…,	21…2C
CLRTRP	/	2F

Der restliche Programmtext wird ohne Leerzeichen in ASCII-Zeichen (die Zeilennummer als 2 Byte große Hexadezimalzahl) abgespeichert. Das Zeilenende wird durch %0D und das Programmende durch %00 gekennzeichnet.

Beispiel

Auszug aus Claßen/Oefler, Wissensspeicher Mikrorechner-Programmierung, 2. Auflage, S. 172-180

Hinweis: das nachfolgende Beispiel erschien im Original nicht für die endgültige Version 3.0 des TINY-MPBASIC des U883, sondern die Vorversion für den UB881D-004 mit Tiny-MPBASIC 2.0.

Untschiede im MC-Code zw. UB881D-BM004 und U883:

1. Die Zeilennummernkodierung ist z.B. abweichend (kein Hi-Bit gesetzt)
2. statt THEN (;) wird ein Komma verwendet
3. kein ELSE
4. völlig andere interne Implementierung, andere Registernutzung
5. GET_CHAR Übergabe R3 statt R8
6. PUT_CHAR Übergabe R5 statt R8
7. u.a.m.

Die finale Beschreibung des Tiny-MPBASIC U883 ist bei MPBASIC, in rfe 3/1985, im Wissensspeicher 4. Auflage sowie in der Doku zum JU+TE zu finden.

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.

```
                    01 U883TEST MODULE
                    02
                    03     ! MOEGLICHES ANWENDERPROGRAMM IM EXT. SPEICHER !
                    04 INTERNAL
                    05         EINTRITT PROCEDURE
                    06           ENTRY
P 0812              07           $ABS %812     ! EINTRITTSPUNKTE !
P 0812 8D 081B      08             JP BEGIN
P 0815 8D 0841      09             JP GET_CHAR
P 0818 8D 0851      10             JP PUT_CHAR
                    11         END EINTRITT
                    12
                    13     ! INITIALISIERUNG UND AUFRUF DES INTERPRETERS !
                    14
P 081B              15         BEGIN PROCEDURE
                    16           ENTRY
P 081B 8C  96       17             LD R8,#%(2)10010110  ! NORMAL EXT. TIMING
!
P 081D E6  FF  80   18             LD SPL,#%80
P 0820 E6  F4  40   19             LD TO,#%40          ! TIMER FUER 300 BAUD
!
P 0823 E6  F7  49   20             LD P3M,#%(2)01001001
P 0826 E6  F5  0D   21             LD PRE0,#%(2)00001101
P 0829 E6  F1  43   22             LD TMR,#%(2)01000011
P 082C B0  FB       23             CLR IMR
P 082E 9F           24             EI
P 082F E6  06  08   25             LD 6,#HI BASICPROGRAMM
P 0832 E6  07  69   26             LD 7,#LO BASICPROGRAMM
P 0835 B0  08       27             CLR 8           ! KEINE EXT. PROZEDUR !
P 0837 B0  09       28             CLR 9
P 0839 E6  F0  23   29             LD SIO ,#'#'    ! PROMPT-ZEICHEN  !
P 083C D6  07FD     30             CALL %7FD       ! BASIC INTERPRETER !
P 083F 8B  DA       31             JR BEGIN
P 0841              32         END BEGIN
                    33
                    34     ! ZEICHEN HOLEN BZW. SENDEN
                    35       FERNSCHREIBER MIT FULL-DUPLEX !
                    36
P 0841              37         GET_CHAR PROCEDURE
                    38           ENTRY
P 0841 56  FA  F7   39             AND IRQ,#%F7    ! ALTEN REQUEST RUECKS. !
P 0844 76  FA  08   40 GTC10:      TM IRQ,#%8      ! ZEICHEN DA ?          !
P 0847 6B  FB       41             JR Z,GTC10
P 0849 56  FA  F7   42             AND IRQ,#%F7    ! REQUEST RUECKSETZEN   !
P 084C 88  F0       43             LD R8,SIO
P 084E 56  E8  7F   44             AND R8,#%7F     ! PARITAET RUECKSETZEN  !
P 0851              45         END GET_CHAR
                    46
```

```
P 0851                47         PUT_CHAR PROCEDURE
                      48           ENTRY
P 0851 76  FA  10     49             TM IRQ,#%10     ! LETZTES ZEICHEN RAUS? !
P 0854 6B  FB         50             JR Z, PUT_CHAR
P 0856 56  FA  EF     51             AND IRQ,#%,EF   ! REQUEST RUECKSETZEN   !
P 0859 89  F0         52             LD SIO,R8
P 085B A6  E8  0D     53             CP R8,#'%r'
P 085E 6B  01         54             JR Z,PTC10
P 0860 AF             55             RET
P 0861 8C  0A         56 PTC10:      LD R8,#'%l'
P 0863 D6  0851       57             CALL PUT_CHAR   ! AUTO LINEFEED         !
P 0866 8C  0D         58             LD R8,#'%r'
P 0868 AF             59             RET
P 0869                60         END PUT_CHAR
                      61
                      62 $SECTION PROGRAM
                      63
                      64     ! BEGINN DES BASICPROGRAMMS !
P 0869                65         BASICPROGRAMM ARRAY[0 BYTE]
                      66 END U883TEST
```

Bild 8.4. Assemblerprogramm für TINY-MPBASIC

```
1  REM    BASIC DEMONSTRATION
10 PRINT "WAEHLEN 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   END
```

Bild 8.5. TINY-MPBASIC-Programm (Quellform)

```
0000h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ; ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0010h: FF FF 8D 08 1B 8D 08 41 8D 08 51 8C 96 E6 FF 80 ; ÿÿ□..□.A□.QŒ–æÿ€
0020h: E6 F4 40 E6 F7 41 E6 F5 0D E6 F1 43 B0 FB 9F E6 ; æô@æ÷Aæõ.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÷ˆðVè
0050h: 7F 76 FA 10 6B FB 56 FA EF 89 F0 A6 E8 0D 6B 01 ; □vú.kûVúï‰ð¦è.k.
0060h: AF 8C 0A D6 08 51 8C 0D AF 00 00 4D 42 41 53 49 ; ¯Œ.Ö.QŒ.¯..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..ÜPB..æ
01e0h: 4C 41 3D 41 2F 42 3B 47 31 38 30 0D 00 F0 50 22 ; LA=A/B;G180..ðP"
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..ÌH
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 ; lH"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.ƒ>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 ; .ƒRG100..„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 ; òOSETR[%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..üOSETR[%F0,Z
0410h: 5D 0D 04 06 52 0D 07 D0 45 0D FF FF FF FF FF FF ; ]...R..ÐE.ÿÿÿÿÿÿ
```

Bild 8.6. TINY-MPBASIC-Programm (verdichtete Form, Format U881 BM004)

```
todo
```

Bild 8.6. TINY-MPBASIC-Programm (verdichtete Form, Format U883 final)

- u88x-emr-beispiel.zip

obiges Beispielprogramm im Original und auch in Umsetzung auf Arnold-Assembler + JTCEMU

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

Permanent link:
https://hc-ddr.hucki.net/wiki/doku.php/elektronik/u883/mpbasic2

Last update: **2025/04/25 12:04**