U883

🔧 Fix Me!

Der U883 ist ein spezieller Vertreter der U881-Familie der Einchipmikrorechner (EMR) der DDR. Der U883 ist maskenprogammierter 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. Assemberquellcode

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, Wissensspeicher 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, Wissensspeicher 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 Möglichlkeit, 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 "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   ENDE
```

Bild 8.5. TINY-MPBASIC-Programm (Quellform)