Tiny-Basic

Mt dem Z1013 wurde ein kleiner 3K-BASIC-Interpreter „robotron Z1013 BASIC 3.01“ ausgeliefert (als Hexdump in der Bedienungsanleitung/Anlagenteil oder auf Kassette M0111).

Start: J 100
Restart: J 103
Speicher: 100h-0BFFh

Das TINY-BASIC ist im Handbuch Teil IIA beschrieben.

Es gibt eine Vielzahl modifizierter Versionen der originalen Version 3.01 von Riesa. 3 Versionen werden hier vorgestellt (s. Inhaltsverzeichnis)

Geschichte

Das TINY-Basic stammt vom Palo Alto Tiny BASIC ab, s. Tiny_BASIC und Autor Li-Chen_Wang. Dessen Implementation des TINY-Basic für den 8080-Prosessor war wiederum die Basis für die Tiny-Basic Version von Rolf-Dieter Klein (→ BASIC für 8080-Systeme).

Ich vermute, dass ebendiese Version die Grundlage für das Z1013-Tinybasic ist. Das **„3K-BASIC VON RER“** (Robotron-Elektronik Riesa) kennt beispielsweise wie bei Rolf-Dieter Klein hinzugekommenen Befehle wie TOP und LEN, und nutzt 0-terminierte Zeichenketten für die BASIC-Befehle[1)].

In Details gibt es aber durchaus Abweichungen: Die Zeichentestfunktion TSTC arbeitet mit einer 2-Byte-Adresse anstelle eines 1-Byte-Offsets zur nächsten Funktion bei negativem Testergebnis. Neu sid auch die Befehle BYTE und WORD.

2018: Die Vermutung ist bestätigt! Auch das Minibasic des AC1 basiert auf dem RDK-Basic: ein Entwickler des AC1 hat später den Z1013 mitentwickelt und die Software des AC1 dabei als Grundlage genommen. Daher hat auch das Betriebssystem des Z1013 viele Gemeinsamkeiten mit dem des AC1 (und ZETBUG).

Befehlsübersicht

	Abkürzung	**Bemerkung**
Kommandos		
LIST [n]	L.[n]	Listen(20 Zeilen ab >=n)
RUN	R.	
NEW	N.	
BYE	B.	
END n	E.n	Speicherende setzen
CSAVE„name“	C.„name	
CLOAD	CL.	
Befehle		
FOR	F.	

	Abkürzung	**Bemerkung**
Kommandos		
TO	T.	
NEXT	N.	
IF	I.	
GOTO	G.	
GOSUB	GO.	
RETURN	R.	
PRINT	P.	
INPUT[“…„]a	I.[“…„]a	
LET	L.	
REM	RE.	
CALL	C.	
POKE	PO.	
OUTCHARn	O.n	
OUT(n)	OU.(n)	
TAB(n)	T.(n)	Ausgabe n Leerzeichen
BYTE(n)	B.(n)	Ausgabe 2stellige Hexzahl
WORD(n)	W.(n)	Ausgabe 4stellige Hexzahl
O$(n)		Ausgabe Zeichenkette von (n)
I$(n)		Einlesen Zeichenkette nach (n)
Anweisungen		
RND(n)	R.(n)	Zufallszahl 1…n
ABS(n)	A.(n)	
PEEK(n)	P.(n)	
IN(n)		
INCHAR	I.	ein Zeichen einlesen
STEP	ST.	
HEX n	H.n	Wandlung Hex–>Dez
TOP	T.	erster freier Speicher nach BASIC
SIZE	S.	freier BASIC-Speicher
'a'		ASCII-Code von a
C		
@( )		eindimensionales Feld

Das Programm wird im RAM von 1152H bis (101FH) abgelegt, aber bereits ab 1000H abgeSAVEt. Jede Zeile hat folgenden Aufbau:

```
    2 byte Zeilennummer
    x byte Text
    1 byte Zeilenabschluss (0DH)
```

TINY-BASIC 4.01

Version von V. Pohlers, 1989

Das BASIC 3.01 von Riesa wurde um einige Befehle erweitert bzw. verbessert:

LIST (n) funktioniert jetzt analog HC-BASIC

EDIT n - einfacher Zeileneditor

der Kursor muss hinter das letzte Zeichen gebracht werden. Eine Änderung der Zeilennummer ist möglich, leider noch nicht INS/DEL

HSAVE -Aufruf des HEADER-SAVE ueber Sprungverteiler, TYP b

HLOAD -Einlesen von HEAD- Programmen, Aufruf ueber Sprungverteiler

TINY-BASIC 4.01 belegt den Speicherbereich: 100H-0C41H

Damit sind einige Programme, die das freie Kilobyte 0C00H-1000H nutzen, nicht mehr lauffähig. Dieser Bereich ist von robotron eigentlich für Erweiterungen reserviert, in eigenen Programmen also bitte n i c h t nutzen.

3K-rs-BASIC V. 1.12

Version von Uwe Rehn, 03/1988

Programmstandort: 100h - DBBh
Programmstart : 100h , 103h

Der 3k-rs-BASIC-Interpreter enthaelt als Kern den Orginal-3k- BASIC-Interpreter von RIESA. Alle Aenderungen und Ergaenzungen wurden so vorgenommen, dass bisherige Programme ohne Einschraenkungen ladbar und lauffaehig sind. Um alle neuen Befehle nutzen zu koennen, sind das Programm „HEADERSAVE“ und ein physischer Druckertreiber erforderlich.

Folgende Befehle wurden geaendert bzw. neu aufgenommen:

HSAVE (HS.)

Mit diesem Befehl werden BASIC-Programme ueber die TBG-Routine des HEADERSAVE auf Kassette abgelegt.

Synt.: HSAVE (ENT) , Titel

HLOAD (HL.)

Mit diesem Befehl koennen die mit dem neuen „HSAVE“ abgelegten Programme geladen werden. Vorher vorhandene Programme werden geloescht.

Synt.: HLOAD (ENT) , Titel

Wird kein Titel angegeben, so wird das erste BASIC-Programm geladen.

LIST (L.)

Es werden 16 Zeilen ab angegebener Zeilennumer gelistet. Mit ENT wird fortgesetzt, mit BRAEK

abgebrochen.

Synt.: LIST (Znr.)

LLIST (LL.)

Mit „LLIST“ koennen Programmlistings an einen Druckertreiber ausgegeben werden. Dazu wird nach Aufruf zuerst der Druckertreiber auf 11Dh mit „CALL DRINI“ initialisiert. Danach werden alle Zeichen ueber Adresse 121h mit „CALL DRAKK“ an den Druckertreiber uebergeben.

Synt.: LLIST

HELP (H.)

Mit diesem Befehl wird eine Liste der vorhandenen Befehlsworte ausgegeben.

Synt.: HELP

EDIT (E.)

Es erfolgt ein Editieren einer Zeile. In der Zeile koennen Aenderungen durch Ueberschreiben, durch INSERT (12h) oder durch DELETE (10h) durchgefuehrt werden. Mit ENT werden alle Zeichen bis zum Cursor als neue Zeile uebernommen. Mit BRAEK (03h) erfolgt keine Uebernahme. Die Zeilennummer kann auch geaendert werden.

Synt.: EDIT Znr.

LPRINT (LP.)

Entsprechend dem PRINT-Befehl erfolgt eine Bildschirm- und Druckerausgabe.

Synt.: LPRINT (weiter wie PRINT)

Zusammenfassung der neuen Rufe:

```
10E :  CD C1 FA     HSAVE (CALL SARUF)
112 :  CD A4 FB     HLOAD (CALL LORUF)
11D :  CD 00 E8     CALL DRINI
121 :  CD 09 E8     CALL DRAKK
```

TINY-BASIC 3.20H

von Kraft/IG-HC TU Dresden

Mit HEADERSAVE und einem MENUE (Auflistung aller BASIC-Befehle, die implementiert sind)

empfohlen als Standard (→ Informationen). 3K-rs-BASIC is. m.E. besser!

Erfahrungen

von mir, 1991

Das Z1013-BASIC

Obwohl die Original-Version von Riesa kaum noch genutzt wird, es gibt ja die Version 3.20 von Uwe Rehn (?) o. die 4.01er, war es dennoch für mich interessant, dieses BASIC mal zu analysie- ren. Wer sich für Computergeschichte interessiert, dem sei ge- sagt, daß dieses Basic warscheinlich von Herrn Rolf-Dieter Klein entwickelt wurde und seit der meines Wissens ersten Veröffent- lichung im Oktober 1978 in 'Hobby Computer', Sonderheft der ELO, Funkschau, Elektronik; Franzig-Velag auf viele Systeme übernom- men wurde. Das Tiny-BASIC nutzt keine speziellen U880-Befehle, sondern basiert lediglich auf dem Befehlssatz eines 8080-Prozessors. Auch daraus resultiert die Lahmheit dieses Interpreters. Offen- sichtlich war man in Riesa bei der Übernahme des Basics nicht bereit, hier einiges zum Vorteil zu ändern. so wird beispiels- weise der Befehl SUB HL,DE als Unterprogramm mit 8bit-Registern ausgeführt. Interessierten empfehle ich, sich einmal die Rou- tinen OUT oder IN anzusehen. Sie stehen ab 0A32H bzw. 0A69H. Es wird ein kleines Maschinenprogramm zu Ansprechen der Portadresse in den RAM 'gepokt'; der 8080 kennt da zuwenig Befehle, maß mußte sich auf diesem Prozessor mit solch einem Trick helfen. Dafür hat Riesa gleich mehere Sachen mit eingebaut, die in keiner Beschreibung erwähnt werden. So folgt dem Basic-Interpre- ter ab Adresse 0B86H eine kurze Tastaturpollingroutine, die die kleine 8×4-Tastatur d i r e k t abfragt und bei S4-S für eine kurze Zeit in einer Warteschleife verharrt, bei S4-K dage- gen das laufende Basicprogramm abbricht und in die Grundschleife zurückkehrt. Bei exotischen Tastaturen, die keine solche Direkt- abfrage vertragen, muß diese Routine z.B. mit einem RETurn kurz- geschlossen werden! Auch die PRINT-Funktion ist recht interessant. Beim Ausgeben von Texten können diese sowohl in “...„ als auch in '...' einge- schlossen werden. Dem Zeichen _ kommt hier eine eigenartige Funktion zu: Durch PRINT _ (ohne Apostroph o.ä.) wird ein zweifacher Zeilenvorschub ausgeführt. Die Basic-Befehle können wesentlich stärker gekürzt werden als im Handbuch angegeben; das führt zu einem Geschwindigkeitsan- stieg, leider aber auch zu geringerer Lesbarkeit. Hier sind die maximal möglichen Abkürzungen:

Kommandos

```
LIST (n)        .(n)     Listen(20 Zeilen ab >=n)
RUN             R.
NEW             N.
BYE             B.
END n           E.n       Speicherende setzen
CSAVE"name"     C."name
CLOAD           CL.
```

Prozeduren

```
FOR             F.
TO              T.
NEXT            .              (es reicht wirklich der Punkt)
IF              I.
GOTO            G.
GOSUB           GO.
```

```
RETURN          R.
PRINT           P.
INPUT("...")A   IN.("...")A
LET             L.
REM             RE.           (hat zwar auch drei Zeichen,
CALL            C.             ist aber schneller!)
STOP            S.
POKE            PO.
OUTCHARn        O.n
OUT(n)          OU.(n)
TAB(n)          T.(n)     Ausgabe n Leerzeichen
BYTE(n)         B.(n)     Ausgabe 2stellige Hexzahl
WORD(n)         W.(n)     Ausgabe 4stellige Hexzahl
O$(n)                     Ausgabe Zeichenkette von (n)
I$(n)                     Einlesen Zeichenkette nach (n)
```

Funktionen

```
RND(n)          .(n)     Zufallszahl 1...n
ABS(n)          A.(n)
PEEK(n)         P.(n)
IN(n)
INCHAR          I.        ein Zeichen einlesen
STEP            S.
HEX n           H.n       Wandlung Hex-->Dez
TOP             T.        erster freier Speicher nach BASIC
SIZE            S.        freier BASIC-Speicher
LEN             L.
```

Und nun der Test: Was macht folgendes Programm?

```
 5 P._,_,_
10 F.I=1T.10S.2
20 P.I;.I
30 P..(8)
```

Richtig, es liest sich so:

```
 5 PRINT; PRINT; PRINT; PRINT; PRINT; PRINT
10 FOR I=1 TO 10 STEP 2
20 PRINT I; NEXT I
30 PRINT RND(8)
```

Zum internen Aufbau der Tiny-Basic-Programme: Das Programm wird im RAM von 1152H bis (101FH) abgelegt, aber bereits ab 1000H abgesavet. Jede Zeile hat folgenden Aufbau:

```
2 byte Zeilennummer
x byte Text in ASCII-Notation
1 byte Zeilenabschluss (0DH)
```

Der Bereich von 1000H bis 1151H dient als Arbeitsspeicher und enthält unter anderem:

```
1006H-1008H: Maschinencode bei IN/OUT
101FH:       Programmende
1049H-1113H: Basic-Stack
1115H-114BH: Variablenspeicher A..Z
114CH:       Speicherende für Basic-Programm
114EH:       Zeiger auf Eingabepuffer
1150H:       Ende Eingabepuffer
```

```
1152H:       Beginn Basic-Programm
```

Der Eingabepuffer liegt standartmäßig von 3094H bis 30D6H; wird mit END eine neue Speicherobergrenze definiert, so wird der Eingabepuffer vor diesem neuen Ende in einer Größe von 132 Byte installiert. Die Feldvariablen @(.) liegen ab (114CH) abwärts im Speicher.

Ich denke, das reicht zum Tiny-Basic.

[1)]
Im Palo Alto Tiny BASIC wird das Ende durch ein gesetztes 7. Bit gekennzeichnet