BASIC/DEBUG

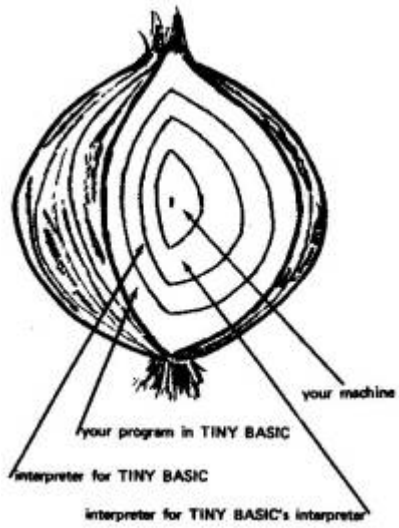
BASIC/DEBUG ist eine Tiny-BASIC-Implementierung für den Z8. Entstanden ist sie vermutlich 1979/1980.
In nur 2 KByte ROM steckt ein vollwertiges BASIC mit folgenden Eigenschaften:

- Editor
- 15 BASIC-Kommandos
- 16-Bit Ganzzahlarithmetik
- Punkt-vor-Strich-Rechnung
- Hexadezimalzahlen
- Direktzugriff auf Register und Speicher

Das BASIC ist im Handbuch **z8671_basic_debug.pdf** ausführlich beschrieben.

Das Basic orientiert sich sehr stark an das TINY-BASIC von Tom Pittman (http://www.ittybittycomputers.com/ittybitty/tinybasic/). Literatur und Programme zu den dort beschriebenen BASICs für 1802, 6502, 6800-Controller passen auch zum BASIC/DEBUG des Z8671.

Aufbau



2021 habe ich den ROM vollständig und umfassend reassembliert, analysiert und dokumentiert:

BASIC/DEBUG ist in einer eigenen Codesprache geschrieben: TBIL (Tiny Basic Interpreter Language) Tiny_BASIC#Implementation_in_a_virtual_machine.

Das BASIC selbst ist nur 432 Byte groß. In Z8-Code ist der Interpreter für diese Codesprache enthalten sowie Initialisierung und In/Out-Code. Diese Art der Implementierung ist extrem speichersparend. Nur so konnten die umfangreichen Fähigkeiten des BASIC/DEBUG in nur 20248 Byte untergebracht werden.

Das Zwiebelmodell beschreibt sehr schön den Ablauf: Das BASIC-Anwenderprogramm wird vom BASIC-Interpreter (in TBIL) interpretiert. Dieser wiederum wird vom TBIL-Interpreter (in Z8-Code) interpretiert, und das ganze läuft auf einem Z8-Prozessor und kommuniziert mit der Außenwelt.

BASIC/DEBUG ist vielfältig konfigurierbar und bietet u.a. folgende Optionen

- beliebige RAM-Unterstützung oder reine ROM-Version
- mehrere BASIC-Programm im Speicher
- Autostart eines Programms nach RESET
- Editierzeichen für DEL, Zeilenende-Zeichenfolge frei vergebbar
- eigene I/O-Treiber anstelle der Z8-SIO

Literatur

- z8671_basic_debug.pdf von Zilog. (der Hochpfeil in der Doku ist das Zeichen „^“)

Literatur zu Tiny-BASIC:

orig. Scans

- http://retro.hansotten.nl/uploads/files/tiny%20basic%20manual.pdf
- http://retro.hansotten.nl/uploads/files/tiny%20basic%20exp%20kit.pdf
- http://retro.hansotten.nl/uploads/files/tiny%20basic%20getting%20the%20most%20of.pdf

HTML:

- TINY BASIC User Manual
  http://www.ittybittycomputers.com/ittybitty/tinybasic/TBuserMan.htm

Beispielprogramme:

- TBprogs.zip, Advent.txt, Euphoria.txt, LifeTB.txt
  http://www.ittybittycomputers.com/ittybitty/tinybasic/
- The First Book of Tiny BASIC Programs
  https://www.retrotechnology.com/memship/Son_of_TFBOTBAS.HTM
- Tiny Basic Experimenters Kit
  http://www.ittybittycomputers.com/ittybitty/tinybasic/TBEK.txt
- Buch „Mikroprozessoren in der Meß- und Regeltechnik. Funktion - Aufbau und Programmierung“ von Gerhard Ledig, Franzis-Verlag 1988
- originale Scans und Beispiele und weitere Dokumente
  http://retro.hansotten.nl/6502-sbc/kim-1-manuals-and-software/kim-1-software/

Beispiel

Das folgende Programm zeigt eine Autostart-BASIC-Programm und eigene I/O-Treiber:

Damit Autostart funktioniert, muss das Programm auf Adresse 1020h beginnen. Die Zeilennummer muss < 256 sein (d.h. das erste Byte == 0).

Im Zeile 1 wird Register %FB (Interrupt mask register) gesetzt. Dadurch werden die externen I/O-Teiber aktiviert. Nachfolgende PRINT-Befehle geben dann über diese Treiber aus. Als Beispiel wurde zu Testzwecken ein einfacher Ringpuffer implementiert.

External I/O drivers must conform to the following requirements: They must pass a single ASCII input or output character in R19 with the register pointer set to R16-31. Registers R4-R15 and R22-32 must be preserved. Location %1012 must contain a jump to the character input driver, and %1015 a jump to the character output driver. The input routine must do any necessary echoing, and the parity bit (bit 7) of all input characters must be set to 0. Drivers return to Basic/Debug with a RETURN instruction (hex AF)

```
    cpu    z8601
    org 1000h
; interrupt routines
    jp    0    ; IRQ 0
    jp    0    ; IRQ 1
    jp    0    ; IRQ 2
    jp    0    ; IRQ 3
    jp    0    ; IRQ 4
    jp    0    ; IRQ 5
; character io
        org 1012h
    jp    CI
    jp    CO
; Basic-Pgm
    org 1020h
    db    0,5,"@251=0:REM EXTERNAL IO",0
    db    0,10,"LET A=%1234:PRINT HEX(A)",0
    db    0,20,"STOP",0
    db    0ffh,0ffh
; input driver
CI: ret
; output driver
; use ram ringbuffer 8000h..9000h for simple output
CO: srp    #60h
    cp    r0, #90h
    jr    ge, co2
    cp    r0, #80h
    jr    ge, co1
co2:    ld    r0,#80h
    ld    r1,#0
co1:    ld    r3,13h
    ldc    @rr0,r3
    incw    rr0
    srp    #10h
    ret
```

eigene TBIL

Nicht im Handbuch erwähnt ist die Möglichkeit, einen eigenen BASIC-Interpreter (in TBIL) anstelle des im ROM enthaltenen zu nutzen: Die Adresse des Interpreters steht im Constant Block in Byte 0 und 1. Wird ein externer Constant Block genutzt, kann ein eigener/erweiterter BASIC-Interpreter angesprochen werden. Es ist sogar möglich, maximal drei eigene TBIL-Codes (19, 1E, 1F) zu definieren und zu verwenden! Diese drei OP-Codes sind original nicht genutzt. Auf den Adressen 1018h, 101Bh, 101Eh (nur 2 Byte) sind Sprünge zu den zugehörigen Implementierungen zu setzen.

```
    org 1018h
; IL Extension
    jp    xxxx    ; IL_19
    jp    xxxx    ; IL_1E
    jr    xxxx    ; IL_1F    101Eh; nur 2 Byte, sonst memory overlapping mit
Autostart-Programm
```