Programmieren

Für den Z9001 stehen diverse Programmiersprachen bereit:

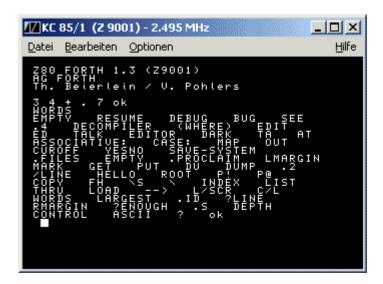
- BASIC
- FORTH
- PASCAL
- Assembler

BASIC

Der Einsteiger sollte mit BASIC beginnen. Mit einem BASIC-ROM-Modul oder mit dem eingebauten BASIC des KC 87 2.x kann man gleich nach dem Einschalten mit der Arbeit beginnen. Das BASIC wird ausführlich im **Programmierhandbuch** und im **Anhang zum Programmierhandbuch** beschrieben (s. Handbücher und Downloads der Handbücher bei http://www.sax.de/~zander/.

FORTH

Es gibt verschiedene FORTH-Versionen. Bekannt ist das f.i.g.-FORTH **FORTH** und das FG FORTH83 d. DDR **F83**. Beide sind im Mega-Flash-Modul enthalten. F83 ist deutlich umfangreicher und schneller.



Beim KC-Club Treffen 2012 gab es eine kleine Einführung ins FG-FORTH FORTH83 z9001 f83.pdf.

PASCAL

Für den KC gibt es das KC-PASCAL, eine Variante des bekannten Hisoft-PASCAL, sowie als 32K-Modul ein Turbo-Pascal-ähnliches Pascal von der TH Leipzig (s. weitere Module, das Handbuch findet man bei http://www.sax.de/~zander/).

Assembler

Zur vollständigen und systemnahen Programmierung eignet sich Assembler. robotron bietet mit EDAS und IDAS gleich zwei Assembler an. Mit **ZSID** und **R80** stehen außerdem Debugger und Reassembler im Mega-Flash-Modul zur Verfügung.

Allerdings ist das Programmieren in Assembler um einiges komplexer und schwerer als in den "höheren" Programmiersprachen.

Als Basis sollten unbedingt die Beschreibung des Betriebssystems incl. Betriebssystemlisting **robotron Betriebssystem KC 85/1 (Z9001)** studiert werden. Das Handbuch findet man bei http://www.sax.de/~zander/.

Systemfunktionen

Zur systemunabhängigen Programmierung werden vom Betriebssystem 33 Systemrufe bereitgestellt. Diese werden analog CP/M über CALL 0005 aufgerufen. Die Auswahl des gewünschten Systemrufes erfolgt über das C-Register, dessen Inhalt den Systemruf adressiert. Verschiedene Systemrufe erwarten Eingabeparameter bzw. liefern Parameter zurück.

Eingabeparameter:

- Bytewerte im E -Register
- Wortwerte im DE-Register

Ausgabeparameter:

- Bytewerte im A -Register
- Wortwerte im BC-Register

Wichtige Systemrufe:

Rufnr.	Name	Funktion
01	CONSI	Eingabe eines Zeichens von CONST
02	CONSO	Ausgabe eines Zeichens zu CONST
09	PRNST	Ausgabe einer Zeichenkette zu CONST
10	RCONB	Eingabe einer Zeichenkette von CONST
11	CSTS	Abfrage Status CONST
17	GETCU	Abfrage logische und pyhsische Cursoradresse
18	SETCU	Setzen logische Cursoradresse

Der OS-Rahmen

Damit eigene Programme vom OS aus gestartet werden können, wird ein spezieller Code benötigt, der sogenannte OS-Rahmen. Damit erscheinen Programme als transiente Kommandos im OS und können über den Kommandoname aufgerufen werden. Außerdem können Parameter übergeben werden (s. z.B. Code von OS-SAVE).

Ein OS-Rahmen muss auf einer integralen 100H-Grenze (300h ... 0BF00h) beginnen. Es können beliebig viele Kommandos in einem OS-Rahmen angegeben werden.

```
ORG
           xx00h
    JP
          AUSF
                       ;Sprung zur Kommandoausführung1
                          ;Kommandoname1 (im OS-Mode einzugeben)
    DB
          'NAME
                   ',0
                ;8 Zeichen, ggf. mit Leerzeichen auffüllen, Null-Byte
    JP
                        ;Sprung zur Kommandoausführung2
          AUSF2
                            ;Kommandoname2 (im OS-Mode einzugeben)
          'NAME2
    DB
                     ',0
    DB
          0
                    :Kennzeichen OS-Rahmen Ende
AUSF:
```

Kommandozeile

Die eingegebene Kommandozeile wird im Puffer CONBU abgelegt, am Ende wird ein Nullbyte angefügt. Mit GVAL werden die einzelnen Parameter nacheinander geholt, d.h. nach INTLN kopiert und in CONBU wird der Parameter durch Leerzeichen ersetzt. Nach Start des Programms ist der Programmname bereits nicht mehr in CONBU lesbar¹⁾.

erneuter Aufruf

Ein Warmstart kann erkannt werden, indem beim Einsprung HL auf den Einsprungswert verglichen wird (AUSF, AUSF2,...). Ist er gleich, wurde das Programm nicht von Kassette geladen (oder vom Megamodul), sondern ist bereits im Speicher abgelegt und wurde von dort gestartet (mittels Routine CPROM).

Beispiele

Folgendes Programm gibt den Text "Hallo User!" auf den Bildschirm aus. Das Programm wird mit dem Kommando TEST gestartet.

```
cpu
           z80
            300h
    org
Beispiel:
;Löschen Bildschirm in Hintergrundfarbe blau
;Ausgabe einer Kopfzeile in der Farbe rot
; Start im OS mit TEST
    jр
          main
                    " , 0
          "TEST
                           ; 8 Zeichen; Ende der Zeichenkette
    db
    dh
                        ; Ende des Headers
              de, text
main:
        ld
    ld
          c,9
    call
            5
          0
    jр
```

```
:Zeichenkettendefinition
TEXT: DB 15H ;Farbs
DB 4 ;Farbe BLAU
                          ;Farbsteuercode Hintergrund
                    ;Code für CLEAR SCREEN
   DB OCH
   DB 14H
                      ;Farbsteuercode Vordergrund
   DB 1
                    ; Farbe ROT
   DB "Hallo User!"
   DA OAODH
                    ; CRLF
                    ;Ende der Zeichenkette
   DB 0
   end
```

Beispiel2: Tastaturabfrage

```
cpu
            z80
      org
            300h
; Ausgabe Taste hexadezimal
; Start im OS mit TEST
      jp
           main
          "TEST ",0 ; 8 Zeichen; Ende der Zeichenkette
      db
                      ; Ende des Headers
      db
      ld c, 11 ; CSTS
main:
             5
      call
      push
             af
      call
            out a
      pop
            af
; jr main ; variante A: der Tastcode bleibt erhalten
      or
      jr
          z,main
                   ; keine Taste gedrückt
                   ; CONSI
      ld
           c,1
      call 5
                    ; sonst Taste aus Puffer holen
      ir main
; Ausgabe A hexadezimal ASCII 2 Stellen
out a: push af
      and 0F0h
      rlca
      rlca
      rlca
      rlca
            out a1
      call
            af
      pop
      and
            0Fh
            add a, 30h ; '0'
out a1:
```

```
cp 3Ah ; '9'+1
jr c, out_a2
add a, 7
; Zeichenausgabe A
out_a2: ld e, a
    ld c,2 ; CONSO
    call 5
    ret
end
```

Beispiel 3: Testprogramm für Parameterübergabe TESTPARA

Es werden Parametertyp, Parameterwert, eventuelle Fehler sowie der Inhalt des Buffers CONBU nach jedem Holen des nächsten Parameters mit GVAL angezeigt.

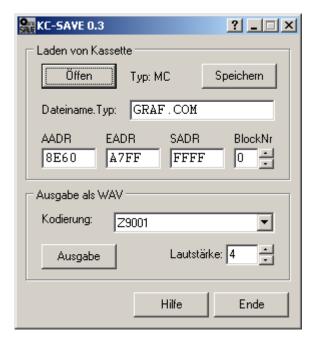
```
; Testprogramm für GVAL-Funktion (Parameterübergabe an Programme)
       cpu
             z80
CONBU:
          EQU
                80H
                        ; CCP ZEICHENKETTENPUFFER
INTLN:
                0100h
                           ; interner Zeichenkettenpuffer
          equ
          EQU
EQU
OCRLF:
                0F2FEH
OUTA:
         EQU
                0F305H
OSPAC:
          EQU
                0F310H
GVAL
                0F1EAh
          equ
      org 300h
              ______
; Kommando-Rahmen
       jp
            para
            "TESTPARA", 0
       db
      db
 TESTPARA
      ex af, af'
para:
      jr c, ende
                            ; keine weiteren Parameter
next_param:
       call anz_conbu
                       ; Anzeige CONBU
```

```
; nächsten Parameter holen
       call qval
; GVAL
; Funktion: Löschen internen Puffer (INTLN).
         Übernahme Parameter aus CONBU nach INTLN
         Test auf Parameterart
         Konvertieren Parameter, wenn dieser ein Wert ist
 Return
     Parameter: Z 1 Parameter war Dezimalzahl
               O Parameter war keine Zahl
            CY 0 kein Fehler
                1 Fehler im Parameter
            A Konvertierte Dezimalzahl, wenn Z = 1 und CY = 0
            C den Parameter begrenzendes Trennzeichen
            B Länge des Parameters
            HL Adresse des nächsten Zeichens in CONBU
            CY' O weitere Parameter in CONBU (ist in Doku falsch!)
                1 keine weiteren Parameter (ist in Doku falsch!)
            A' den Parameter begrenzendes Trennzeichen
            INTLN Länge des Parameters
            INTLN+1. . . übernommener Parameter
            CONBU übernommener Parameter und Trennzeichen gelöscht mit
              Leerzeichen
       jr z,zahl parameter
       call
               prnst
       db
             "Text ", 0
       jr
             para1
zahl parameter:
       push
               af
       call
               prnst
             "Zahl ", 0
       db
       pop
              af
       jr
             nc, paral ; Fehler in Zahl?
       call
              prnst
             "mit Fehler! ",0
       db
       call
paral:
                  prnst
             "Laenge=", 0
       db
                          ; Länges des Parameters
       ld
             a, (INTLN)
             a,'0'
       add
               OUTA
       call
               OSPAC
       call
       ld
             a, (INTLN)
       or
       jr
                               ; bei Länge 0 nicht anzeigen
             z,para2
```

```
ld de, INTLN+1
      ld c,9
      call 5
                        ; Anzeige Text
      call OCRLF
para2
      ex af, af'
                           71
          c, ende
                           ; wenn kein Parameter folgt
      jr
      jr next_param
      call anz_conbu
ende:
      call
            prnst
          "-- kein weiterer Parameter --"
      db
      db
           0dh,0ah,0
      or
      ret
;Ausgabe String, bis 0
prnst: EX (SP),HL
                              ;Adresse hinter CALL
      LD A, (HL)
PRS1:
      INC
           HL
                     ;Ende (A=0=?
      or
           Α
      JR
           Z, PRS2
                     ;ja
      CALL OUTA
      JR PRS1 ;nein
EX (SP),HL ;neue Returnadresse
          PRS1
PRS2:
      RET
; Anzeige CONBU
anz_conbu: call prnst
      db "CONBU >", 0
      ld de, CONBU+2
      ld
           c,9
anz
      call 5
      ld a, '<'
      call
            outa
      call
             0CRLF
      ret
```

Programmerstellung am PC

Bei großen Programmen ist es leichter diese am PC zu schreiben und zu assemblieren. Ich nutze dafür den arnold-assembler. Kleine in Perl geschriebene Hilfstools unterstützen den Prozess und erzeugen z.B. gleich tap-Dateien, die im Emulator geladen werden können oder mit KCSAVE kcsave.rar als Audiosignal am realen KC geladen werden können.



steht aber noch in INTLN

From:

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

Permanent link:

https://hc-ddr.hucki.net/wiki/doku.php/z9001/programmieren?rev=1733142678

Last update: 2024/12/02 12:31

