

Listing MP-BASIC

Der komplette TINY-MP-BASIC-Interpreter des ROMs des U883

```

1/      0 :                               ; reass: Volker Pohlrs
02.2007/07.2017
2/      0 :
3/      0 :
4/      0 :                               ;AS-Funktionen
5/      0 :                               hi           function
x, (x>>8)&255
6/      0 :                               lo           function x, x&255
7/      0 :
8/      0 :
9/      0 :                               cpu      z8601
10/     0 :                               include   stddefz8.inc
(1) 1/      0 :                               save
(1) 55/     0 : ALL                          restore
; wieder erlauben
(1) 56/     0 :
(1) 57/     0 :
11/     0 :                               page      0
12/     0 :
13/     0 :                               org        0
14/     0 :                               assume   RP:0C0h      ;
keine Optimierung durch AS!
15/     0 :
16/     0 : =4H                               reg_04     equ     4      ; Hi TRAP
17/     0 : =5H                               reg_05     equ     5      ; Lo TRAP
18/     0 : =6H                               reg_06     equ     6      ; Hi Adr.
Basic-Programm
19/     0 : =7H                               reg_07     equ     7      ; Lo Adr.
Basic-Programm
20/     0 : =8H                               reg_08     equ     8      ; Hi Adr.
ext. Prozedurliste
21/     0 : =9H                               reg_09     equ     9      ; Lo Adr.
ext. Prozedurliste
22/     0 : =0AH                               reg_0A     equ     0Ah
23/     0 : =0BH                               reg_0B     equ     0Bh
24/     0 : =0CH                               reg_0C     equ     0Ch
25/     0 : =0EH                               reg_0E     equ     0Eh      ;
Bit0..Bit3: Verschachtelungstiefe f. GOSUB
26/     0 :                               ; Bit5 = 1: return
without gosub
27/     0 :                               ; Bit6
28/     0 :                               ; Bit7
29/     0 :
30/     0 : =0FH                               reg_0F     equ     0Fh      ;
aktueller State

```

```

31/      0 :                               ; Bit 0 01: 1=ELSE
32/      0 :                               ; Bit 1 02: END
33/      0 :                               ; Bit 2 04:
34/      0 :                               ; Bit 3 08: STOP
35/      0 :                               ; ...
36/      0 :                               ; Bit 4 10: <
37/      0 :                               ; Bit 5 20: >
38/      0 :                               ; Bit 6 40: =
39/      0 :                               ; Bit 7 80:
40/      0 : =7FH          reg_7F          equ    7Fh
41/      0 :
42/      0 :                               ;reg_20h..reg53h Variablen A..Z
(Doppelregister, 16 Bit)
43/      0 :                               ;ab 54h frei nutzbar, z.B. f. Stack
44/      0 :
45/      0 :                               ; ext. Routinen
46/      0 :                               ;saddrh equ    0E000h    ; startadr
RAM s. init10
47/      0 :                               ;saddr equ    812h    ; startadr
ROM s. init10
48/      0 : =815H          getch equ    815h
49/      0 : =818H          putch equ    818h
50/      0 :
51/      0 :
52/      0 :                               ; Register
53/      0 : =0FEH          gpr          equ    0FEh    ; General
purpose register bzw. Stackpointer, Highteil
54/      0 :
55/      0 :                               ;-----
-----
56/      0 :                               ;
57/      0 :                               ;-----
-----
58/      0 :
59/      0 : 08 00          irq0:         dw    800h
60/      2 : 08 03          irq1:         dw    803h
61/      4 : 08 06          irq2:         dw    806h
62/      6 : 08 09          irq3:         dw    809h
63/      8 : 08 0C          irq4:         dw    80Ch
64/     A : 08 0F          irq5:         dw    80Fh
65/      C :
66/      C :                               ;-----
-----
67/      C :                               ; Bootstrap MODULE
68/      C :                               ; s. Kieser/Bankel Seite    228-229
69/      C :                               ;-----
-----
70/      C :
71/      C :                               ; public init
72/      C : 31 00          init:         srp    #0    ;
Registerpointer auf 00-Gruppe

```

```

    73/      E : 3C 0F          ld    R3, #0Fh    ; Test,
ob P32 und P35 gebrückt sind
    74/      10 : FF          nop
    75/      11 : 76 E3 04     tm    R3, #4
    76/      14 : 3C FF       ld    R3, #0FFh
    77/      16 : EB 05       jr    NZ, init10
    78/      18 : 76 E3 04     tm    R3, #4
    79/      1B : EB 20       jr    NZ, test    ; wenn
gebrückt, dann weiter im Test-Programm
    80/      1D :              ;
    81/      1D : E6 F8 B6     init10:  ld    p01m, #10110110b ;
Festlegung für erweitertes Zeitverhalten
    82/      20 : E6 F7 08     ld    p3m, #00001000b ;
Programmierung Port 3
    83/      23 : 4C 08       ld    R4, #8      ; Test,
ob sich auf Adresse 0812h RAM befindet
    84/      25 : 5C 12       ld    R5, #12h
    85/      27 : C2 64       ldc   R6, @RR4
    86/      29 : 60 E6       com   R6
    87/      2B : D2 64       ldc   @RR4, R6
    88/      2D : C2 74       ldc   R7, @RR4
    89/      2F : 60 E6       com   R6
    90/      31 : D2 64       ldc   @RR4, R6
    91/      33 : B2 67       xor   R6, R7
    92/      35 : 31 F0       srp   #0F0h
    93/      37 : ED E0 00     jp    NZ, 0E000h  ; wenn
RAM lokalisiert ist, Sprung nach 0E000h
    94/      3A : 8D 08 12     jp    812h        ; sonst
nach Adresse 0812h springen
    95/      3D :
    96/      3D :
    97/      3D :              ;-----
-----
    98/      3D :              ; Testhilfe zur Ausgabe von
alternierenden
    99/      3D :              ; Impulsen am Port P1 und P0 mit
100/      3D :              ; Triggerimpuls an Leitung P35
101/      3D :              ;-----
-----
    102/     3D :
    103/     3D : E6 F8 04     test:  ld    p01m, #4    ; Port
P1 und P0 auf Ausgabe
    104/     40 : E6 F1 C0     ld    tmr, #0C0h    ;
internen Takt ausgeben
    105/     43 : 0C FF       ld    R0, #0FFh
    106/     45 : 1C FF       ld    R1, #0FFh
    107/     47 : CF         rcf
    108/     48 : 56 E3 DF     test10: and   R3, #0DFh ;
Triggerimpuls an Leitung P35
    109/     4B : 46 E3 20     or    R3, #20h
    110/     4E : 10 E1     test20: rlc   R1

```

```

111/      50 : 10 E0                rlc    R0
112/      52 : 76 E3 04            tm     R3, #4      ;
Abfrage für tristate
113/      55 : 6B 04                jr     Z, tristate
114/      57 : 7B F5                jr     C, test20
115/      59 : 8B ED                jr     test10
116/      5B : E6 F8 7F            tristate: ld    p01m, #7Fh  ;
Ports    0-1 mode
117/      5E : 8B FE            halt:   jr     halt      ;
Warteschleife
118/      60 :
119/      60 :                    ;-----
-----
120/      60 :                    ; dma MODULE
121/      60 :                    ; s. Kieser/Bankel Seite 235
122/      60 :
123/      60 :                    ; Servicerroutine für BUSREQ,
124/      60 :                    ; mit BUSREQ an P32 und
125/      60 :                    ; BUSACK ans P35, beide L-aktiv
126/      60 :                    ;-----
-----
127/      60 :
128/      60 :                    busreq:   ; Festlegung
der Kodierung für Tristate für Port 0 und P1
129/      60 : 46 7F 18            or     reg_7F, #18h  ; P01M
muß in Register %7F verfügbar sein, da P01M nicht lesbar
130/      63 : E4 7F F8            ld     p01m, reg_7F  ;
Ports    0-1 mode
131/      66 : 56 03 DF            and    p3, #0DFh    ;
Ausgabe von BUSACK
132/      69 : 76 03 04            busreq10: tm    p3, #4      ;
Port 3
133/      6C : 6B FB                jr     Z, busreq10   ; Ende
abwarten
134/      6E : 56 7F F7            and    reg_7F, #0F7h  ;
Festlegung der Kodierung für AD0..AD7 an Port 0 und Port 1
135/      71 : 46 03 20            or     p3, #20h     ; BUSACK
zurücknehmen
136/      74 : E4 7F F8            ld     p01m, reg_7F  ;
wieder auf Bus gehen
137/      77 : FF                    nop                      ; Ausführung
(wegen Pipeline) abwarten
138/      78 : BF                    ired
139/      79 :
140/      79 :                    ; Beginn TINY-MP-BASIC
141/      79 :                    ; Variablen A..Z liegen in Register
20h..53h
142/      79 :                    ; Registernutzung RP = 10h
143/      79 :                    ; RR0 Pointer aktuelles Zeichen in
BASIC-Programm
144/      79 :                    ; RR2 Y, Rückgabewert

```

```

145/      79 :                ; RR4  X, Eingabewert
146/      79 :                ; R6   aktuelles Zeichen
147/      79 :
148/      79 :
149/      79 :                ; Arithmetik: Parameter1 X R4+R5,
Rückgabewert Y R2+R3
150/      79 :
151/      79 :                ;-----
-----
152/      79 :                ; Y := X
153/      79 :                ;-----
-----
154/      79 :
155/      79 : 28 E4          p_let:      ld   R2, R4          ; Y
:= X
156/      7B : 38 E5                ld   R3, R5
157/      7D : AF                ret
158/      7E :
159/      7E :                ;-----
-----
160/      7E :                ; Y := Y + X
161/      7E :                ;-----
-----
162/      7E :
163/      7E : 02 35          p_plus:      add   R3, R5          ; Y
:= Y + X
164/      80 : 12 24                adc   R2, R4
165/      82 : 4B 16                jr    0V, p_abs3
166/      84 : AF                ret
167/      85 :
168/      85 :                ;-----
-----
169/      85 :                ; Y := Y - X
170/      85 :                ;-----
-----
171/      85 :
172/      85 : 22 35          p_minus:     sub   R3, R5          ; Y
:= Y - X
173/      87 : 32 24                sbc   R2, R4
174/      89 : 4B 0F                jr    0V, p_abs3
175/      8B : AF                ret
176/      8C :
177/      8C :                ;-----
-----
178/      8C :                ; interne Prozedur ABS
179/      8C :                ; ABS[parameter] absoluter Betrag
180/      8C :                ;-----
-----
181/      8C :                ; Y := ABS (X)
182/      8C : 76 E4 80          p_abs:      tm   R4, #80h          ; X
Test Vorzeichen Bit      7

```

```

183/      8F : 6B E8                jr    Z, p_let    ; Y := X
184/      91 :                    ; sonst negieren
185/      91 : B0 E2                p_abs1:  clr    R2    ; Y :=
0
186/      93 : B0 E3                clr    R3
187/      95 : 8B EE                jr    p_minus    ; Y :=
Y - X
188/      97 : 46 0F 80            p_abs2:  or     reg_0F, #80h    ;
Bit7
189/      9A : 46 0E 80            p_abs3:  or     reg_0E, #80h    ;
Bit7
190/      9D : AF                    ret
191/      9E :
192/      9E :
193/      9E :                    ;-----
-----
194/      9E :                    ; UP zu DIV/MULT
195/      9E :                    ;-----
-----
196/      9E :
197/      9E :                    ; ? Vorzeichen behandeln
198/      9E : 88 E2                sign:    ld     R8, R2
199/      A0 : B2 84                xor     R8, R4
200/      A2 : 9C 02                ld     R9, #2
201/      A4 :                    ;
202/      A4 : 68 E2                sign1:   ld     R6, R2
203/      A6 : 78 E3                ld     R7, R3
204/      A8 : D6 00 8C            call   p_abs    ;
ABS[parameter] absoluter Betrag
205/      AB : 48 E6                ld     R4, R6
206/      AD : 58 E7                ld     R5, R7
207/      AF : 9A F3                djnz   R9, sign1
208/      B1 : 68 E2                ld     R6, R2
209/      B3 : 78 E3                ld     R7, R3
210/      B5 : B0 E2                clr    R2
211/      B7 : B0 E3                clr    R3
212/      B9 : AF                    ret
213/      BA :
214/      BA :                    ;-----
-----
215/      BA :                    ; Y := Y * X
216/      BA :                    ;-----
-----
217/      BA :
218/      BA : D6 00 9E            p_mult:  call   sign
219/      BD : BC 0F                ld     R11, #0Fh
220/      BF : D0 E6                p_mult1: sra   R6
221/      C1 : C0 E7                rrc    R7
222/      C3 : FB 06                jr     NC, p_mult2
223/      C5 : 02 35                add    R3, R5
224/      C7 : 12 24                adc    R2, R4

```

```

225/      C9 : 4B CC                jr      0V, p_abs2
226/      CB : 10 E5                p_mult2: rlc      R5
227/      CD : 10 E4                rlc      R4
228/      CF : CB 04                jr      NOV, p_mult3
229/      D1 : 42 67                or       R6, R7
230/      D3 : EB C2                jr      NZ, p_abs2
231/      D5 : BA E8                p_mult3: djnz     R11, p_mult1
232/      D7 : 48 E2                p_mult4: ld       R4, R2
233/      D9 : 58 E3                ld       R5, R3
234/      DB : 10 E8                rlc      R8
235/      DD : 7B B2                jr      C, p_abs1
236/      DF : AF                ret
237/      E0 :
238/      E0 :                      ;-----
-----
239/      E0 :                      ; Y := Y / X
240/      E0 :                      ;-----
-----
241/      E0 :
242/      E0 : D6 00 9E                p_div:   call     sign
243/      E3 : 9C 10                ld       R9, #10h
244/      E5 : CF                    rcf
245/      E6 : B0 EA                clr      R10          ; RR10 =
Divisionstrest = 0
246/      E8 : B0 EB                clr      R11
247/      EA : 10 E7                p_div1:  rlc      R7
248/      EC : 10 E6                rlc      R6
249/      EE : 10 EB                rlc      R11
250/      F0 : 10 EA                rlc      R10
251/      F2 : 7B 0A                jr      C, p_div2
252/      F4 : A2 4A                cp       R4, R10
253/      F6 : BB 0B                jr      UGT, p_div3
254/      F8 : 7B 04                jr      C, p_div2
255/      FA : A2 5B                cp       R5, R11
256/      FC : BB 05                jr      UGT, p_div3
257/      FE : 22 B5                p_div2:  sub      R11, R5
258/      100 : 32 A4                sbc     R10, R4
259/      102 : DF                    scf
260/      103 : 9A E5                p_div3:  djnz     R9, p_div1
261/      105 : 98 E4                ld       R9, R4
262/      107 : 42 95                or       R9, R5
263/      109 : 6B 0A                jr      Z, p_div5
264/      10B : 10 E7                rlc      R7
265/      10D : 10 E6                rlc      R6
266/      10F :                      ;
267/      10F : 28 E6                p_div4:  ld       R2, R6
268/      111 : 38 E7                ld       R3, R7
269/      113 : 8B C2                jr      p_mult4
270/      115 :                      ;
271/      115 : D6 01 0F                p_div5:  call     p_div4
272/      118 : 46 0E 40                or      reg_0E, #40h ; Bit6

```

```

    273/      11B : 46 0F 80          p_div6:      or      reg_0F, #80h      ;
Bit7
    274/      11E : AF                      ret
    275/      11F :
    276/      11F :                      ;-----
-----
    277/      11F :                      ; $-Operator
    278/      11F :                      ; Y := Y MOD X
    279/      11F :                      ;-----
-----
    280/      11F :
    281/      11F : D6 00 E0          p_mod:      call     p_div
    282/      122 : 88 E2              ld      R8, R2
    283/      124 : 28 EA              ld      R2, R10      ;
Divisonsrest
    284/      126 : 38 EB              ld      R3, R11
    285/      128 : 8B AD              jr      p_mult4
    286/      12A :
    287/      12A :                      ;-----
-----
    288/      12A :                      ; $-Operator
    289/      12A :                      ; Y := Y OR X
    290/      12A :                      ;-----
-----
    291/      12A :
    292/      12A : 42 24          p_or:      or      R2, R4
    293/      12C : 42 35              or      R3, R5
    294/      12E : AF                      ret
    295/      12F :
    296/      12F :                      ;-----
-----
    297/      12F :                      ; $-Operator
    298/      12F :                      ; Y := Y AND X
    299/      12F :                      ;-----
-----
    300/      12F :
    301/      12F : 52 24          p_and:     and     R2, R4
    302/      131 : 52 35              and     R3, R5
    303/      133 : AF                      ret
    304/      134 :
    305/      134 :                      ;-----
-----
    306/      134 :                      ; $-Operator
    307/      134 :                      ; Y := Y XOR X
    308/      134 :                      ;-----
-----
    309/      134 :
    310/      134 : B2 24          p_xor:     xor     R2, R4
    311/      136 : B2 35              xor     R3, R5
    312/      138 : AF                      ret
    313/      139 :

```



```

314/      139 : ;-----
-----
315/      139 : ; interne Prozedur NOT
316/      139 : ;-----
-----
317/      139 : ; Y := NOT X
318/      139 : ; NOT[parameter] bitweise logische
Negation
319/      139 :
320/      139 : D6 00 79      p_not:      call      p_let      ; Y
:= X
321/      13C : 60 E2              com      R2              ; Y := NOT
X
322/      13E : 60 E3              com      R3
323/      140 : AF              ret
324/      141 :
325/      141 : ;-----
-----
326/      141 : ; Vergleich RR2 und RR4 auf <,>:
out: reg_0F
327/      141 : ;-----
-----
328/      141 :
329/      141 : 56 0F 8F      relcmp:      and      reg_0F, #8Fh      ;
Bit 4..6 = 0
330/      144 : A2 24              cp      R2, R4
331/      146 : 6B 0A              jr      Z, relcmp3
332/      148 : 7C 20              ld      R7, #20h
333/      14A : AB 02              jr      GT, relcmp2      ; >
Bitmaske 20h
334/      14C : 7C 10      relcmp1:      ld      R7, #10h      ; <
Bitmaske 10h
335/      14E : 44 E7 0F      relcmp2:      or      reg_0F, R7
336/      151 : AF              ret
337/      152 : ;
338/      152 : 7C 40      relcmp3:      ld      R7, #40h
339/      154 : A2 35              cp      R3, R5
340/      156 : 6B F6              jr      Z, relcmp2      ; =
Bitmaske 40h
341/      158 : 7C 20              ld      R7, #20h
342/      15A : BB F2              jr      UGT, relcmp2
343/      15C : 8B EE              jr      relcmp1
344/      15E :
345/      15E : ;-----
-----
346/      15E : ; UP zu c_PRINTHEX
347/      15E : ; Konvertierung nach hexadezimal
348/      15E : ;-----
-----
349/      15E :
350/      15E : 88 E3      tohex:      ld      R8, R3      ;

```

```

Byte->Nibble
 351/      160 : 78 E3          ld      R7, R3          ; R2/R3
in R5..R8 kopieren
 352/      162 : 68 E2          ld      R6, R2          ;
 353/      164 : 58 E2          ld      R5, R2          ;
 354/      166 : F0 E5          swap    R5              ; Hi
Nibble in untere Tetrade
 355/      168 : F0 E7          swap    R7              ; Hi
Nibble in untere Tetrade
 356/      16A : 4C 25          ld      R4, #'%'
 357/      16C :                ;
 358/      16C : AC 04          ld      R10, #4         ; 4
Stellen
 359/      16E : BC 15          tohex1: ld      R11, #15h ;
Buffer
 360/      170 : 57 EB 0F          tohex2: and     @R11, #0Fh
 361/      173 : 07 EB 30          add     @R11, #30h     ; '0'
 362/      176 : A7 EB 3A          cp     @R11, #3Ah     ; '9'+1
 363/      179 : 7B 03          jr     C, tohex3      ; wenn
größer '9', dann + 7
 364/      17B : 07 EB 07          add     @R11, #7      ; ergibt
'A'..'F'
 365/      17E : BE          tohex3: inc     R11     ;
nächste Ausgabepos
 366/      17F : AA EF          djnz   R10, tohex2   ;
nächste Stelle
 367/      181 : AF          ret
 368/      182 :
 369/      182 :                ;-----
-----
 370/      182 :                ; UP zu c_PRINT
 371/      182 :                ; Konvertierung nach dezimal
 372/      182 :                ;-----
-----
 373/      182 :
 374/      182 : 48 E2          todez:  ld      R4, R2
 375/      184 : 58 E3          ld      R5, R3
 376/      186 : D6 00 8C          call   p_abs         ;
absoluter Betrag
 377/      189 : 10 E4          rlc     R4
 378/      18B : 4C 20          ld      R4, #' '     ; bei pos.
Zahl Leerzeichen
 379/      18D : FB 02          jr     NC, todez1
 380/      18F : 4C 2D          ld      R4, #'-'     ; bei neg.
Zahl '-'
 381/      191 : BC 06          todez1: ld      R11, #6         ; 6
Stellen
 382/      193 : AC 15          ld      R10, #15h    ; Buffer
 383/      195 : B1 EA          todez2: clr     @R10     ;
leeren
 384/      197 : AE          inc     R10

```

```

385/      198 : BA FB                djnz   R11, todez2
386/      19A :                    ;hex-> bcd
387/      19A : BC 0F                ld     R11, #0Fh
388/      19C : D0 E2                todez3: sra   R2
389/      19E : C0 E3                rrc   R3
390/      1A0 : FB 0C                jr    NC, todez4
391/      1A2 : 02 7A                add   R7, R10
392/      1A4 : 40 E7                da    R7
393/      1A6 : 12 69                adc   R6, R9
394/      1A8 : 40 E6                da    R6
395/      1AA : 12 58                adc   R5, R8
396/      1AC : 40 E5                da    R5
397/      1AE : 02 AA                todez4: add   R10, R10
398/      1B0 : 40 EA                da    R10
399/      1B2 : 12 99                adc   R9, R9
400/      1B4 : 40 E9                da    R9
401/      1B6 : 12 88                adc   R8, R8
402/      1B8 : 40 E8                da    R8
403/      1BA : BA E0                djnz   R11, todez3
404/      1BC : 88 E7                ld    R8, R7
405/      1BE : 98 E7                ld    R9, R7
406/      1C0 : 78 E6                ld    R7, R6
407/      1C2 : F0 E6                swap  R6
408/      1C4 : F0 E8                swap  R8
409/      1C6 :                    ;bcd->ascii
410/      1C6 : AC 05                ld    R10, #5          ; 5
Stellen
411/      1C8 : 8B A4                jr    tohex1          ; nach
ASCII wandeln
412/      1CA :
413/      1CA :                    ;-----
-----
414/      1CA :                    ; Ziffer in Result einschieben (i.e.
* 16 + stelle)
415/      1CA :                    ; UP zu number
416/      1CA :                    ;-----
-----
417/      1CA :
418/      1CA : F1 ED                rotate: swap  @R13      ;
Ziffer in oberes Nibble
419/      1CC : EC 04                ld    R14, #4         ; alles
4 Stellen nach links
420/      1CE : 11 ED                rotatel: rlc  @R13     ;
d.h. * 16
421/      1D0 : 10 E5                rlc   R5
422/      1D2 : 10 E4                rlc   R4
423/      1D4 : 10 E3                rlc   R3
424/      1D6 : EA F6                djnz   R14, rotatel
425/      1D8 : AF                ret
426/      1D9 :
427/      1D9 :                    ;-----

```

```

-----
    428/    1D9 :                ; Konvertierung ASCII->Zahl (hex, o.
dez signed)
    429/    1D9 :                ; in: String in Konvertierungspuffer
ab 16h (R6 ff) ..
    430/    1D9 :                ; ret: RR4 = Wert
    431/    1D9 :                ;-----
-----
    432/    1D9 :
    433/    1D9 : DC 16          number:    ld    R13, #16h    ;
Konvertierungspuffer
    434/    1DB : B0 E4          clr    R4          ; Startwert
0
    435/    1DD : B0 E5          clr    R5
    436/    1DF : A6 E6 25       cp    R6, #'%'    ; Präfix
Hexzahl
    437/    1E2 : EB 27          jr    NZ, number5
    438/    1E4 :
    439/    1E4 :                ; Hex-Zahl
    440/    1E4 : CC 05          ld    R12, #5     ; max.
5 Stellen
    441/    1E6 : DE            number1:  inc    R13
    442/    1E7 : 27 ED 30       sub    @R13, #'0'
    443/    1EA : 7B 19          jr    C, number4
    444/    1EC : A7 ED 0A       cp    @R13, #0Ah  ; 9+1
    445/    1EF : 7B 0D          jr    C, number2
    446/    1F1 : 27 ED 11       sub    @R13, #11h ; A..F
-> 0..5
    447/    1F4 : 7B 0F          jr    C, number4
    448/    1F6 : 07 ED 0A       add    @R13, #0Ah ; +10
    449/    1F9 : A7 ED 10       cp    @R13, #10h
    450/    1FC : FB 07          jr    NC, number4
    451/    1FE : D6 01 CA       number2:  call   rotate    ;
* 16 + Stelle
    452/    201 : CA E3          djnz   R12, number1
    453/    203 : CF            number3:  rcf
    454/    204 : AF            ret
    455/    205 :                ;
    456/    205 : A6 EC 05       number4:  cp    R12, #5
    457/    208 : 6B F9          jr    Z, number3
    458/    20A : AF            ret
    459/    20B :
    460/    20B :                ; Dezimalzahl
    461/    20B : CC 06         number5:  ld    R12, #6    ;
max. 6 Stellen
    462/    20D : B0 E2          clr    R2
    463/    20F : B0 E3          clr    R3
    464/    211 : A6 E6 2D       cp    R6, #'-'    ;
Vorzeichen
    465/    214 : EB 02          jr    NZ, number7
    466/    216 : 2E            inc    R2

```

```

467/      217 : DE          number6:   inc   R13
468/      218 : 27 ED 30   number7:   sub   @R13, #'0'
469/      21B : 7B 0C          jr    C, number9 ; wenn
keine Ziffer mehr
470/      21D : A7 ED 0A          cp   @R13, #0Ah ; 9+1
471/      220 : FB 07          jr   NC, number9 ; wenn
keine Ziffer mehr
472/      222 : D6 01 CA          call  rotate ;
sonst zu BCD konvertieren
473/      225 : CA F0          djnz  R12, number6
474/      227 : CF          number8:   rcf
475/      228 : AF          ret
476/      229 :
477/      229 :          ; BCD R3R4R5 -> hex RR4
478/      229 : A6 EC 06   number9:   cp   R12, #6
479/      22C : 6B F9          jr   Z, number8
480/      22E : A6 E3 04          cp   R3, #4
481/      231 : FB F4          jr   NC, number8
482/      233 : B0 E6          clr  R6
483/      235 : B0 E7          clr  R7
484/      237 : BC 13          ld   R11, #13h
485/      239 : 8C 27          ld   R8, #27h ; 2710h =
10000
486/      23B : 9C 10          ld   R9, #10h
487/      23D : D6 02 71          call number11
488/      240 : BE          inc  R11
489/      241 : 8C 03          ld   R8, #3 ; 3E8h =
1000
490/      243 : 9C E8          ld   R9, #0E8h
491/      245 : D6 02 6F          call number10
492/      248 : 8C 00          ld   R8, #0 ; 064h =
100
493/      24A : 9C 64          ld   R9, #64h
494/      24C : D6 02 6F          call number10
495/      24F : BE          inc  R11
496/      250 : 9C 0A          ld   R9, #0Ah ; 10
497/      252 : D6 02 6F          call number10
498/      255 : 9C 01          ld   R9, #1 ; 1
499/      257 : D6 02 6F          call number10
500/      25A : 48 E6          ld   R4, R6
501/      25C : 58 E7          ld   R5, R7
502/      25E : 10 E6          rlc  R6
503/      260 : 7B C5          jr   C, number8
504/      262 : C0 E2          rrc  R2
505/      264 : FB 18          jr   NC, number13
506/      266 : D6 00 91          call p_abs1
507/      269 : 48 E2          ld   R4, R2
508/      26B : 58 E3          ld   R5, R3
509/      26D : 8B 0F          jr   number13
510/      26F :
511/      26F :          ;

```

```

512/      26F : F1 EB          number10:  swap   @R11
513/      271 : E3 AB          number11:  ld     R10, @R11 ;
Stelle n
514/      273 : 56 EA 0F          and     R10, #0Fh
515/      276 : 6B 06          jr      Z, number13
516/      278 : 02 79          number12:  add     R7, R9 ;
RR8 = Stelligkeit
517/      27A : 12 68          adc     R6, R8
518/      27C : AA FA          djnz   R10, number12 ; n
mal Addieren
519/      27E : DF          number13:  scf
520/      27F : AF          ret
521/      280 :
522/      280 : ;-----
-----
523/      280 : ; bei Prozeduraufruf PROC
[Y1,Y2,Y3,...,Ym] = prozedurname [X1,X2,X3,...,Xn] gilt:
524/      280 : ; SP returnadr.
525/      280 : ; SP+2 .. SP+2n-2 n-1
Parameter, wenn n>1
526/      280 : ; letzter Parameter
aus Liste ist in RR4
527/      280 : ; SP+2n (intern f.
interpreter)
528/      280 : ; SP+2n+2 .. SP+2m+2n Platz
für m-1 Ergebnisvariable, wenn m>1
529/      280 : ; letzter Parameter
aus Liste ist in RR2 zu übergeben
530/      280 : ;-----
-----
531/      280 :
532/      280 : ;-----
-----
533/      280 : ; Ret.Adr. und ersten Parameter vom
Stack nach R6/R7 holen
534/      280 : ;-----
-----
535/      280 :
536/      280 : 50 E8          para1:    pop     R8 ; RR8
Rückkehradresse dieser Routine
537/      282 : 50 E9          pop     R9
538/      284 : 50 E2          pop     R2 ; RR2
Rückkehradresse zum Interpreter
539/      286 : 50 E3          pop     R3
540/      288 : 50 E6          pop     R6 ; RR6
Adresse von X1
541/      28A : 50 E7          pop     R7
542/      28C : 30 E8          jp     @RR8 ; RET
543/      28E :
544/      28E : ;-----
-----

```

```

545/      28E :                ; interne Prozedur
546/      28E :                ; SETRR [register,wert]
Doppelregister setzen
547/      28E :                ;-----
-----
548/      28E :
549/      28E : D6 02 80      p_setrr:   call   para1   ;
Parameter register nach R6/R7
550/      291 : F3 74                ld    @R7, R4    ;
register (hi) := wert
551/      293 : 7E                inc    R7
552/      294 : 8B 03                jr    p_setr1
553/      296 :
554/      296 :                ;-----
-----
555/      296 :                ; interne Prozedur
556/      296 :                ; SETR [register,wert] Register
setzen
557/      296 :                ;-----
-----
558/      296 :
559/      296 : D6 02 80      p_setr:   call   para1   ;
Parameter register nach R6/R7
560/      299 : F3 75                p_setr1:  ld    @R7, R5    ;
register (lo) := wert
561/      29B : 30 E2                jp    @RR2      ; zurück
zum Interpreter
562/      29D :
563/      29D :                ;-----
-----
564/      29D :                ; interne Prozedur
565/      29D :                ; SETEW [adresse,wert] externes
Wort setzen
566/      29D :                ;-----
-----
567/      29D :
568/      29D : D6 02 80      p_setew:  call   para1   ;
Parameter adresse nach R6/R7
569/      2A0 : 92 46                lde    @RR6, R4
570/      2A2 : A0 E6                incw   RR6
571/      2A4 : 8B 03                jr    p_seteb1
572/      2A6 :
573/      2A6 :                ;-----
-----
574/      2A6 :                ; interne Prozedur
575/      2A6 :                ; SETEB [adresse,wert] externes
Byte setzen
576/      2A6 :                ;-----
-----
577/      2A6 :
578/      2A6 : D6 02 80      p_seteb:  call   para1   ;

```

```

Parameter adresse nach R6/R7
579/      2A9 : 92 56          p_seteb1:  lde    @RR6, R5
580/      2AB : 30 E2          ;          jp     @RR2
581/      2AD :
582/      2AD : ;-----
-----
583/      2AD : ; interne Prozedur
584/      2AD : ; GETRR [register] liefert Inhalt
des Doppelregisters
585/      2AD : ;-----
-----
586/      2AD :
587/      2AD : E3 25          p_getrr:   ld     R2, @R5
588/      2AF : 5E              ;          inc     R5
589/      2B0 : 0D              ;          db     0Dh      ; JP FALSE
mit nächstem Befehl
590/      2B1 :
591/      2B1 : ;-----
-----
592/      2B1 : ; interne Prozedur
593/      2B1 : ; GETR [register] liefert Inhalt des
Registers
594/      2B1 : ;-----
-----
595/      2B1 :
596/      2B1 : B0 E2          p_getr:    clr     R2      ; Hi =
0
597/      2B3 : E3 35          ;          ld     R3, @R5
598/      2B5 : AF              ;          ret
599/      2B6 :
600/      2B6 : ;-----
-----
601/      2B6 : ; interne Prozedur
602/      2B6 : ; GETEW [register] holt   Wortwert
aus externem Speicher
603/      2B6 : ;-----
-----
604/      2B6 :
605/      2B6 : 82 24          p_getew:   lde     R2, @RR4
606/      2B8 : A0 E4          ;          incw   RR4
607/      2BA : 0D              ;          db     0Dh      ; JP FALSE
mit nächstem Befehl
608/      2BB :
609/      2BB : ;-----
-----
610/      2BB : ; interne Prozedur
611/      2BB : ; GETEB [register] holt   Bytewert
aus externem Speicher
612/      2BB : ;-----
-----
613/      2BB :

```



```

614/      2BB : B0 E2          p_geteb:  clr   R2
615/      2BD : 82 34          lde   R3, @RR4
616/      2BF : AF           ret
617/      2C0 :
618/      2C0 : ;-----
-----
619/      2C0 : ; angezeigtes Zeichen löschen
620/      2C0 : ;-----
-----
621/      2C0 : ; UP zu p_input6
622/      2C0 :
623/      2C0 : D6 02 C8      delc:   call  delc1   ;
ein Zeichen zurück
624/      2C3 : 5C 20          ld    R5, #' '   ; mit
Leerzeichen überschreiben
625/      2C5 : D6 08 18      call  putch      ;
PUT_CHAR
626/      2C8 : 5C 08          delc1:  ld    R5, #8   ; BS
627/      2CA : 8D 08 18      jp    putch      ;
PUT_CHAR
628/      2CD :
629/      2CD : ;-----
-----
630/      2CD : ; interne Prozedur
631/      2CD : ; ; RL[x] x links rotieren
632/      2CD : ;-----
-----
633/      2CD :
634/      2CD : CF           p_rl:   rcf                ; Cy = 0
635/      2CE : 10 E5          rlc   R5
636/      2D0 : 10 E4          rlc   R4
637/      2D2 : 16 E5 00      adc   R5, #0
638/      2D5 : 8D 00 79      p_rl1:  jp    p_let        ; Y
:= X
639/      2D8 :
640/      2D8 : ;-----
-----
641/      2D8 : ; interne Prozedur
642/      2D8 : ; RR[x] x rechts rotieren
643/      2D8 : ;-----
-----
644/      2D8 :
645/      2D8 : CF           p_rr:   rcf                ; Cy = 0
646/      2D9 : C0 E4          rrc   R4
647/      2DB : C0 E5          rrc   R5
648/      2DD : FB F6          jr    NC, p_rl1
649/      2DF : 46 E4 80      or    R4, #80h   ;
Vorzeichen löschen
650/      2E2 : 8B F1          jr    p_rl1
651/      2E4 :
652/      2E4 : ;-----

```

```

-----
653/      2E4 :                               ; interne Prozedur
654/      2E4 :                               ; INPUT Zahleneingabe vom Terminal
655/      2E4 :                               ;-----
-----
656/      2E4 :
657/      2E4 : 5C 3F      p_input:   ld    R5, #'?'
658/      2E6 : D6 08 18      call    putch      ;
PUT_CHAR
659/      2E9 : FC 15      p_input1:  ld    R15, #15h  ;
Konvertierungspuffer
660/      2EB : FE      p_input2:  inc    R15
661/      2EC : A6 EF 1F      cp    R15, #1Fh
662/      2EF : 6B 20      jr    Z, p_input4
663/      2F1 : D6 08 15      p_input3:  call   getch      ;
GET_CHAR
664/      2F4 : F3 F3      ld    @R15, R3
665/      2F6 : A6 E3 0D      cp    R3, #0Dh  ; CR
666/      2F9 : 6B 16      jr    Z, p_input4
667/      2FB : A6 E3 08      cp    R3, #8    ; BS
668/      2FE : EB 19      jr    NZ, p_input5
669/      300 : 5C 20      ld    R5, #' '
670/      302 : D6 08 18      call   putch      ;
PUT_CHAR
671/      305 : 00 EF      dec    R15
672/      307 : A6 EF 15      cp    R15, #15h
673/      30A : 6B DF      jr    Z, p_input2
674/      30C : D6 02 C8      call   delc1
675/      30F : 8B E0      jr    p_input3
676/      311 : D6 01 D9      p_input4:  call   number
677/      314 : FB CE      jr    NC, p_input  ; INPUT
Zahleneingabe vom Terminal
678/      316 : 8D 00 79      jp    p_let      ; Y := X
679/      319 : 3E      p_input5:  inc    R3
680/      31A : DB CF      jr    PL, p_input2
681/      31C : A6 EF 16      p_input6:  cp    R15, #16h
682/      31F : 6B C8      jr    Z, p_input1
683/      321 : D6 02 C0      call   delc      ;
angezeigtes Zeichen löschen
684/      324 : 00 EF      dec    R15
685/      326 : 8B F4      jr    p_input6
686/      328 :
687/      328 :                               ;-----
-----
688/      328 :                               ; Liste der internen Prozeduren
689/      328 :                               ;-----
-----
690/      328 :
691/      328 : 03 4E 4F 54      tab_prc:  db    3, "NOT"
692/      32C : 01 39      dw    p_not      ;
NOT[parameter] bitweise logische Negation

```

```

693/      32E : 03 41 42 53          db   3,"ABS"
694/      332 : 00 8C                dw   p_abs          ;
ABS[parameter] absoluter Betrag
695/      334 : 05 53 45 54 52 52   db   5,"SETRR"
696/      33A : 02 8E                dw   p_setrr       ; SETRR
[register,wert]  Doppelregister setzen
697/      33C : 04 53 45 54 52   db   4,"SETR"
698/      341 : 02 96                dw   p_setr        ; SETR
[register,wert] Register setzen
699/      343 : 05 53 45 54 45 57   db   5,"SETEW"
700/      349 : 02 9D                dw   p_setew       ; SETEW
[adresse,wert] externes Wort setzen
701/      34B : 05 53 45 54 45 42   db   5,"SETEB"
702/      351 : 02 A6                dw   p_seteb       ; SETEB
[adresse,wert] externes Byte setzen
703/      353 : 05 47 45 54 52 52   db   5,"GETRR"
704/      359 : 02 AD                dw   p_getrr       ; GETRR
[register] liefert Inhalt des Doppelregisters
705/      35B : 04 47 45 54 52   db   4,"GETR"
706/      360 : 02 B1                dw   p_getr        ; GETR
[register] liefert Inhalt des Registers
707/      362 : 05 47 45 54 45 57   db   5,"GETEW"
708/      368 : 02 B6                dw   p_getew       ; GETEW
[register] holt Wortwert aus externem Speicher
709/      36A : 05 47 45 54 45 42   db   5,"GETEB"
710/      370 : 02 BB                dw   p_geteb       ; GETEB
[register] holt Bytewert aus externem Speicher
711/      372 : 02 52 4C            db   2,"RL"
712/      375 : 02 CD                dw   p_rl          ; RL[x]
x links rotieren
713/      377 : 02 52 52            db   2,"RR"
714/      37A : 02 D8                dw   p_rr          ; RR[x]
x rechts rotieren
715/      37C : 05 49 4E 50 55 54   db   5,"INPUT"
716/      382 : 02 E4                dw   p_input       ; INPUT
Zahleneingabe vom Terminal
717/      384 : 03 47 54 43          db   3,"GTC"
718/      388 : 08 15                dw   getch         ; Get
Char extern !
719/      38A : 03 50 54 43          db   3,"PTC"
720/      38E : 08 18                dw   putch         ; Put
Char extern !
721/      390 : FF                    db   0FFh         ;
Listenende
722/      391 :
723/      391 : ;-----
-----
724/      391 : ;nächstes Zeichen aus Programmtext
holen
725/      391 : ;in: R7 Vergleichszeichen ret: Z=1
Zeichen gefunden

```

```

726/      391 :      ;-----
-----
727/      391 : FC 16      next_char: ld    R15, #16h    ; R15
vorbelegen für buffer ?
728/      393 : C2 60      ldc    R6, @RR0    ;
nächstes Zeichen aus Basic-Programmcode
729/      395 : A0 E0      incw   RR0        ; Pointer
weiterstellen
730/      397 : A2 67      cp     R6, R7     ;
Vergleich mit R7
731/      399 : AF      ret
732/      39A :
733/      39A :      ;-----
-----
734/      39A :      ; Operator in Operator-Tabelle
suchen
735/      39A :      ; ret: R7 = op, RR8 = proc.addr.
736/      39A :      ;-----
-----
737/      39A : AC 03      oper:   ld    R10, #HI(tab_op)
738/      39C : BC C1      ld     R11, #L0(tab_op)
739/      39E : DC 03      ld     R13, #3    ; 4
arithm. Operatoren
740/      3A0 : D6 03 AF      call   oper1     ; suche
741/      3A3 : 6B 1B      jr     Z, oper3   ; Operator
nicht gefunden
742/      3A5 : A6 E6 24      cp     R6, #'$'   ; Präfix
logischer Operator
743/      3A8 : EB 16      jr     NZ, oper3  ; Präfix
stimmt nicht
744/      3AA : D6 03 91      call   next_char
745/      3AD : DC 03      ld     R13, #3    ; 4
logische Operatoren
746/      3AF :
747/      3AF :      oper1:   ; tab_op durchsuchen
748/      3AF : D6 03 B6      call   oper2
749/      3B2 : 6B 0C      jr     Z, oper3   ; wenn
Operator gefunden
750/      3B4 : DA F9      djnz   R13, oper1
751/      3B6 :
752/      3B6 : CC 17      oper2:   ld     R12, #17h   ; 3
Byte nach 17h kopieren
753/      3B8 : C3 CA      ldci   @R12, @RR10 ; R7
:= Op
754/      3BA : C3 CA      ldci   @R12, @RR10 ; RR8
:= Fkt.Adresse
755/      3BC : C3 CA      ldci   @R12, @RR10
756/      3BE : A2 67      cp     R6, R7     ;
Operator mit Suchwert vergleichen
757/      3C0 : AF      oper3:   ret
758/      3C1 :

```

```

759/      3C1 : ;-----
-----
760/      3C1 : ; Tabelle Arithmetik-/Logik-
Operatoren, s. oper
761/      3C1 : ;-----
-----
762/      3C1 :
763/      3C1 :          tab_op:
764/      3C1 : 2B          db      '+'
765/      3C2 : 00 7E          dw      p_plus      ; Y := Y
+ X
766/      3C4 : 2D          db      '-'
767/      3C5 : 00 85          dw      p_minus      ; Y :=
Y - X
768/      3C7 : 2A          db      '*'
769/      3C8 : 00 BA          dw      p_mult       ; Y := Y
* X
770/      3CA : 2F          db      '/'
771/      3CB : 00 E0          dw      p_div        ; Y := Y
/ X
772/      3CD : ; $-Operatoren (mit vorausgehendem
'$')
773/      3CD : 41          db      'A'
774/      3CE : 01 2F          dw      p_and        ; Y := Y
AND X
775/      3D0 : 4F          db      'O'
776/      3D1 : 01 2A          dw      p_or         ; Y := Y
OR X
777/      3D3 : 58          db      'X'
778/      3D4 : 01 34          dw      p_xor        ; Y := Y
XOR X
779/      3D6 : 4D          db      'M'
780/      3D7 : 01 1F          dw      p_mod        ; Y := Y
MOD X
781/      3D9 :
782/      3D9 :
783/      3D9 : ;-----
-----
784/      3D9 : ; Zeichenklassentests
785/      3D9 : ;-----
-----
786/      3D9 :
787/      3D9 : ;in: @R15 out: cy=1 bei Buchstaben
A..Z
788/      3D9 : A7 EF 41      is_letter: cp      @R15, #41h ;
'A'
789/      3DC : 7B 12          jr      C, is_notf
790/      3DE : A7 EF 5B          cp      @R15, #5Bh ; 'Z'+1
791/      3E1 : AF          ret
792/      3E2 :
793/      3E2 : ;in: @R15 out: cy=1 bei Buchstaben

```

```

A..Z oder Zahl 0..9
794/      3E2 : D6 03 D9      is_char:   call   is_letter
795/      3E5 : 7B 08                jr     C, is_digit_ret
796/      3E7 :
797/      3E7 :                ;in: @R15 out: cy=1 bei Zahl 0..9
798/      3E7 : A7 EF 30      is_digit:  cp     @R15, #30h   ;
'0'
799/      3EA : 7B 04                jr     C, is_notf
800/      3EC : A7 EF 3A                cp     @R15, #3Ah     ; '9'+1
801/      3EF : AF                is_digit_ret:  ret
802/      3F0 :
803/      3F0 :                ;
804/      3F0 : CF                is_notf:   rcf                ; Cy=0
805/      3F1 : AF                ret
806/      3F2 :
807/      3F2 :                ;in: @R15 out: cy=1 bei Zahl 0..F
808/      3F2 : D6 03 E7      is_hexdigit: call   is_digit
809/      3F5 : 7B F8                jr     C, is_digit_ret
810/      3F7 : A7 EF 41                cp     @R15, #41h     ; 'A'
811/      3FA : 7B F4                jr     C, is_notf
812/      3FC : A7 EF 47                cp     @R15, #47h     ; 'F'+1
813/      3FF : AF                ret
814/      400 :
815/      400 :                ;-----
-----
816/      400 :                ; Prozedurnamen in Proc-Tabelle
suchen
817/      400 :                ; ret: Cy=gefunden, RR8=Proc-Adresse
818/      400 :                ;-----
-----
819/      400 :                ; Aufbau Tabelle: je 1 Byte
Namenslänge
820/      400 :                ;                Procedurenamen
821/      400 :                ;                2 Byte Adresse
822/      400 :                ;                Ende mit Namenslänge = FFh
823/      400 :                ;-----
-----
824/      400 :
825/      400 :                ; zu func1
826/      400 :      find_name: ; Suche in interner
Tabelle
827/      400 : 2C 03                ld     R2, #HI(tab_prc)
828/      402 : 3C 28                ld     R3, #L0(tab_prc)
829/      404 : D6 04 12              call   find_name1
830/      407 : 7B 15                jr     C, find_name4   ;
Ende wenn gefunden
831/      409 :                ; Suche in externer Tabelle
832/      409 : B0 E2                clr    R2
833/      40B : 44 08 E2              or     R2, reg_08      ; HI
ext. Proc-Table
834/      40E : 6B 0D                jr     Z, find_name3   ;
    
```

```

wenn 0 (= nicht vorhanden)
  835/      410 : 38 09                ld    R3, reg_09    ; L0
ext. Proc-Table
  836/      412 :                ;
  837/      412 : 48 E0            find_name1: ld    R4, R0    ;
akt. Position Basicprogramm
  838/      414 : 58 E1                ld    R5, R1        ; d.h.
Procedur-Name
  839/      416 : C2 82            find_name2: ldc    R8, @RR2  ;
Länge Procedurname
  840/      418 : A6 E8 FF                cp    R8, #0FFh    ;
tap_prc-Ende ?
  841/      41B : EB 02                jr    NZ, comp_name
  842/      41D : CF                find_name3: rcf
  843/      41E : AF                find_name4: ret
  844/      41F :
  845/      41F :                ; Namensvergleich
  846/      41F : A0 E2            comp_name: incw   RR2        ; auf
nächstes Zeichen Procedur-Name
  847/      421 : C2 72                ldc    R7, @RR2    ; Zeichen
nach R7 f. Vergleich
  848/      423 : D6 03 91            call   next_char    ;
gesuchte Procedur nächstes Zeichen
  849/      426 : EB 14                jr    NZ, comp_name2  ;
passt nicht -> nächsten Prozedurnamen vergleichen
  850/      428 : 8A F5                djnz   R8, comp_name  ;
weiter bis alle Zeichen verglichen
  851/      42A :                ; Procedur-Name in Tabelle
gefunden
  852/      42A : C2 60                ldc    R6, @RR0    ; Zeichen
an akt. Position Basicprogramm
  853/      42C : D6 03 E2            call   is_char      ; ist
der gesuchte Name länger?
  854/      42F : 7B 0A                jr    C, comp_name1  ;
dann weitersuchen
  855/      431 : A0 E2                incw   RR2
  856/      433 : C2 82                ldc    R8, @RR2    ;
sonst Proceduradresse nach RR8
  857/      435 : A0 E2                incw   RR2
  858/      437 : C2 92                ldc    R9, @RR2
  859/      439 : DF                scf                                ; Cy=gefunden
  860/      43A : AF                ret
  861/      43B :                ; nächsten Prozedurnamen vergleichen
  862/      43B : 8E                comp_name1: inc    R8        ;
Restlänge v. Namen + 1
  863/      43C : 8E                comp_name2: inc    R8        ; + 2
Byte Adresse
  864/      43D : 8E                inc    R8
  865/      43E : 02 38                add    R3, R8        ; zu
Pos in Proc-Table addieren
  866/      440 : 16 E2 00            adc    R2, #0

```

```

867/      443 : 08 E4          ld    R0, R4          ; akt.
Position Basicprogramm
868/      445 : 18 E5          ld    R1, R5          ;
rücksetzen
869/      447 : 8B CD          jr    find_name2     ;
weitersuchen
870/      449 :
871/      449 :                ;-----
-----
872/      449 :                ; Prozedur/Fkt.namen suchen,
Eingabeparameter ablegen, Prozedur ausführen
873/      449 :                ;-----
-----
874/      449 :
875/      449 : 80 E0          func:    decw    RR0
876/      44B :
877/      44B :                ; zu c_PROOC3
878/      44B : D6 04 00      func1:    call   find_name    ;
Prozedurnamen suchen
879/      44E : C2 A0          ldc    R10, @RR0     ;
Zeichen an akt. Position Basicprogramm
880/      450 : A6 EA 5B          cp    R10, #'['     ; folgen
Eingabeparameter?
881/      453 : EB 1A          jr    NZ, func3     ; nein
882/      455 : A0 E0          incw   RR0
883/      457 :                ; Eingabeparameter
evaluieren und auf Stack ablegen
884/      457 : 70 E8          func2:    push   R8          ;
Proc-Adr. sichern
885/      459 : 70 E9          push   R9
886/      45B : D6 04 C7          call   expr          ;
Eingabeparameter evaluieren -> RR2
887/      45E : 50 E9          pop    R9            ; Proc-Adr.
restaurieren
888/      460 : 50 E8          pop    R8
889/      462 : 7C 5D          ld    R7, #']'
890/      464 : D6 03 91          call   next_char
891/      467 : 6B 06          jr    Z, func3     ; wenn
Ende Parameterliste
892/      469 : 70 E3          push   R3            ; sonst
Wert auf Stack
893/      46B : 70 E2          push   R2
894/      46D : 8B E8          jr    func2         ; und
nächsten Parameter
895/      46F :                ; X1..Xn-1 liegen auf Stack, Xn in
RR2
896/      46F : 48 E2          func3:    ld    R4, R2          ; Y
= X setzen
897/      471 : 58 E3          ld    R5, R3
898/      473 : D4 E8          call   @RR8         ;
Procedur starten

```



```

      899/      475 : 31 10                srp      #10h          ; zur
Sicherheit f. ext. Prozeduren
      900/      477 : DF                scf                      ; Cy=Erfolg
      901/      478 : AF                ret
      902/      479 :
      903/      479 :                    ;-----
-----
      904/      479 :                    ; Ausdruck
      905/      479 :                    ; Variable, Prozedur[..], Hex-Zahl,
Dez-Zahl
      906/      479 :                    ;-----
-----
      907/      479 :
      908/      479 : D6 03 D9          factor:   call    is_letter
      909/      47C : FB 17                jr      NC, factor1    ; wenn
kein Buchstabe A..Z
      910/      47E : C2 70                ldc     R7, @RR0
      911/      480 : FE                inc     R15
      912/      481 : D6 03 E2          call    is_char        ;
folgt Buchstabe oder Zahl?
      913/      484 : 7B C3                jr      C, func        ; ja,
dann Prozedurname
      914/      486 :                    ; sonst einzelne Variable
A..Z
      915/      486 : 26 E6 41          sub     R6, #41h       ; 'A'
      916/      489 : 90 E6                rl      R6              ; *2
      917/      48B : 06 E6 20          add     R6, #20h       ;
Register 20h..53h
      918/      48E : E3 26                ld      R2, @R6        ; Wert
nach R2/R3 (Y)
      919/      490 : 6E                inc     R6
      920/      491 : E3 36                ld      R3, @R6
      921/      493 : DF                scf
      922/      494 : AF                ret
      923/      495 :
      924/      495 : A7 EF 3B          factor1:  cp      @R15, #';'    ;
Statement-Ende
      925/      498 : 6B 19                jr      Z, factor3
      926/      49A : A7 EF 0D          cp      @R15, #0Dh     ; CR,
Zeilenende
      927/      49D : 6B 14                jr      Z, factor3
      928/      49F : FE                inc     R15
      929/      4A0 : C3 F0          factor2:  ldci   @R15, @RR0    ;
Ablegen in Konvertierungspuffer
      930/      4A2 : 00 EF                dec     R15
      931/      4A4 : D6 03 F2          call    is_hexdigit
      932/      4A7 : FE                inc     R15
      933/      4A8 : 7B F6                jr      C, factor2    ;
solange (Hex-)Ziffer 0..F
      934/      4AA : 80 E0          decw   RR0
      935/      4AC : D6 01 D9          call    number        ; Zahl

```

```

konvertieren
  936/    4AF : 28 E4                ld    R2, R4
  937/    4B1 : 38 E5                ld    R3, R5
  938/    4B3 : AF                    factor3:  ret
  939/    4B4 :
  940/    4B4 :                      ;-----
-----
  941/    4B4 :                      ; geklammerter Ausdruck
  942/    4B4 :                      ;-----
-----
  943/    4B4 :
  944/    4B4 : 7C 28                term:    ld    R7, #'(' ;
linke Klammer
  945/    4B6 : D6 03 91             call   next_char
  946/    4B9 : EB BE                jr     NZ, factor ; wenn
keine Klammer
  947/    4BB : D6 04 C7             call   expr ;
Ausdruck berechnen
  948/    4BE : FB 06                jr     NC, term1
  949/    4C0 : 7C 29                ld    R7, #'')' ; rechte
Klammer
  950/    4C2 : D6 03 91             call   next_char
  951/    4C5 : EF                    ccf
  952/    4C6 : AF                    term1:   ret
  953/    4C7 :
  954/    4C7 :                      ;-----
-----
  955/    4C7 :                      ; Ausdruck berechnen von links nach
rechts (Kettenrechnen)
  956/    4C7 :                      ; Rückgabe Wert in RR2
  957/    4C7 :                      ;-----
-----
  958/    4C7 :
  959/    4C7 : D6 04 B4             expr:    call   term ;
linker Wert
  960/    4CA : FB 09                jr     NC, expr2
  961/    4CC : D6 03 91             expr1:   call   next_char
  962/    4CF : D6 03 9A             call   oper ;
Operator
  963/    4D2 : 6B 04                jr     Z, expr3 ; wenn
Operator gefunden
  964/    4D4 : DF                    scf ; sonst
Fehler
  965/    4D5 : 80 E0             expr2:   decw   RR0
  966/    4D7 : AF                    ret
  967/    4D8 :                      ;
  968/    4D8 : 70 E8             expr3:   push   R8 ; RR8
= Op-Fkt.Adresse
  969/    4DA : 70 E9             push   R9
  970/    4DC : 70 E2             push   R2 ;
aktueller linker Wert (Y)

```

```

    971/    4DE : 70 E3                push   R3
    972/    4E0 : D6 04 B4            call   term      ;
rechter Wert
    973/    4E3 : 48 E2                ld     R4, R2    ; nach
RR4 (X)
    974/    4E5 : 58 E3                ld     R5, R3
    975/    4E7 : 50 E3                pop    R3        ; Y
restaurieren
    976/    4E9 : 50 E2                pop    R2
    977/    4EB : 50 E9                pop    R9        ; Adr.
restaurieren
    978/    4ED : 50 E8                pop    R8
    979/    4EF : D4 E8                call   @RR8     ;
berechnen
    980/    4F1 :                      ;
    981/    4F1 : 8B D9                jr     expr1     ; weitere
Operationen
    982/    4F3 :                      ;
    983/    4F3 :                      ;-----
-----
    984/    4F3 :                      ; holt Speicheradresse zu Variable
A..Z
    985/    4F3 :                      ; out RR8 = Adr. in Registerspeicher
(20h..53h)
    986/    4F3 :                      ;-----
-----
    987/    4F3 :                      ;
    988/    4F3 : D6 03 91            get_var: call   next_char
    989/    4F6 : 26 E6 41            sub    R6, #'A'
    990/    4F9 : 02 66                add    R6, R6
    991/    4FB : 8C 20                ld     R8, #20h ; Register
20h..53h
    992/    4FD : 02 86                add    R8, R6
    993/    4FF : AF                    ret
    994/    500 :                      ;
    995/    500 :                      ;-----
-----
    996/    500 :                      ; Zeilennummer ermitteln
    997/    500 :                      ; ret: RR4 = zeilennummer, RR0 =
Zeilenanfang; Cy=1 Programmanfang
    998/    500 :                      ;-----
-----
    999/    500 :                      ;
1000/    500 :                      ; UP zu GOTO
1001/    500 :                      linum:   ;auf Zeilenanfang
1002/    500 : C2 40                ldc    R4, @RR0
1003/    502 : 80 E0                decw   RR0      ;
aktuelle Zeile rückwärts
    1004/   504 : 76 E4 80            tm     R4, #80h ; Bit7
gesetzt?
    1005/   507 : 6B F7                jr     Z, linum ;

```

```

rückwärts bis Zeilenanfang
1006/    509 : A0 E0                incw    RR0
1007/    50B :                    ;Programmmanfang erreicht?
1008/    50B : 56 E4 7F            and     R4, #7Fh    ; strip
hi bit
1009/    50E : A4 06 E0            cp     R0, reg_06   ;
Vergleich Adr. mit Programmmanfang
1010/    511 : EB 06                jr     NZ, linum1   ;
1011/    513 : A4 07 E1            cp     R1, reg_07
1012/    516 : DF                  scf
1013/    517 : 6B 0F                jr     Z, linum2    ;
Programmmanfang erreicht
1014/    519 :                    ;
1015/    519 :                    linum1: ;steht vor gesetztem hi-
bit ein CR? dann Zeilenanfang gefunden
1016/    519 : 80 E0                decw    RR0
1017/    51B : C2 50                ldc     R5, @RR0
1018/    51D : A0 E0                incw    RR0
1019/    51F : A6 E5 0D            cp     R5, #0Dh    ; CR,
Suche Zeilenende
1020/    522 : 6B 04                jr     Z, linum2    ; ja,
Zeilenanfang gefunden
1021/    524 : 80 E0                decw    RR0
1022/    526 : 8B D8                jr     linum        ; sonst
weitersuchen
1023/    528 :                    ;
1024/    528 : A0 E0                linum2: incw    RR0
1025/    52A : C2 50                ldc     R5, @RR0   ;
RR4=Zeilennummer
1026/    52C : 80 E0                decw    RR0
1027/    52E : AF                    ret
1028/    52F :
1029/    52F :                    ;-----
-----
1030/    52F :                    ; Ausgabe "text"
1031/    52F :                    ;-----
-----
1032/    52F :
1033/    52F : 7C 22                prnstr: ld     R7, #'"'
1034/    531 : D6 03 91            call    next_char
1035/    534 : EB 0C                jr     NZ, prnstr2  ; kein
Quote -> Abbruch
1036/    536 : D6 03 91            prnstr1: call   next_char ;
nächstes Zeichen
1037/    539 : 6B 09                jr     Z, prnstr3   ; Ende
beim abschließenden Quote
1038/    53B : 58 E6                ld     R5, R6       ; nach
R5
1039/    53D : D6 08 18            call    putch       ; mit
PUT_CHAR ausgeben
1040/    540 : 8B F4                jr     prnstr1

```

```

1041/      542 :      ;
1042/      542 : 80 E0      prnstr2:  decw  RR0      ;
Pointer zurückstellen
1043/      544 : AF      prnstr3:  ret
1044/      545 :
1045/      545 :
1046/      545 :      ;-----
-----
1047/      545 :      ; logischer Ausdruck
1048/      545 :      ; expr. relop expr., ret NZ wenn
true, Z wenn false
1049/      545 :      ;-----
-----
1050/      545 :
1051/      545 : D6 04 C7      relop:  call  expr      ; linker
wert
1052/      548 : B0 EA      clr    R10
1053/      54A : 9C 02      ld     R9, #2      ; max 2
Zeichen
1054/      54C : 8C 10      relop1:  ld     R8, #10h
1055/      54E : C2 60      ldc    R6, @RR0
1056/      550 : A6 E6 3C      cp     R6, #'<'
1057/      553 : 6B 0E      jr     Z, relop2      ; < R8=10
1058/      555 : 90 E8      rl     R8
1059/      557 : A6 E6 3E      cp     R6, #'>'
1060/      55A : 6B 07      jr     Z, relop2      ; > R8=20
1061/      55C : 90 E8      rl     R8
1062/      55E : A6 E6 3D      cp     R6, #'='
1063/      561 : EB 06      jr     NZ, relop3      ; =
R8=40
1064/      563 : 42 A8      relop2:  or     R10, R8      ;
<= R8=50, >= R8=60
1065/      565 : A0 E0      incw  RR0
1066/      567 : 9A E3      djnz  R9, relop1
1067/      569 :
1068/      569 : 70 EA      relop3:  push  R10
1069/      56B : 70 E2      push  R2
1070/      56D : 70 E3      push  R3
1071/      56F : D6 04 C7      call  expr      ;
rechter Wert
1072/      572 : 48 E2      ld     R4, R2      ; RR4 =
rechter Wert
1073/      574 : 58 E3      ld     R5, R3
1074/      576 : 50 E3      pop   R3      ; RR2 =
linker Wert
1075/      578 : 50 E2      pop   R2
1076/      57A : D6 01 41      call  relcmp      ;
Vergleich
1077/      57D : 50 E8      pop   R8
1078/      57F : 54 0F E8      and   R8, reg_0F      ; Mit
relop verknüpfen

```

```

1079/      582 : AF                ret                ; Z=1 =>
relop false
1080/      583 :
1081/      583 :                ;-----
-----
1082/      583 :                ; REM kommentar
1083/      583 :                ;-----
-----
1084/      583 :
1085/      583 : 7C 3B      c_REM:      ld    R7, #';'
1086/      585 : D6 03 91  c_REM1:     call  next_char
1087/      588 : 6B 06                jr    Z, c_REM2    ;
überlesen bis ;
1088/      58A : A6 E6 0D      cp    R6, #0Dh    ; CR
1089/      58D : DF                scf
1090/      58E : EB F5                jr    NZ, c_REM1   ; oder
Zeilenende
1091/      590 : 80 E0      c_REM2:     decw  RR0
1092/      592 : AF      c_REM3:     ret
1093/      593 :
1094/      593 :
1095/      593 :                ;-----
-----
1096/      593 :                ; Goto, If, Else: Rest bis
Zeilenende übergehen
1097/      593 :                ;-----
-----
1098/      593 :
1099/      593 : D6 05 83      skip:      call  c_REM        ;
Rest übergehen
1100/      596 : 7B FA                jr    C, c_REM3    ; Ende
bei CR
1101/      598 : A0 E0                incw  RR0          ; sonst
auch nächste Anw.
1102/      59A : 8B F7                jr    skip         ;
übergehen
1103/      59C :
1104/      59C :                ;-----
-----
1105/      59C :                ; LET variable=ausdruck
1106/      59C :                ;-----
-----
1107/      59C :
1108/      59C : D6 04 F3      c_LET:      call  get_var      ;
@RR8 = Variablenadr.
1109/      59F : 70 E8                push  R8
1110/      5A1 : A0 E0                incw  RR0
1111/      5A3 : D6 04 C7      call  expr        ; RR2 =
ausdruck
1112/      5A6 : 50 E8                pop   R8
1113/      5A8 : F3 82      c_LET1:     ld    @R8, R2      ;

```

```

Wert abspeichern
 1114/    5AA : 8E                inc    R8
 1115/    5AB : F3 83            ld     @R8, R3
 1116/    5AD : AF                ret
 1117/    5AE :
 1118/    5AE :
 1119/    5AE : ;-----
-----
 1120/    5AE : ; TRAP
 1121/    5AE : ; TRAP bedingung T0 ausdruck
 1122/    5AE : ;-----
-----
 1123/    5AE : ; Trap setzen
 1124/    5AE : ; der Rest der Anweisung wird hier
übergangen
 1125/    5AE :
 1126/    5AE : 09 04            c_TRAP: ld    reg_04, R0 ;
RR0 = akt. Pos. in Programm
 1127/    5B0 : 19 05                ld     reg_05, R1
 1128/    5B2 : 8B CF            c_TRAP1: jr    c_REM
 1129/    5B4 :
 1130/    5B4 : ;-----
-----
 1131/    5B4 : ; PROC [variablenliste] =
prozedurname [parameterliste]
 1132/    5B4 : ; PROC [Y1,Y2,Y3,...,Ym] =
prozedurname [X1,X2,X3,...,Xn]
 1133/    5B4 : ;-----
-----
 1134/    5B4 : ; bei Prozeduraufruf gilt:
 1135/    5B4 : ; Stack SP+2m+2n    Ym-1
je 2 Byte Werte
 1136/    5B4 : ; ..            Ym-2
 1137/    5B4 : ; ..            ..
 1138/    5B4 : ; SP+2n+4    Y2
 1139/    5B4 : ; SP+2n+2    Y1
 1140/    5B4 : ; SP+2n    (intern f.
interpreter)
 1141/    5B4 : ; SP+2n-2    X1
 1142/    5B4 : ; ..            ..
 1143/    5B4 : ; SP+4    Xn-2
 1144/    5B4 : ; SP+2    Xn-1
 1145/    5B4 : ; SP    returnadr. zum
interpreter
 1146/    5B4 : ; die rechten Parameter Ym, Xn
werden in Registern abgelegt
 1147/    5B4 : ; out RR2 = Ym, in RR4 Xn
 1148/    5B4 : ;-----
-----
 1149/    5B4 :
 1150/    5B4 : 70 04            c_PROC: push  reg_04 ;

```

```

aktuelle TRAP sichern
  1151/      5B6 : 70 05                push   reg_05
  1152/      5B8 : 09 04                ld     reg_04, R0   ;
aktuelle Programmposition sichern
  1153/      5BA : 19 05                ld     reg_05, R1
  1154/      5BC :                      ;Ausgabeparameter auf Stack
initialisieren
  1155/      5BC : C2 60                ldc    R6, @RR0    ;
nächstes Zeichen testen
  1156/      5BE : A6 E6 5B            cp     R6, #'['    ; folgen
Ausgabeparameter Y?
  1157/      5C1 : EB 15                jr     NZ, c_PROC3 ; nein
  1158/      5C3 : A0 E0                incw   RR0
  1159/      5C5 : B0 E8                clr    R8          ; 0
  1160/      5C7 : 7C 5D                ld     R7, #']'
  1161/      5C9 : A0 E0                c_PROC1: incw   RR0
  1162/      5CB : D6 03 91            call   next_char
  1163/      5CE : 6B 06                jr     Z, c_PROC2
  1164/      5D0 : 70 E8                push   R8          ;
Stackspeicher f. Ausgabeparameter
  1165/      5D2 : 70 E8                push   R8          ; belegen
(mit Wert 0)
  1166/      5D4 : 8B F3                jr     c_PROC1
  1167/      5D6 : A0 E0                c_PROC2: incw   RR0
  1168/      5D8 :                      ;Prozedurnamen suchen,
Eingabeparameter evaluieren und ablegen, Prozedur ausführen
  1169/      5D8 : D6 04 4B            c_PROC3: call   func1
  1170/      5DB :                      ;Ausgabeparameter von Stack in
Variablen schreiben
  1171/      5DB : 08 04                ld     R0, reg_04  ;
Programmposition auf Anfang setzen
  1172/      5DD : 18 05                ld     R1, reg_05
  1173/      5DF : 7C 5B                ld     R7, #'['
  1174/      5E1 : D6 03 91            call   next_char
  1175/      5E4 : EB 18                jr     NZ, c_PROC6 ; wenn
keine Ausgabeparameter
  1176/      5E6 : 7C 5D                ld     R7, #']'
  1177/      5E8 : D6 04 F3            c_PROC4: call   get_var ;
get Variable
  1178/      5EB : D6 03 91            call   next_char
  1179/      5EE : 6B 0B                jr     Z, c_PROC5 ; Ende
bei ]
  1180/      5F0 : 50 EA                pop    R10         ;
Parameter
  1181/      5F2 : F3 8A                ld     @R8, R10    ; in
Variable schreiben (Hi)
  1182/      5F4 : 50 EA                pop    R10
  1183/      5F6 : 8E                    inc    R8
  1184/      5F7 : F3 8A                ld     @R8, R10    ; Lo
  1185/      5F9 : 8B ED                jr     c_PROC4     ;
nächsten Parameter

```



```

1186/      5FB : D6 05 A8          c_PROC5:   call    c_LET1
1187/      5FE : 50 05          c_PROC6:   pop     reg_05      ;
TRAP wiederherstellen
1188/      600 : 50 04                pop     reg_04
1189/      602 : 8B AE                jr      c_TRAP1
1190/      604 :
1191/      604 :                      ;-----
-----
1192/      604 :                      ; GOTO ausdruck
1193/      604 :                      ;-----
-----
1194/      604 :                      ; suche passende zeile oder
nächstgrößere; bei Programmende -> Ende
1195/      604 :
1196/      604 : D6 04 C7          c_GOTO:    call    expr      ;
RR2 = ausdruck
1197/      607 : D6 05 00          call    linum     ; RR4 =
aktuelle Zeilennummer
1198/      60A : D6 01 41          call    relcmp    ;
vergleichen
1199/      60D : 76 0F 50          tm     reg_0F, #50h ; <=
1200/      610 : EB 1F            jr      NZ, c_GOTO3 ; wenn
ausdruck <= akt. Zeile
1201/      612 :
1202/      612 :                      c_GOTO1:    ; Suche vorwärts
1203/      612 : D6 05 93          call    skip      ; Rest
bis Zeilenende übergehen
1204/      615 : A0 E0            incw   RR0
1205/      617 : C2 40            ldc    R4, @RR0   ; RR4 =
nächste Zeilennummer
1206/      619 : A0 E0            incw   RR0
1207/      61B : C2 50            ldc    R5, @RR0
1208/      61D : 80 E0            decw   RR0
1209/      61F : 90 E4            rl     R4          ; Bit7 nach
Cy und Bit0
1210/      621 : CF              rcf                    ; Cy=0
1211/      622 : C0 E4            rrc    R4          ; Bit7=0,
Cy=orig. Bit7
1212/      624 : FB 08            jr     NC, c_GOTO2  ;
Programmende erreicht?
1213/      626 : D6 01 41          call    relcmp
1214/      629 : 76 0F 50          tm     reg_0F, #50h ; <=
1215/      62C : 6B E4            jr     Z, c_GOTO1  ;
weiter, solange ausdruck <= akt. Zeile
1216/      62E :                      c_GOTO2:    ; Programmende erreicht
1217/      62E : 80 E0            decw   RR0
1218/      630 : AF              ret
1219/      631 :
1220/      631 : D6 05 00          c_GOTO3:    call    linum     ;
RR4 = aktuelle Zeilennummer
1221/      634 : FB 07            jr     NC, c_GOTO4  ;

```

```

solange
  1222/    636 :
  1223/    636 :                ; Wenn Zeile gefunden
  1224/    636 : A0 FE                incw   gpr      ;
Stackpointer
  1225/    638 : A0 FE                incw   gpr      ;
Stackpointer
  1226/    63A : 8D 07 3E                jp     run2     ; Zeile
abearbeiten
  1227/    63D :
  1228/    63D :                c_GOT04:    ; suche Rückwärts
  1229/    63D : D6 01 41                call   relcmp
  1230/    640 : 76 0F 60                tm     reg_0F, #60h    ; >=
  1231/    643 : EB E9                jr     NZ, c_GOT02
  1232/    645 : 80 E0                decw   RR0     ; ein
Zeichen zurück
  1233/    647 : 8B E8                jr     c_GOT03    ;
weilersuchen
  1234/    649 :
  1235/    649 :                ;-----
-----
  1236/    649 :                ; IF bedingung THEN anweisungen
  1237/    649 :                ;-----
-----
  1238/    649 :
  1239/    649 : 56 0F FE                c_IF:      and     reg_0F, #0FEh
; Bit0 = 0
  1240/    64C : D6 05 45                call   relop     ;
bedingung auswerten
  1241/    64F : EB 05                jr     NZ, c_IF1  ;
Bedingung erfüllt
  1242/    651 :                ; -> nächste
Anweisung abearbeiten (nach ;)
  1243/    651 : 20 0F                inc    reg_0F    ; Bit0
= 1 ( -> ELSE)
  1244/    653 : 8D 05 93                jp     skip     ; Rest bis
Zeilenende übergehen
  1245/    656 : AF                c_IF1:     ret
  1246/    657 :
  1247/    657 :                ;-----
-----
  1248/    657 :                ; ELSE anweisungen
  1249/    657 :                ;-----
-----
  1250/    657 :
  1251/    657 : 76 0F 01                c_ELSE:    tm     reg_0F, #1    ;
ELSE aktiv?
  1252/    65A : 6D 05 93                jp     Z, skip   ; nein,
Rest bis Zeilenende übergehen
  1253/    65D : 56 0F FE                and     reg_0F, #0FEh ;
Bit0 = 0 (kein ELSE aktiv)

```

```

1254/      660 : AF                      ret
1255/      661 :
1256/      661 :                          ;-----
-----
1257/      661 :                          ; RETURN
1258/      661 :                          ;-----
-----
1259/      661 :
1260/      661 : A6 0E 00      c_RETURN:  cp   reg_0E, #0      ;
Verschachtelungstiefe 0?
1261/      664 : EB 06                      jr   NZ, c_RETURN2      ;
nein
1262/      666 : 46 0E 20      or   reg_0E, #20h      ; Bit5
= 1
1263/      669 : 8D 01 1B      c_RETURN1: jp   p_div6      ;
reg_0F Bit7=1
1264/      66C :
1265/      66C : 00 0E      c_RETURN2: dec   reg_0E      ;
Verschachtelungstiefe verringern
1266/      66E : 50 E6                      pop  R6      ; Return-
Adr.
1267/      670 : 50 E7                      pop  R7
1268/      672 : 50 E0                      pop  R0      ; RR0= Pos.
nach GOSUB
1269/      674 : 50 E1                      pop  R1
1270/      676 : 56 0F FE      and   reg_0F, #0FEh      ;
Bit0 = 0 (kein ELSE aktiv)
1271/      679 : 30 E6                      jp   @RR6      ; return
1272/      67B :
1273/      67B :                          ;-----
-----
1274/      67B :                          ; GOSUB ausdruck
1275/      67B :                          ;-----
-----
1276/      67B :
1277/      67B : 88 E0      c_GOSUB:  ld   R8, R0      ;
aktuelle Position
1278/      67D : 98 E1                      ld   R9, R1      ;
sichern
1279/      67F : D6 05 83      call  c_REM      ;
RR0=Kommandoende
1280/      682 : 50 EA                      pop  R10      ; Return-
Adresse vom Stack
1281/      684 : 50 EB                      pop  R11
1282/      686 : 70 E1                      push R1      ; Pos.
nach GOSUB
1283/      688 : 70 E0                      push R0      ; auf
Stack
1284/      68A : 70 EB                      push R11      ; Return-
Adresse restaurieren
1285/      68C : 70 EA                      push R10

```

```

1286/      68E : 08 E8                ld    R0, R8          ; orig.
Position
1287/      690 : 18 E9                ld    R1, R9          ;
rückschreiben
1288/      692 : 20 0E                c_GOSUB1: inc    reg_0E      ;
Verschachtelungstiefe erhöhen
1289/      694 : 76 0E 10            tm    reg_0E, #10h    ; > 15
?
1290/      697 : 6D 06 04            jp    Z, c_GOTO      ; nein,
UP aufrufen
1291/      69A :                      ; max Verschachtelungstiefe
erreicht
1292/      69A : D6 05 83            call   c_REM          ; bis
Kommandoende überlesen
1293/      69D : 8B CA                jr    c_RETURN1      ; und
Verschachtelungstiefe verringern
1294/      69F :
1295/      69F :                      ;-----
-----
1296/      69F :                      ; WAIT ausdruck
1297/      69F :                      ; Schleifendauer 1 ms bei 8 MHz
Taktfrequenz
1298/      69F :                      ;-----
-----
1299/      69F :
1300/      69F : D6 04 C7            c_WAIT:   call   expr      ;
RR2 = ausdruck
1301/      6A2 : 68 E2                ld    R6, R2
1302/      6A4 : 42 63                or    R6, R3          ; = 0?
1303/      6A6 : 6B 10                jr    Z, c_WAIT3      ; dann
kein Wait
1304/      6A8 : 40 E6                c_WAIT1:   da    R6
1305/      6AA : 6C 00                ld    R6, #0
1306/      6AC : 7C B4                ld    R7, #0B4h
1307/      6AE : 80 E6                c_WAIT2:   decw   RR6
1308/      6B0 : ED 06 AE            jp    NZ, c_WAIT2
1309/      6B3 : 80 E2                decw   RR2
1310/      6B5 : ED 06 A8            jp    NZ, c_WAIT1
1311/      6B8 : AF                c_WAIT3:   ret
1312/      6B9 :
1313/      6B9 :                      ;-----
-----
1314/      6B9 :                      ; CALL ausdruck
1315/      6B9 :                      ;-----
-----
1316/      6B9 :
1317/      6B9 : D6 04 C7            c_CALL:   call   expr      ;
RR2 = ausdruck
1318/      6BC : D4 E2                call   @RR2          ; UP
aufrufen (berechnete Adr.)
1319/      6BE : 31 10                srp    #10h          ; RP auf

```

```

Standard
  1320/    6C0 : AF                ret
  1321/    6C1 :
  1322/    6C1 :                ;-----
-----
  1323/    6C1 :                ; STOP
  1324/    6C1 :                ;-----
-----
  1325/    6C1 :
  1326/    6C1 : 46 0F 08        c_STOP:    or    reg_0F, #8    ;
Bit3 = 1
  1327/    6C4 : AF                ret
  1328/    6C5 :
  1329/    6C5 :                ;-----
-----
  1330/    6C5 :                ; END
  1331/    6C5 :                ;-----
-----
  1332/    6C5 :
  1333/    6C5 : 46 0F 02        c_END:    or    reg_0F, #2    ;
Bit1 = 1
  1334/    6C8 : AF                ret
  1335/    6C9 :
  1336/    6C9 :                ;-----
-----
  1337/    6C9 :                ; CLRTRP
  1338/    6C9 :                ;-----
-----
  1339/    6C9 :
  1340/    6C9 : B0 04          c_CLRTRAP: clr    reg_04    ;
Trap auf 0 setzen
  1341/    6CB : B0 05                clr    reg_05
  1342/    6CD : AF                ret
  1343/    6CE :
  1344/    6CE :                ;-----
-----
  1345/    6CE :                ; PRINTHEX "text" ausdruck
  1346/    6CE :                ;-----
-----
  1347/    6CE :
  1348/    6CE : D6 05 2F        c_PRINTHEX: call   prnstr    ;
Ausgabe text
  1349/    6D1 : D6 04 C7                call   expr    ;
Ausdruck berechnen
  1350/    6D4 : FB 22                jr    NC, c_PRINT3    ; bei
Fehler o. ohne Ausdruck
  1351/    6D6 : D6 01 5E                call   tohex    ;
Konvertierung nach Hex-String
  1352/    6D9 :                ; In Buffer ab
#14h
  1353/    6D9 : AC 05                ld    R10, #5    ; 5

```

```

Stellen
 1354/    6DB : 8B 0D                jr    c_PRINT1
 1355/    6DD :
 1356/    6DD :                      ;-----
-----
 1357/    6DD :                      ; PRINT "text" ausdruck
 1358/    6DD :                      ;-----
-----
 1359/    6DD :
 1360/    6DD : D6 05 2F            c_PRINT:  call  prnstr    ;
Ausgabe text
 1361/    6E0 : D6 04 C7            call  expr    ;
Ausdruck berechnen
 1362/    6E3 : FB 13                jr    NC, c_PRINT3    ; bei
Fehler o. ohne Ausdruck keine Ausgabe
 1363/    6E5 :                      ; Ausdruck anzeigen
 1364/    6E5 : D6 01 82            call  todez    ;
Konvertierung nach dezimal
 1365/    6E8 : AC 06                ld    R10, #6    ; 6
Stellen
 1366/    6EA : BC 14                c_PRINT1: ld    R11, #14h ;
Konvertierungspuffer
 1367/    6EC : 70 E5                c_PRINT2: push  R5
 1368/    6EE : E3 5B                ld    R5, @R11
 1369/    6F0 : BE                    inc   R11
 1370/    6F1 : D6 08 18            call  putch    ;
Stelle ausgeben
 1371/    6F4 : 50 E5                pop   R5
 1372/    6F6 : AA F4                djnz  R10, c_PRINT2
 1373/    6F8 :                      ; Am Ende Zeilenende
ausgeben, falls kein Komma
 1374/    6F8 : C2 60                c_PRINT3: ldc   R6, @RR0
 1375/    6FA : A6 E6 2C            cp    R6, #','    ; folgt
Komma?
 1376/    6FD : EB 11                jr    NZ, c_PRINT5    ; nein
-> CR
 1377/    6FF : A0 E0                incw  RR0
 1378/    701 : C2 60                ldc   R6, @RR0
 1379/    703 : A6 E6 3B            cp    R6, #';'    ; folgt
Kdo-Ende?
 1380/    706 : 6B 07                jr    Z, c_PRINT4    ; ja ->
kein CR ausgeben
 1381/    708 : A6 E6 0D            cp    R6, #0Dh    ; oder
folgt Zeilenende CR?
 1382/    70B : 6B 02                jr    Z, c_PRINT4    ; dann
ebenfalls kein CR ausgeben
 1383/    70D : 80 E0                decw  RR0
 1384/    70F : AF                    c_PRINT4: ret
 1385/    710 : 5C 0D                c_PRINT5: ld    R5, #0Dh    ; CR
 1386/    712 : 8D 08 18            jp    putch    ;
ausgeben (Zeilenende)

```

```

1387/    715 :
1388/    715 : ;-----
-----
1389/    715 : ; INPUT "text" variable
1390/    715 : ;-----
-----
1391/    715 :
1392/    715 : D6 05 2F      c_INPUT:  call  prnstr      ;
Ausgabe text
1393/    718 : D6 02 E9              call  p_input1      ;
Abfrage Wert
1394/    71B : D6 04 F3              call  get_var       ;
Variablenadresse
1395/    71E : 8D 05 A8              jp   c_LET1        ; und
Wert zuweisen
1396/    721 :
1397/    721 : ;-----
-----
1398/    721 : ; ???
1399/    721 : ;-----
-----
1400/    721 :
1401/    721 : ;loc_721:
1402/    721 : E6 0F 04          ld   reg_0F, #4
1403/    724 : 8B 10             jr   run1
1404/    726 :
1405/    726 : ;-----
-----
1406/    726 : ; ???
1407/    726 : ;-----
-----
1408/    726 : ;loc_726:
1409/    726 : E6 0F 08          ld   reg_0F, #8
1410/    729 : 8B 0B             jr   run1
1411/    72B :
1412/    72B : ;-----
-----
1413/    72B : ; RUN
1414/    72B : ; in 6,7 = Startadr. Basic-Programm
1415/    72B : ; 8,9 = Adr. Prozedurtabelle
(oder 0)
1416/    72B : ; srp #10h; call $7fd
1417/    72B : ;-----
-----
1418/    72B :
1419/    72B : ; Programmstart
1420/    72B : B0 0F            run:   clr   reg_0F
1421/    72D : B0 0E            clr   reg_0E
1422/    72F : 08 06            ld   R0, reg_06    ; RR0 =
Startadr. Basic-Programm
1423/    731 : 18 07            ld   R1, reg_07

```

```

1424/      733 : D6 06 C9                call   c_CLRTRAP    ; clear
trap
1425/      736 :                          ;
1426/      736 : 31 10                run1:   srp        #10h    ;
Standard setzen
1427/      738 : 50 0A                pop     reg_0A     ;
Return-Adresse f. END
1428/      73A : 50 0B                pop     reg_0B
1429/      73C : 8B 0A                jr      run3
1430/      73E :
1431/      73E :                          ; nächste Zeile abarbeiten
1432/      73E : 76 0F 0A            run2:   tm        reg_0F, #0Ah
1433/      741 : EB 0A                jr      NZ, run4    ; END
1434/      743 : 66 0F 84            tcm    reg_0F, #84h  ; -7Ch
1435/      746 : 6B 05                jr      Z, run4     ; END
1436/      748 :
1437/      748 : C2 60                run3:   ldc      R6, @RR0  ;
R6=Hi(Zeilenummer)+$80
1438/      74A : 6E                          inc     R6
1439/      74B : 6A 02                djnz   R6, run5
1440/      74D :                          ;END
1441/      74D : 30 0A                run4:   jp      @reg_0A    ;
END bei Zeilennummer > 7Fxxh (32512)
1442/      74F :
1443/      74F :                          ;
1444/      74F : A4 E0 06            run5:   cp      reg_06, R0
1445/      752 : EB 05                jr      NZ, run6
1446/      754 : A4 E1 07            cp      reg_07, R1
1447/      757 : 6B 2C                jr      Z, run8
1448/      759 :                          ; Test auf TRAP
1449/      759 : 68 04                run6:   ld      R6, reg_04  ;
TRAP
1450/      75B : 44 05 E6            or      R6, reg_05
1451/      75E : 6B 25                jr      Z, run8     ; =0? kein
TRAP
1452/      760 :                          ; TRAP
1453/      760 : 70 E1                push   R1          ; akt.
Pogrammpos. sichern
1454/      762 : 70 E0                push   R0
1455/      764 : 08 04                ld     R0, reg_04  ; auf
TRAP-Adr. setzen
1456/      766 : 18 05                ld     R1, reg_05
1457/      768 : D6 05 45            call   relop      ;
logischen Ausdruck auswerten
1458/      76B : 6B 14                jr      Z, run7     ; wenn
nicht erfüllt
1459/      76D :                          ; wenn erfüllt
1460/      76D : D6 06 C9            call   c_CLRTRAP  ; TRAP
zurücksetzen
1461/      770 : 50 E6                pop     R6          ; aktuelle
Return-Adr

```



```

1462/      772 : 50 E7                pop    R7
1463/      774 : 80 E6                decw   RR6          ;
vermindern
1464/      776 : 70 E7                push   R7
1465/      778 : 70 E6                push   R6
1466/      77A : A0 E0                incw   RR0          ; akt.
Pogrammpos. (TRAP) erhöhen
1467/      77C : D6 06 92            call   c_GOSUB1    ; TRAP
ausführen
1468/      77F : 8B 38                jr     run14
1469/      781 :
1470/      781 :                      ; Zeile abarbeiten
1471/      781 : 50 E0                run7:   pop    R0
1472/      783 : 50 E1                pop    R1
1473/      785 : A0 E0                run8:   incw   RR0          ;
Zeilennummer übergehen
1474/      787 : A0 E0                incw   RR0
1475/      789 : C2 30                run9:   ldc    R3, @RR0    ;
erstes Zeichen (Kommando)
1476/      78B : A0 E0                incw   RR0
1477/      78D : A6 E3 3E            cp     R3, #'>'    ; ELSE?
1478/      790 : 6B 03                jr     Z, run10    ; dann
ELSE-Flag beibehalten
1479/      792 : 56 0F FE            and    reg_0F, #0FEh ;
sonst ELSE-Flag rücksetzen (Bit0)
1480/      795 : 6C 07                run10:  ld     R6, #HI(tab_kdo) ;
Liste der Kommandos
1481/      797 : 7C C8                ld     R7, #L0(tab_kdo)
1482/      799 : B0 E2                clr    R2          ; R2=0
1483/      79B : C2 86                run11:  ldc    R8, @RR6
1484/      79D : A2 83                cp     R8, R3      ;
Kommando in Liste suchen
1485/      79F : 6B 05                jr     Z, run12    ; wenn
gefunden
1486/      7A1 : A0 E6                incw   RR6
1487/      7A3 : 2E                    inc    R2
1488/      7A4 : 8B F5                jr     run11
1489/      7A6 :
1490/      7A6 : 02 22                run12:  add    R2, R2
1491/      7A8 : 6C 07                ld     R6, #HI(tab_kdo2) ;
Adressliste der Kommandos
1492/      7AA : 7C D9                ld     R7, #L0(tab_kdo2)
1493/      7AC : 02 72                add    R7, R2
1494/      7AE : 16 E6 00            adc    R6, #0
1495/      7B1 : 2C 0C                ld     R2, #0Ch
1496/      7B3 : C3 26                ldci   @R2, @RR6    ;
Adresse nach reg_0ch
1497/      7B5 : C3 26                ldci   @R2, @RR6
1498/      7B7 : D4 0C                run13:  call   @reg_0C      ;
Kommando aufrufen
1499/      7B9 :                      ;

```

```

1500/      7B9 : 7C 0D          run14:      ld    R7, #0Dh    ; CR
Zeilenende?
1501/      7BB : D6 03 91          call   next_char
1502/      7BE : 6D 07 3E          jp    Z, run2      ; dann
nächste Zeile
1503/      7C1 : A6 E6 3B          cp    R6, #' ; '   ;
Kommandoende?
1504/      7C4 : 6B C3          jr    Z, run9     ; dann
nächstes Kommando in Zeile
1505/      7C6 :                      ;
1506/      7C6 : 8B EF          jr    run13      ; sonst
Wiederholung aktuelles Kdo
1507/      7C8 :
1508/      7C8 :                      ;-----
-----
1509/      7C8 :                      ; Tabelle der BASIC-Kommandos
1510/      7C8 :                      ;-----
-----
1511/      7C8 :
1512/      7C8 : 4C          tab_kdo:  db    'L'      ; LET
1513/      7C9 : 4F          db    'O'      ; PROC
1514/      7CA : 47          db    'G'      ; GOTO
1515/      7CB : 46          db    'F'      ; IF..THEN
F..;
1516/      7CC : 3E          db    '>'      ; ELSE
>;
1517/      7CD : 52          db    'R'      ; RETURN
1518/      7CE : 53          db    'S'      ; GOSUB
1519/      7CF : 57          db    'W'      ; WAIT
1520/      7D0 : 4D          db    'M'      ; REM
1521/      7D1 : 43          db    'C'      ; CALL
1522/      7D2 : 54          db    'T'      ; STOP
1523/      7D3 : 45          db    'E'      ; END
1524/      7D4 : 21          db    '!'      ; TRAP..TO
!...
1525/      7D5 : 2F          db    '/'      ; CLRTRAP
1526/      7D6 : 50          db    'P'      ; PRINT
1527/      7D7 : 48          db    'H'      ; PRINTHEX
1528/      7D8 : 49          db    'I'      ; INPUT
1529/      7D9 :
1530/      7D9 : 05 9C          tab_kdo2: dw    c_LET
1531/      7DB : 05 B4          dw    c_PROC
1532/      7DD : 06 04          dw    c_GOTO
1533/      7DF : 06 49          dw    c_IF
1534/      7E1 : 06 57          dw    c_ELSE
1535/      7E3 : 06 61          dw    c_RETURN
1536/      7E5 : 06 7B          dw    c_GOSUB
1537/      7E7 : 06 9F          dw    c_WAIT
1538/      7E9 : 05 83          dw    c_REM
1539/      7EB : 06 B9          dw    c_CALL
1540/      7ED : 06 C1          dw    c_STOP

```

```

1541/    7EF : 06 C5                dw    c_END
1542/    7F1 : 05 AE                dw    c_TRAP
1543/    7F3 : 06 C9                dw    c_CLRTRAP
1544/    7F5 : 06 DD                dw    c_PRINT
1545/    7F7 : 06 CE                dw    c_PRINTHEX
1546/    7F9 : 07 15                dw    c_INPUT
1547/    7FB :
1548/    7FB :                      ;-----
-----
1549/    7FB :                      ;
1550/    7FB :                      ;-----
-----
1551/    7FB :
1552/    7FB : FF                db    0FFh
1553/    7FC : FF                db    0FFh
1554/    7FD :
1555/    7FD :                      ; Eintrittspunkt BASIC-Interpreter
1556/    7FD :                      ;RUN
1557/    7FD : 8D 07 2B            jp    run
1558/    800 :
1559/    800 :                      ; end of 'ROM'
1560/    800 :
1561/    800 :                      end

```

From:

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

Permanent link:

<https://hc-ddr.hucki.net/wiki/doku.php/elektronik/u883/listing?rev=1650374386>

Last update: **2022/04/19 13:19**

