Listing MP-BASIC

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

```
 AS V1.42 Beta [Bld 157] - Source File U883_MME_BAS.asm - Page 1 - 8/16/2021 14:5:33
       1/       0 :                     ; reass: Volker Pohlers 02.2007/07.2021
       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: CONT
      34/       0 :                                     ; Bit 3 08: STOP, STEP
      35/       0 :                                     ; Bit 4 10: <
      36/       0 :                                     ; Bit 5 20: >
      37/       0 :                                     ; Bit 6 40: =
      38/       0 :                                     ; Bit 7 80:
      39/       0 : =7FH                reg_7F      equ    7Fh
      40/       0 :
      41/       0 :                     ;reg_20h..reg53h Variablen A..Z (Doppelregister,
16 Bit)
```

```
      42/       0 :                     ;ab 54h frei nutzbar, z.B. f. Stack
      43/       0 :
      44/       0 :                     ; ext. Routinen
      45/       0 :                     ;saddrh equ    0E000h    ; startadr RAM s. init10
      46/       0 :                     ;saddr  equ    812h    ; startadr ROM s. init10
      47/       0 : =815H               getch   equ    815h
      48/       0 : =818H               putch   equ    818h
      49/       0 :
      50/       0 :
      51/       0 :                     ; Register
      52/       0 : =0FEH               gpr         equ    0FEh    ; General purpose
register bzw. Stackpointer, Highteil
      53/       0 :
      54/       0 :                     ;-----------------------------------------------
------------------------------
      55/       0 :                     ;
      56/       0 :                     ;-----------------------------------------------
------------------------------
      57/       0 :
      58/       0 : 08 00               irq0:       dw    800h
      59/       2 : 08 03               irq1:       dw    803h
      60/       4 : 08 06               irq2:       dw    806h
      61/       6 : 08 09               irq3:       dw    809h
      62/       8 : 08 0C               irq4:       dw    80Ch
      63/       A : 08 0F               irq5:       dw    80Fh
      64/       C :
      65/       C :                     ;-----------------------------------------------
------------------------------
      66/       C :                     ; Bootstrap MODULE
      67/       C :                     ; s. Kieser/Bankel Seite    228-229
      68/       C :                     ;-----------------------------------------------
------------------------------
      69/       C :
      70/       C :                             ; public init
      71/       C : 31 00               init:       srp    #0        ; Registerprointer
auf 00-Gruppe
      72/       E : 3C 0F                       ld    R3, #0Fh    ; Test,    ob P32 und
P35 gebrückt    sind
      73/      10 : FF                          nop
      74/      11 : 76 E3 04                    tm    R3, #4
      75/      14 : 3C FF                       ld    R3, #0FFh
      76/      16 : EB 05                       jr    NZ, init10
      77/      18 : 76 E3 04                    tm    R3, #4
      78/      1B : EB 20                       jr    NZ, test    ; wenn gebrückt, dann
weiter im    Test-Programm
      79/      1D :                     ;
      80/      1D : E6 F8 B6           init10:     ld    p01m, #10110110b ; Festlegung
für erweitertes Zeitverhalten
      81/      20 : E6 F7 08                    ld    p3m, #00001000b    ; Programmierung
Port 3
      82/      23 : 4C 08                       ld    R4, #8        ; Test,    ob sich
auf Adresse 0812h RAM befindet
      83/      25 : 5C 12                       ld    R5, #12h
      84/      27 : C2 64                       ldc    R6, @RR4
      85/      29 : 60 E6                       com    R6
      86/      2B : D2 64                       ldc    @RR4, R6
      87/      2D : C2 74                       ldc    R7, @RR4
      88/      2F : 60 E6                       com    R6
      89/      31 : D2 64                       ldc    @RR4, R6
      90/      33 : B2 67                       xor    R6, R7
```

```
      91/      35 : 31 F0                        srp    #0F0h
      92/      37 : ED E0 00                     jp    NZ, 0E000h    ; wenn RAM
lokalisiert ist, Sprung nach    0E000h
      93/      3A : 8D 08 12                     jp    812h        ; sonst    nach Adresse
0812h springen
      94/      3D :
      95/      3D :
      96/      3D :                    ;-----------------------------------------------
------------------------------
      97/      3D :                    ; Testhilfe zur Ausgabe    von alternierenden
      98/      3D :                    ; Impulsen am Port P1 und P0 mit
      99/      3D :                    ; Triggerimpuls an Leitung P35
     100/      3D :                    ;-----------------------------------------------
------------------------------
     101/      3D :
     102/      3D : E6 F8 04           test:       ld    p01m, #4    ; Port P1 und P0
auf Ausgabe
     103/      40 : E6 F1 C0                     ld    tmr, #0C0h    ; internen Takt
ausgeben
     104/      43 : 0C FF                        ld    R0, #0FFh
     105/      45 : 1C FF                        ld    R1, #0FFh
     106/      47 : CF                           rcf
     107/      48 : 56 E3 DF           test10:     and    R3, #0DFh    ; Triggerimpuls
an Leitung P35
     108/      4B : 46 E3 20                     or    R3, #20h
     109/      4E : 10 E1              test20:     rlc    R1
     110/      50 : 10 E0                        rlc    R0
     111/      52 : 76 E3 04                     tm    R3, #4        ; Abfrage für
tristate
     112/      55 : 6B 04                        jr    Z, tristate
     113/      57 : 7B F5                        jr    C, test20
     114/      59 : 8B ED                        jr    test10
     115/      5B : E6 F8 7F           tristate:   ld    p01m, #7Fh    ; Ports    0-1
mode
     116/      5E : 8B FE              halt:       jr    halt        ; Warteschleife
     117/      60 :
     118/      60 :                    ;-----------------------------------------------
------------------------------
     119/      60 :                    ; dma MODULE
     120/      60 :                    ; s. Kieser/Bankel Seite    235
     121/      60 :
     122/      60 :                    ; Serviceroutine für BUSREQ,
     123/      60 :                    ; mit BUSREQ an P32 und
     124/      60 :                    ; BUSACK ans P35, beide L-aktiv
     125/      60 :                    ;-----------------------------------------------
------------------------------
     126/      60 :
     127/      60 :                    busreq:                 ; Festlegung der
Kodierung für Tristate    für Port 0 und P1
     128/      60 : 46 7F 18                     or    reg_7F, #18h    ; P01M muß in
Register %7F verfügbar sein, da P01M nicht lesbar
     129/      63 : E4 7F F8                     ld    p01m, reg_7F    ; Ports    0-1 mode
     130/      66 : 56 03 DF                     and    p3, #0DFh    ; Ausgabe von BUSACK
     131/      69 : 76 03 04           busreq10:   tm    p3, #4        ; Port 3
     132/      6C : 6B FB                        jr    Z, busreq10    ; Ende abwarten
     133/      6E : 56 7F F7                     and    reg_7F, #0F7h    ; Festlegung der
Kodierung für AD0..AD7    an Port    0 und Port 1
     134/      71 : 46 03 20                     or    p3, #20h    ; BUSACK zurücknehmen
     135/      74 : E4 7F F8                     ld    p01m, reg_7F    ; wieder auf Bus
gehen
```

```
     136/      77 : FF                          nop            ; Ausführung (wegen
Pipeline) abwarten
     137/      78 : BF                          iret
     138/      79 :
     139/      79 :                       ; Beginn TINY-MP-BASIC
     140/      79 :                       ; Variablen A..Z liegen in Register 20h..53h
     141/      79 :                       ; Registernutzung RP = 10h
     142/      79 :                       ; RR0   Pointer aktuelles Zeichen in BASIC-
Programm
     143/      79 :                       ; RR2   Y, Rückgabewert
     144/      79 :                       ; RR4   X, Eingabewert
     145/      79 :                       ; R6    aktuelles Zeichen
     146/      79 :
     147/      79 :
     148/      79 :                       ; Arithmetik: Parameter1 X R4+R5,  Rückgabewert Y
R2+R3
     149/      79 :
     150/      79 :                       ;-----------------------------------------------
-----------------------------
     151/      79 :                       ; Y := X
     152/      79 :                       ;-----------------------------------------------
-----------------------------
     153/      79 :
     154/      79 : 28 E4                 p_let:      ld    R2, R4        ; Y := X
     155/      7B : 38 E5                         ld    R3, R5
     156/      7D : AF                            ret
     157/      7E :
     158/      7E :                       ;-----------------------------------------------
-----------------------------
     159/      7E :                       ; Y := Y + X
     160/      7E :                       ;-----------------------------------------------
-----------------------------
     161/      7E :
     162/      7E : 02 35                 p_plus:     add    R3, R5        ; Y := Y + X
     163/      80 : 12 24                         adc    R2, R4
     164/      82 : 4B 16                         jr    OV, p_abs3
     165/      84 : AF                            ret
     166/      85 :
     167/      85 :                       ;-----------------------------------------------
-----------------------------
     168/      85 :                       ; Y := Y - X
     169/      85 :                       ;-----------------------------------------------
-----------------------------
     170/      85 :
     171/      85 : 22 35                 p_minus:    sub    R3, R5        ; Y := Y - X
     172/      87 : 32 24                         sbc    R2, R4
     173/      89 : 4B 0F                         jr    OV, p_abs3
     174/      8B : AF                            ret
     175/      8C :
     176/      8C :                       ;-----------------------------------------------
-----------------------------
     177/      8C :                       ; interne Prozedur ABS
     178/      8C :                       ; ABS[parameter] absoluter Betrag
     179/      8C :                       ;-----------------------------------------------
-----------------------------
     180/      8C :                       ; Y := ABS (X)
     181/      8C : 76 E4 80              p_abs:      tm    R4, #80h    ; X Test Vorzeichen
Bit    7
     182/      8F : 6B E8                         jr    Z, p_let    ; Y := X
     183/      91 :                               ; sonst negieren
```

```
     184/      91 : B0 E2              p_abs1:     clr    R2        ; Y := 0
     185/      93 : B0 E3                      clr    R3
     186/      95 : 8B EE                      jr    p_minus        ; Y := Y - X
     187/      97 : 46 0F 80           p_abs2:     or    reg_0F, #80h    ; Bit7
     188/      9A : 46 0E 80           p_abs3:     or    reg_0E, #80h    ; Bit7
     189/      9D : AF                         ret
     190/      9E :
     191/      9E :
     192/      9E :                    ;-----------------------------------------------
-----------------------------
     193/      9E :                    ; UP zu DIV/MULT
     194/      9E :                    ;-----------------------------------------------
-----------------------------
     195/      9E :
     196/      9E :                    ; ? Vorzeichen behandeln
     197/      9E : 88 E2              sign:       ld    R8, R2
     198/      A0 : B2 84                      xor    R8, R4
     199/      A2 : 9C 02                      ld    R9, #2
     200/      A4 :                    ;
     201/      A4 : 68 E2              sign1:      ld    R6, R2
     202/      A6 : 78 E3                      ld    R7, R3
     203/      A8 : D6 00 8C                   call    p_abs        ; ABS[parameter]
absoluter Betrag
     204/      AB : 48 E6                      ld    R4, R6
     205/      AD : 58 E7                      ld    R5, R7
     206/      AF : 9A F3                      djnz    R9, sign1
     207/      B1 : 68 E2                      ld    R6, R2
     208/      B3 : 78 E3                      ld    R7, R3
     209/      B5 : B0 E2                      clr    R2
     210/      B7 : B0 E3                      clr    R3
     211/      B9 : AF                         ret
     212/      BA :
     213/      BA :                    ;-----------------------------------------------
-----------------------------
     214/      BA :                    ; Y := Y * X
     215/      BA :                    ;-----------------------------------------------
-----------------------------
     216/      BA :
     217/      BA : D6 00 9E           p_mult:     call    sign
     218/      BD : BC 0F                      ld    R11, #0Fh
     219/      BF : D0 E6              p_mult1:    sra    R6
     220/      C1 : C0 E7                      rrc    R7
     221/      C3 : FB 06                      jr    NC, p_mult2
     222/      C5 : 02 35                      add    R3, R5
     223/      C7 : 12 24                      adc    R2, R4
     224/      C9 : 4B CC                      jr    OV, p_abs2
     225/      CB : 10 E5              p_mult2:    rlc    R5
     226/      CD : 10 E4                      rlc    R4
     227/      CF : CB 04                      jr    NOV, p_mult3
     228/      D1 : 42 67                      or    R6, R7
     229/      D3 : EB C2                      jr    NZ, p_abs2
     230/      D5 : BA E8              p_mult3:    djnz    R11, p_mult1
     231/      D7 : 48 E2              p_mult4:    ld    R4, R2
     232/      D9 : 58 E3                      ld    R5, R3
     233/      DB : 10 E8                      rlc    R8
     234/      DD : 7B B2                      jr    C, p_abs1
     235/      DF : AF                         ret
     236/      E0 :
     237/      E0 :                    ;-----------------------------------------------
-----------------------------
```

```
     238/      E0 :                    ; Y := Y / X
     239/      E0 :                    ;-----------------------------------------------
-----------------------------
     240/      E0 :
     241/      E0 : D6 00 9E           p_div:      call    sign
     242/      E3 : 9C 10                      ld    R9, #10h
     243/      E5 : CF                         rcf
     244/      E6 : B0 EA                      clr    R10        ; RR10 = Divisionstrest
= 0
     245/      E8 : B0 EB                      clr    R11
     246/      EA : 10 E7              p_div1:     rlc    R7
     247/      EC : 10 E6                      rlc    R6
     248/      EE : 10 EB                      rlc    R11
     249/      F0 : 10 EA                      rlc    R10
     250/      F2 : 7B 0A                      jr    C, p_div2
     251/      F4 : A2 4A                      cp    R4, R10
     252/      F6 : BB 0B                      jr    UGT, p_div3
     253/      F8 : 7B 04                      jr    C, p_div2
     254/      FA : A2 5B                      cp    R5, R11
     255/      FC : BB 05                      jr    UGT, p_div3
     256/      FE : 22 B5              p_div2:     sub    R11, R5
     257/     100 : 32 A4                      sbc    R10, R4
     258/     102 : DF                         scf
     259/     103 : 9A E5              p_div3:     djnz    R9, p_div1
     260/     105 : 98 E4                      ld    R9, R4
     261/     107 : 42 95                      or    R9, R5
     262/     109 : 6B 0A                      jr    Z, p_div5
     263/     10B : 10 E7                      rlc    R7
     264/     10D : 10 E6                      rlc    R6
     265/     10F :                    ;
     266/     10F : 28 E6              p_div4:     ld    R2, R6
     267/     111 : 38 E7                      ld    R3, R7
     268/     113 : 8B C2                      jr    p_mult4
     269/     115 :                    ;
     270/     115 : D6 01 0F           p_div5:     call    p_div4
     271/     118 : 46 0E 40                   or    reg_0E, #40h    ; Bit6
     272/     11B : 46 0F 80           p_div6:     or    reg_0F, #80h    ; Bit7
     273/     11E : AF                         ret
     274/     11F :
     275/     11F :                    ;-----------------------------------------------
-----------------------------
     276/     11F :                    ; $-Operator
     277/     11F :                    ; Y := Y MOD X
     278/     11F :                    ;-----------------------------------------------
-----------------------------
     279/     11F :
     280/     11F : D6 00 E0           p_mod:      call    p_div
     281/     122 : 88 E2                      ld    R8, R2
     282/     124 : 28 EA                      ld    R2, R10        ; Divisonsrest
     283/     126 : 38 EB                      ld    R3, R11
     284/     128 : 8B AD                      jr    p_mult4
     285/     12A :
     286/     12A :                    ;-----------------------------------------------
-----------------------------
     287/     12A :                    ; $-Operator
     288/     12A :                    ; Y := Y OR X
     289/     12A :                    ;-----------------------------------------------
-----------------------------
     290/     12A :
     291/     12A : 42 24              p_or:       or    R2, R4
```

```
     292/     12C : 42 35                        or    R3, R5
     293/     12E : AF                           ret
     294/     12F :
     295/     12F :                    ;-----------------------------------------------
-----------------------------
     296/     12F :                    ; $-Operator
     297/     12F :                    ; Y := Y AND X
     298/     12F :                    ;-----------------------------------------------
-----------------------------
     299/     12F :
     300/     12F : 52 24              p_and:      and    R2, R4
     301/     131 : 52 35                         and    R3, R5
     302/     133 : AF                           ret
     303/     134 :
     304/     134 :                    ;-----------------------------------------------
-----------------------------
     305/     134 :                    ; $-Operator
     306/     134 :                    ; Y := Y XOR X
     307/     134 :                    ;-----------------------------------------------
-----------------------------
     308/     134 :
     309/     134 : B2 24              p_xor:      xor    R2, R4
     310/     136 : B2 35                         xor    R3, R5
     311/     138 : AF                           ret
     312/     139 :
     313/     139 :                    ;-----------------------------------------------
-----------------------------
     314/     139 :                    ; interne Prozedur NOT
     315/     139 :                    ;-----------------------------------------------
-----------------------------
     316/     139 :                    ; Y := NOT X
     317/     139 :                    ; NOT[parameter] bitweise logische Negation
     318/     139 :
     319/     139 : D6 00 79           p_not:      call    p_let        ; Y := X
     320/     13C : 60 E2                         com    R2        ; Y := NOT X
     321/     13E : 60 E3                         com    R3
     322/     140 : AF                           ret
     323/     141 :
     324/     141 :                    ;-----------------------------------------------
-----------------------------
     325/     141 :                    ; Vergleich RR2 und RR4 auf <,=,>: out: reg_0F
     326/     141 :                    ;-----------------------------------------------
-----------------------------
     327/     141 :
     328/     141 : 56 0F 8F           relcmp:     and    reg_0F, #8Fh    ; Bit 4..6 = 0
     329/     144 : A2 24                         cp    R2, R4
     330/     146 : 6B 0A                         jr    Z, relcmp3
     331/     148 : 7C 20                         ld    R7, #20h
     332/     14A : AB 02                         jr    GT, relcmp2    ; > Bitmaske 20h
     333/     14C : 7C 10              relcmp1:    ld    R7, #10h    ; < Bitmaske 10h
     334/     14E : 44 E7 0F           relcmp2:    or    reg_0F, R7
     335/     151 : AF                           ret
     336/     152 :                    ;
     337/     152 : 7C 40              relcmp3:    ld    R7, #40h
     338/     154 : A2 35                         cp    R3, R5
     339/     156 : 6B F6                         jr    Z, relcmp2    ; = Bitmaske 40h
     340/     158 : 7C 20                         ld    R7, #20h
     341/     15A : BB F2                         jr    UGT, relcmp2
     342/     15C : 8B EE                         jr    relcmp1
     343/     15E :
```

```
     344/     15E :                    ;-----------------------------------------------
-----------------------------
     345/     15E :                    ; UP zu c_PRINTHEX
     346/     15E :                    ; Konvertierung nach hexadezimal
     347/     15E :                    ;-----------------------------------------------
-----------------------------
     348/     15E :
     349/     15E : 88 E3              tohex:      ld    R8, R3        ; Byte->Nibble
     350/     160 : 78 E3                          ld    R7, R3        ; R2/R3 in R5..R8
kopieren
     351/     162 : 68 E2                          ld    R6, R2        ;
     352/     164 : 58 E2                          ld    R5, R2        ;
     353/     166 : F0 E5                          swap    R5        ; Hi Nibble in untere
Tetrade
     354/     168 : F0 E7                          swap    R7        ; Hi Nibble in untere
Tetrade
     355/     16A : 4C 25                          ld    R4, #'%'
     356/     16C :                                ;
     357/     16C : AC 04                          ld    R10, #4         ; 4 Stellen
     358/     16E : BC 15              tohex1:     ld    R11, #15h     ; Buffer
     359/     170 : 57 EB 0F           tohex2:     and    @R11, #0Fh
     360/     173 : 07 EB 30                       add    @R11, #30h    ; '0'
     361/     176 : A7 EB 3A                       cp    @R11, #3Ah     ; '9'+1
     362/     179 : 7B 03                          jr    C, tohex3     ; wenn größer '9',
dann + 7
     363/     17B : 07 EB 07                       add    @R11, #7     ; ergibt 'A'..'F'
     364/     17E : BE                 tohex3:     inc    R11         ; nächste
Ausgabepos
     365/     17F : AA EF                          djnz    R10, tohex2    ; nächste Stelle
     366/     181 : AF                             ret
     367/     182 :
     368/     182 :                    ;-----------------------------------------------
-----------------------------
     369/     182 :                    ; UP zu c_PRINT
     370/     182 :                    ; Konvertierung nach dezimal
     371/     182 :                    ;-----------------------------------------------
-----------------------------
     372/     182 :
     373/     182 : 48 E2              todez:      ld    R4, R2
     374/     184 : 58 E3                          ld    R5, R3
     375/     186 : D6 00 8C                       call    p_abs         ; absoluter Betrag
     376/     189 : 10 E4                          rlc    R4
     377/     18B : 4C 20                          ld    R4, #' '     ; bei pos. Zahl
Leerzeichen
     378/     18D : FB 02                          jr    NC, todez1
     379/     18F : 4C 2D                          ld    R4, #'-'     ; bei neg. Zahl '-'
     380/     191 : BC 06              todez1:     ld    R11, #6         ; 6 Stellen
     381/     193 : AC 15                          ld    R10, #15h     ; Buffer
     382/     195 : B1 EA              todez2:     clr    @R10         ; leeren
     383/     197 : AE                             inc    R10
     384/     198 : BA FB                          djnz    R11, todez2
     385/     19A :                                ;hex-> bcd
     386/     19A : BC 0F                          ld    R11, #0Fh
     387/     19C : D0 E2              todez3:     sra    R2
     388/     19E : C0 E3                          rrc    R3
     389/     1A0 : FB 0C                          jr    NC, todez4
     390/     1A2 : 02 7A                          add    R7, R10
     391/     1A4 : 40 E7                          da    R7
     392/     1A6 : 12 69                          adc    R6, R9
     393/     1A8 : 40 E6                          da    R6
```

```
     394/     1AA : 12 58                        adc    R5, R8
     395/     1AC : 40 E5                        da    R5
     396/     1AE : 02 AA                todez4:     add    R10, R10
     397/     1B0 : 40 EA                        da    R10
     398/     1B2 : 12 99                        adc    R9, R9
     399/     1B4 : 40 E9                        da    R9
     400/     1B6 : 12 88                        adc    R8, R8
     401/     1B8 : 40 E8                        da    R8
     402/     1BA : BA E0                        djnz    R11, todez3
     403/     1BC : 88 E7                        ld    R8, R7
     404/     1BE : 98 E7                        ld    R9, R7
     405/     1C0 : 78 E6                        ld    R7, R6
     406/     1C2 : F0 E6                        swap    R6
     407/     1C4 : F0 E8                        swap    R8
     408/     1C6 :                              ;bcd->ascii
     409/     1C6 : AC 05                        ld    R10, #5        ; 5 Stellen
     410/     1C8 : 8B A4                        jr    tohex1        ; nach ASCII wandeln
     411/     1CA :
     412/     1CA :                      ;-----------------------------------------------
-----------------------------
     413/     1CA :                      ; Ziffer in Result einschieben (i.e. * 16 +
stelle)
     414/     1CA :                      ; UP zu number
     415/     1CA :                      ;-----------------------------------------------
-----------------------------
     416/     1CA :
     417/     1CA : F1 ED                rotate:     swap    @R13        ; Ziffer in
oberes Nibble
     418/     1CC : EC 04                        ld    R14, #4        ; alles 4 Stellen
nach links
     419/     1CE : 11 ED                rotate1:    rlc    @R13        ; d.h. * 16
     420/     1D0 : 10 E5                        rlc    R5
     421/     1D2 : 10 E4                        rlc    R4
     422/     1D4 : 10 E3                        rlc    R3
     423/     1D6 : EA F6                        djnz    R14, rotate1
     424/     1D8 : AF                           ret
     425/     1D9 :
     426/     1D9 :                      ;-----------------------------------------------
-----------------------------
     427/     1D9 :                      ; Konvertierung ASCII->Zahl (hex, o. dez signed)
     428/     1D9 :                      ; in: String in Konvertierungspuffer ab 16h (R6
ff) ..
     429/     1D9 :                      ; ret: RR4 = Wert
     430/     1D9 :                      ;-----------------------------------------------
-----------------------------
     431/     1D9 :
     432/     1D9 : DC 16                number:     ld    R13, #16h    ;
Konvertierungspuffer
     433/     1DB : B0 E4                        clr    R4        ; Startwert 0
     434/     1DD : B0 E5                        clr    R5
     435/     1DF : A6 E6 25                     cp    R6, #'%'    ; Präfix Hexzahl
     436/     1E2 : EB 27                        jr    NZ, number5
     437/     1E4 :
     438/     1E4 :                      ; Hex-Zahl
     439/     1E4 : CC 05                        ld    R12, #5        ; max. 5 Stellen
     440/     1E6 : DE                   number1:    inc    R13
     441/     1E7 : 27 ED 30                     sub    @R13, #'0'
     442/     1EA : 7B 19                        jr    C, number4
     443/     1EC : A7 ED 0A                     cp    @R13, #0Ah    ; 9+1
     444/     1EF : 7B 0D                        jr    C, number2
```

```
     445/     1F1 : 27 ED 11                    sub    @R13, #11h    ; A..F -> 0..5
     446/     1F4 : 7B 0F                       jr    C, number4
     447/     1F6 : 07 ED 0A                    add    @R13, #0Ah    ; +10
     448/     1F9 : A7 ED 10                    cp    @R13, #10h
     449/     1FC : FB 07                       jr    NC, number4
     450/     1FE : D6 01 CA           number2:    call    rotate        ; * 16 + Stelle
     451/     201 : CA E3                       djnz    R12, number1
     452/     203 : CF                 number3:    rcf
     453/     204 : AF                          ret
     454/     205 :                    ;
     455/     205 : A6 EC 05           number4:    cp    R12, #5
     456/     208 : 6B F9                       jr    Z, number3
     457/     20A : AF                          ret
     458/     20B :
     459/     20B :                    ; Dezimalzahl
     460/     20B : CC 06              number5:    ld    R12, #6        ; max. 6 Stellen
     461/     20D : B0 E2                       clr    R2
     462/     20F : B0 E3                       clr    R3
     463/     211 : A6 E6 2D                    cp    R6, #'-'    ; Vorzeichen
     464/     214 : EB 02                       jr    NZ, number7
     465/     216 : 2E                          inc    R2
     466/     217 : DE                 number6:    inc    R13
     467/     218 : 27 ED 30           number7:    sub    @R13, #'0'
     468/     21B : 7B 0C                       jr    C, number9    ; wenn keine Ziffer
mehr
     469/     21D : A7 ED 0A                    cp    @R13, #0Ah    ; 9+1
     470/     220 : FB 07                       jr    NC, number9    ; wenn keine Ziffer
mehr
     471/     222 : D6 01 CA                    call    rotate        ; sonst zu BCD
konvertieren
     472/     225 : CA F0                       djnz    R12, number6
     473/     227 : CF                 number8:    rcf
     474/     228 : AF                          ret
     475/     229 :
     476/     229 :                    ; BCD R3R4R5 -> hex RR4
     477/     229 : A6 EC 06           number9:    cp    R12, #6
     478/     22C : 6B F9                       jr    Z, number8
     479/     22E : A6 E3 04                    cp    R3, #4
     480/     231 : FB F4                       jr    NC, number8
     481/     233 : B0 E6                       clr    R6
     482/     235 : B0 E7                       clr    R7
     483/     237 : BC 13                       ld    R11, #13h
     484/     239 : 8C 27                       ld    R8, #27h    ; 2710h = 10000
     485/     23B : 9C 10                       ld    R9, #10h
     486/     23D : D6 02 71                    call    number11
     487/     240 : BE                          inc    R11
     488/     241 : 8C 03                       ld    R8, #3        ; 3E8h = 1000
     489/     243 : 9C E8                       ld    R9, #0E8h
     490/     245 : D6 02 6F                    call    number10
     491/     248 : 8C 00                       ld    R8, #0        ; 064h = 100
     492/     24A : 9C 64                       ld    R9, #64h
     493/     24C : D6 02 6F                    call    number10
     494/     24F : BE                          inc    R11
     495/     250 : 9C 0A                       ld    R9, #0Ah    ; 10
     496/     252 : D6 02 6F                    call    number10
     497/     255 : 9C 01                       ld    R9, #1        ; 1
     498/     257 : D6 02 6F                    call    number10
     499/     25A : 48 E6                       ld    R4, R6
     500/     25C : 58 E7                       ld    R5, R7
     501/     25E : 10 E6                       rlc    R6
```

```
     502/     260 : 7B C5                        jr    C, number8
     503/     262 : C0 E2                        rrc    R2
     504/     264 : FB 18                        jr    NC, number13
     505/     266 : D6 00 91                     call    p_abs1
     506/     269 : 48 E2                        ld    R4, R2
     507/     26B : 58 E3                        ld    R5, R3
     508/     26D : 8B 0F                        jr    number13
     509/     26F :
     510/     26F :                         ;
     511/     26F : F1 EB                   number10:   swap    @R11
     512/     271 : E3 AB                   number11:   ld    R10, @R11    ; Stelle n
     513/     273 : 56 EA 0F                         and    R10, #0Fh
     514/     276 : 6B 06                            jr    Z, number13
     515/     278 : 02 79                   number12:   add    R7, R9        ; RR8 =
Stelligkeit
     516/     27A : 12 68                            adc    R6, R8
     517/     27C : AA FA                            djnz    R10, number12    ; n mal Addieren
     518/     27E : DF                      number13:   scf
     519/     27F : AF                               ret
     520/     280 :
     521/     280 :                         ;-----------------------------------------------
------------------------------
     522/     280 :                         ; bei Prozeduraufruf PROC [Y1,Y2,Y3,..,Ym] =
prozedurname [X1,X2,X3,..,Xn] gilt:
     523/     280 :                         ;   SP            returnadr.
     524/     280 :                         ;   SP+2 .. SP+2n-2    n-1     Parameter, wenn
n>1
     525/     280 :                         ;               letzter Parameter aus Liste ist
in RR4
     526/     280 :                         ;   SP+2n            (intern f. interpreter)
     527/     280 :                         ;   SP+2n+2    .. SP+2m+2n    Platz für m-1
Ergebnisvariable, wenn m>1
     528/     280 :                         ;               letzter Parameter aus Liste ist
in RR2 zu übergeben
     529/     280 :                         ;-----------------------------------------------
------------------------------
     530/     280 :
     531/     280 :                         ;-----------------------------------------------
------------------------------
     532/     280 :                         ; Ret.Adr. und ersten Parameter vom Stack nach
R6/R7 holen
     533/     280 :                         ;-----------------------------------------------
------------------------------
     534/     280 :
     535/     280 : 50 E8                   para1:      pop    R8        ; RR8
Rückkehradresse dieser Routine
     536/     282 : 50 E9                            pop    R9
     537/     284 : 50 E2                            pop    R2        ; RR2 Rückkehradresse
zum Interpreter
     538/     286 : 50 E3                            pop    R3
     539/     288 : 50 E6                            pop    R6        ; RR6 Adresse von X1
     540/     28A : 50 E7                            pop    R7
     541/     28C : 30 E8                            jp    @RR8        ; RET
     542/     28E :
     543/     28E :                         ;-----------------------------------------------
------------------------------
     544/     28E :                         ; interne Prozedur
     545/     28E :                         ; SETRR [register,wert]    Doppelregister setzen
     546/     28E :                         ;-----------------------------------------------
------------------------------
```

```
     547/     28E :
     548/     28E : D6 02 80            p_setrr:    call    para1        ; Parameter
register nach R6/R7
     549/     291 : F3 74                       ld    @R7, R4        ; register (hi) :=
wert
     550/     293 : 7E                          inc    R7
     551/     294 : 8B 03                       jr    p_setr1
     552/     296 :
     553/     296 :                     ;-----------------------------------------------
----------------------------
     554/     296 :                     ; interne Prozedur
     555/     296 :                     ; SETR [register,wert] Register setzen
     556/     296 :                     ;-----------------------------------------------
----------------------------
     557/     296 :
     558/     296 : D6 02 80            p_setr:     call    para1        ; Parameter
register nach R6/R7
     559/     299 : F3 75               p_setr1:    ld    @R7, R5        ; register (lo)
:= wert
     560/     29B : 30 E2                       jp    @RR2        ; zurück zum
Interpreter
     561/     29D :
     562/     29D :                     ;-----------------------------------------------
----------------------------
     563/     29D :                     ; interne Prozedur
     564/     29D :                     ; SETEW [adresse,wert] externes    Wort setzen
     565/     29D :                     ;-----------------------------------------------
----------------------------
     566/     29D :
     567/     29D : D6 02 80            p_setew:    call    para1        ; Parameter
adresse nach R6/R7
     568/     2A0 : 92 46                       lde    @RR6, R4
     569/     2A2 : A0 E6                       incw    RR6
     570/     2A4 : 8B 03                       jr    p_seteb1
     571/     2A6 :
     572/     2A6 :                     ;-----------------------------------------------
----------------------------
     573/     2A6 :                     ; interne Prozedur
     574/     2A6 :                     ; SETEB [adresse,wert] externes    Byte setzen
     575/     2A6 :                     ;-----------------------------------------------
----------------------------
     576/     2A6 :
     577/     2A6 : D6 02 80            p_seteb:    call    para1        ; Parameter
adresse nach R6/R7
     578/     2A9 : 92 56               p_seteb1:   lde    @RR6, R5
     579/     2AB : 30 E2                       jp    @RR2
     580/     2AD :
     581/     2AD :                     ;-----------------------------------------------
----------------------------
     582/     2AD :                     ; interne Prozedur
     583/     2AD :                     ; GETRR [register] liefert Inhalt des
Doppelregisters
     584/     2AD :                     ;-----------------------------------------------
----------------------------
     585/     2AD :
     586/     2AD : E3 25               p_getrr:    ld    R2, @R5
     587/     2AF : 5E                          inc    R5
     588/     2B0 : 0D                          db    0Dh         ; JP FALSE mit nächstem
Befehl
     589/     2B1 :
```

```
     590/     2B1 :                    ;-----------------------------------------------
-----------------------------
     591/     2B1 :                    ; interne Prozedur
     592/     2B1 :                    ; GETR [register] liefert Inhalt des Registers
     593/     2B1 :                    ;-----------------------------------------------
-----------------------------
     594/     2B1 :
     595/     2B1 : B0 E2              p_getr:     clr    R2        ; Hi = 0
     596/     2B3 : E3 35                         ld    R3, @R5
     597/     2B5 : AF                            ret
     598/     2B6 :
     599/     2B6 :                    ;-----------------------------------------------
-----------------------------
     600/     2B6 :                    ; interne Prozedur
     601/     2B6 :                    ; GETEW [register] holt    Wortwert aus externem
Speicher
     602/     2B6 :                    ;-----------------------------------------------
-----------------------------
     603/     2B6 :
     604/     2B6 : 82 24              p_getew:    lde    R2, @RR4
     605/     2B8 : A0 E4                         incw    RR4
     606/     2BA : 0D                            db    0Dh         ; JP FALSE mit nächstem
Befehl
     607/     2BB :
     608/     2BB :                    ;-----------------------------------------------
-----------------------------
     609/     2BB :                    ; interne Prozedur
     610/     2BB :                    ; GETEB [register] holt    Bytewert aus externem
Speicher
     611/     2BB :                    ;-----------------------------------------------
-----------------------------
     612/     2BB :
     613/     2BB : B0 E2              p_geteb:    clr    R2
     614/     2BD : 82 34                         lde    R3, @RR4
     615/     2BF : AF                            ret
     616/     2C0 :
     617/     2C0 :                    ;-----------------------------------------------
-----------------------------
     618/     2C0 :                    ; angezeigtes Zeichen löschen
     619/     2C0 :                    ;-----------------------------------------------
-----------------------------
     620/     2C0 :                    ; UP zu p_input6
     621/     2C0 :
     622/     2C0 : D6 02 C8           delc:       call    delc1        ; ein Zeichen
zurück
     623/     2C3 : 5C 20                         ld    R5, #' '    ; mit Leerzeichen
überschreiben
     624/     2C5 : D6 08 18                      call    putch        ; PUT_CHAR
     625/     2C8 : 5C 08              delc1:      ld    R5, #8         ; BS
     626/     2CA : 8D 08 18                      jp    putch        ; PUT_CHAR
     627/     2CD :
     628/     2CD :                    ;-----------------------------------------------
-----------------------------
     629/     2CD :                    ; interne Prozedur
     630/     2CD :                    ; ; RL[x] x links rotieren
     631/     2CD :                    ;-----------------------------------------------
-----------------------------
     632/     2CD :
     633/     2CD : CF                 p_rl:       rcf            ; Cy = 0
     634/     2CE : 10 E5                         rlc    R5
```

```
     635/     2D0 : 10 E4                        rlc    R4
     636/     2D2 : 16 E5 00                     adc    R5, #0
     637/     2D5 : 8D 00 79           p_rl1:      jp    p_let        ; Y := X
     638/     2D8 :
     639/     2D8 :                    ;-----------------------------------------------
-----------------------------
     640/     2D8 :                    ; interne Prozedur
     641/     2D8 :                    ; RR[x] x rechts rotieren
     642/     2D8 :                    ;-----------------------------------------------
-----------------------------
     643/     2D8 :
     644/     2D8 : CF                 p_rr:       rcf            ; Cy = 0
     645/     2D9 : C0 E4                         rrc    R4
     646/     2DB : C0 E5                         rrc    R5
     647/     2DD : FB F6                         jr    NC, p_rl1
     648/     2DF : 46 E4 80                      or    R4, #80h    ; Vorzeichen löschen
     649/     2E2 : 8B F1                         jr    p_rl1
     650/     2E4 :
     651/     2E4 :                    ;-----------------------------------------------
-----------------------------
     652/     2E4 :                    ; interne Prozedur
     653/     2E4 :                    ; INPUT Zahleneingabe vom Terminal
     654/     2E4 :                    ;-----------------------------------------------
-----------------------------
     655/     2E4 :
     656/     2E4 : 5C 3F              p_input:    ld    R5, #'?'
     657/     2E6 : D6 08 18                      call    putch        ; PUT_CHAR
     658/     2E9 : FC 15              p_input1:   ld    R15, #15h    ;
Konvertierungspuffer
     659/     2EB : FE                 p_input2:   inc    R15
     660/     2EC : A6 EF 1F                      cp    R15, #1Fh
     661/     2EF : 6B 20                         jr    Z, p_input4
     662/     2F1 : D6 08 15           p_input3:   call    getch        ; GET_CHAR
     663/     2F4 : F3 F3                         ld    @R15, R3
     664/     2F6 : A6 E3 0D                      cp    R3, #0Dh    ; CR
     665/     2F9 : 6B 16                         jr    Z, p_input4
     666/     2FB : A6 E3 08                      cp    R3, #8        ; BS
     667/     2FE : EB 19                         jr    NZ, p_input5
     668/     300 : 5C 20                         ld    R5, #' '
     669/     302 : D6 08 18                      call    putch        ; PUT_CHAR
     670/     305 : 00 EF                         dec    R15
     671/     307 : A6 EF 15                      cp    R15, #15h
     672/     30A : 6B DF                         jr    Z, p_input2
     673/     30C : D6 02 C8                      call    delc1
     674/     30F : 8B E0                         jr    p_input3
     675/     311 : D6 01 D9           p_input4:   call    number
     676/     314 : FB CE                         jr    NC, p_input    ; INPUT
Zahleneingabe vom Terminal
     677/     316 : 8D 00 79                      jp    p_let        ; Y := X
     678/     319 : 3E                 p_input5:   inc    R3
     679/     31A : DB CF                         jr    PL, p_input2
     680/     31C : A6 EF 16           p_input6:   cp    R15, #16h
     681/     31F : 6B C8                         jr    Z, p_input1
     682/     321 : D6 02 C0                      call    delc        ; angezeigtes Zeichen
löschen
     683/     324 : 00 EF                         dec    R15
     684/     326 : 8B F4                         jr    p_input6
     685/     328 :
     686/     328 :                    ;-----------------------------------------------
-----------------------------
```

```
     687/     328 :                    ; Liste der internen Prozeduren
     688/     328 :                    ;-----------------------------------------------
----------------------------
     689/     328 :
     690/     328 : 03 4E 4F 54        tab_prc:    db    3,"NOT"
     691/     32C : 01 39                          dw    p_not        ; NOT[parameter]
bitweise logische Negation
     692/     32E : 03 41 42 53                    db    3,"ABS"
     693/     332 : 00 8C                          dw    p_abs        ; ABS[parameter]
absoluter Betrag
     694/     334 : 05 53 45 54 52 52              db    5,"SETRR"
     695/     33A : 02 8E                          dw    p_setrr        ; SETRR
[register,wert]    Doppelregister setzen
     696/     33C : 04 53 45 54 52                 db    4,"SETR"
     697/     341 : 02 96                          dw    p_setr        ; SETR
[register,wert] Register    setzen
     698/     343 : 05 53 45 54 45 57              db    5,"SETEW"
     699/     349 : 02 9D                          dw    p_setew        ; SETEW
[adresse,wert] externes    Wort setzen
     700/     34B : 05 53 45 54 45 42              db    5,"SETEB"
     701/     351 : 02 A6                          dw    p_seteb        ; SETEB
[adresse,wert] externes    Byte setzen
     702/     353 : 05 47 45 54 52 52              db    5,"GETRR"
     703/     359 : 02 AD                          dw    p_getrr        ; GETRR
[register] liefert Inhalt des Doppelregisters
     704/     35B : 04 47 45 54 52                 db    4,"GETR"
     705/     360 : 02 B1                          dw    p_getr        ; GETR [register]
liefert Inhalt des Registers
     706/     362 : 05 47 45 54 45 57              db    5,"GETEW"
     707/     368 : 02 B6                          dw    p_getew        ; GETEW
[register] holt    Wortwert aus externem Speicher
     708/     36A : 05 47 45 54 45 42              db    5,"GETEB"
     709/     370 : 02 BB                          dw    p_geteb        ; GETEB
[register] holt    Bytewert aus externem Speicher
     710/     372 : 02 52 4C                       db    2,"RL"
     711/     375 : 02 CD                          dw    p_rl        ; RL[x]    x links
rotieren
     712/     377 : 02 52 52                       db    2,"RR"
     713/     37A : 02 D8                          dw    p_rr        ; RR[x]    x rechts
rotieren
     714/     37C : 05 49 4E 50 55 54              db    5,"INPUT"
     715/     382 : 02 E4                          dw    p_input        ; INPUT
Zahleneingabe vom Terminal
     716/     384 : 03 47 54 43                    db    3,"GTC"
     717/     388 : 08 15                          dw    getch        ; Get Char extern !
     718/     38A : 03 50 54 43                    db    3,"PTC"
     719/     38E : 08 18                          dw    putch        ; Put Char extern !
     720/     390 : FF                             db    0FFh        ; Listenende
     721/     391 :
     722/     391 :                    ;-----------------------------------------------
----------------------------
     723/     391 :                    ;nächstes Zeichen aus Programmtext holen
     724/     391 :                    ;in: R7 Vergleichszeichen ret: Z=1 Zeichen
gefunden
     725/     391 :                    ;-----------------------------------------------
----------------------------
     726/     391 : FC 16              next_char:  ld    R15, #16h    ; R15 vorbelegen
für buffer ?
     727/     393 : C2 60                          ldc    R6, @RR0    ; nächstes Zeichen aus
Basic-Programmcode
```

```
     728/     395 : A0 E0                        incw    RR0        ; Pointer
weiterstellen
     729/     397 : A2 67                        cp    R6, R7        ; Vergleich mit R7
     730/     399 : AF                           ret
     731/     39A :
     732/     39A :                    ;-----------------------------------------------
-----------------------------
     733/     39A :                    ; Operator in Operator-Tabelle suchen
     734/     39A :                    ; ret: R7 = op, RR8 = proc.addr.
     735/     39A :                    ;-----------------------------------------------
-----------------------------
     736/     39A : AC 03              oper:       ld    R10, #HI(tab_op)
     737/     39C : BC C1                         ld    R11, #LO(tab_op)
     738/     39E : DC 03                         ld    R13, #3        ; 4 arithm.
Operatoren
     739/     3A0 : D6 03 AF                      call    oper1        ; suche
     740/     3A3 : 6B 1B                         jr    Z, oper3    ; Operator nicht
gefunden
     741/     3A5 : A6 E6 24                      cp    R6, #'$'    ; Präfix logischer
Operator
     742/     3A8 : EB 16                         jr    NZ, oper3    ; Präfix stimmt nicht
     743/     3AA : D6 03 91                      call    next_char
     744/     3AD : DC 03                         ld    R13, #3        ; 4 logische
Operatoren
     745/     3AF :
     746/     3AF :                    oper1:      ; tab_op durchsuchen
     747/     3AF : D6 03 B6                      call    oper2
     748/     3B2 : 6B 0C                         jr    Z, oper3    ; wenn Operator
gefunden
     749/     3B4 : DA F9                         djnz    R13, oper1
     750/     3B6 :                               ;
     751/     3B6 : CC 17              oper2:      ld    R12, #17h    ; 3 Byte nach 17h
kopieren
     752/     3B8 : C3 CA                         ldci    @R12, @RR10    ; R7 := Op
     753/     3BA : C3 CA                         ldci    @R12, @RR10    ; RR8 :=
Fkt.Adresse
     754/     3BC : C3 CA                         ldci    @R12, @RR10
     755/     3BE : A2 67                         cp    R6, R7        ; Operator mit
Suchwert vergleichen
     756/     3C0 : AF                 oper3:      ret
     757/     3C1 :
     758/     3C1 :                    ;-----------------------------------------------
-----------------------------
     759/     3C1 :                    ; Tabelle Arithmetik-/Logik-Operatoren, s. oper
     760/     3C1 :                    ;-----------------------------------------------
-----------------------------
     761/     3C1 :
     762/     3C1 :                    tab_op:
     763/     3C1 : 2B                            db    '+'
     764/     3C2 : 00 7E                         dw    p_plus        ; Y := Y + X
     765/     3C4 : 2D                            db    '-'
     766/     3C5 : 00 85                         dw    p_minus        ; Y := Y - X
     767/     3C7 : 2A                            db    '*'
     768/     3C8 : 00 BA                         dw    p_mult        ; Y := Y * X
     769/     3CA : 2F                            db    '/'
     770/     3CB : 00 E0                         dw    p_div        ; Y := Y / X
     771/     3CD :                    ; $-Operatoren (mit vorausgehendem '$')
     772/     3CD : 41                            db    'A'
     773/     3CE : 01 2F                         dw    p_and        ; Y := Y AND X
     774/     3D0 : 4F                            db    'O'
```

```
     775/     3D1 : 01 2A                        dw    p_or        ; Y := Y OR X
     776/     3D3 : 58                           db    'X'
     777/     3D4 : 01 34                        dw    p_xor        ; Y := Y XOR X
     778/     3D6 : 4D                           db    'M'
     779/     3D7 : 01 1F                        dw    p_mod        ; Y := Y MOD X
     780/     3D9 :
     781/     3D9 :
     782/     3D9 :                    ;-----------------------------------------------
-----------------------------
     783/     3D9 :                    ; Zeichenklassentests
     784/     3D9 :                    ;-----------------------------------------------
-----------------------------
     785/     3D9 :
     786/     3D9 :                    ;in: @R15 out: cy=1 bei Buchstaben A..Z
     787/     3D9 : A7 EF 41           is_letter:  cp    @R15, #41h    ; 'A'
     788/     3DC : 7B 12                         jr    C, is_notf
     789/     3DE : A7 EF 5B                      cp    @R15, #5Bh    ; 'Z'+1
     790/     3E1 : AF                            ret
     791/     3E2 :
     792/     3E2 :                    ;in: @R15 out: cy=1 bei Buchstaben A..Z oder Zahl
0..9
     793/     3E2 : D6 03 D9           is_char:    call    is_letter
     794/     3E5 : 7B 08                         jr    C, is_digit_ret
     795/     3E7 :
     796/     3E7 :                    ;in: @R15 out: cy=1 bei Zahl 0..9
     797/     3E7 : A7 EF 30           is_digit:   cp    @R15, #30h    ; '0'
     798/     3EA : 7B 04                         jr    C, is_notf
     799/     3EC : A7 EF 3A                      cp    @R15, #3Ah    ; '9'+1
     800/     3EF : AF                 is_digit_ret:   ret
     801/     3F0 :
     802/     3F0 :                    ;
     803/     3F0 : CF                 is_notf:    rcf            ; Cy=0
     804/     3F1 : AF                            ret
     805/     3F2 :
     806/     3F2 :                    ;in: @R15 out: cy=1 bei Zahl 0..F
     807/     3F2 : D6 03 E7           is_hexdigit:    call    is_digit
     808/     3F5 : 7B F8                         jr    C, is_digit_ret
     809/     3F7 : A7 EF 41                      cp    @R15, #41h    ; 'A'
     810/     3FA : 7B F4                         jr    C, is_notf
     811/     3FC : A7 EF 47                      cp    @R15, #47h    ; 'F'+1
     812/     3FF : AF                            ret
     813/     400 :
     814/     400 :                    ;-----------------------------------------------
-----------------------------
     815/     400 :                    ; Prozedurnamen in Proc-Tabelle suchen
     816/     400 :                    ; ret: Cy=gefunden, RR8=Proc-Adresse
     817/     400 :                    ;-----------------------------------------------
-----------------------------
     818/     400 :                    ; Aubau Tabelle:  je 1 Byte Namenslänge
     819/     400 :                    ;             Procedurenamen
     820/     400 :                    ;             2 Byte Adresse
     821/     400 :                    ;          Ende mit Namenslänge = FFh
     822/     400 :                    ;-----------------------------------------------
-----------------------------
     823/     400 :
     824/     400 :                    ; zu func1
     825/     400 :                    find_name:  ; Suche in interner Tabelle
     826/     400 : 2C 03                         ld    R2, #HI(tab_prc)
     827/     402 : 3C 28                         ld    R3, #LO(tab_prc)
     828/     404 : D6 04 12                      call    find_name1
```

```
     829/     407 : 7B 15                        jr    C, find_name4    ; Ende wenn
gefunden
     830/     409 :                              ; Suche in externer Tabelle
     831/     409 : B0 E2                        clr    R2
     832/     40B : 44 08 E2                     or    R2, reg_08    ; HI ext. Proc-Table
     833/     40E : 6B 0D                        jr    Z, find_name3    ; wenn 0 (= nicht
vorhanden)
     834/     410 : 38 09                        ld    R3, reg_09    ; LO ext. Proc-Table
     835/     412 :                      ;
     836/     412 : 48 E0                find_name1: ld    R4, R0        ; akt. Position
Basicprogramm
     837/     414 : 58 E1                        ld    R5, R1        ; d.h. Procedur-Name
     838/     416 : C2 82                find_name2: ldc    R8, @RR2    ; Länge
Procedurname
     839/     418 : A6 E8 FF                     cp    R8, #0FFh    ; tap_prc-Ende ?
     840/     41B : EB 02                        jr    NZ, comp_name
     841/     41D : CF                   find_name3: rcf
     842/     41E : AF                   find_name4: ret
     843/     41F :
     844/     41F :                      ; Namensvergleich
     845/     41F : A0 E2                comp_name:  incw    RR2        ; auf nächstes
Zeichen Procedur-Name
     846/     421 : C2 72                        ldc    R7, @RR2    ; Zeichen nach R7 f.
Vergleich
     847/     423 : D6 03 91                     call    next_char    ; gesuchte Procedur
nächstes Zeichen
     848/     426 : EB 14                        jr    NZ, comp_name2    ; passt nicht ->
nächsten Prozedurnamen vergleichen
     849/     428 : 8A F5                        djnz    R8, comp_name    ; weiter bis
alle Zeichen verglichen
     850/     42A :                              ; Procedur-Name in Tabelle gefunden
     851/     42A : C2 60                        ldc    R6, @RR0    ; Zeichen an akt.
Position Basicprogramm
     852/     42C : D6 03 E2                     call    is_char        ; ist der gesuchte
Name länger?
     853/     42F : 7B 0A                        jr    C, comp_name1    ; dann
weitersuchen
     854/     431 : A0 E2                        incw    RR2
     855/     433 : C2 82                        ldc    R8, @RR2       ; sonst
Proceduradresse nach RR8
     856/     435 : A0 E2                        incw    RR2
     857/     437 : C2 92                        ldc    R9, @RR2
     858/     439 : DF                           scf            ; Cy=gefunden
     859/     43A : AF                           ret
     860/     43B :                      ; nächsten Prozedurnamen vergleichen
     861/     43B : 8E                   comp_name1: inc    R8        ; Restlänge v. Namen
+ 1
     862/     43C : 8E                   comp_name2: inc    R8        ; + 2 Byte Adresse
     863/     43D : 8E                           inc    R8
     864/     43E : 02 38                        add    R3, R8        ; zu Pos in Proc-
Table addieren
     865/     440 : 16 E2 00                     adc    R2, #0
     866/     443 : 08 E4                        ld    R0, R4        ; akt. Position
Basicprogramm
     867/     445 : 18 E5                        ld    R1, R5        ; rücksetzen
     868/     447 : 8B CD                        jr    find_name2    ; weitersuchen
     869/     449 :
     870/     449 :                      ;-----------------------------------------------
------------------------------
     871/     449 :                      ; Prozedur/Fkt.namen suchen, Eingabeparameter
```

```
ablegen, Prozedur ausführen
     872/     449 :                    ;-----------------------------------------------
----------------------------
     873/     449 :
     874/     449 : 80 E0              func:       decw    RR0
     875/     44B :
     876/     44B :                    ; zu c_PROC3
     877/     44B : D6 04 00           func1:      call    find_name    ; Prozedurnamen
suchen
     878/     44E : C2 A0                         ldc    R10, @RR0    ; Zeichen an akt.
Position Basicprogramm
     879/     450 : A6 EA 5B                      cp    R10, #'['    ; folgen
Eingabeparameter?
     880/     453 : EB 1A                         jr    NZ, func3    ; nein
     881/     455 : A0 E0                         incw    RR0
     882/     457 :                               ; Eingabeparameter evaluieren und auf
Stack ablegen
     883/     457 : 70 E8              func2:      push    R8        ; Proc-Adr. sichern
     884/     459 : 70 E9                         push    R9
     885/     45B : D6 04 C7                      call    expr        ; Eingabeparameter
evaluieren -> RR2
     886/     45E : 50 E9                         pop    R9        ; Proc-Adr. restaurieren
     887/     460 : 50 E8                         pop    R8
     888/     462 : 7C 5D                         ld    R7, #']'
     889/     464 : D6 03 91                      call    next_char
     890/     467 : 6B 06                         jr    Z, func3    ; wenn Ende
Parameterliste
     891/     469 : 70 E3                         push    R3        ; sonst Wert auf Stack
     892/     46B : 70 E2                         push    R2
     893/     46D : 8B E8                         jr    func2        ; und nächsten
Parameter
     894/     46F :                    ; X1..Xn-1 liegen auf Stack, Xn in RR2
     895/     46F : 48 E2              func3:      ld    R4, R2        ; Y = X setzen
     896/     471 : 58 E3                         ld    R5, R3
     897/     473 : D4 E8                         call    @RR8        ; Procedur starten
     898/     475 : 31 10                         srp    #10h        ; zur Sicherheit f.
ext. Proceduren
     899/     477 : DF                            scf            ; Cy=Erfolg
     900/     478 : AF                            ret
     901/     479 :
     902/     479 :                    ;-----------------------------------------------
----------------------------
     903/     479 :                    ; Ausdruck
     904/     479 :                    ; Variable, Prozedur[..], Hex-Zahl, Dez-Zahl
     905/     479 :                    ;-----------------------------------------------
----------------------------
     906/     479 :
     907/     479 : D6 03 D9           factor:     call    is_letter
     908/     47C : FB 17                         jr    NC, factor1    ; wenn kein
Buchstabe A..Z
     909/     47E : C2 70                         ldc    R7, @RR0
     910/     480 : FE                            inc    R15
     911/     481 : D6 03 E2                      call    is_char        ; folgt Buchstabe
oder Zahl?
     912/     484 : 7B C3                         jr    C, func        ; ja, dann
Prozedurname
     913/     486 :                               ; sonst einzelne Variable A..Z
     914/     486 : 26 E6 41                      sub    R6, #41h    ; 'A'
     915/     489 : 90 E6                         rl    R6        ; *2
     916/     48B : 06 E6 20                      add    R6, #20h    ; Register 20h..53h
```

```
     917/     48E : E3 26                        ld    R2, @R6        ; Wert nach R2/R3
(Y)
     918/     490 : 6E                           inc    R6
     919/     491 : E3 36                        ld    R3, @R6
     920/     493 : DF                           scf
     921/     494 : AF                           ret
     922/     495 :
     923/     495 : A7 EF 3B           factor1:    cp    @R15, #';'    ; Statement-Ende
     924/     498 : 6B 19                        jr    Z, factor3
     925/     49A : A7 EF 0D                     cp    @R15, #0Dh    ; CR, Zeilenende
     926/     49D : 6B 14                        jr    Z, factor3
     927/     49F : FE                           inc    R15
     928/     4A0 : C3 F0              factor2:    ldci    @R15, @RR0    ; Ablegen in
Konvertierungspuffer
     929/     4A2 : 00 EF                        dec    R15
     930/     4A4 : D6 03 F2                     call    is_hexdigit
     931/     4A7 : FE                           inc    R15
     932/     4A8 : 7B F6                        jr    C, factor2    ; solange (Hex-
)Ziffer 0..F
     933/     4AA : 80 E0                        decw    RR0
     934/     4AC : D6 01 D9                     call    number        ; Zahl konvertieren
     935/     4AF : 28 E4                        ld    R2, R4
     936/     4B1 : 38 E5                        ld    R3, R5
     937/     4B3 : AF                 factor3:    ret
     938/     4B4 :
     939/     4B4 :                    ;-----------------------------------------------
-----------------------------
     940/     4B4 :                    ; geklammerter Ausdruck
     941/     4B4 :                    ;-----------------------------------------------
-----------------------------
     942/     4B4 :
     943/     4B4 : 7C 28              term:       ld    R7, #'('    ; linke Klammer
     944/     4B6 : D6 03 91                     call    next_char
     945/     4B9 : EB BE                        jr    NZ, factor    ; wenn keine Klammer
     946/     4BB : D6 04 C7                     call    expr        ; Ausdruck berechnen
     947/     4BE : FB 06                        jr    NC, term1
     948/     4C0 : 7C 29                        ld    R7, #')'    ; rechte Klammer
     949/     4C2 : D6 03 91                     call    next_char
     950/     4C5 : EF                           ccf
     951/     4C6 : AF                 term1:      ret
     952/     4C7 :
     953/     4C7 :                    ;-----------------------------------------------
-----------------------------
     954/     4C7 :                    ; Ausdruck berechnen von links nach rechts
(Kettenrechnen)
     955/     4C7 :                    ; Rückgabe Wert in RR2
     956/     4C7 :                    ;-----------------------------------------------
-----------------------------
     957/     4C7 :
     958/     4C7 : D6 04 B4           expr:       call    term        ; linker Wert
     959/     4CA : FB 09                        jr    NC, expr2
     960/     4CC : D6 03 91           expr1:      call    next_char
     961/     4CF : D6 03 9A                     call    oper        ; Operator
     962/     4D2 : 6B 04                        jr    Z, expr3    ; wenn Operator
gefunden
     963/     4D4 : DF                           scf            ; sonst Fehler
     964/     4D5 : 80 E0              expr2:      decw    RR0
     965/     4D7 : AF                           ret
     966/     4D8 :                    ;
     967/     4D8 : 70 E8              expr3:      push    R8        ; RR8 = Op-
```

```
Fkt.Adresse
     968/     4DA : 70 E9                        push    R9
     969/     4DC : 70 E2                        push    R2        ; aktueller linker Wert
(Y)
     970/     4DE : 70 E3                        push    R3
     971/     4E0 : D6 04 B4                     call    term        ; rechter Wert
     972/     4E3 : 48 E2                        ld    R4, R2        ; nach RR4 (X)
     973/     4E5 : 58 E3                        ld    R5, R3
     974/     4E7 : 50 E3                        pop    R3        ; Y restaurieren
     975/     4E9 : 50 E2                        pop    R2
     976/     4EB : 50 E9                        pop    R9        ; Adr. restaurieren
     977/     4ED : 50 E8                        pop    R8
     978/     4EF : D4 E8                        call    @RR8        ; berechnen
     979/     4F1 :                      ;
     980/     4F1 : 8B D9                        jr    expr1        ; weitere Operationen
     981/     4F3 :
     982/     4F3 :                      ;-----------------------------------------------
-----------------------------
     983/     4F3 :                      ; holt Speicheradresse zu Variable A..Z
     984/     4F3 :                      ; out RR8 = Adr. in Registerspeicher (20h..53h)
     985/     4F3 :                      ;-----------------------------------------------
-----------------------------
     986/     4F3 :
     987/     4F3 : D6 03 91           get_var:    call    next_char
     988/     4F6 : 26 E6 41                     sub    R6, #'A'
     989/     4F9 : 02 66                        add    R6, R6
     990/     4FB : 8C 20                        ld    R8, #20h    ; Register 20h..53h
     991/     4FD : 02 86                        add    R8, R6
     992/     4FF : AF                           ret
     993/     500 :
     994/     500 :                      ;-----------------------------------------------
-----------------------------
     995/     500 :                      ; Zeilennummer ermitteln
     996/     500 :                      ; ret: RR4 = zeilennummer, RR0 = Zeilenanfang;
Cy=1 Programmanfang
     997/     500 :                      ;-----------------------------------------------
-----------------------------
     998/     500 :
     999/     500 :                      ; UP zu GOTO
    1000/     500 :                      linum:      ;auf Zeilenanfang
    1001/     500 : C2 40                        ldc    R4, @RR0
    1002/     502 : 80 E0                        decw    RR0        ; aktuelle Zeile
rückwärts
    1003/     504 : 76 E4 80                     tm    R4, #80h    ; Bit7 gesetzt?
    1004/     507 : 6B F7                        jr    Z, linum    ; rückwärts bis
Zeilenanfang
    1005/     509 : A0 E0                        incw    RR0
    1006/     50B :                              ;Programmanfang erreicht?
    1007/     50B : 56 E4 7F                     and    R4, #7Fh    ; strip hi bit
    1008/     50E : A4 06 E0                     cp    R0, reg_06    ; Vergleich Adr. mit
Programmanfang
    1009/     511 : EB 06                        jr    NZ, linum1    ;
    1010/     513 : A4 07 E1                     cp    R1, reg_07
    1011/     516 : DF                           scf
    1012/     517 : 6B 0F                        jr    Z, linum2    ; Programmanfang
erreicht
    1013/     519 :                              ;
    1014/     519 :                      linum1:     ;steht vor gesetztem hi-bit ein CR?
dann Zeilenanfang gefunden
    1015/     519 : 80 E0                        decw    RR0
```

```
    1016/     51B : C2 50                       ldc    R5, @RR0
    1017/     51D : A0 E0                       incw    RR0
    1018/     51F : A6 E5 0D                    cp    R5, #0Dh    ; CR, Suche Zeilenende
    1019/     522 : 6B 04                       jr    Z, linum2    ; ja, Zeilenanfang
gefunden
    1020/     524 : 80 E0                       decw    RR0
    1021/     526 : 8B D8                       jr    linum        ; sonst weitersuchen
    1022/     528 :                             ;
    1023/     528 : A0 E0               linum2:     incw    RR0
    1024/     52A : C2 50                       ldc    R5, @RR0    ; RR4=Zeilennummer
    1025/     52C : 80 E0                       decw    RR0
    1026/     52E : AF                          ret
    1027/     52F :
    1028/     52F :                     ;-----------------------------------------------
------------------------------
    1029/     52F :                     ; Ausgabe "text"
    1030/     52F :                     ;-----------------------------------------------
------------------------------
    1031/     52F :
    1032/     52F : 7C 22               prnstr:     ld    R7, #'"'
    1033/     531 : D6 03 91                    call    next_char
    1034/     534 : EB 0C                       jr    NZ, prnstr2    ; kein Quote ->
Abbruch
    1035/     536 : D6 03 91            prnstr1:    call    next_char    ; nächstes
Zeichen
    1036/     539 : 6B 09                       jr    Z, prnstr3    ; Ende beim
abschließenden Quote
    1037/     53B : 58 E6                       ld    R5, R6        ; nach R5
    1038/     53D : D6 08 18                    call    putch        ; mit PUT_CHAR
ausgeben
    1039/     540 : 8B F4                       jr    prnstr1
    1040/     542 :                     ;
    1041/     542 : 80 E0               prnstr2:    decw    RR0        ; Pointer
zurückstellen
    1042/     544 : AF                  prnstr3:    ret
    1043/     545 :
    1044/     545 :
    1045/     545 :                     ;-----------------------------------------------
------------------------------
    1046/     545 :                     ; logischer Ausdruck
    1047/     545 :                     ; expr. relop expr., ret NZ wenn true, Z wenn
false
    1048/     545 :                     ;-----------------------------------------------
------------------------------
    1049/     545 :
    1050/     545 : D6 04 C7            relop:  call    expr        ; linker wert
    1051/     548 : B0 EA                       clr    R10
    1052/     54A : 9C 02                       ld    R9, #2        ; max 2 Zeichen
    1053/     54C : 8C 10               relop1:     ld    R8, #10h
    1054/     54E : C2 60                       ldc    R6, @RR0
    1055/     550 : A6 E6 3C                    cp    R6, #'<'
    1056/     553 : 6B 0E                       jr    Z, relop2    ; < R8=10
    1057/     555 : 90 E8                       rl    R8
    1058/     557 : A6 E6 3E                    cp    R6, #'>'
    1059/     55A : 6B 07                       jr    Z, relop2    ; > R8=20
    1060/     55C : 90 E8                       rl    R8
    1061/     55E : A6 E6 3D                    cp    R6, #'='
    1062/     561 : EB 06                       jr    NZ, relop3    ; = R8=40
    1063/     563 : 42 A8               relop2:     or    R10, R8        ; <= R8=50, >=
R8=60
```

```
    1064/     565 : A0 E0                        incw    RR0
    1065/     567 : 9A E3                        djnz    R9, relop1
    1066/     569 :                              ;
    1067/     569 : 70 EA                relop3:     push    R10
    1068/     56B : 70 E2                        push    R2
    1069/     56D : 70 E3                        push    R3
    1070/     56F : D6 04 C7                     call    expr        ; rechter Wert
    1071/     572 : 48 E2                        ld    R4, R2        ; RR4 = rechter Wert
    1072/     574 : 58 E3                        ld    R5, R3
    1073/     576 : 50 E3                        pop    R3        ; RR2 = linker Wert
    1074/     578 : 50 E2                        pop    R2
    1075/     57A : D6 01 41                     call    relcmp        ; Vergleich
    1076/     57D : 50 E8                        pop    R8
    1077/     57F : 54 0F E8                     and    R8, reg_0F    ; Mit relop
verknüpfen
    1078/     582 : AF                           ret            ; Z=1 => relop false
    1079/     583 :
    1080/     583 :                    ;-------------------------------------------------
----------------------------
    1081/     583 :                    ; REM kommentar
    1082/     583 :                    ;-------------------------------------------------
----------------------------
    1083/     583 :
    1084/     583 : 7C 3B              c_REM:      ld    R7, #';'
    1085/     585 : D6 03 91           c_REM1:     call    next_char
    1086/     588 : 6B 06                        jr    Z, c_REM2    ; überlesen bis ;
    1087/     58A : A6 E6 0D                     cp    R6, #0Dh    ; CR
    1088/     58D : DF                           scf
    1089/     58E : EB F5                        jr    NZ, c_REM1    ; oder Zeilenende
    1090/     590 : 80 E0              c_REM2:     decw    RR0
    1091/     592 : AF                 c_REM3:     ret
    1092/     593 :
    1093/     593 :
    1094/     593 :                    ;-------------------------------------------------
----------------------------
    1095/     593 :                    ; Goto, If, Else: Rest bis Zeilenende übergehen
    1096/     593 :                    ;-------------------------------------------------
----------------------------
    1097/     593 :
    1098/     593 : D6 05 83           skip:       call    c_REM        ; Rest übergehen
    1099/     596 : 7B FA                        jr    C, c_REM3    ; Ende bei CR
    1100/     598 : A0 E0                        incw    RR0        ; sonst auch nächste
Anw.
    1101/     59A : 8B F7                        jr    skip        ; übergehen
    1102/     59C :
    1103/     59C :                    ;-------------------------------------------------
----------------------------
    1104/     59C :                    ; LET variable=ausdruck
    1105/     59C :                    ;-------------------------------------------------
----------------------------
    1106/     59C :
    1107/     59C : D6 04 F3           c_LET:      call    get_var        ; @RR8 =
Variablenadr.
    1108/     59F : 70 E8                        push    R8
    1109/     5A1 : A0 E0                        incw    RR0
    1110/     5A3 : D6 04 C7                     call    expr        ; RR2 = ausdruck
    1111/     5A6 : 50 E8                        pop    R8
    1112/     5A8 : F3 82              c_LET1:     ld    @R8, R2        ; Wert
abspeichern
    1113/     5AA : 8E                           inc    R8
```

```
    1114/     5AB : F3 83                       ld    @R8, R3
    1115/     5AD : AF                          ret
    1116/     5AE :
    1117/     5AE :
    1118/     5AE :                   ;-----------------------------------------------
------------------------------
    1119/     5AE :                   ; TRAP
    1120/     5AE :                   ; TRAP bedingung TO ausdruck
    1121/     5AE :                   ;-----------------------------------------------
------------------------------
    1122/     5AE :                   ; Trap setzen
    1123/     5AE :                   ; der Rest der Anweisung wird hier übergangen
    1124/     5AE :
    1125/     5AE : 09 04             c_TRAP:     ld    reg_04, R0    ; RR0 = akt. Pos.
in Programm
    1126/     5B0 : 19 05                         ld    reg_05, R1
    1127/     5B2 : 8B CF             c_TRAP1:    jr    c_REM
    1128/     5B4 :
    1129/     5B4 :                   ;-----------------------------------------------
------------------------------
    1130/     5B4 :                   ; PROC [variablenliste] = prozedurname
[parameterliste]
    1131/     5B4 :                   ; PROC [Y1,Y2,Y3,..,Ym] = prozedurname
[X1,X2,X3,..,Xn]
    1132/     5B4 :                   ;-----------------------------------------------
------------------------------
    1133/     5B4 :                   ; bei Prozeduraufruf gilt:
    1134/     5B4 :                   ; Stack SP+2m+2n    Ym-1            je 2 Byte
Werte
    1135/     5B4 :                   ;   ..        Ym-2
    1136/     5B4 :                   ;   ..        ..
    1137/     5B4 :                   ;   SP+2n+4       Y2
    1138/     5B4 :                   ;   SP+2n+2       Y1
    1139/     5B4 :                   ;   SP+2n        (intern f. interpreter)
    1140/     5B4 :                   ;   SP+2n-2       X1
    1141/     5B4 :                   ;   ..        ..
    1142/     5B4 :                   ;   SP+4        Xn-2
    1143/     5B4 :                   ;   SP+2        Xn-1
    1144/     5B4 :                   ;   SP        returnadr. zum interpreter
    1145/     5B4 :                   ; die rechten Parameter Ym, Xn werden in
Registern abgelegt
    1146/     5B4 :                   ; out RR2 = Ym, in RR4 Xm
    1147/     5B4 :                   ;-----------------------------------------------
------------------------------
    1148/     5B4 :
    1149/     5B4 : 70 04             c_PROC:     push    reg_04        ; aktuelle TRAP
sichern
    1150/     5B6 : 70 05                         push    reg_05
    1151/     5B8 : 09 04                         ld    reg_04, R0    ; aktuelle
Programmposition sichern
    1152/     5BA : 19 05                         ld    reg_05, R1
    1153/     5BC :                   ;Ausgabeparameter auf Stack initialisieren
    1154/     5BC : C2 60                         ldc    R6, @RR0    ; nächstes Zeichen
testen
    1155/     5BE : A6 E6 5B                      cp    R6, #'['    ; folgen
Ausgabeparameter Y?
    1156/     5C1 : EB 15                         jr    NZ, c_PROC3    ; nein
    1157/     5C3 : A0 E0                         incw    RR0
    1158/     5C5 : B0 E8                         clr    R8        ; 0
    1159/     5C7 : 7C 5D                         ld    R7, #']'
```

```
    1160/     5C9 : A0 E0               c_PROC1:    incw    RR0
    1161/     5CB : D6 03 91                        call    next_char
    1162/     5CE : 6B 06                           jr    Z, c_PROC2
    1163/     5D0 : 70 E8                           push    R8        ; Stackspeicher f.
Ausgabeparameter
    1164/     5D2 : 70 E8                           push    R8        ; belegen (mit Wert 0)
    1165/     5D4 : 8B F3                           jr    c_PROC1
    1166/     5D6 : A0 E0               c_PROC2:    incw    RR0
    1167/     5D8 :                     ;Prozedurnamen suchen, Eingabeparameter
evaluieren und ablegen, Prozedur ausführen
    1168/     5D8 : D6 04 4B            c_PROC3:    call    func1
    1169/     5DB :                     ;Ausgabeparameter von Stack in Variablen
schreiben
    1170/     5DB : 08 04                           ld    R0, reg_04    ; Programmposition
auf Anfang setzen
    1171/     5DD : 18 05                           ld    R1, reg_05
    1172/     5DF : 7C 5B                           ld    R7, #'['
    1173/     5E1 : D6 03 91                        call    next_char
    1174/     5E4 : EB 18                           jr    NZ, c_PROC6    ; wenn keine
Ausgabeparameter
    1175/     5E6 : 7C 5D                           ld    R7, #']'
    1176/     5E8 : D6 04 F3            c_PROC4:    call    get_var        ; get Variable
    1177/     5EB : D6 03 91                        call    next_char
    1178/     5EE : 6B 0B                           jr    Z, c_PROC5    ; Ende bei ]
    1179/     5F0 : 50 EA                           pop    R10        ; Parameter
    1180/     5F2 : F3 8A                           ld    @R8, R10    ; in Variable schreiben
(Hi)
    1181/     5F4 : 50 EA                           pop    R10
    1182/     5F6 : 8E                              inc    R8
    1183/     5F7 : F3 8A                           ld    @R8, R10    ; Lo
    1184/     5F9 : 8B ED                           jr    c_PROC4        ; nächsten Parameter
    1185/     5FB : D6 05 A8            c_PROC5:    call    c_LET1
    1186/     5FE : 50 05               c_PROC6:    pop    reg_05        ; TRAP
wiederherstellen
    1187/     600 : 50 04                           pop    reg_04
    1188/     602 : 8B AE                           jr    c_TRAP1
    1189/     604 :
    1190/     604 :                     ;-------------------------------------------------
------------------------------
    1191/     604 :                     ; GOTO ausdruck
    1192/     604 :                     ;-------------------------------------------------
------------------------------
    1193/     604 :                     ; suche passende zeile oder nächstgrößere; bei
Programmende -> Ende
    1194/     604 :
    1195/     604 : D6 04 C7            c_GOTO:     call    expr        ; RR2 = ausdruck
    1196/     607 : D6 05 00                        call    linum        ; RR4 = aktuelle
Zeilennummer
    1197/     60A : D6 01 41                        call    relcmp        ; vergleichen
    1198/     60D : 76 0F 50                        tm    reg_0F, #50h    ; <=
    1199/     610 : EB 1F                           jr    NZ, c_GOTO3    ; wenn ausdruck <=
akt. Zeile
    1200/     612 :
    1201/     612 :                     c_GOTO1:    ; Suche vorwärts
    1202/     612 : D6 05 93                        call    skip        ; Rest bis Zeilenende
übergehen
    1203/     615 : A0 E0                           incw    RR0
    1204/     617 : C2 40                           ldc    R4, @RR0    ; RR4 = nächste
Zeilennummer
    1205/     619 : A0 E0                           incw    RR0
```

```
    1206/     61B : C2 50                       ldc    R5, @RR0
    1207/     61D : 80 E0                       decw    RR0
    1208/     61F : 90 E4                       rl    R4        ; Bit7 nach Cy und Bit0
    1209/     621 : CF                          rcf            ; Cy=0
    1210/     622 : C0 E4                       rrc    R4        ; Bit7=0, Cy=orig. Bit7
    1211/     624 : FB 08                       jr    NC, c_GOTO2    ; Programmende
erreicht?
    1212/     626 : D6 01 41                    call    relcmp
    1213/     629 : 76 0F 50                    tm    reg_0F, #50h    ; <=
    1214/     62C : 6B E4                       jr    Z, c_GOTO1    ; weiter, solange
ausdruck <= akt. Zeile
    1215/     62E :                   c_GOTO2:    ; Programmende erreicht
    1216/     62E : 80 E0                       decw    RR0
    1217/     630 : AF                          ret
    1218/     631 :
    1219/     631 : D6 05 00          c_GOTO3:    call    linum        ; RR4 = aktuelle
Zeilennummer
    1220/     634 : FB 07                       jr    NC, c_GOTO4    ; solange
    1221/     636 :
    1222/     636 :                             ; Wenn Zeile gefunden
    1223/     636 : A0 FE                       incw    gpr        ; Stackpointer
    1224/     638 : A0 FE                       incw    gpr        ; Stackpointer
    1225/     63A : 8D 07 3E                    jp    run2        ; Zeile abarbeiten
    1226/     63D :
    1227/     63D :                   c_GOTO4:    ; suche Rückwärts
    1228/     63D : D6 01 41                    call    relcmp
    1229/     640 : 76 0F 60                    tm    reg_0F, #60h    ; >=
    1230/     643 : EB E9                       jr    NZ, c_GOTO2
    1231/     645 : 80 E0                       decw    RR0        ; ein Zeichen zurück
    1232/     647 : 8B E8                       jr    c_GOTO3        ; weitersuchen
    1233/     649 :
    1234/     649 :                   ;-----------------------------------------------
------------------------------
    1235/     649 :                   ; IF bedingung THEN anweisungen
    1236/     649 :                   ;-----------------------------------------------
------------------------------
    1237/     649 :
    1238/     649 : 56 0F FE          c_IF:       and    reg_0F, #0FEh    ; Bit0 = 0
    1239/     64C : D6 05 45                    call    relop        ; bedingung
auswerten
    1240/     64F : EB 05                       jr    NZ, c_IF1    ; Bedinung erfüllt
    1241/     651 :                                     ; -> nächste Anweisung
abarbeiten (nach ;)
    1242/     651 : 20 0F                       inc    reg_0F        ; Bit0 = 1 ( ->
ELSE)
    1243/     653 : 8D 05 93                    jp    skip        ; Rest bis Zeilenende
übergehen
    1244/     656 : AF                c_IF1:      ret
    1245/     657 :
    1246/     657 :                   ;-----------------------------------------------
------------------------------
    1247/     657 :                   ; ELSE anweisungen
    1248/     657 :                   ;-----------------------------------------------
------------------------------
    1249/     657 :
    1250/     657 : 76 0F 01          c_ELSE:     tm    reg_0F, #1    ; ELSE aktiv?
    1251/     65A : 6D 05 93                    jp    Z, skip        ; nein, Rest bis
Zeilenende übergehen
    1252/     65D : 56 0F FE                    and    reg_0F, #0FEh    ; Bit0 = 0 (kein
ELSE aktiv)
```

```
    1253/     660 : AF                          ret
    1254/     661 :
    1255/     661 :                     ;-----------------------------------------------
------------------------------
    1256/     661 :                     ; RETURN
    1257/     661 :                     ;-----------------------------------------------
------------------------------
    1258/     661 :
    1259/     661 : A6 0E 00            c_RETURN:   cp    reg_0E, #0    ;
Verschachtelungstiefe 0?
    1260/     664 : EB 06                       jr    NZ, c_RETURN2    ; nein
    1261/     666 : 46 0E 20                    or    reg_0E, #20h    ; Bit5 = 1
    1262/     669 : 8D 01 1B            c_RETURN1:  jp    p_div6        ; reg_0F Bit7=1
    1263/     66C :                             ;
    1264/     66C : 00 0E               c_RETURN2:  dec    reg_0E        ;
Verschachtelungstiefe verringern
    1265/     66E : 50 E6                       pop    R6        ; Return-Adr.
    1266/     670 : 50 E7                       pop    R7
    1267/     672 : 50 E0                       pop    R0        ; RR0= Pos. nach GOSUB
    1268/     674 : 50 E1                       pop    R1
    1269/     676 : 56 0F FE                    and    reg_0F, #0FEh    ; Bit0 = 0 (kein
ELSE aktiv)
    1270/     679 : 30 E6                       jp    @RR6        ; return
    1271/     67B :
    1272/     67B :                     ;-----------------------------------------------
------------------------------
    1273/     67B :                     ; GOSUB ausdruck
    1274/     67B :                     ;-----------------------------------------------
------------------------------
    1275/     67B :
    1276/     67B : 88 E0               c_GOSUB:    ld    R8, R0        ; aktuelle
Position
    1277/     67D : 98 E1                       ld    R9, R1        ; sichern
    1278/     67F : D6 05 83                    call    c_REM        ; RR0=Kommandoende
    1279/     682 : 50 EA                       pop    R10        ; Return-Adresse vom
Stack
    1280/     684 : 50 EB                       pop    R11
    1281/     686 : 70 E1                       push    R1        ; Pos. nach GOSUB
    1282/     688 : 70 E0                       push    R0        ; auf Stack
    1283/     68A : 70 EB                       push    R11        ; Return-Adresse
restaurieren
    1284/     68C : 70 EA                       push    R10
    1285/     68E : 08 E8                       ld    R0, R8        ; orig. Position
    1286/     690 : 18 E9                       ld    R1, R9        ; rückschreiben
    1287/     692 : 20 0E               c_GOSUB1:   inc    reg_0E        ;
Verschachtelungstiefe erhöhen
    1288/     694 : 76 0E 10                    tm    reg_0E, #10h    ; > 15 ?
    1289/     697 : 6D 06 04                    jp    Z, c_GOTO    ; nein, UP aufrufen
    1290/     69A :                             ; max Verschachtelungstiefe erreicht
    1291/     69A : D6 05 83                    call    c_REM        ; bis Kommandoende
überlesen
    1292/     69D : 8B CA                       jr    c_RETURN1    ; und
Verschachtelungstiefe verringern
    1293/     69F :
    1294/     69F :                     ;-----------------------------------------------
------------------------------
    1295/     69F :                     ; WAIT ausdruck
    1296/     69F :                     ; Schleifendauer 1 ms bei 8 MHz Taktfrequenz
    1297/     69F :                     ;-----------------------------------------------
------------------------------
```

```
    1298/     69F :
    1299/     69F : D6 04 C7            c_WAIT:     call    expr        ; RR2 = ausdruck
    1300/     6A2 : 68 E2                       ld    R6, R2
    1301/     6A4 : 42 63                       or    R6, R3        ; = 0?
    1302/     6A6 : 6B 10                       jr    Z, c_WAIT3    ; dann kein Wait
    1303/     6A8 : 40 E6               c_WAIT1:    da    R6
    1304/     6AA : 6C 00                       ld    R6, #0
    1305/     6AC : 7C B4                       ld    R7, #0B4h
    1306/     6AE : 80 E6               c_WAIT2:    decw    RR6
    1307/     6B0 : ED 06 AE                    jp    NZ, c_WAIT2
    1308/     6B3 : 80 E2                       decw    RR2
    1309/     6B5 : ED 06 A8                    jp    NZ, c_WAIT1
    1310/     6B8 : AF                  c_WAIT3:    ret
    1311/     6B9 :
    1312/     6B9 :                     ;-----------------------------------------------
------------------------------
    1313/     6B9 :                     ; CALL ausdruck
    1314/     6B9 :                     ;-----------------------------------------------
------------------------------
    1315/     6B9 :
    1316/     6B9 : D6 04 C7            c_CALL:     call    expr        ; RR2 = ausdruck
    1317/     6BC : D4 E2                       call    @RR2        ; UP aufrufen
(berechnete Adr.)
    1318/     6BE : 31 10                       srp    #10h        ; RP auf Standard
    1319/     6C0 : AF                          ret
    1320/     6C1 :
    1321/     6C1 :                     ;-----------------------------------------------
------------------------------
    1322/     6C1 :                     ; STOP
    1323/     6C1 :                     ;-----------------------------------------------
------------------------------
    1324/     6C1 :
    1325/     6C1 : 46 0F 08            c_STOP:     or    reg_0F, #8    ; Bit3 = 1
    1326/     6C4 : AF                          ret
    1327/     6C5 :
    1328/     6C5 :                     ;-----------------------------------------------
------------------------------
    1329/     6C5 :                     ; END
    1330/     6C5 :                     ;-----------------------------------------------
------------------------------
    1331/     6C5 :
    1332/     6C5 : 46 0F 02            c_END:      or    reg_0F, #2    ; Bit1 = 1
    1333/     6C8 : AF                          ret
    1334/     6C9 :
    1335/     6C9 :                     ;-----------------------------------------------
------------------------------
    1336/     6C9 :                     ; CLRTRP
    1337/     6C9 :                     ;-----------------------------------------------
------------------------------
    1338/     6C9 :
    1339/     6C9 : B0 04               c_CLRTRAP:  clr    reg_04        ; Trap auf 0
setzen
    1340/     6CB : B0 05                       clr    reg_05
    1341/     6CD : AF                          ret
    1342/     6CE :
    1343/     6CE :                     ;-----------------------------------------------
------------------------------
    1344/     6CE :                     ; PRINTHEX "text" ausdruck
    1345/     6CE :                     ;-----------------------------------------------
------------------------------
```

```
    1346/     6CE :
    1347/     6CE : D6 05 2F           c_PRINTHEX: call    prnstr        ; Ausgabe text
    1348/     6D1 : D6 04 C7                       call    expr        ; Ausdruck berechnen
    1349/     6D4 : FB 22                          jr    NC, c_PRINT3    ; bei Fehler o.
ohne Ausdruck
    1350/     6D6 : D6 01 5E                       call    tohex        ; Konvertierung nach
Hex-String
    1351/     6D9 :                                     ; In Buffer ab #14h
    1352/     6D9 : AC 05                          ld    R10, #5        ; 5 Stellen
    1353/     6DB : 8B 0D                          jr    c_PRINT1
    1354/     6DD :
    1355/     6DD :                    ;-----------------------------------------------
----------------------------
    1356/     6DD :                    ; PRINT "text" ausdruck
    1357/     6DD :                    ;-----------------------------------------------
----------------------------
    1358/     6DD :
    1359/     6DD : D6 05 2F           c_PRINT:    call    prnstr        ; Ausgabe text
    1360/     6E0 : D6 04 C7                       call    expr        ; Ausdruck berechnen
    1361/     6E3 : FB 13                          jr    NC, c_PRINT3    ; bei Fehler o.
ohne Ausdruck keine Ausgabe
    1362/     6E5 :                                ; Ausdruck anzeigen
    1363/     6E5 : D6 01 82                       call    todez        ; Konvertierung nach
dezimal
    1364/     6E8 : AC 06                          ld    R10, #6        ; 6 Stellen
    1365/     6EA : BC 14              c_PRINT1:   ld    R11, #14h    ;
Konvertierungspuffer
    1366/     6EC : 70 E5              c_PRINT2:   push    R5
    1367/     6EE : E3 5B                          ld    R5, @R11
    1368/     6F0 : BE                             inc    R11
    1369/     6F1 : D6 08 18                       call    putch        ; Stelle ausgeben
    1370/     6F4 : 50 E5                          pop    R5
    1371/     6F6 : AA F4                          djnz    R10, c_PRINT2
    1372/     6F8 :                                ; Am Ende Zeilenende ausgeben, falls kein
Komma
    1373/     6F8 : C2 60              c_PRINT3:   ldc    R6, @RR0
    1374/     6FA : A6 E6 2C                       cp    R6, #','    ; folgt Komma?
    1375/     6FD : EB 11                          jr    NZ, c_PRINT5    ; nein -> CR
    1376/     6FF : A0 E0                          incw    RR0
    1377/     701 : C2 60                          ldc    R6, @RR0
    1378/     703 : A6 E6 3B                       cp    R6, #';'    ; folgt Kdo-Ende?
    1379/     706 : 6B 07                          jr    Z, c_PRINT4    ; ja -> kein CR
ausgeben
    1380/     708 : A6 E6 0D                       cp    R6, #0Dh    ; oder folgt Zeilenende
CR?
    1381/     70B : 6B 02                          jr    Z, c_PRINT4    ; dann ebenfalls
kein CR ausgeben
    1382/     70D : 80 E0                          decw    RR0
    1383/     70F : AF                 c_PRINT4:   ret
    1384/     710 : 5C 0D              c_PRINT5:   ld    R5, #0Dh    ; CR
    1385/     712 : 8D 08 18                       jp    putch        ; ausgeben
(Zeilenende)
    1386/     715 :
    1387/     715 :                    ;-----------------------------------------------
----------------------------
    1388/     715 :                    ; INPUT "text" variable
    1389/     715 :                    ;-----------------------------------------------
----------------------------
    1390/     715 :
    1391/     715 : D6 05 2F           c_INPUT:    call    prnstr        ; Ausgabe text
```

```
    1392/     718 : D6 02 E9                  call    p_input1    ; Abfrage Wert
    1393/     71B : D6 04 F3                  call    get_var        ; Variablenadresse
    1394/     71E : 8D 05 A8                  jp    c_LET1        ; und Wert zuweisen
    1395/     721 :
    1396/     721 :                   ;-----------------------------------------------
------------------------------
    1397/     721 :                   ; Einsprung aus BM200 (CONT)
    1398/     721 :                   ;-----------------------------------------------
------------------------------
    1399/     721 :
    1400/     721 : E6 0F 04          cont:       ld    reg_0F, #4
    1401/     724 : 8B 10                     jr    run1
    1402/     726 :
    1403/     726 :                   ;-----------------------------------------------
------------------------------
    1404/     726 :                   ; Einsprung aus BM200 (STEP)
    1405/     726 :                   ;-----------------------------------------------
------------------------------
    1406/     726 :
    1407/     726 : E6 0F 08          step:       ld    reg_0F, #8
    1408/     729 : 8B 0B                     jr    run1
    1409/     72B :
    1410/     72B :                   ;-----------------------------------------------
------------------------------
    1411/     72B :                   ; RUN
    1412/     72B :                   ; in 6,7 = Startadr. Basic-Programm
    1413/     72B :                   ;    8,9 = Adr. Prozedurtabelle (oder 0)
    1414/     72B :                   ; srp #10h; call $7fd
    1415/     72B :                   ;-----------------------------------------------
------------------------------
    1416/     72B :
    1417/     72B :                   ; Programmstart
    1418/     72B : B0 0F             run:        clr    reg_0F
    1419/     72D : B0 0E                     clr    reg_0E
    1420/     72F : 08 06                     ld    R0, reg_06    ; RR0 = Startadr.
Basic-Programm
    1421/     731 : 18 07                     ld    R1, reg_07
    1422/     733 : D6 06 C9                  call    c_CLRTRAP    ; clear trap
    1423/     736 :                   ;
    1424/     736 : 31 10             run1:       srp    #10h        ; Standard setzen
    1425/     738 : 50 0A                     pop    reg_0A        ; Return-Adresse f.
END
    1426/     73A : 50 0B                     pop    reg_0B
    1427/     73C : 8B 0A                     jr    run3
    1428/     73E :
    1429/     73E :                   ; nächste Zeile abarbeiten
    1430/     73E : 76 0F 0A          run2:       tm    reg_0F, #0Ah
    1431/     741 : EB 0A                     jr    NZ, run4    ; END
    1432/     743 : 66 0F 84                  tcm    reg_0F, #84h    ;-7Ch
    1433/     746 : 6B 05                     jr    Z, run4    ; END
    1434/     748 :
    1435/     748 : C2 60             run3:       ldc    R6, @RR0    ;
R6=Hi(Zeilennummer)+$80
    1436/     74A : 6E                        inc    R6
    1437/     74B : 6A 02                     djnz    R6, run5
    1438/     74D :                   ;END
    1439/     74D : 30 0A             run4:       jp    @reg_0A        ; END bei
Zeilenummer > 7Fxxh (32512)
    1440/     74F :
    1441/     74F :                   ;
```

```
    1442/     74F : A4 E0 06            run5:       cp    reg_06, R0
    1443/     752 : EB 05                       jr    NZ, run6
    1444/     754 : A4 E1 07                    cp    reg_07, R1
    1445/     757 : 6B 2C                       jr    Z, run8
    1446/     759 :                     ; Test auf TRAP
    1447/     759 : 68 04               run6:       ld    R6, reg_04    ; TRAP
    1448/     75B : 44 05 E6                    or    R6, reg_05
    1449/     75E : 6B 25                       jr    Z, run8    ; =0? kein TRAP
    1450/     760 :                             ; TRAP
    1451/     760 : 70 E1                       push    R1        ; akt. Pogrammpos.
sichern
    1452/     762 : 70 E0                       push    R0
    1453/     764 : 08 04                       ld    R0, reg_04    ; auf TRAP-Adr.
setzen
    1454/     766 : 18 05                       ld    R1, reg_05
    1455/     768 : D6 05 45                    call    relop        ; logischen Ausdruck
auswerten
    1456/     76B : 6B 14                       jr    Z, run7    ; wenn nicht erfüllt
    1457/     76D :                             ; wenn erfüllt
    1458/     76D : D6 06 C9                    call    c_CLRTRAP    ; TRAP zurücksetzen
    1459/     770 : 50 E6                       pop    R6        ; aktuelle Return-Adr
    1460/     772 : 50 E7                       pop    R7
    1461/     774 : 80 E6                       decw    RR6        ; vermindern
    1462/     776 : 70 E7                       push    R7
    1463/     778 : 70 E6                       push    R6
    1464/     77A : A0 E0                       incw    RR0        ; akt. Pogrammpos.
(TRAP) erhöhen
    1465/     77C : D6 06 92                    call    c_GOSUB1    ; TRAP ausführen
    1466/     77F : 8B 38                       jr    run14
    1467/     781 :
    1468/     781 :                     ; Zeile abarbeiten
    1469/     781 : 50 E0               run7:       pop    R0
    1470/     783 : 50 E1                       pop    R1
    1471/     785 : A0 E0               run8:       incw    RR0        ; Zeilennummer
übergehen
    1472/     787 : A0 E0                       incw    RR0
    1473/     789 : C2 30               run9:       ldc    R3, @RR0    ; erstes Zeichen
(Kommando)
    1474/     78B : A0 E0                       incw    RR0
    1475/     78D : A6 E3 3E                    cp    R3, #'>'    ; ELSE?
    1476/     790 : 6B 03                       jr    Z, run10    ; dann ELSE-Flag
beibehalten
    1477/     792 : 56 0F FE                    and    reg_0F, #0FEh    ; sonst ELSE-Flag
rücksetzen (Bit0)
    1478/     795 : 6C 07               run10:      ld    R6, #HI(tab_kdo) ; Liste der
Kommandos
    1479/     797 : 7C C8                       ld    R7, #LO(tab_kdo)
    1480/     799 : B0 E2                       clr    R2        ; R2=0
    1481/     79B : C2 86               run11:      ldc    R8, @RR6
    1482/     79D : A2 83                       cp    R8, R3        ; Kommando in Liste
suchen
    1483/     79F : 6B 05                       jr    Z, run12    ; wenn gefunden
    1484/     7A1 : A0 E6                       incw    RR6
    1485/     7A3 : 2E                          inc    R2
    1486/     7A4 : 8B F5                       jr    run11
    1487/     7A6 :
    1488/     7A6 : 02 22               run12:      add    R2, R2
    1489/     7A8 : 6C 07                       ld    R6, #HI(tab_kdo2) ; Adressliste der
Kommandos
    1490/     7AA : 7C D9                       ld    R7, #LO(tab_kdo2)
```

```
    1491/     7AC : 02 72                       add    R7, R2
    1492/     7AE : 16 E6 00                    adc    R6, #0
    1493/     7B1 : 2C 0C                       ld    R2, #0Ch
    1494/     7B3 : C3 26                       ldci    @R2, @RR6    ; Adresse nach
reg_0ch
    1495/     7B5 : C3 26                       ldci    @R2, @RR6
    1496/     7B7 : D4 0C               run13:      call    @reg_0C        ; Kommando
aufrufen
    1497/     7B9 :                     ;
    1498/     7B9 : 7C 0D               run14:      ld    R7, #0Dh    ; CR Zeilenende?
    1499/     7BB : D6 03 91                    call    next_char
    1500/     7BE : 6D 07 3E                    jp    Z, run2        ; dann nächste Zeile
    1501/     7C1 : A6 E6 3B                    cp    R6, #';'    ; Kommandoende?
    1502/     7C4 : 6B C3                       jr    Z, run9    ; dann nächstes Kommando
in Zeile
    1503/     7C6 :                             ;
    1504/     7C6 : 8B EF                       jr    run13        ; sonst Wiederholung
aktuelles Kdo
    1505/     7C8 :
    1506/     7C8 :                     ;-----------------------------------------------
------------------------------
    1507/     7C8 :                     ; Tabelle der BASIC-Kommandos
    1508/     7C8 :                     ;-----------------------------------------------
------------------------------
    1509/     7C8 :
    1510/     7C8 : 4C                  tab_kdo:    db    'L'        ; LET
    1511/     7C9 : 4F                            db    'O'        ; PROC
    1512/     7CA : 47                            db    'G'        ; GOTO
    1513/     7CB : 46                            db    'F'        ; IF..THEN    F..;
    1514/     7CC : 3E                            db    '>'        ; ELSE        >;
    1515/     7CD : 52                            db    'R'        ; RETURN
    1516/     7CE : 53                            db    'S'        ; GOSUB
    1517/     7CF : 57                            db    'W'        ; WAIT
    1518/     7D0 : 4D                            db    'M'        ; REM
    1519/     7D1 : 43                            db    'C'        ; CALL
    1520/     7D2 : 54                            db    'T'        ; STOP
    1521/     7D3 : 45                            db    'E'        ; END
    1522/     7D4 : 21                            db    '!'        ; TRAP..TO    !..,
    1523/     7D5 : 2F                            db    '/'        ; CLRTRAP
    1524/     7D6 : 50                            db    'P'        ; PRINT
    1525/     7D7 : 48                            db    'H'        ; PRINTHEX
    1526/     7D8 : 49                            db    'I'        ; INPUT
    1527/     7D9 :
    1528/     7D9 : 05 9C               tab_kdo2:   dw    c_LET
    1529/     7DB : 05 B4                         dw    c_PROC
    1530/     7DD : 06 04                         dw    c_GOTO
    1531/     7DF : 06 49                         dw    c_IF
    1532/     7E1 : 06 57                         dw    c_ELSE
    1533/     7E3 : 06 61                         dw    c_RETURN
    1534/     7E5 : 06 7B                         dw    c_GOSUB
    1535/     7E7 : 06 9F                         dw    c_WAIT
    1536/     7E9 : 05 83                         dw    c_REM
    1537/     7EB : 06 B9                         dw    c_CALL
    1538/     7ED : 06 C1                         dw    c_STOP
    1539/     7EF : 06 C5                         dw    c_END
    1540/     7F1 : 05 AE                         dw    c_TRAP
    1541/     7F3 : 06 C9                         dw    c_CLRTRAP
    1542/     7F5 : 06 DD                         dw    c_PRINT
    1543/     7F7 : 06 CE                         dw    c_PRINTHEX
    1544/     7F9 : 07 15                         dw    c_INPUT
```

```
    1545/     7FB :
    1546/     7FB :                     ;-----------------------------------------------
------------------------------
    1547/     7FB :                     ;
    1548/     7FB :                     ;-----------------------------------------------
------------------------------
    1549/     7FB :
    1550/     7FB : FF                          db    0FFh
    1551/     7FC : FF                          db    0FFh
    1552/     7FD :
    1553/     7FD :                     ; Eintrittspunkt BASIC-Interpreter
    1554/     7FD :                     ;RUN
    1555/     7FD : 8D 07 2B                    jp    run
    1556/     800 :
    1557/     800 :                     ; end of 'ROM'
    1558/     800 :
    1559/     800 :                           end
```