Neue Tastaturdekodierung mit Umlauten und CTRL-Taste

Das hier gezeigte Programm ist eine Neuversion der ROM-Routine, die durch RST 1, DATA #12 aufgerufen wird und aus einer gedrückten Taste den zugehörigen Code ermittelt.
Erstens erlaubt sie einen direkten Zugriff auf die Umlaute und das 'ß'. Das kann man natürlich auch durch einfaches Ändern der Tastaturtabelle erreichen, nur daß dann die Umlaute bei Umschaltung von Groß- auf Kleinbuchstaben nicht mit umgeschaltet werden.
Zweitens gibt es aus diesem Grunde die Änderung, daß 'Buchstabe' nicht mehr A..Z (=#41..#5A) ist, sondern A..Ü (=#41 .. #5D). Die Umschaltung groß/klein wird nicht mehr durch CTRL durchgeführt, sondern durch eine beliebige Taste, die in der Tastaturtabelle einen Wert >=#80 besitzt. In diesem Fall ist es SHIFT-CTRL.
Die dritte und wohl wichtigste Änderung ist folgende: durch gleichzeitiges Drücken von CTRL und eines Buchstabens (s.o.) wird der dem Buchstaben zugehörige Code-#40 geliefert (also CTRL-A = 1 .. CTRL-Ü = 29), und zwar gilt dies unabhängig vom eingeschalteten Groß/Klein-Modus.
Da die TAB-Taste wegen der Umlaute entfallen mußte, ist TAB also jetzt durch CTRL-I zu erreichen
Die CTRL-Codes werden zwar vom BASIC ignoriert (bis auf CTRL-H = CHDEL und CTRL-M = RETURN), können aber im EDIT-Mode z.B. in Strings und REM's eingefügt werden. Den größten Vorteil bieten sie natürlich in Programmen.

Die neue Tastatur sieht dann so aus:
SHIFT-0 = 'ß', SHIFT-CHARDEL = '↑', die TAB-Taste wird ersetzt durch die ';+'-Taste, auf SHIFT-RETURN liegt ESC (=#1B), dort kann man aber auch irgendein häufiger gebrauchtes Zeichen wie z.B. '§' (=#40) oder TAB (=9) (wenn CTRL-I zu umständlich ist) hinlegen. ESC ist übrigens auch durch CTRL-Ä zu erreichen.

Der erste Teil des Hex-Listings enthält das Programm. Hier sind alle Adressen, die beim Verschieben geändert werden müssen, unterstrichen. Im zweiten Teil findet sich die neue Tabelle, wobei hier die Bytes unterstrichen sind, die gegenüber der Original-Tabelle geändert sind. Das Abtippen dieses Teils kann man sich also sparen, wenn man in der Utility ME8C5 E934 38A eingibt und dann nur die unterstrichenen Bytes entsprechend ändert.
Wer nur die CTRL-Codes, aber keine Umlaute benötigt, muß im Programmteil #032B von #5E auf #5B ändern und kann die Original- Tabelle benutzen, wobei nur #03BF von #80 auf #00 gesetzt wird.

Gestartet wird das Programm mit CALLM #300 oder G300; dadurch werden RST1-Vektor und der Zeiger auf die Tastaturtabelle gesetzt. Aber Vorsicht: gibt man danach in der UT 'Z3' ein, wird wieder die ROM-Routine benutzt, aber weiterhin die neue Tabelle!

Für Disassembler-Freaks sei noch angemerkt, daß das Programm etwas merkwürdig aussieht, weil es genau in den alten ROM-Platz passen sollte, wo es in meinem DAI auch schon seit über einem halben Jahr zu meiner vollen Zufriedenheit läuft.

```
0300  F3 21 8A 03 22 A7 02 21 73 03 22 64 00 FB C9 F5
0310  C5 D5 E5 3E 07 90 87 87 87 81 2A A7 02 11 40 00
0320  4F 06 00 09 7E FE 41 DA 3A 03 FE 5E D2 3A 03 3A
0330  AE 02 87 FA 6E 03 3A C3 02 47 3A B0 02 A8 87 F2
0340  45 03 1E 38 19 7E B7 CA 62 03 FA 65 03 47 2A BE
0350  02 E5 CD 9C D6 3A C0 02 BD CA 61 03 22 BE 02 E3
0360  70 E1 C3 56 CB 21 C3 02 7E 2F 77 C3 62 03 7E 93
0370  C3 4D 03 E1 E3 F5 7E 23 FE 12 CA 84 03 2B F1 E3
0380  E5 C3 0E C7 F1 E3 FB C3 0F 03
038A                          30 31 32 33 34 35        012345
0390  36 37 38 39 3A 5C 2C 2D 2E 2F 0D 41 42 43 44 45  6789:Ö,-./ ABCDE
03A0  46 47 48 49 4A 4B 4C 4D 4E 4F 50 51 52 53 54 55  FGHIJKLMNOPQRSTU
03B0  56 57 58 59 5A 5B 5D 20 00 08 10 11 12 13 3B 00  VWXYZÄÜ      ;
03C0  00 00 7E 21 22 23 24 25 26 27 28 29 2A 7C 3C 3D    ß!"#$%&'()*ö<=
03D0  3E 3F 1B 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D  >? abcdefghijklm
03E0  6E 6F 70 71 72 73 74 75 76 77 78 79 7A 7B 7D 20  nopqrstuvwxyzäü
03F0  00 5E 14 15 16 17 2B 80 00 00                     ↑    +
```

Software-Random, HL:=RND(65536)

```
0002
0003                ; Dieses Programm ist genau wie das nachfolgende kein
0004                ; eigenstaendiges, was also z.B. von BASIC aufgerufen
0005                ; werden kann. Vielmehr soll es dem Maschinensprache-
0006                ; programmierer (-anfaenger) als Moeglichkeit dienen,
0007                ; Software-Random auf 16bit-Ebene zu erzeugen. Deshalb
0008                ; taucht im Programm auch kein ORG auf.
0009                ; Die hier dargestellte Random-Routine ist mit der ROM-
0010                ; integrierten (BASIC) identisch, beschraenkt sich aber
0011                ; auf 16bit und arbeitet wesentlich 'direkter', was
0012                ; sich durch eine Geschwindigkeitssteigerung von 25 ms
0013                ; auf unter 1 ms pro Berechnung bemerkbar macht.
0014                ; Zur Arbeitsweise : es werden zuerst die alten Random-
0015                ; Delay-Bytes (die den 'fortlaufenden' Zufall speichern)
0016                ; ins HL-Reg.Paar gelesen, dann wird 5mal (Reg.C als
0017                ; Zaehler) HL mit 59 multipliziert und mit einer Konstan-
0018                ; ten addiert. Durch diese Prozedur ergibt sich ein rela-
0019                ; tiv guter Zufallswert. Abschliessend werden die Random-
0020                ; Delay-Bytes erneuert.
0021                ; Am Anschluss an die Routine sind noch Moeglichkeiten
0022                ; fuer eine Reduzierung auf 8bit (Reg.A) angehaengt.
0023
0024                ; Die Routine rettet nicht die Register (PUSH/POP); es
0025                ; werden so AFCDE (und HL) -Werte zerstoert. PUSH/POP
0026                ; muessen also nach Bedarf selbst eingebaut werden.
0027
0028    0E05    RANDOM  MVI   C,5       5maliger Prozess
0029    2A2F01          LHLD  :12F      RND-Delay-Bytes 2+3
0030    54      RANDOML MOV   D,H       DE=Zwischensumme = 1*HL
0031    5D              MOV   E,L
0032    29              DAD   H         HL verdoppelt
0033    EB              XCHG
0034    19              DAD   D         DE=(1+2)*HL
0035    EB              XCHG
0036    29              DAD   H
0037    29              DAD   H         HL verachtfacht
0038    EB              XCHG
0039    19              DAD   D         DE=(1+2+8)*HL
0040    EB              XCHG
0041    29              DAD   H         16facher Anfangs-HL-Wert
0042    EB              XCHG
0043    19              DAD   D         DE=(1+2+8+16)*HL
0044    EB              XCHG
0045    29              DAD   H         32fach ...
0046    19              DAD   D         HL=(1+2+8+16+32)*HL = 59*HL
0047    114159          LXI   D,:5941   HL=HL+:5941
0048    19              DAD   D
0049    0D              DCR   C         Wiederholung ?
0050    C20500          JNZ   RANDOML   => 5x  HL=HL*59+:5941
0051    222F01          SHLD  :12F      Random-Delay-B. erneuern
0052
0053                ; => HL=RND(65536)
0054
0055    7D              MOV   A,L
0056    AC              XRA   H
0057
0058                ; => A=RND(256)
0059
0060                ; Nun ggf. A=RND(x) :
0061
0062    D600            SUI   x         x subtrahieren bis A<0
0063    D22300          JNC   $-2       und dann
0064    2F              CMA             A=-A-1   => A=0..x-1
0065
0066                ; bzw. mit einem Reg.=x : (schneller !)
0067
0068    90              SUB   B         x ueber Reg. (z.B. B) sub.
0069    D22900          JNC   $-1       bis A<0, dann wieder
0070    2F              CMA             A=-A-1   => A=0..x-1
0071
0072                ; An der entsprechenden Stelle  (HL=RND(..), A=RND(..))
0073                ; das abschliessende RET nach Bedarf einfuegen.
0074
0075                                    Bochum, den 22. Juli 1983
0076                                    Autor : Heinrich Tegethoff
```

```
0001           TITL            MPL-Paddle-Abfrage
0002 ; Wie die Random-Routine soll dieses Programm ein An-
0003 ; haltspunkt zum Benutzen von Paddles in Maschinenprog.
0004 ; sein. Die Routine entspricht im wesentlichen der ROM-
0005 ; integrierten, ist aber noch 2-3mal schneller (durch
0006 ; das Fehlen der Math-Umwandlung). Es wird  A=PDL(D)
0007 ; berechnet (die Register BCDE werden geschuetzt).
0008 ; Das Programm besitzt keinen ORG, da es ja Bestandteil
0009 ; eines anderen MP's sein soll. Anschliessend folgt ein
0010 ; Beispiel, wie ein Kreuzpoti die Cursortasten (oben,
0011 ; unten, links, rechts, ggf. Mitte werden in Zahlen um-
0012 ; gewandelt) ersetzen kann.
0013
0014 PADDLE    PUSH  B         BCDE retten
0015           PUSH  D
0016           DI              Interrupts ausschalten
0017           LDA   :40       Bank-switch & PDL-Select
0018           PUSH  PSW       alten :40-Wert speichern
0019           ANI   :F8
0020           ORA   D         bzw. B,C,E,H oder L
0021           ORI   8         PDL D anwaehlen
0022           LXI   B,:FD06   Hardwaregegenstueck von :40
0023           STAX  B         :FD06 with PDL-Nr. D laden
0024           DCR   B         BC=:FC06 = Sound-Command-Register
0025           MVI   A,:30
0026           STAX  B         Sound-Command: 30 = Paddle-Count
0027           LXI   H,65535
0028           SHLD  :FC00     S.-Counter 0 mit 65535 laden
0029           LDA   :FD01     Start Timer (lesen=Trigger)
0030 PADDLEL   XCHG            alten Cntr0-Wert speichern
0031           XRA   A
0032           STAX  B         Sound-Command: 00 = Cntr0 lesen
0033           LHLD  :FC00
0034           CALL  :DE14     alten/neuen Cntr0-Wert vergleichen
0035           JC    PADDLEL   neue<alt => weiterzaehlen
0036           CALL  :DE26     HL:=-HL  => Gemessenen Zeit in HL
0037           MVI   A,:36
0038           STAX  B         Sound-Command: 36 = Cntr. stop
0039           POP   PSW       Alter :40-Wert
0040           INR   B         BC = :FD06
0041           STAX  B         Reset :FD06 ( =(:40) )
0042           EI
0043           XRA   A         A=0
0044           LXI   D,-50     Gem. Zeit (HL) anpassen :
0045           DAD   D         HL:=HL-50
0046           POP   D         Reget BCDE
0047           POP   B
0048           RNC             Zeit zu kurz, A=PDL(D)=0
0049           DCR   A         A=255
0050           INR   H         HL>255 ?
0051           DCR   H
0052           RNZ             Zeit zu gross, A=PDL(D)=255
0053           MOV   A,L
0054           RET             A=0..255
0055
0056 ; Das nachfolgende Beispielprogamm formt die Stellung
0057 ; des Kreuzpotis (die Pdl-Nr'n stehen hier in DE) in
0058 ; eine Zahl um, wie sie von den Cursortasten wieder-
0059 ; gegeben wird :
0060
0061 ; 0/255 ***************** 255/255   0=oben
0062 ;       * *     0     * *
0063 ;       *   *       *   *           1=unten
0064 ;       *    -------    *
0065 ;       * 2  !  4  !  3 *           2=links
0066 ;       *    -------    *
0067 ;       *   *       *   *           3=rechts
0068 ;       * *     1     * *
0069 ; 0/0   ***************** 255/0     4=Mitte  (einstellbar)
0070
0071 ; Die Mitte kann von 0*0 bis 128*128 gross sein. Sie
```

NP 52

```
0072 ; kann z.B. den Stillstand einer Figur bedeuten.
0073 ; Die Prozedur geht davon aus, dass X+Y entweder kleiner
0074 ; oder groesser 255, X-Y kleiner/groesser 0 ist (s.
0075 ; Skizze). Die Mitte, jeweils in 2er-Potenz-Breite/Hoehe,
0076 ; wird erkannt, indem die linke, untere Mittenecke durch
0077 ; SUI SUB in den Ursprung verlegt und dann getestet wird
0078 ; ob X und Y kleiner als die Breite/Hoehe sind. Da man
0079 ; sich auf 0,1,2,4,8,16,32,64 und 128 als B./H. beschraen-
0080 ; ken kann, genuegt A:= X OR Y (Bit-OR).
0081
0082 ; Die Prozedur rettet BCDEHL.
0083 ; D und E als PDL-X, PDL-Y-Nummern
0084 ; A als Ergebnis wie oben
0085
0086 RICHT    PUSH  H
0087          PUSH  D
0088          PUSH  B
0089          CALL  PADDLE   A=PDL(D)  X-Wert
0090          MOV   B,A
0091          MOV   D,E
0092          CALL  PADDLE   A=PDL(E)  Y-Wert
0093          MOV   D,A
0094          SUI   SUB
0095          MOV   C,A
0096          MOV   A,B
0097          SUI   SUB
0098          ORA   C        X OR Y, s.o.
0099          CPI   B./H.
0100          MVI   A,4
0101          JC    MITTE    Mitte, A=4
0102          MOV   A,B
0103          ADD   D        CY=0 => 2/1, CY=1 => 0/3
0104          SBB   A        ''      A=00,  ''    A=FF
0105          CMA
0106          ANI   2        ''      A=2 ,  ''    A=0
0107          MOV   C,A
0108          MOV   A,B
0109          SUB   D        X-Y
0110          MOV   A,C
0111          JC    $+5      X-Y<0 => A=0/2
0112          XRI   3        => A=3/1
0113 MITTE    POP   B
0114          POP   D
0115          POP   H
0116          RET
0117
0118 B./H.    EQU   32       0,1,2,4,8,16,32,64 oder 128
0119 SUB      EQU   (256-B./H.).DIV.2
0120
0121 ; Will man die Mitte dazu benutzen, z.B. unterschied-
0122 ; liche Geschwindigkeiten einzustellen (innen 0-3,
0123 ; aussen 4-7), so kann MVI A,4  JC MITTE  durch
0124          SBB   A
0125          CMA
0126          ANI   4
0127          MOV   E,A
0128 ; ersetzt werden. Nach XRI 3 kommt dann noch ein ORA E.
0129
0130 ; Mit SUB EQU -(B./H.).DIV.2 kann getestet werden, ob
0131 ; gerade eine Ecke angesteuert wird. Dadurch kann das
0132 ; Kreuzpoti nach 8 Richtungen (4 Diagonale) abgefragt
0133 ; werden.
0134
0135                        Bochum, den 22. Juni 1983
0136                        Autor : Heinrich Tegethoff
```

```
0002
0003                ; Mit diesem Programm kann eine Zeit von max. 35 Min.
0004                ; sehr genau gemessen werden (Fehler 0.1%), ohne
0005                ; RST-Vectoren zu benutzen, und dazu noch in Mikrosec. !
0006                ; Besonders wichtig : Die Zaehler laufen auch waehrend
0007                ; eines DI, z.B. bei SAVE oder LOAD !
0008                ; Der Nachteil des Programmes : die benutzten Zaehler
0009                ; sind auch die SOUND-Counter. Deshalb wird die Uhr durch
0010                ; SOUND 1 . . . . oder SOUND 2 . . . . durcheinander ge-
0011                ; bracht oder durch SOUND OFF gestoppt. Das unangenehme
0012                ; daran ist, dass 'BREAK' (auch beim EDIT) ebenfalls ein
0013                ; SOUND OFF beinhaltet.
0014                ; Funktionsweise : beim Aufruf beider Programme (RESET,
0015                ; TIMEGET) wird das Set/Get-Programm auf den Stack ge-
0016                ; bracht. Dies ist notwendig, da Set/Get voellig zeit-
0017                ; synchron laufen muessen und die 48K dynamischen RAM's
0018                ; gegenueber dem statischem Stack-RAM Wait-Zyklen haben.
0019                ; Natuerlich sind auch die Interrupts ausgeschaltet (DI).
0020                ; Beim Setzen (Zeit=0) der Zaehler wird einer mit 0, der
0021                ; andere mit -1 (65536/65535) geladen, der Arbeitsmodus
0022                ; der Zaehler ist : Count-down bis 0, dann weiter mit
0023                ; gespeichertem Wert. So wird bei jedem Durchlauf (65536/
0024                ; 65535..0) die Differenz zwischen beiden Zaehlern um 1
0025                ; groesser. Aus der Differenz und dem tatsaechlichem Stand
0026                ; wird dann bei TIMEGET die Zeit berechnet und in Mikro-
0027                ; sekunden der Variable (INTeger) zugeordnet. Ist die
0028                ; Variable vom Typ FPT, so steht anschliessend der Wert
0029                ; in Sekunden (ohne ihn abzurunden !) in der Variablen.
0030                ; Die Werte fuer RESET und TIMEGET stehen am Ende des
0031                ; Programmes. BASIC-Aufruf der Programme :
0032
0033                ; CALLM RESET          => Zeit = 0
0034                ; CALLM TIMEGET,VAR    => VAR=Zeit  (in us/INT, sek/FPT)
0035
0036      ××0300            ORG   :300      (beliebig)
0037 0300
0038 0300 ××0010  STKERR    EQU   :0010     "STACK OVERFLOW"
0039 0300 ××DA0B  SYNERR    EQU   :DA0B     "SYNTAX ERROR"
0040 0300 ××DA1A  TYPERR    EQU   :DA1A     "TYPE MISMATCH"
0041 0300 ××DE1A  SUB       EQU   :DE1A     HL:=HL-DE
0042 0300 ××DE26  CMHL      EQU   :DE26     HL:=-HL
0043 0300
0044 0300 ××F800  STACK     EQU   :F800     Stack-Bottom
0045 0300 ××FC02  CTR1AD    EQU   :FC02     Adr. Counter 1
0046 0300 ××FC04  CTR2AD    EQU   :FC04     Adr. Counter 2
0047 0300 ××FC07  COMMAND   EQU   :FC07     Adr. Command-Reg.
0048 0300 ××0040  CTR1      EQU   :40       Befehl an Cnt.1
0049 0300 ××0080  CTR2      EQU   :80       Befehl an Cnt.2
0050 0300 ××0004  MODE2     EQU   4         Schreib/Lese-Modus
0051 0300 ××0030  LODCTR    EQU   :30       Laden der Counter
0052 0300 ××0000  REDCTR    EQU   0         Lesen der    ''
0053 0300
0054 0300 C5      RESET     PUSH  B         BC retten (fuer's BASIC)
0055 0301 118A03            LXI   D,SetE    Ende v. Set-Prog.
0056 0304 CD6903            CALL  MOVE      Set-Prog auf den Stack, BC laden
0057 0307 210000            LXI   H,0
0058 030A 11FFFF            LXI   D,-1
0059 030D 3E74              MVI   A,CTR1+LODCTR+MODE2
0060 030F CD00F8            CALL  STACK     Counter mit 65536, 65535 laden
0061 0312 FB                EI              und starten
0062 0313 C1                POP   B         BC zurueck und
0063 0314 C9                RET             zurueck zum BASIC
0064 0315
0065 0315 C5      TIMEGET   PUSH  B         BC retten (fuer's BASIC)
```

```
0066 0316 F5              PUSH  PSW       Variablentyp retten
0067 0317 E6A0            ANI   :A0       Variablentyp testen
0068 0319 FA0BDA          JM    SYNERR    keine Variable angegeben (A=FF)
0069 031C C21ADA          JNZ   TYPERR    String-Variable (A=22/62)
0070 031F E5              PUSH  H         VARPTR retten
0071 0320 119603          LXI   D,GetE    Ende Get-Prog
0072 0323 CD6903          CALL  MOVE      Get-Prog auf den Stack, BC laden
0073 0326 3E40            MVI   A,CTR1+REDCTR  Befehl zum Cnt. 1 lesen
0074 0328 CD00F8          CALL  STACK     Counter-Werte lesen
0075 032B FB              EI
0076 032C EB              XCHG            Umrechnung Cnt.1/2 => 0.5 us
0077 032D E5              PUSH  H         Die Cntr. zaehlen eigentlich
0078 032E 2B              DCX   H         in 0.5 us-Schritten (2 MHz) !
0079 032F CD1ADE          CALL  SUB
0080 0332 D23603          JNC   $+4
0081 0335 2B              DCX   H
0082 0336 EB              XCHG
0083 0337 E1              POP   H
0084 0338 E3              XTHL            HL=VARPTR
0085 0339 7A              MOV   A,D       VAR=Zeit SHR 1 => us-Schritte
0086 033A B7              ORA   A
0087 033B 1F              RAR
0088 033C 77              MOV   M,A
0089 033D 23              INX   H
0090 033E 7B              MOV   A,E
0091 033F 1F              RAR
0092 0340 77              MOV   M,A
0093 0341 23              INX   H
0094 0342 D1              POP   D
0095 0343 1B              DCX   D         )
0096 0344 7A              MOV   A,D
0097 0345 2F              CMA             ) Gehoert noch zur Zeitumrechnung
0098 0346 1F              RAR
0099 0347 77              MOV   M,A
0100 0348 23              INX   H
0101 0349 7B              MOV   A,E
0102 034A 2F              CMA             )
0103 034B 1F              RAR
0104 034C 77              MOV   M,A
0105 034D F1              POP   PSW       A=Variablentyp
0106 034E E610            ANI   :10       INT oder FPT ?
0107 0350 C26303          JNZ   INTeger
0108 0353 2B              DCX   H
0109 0354 2B              DCX   H
0110 0355 2B              DCX   H         HL=VARPTR
0111 0356 E70C            RSTD  4,:C      Wert Var. zum Math-Accu
0112 0358 E74B            RSTD  4,:4B     Umrechnen in FPT
0113 035A E5              PUSH  H
0114 035B 216503          LXI   H,Z1E=6   Konstante 1E-6
0115 035E E706            RSTD  4,6       FPT-Multiplikation
0116 0360 E1              POP   H         => Division durch 1000000
0117 0361 E70F            RSTD  4,:F      Math-Accu in Variable speichern
0118 0363 C1        INTeger POP   B       BC wiederbeKommen
0119 0364 C9              RET             zurueck zum BASIC
0120 0365
0121 0365 6D8637BD  Z1E=6   DT    6D8637BD  Konstante 1E-6
0122 0369
0123 0369 F3        MOVE    DI            Interrupts ausschalten
0124 036A 21F407            LXI   H,-STACK-LEN  Test : genuegend Platz
0125 036D 39                DAD   SP      im Stack ?
0126 036E D21000            JNC   STKERR  "STACK OVERFLOW"
0127 0371 010BF8            LXI   B,STACK+LEN-1
0128 0374 1A        MLOOP   LDAX  D       Programm (Ende=DE) in Stack
0129 0375 02                STAX  B       einspeichern
```

```
0130 0376 1B              DCX   D
0131 0377 0D              DCR   C
0132 0378 F27403          JP    MLOOP
0133 037B 0107FC          LXI   B,COMMAND  BC als Kommandoregisterzeiger
0134 037E C9              RET             vorbereiten
0135 037F
0136 037F           ; Stack-Programme :
0137 037F
0138 037F 02        Set   STAX  B         Kommando 'Cnt1 laden'
0139 0380 2202FC          SHLD  CTR1AD    mit 0 laden
0140 0383 EB              XCHG
0141 0384 C640            ADI   CTR2-CTR1  => A= 'Cnt2 laden'
0142 0386 02              STAX  B         und befehlen
0143 0387 2204FC          SHLD  CTR2AD    mit -1 laden
0144 038A C9        SetE  RET
0145 038B 02        Get   STAX  B         'Cnt1 lesen' befehlen
0146 038C 2A02FC          LHLD  CTR1AD    und durchfuehren
0147 038F EB              XCHG
0148 0390 C640            ADI   CTR2-CTR1  => A= 'Cnt2 lesen'
0149 0392 02              STAX  B         befehlen
0150 0393 2A04FC          LHLD  CTR2AD    und durchfuehren
0151 0396 C9        GetE  RET
0152 0397 **000C    LEN   EQU   Get-Set   Laenge der Stackprogramme
0153 0397
0154 0397 **0300    RESET :=:   :0300     Einsprungadressen vom BASIC
0155 0397 **0315    TIMEGET :=: :0315
0156 0397
0157 0397                                 Bochum, den 28. Juli 1983
0158 0397                                 Autor : Heinrich Tegethoff
```

```
**************************
* S Y M B O L   T A B L E *
**************************

CMHL    DE26   COMMAND FC07   CTR1    0040   CTR1AD  FC02
CTR2    0080   CTR2AD  FC04   Get     038B   GetE    0396
INTeger 0363   LEN     000C   LODCTR  0030   MLOOP   0374
MODE2   0004   MOVE    0369   REDCTR  0000   RESET   0300
STACK   F800   STKERR  0010   SUB     DE1A   SYNERR  DA0B
Set     037F   SetE    038A   TIMEGET 0315   TYPERR  DA1A
Z1E=6   0365
```

Die unterstrichenen Bytes sind ORG-abhaengig (03,:300) :

```
0054 0300 C5 11 8A 03 CD 69 03 21 00 00 11 FF FF 3E 74 CD
0060 0310 00 F8 FB C1 C9 C5 F5 E6 A0 FA 0B DA C2 1A DA E5
0071 0320 11 96 03 CD 69 03 3E 40 CD 00 F8 FB EB E5 2B CD
0079 0330 1A DE D2 36 03 2B EB E1 E3 7A B7 1F 77 23 7B 1F
0092 0340 77 23 D1 1B 7A 2F 1F 77 23 7B 2F 1F 77 F1 E6 10
0107 0350 C2 63 03 2B 2B 2B E7 0C E7 4B E5 21 65 03 E7 06
0116 0360 E1 E7 0F C1 C9 6D 86 37 BD F3 21 F4 07 39 D2 10
0126 0370 00 01 0B F8 1A 02 1B 0D F2 74 03 01 07 FC C9 02
0139 0380 22 02 FC EB C6 40 02 22 04 FC C9 02 2A 02 FC EB
0148 0390 C6 40 02 2A 04 FC C9
```

Bernd Simon

Carl - Schurz - Str. 18

5042 Erftstadt, 11.1o.83

Tabulator - Routine

Da der DAI keine PRINT-USING - Befehle kennt als auch im Zusammenhang mit einem Drucker keine TAB-Funktion, habe ich mir 3 kleine Unterprogramme erstellt, die bei Tabellierungsvorgängen recht nützlich sein können. Insbesondere dann, wenn man Wert darauf legt, daß der Ausdruck auf dem Bildschirm der gleiche ist wie auf dem Papier.

Die zu tabellierenden Größen (Konstanten oder Variablen) sind im Programm wie folgt darzustellen:

Ausdruck in einer Zeile

```
XX TA = <Tabulator Wert 1> : H = <Variable 1> : GOSUB xxxx
XX TA =     ...  etc. ...
XX TA = <Tabulator Wert n> : H = <Variable n> : GOSUB xxxx
XX PRINT   (Abschluß einer Zeile)
```

Ist die zu tabellierende Größe ein String, muß natürlich anstelle von H ein H$ stehen.

Optionen:

GOSUB 1ooo : Numerische Größe linksbündig GOSUB 11oo : Zeichenkette "	Erste Zeichen der Var. erscheinen an den Positionen TA (gezählt wird von ø an).
GOSUB 12oo : Numerische Größe rechtsbündig GOSUB 13oo : Zeichenketten "	Letzte Zeichen der Var. erscheinen an den Positionen TA
GOSUB 14oo : Num.Größe 'Komma unter Komma'	Die Kommata erscheinen an den Positionen TA.

Zur Veranschaulichung eine kleine Demo einschl. der Unterroutinen.

```
1000  REM *** (1)  TABULATOR-ROUTINE , linksbuendig ***
1010  H$=STR$(H)
1100  T$="                                                  "
1110  TA$=LEFT$(T$,TA-CURX)
1120  PRINT TA$;H$;
1130  RETURN
1200  REM *** (2)  TABULATOR-ROUTINE , rechtsbuendig ***
1210  H$=STR$(H)
1300  T$="                                                  "
1310  TA$=RIGHT$(T$+H$,TA-CURX+1)
1320  PRINT TA$;
1330  RETURN
1400  REM ***  (3) TABULATOR-ROUTINE , Komma unter Komma ***
1410  T$="                                                  "
1420  H$=STR$(H)
1430  L=LEN(STR$(INT(ABS(H))))
1440  IF L>8.0 THEN L=L-6.0
1450  TA$=LEFT$(T$,TA-CURX-L+2)
1460  PRINT TA$;H;
1470  RETURN
1500  PRINT
```

```
1     REM ///  Demo zur Tabulator-Routine                          ///
2     REM ///  Die Routine liegt zwischen Z.Nr. 1000 u. 1480       ///
10    PRINT "Linksbuendig :":PRINT
15    TA=10.0:H$="I":GOSUB 1100
20    TA=25.0:H$="(I*I)^4":GOSUB 1100
25    TA=45.0:H$="SQR(SQR(I*I))":GOSUB 1100
30    PRINT
35    TA=10.0:H$="----":GOSUB 1100
40    TA=25.0:H$="----------":GOSUB 1100
45    TA=45.0:H$="----------":GOSUB 1100
50    PRINT
55    FOR I=-8.0 TO 8.0 STEP 2.0
60    A=I:B=(I*I)^4.0:C=SQR(SQR(I*I))
65    TA=10.0:H=A:GOSUB 1000
70    TA=25.0:H=B:GOSUB 1000
75    TA=45.0:H=C:GOSUB 1000
80    PRINT
85    NEXT
90    PRINT "01234567890123456789012345678901234567890123456789"
95    PRINT
100   PRINT
105   PRINT "Rechtsbuendig :"
115   TA=10.0:H$="I":GOSUB 1300
120   TA=25.0:H$="(I*I)^4":GOSUB 1300
125   TA=45.0:H$="SQR(SQR(I*I))":GOSUB 1300
130   PRINT
135   TA=10.0:H$="----":GOSUB 1300
140   TA=25.0:H$="----------":GOSUB 1300
145   TA=45.0:H$="----------":GOSUB 1300
150   PRINT
155   FOR I=-8.0 TO 8.0 STEP 2.0
160   A=I:B=(I*I)^4.0:C=SQR(SQR(I*I))
165   TA=10.0:H=A:GOSUB 1200
170   TA=25.0:H=B:GOSUB 1200
175   TA=45.0:H=C:GOSUB 1200
180   PRINT
185   NEXT
190   PRINT "01234567890123456789012345678901234567890123456789"
195   PRINT
200   PRINT
205   PRINT "Komma unter Komma :"
215   TA=10.0:H$="I":GOSUB 1300
220   TA=30.0:H$="(I*I)^4":GOSUB 1300
225   TA=55.0:H$="SQR(SQR(I*I))":GOSUB 1300
230   PRINT
235   TA=11.0:H$="----":GOSUB 1300
240   TA=32.0:H$="----------":GOSUB 1300
245   TA=55.0:H$="----------":GOSUB 1300
250   PRINT
255   FOR I=-8.0 TO 8.0 STEP 2.0
260   A=I:B=(I*I)^4.0:C=SQR(SQR(I*I))
265   TA=10.0:H=A:GOSUB 1400
270   TA=25.0:H=B:GOSUB 1400
275   TA=45.0:H=C:GOSUB 1400
280   PRINT
285   NEXT
290   PRINT
295   PRINT "01234567890123456789012345678901234567890123456789"
300   GOTO 1500
```

Roland Winde
Bundesstraße 8
2000 Hamburg 13
Tel.: 040/41 86 00

Ausdruck von Maschinen-Programmen im HEX-Format

Viele ML-Programme werden statt im Assembler-Source-Code im HEX-Format abgedruckt. Werden diese Programme dann abgetippt, bleiben besonders bei längeren Programmen Eingabefehler nicht aus. Die Suche nach diesen Tippfehlern kann mitunter länger dauern als die Eingabe selbst.

Eine wesentliche Hilfe wäre es, wenn zu jeder Programm-Zeile eine Prüfsumme abgedruckt würde. Damit ließen sich Fehler leichter lokalisieren.

Folgendes BASIC-Programm übernimmt diese Aufgabe.
Außerdem unterstreicht das Programm alle Bytes, die bei einer Programm-Verschiebung in einen anderen Speicherbereich geändert werden müssen. Auf Wunsch werden auch noch die ASCII-Zeichen der Prg.-Bytes ausgedruckt. Damit lassen sich die evtl. in einem ML-Programm enthaltenen Kommentar- oder Befehls-Texte leicht auffinden.

Einige Bemerkungen zum Programm:

Die Drucker-Befehle beziehen sich auf ein Treiber-Programm, welches ich für meine Silver Reed EX-42 geschrieben habe (Betrieb am DCE-Bus).
Leider lassen sich die Drucker-Funktionen Zeilen-Breite, Zeilen-Schaltung (1-, 1.5-, 2-zeilig), Rand usw. nicht fernsteuern, sodaß ich diese Einstellungen manuell vornehmen muß.
Auch ist die Druck-Geschwindigkeit nicht berauschend.
Diese Nachteile werden aber durch das hervorragende Schriftbild wieder wettgemacht.

```
   IMP INT

10       REM ROLAND WINDE
20       REM BUNDESSTRASSE 8
30       REM 2000 HAMBURG 13
40       REM 12.10.83
50       REM
60       REM AUSDRUCK VON ML-PROGRAMMEN IM HEX-FORMAT
70       REM MIT UNTERSTREICHUNG DER ZU AENDERNDEN
80       REM ADRESSEN UND PRUEFSUMMEN-AUSDRUCK.
90       REM
100       IF PEEK(#2DD)=#C3 AND PEEK(#2DE)=#10 AND PEEK(#2DF)=#90 THEN 20
          0
110       POKE #29B,#2:POKE #29C,#92
120       CLEAR 1000
130       GOSUB 10000
140       OUT (#80),4:REM BAND AUS
150       POKE #2DD,#C3:POKE #2DE,#10:POKE #2DF,#90
200       PRINT CHR$(12):CURSOR 0,22:ENVELOPE 0 15
210       PRINT "     xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
215       PRINT "     x                                              x"
220       PRINT "     x  AUSDRUCK VON ML-PROGRAMMEN IM HEX-FORMAT.  x"
230       PRINT "     x   ZU AENDERNDE ADRESSEN  (BEI PROGRAMM-     x"
240       PRINT "     x    VERSCHIEBUNG) WERDEN UNTERSTRICHEN.      x"
250       PRINT "     x                                              x"
260       PRINT "     x    AUSDRUCK EINER PRUEFSUMME UND DER         x"
270       PRINT "     x  ASCII-ZEICHEN FUER JEDE  PROGRAMMZEILE.    x"
280       PRINT "     x             (AUF WUNSCH)                     x"
290       PRINT "     xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
300       DIM HEXA(30):RESTORE
310       READ MARKE$:IF MARKE$kg"HEXAS" THEN 310
320       FOR I=1 TO 30:READ HEXA(I):NEXT
325       PRINT :PRINT :PRINT "     TYPE SPACE ";
327       A=GETC:IF Akg32.0 THEN 327
328       PRINT CHR$(12):CURSOR 0,18
330       PRINT "GUELTIGER ADRESSEN-BEREICH : #2EC BIS #8FFF"
340       PRINT "ADRESSEN HEXADEZIMAL EINGEBEN (ohne  :  oder  #)."
350       PRINT :PRINT "PROGRAMM-AUSDRUCK  :"
360       CURSOR 22,CURY+1:INPUT "START-ADRESSE ";ADR$:PRINT :GOSUB 2000
370       VON=HEX:IF HEXg0 AND VONg#2EB THEN PRINT :GOTO 400
380       SOUND 0 0 15 0 FREQ(800):CURSOR 37,CURY+1
390       FOR I=1 TO 23:PRINT " ";:NEXT:PRINT :SOUND OFF :GOTO 360
400       CURSOR 22,CURY+1:INPUT "END - ADRESSE ";ADR$:PRINT :GOSUB 2000
410       BIS=HEX:IF HEXg0 AND BISgVON AND BISk#9000 THEN 500
420       SOUND 0 0 15 0 FREQ(800):CURSOR 37,CURY+1
430       FOR I=1 TO 23:PRINT " ";:NEXT:PRINT :SOUND OFF :GOTO 400
500       PRINT :PRINT "IN WELCHEM ADRESSEN-BEREICH SOLL AUF ADRESSEN-AEN
          DERUNG"
510       PRINT "GEPRUEFT WERDEN (DATEN- / PUFFER-SPEICHER ?)":PRINT :PRI
          NT
520       CURSOR 22,CURY+1:INPUT "START-ADRESSE ";ADR$:PRINT :IF ADR$=""
          THEN VON1=0:BIS1=0:GOTO 600
525       GOSUB 2000
530       VON1=HEX:IF HEXg0 AND VON1g#2EB THEN PRINT :GOTO 560
540       SOUND 0 0 15 0 FREQ(800):CURSOR 37,CURY+1
550       FOR I=1 TO 23:PRINT " ";:NEXT:PRINT :SOUND OFF :GOTO 520
560       CURSOR 22,CURY+1:INPUT "END - ADRESSE ";ADR$:PRINT :GOSUB 2000
570       BIS1=HEX:IF HEXg0 AND BIS1gVON1 AND BIS2k#9000 THEN 600
580       SOUND 0 0 15 0 FREQ(800):CURSOR 37,CURY+1
590       FOR I=1 TO 23:PRINT " ";:NEXT:PRINT :SOUND OFF :GOTO 560
```

```
600       PRINT :PRINT "      ASCII-ZEICHEN DRUCKEN (J/N)  ?";:CURSOR CUR
           X-1,CURY:ASCI=0
605       P=GETC:IF P=ASC("J") THEN ASCI=1:GOTO 615
610       IF PkgASC("N") THEN 605
615       PRINT CHR$(12):CURSOR 0,12
620       PRINT "  DRUCKER AN, AUF SEITEN-ANFANG UND 1.5-SCHALTUNG STELLE
           N."
625       PRINT "  DRUCKER AUF 78 ZEICHEN / ZEILE UND 12 PITCH  EINSTELLE
           N."
630       PRINT :PRINT "             SPACE DRUECKEN ";
635       P=GETC:IF Pkg32 THEN 635
640       POKE #131,3:PRINT CHR$(10);:REM SEITENANFANG
645       PRINT CHR$(1);CHR$(78);CHR$(2);CHR$(40);CHR$(3);CHR$(48);
650       ZAEHLER=VON:F=0
655       ADR$=HEX$(VON):SUMME=0
660       IF LEN(ADR$)k4 THEN ADR$="0"+ADR$
670       PRINT ADR$;"  ";
700       P=PEEK(ZAEHLER)
710       IF Fg0 THEN 800
720       FLAG=0:FOR I=1 TO 30
730       IF P=HEXA(I) THEN FLAG=1
740       NEXT
750       IF FLAG=0 THEN 800
760       PRUEF=PEEK(ZAEHLER+1)+PEEK(ZAEHLER+2)x256
770       IF PRUEFg=VON1 AND PRUEFk=BIS1 THEN F=3
800       WERT$=HEX$(P)
810       IF LEN(WERT$)k2 THEN WERT$="0"+WERT$
820       WERT$=WERT$+" "
830       PRINT WERT$;
835       IF F=3 THEN F=F-1:GOTO 850
840       IF Fg0 THEN PRINT CHR$(9);CHR$(9);CHR$(9);"?? ";:F=F-1
850       SUMME=SUMME+P
860       ZAEHLER=ZAEHLER+1:IF ZAEHLERgBIS THEN 970
870       IF ZAEHLER MOD 16g0 THEN 700
880       SPACE=VON MOD 16
890       IF SPACEg0 THEN PRINT "   ";:SPACE=SPACE-1:GOTO 890
895       GOSUB 900:GOTO 655
900       SUMME$=HEX$(SUMME)
910       IF LEN(SUMME$)k4 THEN SUMME$="0"+SUMME$:GOTO 910
920       PRINT "  ";SUMME$;"  ";:IF ASCI=0 THEN 960
930       FOR ASCII=VON TO ZAEHLER-1
940       IF PEEK(ASCII)k32 OR PEEK(ASCII)g#7F THEN PRINT " ";:GOTO 955
950       PRINT CHR$(PEEK(ASCII));
955       NEXT
960       PRINT :VON=ZAEHLER:RETURN
970       IF ZAEHLER MOD 16g0 THEN PRINT "   ";:ZAEHLER=ZAEHLER+1:GOTO 97
           0
980       GOSUB 900
990       PRINT CHR$(2);CHR$(66);CHR$(3);CHR$(72);:POKE #131,1:GOTO 200
1000       DATA HEXAS
1010       REM REGISTER PAIR GROUP
1020       DATA #0A,#1A,#02,#12
1030       REM DIRECT ADRESS GROUP
1040       DATA #2A,#22,#3A,#32
1050       REM JUMP GROUP
1060       DATA #C3,#C2,#CA,#D2,#DA,#E2,#EA,#F2,#FA
1070       REM CALL GROUP
1080       DATA #CD,#C4,#CC,#D4,#DC,#E4,#EC,#F4,#FC
1090       REM IMMEDIATE GROUP
1100       DATA #01,#11,#21,#31
2000       HEX=0:IF ADR$="" OR LEN(ADR$)g4 THEN RETURN
```

```
2010      P=0
2020      FOR I=LEN(ADR$)-1 TO 0 STEP -1
2030      F=ASC(MID$(ADR$,I,1))
2040      IF Fg47 AND Fk58 THEN F1=F-48:GOTO 2100
2050      IF Fg64 AND Fk71 THEN F1=F-55:GOTO 2100
2060      HEX=0:RETURN
2100      HEX=HEX+F1x(#10²P)+0.5
2110      P=P+1
2120      NEXT
2130      RETURN
10000      POKE #62,#5D:POKE #63,#EB
10005      READ MARKE$:IF MARKE$kg"LOAD" THEN 10005
10010      FOR I=#F800 TO #F82D:READ A:POKE I,A:NEXT
10020      PRINT CHR$(12):PRINT "READING ... ";
10040      CALLM #F800:RETURN
10045      DATA LOAD
10050      DATA #C5,#D5,#E5,#F5,#21,#00,#00,#01,#FF,#31,#CD,#CE,#02,#21
10060      DATA #54,#F8,#11,#56,#F8,#CD,#D1,#02,#21,#00,#B0,#EB,#2A,#54
10070      DATA #F8,#DC,#D1,#02,#CD,#D4,#02,#D2,#A8,#D2,#CD,#5E,#DD,#F1
10080      DATA #E1,#D1,#C1,#C9
```

g = > k = < ² = ↑

```
10    REM Demonstration
15    CLEAR 1000
20    INPUT " Bitte geben Sie eine Zahl ein ";EINGABE:PRINT
25    GOSUB 1000
30    PRINT " Die eingegebene Zahl";EINGABE;" entspricht:"
35    PRINT " ";E$:PRINT
40    GOTO 20
1000  REM ************************************************************
1005  REM Konvertierung Zifferndarstellung - Ausgeschriebene Zahl
1010  REM Einsprung: Zahl in der Variablen 'EINGABE'
1015  REM Rueckkehr: String in 'E$'
1020  REM (c) Stefan Goller, Buschweg 14, 5309 Meckenheim-Merl
1025  REM ************************************************************
1030  E=EINGABE-INT(EINGABE/1000)*1000
1035  GOSUB 1085
1040  MERKE$=E$:IF EINGABE<1000 THEN 1060
1045  E=INT(EINGABE/1000)
1050  GOSUB 1085
1055  MERKE$=E$+"tausend"+MERKE$
1060  E$=MERKE$+"xx"
1065  E=ASC(LEFT$(E$,1))-ASC("a")+ASC("A")
1070  E$=CHR$(E)+RIGHT$(E$,LEN(E$)-1)
1075  RETURN
1080  REM ************************************************************
1085  VAR=E
1090  E=VAR-INT(VAR/10)*10:GOSUB 2000:E$=EINER$
1095  E=VAR-INT(VAR/100)*100:E=INT(E/10):GOSUB 3000
1100  E=VAR-INT(VAR/100)*100
1105  IF E>19 AND EINER$<>"" THEN E$=EINER$+"und"+ZEHNER$:GOTO 1135
1110  IF E>19 THEN E$=ZEHNER$:GOTO 1135
1115  IF E>12 THEN E$=EINER$+ZEHNER$:GOTO 1135
1120  IF E=12 THEN E$="zwoelf":GOTO 1135
1125  IF E=11 THEN E$="elf":GOTO 1135
1130  E$=EINER$+ZEHNER$
1135  IF VAR<100 THEN RETURN
1140  E=INT(VAR/100):GOSUB 2000
1145  E$=EINER$+"hundert"+E$
1150  RETURN
2000  REM ************************************************************
2005  ON E+1 GOSUB 2015,2020,2025,2030,2035,2040,2045,2050,2055,2060
2010  RETURN
2015  EINER$="":RETURN
2020  EINER$="ein":RETURN
2025  EINER$="zwei":RETURN
2030  EINER$="drei":RETURN
2035  EINER$="vier":RETURN
2040  EINER$="fuenf":RETURN
2045  EINER$="sechs":RETURN
2050  EINER$="sieben":RETURN
2055  EINER$="acht":RETURN
2060  EINER$="neun":RETURN
3000  REM ************************************************************
3005  ON E+1 GOSUB 3015,3020,3025,3030,3035,3040,3045,3050,3055,3060
3010  RETURN
3015  ZEHNER$="":RETURN
3020  ZEHNER$="zehn":RETURN
3025  ZEHNER$="zwanzig":RETURN
3030  ZEHNER$="dreissig":RETURN
3035  ZEHNER$="vierzig":RETURN
3040  ZEHNER$="fuenfzig":RETURN
3045  ZEHNER$="sechzig":RETURN
3050  ZEHNER$="siebzig":RETURN
3055  ZEHNER$="achtzig":RETURN
3060  ZEHNER$="neunzig":RETURN
```

NP56

Zwei mathematische Spielereien: Die Bestimmung der Eulerschen Zahl e und der Zahl π auf bis zu 1250 Stellen genau.

(Eine Erhöhung der Stellenzahl ist möglich, wenn man die verwendeten eindimensionalen Felder in zweidimensionale verändert: für je 5 Stellen wird ein Integerspeicherplatz benötigt)

Kürzlich las ich, daß es japanischen Mathematikern gelungen ist, π innerhalb weniger Wochen auf zwei, dann vier und schließlich auf über acht Millionen Stellen zu berechnen (Rechenzeit nicht ganz 7 Stunden). Die Arbeiten sollen weitergeführt werden und man rechnet damit, in einigen Monaten 16 Millionen Stellen zu erreichen. Nicht aus reiner Spielerei, sondern um Information über die Struktur transzendenter Zahlen zu erhalten. Kurz und gut, mich interessierte die Übertragung dieses Problems (wenigstens in bescheidenem Rahmen) auf den DAI.

Bestimmung von e: Die Berechnung erfolgt mit der Reihe

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + r_n$$

$$\text{mit } r_n < \frac{n+2}{n+1} \cdot \frac{1}{(n+1)!}$$

Ab Zeile 300 wird entsprechend der gewünschten Stellenanzahl die Anzahl der zu berechnenden Reihenglieder bestimmt (bei 1000 Stellen N=144 Folgenglieder). Die Dezimalstellen werden in einem Integerfeld X berechnet: e=2, 71828 18284 .....

X(0) X(1) X(2) usw.

In diesem Zusammenhang interessant dürfte ein Vergleichstest sein: Bei einem leicht modifizierten Programm (N=144) fest vorgegeben) benötigte für die 1000 Dezimalstellen der DAI ca 41 min, der CBM 8032 über 90 min, der Apple über 70 min und der VC64 über 50 min.

Bestimmung von π:

π wird mit einer Formel berechnet, die von 1706 stammt:

$\pi = 16\arctan\frac{1}{5} - 4\arctan\frac{1}{239}$. Mit zwei ähnlichen Formeln von Gauß und Störmer wurde 1961 π auf 100625 Stellen genau bei 4 h Rechenzeit bestimmt. Für $\arctan\frac{1}{p}$ wird die Reihe

$$\arctan\frac{1}{p} = \frac{1}{p} - \frac{1}{3p^3} + \frac{1}{5p^5} - \frac{1}{7p^7} + \frac{1}{9p^9} - + \ldots + (-1)^n r_n$$

$$\text{mit } r_n < \frac{1}{(2n+1)p^{2n+1}}$$

Auch in diesem Programm wird in Zeile 1300-1320 die Anzahl der zu berechnenden Reihenglieder bestimmt. Leider konvergieren die Reihen nicht so schnell, so daß auch aufgrund der umständlicheren Berechnung mit einer Rechenzeit von etwas über 4h bei 1000 Dezimalen gerechnet werden muß.

Ernst Jäckle
Am Heimgarten 7
7218 Trossingen

e auf 1000 Stellen

2 71828 18284 59045 23536 02874 71352 66249 77572 47093 69995
95749 66967 62772 40766 30353 54759 45713 82178 52516 64274
27466 39193 20030 59921 81741 35966 29043 57290 03342 95260
59563 07381 32328 62794 34907 63233 82988 07531 95251 01901
15738 34187 93070 21540 89149 93488 41675 09244 76146 06680
82264 80016 84774 11853 74234 54424 37107 53907 77449 92069
55170 27618 38606 26133 13845 83000 75204 49338 26560 29760
67371 13200 70932 87091 27443 74704 72306 96977 20931 01416
92836 81902 55151 08657 46377 21112 52389 78442 50569 53696
77078 54499 69967 94686 44549 05987 93163 68892 30098 79312
77361 78215 42499 92295 76351 48220 82698 95193 66803 31825
28869 39849 64651 05820 93923 98294 88793 32036 25094 43117
30123 81970 68416 14039 70198 37679 32068 32823 76464 80429
53118 02328 78250 98194 55815 30175 67173 61332 06981 12509
96181 88159 30416 90351 59888 85193 45807 27386 67385 89422
87922 84998 92086 80582 57492 79610 48419 84443 63463 24496
84875 60233 62482 70419 78623 20900 21609 90235 30436 99418
49146 31409 34317 38143 64054 62531 52096 18369 08887 07016
76839 64243 78140 59271 45635 49061 30310 72085 10383 75051
01157 47704 17189 86106 87396 96552 12671 54688 95703 50354

```
5     CLEAR 2000:POKE #131,1:B=100000
10    PRINT TAB(10);"e auf N Stellen mit N<1251 und N Vielfaches von 5":PRINT
15    PRINT :PRINT :INPUT "Geben Sie N ein";S:PRINT
30    IF S>1250 OR S MOD 5<>0 THEN 15
40    GOSUB 300:G=S/5+1:DIM N$(5),X(G+2):X(0)=1
50    FOR I=1 TO 5:N$(I)=N$(I-1)+"0":NEXT I
70    FOR L=N TO 1 STEP -1
80    FOR I=0 TO G:Q=X(I)/L:R=X(I) MOD L
90    X(I)=Q:X(I+1)=X(I+1)+B*R:NEXT I:X(G+1)=0
100   X(0)=X(0)+1:NEXT L
110   RE!=X(G):X(G-1)=X(G-1)+INT(RE!/B+0.5)
120   FOR I=G-1 TO 1 STEP -1
130   U=X(I)/B:X(I)=X(I) MOD B:X(I-1)=X(I-1)+U
140   NEXT I
150   POKE #131,0:PRINT CHR$(27);"E";CHR$(27);"G";CHR$(27);"D";
160   FOR I=3 TO 83 STEP 8:PRINT CHR$(I);:NEXT I:PRINT CHR$(0)
165   PRINT TAB(7);CHR$(14);"e auf";S;" Stellen"
170   FOR I=0 TO G-1:Z$=STR$(X(I)):Z$=MID$(Z$,1,LEN(Z$)-3)
180   IF I=0 THEN PRINT Z$;:GOTO 210
190   Z$=N$(5-LEN(Z$))+Z$
200   PRINT CHR$(9);Z$;:IF I MOD 10=0 THEN PRINT
210   NEXT I:PRINT :LIST:POKE #131,1:END
300   SU!=0:LL!=0:FOR I=2 TO S+1:L!=LOGT(I):SU!=SU!+L!:F!=L!+SU!-LL!
310   IF F!>S THEN N=I:RETURN
320   LL!=L!:NEXT I
```

Berechnung von e

Pi auf 1000 Stellen

```
3  14159  26535  89793  23846  26433  83279  50288  41971  69399  37510
   58209  74944  59230  78164  06286  20899  86280  34825  34211  70679
   82148  08651  32823  06647  09384  46095  50582  23172  53594  08128
   48111  74502  84102  70193  85211  05559  64462  29489  54930  38196
   44288  10975  66593  34461  28475  64823  37867  83165  27120  19091
   45648  56692  34603  48610  45432  66482  13393  60726  02491  41273
   72458  70066  06315  58817  48815  20920  96282  92540  91715  36436
   78925  90360  01133  05305  48820  46652  13841  46951  94151  16094
   33057  27036  57595  91953  09218  61173  81932  61179  31051  18548
   07446  23799  62749  56735  18857  52724  89122  79381  83011  94912
   98336  73362  44065  66430  86021  39494  63952  24737  19070  21798
   60943  70277  05392  17176  29317  67523  84674  81846  76694  05132
   00056  81271  45263  56082  77857  71342  75778  96091  73637  17872
   14684  40901  22495  34301  46549  58537  10507  92279  68925  89235
   42019  95611  21290  21960  86403  44181  59813  62977  47713  09960
   51870  72113  49999  99837  29780  49951  05973  17328  16096  31859
   50244  59455  34690  83026  42522  30825  33446  85035  26193  11881
   71010  00313  78387  52886  58753  32083  81420  61717  76691  47303
   59825  34904  28755  46873  11595  62863  88235  37875  93751  95778
   18577  80532  17122  68066  13001  92787  66111  95909  21642  01989
```

```
5     CLEAR 5000:POKE #131,1:B=100000
10    PRINT TAB(10);"Pi auf N Stellen mit N<1251 und N Vielfaches von 5":PRINT
15    PRINT :PRINT :INPUT "Geben Sie N ein";S:PRINT
30    IF S>1250 OR S MOD 5<>0 THEN 15
40    P=239:GOSUB 1300:G=S/5+2:DIM N$(5),A(G+1),B(G+1),H(G+1)
50    FOR I=1 TO 5:N$(I)=N$(I-1)+"0":NEXT I
60    H(0)=1:T=P:GOSUB 2000
100   FOR I=0 TO G+1:A(I)=H(I):NEXT I
105   Z=1:IF P=5 THEN P=P*P
110   FOR K=1 TO N:Z=-Z:T=P:GOSUB 2000:IF P=239 THEN GOSUB 2000
120   T=(2*K+1):GOSUB 2000:F=2*K-1:IF F=1 THEN 140
130   FOR I=0 TO G:H(I)=H(I)*F:NEXT I:GOSUB 4000
140   IF Z>0 THEN FOR I=0 TO G:A(I)=A(I)+H(I):NEXT I:GOSUB 4500:GOTO 170
150   FOR I=1 TO G:IF A(I)<H(I) THEN A(I)=A(I)+B:A(I-1)=A(I-1)-1
160   NEXT I:FOR I=0 TO G:A(I)=A(I)-H(I):NEXT I
170   NEXT K:REM ?:?:GOSUB 5000
180   IF FLAG=1 THEN 220
190   FOR I=0 TO G+1:B(I)=A(I):A(I)=0:H(I)=0:NEXT I
200   P=5:GOSUB 1300:FLAG=1:GOTO 60
210   REM PI=4(4A-B)
220   FOR I=0 TO G:A(I)=A(I)*4:NEXT I:GOSUB 4500
230   FOR I=1 TO G:IF A(I)<B(I) THEN A(I)=A(I)+B:A(I-1)=A(I-1)-1
240   NEXT I:FOR I=0 TO G:A(I)=A(I)-B(I):NEXT I
250   FOR I=0 TO G:A(I)=A(I)*4:NEXT I:GOSUB 4500
260   RE!=A(G):A(G-1)=A(G-1)+INT(RE!/B+0.5)
270   G=G-1:GOSUB 4500
400   POKE #131,0:PRINT CHR$(27);"E";CHR$(27);"G";CHR$(27);"D";
410   FOR I=3 TO 83 STEP 8:PRINT CHR$(I);:NEXT I:PRINT CHR$(0)
420   PRINT TAB(7);CHR$(14);"Pi auf";S;" Stellen":PRINT
430   FOR I=0 TO G-1:Z$=STR$(A(I)):Z$=MID$(Z$,1,LEN(Z$)-3)
440   IF I=0 THEN PRINT Z$;:GOTO 470
450   Z$=N$(5-LEN(Z$))+Z$
460   PRINT CHR$(9);Z$;:IF I MOD 10=0 THEN PRINT
470   NEXT I:PRINT :LIST:POKE #131,1:END
1300  L!=LOGT(P):FOR I=5 TO S+10:F=2*I+1
1310  IF LOGT(F)+F*L!>S+10 THEN N=I:RETURN
1320  NEXT I
2000  FOR I=0 TO G:Q=H(I)/T:R=H(I) MOD T
2010  H(I)=Q:H(I+1)=H(I+1)+B*R:NEXT I:H(G+1)=0:RETURN
4000  FOR I=G TO 1 STEP -1
4130  U=H(I)/B:H(I)=H(I) MOD B:H(I-1)=H(I-1)+U
4140  NEXT I:RETURN
4500  FOR I=G TO 1 STEP -1:U=A(I)/B:A(I)=A(I) MOD B
4510  A(I-1)=A(I-1)+U:NEXT I:RETURN
```

NP 57,2

```
0000                    TITL    'DCR-STEUERUNG UND PRINTER-SPOOLER'
0000
0000                ;
0000                ; DISASSEMBLIERT FUER CLUBZEITUNG  BP 1/84
0000                ;
0000                ; BEDIENUNG DER PROGRAMM-TEILE:
0000                ;
0000                ; 1. CALLM REW% (HIER #2F8):
0000                ;    DIE DCR-CASSETTE IN DER AUSGEWAEHLTEN DCR WIRD
0000                ;    INTERRUPT-GETRIEBEN AN DEN ANFANG ZURUECKGESPULT,
0000                ;    D.H. DAS BASIC-PROGRAMM KANN NACH DEM AUFRUF WEITER-
0000                ;    LAUFEN. WIRD DIE (EVTL. NOCH NICHT GANZ RUECKGESPUL-
0000                ;    TE) KASSETTE DANN GEBRAUCHT: 'CALLM #F000:REM REW'.
0000                ; 2. CALLM SKIP% (HIER #2FB):
0000                ;    HIER GILT DAS GLEICHE WIE BEI REW%, NUR DASS DIE
0000                ;    KASSETTE BIS GANZ ANS ENDE GESPULT WIRD. NACH DEM
0000                ;    WENDEN DANN SICHERHEITSHALBER NOCH 'CALLM #F000:
0000                ;    REM REW' AUFRUFEN (FALLS SIE VOR DEM WENDEN NOCH
0000                ;    NICHT AM ENDE WAR).
0000                ; 3. POKE #CB,0:CALLM LOOKNEW%:IF PEEK(#CB)=#FF THEN ...:
0000                ;    HIER GESCHIEHT DAS GLEICHE WIE BEI 'CALLM #F000:
0000                ;    REM LOOK', D.H. DER FILENAME WIRD AN DER CURSOR-
0000                ;    POSITION AUF DEN SCHIRM GESCHRIEBEN. IST JEDOCH
0000                ;    KEIN FILE MEHR VORHANDEN, SO WIRD NICHT ZURUECK-
0000                ;    GESPULT, SONDERN (#CB) AUF #FF GESETZT.
0000                ; 4. IF PEEK(#63)=#EB THEN CALLM INITSPOOLER%
0000                ;    (FALLS DER SPOOLER SCHON ODER NOCH LAEUFT, IST
0000                ;    PEEK(#63)<>#EB.) NACH DEM START WIRD DER INHALT DES
0000                ;    EDIT-PUFFERS INTERRUPT-GETRIEBEN AN DIE RS232
0000                ;    AUSGEGEBEN, WOBEI NACH JEDEM CR (=#0D) ZUSAETZLICH
0000                ;    EIN LINE FEED (=#0A) ANGEFUEGT WIRD. DAS ENDE DES
0000                ;    PUFFERS MUSS DURCH #FF GEKENNZEICHNET SEIN; BEI
0000                ;    ERREICHEN DIESES ZEICHENS WIRD WIEDER AM ANFANG
0000                ;    BEGONNEN (WO JETZT NATUERLICH ETWAS ANDERES STEHEN
0000                ;    KANN!). TRIFFT DER SPOOLER AUF DAS ZEICHEN #FE, SO
0000                ;    SCHALTET ER SICH SELBST AB (TEXTENDE).
0000                ;    VOM BASIC WIRD IN DEN PUFFER MITTELS SETZEN DER
0000                ;    POINTER UND 'POKE #131,2' GEDRUCKT. GENAUES SIEHE
0000                ;    NP44 ZEILEN 740-860 UND 1110-1140 !
0000                ;
0000                ; BENUTZTE RAM-ADRESSEN:
0000                ;
0000  §=005F IMASKM  EQU     5FH         ;INTERRUPT-MASKE IM RAM
0000  §=0062 IVEKT0  EQU     62H         ;INTR0-VEKTOR-SPEICHER
0000  §=006E IVEKT6  EQU     6EH         ;INTR6-VEKTOR-SPEICHER
0000  §=00A2 ED.BUF  EQU     0A2H        ;ZEIGER ANFANG ED-BUFFER
0000  §=011F DCR.NR  EQU     11FH        ;NUMMER DER DCR (0..3)
0000  §=0131 OTSW    EQU     131H        ;AUSGABE-SCHALTER
0000                ;
0000                ; ROM- UND I/O-ADRESSEN:
0000                ;
0000  §=C14D POPRET  EQU     0C14DH      ;POP REGISTER, RETURN
0000  §=D578 KBINT   EQU     0D578H      ;TASTATUR-INTR-ROUTINE
0000  §=D98F SND.DI  EQU     0D98FH
0000  §=EB5D UTRST0  EQU     0EB5DH      ;UTILITY-INTR-ROUTINE
0000  §=FD00 PORI    EQU     0FD00H      ;EINGABE-PORT
0000  §=FE00 PORT.A  EQU     0FE00H      ;PORT A DES 8255
0000  §=FE01 PORT.B  EQU     0FE01H      ;PORT B DES 8255
0000  §=FE03 C.8255  EQU     0FE03H      ;KOMMANDO-REGISTER 8255
0000  §=FFF3 STATUS  EQU     0FFF3H      ;STATUS-REGISTER 5501
0000  §=FFF6 SERBUF  EQU     0FFF6H      ;AUSGABE AN RS232 HIER
0000  §=FFF8 IMASK   EQU     0FFF8H      ;INTR-MASKE IM 5501
0000  §=FFF9 TIMER1  EQU     0FFF9H
0000                ;
0000                ; PROGRAMM-VARIABLE:   (DIESES PROGRAMMS)
```

DCR-STEUERUNG UND PRINTER-SPOOLER

```
0000                    ;
0000  §=00C8  PRTPTR    EQU     0C8H            ;ZEIGER AUF NAECHST. ZEICH
0000  §=00CA  LF.FLG    EQU     0CAH            ;FLAG FUER CR->LINE FEED
0000  §=00CB  OKFLAG    EQU     0CBH            ;FLAG F. LOOKNEW: #FF=ENDE
0000                    ;
0000                    ;
0000                    ORG     2ECH            ;START BELIEBIG
02EC                    ;
02EC  C3C603  INTR6     JMP     SPULE           ;INTR6 SPULT DCR
02EF  C35603  INTR0     JMP     DRUCKE          ;INTR0 DRUCKT ED-PUFFER
02F2  C3FE02            JMP     LOKNEW          ;CALLM LOOKNEW%
02F5  C33003            JMP     INITSP          ;CALLM INITSPOOLER%
02F8  C3AD03            JMP     REW             ;CALLM REW%
02FB  C3A203            JMP     SKIP            ;CALLM SKIP%
02FE                    ;
02FE                    ;
02FE                    ; KOPIE DER 'LOOK'-ROUTINE DES TOS, MIT DEM UNTERSCHIED,
02FE                    ; DASS NACH DEM LETZTEN FILE ODER BEI KASSETTEN-ENDE
02FE                    ; NICHT ZURUECKGESPULT, SONDERN NUR 'OKFLAG' (=#CB)
02FE                    ; AUF #FF GESETZT WIRD.
02FE                    ;
02FE  F5      LOKNEW    PUSH PSW
02FF  C5                PUSH B
0300  D5                PUSH D
0301  E5                PUSH H
0302  210000            LXI H       0H
0305  01FF00            LXI B       0FFH
0308  CD0E03            CALL        LH030E
030B  C374F7            JMP         0F774H
030E                    ;
030E  F5      LH030E    PUSH PSW
030F  5E                MOV E,M
0310  23                INX H
0311  AF                XRA A
0312  57                MOV D,A
0313  CD48F3  LH0313    CALL        0F348H
0316  C21303            JNZ         LH0313
0319  F1                POP PSW
031A  CD00F8            CALL        STACK           ;ZEITKRITISCHE ROUTINE
031D  C3F8F1            JMP         0F1F8H          ;WEITER IN 'LOOK'
0320                    ; 'NOTAUSGANG' BEI KASSETTEN-ENDE:
0320  CD1FF3  LH0320    CALL        0F31FH          ;DCE-BUS ABSCHALTEN
0323  21CB00            LXI H       OKFLAG
0326  36FF              MVI M       0FFH            ;FLAG SETZEN
0328  210600            LXI H       6H
032B  39                DAD SP                      ;2 'CALL' UND
032C  F9                SPHL                        ;1 'PUSH' VERNICHTEN
032D  C34DC1            JMP         POPRET          ;ZURUECK INS BASIC
0330                    ;
0330                    ; EINSCHALTEN DES PRINTER-SPOOLERS:
0330                    ;
0330  2AA200  INITSP    LHLD        ED.BUF          ;POINTER AUF ANFANG DES
0333  22C800            SHLD        PRTPTR          ;EDIT-BUFFERS SETZEN
0336  21EF02            LXI H       INTR0           ;INTERRUPT-VEKTOR
0339  226200            SHLD        IVEKT0          ;UMLEITEN
033C  21F9FF            LXI H       TIMER1          ;INTERRUPT-TIMER
033F  3680              MVI M       80H             ;STARTEN
0341  F3                DI
0342  F5                PUSH PSW
0343  3A5F00            LDA         IMASKM          ;INTERRUPT 0  ERLAUBEN
0346  F601              ORI         1H
0348  325F00            STA         IMASKM
034B  32F8FF            STA         IMASK
034E  F1                POP PSW
```

```
034F  FB              EI
0350  213101          LXI H     OTSW            ;AUSGABE NUR AUF SCHIRM
0353  3601            MVI M     1H
0355  C9              RET
0356                  ;
0356                  ; INTERRUPT-ROUTINE (RST0); GIBT, WENN MOEGLICH DAS
0356                  ; NAECHSTE ZEICHEN DES PUFFERS AN DIE RS232 AUS
0356                  ;
0356  F5        DRUCKE  PUSH PSW
0357  3A00FD            LDA       PORI            ;DRUCKER BUSY?
035A  E608              ANI       8H
035C  CA6703            JZ        LH0367          ;JA, ALSO NICHT AUSGEBEN
035F  3AF3FF            LDA       STATUS          ;5501 BEREIT?
0362  E610              ANI       10H
0364  C27903            JNZ       LH0379          ;JA, ALSO AUSGEBEN
0367                  ;
0367  21F9FF  LH0367    LXI H     TIMER1          ;TIMER NEU LADEN
036A  3680              MVI M     80H
036C  FB        LH036C  EI                        ;UND ZURUECK VOM INTR
036D  F1                POP PSW
036E  E1                POP H
036F  C9                RET
0370                  ;
0370                  ; ABSCHALTEN DER SPOOLER-ROUTINE:
0370  215DEB  LH0370    LXI H     UTRST0          ;ALTEN INTR-VEKTOR
0373  226200            SHLD      IVEKT0          ;WIEDERHERSTELLEN
0376  C36C03            JMP       LH036C          ;UND RAUS
0379                  ;
0379                  ; AUSGABE EINES ZEICHENS:
0379                  ;
0379  21CA00  LH0379    LXI H     LF.FLG          ;MUSS ERST EIN LINE-FEED
037C  7E                MOV A,M                   ;AUSGEGEBEN WERDEN (NACH
037D  B7                ORA A                     ;VORHERIGEN 'RETURN')?
037E  CA8803            JZ        LH0388          ;NEIN, NAECHSTES ZEICHEN
0381  3E0A              MVI A     0AH             ;JA, 'LF' AUSGEBEN UND
0383  3600              MVI M     0H              ;FLAG ZURUECKSETZEN
0385  C39403            JMP       LH0394
0388                  ;
0388  2AC800  LH0388    LHLD      PRTPTR          ;NAECHSTES ZEICHEN AUS
038B  7E                MOV A,M                   ;PUFFER HOLEN
038C  B7                ORA A                     ;SPEZIALZEICHEN ?
038D  FADE03            JM        LH03DE          ;JA, ALSO WELCHES?
0390  23                INX H                     ;ZEIGER ERHOEHEN
0391  22C800            SHLD      PRTPTR          ;UND MERKEN
0394  32F6FF  LH0394    STA       SERBUF          ;ZEICHEN AUSGEBEN
0397  FE0D              CPI       0DH             ;='RETURN' ?
0399  C26703            JNZ       LH0367          ;NEIN, ALLES OK, ZURUECK
039C  32CA00            STA       LF.FLG          ;JA, FLAG SETZEN
039F  C36703            JMP       LH0367          ;UND ZURUECK
03A2                  ;
03A2                  ; INTERRUPT-VOR- UND ZURUECKSPULEN DER DCR:
03A2                  ;
03A2  2103FE  SKIP      LXI H     C.8255          ;KASSETTE BIS ENDE SPULEN
03A5  3698              MVI M     98H
03A7  2B                DCX H
03A8  3602              MVI M     2H              ;=VORWAERTS-BIT
03AA  C3B503            JMP       LH03B5
03AD                  ;
03AD  2103FE  REW       LXI H     C.8255          ;KASSETTE BIS ANF. SPULEN
03B0  3698              MVI M     98H
03B2  2B                DCX H
03B3  3601              MVI M     1H              ;=RUECKWAERTS-BIT
03B5                  ;                                          NP58,2
03B5  3A1F01  LH03B5    LDA       DCR.NR          ;DCE-ADRESSE
```

```
03B8  87              ADD A                    ;FUER AUSGEWAEHLTE
03B9  87              ADD A                    ;DCR (0..3)
03BA  F603            ORI       3H             ;BILDEN UND
03BC  3201FE          STA       PORT.B         ;ANLEGEN.
03BF  21EC02          LXI H     INTR6          ;INTERRUPT
03C2  226E00          SHLD      IVEKT6         ;UMLEITEN.
03C5  C9              RET                      ;ZURUECK INS BASIC
03C6            ;
03C6            ; INTERRUPT-ROUTINE FUER SPULEN:
03C6            ;
03C6  F5        SPULE PUSH PSW
03C7  3A00FE          LDA       PORT.A         ;TEST, OB CASSETTE NOCH
03CA  E622            ANI       22H            ;EINGELEGT UND NOCH
03CC  EE02            XRI       2H             ;LAEUFT.
03CE  CADA03          JZ        LH03DA         ;JA, NICHTS TUN
03D1  CD1FF3          CALL      0F31FH         ;NEIN, DCR ABSCHALTEN
03D4  2178D5          LXI H     KBINT          ;UND ALTEN INTERRUPT
03D7  226E00          SHLD      IVEKT6         ;WIEDERHERSTELLEN.
03DA  F1        LH03DA POP PSW
03DB  C378D5          JMP       KBINT          ;WEITER MIT KEYB.-INTR.
03DE            ;
03DE            ; FORTS. DRUCK-ROUTINE:   ZEICHEN IM PUFFER IST >=#80
03DE            ;
03DE  3C        LH03DE INR A                   ;IST ES #FF(=PUFFERENDE) ?
03DF  C27003          JNZ       LH0370         ;NEIN, ALSO 'TEXTENDE'
03E2  2AA200          LHLD      ED.BUF         ;JA, ALSO ZEIGER WIEDER
03E5  22C800          SHLD      PRTPTR         ;AUF ANFANG DES PUFFERS
03E8  C36703          JMP       LH0367         ;UND WEITER
03EB            ;
03EB            ;
03EB            ; UNTERPROGRAMM FUER 'LOOKNEW', MUSS AUS ZEITGRUENDEN
03EB            ; IM STACK LAUFEN. (IST AUCH EINE FAST-KOPIE DES TOS.)
03EB            ;
03EB                  ORG       0F800H         ;MUSS IM STACK LAUFEN!!
F800            ;
F800  C5        STACK PUSH B
F801  CD8FD9          CALL      SND.DI
F804  CDB4F3 LHF804   CALL      0F3B4H
F807  FB     LHF807   EI
F808  CDBBF3          CALL      0F3BBH
F80B  F3              DI
F80C  C22003          JNZ       LH0320         ;KASS. WEG OD. ENDE, STOP
F80F  CD79F3          CALL      0F379H
F812  060F            MVI B     0FH
F814  CD94F4          CALL      0F494H
F817  C207F8          JNZ       LHF807
F81A  0696            MVI B     96H
F81C  CD94F4          CALL      0F494H
F81F  E680            ANI       80H
F821  CA2003          JZ        LH0320         ;NICHTS MEHR DRAUF, STOP
F824  CDD5F3          CALL      0F3D5H
F827  DA04F8          JC        LHF804
F82A  CDA3F4          CALL      0F4A3H
F82D  CDE0F3          CALL      0F3E0H
F830  DA04F8          JC        LHF804
F833  C1              POP B                    ;NORMALER AUSGANG, FILE-
F834  C9              RET                      ;NAME AUF SCHIRM
F835            ;
F835                  END
```

BASIC-Lader für DCR-Steuerung

```
1     REM !!!!!!!!!!  IMP INT  !!!!!!!!!!!!!!!!!!!!!!
10    REM *** Lader f. DCR-Steuerung u. Printer-Spooler ***
11    REM *** Dies sin Teile von NP44 !!!!!!!!
12    REM .. Das ML-Programm wird unterhalb des Heap und im
13    REM .. Stack abgelegt. Bedienung siehe Assembler-Listing!!
20    REM /////// memory management
30    IF PEEK(7)=0 THEN CLEAR 10512:POKE #29C,PEEK(#29C)+2:POKE #29E,PEEK(#29E)-2:POKE
7,#FF
40    CLEAR 10000:START=PEEK(#29B)+(PEEK(#29C)-2)*256
50    POKE #131,1:POKE #C8,0:POKE #C9,0:POKE #62,#5D:POKE #63,#EB
90    GOSUB 1210
100   REM ***  Hier Hauptprogramm  ****************
999   END
1210  REM //////// parameter setting
1220  GOTO 1330
1230  ADRES=START
1240  READ BYTE:IF BYTE<#100 THEN POKE ADRES,BYTE:ADRES=ADRES+1:GOTO 1240
1250  IF BYTE=#FFF THEN RETURN
1260  READ LBYTE,HBYTE:POKE ADRES,BYTE IAND #FF
1270  ON BYTE/#100 GOTO 1280,1300
1280  LBYTE=LBYTE+(START IAND #FF):CARRY=LBYTE SHR 8
1290  POKE ADRES+1,LBYTE IAND #FF:HBYTE=HBYTE+(START SHR 8)+CARRY:GOTO 1320
1300  LBYTE=LBYTE+(STARTB IAND #FF):CARRY=LBYTE SHR 8
1310  POKE ADRES+1,LBYTE IAND #FF:HBYTE=HBYTE+(STARTB SHR 8)+CARRY
1320  POKE ADRES+2,HBYTE:ADRES=ADRES+3:GOTO 1240
1330  STARTB=START:GOSUB 1230:INITSPOOLER=START+#44:REW=START+#C:SKIP=START+#F
1340  LOOKNEW=START+6:START=#F800:GOSUB 1230
1540  RETURN
1550  REM ----------------MLPDATA---------------------
1560  DATA #1C3,#DA,#0,#1C3,#6A,#0,#1C3,#12,#0,#1C3,#44,#0,#1C3,#C1,#0,#1C3
1570  DATA #B6,#0,#F5,#C5,#D5,#E5,#21,#0,#0,#1,#FF,#0,#1CD,#22,#0,#C3
1580  DATA #74,#F7,#F5,#5E,#23,#AF,#57,#CD,#48,#F3,#1C2,#27,#0,#F1,#CD,#00
1590  DATA #F8,#C3,#F8,#F1,#CD,#1F,#F3,#21,#CB,#0,#36,#FF,#21,#6,#0,#39
1600  DATA #F9,#C3,#4D,#C1,#2A,#A2,#0,#22,#C8,#0,#121,#3,#0,#22,#62,#0
1610  DATA #21,#F9,#FF,#36,#80,#F3,#F5,#3A,#5F,#0,#F6,#1,#32,#5F,#0,#32
1620  DATA #F8,#FF,#F1,#FB,#21,#31,#1,#36,#1,#C9,#F5,#3A,#0,#FD,#E6,#8
1630  DATA #1CA,#7B,#0,#3A,#F3,#FF,#E6,#10,#1C2,#8D,#0,#21,#F9,#FF,#36,#80
1640  DATA #FB,#F1,#E1,#C9,#21,#5D,#EB,#22,#62,#0,#1C3,#80,#0,#21,#CA,#0
1650  DATA #7E,#B7,#1CA,#9C,#0,#3E,#A,#36,#0,#1C3,#A8,#0,#2A,#C8,#0,#7E
1660  DATA #B7,#1FA,#F2,#0,#23,#22,#C8,#0,#32,#F6,#FF,#FE,#D,#1C2,#7B,#0
1670  DATA #32,#CA,#0,#1C3,#7B,#0,#21,#3,#FE,#36,#98,#2B,#36,#2,#1C3,#C9
1680  DATA #0,#21,#3,#FE,#36,#98,#2B,#36,#1,#3A,#1F,#1,#87,#87,#F6,#3
1690  DATA #32,#1,#FE,#121,#0,#0,#22,#6E,#0,#C9,#F5,#3A,#0,#FE,#E6,#22
1700  DATA #EE,#2,#1CA,#EE,#0,#CD,#1F,#F3,#21,#78,#D5,#22,#6E,#0,#F1,#C3
1710  DATA #78,#D5,#3C,#1C2,#84,#0,#2A,#A2,#0,#22,#C8,#0,#1C3,#7B,#0,#FFF
1720  DATA #C5,#CD,#8F,#D9,#CD,#B4,#F3,#FB,#CD,#BB,#F3,#F3,#2C2,#34,#0,#CD
1730  DATA #79,#F3,#6,#F,#CD,#94,#F4,#1C2,#07,#0,#6,#96,#CD,#94,#F4,#E6
1740  DATA #80,#2CA,#34,#0,#CD,#D5,#F3,#1DA,#4,#0,#CD,#A3,#F4,#CD,#E0,#F3
1750  DATA #1DA,#4,#0,#C1,#C9,#FFF
```

§ = $ bzw #

§ = $ bzw #

```
002                 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
003                 x                              Berlin, den 8.12.1983  x
004                 x                                                     x
005                 x Dieses Programm dient zum Schreiben und Lesen       x
006                 x von Speicherinhalten auf das bzw. vom Band vom      x
007                 x Basic aus,oder auch von Maschinenprogrammen aus.    x
008                 x                                                     x
009                 x Schreibeprogramm:                                   x
010                 x Im Basic wird zumaechst durch ein ) INPUT NAME§:    x
011                 x POKE§13F,LEN(NAME§) ( ein eventueller Name,unter    x
012                 x dem das Programm abgespeichert werden soll, in      x
013                 x den input-buffer befoerdert.Der Name der Textva-    x
014                 x riablen ist dann ohne Belang.Sie wird nicht mehr    x
015                 x gebraucht. Bei Aufrufen von Maschinenprogrammen     x
016                 x aus, ist dafuer zu sorgen,dass ein evtl. Name       x
017                 x in den input-buffer geschrieben wird,oder dieser    x
018                 x geloescht wird. Der input-buffer hat die Form:      x
019                 x §13E: §19 , §13F: Laenge des Namens ,               x
020                 x §140 - §1BD : Name                                  x
021                 x Anschliessend mird die Anfangsadresse des abzu-     x
022                 x speichernden Bereichs in die mit ADR1 bezeichne-    x
023                 x te Stelle eingepoked (low-byte zuerst). Ebenso      x
024                 x wird beim Einpoken der Endadresse in ADR2 ver-      x
025                 x fahren. Dann wird das Programm mit CALLM WRITE      x
026                 x gestartet. Der Wert von WRITE ist der Symbolta-     x
027                 x belle zu entnehmen.                                 x
028                 x                                                     x
029                 x Leseprogramm: Ein evtl. Programmname wird wie       x
030                 x schon oben beschrieben in den input-buffer ge-      x
031                 x schrieben. Danach wird der Offset fuer das Ein-     x
032                 x lesen (gleiche Bedeutung wie bei R XXXX in der      x
033                 x Utility) in OFFSET wie oben beschrieben einge-      x
034                 x poked. Das Programm wird mit CALLM READ gestar-     x
035                 x tet. READ ist in der Symboltabelle zu finden.       x
036                 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
037
038                 xxx SCHREIBEN EINES SPEICHRBEREICHS AUF'S BAND: xxx
039                 x Deklarationen:
040                 INBUF   EQU   :013F     Startadresse omput-buffer
041                 BSW1    EQU   :0040     Bankswitchadresse 1
042                 BSW2    EQU   :FD06     Bankswitchadresse 2
043                 BANK0   EQU   :30       code fuer Bank 0
044                 BANK3   EQU   :F0       code fuer Bank 3
045
046                         ORG   :0400     Programmbeginn WRITE
047 0400 F5         WRITE   PUSH  PSW
048 0401 E5                 PUSH  H
049 0402 C5                 PUSH  B
050 0403 D5                 PUSH  D
051 0404 3EF0               MVI   A,BANK3   schalte auf Bank 3
052 0406 324000             STA   BSW1
053 0409 3206FD             STA   BSW2
054 040C 211F04             LXI   H,FORT1   Adresse fuer Fortsetzen
```

NP59

```
PAGE 02    WRITE & READ    R.Kietzmann

055 040F E5                PUSH  H          nach Schreiben uebergeben
056 0410 2A2E04            LHLD  ADR2       lade Endadresse
057 0413 EB                XCHG             nach DE,
058 0414 2A2C04            LHLD  ADR1       Anfangsadresse nach HL
059 0417 E5                PUSH  H          Anfangs- und
060 0418 D5                PUSH  D          Endadresse uebergeben
061 0419 213F01            LXI   H,INBUF    Start input-buffer
062 041C C3F0EE            JMP   :EEF0      Aufruf WRITE im ROM
063 041F 3E30       FORT1  MVI   A,BANK0    schalte auf Bank 0
064 0421 324000            STA   BSW1
065 0424 3206FD            STA   BSW2
066 0427 D1                POP   D
067 0428 C1                POP   B
068 0429 E1                POP   H
069 042A F1                POP   PSW
070 042B C9                RET
071 042C            ADR1   RES   2          hier Anfangsadresse
072 042E            ADR2   RES   2          hier Endadresse einpoken
073
074                xxx LESEN EINES SPEICHERBEREICHS VOM BAND: xxx
075
076 0430 F5         READ   PUSH  PSW        Programmbeginn READ
077 0431 E5                PUSH  H
078 0432 C5                PUSH  B
079 0433 D5                PUSH  D
080 0434 3EF0              MVI   A,BANK3    schalte Bank 3
081 0436 324000            STA   BSW1
082 0439 3206FD            STA   BSW2
083 043C 214A04            LXI   H,FORT2    Adresse fuer Fortsetzen
084 043F E5                PUSH  H          nach Lesen uebergeben
085 0440 2A5704            LHLD  OFFSET     lade Offset nach HL
086 0443 E5                PUSH  H          Offset uebergeben
087 0444 213F01            LXI   H,INBUF    Startadresse input-buffer
088 0447 C317EF            JMP   :EF17      Aufruf READ im ROM
089 044A 3E30       FORT2  MVI   A,BANK0    schalte Bank 0
090 044C 324000            STA   BSW1
091 044F 3206FD            STA   BSW2
092 0452 D1                POP   D
093 0453 C1                POP   B
094 0454 E1                POP   H
095 0455 F1                POP   PSW
096 0456 C9                RET
097 0457            OFFSET RES   2          Hier Offset einpoken
098 0459                   END

xxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxx Symbol  Tabelle xxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxx

ADR1   042C   ADR2   042E   BANK0  0030   BANK3  00F0
BSW1   0040   BSW2   FD06   FORT1  041F   FORT2  044A
INBUF  013F   OFFSET 0457   READ   0430   WRITE  0400
```

FEHLER !!!!!!!!!!!!!!!!!!!

Korrektur zu NP 49,4 :
550 TEXT$="✱"+HEX$(...

FEHLER !!!!!!!!!!!!!!!!!!!

```
0400 F5 E5 C5 D5 3E F0 32 40 00 32 06 FD 21 1F 04 E5
0410 2A 2E 04 EB 2A 2C 04 E5 D5 21 3F 01 C3 F0 EE 3E
0420 30 32 40 00 32 06 FD D1 C1 E1 F1 C9

0430 F5 E5 C5 D5 3E F0 32 40 00 32 06 FD 21 4A 04 E5
0440 2A 57 04 E5 21 3F 01 C3 17 EF 3E 30 32 40 00 32
0450 06 FD D1 C1 E1 F1 C9
```

Ausgabe alter Schriftzeichen an einen Drucker

```
1     REM ************************************************
2     REM * Ausgabe alter Buchstaben                     *
3     REM * von Stefan Goller, Meckenheim                *
4     REM * lauft in diese Version auf ITOH 8510         *
5     REM ************************************************
10    CLEAR 20000:DIM S(90,17,1),EINGABE$(60)
11    FOR I=0 TO 90:S(I,0,0)=0:NEXT
12    RESTORE
15    BST=0:NUM=0:FLG=0:GOSUB 40
16    BST=8:NUM=0:FLG=0:GOSUB 40
17    BST=12:NUM=0:FLG=0:GOSUB 40
18    BST=26:NUM=0:FLG=0:GOSUB 40
19    BST=31:NUM=0:FLG=0:GOSUB 40
20    BST=62:NUM=0:FLG=0:GOSUB 40
21    BST=59:NUM=0:FLG=0:GOSUB 40
22    BST=33:NUM=0:FLG=0:GOSUB 40
25    BST=65:NUM=0:FLG=0:GOSUB 40:GOTO 400
40    READ IN
45    IF IN=(-2) THEN RETURN
50    S(BST,NUM,FLG)=IN
55    IF IN=(-1) THEN 300
60    NUM=NUM+1:GOTO 40
201   FOR I=0 TO BST
205   FOR J=0 TO 1
210   K=0
215   PRINT CHR$(27);"S0014";
220   O=S(I,K,J):IF O=(-1) THEN PRINT :GOTO 240
225   PRINT CHR$(O);
230   K=K+1:GOTO 220
240   NEXT J:NEXT I:POKE #131,1:POKE #FE01,0
300   NUM=0:IF FLG=0 THEN FLG=1:GOTO 40
305   FLG=0:BST=BST+1:GOTO 40
400   REM Ausgabe eines Textes
401   POKE #FE03,#A0:POKE #FE01,2
402   PRINT CHR$(12)
403   POKE #131,3:PRINT CHR$(27);"T15";CHR$(27);"Q";CHR$(27);")";CHR$(27);"L008";:POKE #13
1,1
407   GOTO 500:REM oder GOTO 700
409   FOR J=0 TO 1
410   FOR I=0 TO LEN(TEXT$)-1
420   E$=MID$(TEXT$,I,1):Z=ASC(E$)-32
425   IF S(Z,0,0)=0 THEN POKE #131,3:PRINT "Zeichen nicht definiert":GOTO 480
430   L=S(Z,0,J):PRINT CHR$(27);"S00";
435   IF L<10 THEN PRINT "0";:E=L:GOTO 445
440   E=INT(L/10):PRINT MID$(STR$(E),1,1);:E=L-E*10
445   PRINT MID$(STR$(E),1,1);
450   FOR K=1 TO L
455   PRINT CHR$(S(Z,K,J));:NEXT K
456   PRINT CHR$(27);"S0004";:FOR M=1 TO 4:PRINT CHR$(0);:NEXT
460   NEXT I:PRINT :NEXT J:PRINT :RETURN
480   PRINT :PRINT :POKE #131,1:POKE #FE01,0:END
499   REM ************************************************
500   REM Die Eingabe ist so gedacht, dass in einem zweiten
501   REM Programm das Feld EINGABE$ beschrieben wird und
502   REM hier nur noch gelesen zu werden braucht.
503   REM Dabei gilt /=ss  Ä=ae  ;=oe  ^=ue
504   REM Man kann natuerlich den auszugebenden Text auch in
505   REM diese Programm schreiben. Dann ist so vorzugehen
506   REM wie in den Zeilen 700-799 beschrieben.
507   REM ************************************************
509   CALLM #F000:REM LOOK
510   PRINT :INPUT "  Soll diese File eingelesen werden (J,N)";E$:PRINT :PRINT
515   IF E$="J" THEN 540
520   IF E$()"N" THEN 510
525   CALLM #F000:REM SKIP1
```

NP60

```
530   GOTO 509
540   LOADA EINGABE$
541   POKE #FE03,#A0:POKE #FE01,2:POKE #131,3
545   COUNT=0
550   TEXT$=EINGABE$(COUNT):IF TEXT$="ENDE" THEN 480
560   GOSUB 409:COUNT=COUNT+1:GOTO 550
599   REM ************************************************
700   REM TEXT$ direkt im Programm
701   REM Aendere Zeile 407 in GOTO 700
710   POKE #FE03,#A0:POKE #FE01,2:POKE #131,3
720   TEXT$="          Ausgabe alter Schriftzeichen an einen Drucker "
730   GOSUB 409
799   GOTO 480
1000  REM Blank
1010  DATA 5,0,0,0,0,0,-1
1020  DATA 5,0,0,0,0,0,-1
1030  DATA 5,#02,#7E,#FF,#7E,#02,-1
1040  DATA 5,#00,#10,#3B,#10,#00,-1
1050  DATA 5,#06,#02,#00,#06,#02,-1
1060  DATA 5,#00,#00,#00,#00,#00,-1
1070  DATA -2
1080  REM Klammern
1090  DATA 4,#F0,#FC,#06,#01,-1
1100  DATA 4,#03,#0F,#18,#20,-1
1110  DATA 4,#01,#06,#FC,#F0,-1
1120  DATA 4,#20,#18,#0F,#03,-1
1130  DATA -2
1140  REM Komma
1150  DATA 3,#00,#00,#00,-1
1160  DATA 3,#10,#B8,#70,-1
1170  DATA 6,#00,#00,#00,#00,#00,#00,-1
1180  DATA 6,#01,#01,#01,#01,#01,#01,-1
1190  DATA 3,#00,#00,#00,-1
1200  DATA 3,#10,#38,#10,-1
1210  DATA 9,#00,#FC,#FE,#41,#63,#77,#4E,#C4,#C0,-1
1220  DATA 9,#10,#1F,#3F,#00,#C0,#20,#10,#19,#0F,-1
1230  REM Ziffern
1240  DATA 9,#70,#FC,#02,#01,#03,#07,#1E,#FC,#F8,-1
1250  DATA 9,#07,#0F,#1E,#38,#30,#20,#10,#0F,#01,-1
1260  DATA 6,#02,#01,#FF,#FE,#00,#00,-1
1270  DATA 6,#00,#10,#1F,#3F,#20,#10,-1
1280  DATA 10,#00,#00,#04,#0E,#86,#C6,#E3,#73,#3E,#1C,-1
1290  DATA 10,#38,#34,#12,#11,#11,#10,#30,#30,#10,#00,-1
1300  DATA 8,#00,#44,#4E,#46,#E3,#B8,#1E,#04,-1
1310  DATA 8,#18,#30,#30,#30,#38,#1F,#0F,#04,-1
1320  DATA 11,#80,#40,#30,#0C,#02,#03,#FF,#FE,#00,#00,#00,-1
1330  DATA 11,#00,#01,#03,#03,#03,#13,#1F,#3F,#23,#13,#02,-1
1340  DATA 8,#FF,#83,#43,#43,#E3,#E3,#C3,#01,-1
1350  DATA 8,#18,#30,#30,#30,#38,#19,#0F,#04,-1
1360  DATA 10,#E0,#F8,#82,#81,#C3,#43,#C6,#C6,#C6,#02,-1
1370  DATA 10,#07,#0F,#1E,#38,#30,#10,#10,#0F,#07,#00,-1
1380  DATA 9,#04,#02,#83,#C3,#23,#1B,#07,#03,#01,-1
1390  DATA 9,#38,#3F,#1F,#01,#00,#00,#00,#00,#00,-1
1400  DATA 10,#0C,#3E,#71,#C1,#43,#43,#C3,#DE,#8C,#00,-1
1410  DATA 10,#0E,#1F,#1B,#30,#30,#10,#10,#19,#0F,#06,-1
1420  DATA 9,#F0,#F8,#C6,#02,#07,#8F,#BE,#FC,#F8,-1
1430  DATA 9,#00,#11,#19,#39,#31,#30,#20,#1F,#07,-1
1440  DATA -2
1450  REM Doppelpunkt
1460  DATA 3,#80,#C0,#80,-1
1470  DATA 3,#10,#39,#10,-1
1480  DATA 8,#00,#C6,#E2,#10,#10,#36,#E2,#E0,-1
1490  DATA 8,#10,#1F,#3F,#30,#20,#10,#1F,#0F,-1
1500  DATA -2
1510  REM Fragezeichen
1520  DATA 6,#06,#03,#83,#7F,#1E,#0C,-1
```

```
1530  DATA 6,#00,#10,#3B,#10,#00,#00,-1
1540  DATA -2
1550  REM ae
1560  DATA 10,#20,#F0,#F6,#22,#00,#20,#F6,#F2,#20,#00,-1
1570  DATA 10,#10,#1F,#3F,#20,#10,#10,#1F,#3F,#20,#10,-1
1580  DATA -2
1590  REM ue
1600  DATA 9,#20,#30,#26,#62,#60,#66,#E2,#00,#00,-1
1610  DATA 9,#1E,#3D,#39,#11,#12,#0A,#3F,#1F,#10,-1
1620  DATA -2
1630  REM grosse Buchstaben
1640  DATA 13,#00,#00,#C0,#B1,#8B,#87,#9E,#FC,#E2,#01,0,0,0,-1

1650  DATA 13,#38,#26,#0D,#0C,#18,#18,#38,#10,#0F,#1F,#38,#20,#10,-1
1660  DATA 13,#02,#81,#FF,#FE,#01,#F8,#FE,#42,#21,#71,#D7,#CE,#88,-1

1670  DATA 13,#C1,#A8,#17,#13,#28,#27,#33,#30,#30,#10,#10,#0F,#07,-1

1680  DATA 11,#FC,#82,#00,#FC,#FE,#03,#06,#06,#06,#06,#02,-1
1690  DATA 11,#03,#0F,#1E,#39,#30,#30,#30,#20,#20,#10,#10,-1
1700  DATA 12,#01,#02,#E6,#F6,#0E,#06,#F6,#0E,#06,#0E,#FC,#F8,-1

1710  DATA 12,#82,#46,#BF,#4F,#20,#38,#37,#30,#30,#30,#3F,#1F,-1

1720  DATA 11,#FC,#82,#00,#FC,#FE,#03,#66,#76,#6E,#26,#02,-1
1730  DATA 11,#03,#0F,#1E,#39,#30,#30,#30,#20,#20,#10,#10,-1
1740  DATA 11,#00,#20,#10,#10,#FC,#02,#FE,#FF,#12,#12,#11,-1
1750  DATA 11,#40,#40,#80,#40,#3F,#20,#1F,#0F,#00,#00,#00,-1
1760  DATA 12,#F8,#84,#02,#FE,#7E,#C3,#66,#56,#5E,#CC,#C4,#80,-1
1770  DATA 12,#03,#0F,#1E,#39,#30,#30,#30,#20,#20,#10,#1F,#0F,-1
1780  DATA 14,#42,#41,#C3,#FF,#FE,#01,#FC,#FE,#41,#41,#20,#60,#C0,#80,-1
1790  DATA 14,#08,#C4,#A4,#1F,#27,#30,#6F,#67,#60,#40,#00,#60,#9F,#07,-1
1800  DATA 9,#82,#43,#C1,#FD,#03,#FC,#FE,#01,#01,-1
1810  DATA 9,#21,#10,#30,#2F,#20,#1F,#0F,#00,#00,-1
1820  DATA 10,#00,#86,#43,#C3,#FD,#03,#F8,#FC,#01,#01,-1
1830  DATA 10,#80,#41,#C0,#C0,#BF,#40,#3F,#1F,#00,#00,-1
1840  DATA 12,#36,#13,#D3,#FD,#3E,#C1,#BC,#8A,#58,#30,#30,#00,-1

1850  DATA 12,#26,#12,#17,#2B,#34,#33,#30,#31,#13,#07,#3E,#18,-1

1860  DATA 10,#36,#13,#D3,#FF,#3D,#80,#FC,#7E,#01,#00,-1
1870  DATA 10,#26,#12,#13,#2F,#38,#35,#33,#30,#30,#10,-1
1880  DATA 16,#80,#40,#F0,#FC,#32,#01,#82,#FF,#FE,#84,#82,#03,#07,#FE,#F8,#08,-1
1890  DATA 16,#20,#10,#1F,#17,#00,#01,#09,#3F,#1F,#08,#01,#00,#00,#1F,#37,#20,-1

1900  DATA 12,#08,#84,#04,#FC,#FE,#02,#03,#07,#07,#3E,#F8,#00,-1

1910  DATA 12,#21,#10,#31,#33,#0F,#00,#00,#00,#30,#2E,#11,#10,-1

1920  DATA 12,#F8,#84,#3E,#FF,#03,#03,#03,#07,#07,#0E,#1C,#F8,-1
1930  DATA 12,#03,#0F,#1E,#39,#30,#30,#30,#20,#20,#10,#18,#07,-1
1940  DATA 11,#22,#41,#61,#FE,#FC,#08,#04,#0E,#3E,#F8,#E0,-1
1950  DATA 11,#04,#82,#82,#7F,#FF,#88,#08,#04,#02,#01,#00,-1
1960  DATA 14,#F8,#84,#3E,#FF,#03,#03,#03,#07,#07,#0E,#1C,#F8,00,00,-1
1970  DATA 14,#03,#0F,#1E,#39,#30,#30,#30,#20,#20,#10,#18,#3F,#30,#10,-1

1980  DATA 13,#22,#21,#51,#FE,#FC,#08,#F8,#84,#8E,#5C,#38,#10,#00,-1
1990  DATA 13,#20,#10,#18,#1F,#1F,#10,#3F,#20,#01,#03,#07,#1E,#38,-1
2000  DATA 10,#B8,#66,#42,#C3,#C6,#86,#84,#80,#80,#00,-1
2010  DATA 10,#21,#12,#12,#12,#32,#36,#2C,#11,#0B,#07,-1
2020  DATA 13,#04,#02,#F9,#85,#03,#FF,#7F,#03,#03,#02,#02,#02,#01,-1
```

NP60,2

```
2030  DATA 13,#00,#00,#03,#0F,#1E,#39,#30,#30,#30,#20,#20,#10,#10,-1
2040  DATA 15,#02,#03,#01,#F1,#8D,#03,#FD,#F3,#00,#00,#02,#07,#FE,#FC,#00,-1
2050  DATA 15,#00,#00,#00,#03,#0F,#1E,#3B,#31,#30,#30,#20,#20,#1F,#3F,#20,-1
2060  DATA 10,#02,#81,#81,#FE,#F8,#00,#00,#02,#FC,#F0,-1
```

```
2070  DATA 10,#21,#18,#16,#33,#31,#30,#10,#08,#07,#01,-1
2080  DATA 15,#02,#01,#81,#FE,#F8,#00,#00,#02,#FC,#F0,#00,#00,#02,#FC,#F0,-1
2090  DATA 15,#21,#38,#16,#13,#31,#30,#10,#28,#17,#31,#30,#10,#08,#07,#01,-1
2100  DATA 11,#82,#C2,#CF,#FC,#F8,#C0,#E0,#DC,#C2,#47,#06,-1
2110  DATA 11,#1C,#38,#10,#0E,#01,#03,#0F,#1C,#30,#20,#10,-1
2120  DATA 14,#06,#07,#03,#06,#18,#60,#80,#C0,#F0,#3C,#0E,#03,#01,#01,-1
2130  DATA 14,#00,#00,#00,#00,#30,#3C,#0F,#03,#00,#00,#00,#00,#00,#00,-1
2140  DATA 12,#04,#02,#A3,#A6,#A6,#E6,#E3,#B9,#A5,#A3,#01,#00,-1
2150  DATA 12,#00,#20,#30,#2C,#13,#11,#10,#30,#30,#30,#10,#08,-1
2160  DATA -2
2170  REM kleine Buchstaben
2180  DATA 9,#20,#30,#20,#60,#60,#60,#E0,#00,#00,-1
2190  DATA 9,#1E,#3D,#39,#11,#12,#0A,#3F,#1F,#10,-1
2200  DATA 9,#01,#FE,#FC,#83,#81,#40,#40,#E0,#C0,-1
2210  DATA 9,#10,#1F,#3F,#30,#30,#20,#10,#1F,#1F,-1
2220  DATA 6,#20,#E0,#F0,#30,#30,#60,-1
2230  DATA 6,#10,#1F,#3F,#30,#10,#08,-1
2240  DATA 8,#03,#C6,#EC,#1C,#18,#30,#E0,#E0,-1
2250  DATA 8,#10,#1F,#3F,#30,#20,#10,#1F,#0F,-1
2260  DATA 8,#00,#C0,#E0,#10,#10,#30,#E0,#80,-1
2270  DATA 8,#10,#1F,#3F,#32,#11,#11,#10,#00,-1
2280  DATA 7,#10,#10,#FC,#FE,#13,#13,#02,-1
2290  DATA 7,#00,#10,#1F,#3F,#20,#10,#00,-1
2300  DATA 9,#00,#C0,#E0,#30,#30,#60,#E0,#C0,#20,-1
2310  DATA 9,#48,#CF,#9F,#98,#98,#D0,#5F,#6F,#00,-1
2320  DATA 8,#02,#FC,#F8,#06,#82,#80,#C0,#E0,-1
2330  DATA 8,#10,#3F,#9F,#81,#80,#40,#3F,#07,-1
2340  DATA 5,#20,#F0,#F6,#22,#00,-1
2350  DATA 5,#10,#1F,#3F,#20,#10,-1
2360  DATA 4,#20,#10,#F6,#E2,-1
2370  DATA 4,#00,#9C,#BF,#67,-1
2380  DATA 8,#01,#FE,#FC,#93,#89,#58,#38,#30,-1
2390  DATA 8,#10,#1F,#3F,#20,#11,#07,#1E,#38,-1
2400  DATA 5,#01,#FE,#FC,#03,#01,-1
2410  DATA 5,#10,#1F,#3F,#20,#10,-1
2420  DATA 15,#20,#10,#F0,#E0,#20,#10,#10,#F0,#E0,#20,#10,#10,#F0,#E0,#20,-1
2430  DATA 15,#00,#10,#3F,#3F,#10,#00,#10,#3F,#1F,#10,#00,#10,#3F,#3F,#10,-1
2440  DATA 10,#20,#10,#F0,#E0,#20,#10,#10,#F0,#E0,#20,-1
2450  DATA 10,#00,#10,#3F,#3F,#10,#00,#10,#3F,#3F,#10,-1
2460  DATA 8,#00,#C0,#E0,#10,#10,#30,#E0,#E0,-1
2470  DATA 8,#10,#1F,#3F,#30,#20,#10,#1F,#0F,-1
2480  DATA 8,#10,#F0,#C0,#60,#30,#70,#E0,#C0,-1
2490  DATA 8,#80,#7F,#9F,#10,#30,#30,#1F,#0F,-1
2500  DATA 8,#C0,#E0,#30,#30,#20,#E0,#E0,#20,-1
2510  DATA 8,#0F,#1F,#38,#30,#10,#8F,#7F,#80,-1
2520  DATA 8,#30,#F0,#E0,#40,#20,#30,#F0,#60,-1
2530  DATA 8,#10,#1F,#3F,#30,#08,#00,#00,#00,-1
2540  DATA 8,#C0,#E0,#30,#30,#20,#20,#20,#10,-1
2550  DATA 8,#21,#13,#12,#13,#31,#3E,#1E,#00,-1
2560  DATA 6,#08,#0C,#FE,#FF,#08,#08,-1
2570  DATA 6,#00,#10,#1F,#3F,#20,#10,-1
2580  DATA 10,#20,#F0,#F0,#20,#00,#20,#F0,#F0,#20,#00,-1
2590  DATA 10,#10,#1F,#3F,#20,#10,#10,#1F,#3F,#20,#10,-1
2600  DATA 8,#1C,#FE,#F1,#00,#00,#10,#E0,#C0,-1
2610  DATA 8,#10,#1F,#3F,#30,#20,#10,#1F,#1F,-1
2620  DATA 13,#1C,#FE,#F1,#00,#00,#10,#E0,#C0,#00,#00,#10,#E0,#C0,-1

2630  DATA 13,#10,#1F,#3F,#30,#20,#10,#1F,#3F,#30,#20,#10,#1F,#1F,-1
2640  DATA 10,#20,#10,#70,#E0,#80,#00,#80,#60,#10,#30,-1
2650  DATA 10,#30,#20,#10,#0C,#03,#07,#1C,#38,#20,#10,-1
2660  DATA 9,#20,#F0,#F0,#20,#00,#20,#F0,#F0,#20,-1
2670  DATA 9,#48,#0F,#9F,#90,#80,#C8,#7F,#7F,#00,-1
2680  DATA 5,#20,#10,#30,#E0,#40,-1
2690  DATA 5,#C2,#A3,#13,#0E,#00,-1
2700  DATA -2
```

Erstellung des Text-Arrays

```
1     REM ************************************************
2     REM * Erstellung des Feldes, welches das Programm   *
3     REM * zur Schriftzeichenausgabe verarbeitet.        *
4     REM *                        Stefan Goller, Meckenheim *
5     REM ************************************************
9     CLEAR 20000
10    DIM AUSGABE$(60)
20    I=0
30    PRINT CHR$(12);"   Bitte geben Sie den Text ein"
31    PRINT :PRINT " /=ss  Ä=ae  ;=oe  ^=ue  ß=Blank  *=Komma  ENDE":PRINT
40    POKE #2C3,#FF:INPUT E$:PRINT
45    IF E$="ENDE" THEN 100
50    IF LEN(E$)>60 THEN PRINT "  TEXTZEILE ZU LANG":GOTO 40
55    FOR J=0 TO LEN(E$)-1
56    IF MID$(E$,J,1)="ß" THEN E$=LEFT$(E$,J)+" "+RIGHT$(E$,LEN(E$)-(J+1))
57    IF MID$(E$,J,1)="*" THEN E$=LEFT$(E$,J)+","+RIGHT$(E$,LEN(E$)-(J+1))
58    NEXT
60    AUSGABE$(I)=E$:I=I+1
70    IF I=61 THEN PRINT "Keine weiteren Eingaben moeglich"
80    GOTO 40
100   AUSGABE$(I)=E$:PRINT CHR$(12);:FOR J=0 TO I-1
101   PRINT AUSGABE$(J)
102   NEXT
109   PRINT :INPUT " Soll das File hier zu Ende sein <J,N>";I$:PRINT
110   IF I$="N" THEN 40
120   IF I$<>"J" THEN 100
130   PRINT :PRINT :INPUT " Filename";E$:PRINT :E$=" "+E$
140   SAVEA AUSGABE$ E$
150   END
```

Dieses Programm laesst sich natuerlich noch wesentlich benutzerfreundlicher gestalten.

Der Zeichensatz:

ABCDEFGHIJKLMNOPQRSTUVW

abcdefghijklmnopqrstuvw

1234567890

,?!.-"

Exakte Fixpunkt-Rechnung bis zu 10 Stellen

Wer sich etwas näher mit der Rechengenauigkeit des DAI beschäftigt, wird sehr schnell feststellen, daß die behaupteten 6 Stellen doch eher geschmeichelt sind. Da treten nach einigen Operationen schon recht bemerkenswerte Rechen- und Umwandlungsfehler auf, die man zudem noch kaum bemerkt (es sei denn, man rechnet mit dem Taschenrechner nach).
Die Integer-Rechnung hingegen bietet eine Exaktheit bis zu 10 Stellen, genauergesagt bis zur Zahl 2147483647. Hier hat die rechnerinterne Binärdarstellung keinen Einfluß auf die Rechen- und Ausgabegenauigkeit. Als Zugabe bekommt man hier zusätzlich eine Geschwindigkeitserhöhung.
Natürlich gibt es bei der Ganzzahl-Rechnung nicht die FPT-Funktionen, sondern nur '+', '-', '*', 'DIV' und 'MOD'.
Eine andere Möglichkeit, die Genauigkeit zu erhöhen, ist die Benutzung eines Maschinensprache-Paketes für BCD-Rechnungen. Das wäre jedoch für Multiplikation und Division sehr aufwendig. Vernünftige BASICs und verschiedene andere Hochsprachen besitzen so etwas von Haus aus!

Für bestimmte Anwendungen im Finanzbereich ist eine gewisse Exaktheit in den Rechnungen vom Finanzamt vorgeschrieben. Deshalb werden ja z.B. die FIBU-Programme vor dem Vertrieb überprüft. Doch diese Überlegungen können sicher auch in anderen Bereichen von Nutzen sein.

Die Behauptung, nur Italiener könnten ihre Währung in Integer rechnen, ist einfach absurd. Denn haben wir nicht z.B. auch Pfennige? Und meines Wissens hat eine DM 100 davon. Rechnen wir nun einfach alles in Pfennigen oder Zehntelpfennigen, können wir die Berechnungen in Integer-Variablen vornehmen. Das Problem liegt dann nur in der Ein- und Ausgabe der Zahlen. Aus diesem Grunde habe ich zwei Unterprogramme geschrieben, die die Umwandlung von Strings in Zahlen und umgekehrt vornehmen. Die STR$-Funktion ist nicht benutzbar, da sie die Zahl erst in FPT umwandelt und VAL() auch nicht, weil sie ein FPT-Ergebnis liefert!
Das Programm ab Zeile 4000 wandelt eine Zahl in den entsprechenden String für die Ausgabe um, die Routine ab 5000 wandelt Eingabe-Strings in die zugehörigen Zahlen um. Bei der Eingabe habe ich den '.' als Dezimaltrennung vorgesehen, bei der Ausgabe ein ','.

Alles Folgende gilt für NK%=3:NKD%=2 !
Rechnen wir z.B. in Zehntelpfennigen, haben wir für die DM drei Nachkommastellen (deshalb 'Fixpunkt') und NK%=3. In diesem Falle können noch immer Beträge bis über 2 Mio. DM berechnet werden. (Wer mit größeren Beträgen rechnet, kann sich auch einen besseren Rechner leisten!) Drei Stellen nach dem Komma sind also immer exakt. Bei der Ausgabe der Zahlen benötigt man jedoch meist nur zwei Stellen (NKD%=2), sodaß die letzte Stelle gerundet werden muß.

Der Betrag 34,456 DM ist also beim Rechnen die Zahl 34456 und wird als "34,46" DM ausgegeben, 1 DM ist 1000 und wird zu "1,00" DM.
Nehmen wir an, ich schreibe eine Rechnung über zwei DAI-pc a 1250,76DM (Traumpreis) und eine Floppy zu 876,50DM (ebenso).
Dann ist PCPREIS%=1250760 und FLPREIS%=876500.
Der Rechnungsbetrag ist RNETTO%=2*PCPREIS%+FLPREIS% (=3378020).
Ist die Mehrwertsteuer z.B. 14% (=14/100), so rechne ich
RMWST%=RNETTO%/100 (=33780)
IF RNETTO% MOD 100 >= 50 THEN RMWST%=RMWST%+1
RMWST%=RMWST%*14 (=472920) RBRUTTO%=RNETTO%+RMWST% (=3850940)
(Erst dividieren, sonst evtl. OVERFLOW, obwohl erst multiplizieren genauer wäre!)
Um das Ergebnis auszugeben: Z%=RBRUTTO%:GOSUB 4000:PRINT Z$ ("3850,94")

Sehr zu beachten ist, daß in der Berechnung keine FPT-Zahlen (1.0) auftreten, sonst ist die ganze schöne Genauigkeit dahin!

Viel Spaß beim Kassensturz

Bernd Preusing DnDt1/84 NP 61

EXAKTE FIXPUNKT-RECHNUNG

```
1     REM !!!!!!!!!!!!!!!!!!!!!!   IMP INT   !!!!!!!!!!!!!!!!!!!!!!!!!!!!
2     REM (c) Bernd Preusing 1/84 f. DAInamic Dt.
100   CLEAR 1000:REM KLEINER TEST
110   NK=3:NKD=2:REM GENAUIGKEIT 3 STELLEN, 2 STELLEN DRUCKEN
120   INPUT Z$:GOSUB 5000:IF FLAG=1 THEN PRINT :GOTO 120
140   PRINT TAB(20);Z%;:GOSUB 4000
160   PRINT TAB(40);Z$:GOTO 120
999   REM
4000  REM UNTERPROGRAMM ZUR UMWANDLUNG EINER GANZZAHL IN
4010  REM EINEN BETRAGS-STRING MIT KOMMA-SETZUNG UND EVTL.
4020  REM RUNDUNG.  Z.B.: 123456 ==> "123,46"
4030  REM EINGANG:   Z%        AUSGANG:   Z$
4040  REM PARAMETER: NK%  (NACHKOMMASTELLEN)
4050  REM .          NKD% (ZU DRUCKENDE NACHKOMMASTELLEN)
4060  REM BEDINGUNG: NK% >= NKD% > 0
4070  REM BEISPIEL:  Z%=2344557, NK%=4, NKD%=2 => Z$="234,46"
4080  REM .          Z%=-9     , NK%=2, NKD%=1 => Z$="-0,1"
4100  Z1=Z
4110  Z$="":VORZ$="":RD=0
4120  IF Z1<0 THEN VORZ$="-":Z1=-1*Z1
4130  REM STELLEN, DIE MIT DER EVTL. RUNDUNG NICHTS ZU TUN
4131  REM HABEN, ABSCHNEIDEN:
4140  KAPP=NK-NKD-1:IF KAPP>0 THEN Z1=Z1/10^KAPP
4150  REM EVTL LETZTE STELLE RUNDEN:
4160  IF Z1 MOD 10>=5 THEN RD=1
4170  IF NK>NKD THEN Z1=Z1/10+RD
4180  REM JETZT DEN STRING BILDEN:
4200  Z2=Z1 MOD 10:Z1=Z1/10
4210  Z$=CHR$(Z2+#30)+Z$
4220  IF Z1>0 GOTO 4200
4230  Z3=LEN(Z$):IF Z3<NKD+1 THEN Z$="0"+Z$:GOTO 4230
4240  Z$=VORZ$+LEFT$(Z$,Z3-NKD)+","+RIGHT$(Z$,NKD)
4250  REM EVTL. ZUM RECHTSBUENDIGEN DRUCKEN BLANKS VORSETZEN
4260  Z$=SPC(12-LEN(Z$))+Z$
4270  RETURN
5000  REM EINGABE VON FIXKOMMA-ZAHLEN UND UMWANDLUNG IN
5010  REM INTEGER-ZAHLEN
5020  REM EINGANG: Z$    AUSGANG: Z%, FLAG% (0=O.K.)
5030  REM Z$ KANN SEIN: "0.11" ODER "1234"
5040  REM PARAMETER: NK% (NACHKOMMASTELLEN) > 0 , <10
5050  REM BEISPIEL: Z$="158.1"   ,NK%=2 => Z%=15810
5060  REM .         Z$="-45"     ,NK%=3 => Z%=-45000
5070  REM .         Z$=".345"    ,NK%=1 => Z%=3 (OHNE RUND.)
5100  Z1$=Z$:FLAG=0:VORZ=1:Z=0
5110  IF ASC(Z1$)=ASC("-") THEN VORZ=-1:Z1$=RIGHT$(Z1$,LEN(Z1$)-1)
5120  L=LEN(Z1$):IF L=0 THEN 5500:REM FEHLER
5130  REM ZAHL CHECKEN UND KOMMAPOSITION SUCHEN:
5140  K=L:FOR I=0 TO L-1
5150  H$=MID$(Z1$,I,1)
5160  IF H$="." THEN K=I:GOTO 5180:REM KOMMAPOSITION
5170  IF H$<"0" OR H$>"9" THEN 5500:REM KEINE ZIFFER
5180  NEXT I
5200  REM IN ZAHL UMWANDELN:
5210  Z1$="0"+Z1$+"00000000000"
5220  Z1$=LEFT$(Z1$,K+1)+MID$(Z1$,K+2,NK)
5230  L=LEN(Z1$):IF L>11 OR (L=11 AND Z1$>"02147483647") THEN 5500
5240  FOR I=0 TO L-1:Z=Z*10:Z=Z+(ASC(MID$(Z1$,I,1))-#30):NEXT
5250  Z=Z*VORZ
5260  RETURN
5500  REM FEHLERAUSGANG: Z$ UNMOEGLICH
5510  FLAG=1:RETURN
```

Programm fuer MEMOCOM MDCR-D

Wichtige Einschraenkungen:

Dieses Programm kann hinter einem Programm 1USER und/oder einem MLP am Kopf einer Kassette stehen.
Es ist mit 'SAVE"Directory vers.3"' abzuspeichern.
Es wird DCR0 als vorhanden angenommen.
Die Directory verwaltet maximal 50 Programm (Datei) -Namen.
Die maximale Laenge eines Namens ist 59 Zeichen.

Das letzte Zeichen eines Namens darf nie CHR$(160) sein!
Dies ist das Kennzeichen des Directory files.
CHR$(160) ist ueber die Tastatur normalerweise nicht ■ erreichbar.
Hat jemand dieses Zeichen bereits vergeben, so sind Zeile 160 und 1220 zu aendern.

Bei dem ersten Aufruf von 'Directory vers.3' durch RUN erscheint die Frage: INIT YES/NO ?
Sie ist mit 'YES' zu beantworten, um ein Directoryfile zu erstellen.
Diesem File kann ein Name (Kassettenname) gegeben werden. Wird kein Name angegeben,so ist wird 'DIR' eingesetzt.

Die 'command list' zeigt die Befehle.
Die Befehle werden mit Cursor up/down ausgesucht und mit 'RETURN' ausgefuehrt.

Die Befehle und ihre Bedeutung:

-list or execute programs
1. Listen der in dem Directory-file stehenden Programme

In der 'command list' erscheinen die nun erlaubten Befehle.

Das Listen der Programme muss dazu erst mit 'RETURN' gestartet werden!

Die Taste 'E' fuehrt in das Hauptmenue zurueck.
Anmerkung: Wenn man mit 'cursor down' auf die unterste Bildschirmzeile geht, werden die weiteren Programme des Files gelistet. Steht der Cursor auf am oberen Ende der gelisteten Programme, wird mit 'cursor up' um eine Bildschirmseite zurueckgegangen.

Will man in ein Programm laden und starten, so geht man mit dem Cursor auf die entsprechende Zeile und drueckt 'L'.
Das entsprechende Programm wird nun aufgesucht.
Die Directory versucht das Programm unter exakt dem gleichen Namen, der in der Liste steht, zu laden.
Ist nach 5 Versuchen das Programm noch nicht gefunden, (Zeile 600-615), so wird die Suche abgebrochen und ins Menue zurueckgegangen, wobei der Befehl 'update Directory' empfohlen wird.
Ist das gesuchte Programm ein MLP, so wird DBL V1.1 ausgefuehrt.
Ist das gesuchte Programm in BASIC, so wird es geladen und gestartet.
Bei anderen Programmen als Typ 0 oder 1 geht die Directory in den 'command mode' zurueck.
Nun kann man die Directory mit dem Befehl '- end' verlassen.

Die von der Directory erlaubten Labels sind :
0 1 2 & # $ (siehe Zeile 50389,50390)
Es sind noch 4 Labels in Zeile 50390 frei:
4* 48 fuer '0' ist durch einen anderen ASCII-Wert zu ersetzen.

-update directory

2. Update liest das gesamte Band, und traegt die gefundenen Namen im Directory file ein.
Ist ein gefundenes Label nicht erlaubt,(siehe oben) nimmt die Directory einen Lesefehler an.
Das korrigierte Directory file wird auf der Kassette abgespeichert.

- printout to RS232

3. Ausgabe des Directory files fuer einen Drucker ITOH
Bei anderen Druckern sind die Steuersequenzen in
Zeile 2541, 2545, 2555, 2580 und 2590 zu aendern.
Kassetten, die keine Directory enthalten, koennen mit einem Namen versehen werden.
Das Datum kann z.B. in der Form '19.1.1984' eingegeben werden.

- read other tape

4. Es kann eine neue Kassette eingelegt werden, z.B. wenn man nur seine Directory's aktualisieren will.
Enthaelt eine Kassette keine 'Directory vers.3', so wird sie als NONSYS verwaltet. Sie kann nur gelesen werden, indem man den Befehl '-list or execute programs' benutzt.

- end

Man kann sich's denken.

Vorsicht bei groesseren Veraenderungen in der Directory:
Wird das Programm laenger, so wird das folgende Directory file zerstoert. Also bei solchen Veraenderungen REM's entfernen, um Platz zu sparen.

Viel Erfolg beim Eingeben

Detlef Sorgenfrei
Bruesseler Str. 33
1000 Berlin 65

```
  1       REM ************************************************
  2       REM * Directory vers.3                      fuer DCR *
  3       REM * DGS                                    19.1.84 *
  4       REM ************************************************
 10       GOTO 50000
 20       TITLE$="0Directory vers.3"
 30       MODE 0:PRINT CHR$(12)+" "+RIGHT$(TITLE$,LEN(TITLE$)-1)
 80       GOSUB 2100:PRINT " Searching for datafile":GOSUB 49100
 90       GOSUB 1400:PRINT CHR$(12):IF RIGHT$(NAME$,1)=CHR$(160) THEN GOSUB 16
          00:GOTO 210
 95       REM TRY AGAIN FROM BEGINNING OF TAPE
100       CALLM #F000:REM REW
110       FOR I%=0 TO 2:GOSUB 1400:IF NAME$=TITLE$ THEN 116
115       PRINT CHR$(12):NEXT:GOTO 190:REM THERE IS NO DIRECTORY!  - ABORT
116       PRINT CHR$(12):GOSUB 1400:IF RIGHT$(NAME$,1)=CHR$(160) THEN GOSUB 16
          00:GOTO 210
117       REM FOUND DIRECTORY BUT NO DATA-FILE FOLLOWING
120       MODE 0:PRINT CHR$(12):INPUT "Init new directory YES/NO ";YES$
130       PRINT :IF YES$<>"YES" THEN GOTO 190
135       PRINT " searching start of Directory vers.3":CALLM #F000:REM REW
136       FOR I%=0 TO 2:GOSUB 1400:IF NAME$=TITLE$ THEN 140
137       PRINT CHR$(12):NEXT:GOTO 95:REM THERE WAS A READ-ERROR!
140       PRINT CHR$(12);:INPUT " Name of cassette ";NAME$:IF LEN(NAME$)>57 TH
          EN NAME$=LEFT$(NAME$,57)
150       IF NAME$="" THEN NAME$="DIR":PRINT NAME$;
160       TIT$(0)=NAME$+CHR$(160)
170       GOSUB 2040:PRINT :INIT%=1
180       GOSUB 1500:GOTO 210
190       PRINT :GOSUB 2100:CURSOR 1,10:PRINT "this is no systemtape,";
200       PRINT " update is not allowed !":WRITEFLAG%=1:CALLM #F000:REM REW
209       REM COMMANDMODE
210       GOSUB 1000:PRINT CHR$(12)+" - select command line with cursor up/dow
          n":PRINT " - execute command press 'RETURN'":CURSOR 1,20
220       GOSUB 2000:IF A%=16 THEN IF CURY<20 THEN CURSOR 1,CURY+1
230       IF A%=17 THEN IF CURY>16 THEN CURSOR 1,CURY-1
240       IF A%<>13 GOTO 220
250       A%=21-CURY:ON A% GOTO 300,1200,2500,270
260       MODE 0:PRINT CHR$(12):CLEAR 100:END
270       GOSUB 2600:WRITEFLAG%=0:GOTO 30
300       GOSUB 2100:IF WRITEFLAG%=1 THEN IF TIT$(0)<>"NONSYS" THEN TIT$(0)="N
          ONSYS":GOSUB 1000:GOSUB 1300
310       IF IE%=0 THEN GOSUB 740:CURSOR 1,19:GOTO 220
315       ZEILE%=20:FOR I%=0 TO 4:PRINT SPC(59):NEXT:CURSOR 0,ZEILE%:PRINT " U
          se cursor up/down to list programs "
320       PRINT " press 'E' : exit to command mode"
330       PRINT " press 'L' : load /execute the program on cursorposition":PRI
          NT SPC(59):PRINT SPC(59)
340       CURSOR 0,14:PRINT CHR$(12)+" - press 'RETURN' to begin ":ZEILE%=14
350       CURSOR 1,ZEILE%:GOSUB 2000:IF A%<>13 THEN 350
360       IA%=1:GOSUB 740
369       REM LIST MAINPROGRAMM + COMMANDDECODING
370       CURSOR 1,CURY:GOSUB 2010
380       IF A%=16 THEN GOSUB 800
390       IF A%=17 THEN GOSUB 900
400       IF A% IOR 32=101 THEN GOSUB 2100:GOTO 210
410       IF A% IOR 32<>108 THEN 370
499       REM DECODE PROGRAMM AND START IT
510       PG%=IA%+(14-CURY):IF PG%>IE% THEN PG%=IE%
520       NAME$=TIT$(PG%):PRINT CHR$(12);" load /RUN "+NAME$
530       IF PG%>1 THEN POKE #1B0,PG%-1
```

```
 540      IF PG%)1 THEN PRINT " skipping to file ":CALLM #F015
 545      GOSUB 600:IF ERR%)0 THEN ERR%=0:GOTO 100
 550      IF LEFT$(NAME$,1)="0" THEN MODE 0:PRINT CHR$(12):CLEAR 100:CALLM #F8
          31
 560      IF LEFT$(NAME$,1)="1" THEN MODE 0:CLEAR 100:PRINT CHR$(12)+" executi
          ng  DBL ":CALLM #F800
 580      GOSUB 2100:PRINT " can't start file, only skip to."
 590      WAIT TIME 100:GOSUB 2100:WAIT TIME 100:GOTO 210
 600      GOSUB 1400:IF NAME$=TIT$(PG%) THEN 620
 610      ERR%=ERR%+1:IF ERR%<5 THEN 600
 615      FOR I%=0 TO 3:PRINT CHR$(12):WAIT TIME 30:PRINT " file not found":GO
          SUB 2100:NEXT:GOTO 2100
 620      PRINT "found ";NAME$:CALLM #F000:REM REW1
 630      ERR%=0:RETURN
 739      REM MOVE DATA-BLOCK IN SCREEN
 740      PRINT CHR$(12);:IF IE%=0 THEN GOSUB 2100:PRINT "file is empty":WAIT
          TIME 20:GOTO 2100
 750      FOR I%=IA% TO IA%+13:IF I%)IE% THEN PRINT :GOTO 770
 760      PRINT I%,TIT$(I%)
 770      NEXT I%:CURSOR 1,14
 780      RETURN
 799      REM EXECUTE CURSOR OR DATA UP
 800      IF CURY<14 THEN CURSOR 1,CURY+1:RETURN
 810      IF IA%)13 THEN IA%=IA%-13:GOSUB 740
 820      CURSOR 1,14:RETURN
 899      REM EXECUTE CURSOR OR DATA DOWN
 900      IF CURY)1 THEN CURSOR 1,CURY-1:RETURN
 910      IF IA%+13<IE% THEN IA%=IA%+13:GOTO 740
 920      RETURN
1000      MODE 0:PRINT CHR$(12)+" "+TIT$(0):PRINT " "+RIGHT$(TITLE$,16);SPC(29
          );"command list"
1010      PRINT LINE$
1020      PRINT " - list or execute programs":PRINT " - update directory"
1030      PRINT " - printout to RS232":PRINT " - read other tape":PRINT " - en
          d"
1040      PRINT LINE$:GOTO 49100
1199      REM UPDATE  A SYSTAPE
1200      IF WRITEFLAG%=1 THEN 210
1205      PRINT CHR$(12);:GOSUB 2040:PRINT "reading tape for update":GOSUB 121
          0:GOTO 210
1210      FOR I%=1 TO 50
1220      GOSUB 1400:IF RIGHT$(NAME$,1)=CHR$(160) THEN 1260
1230      TIT$(I%)=NAME$:NEXT I%
1240      IE%=50:PRINT "TOO MUCH FILES "
1250      FOR I%=0 TO 3:GOSUB 2100:NEXT I%:RETURN
1260      I%=I%-1:IF TIT$(I%)<>TITLE$ THEN PRINT "DATA ERROR":GOTO 2100
1269      REM CLEAR NAMES FROM TOP OF CASSETTE 1USER 1MLP AND Directory ARE IG
          NORED
1270      TIT$(I%)=SPC(59):I%=I%-1:IF I%<1 THEN 1290
1275      IF LEFT$(TIT$(I%),1)<>"1" THEN 1290
1280      IF TIT$(I%-1)<>"1USER" GOTO 1290
1285      TIT$(I%)=SPC(59):I%=I%-1:IF I%)0 THEN I%=I%-1:TIT$(I%)=SPC(59)
1290      IE%=I%:IF IE%)0 THEN 1500
1291      REM ON  SYS-TAPE THE FIRST 3 FILES 1USER 1 MLP 0DIR ARE NEVER SHOWN !
1295      GOTO 740
1299      REM READ NON-SYS-TAPES
1300      PRINT CHR$(12)+"now reading tape ":FOR I%=1 TO 50:GOSUB 1400
1310      TIT$(I%)=NAME$:IF I%<2 THEN NEXT I%
1320      IF TIT$(I%)=TIT$(1) THEN IE%=I%-1:GOTO 1350
1330      NEXT I%:GOTO 1240
```

```
1349      REM IF "FILES" NOT EMPTY START OF TAPE REACHED
1350      CALLM #F000:REM REW2 CAUTION! ERROR IF ONLY TWO PG'S WITH SAME NAME
          ON TAPE!
1360      IF PEEK(#1B0)<>0 THEN 1380
1370      CALLM #F000:REM SKIP2
1375      GOTO 1330
1380      CALLM #F000:REM REW
1390      RETURN
1399      REM READ NAME$ ON LAST LINE - IF  READ-ERROR  IGNORE, CONTINUE
1400      LI%=PEEK(VARPTR(LI$)) IOR (PEEK(VARPTR(LI$)+1) SHL 8)+1
1410      CURSOR 0,0:NAME$="":CALLM LI%:E%=(CURX-1)*2
1420      FOR LAEN%=0 TO E% STEP 2:NAME$=NAME$+CHR$(PEEK(#B3DD-LAEN%)):NEXT:PR
          INT
1470      S%=ASC(LEFT$(NAME$,1)):FOR LAEN%=0 TO 9:IF S%=ID%(LAEN%) THEN RETURN
1480      NEXT LAEN%:GOSUB 2100:PRINT " data read-error, found: ";NAME$:GOSUB
          2100:PRINT " ignored ":GOTO 1400
1499      REM WRITE DIRECTORY-FILE ON TAPE
1500      IF WRITEFLAG%=1 THEN 2100
1510      GOSUB 2040:IF INIT%=0 THEN CALLM #F000:REM REW1
1520      IF INIT%=1 THEN INIT%=0:PRINT " verifing directory":CALLM #F000:REM
          VER
1530      PRINT " saving file ":SAVEA TIT$ TIT$(0):PRINT " READY":GOTO 2100
1600      GOSUB 2100:PRINT CHR$(12)+" loading ";NAME$
1610      CALLM #F000:REM REW1
1620      LOADA TIT$ :CURSOR 1,23:PRINT TIT$(0.0)+SPC(59-LEN(TIT$(0))):CURSOR
          0,0
1630      FOR I%=0 TO 50:IF TIT$(I%)=SPC(59) THEN IE%=I%-1:RETURN
1640      NEXT:IE%=50:RETURN
2000      A%=GETC:IF A%<>0 THEN 2000
2010      A%=GETC:IF A%=0 THEN 2010:RETURN
2039      REM LOOKING  FOR WEN-PLUG
2040      ERR%=0:TEST%=PEEK(VARPTR(TEST$)) IOR (PEEK(VARPTR(TEST$)+1) SHL 8):C
          ALLM TEST%,ERR%
2050      IF ERR%=0 THEN RETURN
2060      FOR I%=0 TO 2:GOSUB 2100:NEXT:PRINT :PRINT " PUT IN WEN-PLUG":GOSUB
          2070:GOTO 2040
2070      PRINT " hit any key to continue":PRINT
2080      A%=GETC:IF A%<>0 THEN 2080
2090      A%=GETC:IF A%=0 THEN 2090
2100      SOUND 1 1 15 0 2500:WAIT TIME 5:SOUND OFF :WAIT TIME 5:RETURN
2499      REM HERE STARTS PRINTOUT, CHANGE ITOH SEQUENZ FOR YOUR PRINTER
2500      GOSUB 2505:GOTO 210
2505      PRINT :IF IE%=0 THEN GOTO 740
2510      PRINT CHR$(12);:IF TIT$(0)="NONSYS" THEN INPUT " Name of tape ";NAME$
2516      PRINT :IF DA$<>"" THEN 2530
2520      INPUT " today's date (DD.MM.YY) ";DA$:PRINT :IF LEN(DA$)>10 THEN DA$
          =""
2530      PRINT " If printer ready,";:GOSUB 2070:POKE #131,0
2540      IF TIT$(0)<>"NONSYS" THEN NAME$=TIT$(0):IF LEN(NAME$)>36 THEN NAME$=
          LEFT$(NAME$,36)
2541      REM ITHO TYPE E + LEFTMARGIN 8 +ENLARGED + UNDERLINE ON
2545      ESC$=CHR$(27):PRINT ESC$+"E"+ESC$+"L008"+CHR$(14)+ESC$+"X";
2550      PRINT NAME$+SPC(42-LEN(NAME$)-LEN(DA$))+DA$;
2555      PRINT CHR$(15);:REM ENLARGED OFF
2560      PRINT :PRINT :NAME$=SPC(10)+" number        title":NAME$=NAME$+SPC(55)
2565      PRINT NAME$+ESC$+"Y":REM STOP UNDERLINE
2570      PRINT :FOR I%=1 TO IE%:PRINT SPC(10);I%,TIT$(I%):NEXT
2575      A$="SYSTEM":IF WRITEFLAG%=1.0 THEN A$="NONSYS"
2580      PRINT ESC$+"X";:REM UNDERLINE ON
2585      PRINT A$+SPC(75)+"END";
```

```
 2590      PRINT ESC$+"Y":REM UNDERLINE STOP
 2595      PRINT CHR$(12):POKE #131,1:RETURN
 2599      REM DELETE OLD DATA-FILE
 2600      FOR I%=0 TO 50:TIT$(I%)=SPC(59):NEXT:IE%=0
 2610      PRINT CHR$(12)+"change tape in DCR,";:GOTO 2070
49099      REM SCROLL ONLY UNDER COMMANDLINE
49100      IF CURY=0 THEN CURSOR CURX,1
49110      LOC%=#BFEF-134*(23-CURY):POKE #8A,LOC% IAND 255:POKE #8B,LOC% SHR 8:
           RETURN
49999      REM INIT DIRECTORY
50000      MODE 0:COLORT 8 0 0 0:CLEAR 5000:POKE #131,1:PRINT CHR$(12):DIM TIT$
           (50):DIM ID%(9)
50009      REM INIT MLP'S AND STRINGS
50010      ENVELOPE 1 15
50020      FOR I%=0.0 TO 14:READ M%:TEST$=TEST$+CHR$(M%):NEXT
50030      FOR I%=0 TO 20:READ M%:LI$=LI$+CHR$(M%):NEXT
50040      FOR I%=0 TO 59:LINE$=LINE$+CHR$(11):NEXT
50199      REM DAInamic BOOTSTRAPLOADER V1.1
50200      READ BGNADDR%,NMBR%
50210      FOR ADDR%=BGNADDR% TO BGNADDR%+NMBR%-1:READ BYTE%:POKE ADDR%,BYTE%:N
           EXT
50220      READ C%:IF C%<>#FFFF THEN PRINT "DATA READ ERROR":PRINT :END
50230      FOR I%=0 TO 50:TIT$(I%)=SPC(59):NEXT:FOR I%=0 TO 9:READ ID%(I%):NEXT
           :GOTO 20
50240      REM Start DBL V1.1
50250      PRINT "Ä LOADING UTILITY FILE ü":CALLM BGNADDR%
50260      REM ****************** MLP BYTES ***************************
50270      REM TEST FOR WEN-PLUG
50280      DATA #F5,#E5,#37,#CD,9,#F0,#E6,#40,#E1,#23,#23,#23,#77
50290      DATA #F1,#C9
50300      REM TEST FOR NAME OF PG'S
50310      DATA #F5,#C5,#D5,#E5,#01,#40,#00,#11,#B1,#80,#21,#9E,#E6,#CD,#CE,#02
           ,#E1,#D1,#C1,#F1,#C9
50320      DATA #F800,86
50329      REM DBL BYTES
50330      DATA #31,#00,#F9,#21,#00,#00,#22,#00,#01,#01,#FF,#31,#CD,#CE,#02,#21
50340      DATA #54,#F8,#11,#56,#F8,#CD,#D1,#02,#21,#00,#B0,#EB,#2A,#54,#F8,#DC
50350      DATA #D1,#02,#CD,#D4,#02,#D2,#A8,#D2,#22,#9B,#02,#CD,#B8,#DE,#CD,#5E
50360      DATA #DD,#01,#37,#F8,#C3,#92,#C8,#AD,#01,#20,#18,#11,#5B,#20,#4C,#4F
50370      DATA #41,#44,#49,#4E,#47,#20,#42,#41,#53,#49,#43,#20,#5D,#FF,#8B,#19
50380      DATA #00,#AD,#00,#87,#00,#00,#FFFF
50389      REM 0 , 1, 2, #, $, &,4 ID'S NOT USED
50390      DATA 48,49,50,35,36,38,48,48,48,48
```

SIZE : 7166 BYTES DATE : 17.1.84
AUTHOR : Detlef Sorgenfrei TITLE :Directory vers.3

```
INT          :   A          used for GETC and jumptable
INT          :   ADDR       DBL start
INT          :   BGNADDR                 DBL
INT          :   BYTE       DBL
INT          :   C          DBL
INT          :   E          cursorposition in x-direction
INT          :   ERR        test for WEN-PLUG
INT          :   I          loopvariable
INT          :   IA         rel. position in directory
INT          :   IE         last element in file
INT          :   INIT
INT          :   LAEN       loops
INT          :   LI         start of MLP READ NAME
INT          :   LOC        adr. for scroll under commandline
INT          :   M          loops
INT          :   NMBR       DBL
INT          :   PG         rel. position in directoryfile
INT          :   S          ASCII-value of 1. character of name
INT          :   TEST       start of MLP test for WEN-PLUG
INT          :   WRITEFLAG               protection of NONSYS-tapes for writing
INT          :   ZEILE                   cursorposition in y
INT   (ARR)  :   ID         values of ASCII 0 1 2 # $ & and 4 other ID's
STR          :   A$         string 'SYS' or 'NONSYS'
STR          :   DA$        date
STR          :   ESC$       ASCII escape
STR          :   LI$        this is a MLP print name of pg's on last line
STR          :   LINE$                   underline on screen
STR          :   NAME$                   name of any pg
STR          :   TEST$                   this is MLP looking for WEN-PLUG
STR          :   TITLE$                  name 'Directory vers.3'
STR          :   YES$       string for init
STR   (ARR)  :   TIT$       holds field of programnames
```

```
0001                    TITL        Listing mit hervorgehobenen BASIC-Woertern
0002            ;
0003            ************************************************************************
0004            *  Das Programm listet Basic-Programme. Dabei werden Basic-Worte      *
0005            *  klein geschrieben und unterstrichen. Dies jedoch nicht in REM      *
0006            *  Zeilen und zwischen Anfuehrungszeichen.                            *
0007            *  Der Aufruf erfolgt durch CALLM#400:LIST [(line-num)[-][(line-num)]]*
0008            *                                                                     *
0009            *  (c) Stefan Goller, Meckenheim                  Februar 1984        *
0010            ************************************************************************
0011            ;
0012            * Die folgenden Variablen liegen im Bereich des ENVELOPE Speichers
0013     **01F5 MODE     EQU   :1F5     Bit 3: bei zusammengesetztem BASIC-Wort =1
0014                    ;              Bit 2: in Zeile mit fuehrendem REM =1
0015                    ;              Bit 1: zwischen zwei Anfuehrungszeichen =1
0016     **01F6 BUFFORT  EQU   :1F6     Position des naechsten Puffereintrages
0017     **01F8 BUFFLEN  EQU   :1F8     Anzahl der Zeichen im Puffer
0018     **01F9 BUFFER   EQU   :1F9     reservierter Bereich, in den auszugebende Zei-
0019                    ; chen geschrieben und daraufhin ueberprueft werden, ob sie
0020                    ; ein Basic-Wort darstellen. Der Buffer muss so lang wie das
0021                    ; laengste moegliche Wort sein. Also so lang wie die laengste
0022                    ; aus Alphazeichen und '$' bestehende Zeichenkette. Das wird
0023                    ; im allgemeinen entweder das BASIC-Wort ENVELOPE oder ein
0024                    ; Variablenname sein.
0025            *
0026     **CBBF COMTAB  EQU   :CBBF    !
0027     **CF91 OPTAB   EQU   :CF91    ! drei Tabellen, in denen die Basic-Worte stehen
0028     **CFE6 FUNCTAB EQU   :CFE6    !
0029     **DE02 TSTAZ   EQU   :DE02    testet, ob Zeichen Im Akku ein Alpha-Zeichen ist
0030                    ;             Wenn ja, dann ist das Carry-Bit =1
0031     **DE30 ADDA    EQU   :DE30    Addiert den Wert im Akku zu dem Wert in HL
0032     **DE14 COMPDE  EQU   :DE14    Vergleicht die Werte von HL und DE
0033            ;
0034     **0400         ORG   :400
0035 0400 AF    INIT    XRA   A
0036 0401 32F501        STA   MODE     loeschen des Registers fuer Sonderfaelle
0037 0404 212E04        LXI   H,PRINT  :
0038 0407 22DE02        SHLD  :2DE     :
0039 040A 3EC3          MVI   A,:C3    : Umleitung des Vektors :2DD, so dass jede Aus-
0040 040C 32DD02        STA   :2DD     : gabe zur Routine PRINT geht.
0041 040F 3E03          MVI   A,3      :
0042 0411 323101        STA   :131     :
0043 0414 2103FE        LXI   H,:FE03  !
0044 0417 36A1          MVI   M,:A1    ! Initialisierung des DCE-Bus, da die Drucker-
0045 0419 2B            DCX   H        ! ausgabe ueber den Bus gehen soll
0046 041A 2B            DCX   H        !
0047 041B 3602          MVI   M,2      ! Die Kartenadresse soll 2 sein
0048 041D 215505        LXI   H,TAB1
0049 0420 CD4B05        CALL  OUTLIST  gibt die Initialisierung an den Drucker aus
0050 0423 AF    BUFINIT XRA   A        !
0051 0424 32F801        STA   BUFFLEN  ! loeschen der Pufferzeiger
0052 0427 21F901        LXI   H,BUFFER !
0053 042A 22F601        SHLD  BUFFORT  !
0054 042D C9            RET
0055 042E       ;
0056 042E       ************************************************************************
0057 042E       *  Jedes Zeichen, das von LIST ausgegeben werden soll, wird wegen des *
0058 042E       *  Pointers in :2DD zu der Routine PRINT geschickt.                   *
0059 042E       *  Man kann das Programm in ein Format-Listing Programm einbinden, in-*
0060 042E       *  dem man die Zeichen, die von diesem Programm ausgegeben werden     *
0061 042E       *  sollen, an die Routine PRINT sendet.                               *
0062 042E       ************************************************************************
0063 042E       ;
```

```
0064 042E E5       PRINT   PUSH  H
0065 042F D5               PUSH  D
0066 0430 C5               PUSH  B
0067 0431 F5               PUSH  PSW
0068 0432 5F               MOV   E,A
0069 0433 2AF601           LHLD  BUFFORT  HL=Position, in die abzuspeichern ist
0070 0436 3AF801           LDA   BUFFLEN
0071 0439 57               MOV   D,A      D=Laenge des Wortes im Puffer
0072 043A 3AF501           LDA   MODE
0073 043D E602             ANI   %00000010
0074 043F C27804           JNZ   PRNTREM  Ausgabe waehrend einer REM-Zeile
0075 0442 7B               MOV   A,E
0076 0443 FE22             CPI   :22
0077 0445 CA8C04           JZ    FIND22   gefundenes Anfuehrungszeichen
0078 0448 3AF501           LDA   MODE
0079 044B E601             ANI   %00000001
0080 044D C29104           JNZ   PRNTDCE  Ausgabe waehrend eines PRINT "..."
0081 0450 7B               MOV   A,E
0082 0451 CD02DE           CALL  TSTAZ    !
0083 0454 DA5F04           JC    FINDBST  ! wenn auszugebendes Zeichen Alphazeichen
0084 0457 FE24             CPI   "$"      ! oder '$', dann wird das Zeichen in den
0085 0459 CA5F04           JZ    FINDBST  ! Puffer geschrieben.
0086 045C C39104           JMP   PRNTDCE
0087 045F          ;
0088 045F          * entry   : zu listendes Zeichen im Akku
0089 045F          * function: schreibt das Zeichen in den Puffer, erhoeht die beiden
0090 045F          *           Pointer und gibt die Kontrolle an LIST zurueck.
0091 045F          ;
0092 045F 77       FINDBST MOV   M,A
0093 0460 23               INX   H
0094 0461 22F601           SHLD  BUFFORT
0095 0464 14               INR   D
0096 0465 7A               MOV   A,D
0097 0466 32F801           STA   BUFFLEN
0098 0469 F1               POP   PSW
0099 046A C1               POP   B
0100 046B D1               POP   D
0101 046C E1               POP   H
0102 046D C9               RET
0103 046E          ;
0104 046E 3E04     FNDDOPP MVI   A,%00000100  markiert, dass ein moeglicherweise zusam-
0105 0470 C38604           JMP   CHANGE   mengesetztes Basic-Wort gefunden wurde.
0106 0473          ;
0107 0473 3E02     FINDREM MVI   A,%00000010  markiert, dass eine REM-Anweisung gefunden
0108 0475 C38604           JMP   CHANGE   wurde.
0109 0478          ;
0110 0478 7B       PRNTREM MOV   A,E
0111 0479 FE0D             CPI   :0D
0112 047B C29104           JNZ   PRNTDCE
0113 047E 3E02             MVI   A,%00000010  eine REM-Zeile ist nach CR beendet.
0114 0480 CD8604           CALL  CHANGE
0115 0483 C39104           JMP   PRNTDCE
0116 0486          ;
0117 0486 21F501   CHANGE  LXI   H,MODE   Aendern eines Bits im MODE-Flag
0118 0489 AE               XRA   M
0119 048A 77               MOV   M,A
0120 048B C9               RET
0121 048C          ;
0122 048C 3E01     FIND22  MVI   A,%00000001  markiert, dass ein Anfuehrungszeichen ge-
0123 048E CD8604           CALL  CHANGE   funden wurde.
0124 0491          ;
0125 0491          ************************************************************************
0126 0491          *  Es wird analysiert, ob im Puffer etwas steht. Falls dies der Fall  *
```

```
0127 0491          *  ist, so wird der Inhalt daraufhin ueberprueft, ob er ein BASIC-Wort*
0128 0491          *  darstellt. Wenn dem nicht so ist, so wird der Inhalt unveraendert  *
0129 0491          *  auf den Drucker ausgegeben. Sonst werden die Zeichen klein ge-     *
0130 0491          *  schrieben und unterstrichen.                                       *
0131 0491          *  PRNTDCE wird angesprungen, wenn von LIST ein Zeichen ausgegeben    *
0132 0491          *  werden soll, das nicht in den Puffer geschrieben werden kann, also *
0133 0491          *  kein '$' oder Buchstabe ist.                                       *
0134 0491          ************************************************************************
0135 0491          ;
0136 0491 7A       PRNTDCE  MOV   A,D
0137 0492 A7                ANA   A
0138 0493 CA1B05            JZ    NOBASIC  Puffer leer
0139 0496 3AF501            LDA   MODE
0140 0499 E604              ANI   %00000100
0141 049B C2AE04            JNZ   PRNT.1   Kein zusammengesetztes Wort moeglich
0142 049E 3E04              MVI   A,%0000100
0143 04A0 CD8604            CALL  CHANGE
0144 04A3 21C105            LXI   H,SPEZTAB
0145 04A6 1E01              MVI   E,1
0146 04A8 CD6805            CALL  TABLE
0147 04AB D20A05            JNC   BAS.WRD  Zusammengesetzes Wort gefunden
0148 04AE 21B805   PRNT.1   LXI   H,BEFLST
0149 04B1 1E01              MVI   E,1
0150 04B3 CD6805            CALL  TABLE
0151 04B6 D20A05            JNC   BAS.WRD  Wort, nicht aus den ROM Tabellen, gefunden
0152 04B9 21BFCB            LXI   H,COMTAB
0153 04BC 1E04              MVI   E,4
0154 04BE CD6805            CALL  TABLE
0155 04C1 F5                PUSH  PSW      carry wird gespeichert
0156 04C2 D5                PUSH  D
0157 04C3                   *
0158 04C3                   * Die folgenden Vergleiche stellen fest, ob HL nach dem Durch-
0159 04C3                   * suchen der Tabelle auf ein Laengenbyte eines Wortes zeigt,
0160 04C3                   * das gesondert zu behandeln ist.
0161 04C3                   *
0162 04C3 11CBCC            LXI   D,:CCCB  REM
0163 04C6 CD14DE            CALL  COMPDE
0164 04C9 CC7304            CZ    FINDREM
0165 04CC 114ACC            LXI   D,:CC4A  SOUND
0166 04CF CD14DE            CALL  COMPDE
0167 04D2 CC6E04            CZ    FNDDOPP
0168 04D5 1132CC            LXI   D,:CC32  WAIT
0169 04D8 CD14DE            CALL  COMPDE
0170 04DB CC6E04            CZ    FNDDOPP
0171 04DE 1153CC            LXI   D,:CC53  NOISE
0172 04E1 CD14DE            CALL  COMPDE
0173 04E4 CC6E04            CZ    FNDDOPP
0174 04E7                   *
0175 04E7 D1                POP   D
0176 04E8 F1                POP   PSW
0177 04E9 D20A05            JNC   BAS.WRD  BASIC-Kommando gefunden
0178 04EC 2191CF            LXI   H,OPTAB
0179 04EF 1E02              MVI   E,2
0180 04F1 CD6805            CALL  TABLE
0181 04F4 D20A05            JNC   BAS.WRD  BASIC-Operator gefunden
0182 04F7 21E6CF            LXI   H,FUNCTAB
0183 04FA 1E00              MVI   E,0
0184 04FC CD6805            CALL  TABLE
0185 04FF D20A05            JNC   BAS.WRD  BASIC-Funktion gefunden
0186 0502 1E00              MVI   E,0      !
0187 0504 CD2805            CALL  OUTBUFF  !Kein BASIC-Wort im Puffer, daher (E):=0
0188 0507 C31B05            JMP   NOBASIC  !
0189 050A                   *
```

```
0190 050A 216205   BAS.WRD  LXI   H,TAB2
0191 050D CD4B05            CALL  OUTLIST  Ausgabe der ESC-Sequenz: 'Starte Unterstreichen'
0192 0510 1E20              MVI   E,%00100000
0193 0512 CD2B05            CALL  OUTBUFF  Ausgabe des Pufferinhaltes
0194 0515 216505            LXI   H,TAB3
0195 0518 CD4B05            CALL  OUTLIST  Ausgabe der ESC-Sequenz: 'Beende Unterstreichen'
0196 051B                   *
0197 051B F1       NOBASIC  POP   PSW
0198 051C F5                PUSH  PSW      Ausgabe des Zeichens, das nicht mehr in den
0199 051D CD3C05            CALL  POUT     Puffer geschrieben wurde.
0200 0520 CD2304            CALL  BUFINIT  Ruecksetzen des Puffers
0201 0523 F1                POP   PSW
0202 0524 C1                POP   B
0203 0525 D1                POP   D
0204 0526 E1                POP   H
0205 0527 C9                RET
0206 0528                   *
0207 0528          * Ausgabe des Pufferinhaltes unter Beachtung des Inhaltes von reg E
0208 0528 21F901   OUTBUFF  LXI   H,BUFFER
0209 052B 7E       OUTB.0   MOV   A,M
0210 052C CD02DE            CALL  TSTAZ
0211 052F D23305            JNC   OUTB.1   '$' soll nicht veraendert werden
0212 0532 B3                ORA   E
0213 0533 CD3C05   OUTB.1   CALL  POUT
0214 0536 23                INX   H
0215 0537 15                DCR   D
0216 0538 C22B05            JNZ   OUTB.0
0217 053B C9                RET
0218 053C          ;
0219 053C          * Ausgabe des Zeichens im AkKu an DCE-Bus und Bildschirm
0220 053C F5       POUT     PUSH  PSW
0221 053D 3A02FE   POUT.1   LDA   :FE02
0222 0540 A7                ANA   A
0223 0541 F23D05            JP    POUT.1
0224 0544 F1                POP   PSW
0225 0545 3200FE            STA   :FE00
0226 0548 EF03              RSTD  5,3
0227 054A C9                RET
0228 054B          ;
0229 054B          * Gibt die ESC-Tabelle,deren Startadresse in HL uebergeben wird,aus
0230 054B 7E       OUTLIST  MOV   A,M
0231 054C A7                ANA   A
0232 054D C8                RZ            Tabellenende mit '0' gekenzeichnet
0233 054E CD3C05            CALL  POUT
0234 0551 23                INX   H
0235 0552 C34B05            JMP   OUTLIST
0236 0555                   *
0237 0555          * Die in den folgenden drei Tabellen zusammengefassten ESC-Sequenzen
0238 0555          * sind fuer den ITOH 8510 gedacht.
0239 0555 1B45     TAB1     DT    1B,"E"   waehle 96 Zeichen pro Zeile
0240 0557 1B4C3030          DT    1B,"L","0","0","4"  setzte linken Rand
0241 055C 1B41              DT    1B,"A"   Zeilenabstand
0242 055E 1B30              DT    1B,"0"   Keine TAB's
0243 0560 0D00              DT    0D,0
0244 0562 1B5800   TAB2     DT    1B,"X",0  ESC-Sequenz: 'Starte Unterstreichen'
0245 0565 1B5900   TAB3     DT    1B,"Y",0  ESC-Sequenz: 'Beende Unterstreichen'
0246 0568          ;
0247 0568          ********************************************************************
0248 0568          *          Entry: HL: Startadresse der Tabelle                     *
0249 0568          *                 E: Anzahl der Bytes hinter dem Wort + 1          *
0250 0568          *                 D: Laenge des gelesenen Wortes                   *
0251 0568          *          Exit: carry set: nicht gefunden                         *
0252 0568          *                carry reset: gefunden                             *
```

```
0253 0568          *                HL: Zeigt auf Laengenbyte in der Tabelle          *
0254 0568          ************************************************************************
0255 0568          ;
0256 0568 7E       TABLE    MOV   A,M      (A)=Laenge des Wortes in der Tabelle
0257 0569 A7                ANA   A
0258 056A CA9505            JZ    EXTABN   (A)=0 dann Tabellenende
0259 056D BA                CMP   D        Laege des Wortes im Puffer
0260 056E CA9A05            JZ    EQLEN    Laengen sind gleich
0261 0571 CD30DE   RETEQ    CALL  ADDA     suche naechsten Tabelleneintrag
0262 0574 7B                MOV   A,E
0263 0575 A7                ANA   A
0264 0576 C28F05            JNZ   LOPE.1
0265 0579                   *
0266 0579 23       LOPFUNC  INX   H        (E)=0 dann wird die Funktionstabelle untersucht.
0267 057A 7E                MOV   A,M      Diese ist anders angeordnet als die anderen
0268 057B E60F              ANI   :0F      Tabellen. Das finden einer '0' bedeutet nicht
0269 057D A7                ANA   A        unbedingt Tabellenende, und die Anzahl der hin-
0270 057E C27905            JNZ   LOPFUNC  ter einem Wort folgenden Bytes ist nicht immer
0271 0581 23       LOPF.1   INX   H        gleich.
0272 0582 7E                MOV   A,M
0273 0583 A7                ANA   A
0274 0584 CA9505            JZ    EXTABN
0275 0587 E60F              ANI   :0F
0276 0589 C26805            JNZ   TABLE
0277 058C C38105            JMP   LOPF.1
0278 058F                   *
0279 058F CD30DE   LOPE.1   CALL  ADDA     findet naechsten Tabelleneintrag bei 'normalen'
0280 0592 C36805            JMP   TABLE    Tabellen durch zusaetzliche Addition von (E).
0281 0595 37       EXTABN   STC
0282 0596 C9                RET            nichts gefunden
0283 0597 37       EXTABJ   STC
0284 0598 3F                CMC
0285 0599 C9                RET            BASIC-Wort gefunden
0286 059A                   *
0287 059A          * Wird angesprungen wenn die Laengenbytes uebereinstimmen.
0288 059A 01F901   EQLEN    LXI   B,BUFFER
0289 059D D5                PUSH  D
0290 059E E5                PUSH  H
0291 059F 23       LOPEQ    INX   H
0292 05A0 0A                LDAX  B
0293 05A1 BE                CMP   M
0294 05A2 C2AD05            JNZ   EXEQN
0295 05A5 15                DCR   D        alle Buchstaben untersucht und
0296 05A6 CAB305            JZ    EXEQJ    Uebereinstimmung gefunden
0297 05A9 03                INX   B
0298 05AA C39F05            JMP   LOPEQ
0299 05AD E1       EXEQN    POP   H
0300 05AE D1                POP   D
0301 05AF 7E                MOV   A,M
0302 05B0 C37105            JMP   RETEQ
0303 05B3 E1       EXEQJ    POP   H
0304 05B4 D1                POP   D
0305 05B5 C39705            JMP   EXTABJ
0306 05B8          ;
0307 05B8          * Worte, die in keiner Tabelle stehen
0308 05B8 02544F04 BEFLST   DT    2,"TO",4,"THEN",0
0309 05C1          *  Worte, die als zweiter Teil eines zusammengesetzten Wortes auf-
0310 05C1          *  treten koennen.
0311 05C1 034F4646 SPEZTAB  DT    3,"OFF",4,"TIME",3,"MEM",0
```

```
****************************
* S Y M B O L   T A B L E *
****************************

ADDA     DE30   BAS.WRD 050A   BEFLST  05B8   BUFFER  01F9
BUFFLEN 01F8   BUFFORT 01F6   BUFINIT 0423   CHANGE  0486
COMPDE  DE14   COMTAB  CBBF   EQLEN   059A   EXEQJ   05B3
EXEQN   05AD   EXTABJ  0597   EXTABN  0595   FIND22  048C
FINDBST 045F   FINDREM 0473   FNDDOPP 046E   FUNCTAB CFE6
INIT    0400   LOPE.1  058F   LOPEQ   059F   LOPF.1  0581
LOPFUNC 0579   MODE    01F5   NOBASIC 051B   OPTAB   CF91
OUTB.0  052B   OUTB.1  0533   OUTBUFF 0528   OUTLIST 054B
POUT    053C   POUT.1  053D   PRINT   042E   PRNT.1  04AE
PRNTDCE 0491   PRNTREM 0478   RETEQ   0571   SPEZTAB 05C1
TAB1    0555   TAB2    0562   TAB3    0565   TABLE   0568
TSTAZ   DE02


0035 0400 AF 32 F5 01 21 2E 04 22 DE 02 3E C3 32 DD 02 3E
0041 0410 03 32 31 01 21 03 FE 36 A1 2B 2B 36 02 21 55 05
0049 0420 CD 4B 05 AF 32 F8 01 21 F9 01 22 F6 01 C9 E5 D5
0066 0430 C5 F5 5F 2A F6 01 3A F8 01 57 3A F5 01 E6 02 C2
0074 0440 78 04 7B FE 22 CA 8C 04 3A F5 01 E6 01 C2 91 04
0081 0450 7B CD 02 DE DA 5F 04 FE 24 CA 5F 04 C3 91 04 77
0093 0460 23 22 F6 01 14 7A 32 F8 01 F1 C1 D1 E1 C9 3E 04
0105 0470 C3 86 04 3E 02 C3 86 04 7B FE 0D C2 91 04 3E 02
0114 0480 CD 86 04 C3 91 04 21 F5 01 AE 77 C9 3E 01 CD 86
0123 0490 04 7A A7 CA 1B 05 3A F5 01 E6 04 C2 AE 04 3E 04
0143 04A0 CD 86 04 21 C1 05 1E 01 CD 68 05 D2 0A 05 21 B8
0148 04B0 05 1E 01 CD 68 05 D2 0A 05 21 BF CB 1E 04 CD 68
0154 04C0 05 F5 D5 11 CB CC CD 14 DE CC 73 04 11 4A CC CD
0166 04D0 14 DE CC 6E 04 11 32 CC CD 14 DE CC 6E 04 11 53
0171 04E0 CC CD 14 DE CC 6E 04 D1 F1 D2 0A 05 21 91 CF 1E
0179 04F0 02 CD 68 05 D2 0A 05 21 E6 CF 1E 00 CD 68 05 D2
0185 0500 0A 05 1E 00 CD 28 05 C3 1B 05 21 62 05 CD 4B 05
0192 0510 1E 20 CD 28 05 21 65 05 CD 4B 05 F1 F5 CD 3C 05
0200 0520 CD 23 04 F1 C1 D1 E1 C9 21 F9 01 7E CD 02 DE D2
0211 0530 33 05 B3 CD 3C 05 23 15 C2 28 05 C9 F5 3A 02 FE
0222 0540 A7 F2 3D 05 F1 32 00 FE EF 03 C9 7E A7 C8 CD 3C
0233 0550 05 23 C3 4B 05 1B 45 1B 4C 30 30 34 1B 41 1B 30
0243 0560 0D 00 1B 58 00 1B 59 00 7E A7 CA 95 05 BA CA 9A
0260 0570 05 CD 30 DE 7B A7 C2 8F 05 23 7E E6 0F A7 C2 79
0270 0580 05 23 7E A7 CA 95 05 E6 0F C2 68 05 C3 81 05 CD
0279 0590 30 DE C3 68 05 37 C9 37 3F C9 01 F9 01 D5 E5 23
0292 05A0 0A BE C2 AD 05 15 CA B3 05 03 C3 9F 05 E1 D1 7E
0302 05B0 C3 71 05 E1 D1 C3 97 05 02 54 4F 04 54 48 45 4E
0308 05C0 00 03 4F 46 46 04 54 49 4D 45 03 4D 45 4D 00
```

Beispiel für ein solches Listing

```
2     for X=0 to 20
10    T=9.0:mode 6:colorq 3 5 9 14
30    gosub 400:mode 4:gosub 400:mode 2:gosub 400:goto 10
40    for S=0 to 6
41    L=rnd(16):M=rnd(16):N=rnd(16):O=rnd(16)
42    colorq L M N O
44    for A=0 to 700:K=getc:if K=0 then next
50    if K=32 then gosub 202
55    if K>0 and K<>32 then gosub 200
60    next:return
200   L=9:M=L+1:N=L+3:O=L+2
202   for X=0 to 8
205   colorq L M N O:wait time T
210   colorq O L M N:wait time T
220   colorq N O L M:wait time T
225   colorq M N O L:wait time T:next
230   return
400   D=ymax:E=D/2:F=E+0.5:G=F/4:H=(ymax+1)/8
470   M=int(H*4-1):for X=0 to M:MD=int(xmax/2)
500   draw MD+X,0 MD-X,M*2 20+X mod 4
505   draw MD-X,0 MD+X,M*2 20+X mod 4
510   draw (MD-M),M+X MD+M,M-X 20+X mod 4
515   draw (MD-M),M-X MD+M,M+X 20+X mod 4
520   next:gosub 40:return
```

```
0001                          TITL          Break abfangen,  H.Tegethoff,150284
0002       **01F5             ORG   :1F5
0003 01F5 **0275              PROT  :275
0004 01F5
0005 01F5              * Dieses Programm ist eine Unterroutine, die einen
0006 01F5              * Break beim Load, Check, Rewind etc. abfaengt.
0007 01F5
0008 01F5              * Entry's   (generell fuer beliebige MP's) :
0009 01F5              * LDCUTON  : Kann bei LOAD etc. dazu benutzt werden, im
0010 01F5              *            Falle eines Break zu der Adresse LDBRKADR
0011 01F5              *            zu springen. Dort sollte der SP gesetzt
0012 01F5              *            werden (nicht notwendigerweise sofort,
0013 01F5              *            aber auf jeden Fall).
0014 01F5              * LDCUTONA : wie oben, im HL wird die Adresse LDBRKADR
0015 01F5              *            uebergeben.
0016 01F5              * LDCUTOFF : Nach erfolgreichem LOAD (auch LOADING
0017 01F5              *            ERROR, kein Break-Abbruch !!) aufrufen !
0018 01F5              * MDCRTST  : Testet, ob sich ein TOS im DAI befindet
0019 01F5              *            Exit: Z  : TOS im Fxxx-Bereich
0020 01F5              *                  NZ : kein TOS im DAI
0021 01F5
0022 01F5 222602       LDCUTONA  SHLD  LDBRKADR
0023 01F8 E5           LDCUTON   PUSH  H
0024 01F9 2A6E00                 LHLD  :6E
0025 01FC 220D02                 SHLD  MDCRV6C
0026 01FF 210702                 LXI   H,MDCRV6U
0027 0202 226E00                 SHLD  :6E
0028 0205 E1                     POP   H
0029 0206 C9                     RET
0030 0207 210F02       MDCRV6U   LXI   H,MDCRV6R
0031 020A E3                     XTHL
0032 020B E5                     PUSH  H
0033 020C **020D       MDCRV6C   EQU   $+1
0034 020C C30000                 JMP   x
0035 020F F5           MDCRV6R   PUSH  PSW
0036 0210 3AC402                 LDA   :2C4
0037 0213 B7                     ORA   A
0038 0214 C21902                 JNZ   MDCRV6B
0039 0217 F1                     POP   PSW
0040 0218 C9                     RET
0041 0219 CD2802       MDCRV6B   CALL  LDCUTOFF
0042 021C CDD402                 CALL  :2D4
0043 021F CD3102                 CALL  MDCRTST
0044 0222 CC1FF3                 CZ    :F31F
0045 0225 **0226       LDBRKADR  EQU   $+1
0046 0225 C30000                 JMP   x
0047 0228
0048 0228 E5           LDCUTOFF  PUSH  H
0049 0229 2A0D02                 LHLD  MDCRV6C
0050 022C 226E00                 SHLD  :6E
0051 022F E1                     POP   H
0052 0230 C9                     RET
0053 0231
0054 0231 211EF0       MDCRTST   LXI   H,:F01E
0055 0234 7E                     MOV   A,M
0056 0235 FEC3                   CPI   :C3
0057 0237 C0                     RNZ
0058 0238 2D                     DCR   L
0059 0239 2D                     DCR   L
0060 023A 2D                     DCR   L
0061 023B F23402                 JP    MDCRTST+3
0062 023E AF                     XRA   A
0063 023F C9                     RET
0064 0240
```

```
0065 0240            ; Aufrufmoeglichkeiten :
0066 0240
0067 0240            ; CHECK:  obiges MP ergaenzen :
0068 0240
0069 0240 210000     BREAK    LXI   H,0
0070 0243 220401              SHLD  :104      delete FOR's,
0071 0246 221301              SHLD  :113      delete GOSUB's,
0072 0249 3100F9              LXI   SP,:F900  weil Stack geloescht wird !
0073 024C 21C402              LXI   H,:2C4
0074 024F 36FF                MVI   M,:FF     Break-Flag loeschen
0075 0251 7E                  MOV   A,M       Warten auf no-Break
0076 0252 B7                  ORA   A
0077 0253 C25102              JNZ   $-2
0078 0256 CDE4CE              CALL  :CEE4     Bank 0 anwaehlen
0079 0259 0000                DBL   0
0080 025B 016102              LXI   B,GOTOz
0081 025E C37FC8              JMP   :C87F     run BASIC-Command 'GOTO z'
0082 0261 89         GOTOz    DATA  :89
0083 0262 2710                DBL   10000     <== Zeilennr, higher B. first
0084 0264
0085 0264            ; Aufruf :   POKE LDBRKADR,BREAK IAND 255
0086 0264            ;            POKE LDBRKADR+1,BREAK SHR 8
0087 0264            ;            CALLM LDCUTON
0088 0264            ;            CALLM :D2C3     (CHECK)
0089 0264            ;            CALLM LDCUTOFF
0090 0264
0091 0264            ; REWIND :   wie CHECK, ersetze CALLM :D2C3
0092 0264            ; LOADA :    wie REWIND, aber nach LOADING ERROR muss
0093 0264            ;            einmal die Break-Taste gedrueckt,
0094 0264            ;            => GOTO z, oder aber von Hand
0095 0264            ;            CALLM LDCUTOFF eingegeben werden
0096 0264            ;            Bei der BASIC V1.1 kommt nach ERROR manch-
0097 0264            ;            mal ein BREAK automatisch !
0098 0264            ; ..... :    .......
0099 0264
0100 0264            ; Wichtig : Das MP liegt im ENVELOPE-Speicher !

**************************
* S Y M B O L   T A B L E *
**************************

01F5 LDCUTONA       0022
01F8 LDCUTON  0023
0207 MDCRV6U  0030 0026
020D MDCRV6C  0033 0025 0049
020F MDCRV6R  0035 0030
0219 MDCRV6B  0041 0038
0226 LDBRKADR       0045 0022
0228 LDCUTOFF       0048 0041
0231 MDCRTST  0054 0043 0061
0240 BREAK    0069
0261 GOTOz    0082 0080

0022 01F5                22 26 02 E5 2A 6E 00 22 0D 02 21
0026 0200 07 02 22 6E 00 E1 C9 21 0F 02 E3 E5 C3 00 00 F5
0036 0210 3A C4 02 B7 C2 19 02 F1 C9 CD 28 02 CD D4 02 CD
0043 0220 31 02 CC 1F F3 C3 00 00 E5 2A 0D 02 22 6E 00 E1
0052 0230 C9 21 1E F0 7E FE C3 C0 2D 2D 2D F2 34 02 AF C9
0069 0240 21 00 00 22 04 01 22 13 01 31 00 F9 21 C4 02 36
0074 0250 FF 7E B7 C2 51 02 CD E4 CE 00 00 01 61 02 C3 7F
0081 0260 C8 89 27 10
```

```
PAGE 01    BILDSCHIRM-COPY

002                    *** GESCHRIEBEN VON REINHALD HOCHE
003                    *** COPIERT BILDSCHIRM IM MODE 0
004                    *** UEBER RS-232 AUF DEN DRUCKER
005                            ORG    :300
006 0300 F5                    PUSH   PSW
007 0301 C5                    PUSH   B
008 0302 E5                    PUSH   H
009 0303 21E9BF                LXI    H.:BFE9    ANFANG VIDEO-RAM
010 0306 0618                  MVI    B.24       ZEILENZAEHLER
011 0308 0E7A          NEUZEI  MVI    C.122      POSITIONENZAHLER
012 030A CD2503        DRUCKE  CALL   DRUCK
013 030D 2B                    DCX    H
014 030E 0D                    DCR    C
015 030F C20A03                JNZ    DRUCKE
016 0312 3E0D                  MVI    A.13
017 0314 CD2603                CALL   CR         SENDE CR
018 0317 05                    DCR    B
019 0318 CA3003                JZ     OVER
020 031B 3E0C                  MVI    A.12
021 031D 2B            MINUS   DCX    H
022 031E 3D                    DCR    A
023 031F C21D03                JNZ    MINUS
024 0322 C30803                JMP    NEUZEI
025 0325 7E            DRUCK   MOV    A.M
026 0326 CD94DD        CR      CALL   :DD94
027 0329 3E64                  MVI    A.100      WARTE AUF DRUCKER
028 032B 3D            WARTE   DCR    A          ZAEHLE ACCU
029 032C C22B03                JNZ    WARTE      VON 100 BIS 0
030 032F C9                    RET
031 0330 E1            OVER    POP    H
032 0331 C1                    POP    B
033 0332 F1                    POP    PSW
034 0333 C9                    RET
035 0334               ENDE    END

*****************************
* S Y M B O L   T A B L E *
*****************************

CR      0326   DRUCK  0325   DRUCKE 030A   ENDE   0334
MINUS   031D   NEUZEI 0308   OVER   0330   WARTE  032B
```

Höchst,den 13.10.1983

RESTART

Ein Programm zum Wiederauffinden von Basicprogrammen nach 'Reset'.
Das Programm ist für den Speicherbereich von #5000 bis #5055 ausgelegt,da dieses ein Bereich ist,der normalerweise weder durch Systemneuinitialisierung,noch durch die verschiedenen Graphikmodi,noch durch 'EDIT',noch durch kurze bis mittellange Basicprogramme überschrieben wird.(Bei längeren Programmen sollte man sich dennoch besser davon vergewissern.)
Zum Verschieben sind die unterstrichenen Daten zu ändern.
Das Programm leitet den Interruptvektor 1 um.Dieser muß deshalb am Anfang ein Mal initialisiert werden.
(Entweder durch UT;V1 C70E-5000;B oder durch POKE #64,0: POKE #65,#50) EDIT BREAK RETURN
Nach einem 'Reset' ist dann ein 'CALL #503C' durch zu führen.Dieser Aufruf setzt den Vektor 1 automatisch wieder auf #5000.
Danach kann wieder beliebig gelistet,editiert,gestartet und sonstiges gemacht werden.
Achtung:

1.) Das Programm wirkt nicht bei 'NEW'. Return direkt
2.) Nach der letzten Programmänderung ist mindestens ein Befehl ('LIST','PRINT','RUN',o.ä.) ein zu geben.

Hexadezimal-Dump:

```
5000  C5 D5 E5 F5 0E 01 21 9B 02 11 16 50 CD 22 50 F1
5010  E1 D1 C1 C3 0E C7 00 00 00 00 00 00 00 00 00 00
5020  00 00 06 0A 7E 12 13 23 05 C2 24 50 0C 21 32 01
5030  0D C2 35 50 EB 7E 12 13 23 7E 12 C9 C5 D5 E5 F5
5040  21 00 50 22 64 00 0E 00 11 9B 02 21 16 50 CD 22
5050  50 F1 E1 D1 C1 C9
```

Anmerkung:Die Speicherplätze von #5016 bis#5021 sind Datenspeicher.

Volker Raab

NP 66

Z A H L E N S Y S T E M E

```
65400 REM Umrechnung von Werten in Zahlensystemen mit
65401 REM beliebigen Basen (z.B. DUAL-2,OKTAL-8,DEZIMAL-10,    HEXADE
ZIMAL-16 usw.)
65402 CLEAR 1000:PRINT CHR$(12):PRINT "     BASIS-TRANSFORMATION":PRI
NT
65403 INPUT "     BASIS 1 = ";B1%:INPUT "   =>   BASIS 2 - ";B2%:PRIN
T :PRINT
65404 INPUT "     ";A$:PRINT " (";B1%;")";
65406 B%=0:FOR I%=0 TO LEN(A$)-1:G$=MID$(A$,LEN(A$)-I%-1,1):IF ASC(G$
)>57 THEN G%=ASC(G$)-55:GOTO 65408
65407 G%=VAL(G$)
65408 IF G%>B1%-1 GOTO 65424
65410 B%=B%+INT(G%*B1%^I%+0.5):NEXT
65412 C$=" ":X%=B%:I%=0
65414 K%=X% MOD B2%:F%=0:IF K%>9 THEN F%=7
65416 C$=CHR$(48+K%+F%)+C$:X%=INT(X%/B2%):I%=I%+1:IF X%<>0 GOTO 65414
65418 PRINT TAB(23);" => ";B%;"(10)";TAB(38);" => ";C$;"(";B2%;")"
65420 GOTO 65404
65424 PRINT "  FALSCHE EINGABE. - BITTE WIEDERHOLEN":GOTO 65404
```

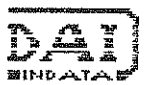
F A K U L T Ä T

```
190   PRINT CHR$(12):PRINT "     F A K U L T A E T  nach STIRLING":PR
INT
200   INPUT "       N = ";N:IF N<20.0 GOTO 500
210   NF=LOGT(SQR(2.0*PI))+(N+0.5)*LOGT(N)+LOGT(EXP(1.0))*(-N)
220   E%=INT(NF):M=ALOG(FRAC(NF)):PRINT TAB(25);"N! = ";M;TAB(38);" *
 10 ^";E%:GOTO 200
500   M=1.0:FOR I%=1 TO N:M=I%*M:NEXT:NF=LOGT(M):GOTO 220
```

H E X - D U M P M I T A S C I I

Mit diesem Programm kann man Speicherbereiche mit ASCII-Zeichen auslisten. Taste=STOP, Space=WEITER HL:=von, DE:=bis, G9000

```
9000  C5 7D E6 F0 6F 7B F6 0F 5F CD 53 90 DA 51 90 CD
9010  14 DE D2 51 90 E5 7C CD 6C 90 7D CD 6C 90 06 04
9020  3E 20 CD 60 DD 0E 04 7E 23 CD 6C 90 0D C2 27 90
9030  05 C2 20 90 E1 06 04 3E 20 CD 60 DD 0E 04 7E 23
9040  CD 5F 90 CD 60 DD 0D C2 3E 90 05 C2 37 90 C3 09
9050  90 C1 C9 3E 0D CD 60 DD CD BB D6 C4 DA D6 C9 E6
9060  FF FE 20 DA 69 90 FE 7F D8 3E 2E C9 F5 0F 0F 0F
9070  0F CD 75 90 F1 E6 0F C6 90 27 CE 40 27 CD 60 DD
9080  C9
```



```
1       REM ********************************
2       REM * DAI LOGO  on  EPSON RX-80 *
3       REM * DUAL DENSITY  BIT GRAPHIC *
4       REM *  Daniel    R O S S I O N  *
5       REM *  An der Gerstenmuehle,11  *
6       REM *  D - 5160    D U E R E N  *
7       REM ********************************
10      RESTORE:GOTO 100
20      PRINT CHR$(X%);
30      RETURN
100     FOR A%=1 TO 4
110     PRINT CHR$(27);"L";
120     X%=98:GOSUB 20:X%=0:GOSUB 20
130     FOR I%=1 TO 49:READ X%:GOSUB 20:GOSUB 20:NEXT I%
140     PRINT CHR$(27);"A";CHR$(8)
150     NEXT A%
160     PRINT :PRINT :GOTO 10
200     END
1000    DATA 0,0,0,0,24,24,24,24,24,24,24,24,24,24
1010    DATA 24,24,24,24,24,24,24,24,24,24,24,24,24,24
1020    DATA 24,24,24,24,24,24,24,24,24,24,24,24,24,28
1030    DATA 14,7,3,0,0,0,0
1040    DATA 0,0,0,0,96,123,123,96,112,56,26,11,3,1
1050    DATA 0,0,0,0,0,3,27,56,96,96,120,123,27,3
1060    DATA 0,0,0,0,0,96,123,123,123,96,0,0,0,0
1070    DATA 0,255,255,0,0,0,0
1080    DATA 0,0,0,0,6,222,222,6,6,14,28,216,208,192
1090    DATA 2,6,30,26,208,208,208,208,208,208,208,208,214,222
1100    DATA 222,30,6,0,0,6,222,222,222,6,0,0,0,0
1110    DATA 0,255,255,0,0,0,0
1120    DATA 0,0,0,0,60,60,60,60,0,60,0,60,16,8
1130    DATA 60,0,60,36,24,0,4,12,24,40,24,12,4,32
1140    DATA 60,32,4,12,24,40,24,12,4,0,60,60,60,120
1150    DATA 240,224,192,0,0,0,0
```

HEX-INPUT

```
1       REM Komfortable Hexadezimal-Eingabe   Autor?????
2       REM Entspricht 'S' in der UTILITY
3       REM Zeile 36 eingefügt Bernd Preusing 4/84
4       REM Erlaubte Tasten:  Cursoren und Hex-Ziffern
5       REM IMP INT !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
6       CLEAR 17:MODE 0:INPUT "START";A
10      P=0:Y=0:IF A IAND #FFF0=A THEN PRINT :PRINT HEX$(A);
20      CX=6+3*(A IAND #F):CY=CURY+Y:Y=0:CURSOR CX,CY
25      FOR B=16 TO 1 STEP -15
30      C=GETC:IF C=0 THEN 30
32      IF GETC>0 THEN 32
35      IF C<16 OR C>19 THEN 60
36      IF B=1 THEN 30
38      ON C-15 GOTO 40,45,50,55
40      A=A-16:Y=+1:GOTO 20
45      A=A+16:Y=-1:GOTO 20
50      IF A IAND #FFF0<A THEN A=A-1:GOTO 20
55      P=PEEK(A):GOTO 100
60      IF C>47 AND C<58 THEN P=P+(C-48)*B:GOTO 90
65      IF C>64 AND C<71 THEN P=P+(C-55)*B:GOTO 90
80      GOTO 30
90      PRINT CHR$(C);:NEXT B
100     P$=HEX$(P):IF P<16 THEN LET P$="0"+P$
110     CURSOR CX,CY:PRINT P$;:POKE A,P:A=A+1:GOTO 10
```

S C R O L L

SCROLL ist ein kleines Maschinenprogramm, das in MODE 5 oder 6 bei jedem Aufruf den Bildschirminhalt um 8 Punkte nach links verschiebt, dabei gehen am linken Rand 8 Punkte verloren, am rechten Rand entstehen 8 leere Spalten.
Anwendung z.B. um einen Kursverlauf zu verschieben.

Autor: Ralf Degenhardt

Bei einer Programmverschiebung zu ändernde Adressen sind unterstrichen.

```
0300 F5 C5 D5 E5 3E 2A 11 43 66 21 42 66 E5 D5 46 23
0310 23 70 EB 46 23 23 70 EB 3D CA 30 03 1B 1B 1B 1B
0320 2B 2B 2B 2B C3 0E 03 3A 50 4F 4B 45 20 4E 25 2C
0330 01 5A 00 E1 09 EB E1 09 E5 D5 3E BF BC FA 45 03
0340 3E 2A C3 0E 03 E1 D1 E1 D1 C1 F1 C9
```

ML-Unterprogramme des INDATA-Demo

Selbst eingefleischte Apple- oder Commodore-Fans werden blaß, wenn wenn sie das Demo laufen sehen. Auch ich lege die Disketten ab und zu ein, wenn ich eine Wut auf diese 'Schrottkiste' habe. Der Haken liegt nämlich in der Software: damit läßt sich doch fast alles machen!

Das gesamte Demo ist einfach zu umfangreich, um es hier in der Zeitung zu veröffentlichen, aber ich habe mir einmal die Mühe gemacht, wenigstens den ML-Teil zu untersuchen.

Dieser Teil besteht hauptsächlich aus sehr schönen Grafik-Utilities und einem Sound-Teil, der auch Musik während des Programmlaufs oder bei der Floppy sogar während des Ladens von Programmen (weil da aus Timing-Gründen die Interrupts nicht gesperrt zu sein brauchen).

Zunächst wird hier eine recht interessante Methode der Parameterübergabe an Maschinenprogramme demonstriert. Da wird nicht vor jedem Aufruf ein Wert irgendwohin gepoket oder mehrere Werte in eine Variable 'geschiftet', sondern im gesamten Programm nur ein einziges Mal die Adresse(n) der Variablen übergeben, aus der sich dann das ML-Programm beim Aufruf den Wert holt;

CALLM #1000,X% übergibt die Adresse von X% (der Name ist natürlich beliebig), aus X% werden allerdings nur die beiden letzten Bytes genommen, d.h. ihr Wert darf max #FFFF sein.

CALLM #1003,Y% ist analog dazu, hier wird nur ein Byte benutzt.

CALLM #1006,R% s.o., hier werden zwar 2 Byte vorbereitet, aber im Endeffekt wird von allen Routinen nur 1 Byte benutzt.

X% und Y% enthalten jeweils die X- und Y-Koordinaten, an denen etwas 'passieren' soll, R% wird nur von COPY und DOUBLE gebraucht und übergibt die gewünschte Größe des Grafik-Feldes.

Alle Programme laufen in allen Modi, jedoch muß darauf geachtet werden, daß die X%- und Y%-Werte für den Mode nicht zu groß sind, bebesonders im A-Mode darf Y nicht größer als der sichtbare Teil sein!

CALLM #1009 merkt sich für COPY und DOUBLE die Koordinaten des alten Bereiches und R% muß für beide Programme die Höhe enthalten (Anzahl der Zeilen). Für COPY, DOUBLE und Sprite legt #1009 auch die 'Richtung' der Operation fest. Dazu muß vorher in #1075 ein Wert eingeschrieben werden: =0: von unten nach oben, also X,Y gibt die die linke, untere Ecke an, <>0: von oben nach unten; X,Y ist die linke, obere Ecke.

Bei allen Routinen außer SPRITE2 wird immer nur entweder das Low- oder High-Byte des Bildschirm-RAM verändert. Die Auswahl erfolgt mit POKE #1236,3 (ergibt High-Byte) oder POKE #1236,4 (Low-Byte). Will man also alle 4 oder gar 16 Farben haben, muß jede Operation zweimal ausgeführt werden (mit verschiedenen (#1236)).

Zu den einzelnen Routinen:
dabei sei jeweils:

X1%,Y1%	:	Koordinaten der Quelle
H%	:	Höhe der Quelle (Anz. Zeilen)
B%	:	Breite der Quelle (Anz.punkte)/8
X2%,Y2%	:	Koordinaten des Ziels

COPY

Kopiert einen Grafik-Bereich an eine andere Stelle.
Vor dem ersten Aufruf mind. einmal:

```
CALLM #1000,X%:CALLM #1003,Y%:CALLM #1006,R%
```

Aufruf:

```
X%=X1%:Y%=Y1%:R%=H%:CALLM #1009
X%=X2%:Y%=Y2%:R%=B%:CALLM #1012
```

DOUBLE

Zeichnet einen Grafik-Bereich an anderer Stelle doppelt so groß.
Vor dem ersten Aufruf mind. einmal:
```
CALLM #1000,X%:CALLM #1003,Y%:CALLM #1006,R%
```

Aufruf: X%=X1%:Y%=Y1%:R%=H%:CALLM #1009
X%=X2%:Y%=Y2%:R%=B%:CALLM #100F

SPRITE

Bringt ein Sprite aus einer String-Variablen auf den Schirm.
Vor dem ersten Aufruf mind. einmal:
```
CALLM #1000,X%:CALLM #1003,Y%:CALLM #1009
```
Dann POKE #10F3,Breite in Bytes: POKE #1105,Ende-Kennung
(Die Ende-Kennung darf sonst im String nicht vorkommen!)
Aufruf: X%=X2%:Y%=Y2%:CALLM #100C

SPRITE2

Bewegt Grafik-Bereiche von und zu einem INT-Array.
Läßt sich gut benutzen, um z.B. Grafiken von einem MODE in einen anderen zu bringen, oder einfach wie SPRITE verwenden. Im Gegensatz zu SPRITE werden hier alle Bytes genommen (also alle Farben).
Dieser Teil wurde von mir aus einem anderen Demo genommen und im Assembler-Listing angehängt.
Der Array kann belibig viele Dimensionen besitzen (mind. 2), wobei die letzte Dimension(+1) gibt die Höhe an, die vorletzte(+1) die Breite in 16Punkt. Z.B. ein Array DIM A%(4,2,31) enthält 5 Sprites (0..4), ist 3*16=48 Punkte breit und 32 Punkte hoch.
X%,Y% gibt hier immer die linke, obere Ecke an!
Vor dem ersten Aufruf mind. einmal:
```
DIM A%(...):CALLM #1000,X%:CALLM #1003,Y%:CALLM #14E0,A%(0,0..,0)
```

Aus Array auf Schirm:
```
Aufruf:  CALLM #14F1,A%(..,0,0)
```
Von Schirm in Array:
```
Aufruf:  CALLM #14F8,A%(..,0,0)
```

SOUND

macht interruptgetriebene Musik.

Die 'Musik' muß dabei irgendwo im Speicher leigen. Format:

Jeder Ton besteht aus 3 Byte:
1.Byte: Frequenz Low-Byte
2.Byte: Frequenz High-Byte
3.Byte: Dauer in 20 msec

Die Frequenz-Bytes können z.B. mit der Funktion FREQ() ermittelt werden, wobei der Wert in Klammern die Frequenz in Hertz ist.
Sonderfälle:
1.+2.Byte=0: Pause
3.Byte=#FF : Ende der Daten (SOUND OFF)

Vor der Benutzung muß in #1073 die Anzahl der Stimmen (Kanäle) eingetragen werden.
Aufruf: ADR%=Startadresse der Daten:CALLM #1015,ADR%

Die restlichen Routinen sind nicht so allgemein verwendbar, ihren Sinn und ihre Verwendung entnehmt bitte aus dem Ass-Listing!

```
0000                    TITL       ROUTINEN DES INDATA-DEMOS'
0000            ;
0000            ; DISASSEMBLIERT UND KOMMENTIERT VON BERND PREUSING 4/84
0000            ;
0000  §=0040 POROM   EQU        40H             ;MERKER F. OUTPUT-PORT
0000  §=006E I6USA   EQU        6EH             ;VEKTOR INTERRUPT 6
0000  §=0070 I7USA   EQU        70H             ;VEKTOR INTERRUPT 7
0000  §=0074 CURTYP  EQU        74H             ;CURSOR-FLAG
0000  §=0075 CURIN   EQU        75H             ;CURSOR-INHALT (ZEICHEN)
0000  §=0088 GRE     EQU        88H             ;ZEIGER: ENDE DES GRAFIK-BEREICHES
0000  §=0098 LOL     EQU        98H             ;GRAFIK: ANZAHL BYTES PRO ZEILE
0000  §=0100 CURRNT  EQU        100H            ;BASIC: ADR NEXT LINE
0000  §=013E EBUF    EQU        13EH
0000  §=0141 LH0141  EQU        141H
0000  §=01BE TIMER   EQU        1BEH            ;TIMER FÜR WAIT TIME
0000  §=0297 CMDTAB  EQU        297H            ;ENTHALT IM DOS EINEN ZEIGER
0000            ;                               ;AUF DIE KOMMANDO-TABELLE
0000  §=029B HEAP    EQU        29BH            ;ANFANG DES HEAP
0000  §=029F TXTBGN  EQU        29FH            ;ANF DES BAS-PROGS
0000  §=02A1 STBBGN  EQU        2A1H            ;ANF DER SYMBOL-TABELLE
0000  §=02A3 STBUSE  EQU        2A3H            ;ENDE DER -----"----
0000  §=02A5 SCRBOT  EQU        2A5H            ;ANF DES VIDEO-RAM
0000  §=02B9 KNSCAN  EQU        2B9H            ;FLAG: NUR BREAK ODER ALLE
0000  §=02BA KLIND   EQU        2BAH            ;TASTATURPUFFER
0000  §=02C0 KLIDU   EQU        2C0H            ;ZEIGER AUF NACHSTES ZEICHEN
0000            ;                               ;DARIN
0000  §=02C4 KBRFL   EQU        2C4H
0000  §=02CE ROPEN   EQU        2CEH            ;ÖFFNE FILE
0000  §=02D1 RBLK    EQU        2D1H            ;LIES BLOCK VON FILE
0000  §=02D4 RCLOSE  EQU        2D4H            ;SCHLIESSE FILE
0000  §=02E5 BASFLG  EQU        2E5H            ;WIRD VON DIESEM PROGRAMM AUF
0000            ;                               ;#FF GESETZT, WENN EIN BASIC-
0000            ;                               ;PROGRAMM DURCH DAS MENÜ GE-
0000            ;                               ;LADEN WURDE. DIESES KANN DANN
0000            ;                               ;VOM PROG BENUTZT WERDEN
0000  §=A000 LHA000  EQU        0A000H          ;START DES BLOCKS IM RAM
0000  §=A4E0 LHA4E0  EQU        0A4E0H          ;ENDE DES BLOCKS IM RAM
0000  §=D578 RST6    EQU        0D578H          ;NORM. KEYBOARD-INTR RST 6
0000  §=D6BE GETC    EQU        0D6BEH          ;HOLE EIN ZEICHEN VON TAST
0000  §=D9A9 RST7    EQU        0D9A9H          ;NORM. CLOCK-INTR RST 7
0000  §=DAD4 PRTMSG  EQU        0DAD4H          ;PRINT MESSAGE (HL...)
0000  §=DE30 ADDA    EQU        0DE30H          ;HL:=HL+A
0000  §=DE4F MOVEBL  EQU        0DE4FH          ;MOVE DE..HL NACH BC
0000  §=DE8F MULA    EQU        0DE8FH          ;HL:=A*HL
0000  §=DEB0 NEW256  EQU        0DEB0H          ;CLEAR 256:NEW IN BASIC V1.1
0000  §=DF9E RUNBAS  EQU        0DF9EH          ;RUN BASIC PROG
0000  §=E019 WTTIM1  EQU        0E019H          ;IN 'WAIT TIME'
0000  §=F8FF STACK1  EQU        0F8FFH          ;STACK  (??????)
0000  §=F900 STACK   EQU        0F900H          ;INIT SP
0000  §=FC00 SNDAD   EQU        0FC00H          ;BASIS-ADRESSE 8253 (SOUND)
0000  §=FC06 SNDC    EQU        0FC06H          ;KOMMANDO 8253
0000  §=FD06 PORO    EQU        0FD06H          ;OUTPUT-PORT
0000            ;
0000            ;
0000  §=1000 START   EQU        1000H
0000            ;
0000                    ORG        START
1000            ;
1000            ;   SPRUNG-VEKTOREN:
1000            ;
1000  C32110 LH1000  JMP        SETX            ;',VAR%' ÜBERGEBE X%-ADRESSE
1003  C32710 LH1003  JMP        SETY            ;',VAR%' ÜBERGEBE Y%-ADRESSE
1006  C32E10 LH1006  JMP        SETR            ;',VAR%' ÜBERGEBE R%-ADRESSE
```

```
1000  C33410 LH1000  JMP       SETPAR          ;         PARAMETER ABLEGEN
100C  C3E710 LH100C  JMP       SPRITE          ; ',VAR$' 'SPRITE' AUF SCHIRM
100F  C32111 LH100F  JMP       DOUBLE          ;         BEREICH VERGRöSSERN
1012  C39D10 LH1012  JMP       COPY            ;         BEREICH KOPIEREN
1015  C38A11 LH1015  JMP       SOUND           ; ',ADR%' STARTE SOUND
1018  C37710         JMP       INTR.6          ;INTR6 FÜR TASTATUR
101B  C36612 LH101B  JMP       LOADUT          ; ',VAR$' LADE FILE
101E  C38CA2         JMP       0A28CH          ;UNSINN
1021          ;
1021          ;   AUFRUF VON BASIC: CALLM #1000,X%  (BELIEBIGE INT-VARIABLE)
1021          ;   MERKT SICH DIE ADRESSE DER VARIABLEN (2 BYTE)
1021          ;
1021  23      SETX     INX H                  ;2 BYTE ÜBERSPRINGEN
1022  23               INX H
1023  226910           SHLD      PTR.X        ;ADRESSE MERKEN
1026  C9               RET
1027          ;
1027          ;   AUFRUF VOM BASIC: CALLM #1003,Y%  (BELIEBIGE INT-VARIABLE)
1027          ;   MERKT SICH DIE ADRESSE DER VARIABLEN (1 BYTE)
1027          ;
1027  23      SETY     INX H                  ;3 BYTE ÜBERSPRINGEN
1028  23               INX H
1029  23               INX H
102A  226B10           SHLD      PTR.Y        ;ADRESSE MERKEN
102D  C9               RET
102E          ;
102E          ; SIEHE SETX          CALLM #1006,R%
102E          ;
102E  23      SETR     INX H
102F  23               INX H
1030  226D10           SHLD      PTR.R
1033  C9               RET
1034          ;
1034          ;   AUFRUF VOM BASIC: CALLM #1009
1034          ;   PARAMETER-FESTLEGUNG FÜR 'SPRITE' 'COPY' UND 'DOUBLE'
1034          ;
1034  F5      SETPAR   PUSH PSW
1035  E5               PUSH H
1036  D5               PUSH D
1037  C5               PUSH B
1038  CD1712           CALL      CLCADR         ;BERECHNE MOM. SCREEN-ADRESSE
103B  227110           SHLD      SCRADR         ;MERKEN
103E  2A6D10           LHLD      PTR.R          ;ZEIGER AUF R%
1041  23               INX H                    ;BRAUCHT NUR 1 BYTE
1042  7E               MOV A,M                  ;HOLEN...
1043  327410           STA       R              ;...UND MERKEN
1046  2A9800           LHLD      LOL            ;BYTES PRO GRAFIK-ZEILE IN L
1049  2600             MVI H     0H
104B  3A7510           LDA       DIRFLG         ;FLAG UP, DOWN
104E  B7               ORA A
104F  CA5810           JZ        SETPR1         ;VON UNTEN NACH OBEN
1052  7D               MOV A,L                  ;HL:=0-HL
1053  2F               CMA
1054  6F               MOV L,A
1055  26FF             MVI H     0FFH
1057  23               INX H
1058  226710 SETPR1    SHLD      LINOFF         ;ZEILEN-OFFSET MERKEN
105B  C1               POP B
105C  D1               POP D
105D  E1               POP H
105E  F1               POP PSW
105F  C9               RET
1060
```

```
1060            ; VARIABLE:
1060            ;
1060                    DS      2H              ;NICHT BENUTZT
1062            PTRSND  DS      2H              ;ENTHÄLT DIE NÄCHSTE MUSIK-ADRESSE
1064            TIMERS  DS      3H              ;JE TON-KANAL 1 TIMER
1067            LINOFF  DS      1H              ;LOW-BYTE: BYTES/ZEILE
1068            LINOFH  DS      1H              ;HIGH-BYTE: ---"----
1069            PTR.X   DS      2H              ;ZEIGER AUF X% (INHALT 2 BYTE)
106B            PTR.Y   DS      2H              ;ZEIGER AUF Y% (INHALT 1 BYTE)
106D            PTR.R   DS      2H              ;ZEIGER AUF R% (INHALT 2 BYTE)
106F            DSTADR  DS      2H              ;2. MERKER F. SCREEN-ADR.
1071            SCRADR  DS      2H              ;1. ----------"----------
1073  00        CHNCNT  DB      0H              ;ENTHÄLT ANZAHL DER SOUND-KANÄLE
1074            R       DS      1H              ;MERKER FÜR R%-WERT
1075            ;
1075            ; RICHTUNGS-FLAG: VOM BASIC AUS EINPOKEN
1075            ; =0: GRAFIK LÄUFT VON UNTEN NACH OBEN, X%,Y%=ECKE LINKS,UNTEN
1075            ;<>0: GRAFIK LÄUFT VON OBEN NACH UNTEN, X%,Y%=ECKE LINKS,OBEN
1075  00        DIRFLG  DB      0H
1076            ;
1076  00        MENFLG  DB      0H              ;FLAG, OB BEI TASTENDRUCK DAS
1077            ;                               ;MENÜ ERSCHEINEN SOLL (0=NEIN)
1077            ;
1077            ;
1077            ;   INTERRUPT 6 (TASTATUR-ABFRAGE)
1077            ;   UNTERDRÜCKT BREAK UND SPRINGT BEI GEDRÜCKTER TASTE UND
1077            ;   GESETZTEM FLAG ZUM MENÜ
1077            ;
1077  E1        INTR.6  POP H                   ;STACK KORRIGIEREN (V6)
1078  F5                PUSH PSW
1079  AF                XRA A                 ;ALLE TASTEN ABFRAGEN
107A  32B902            STA     KNSCAN
107D  F1                POP PSW
107E  F5                PUSH PSW
107F  CD9910            CALL    LH1099        ;ABFRAGE (SCAN= NORM. RST 6)
1082  AF                XRA A                 ;VERNICHTE EVTL. BREAK
1083  32C402            STA     KBRFL
1086  3A7610            LDA     MENFLG        ;MENÜ ERLAUBT?
1089  B7                ORA A
108A  CA9710            JZ      LH1097        ;NEIN, ENDE INTR
108D  E5                PUSH H
108E  2AC002            LHLD    KLIOU         ;IST TASTE GEDRÜCKT?
1091  7E                MOV A,M
1092  B7                ORA A
1093  C2C312            JNZ     MENUE         ;JA, ALSO MENÜ
1096  E1                POP H
1097  F1        LH1097  POP PSW
1098  C9                RET
1099            ;
1099  E5        LH1099  PUSH H                  ;FÜHRE NORM. RST6 AUS
109A  C378D5            JMP     RST6
109D            ;
109D            ;
109D            ;           KOPIERE GRAFIK-BEREICH
109D            ;
109D            ; AUFRUF:
109D            ;  1. QUELLE ÜBERGEBEN:
109D            ;       X%=..,Y%=..,R%=(ANZAHL GRAFIK-ZEILEN), CALLM #1009
109D            ;  2. ZIEL ÜBERGEBEN:
109D            ;       X%=..,Y%=..,R%=(ANZ. HORIZONTAL-PUNKTE)/8, CALLM #1012
109D            ;
109D  F5        COPY    PUSH PSW
109E  E5                PUSH H
```

```
109F  D5              PUSH D
10A0  C5              PUSH B
10A1  CD1712          CALL      CLCADR          ;BERECHNE ZIEL-ADRESSE
10A4  226F10          SHLD      DSTADR          ;MERKEN
10A7  EB              XCHG
10A8  2A6D10          LHLD      PTR.R
10AB  23              INX H
10AC  7E              MOV A,M                   ;R%
10AD  87              ADD A                     ;*2 = ANZAHL BYTES PRO ZEILE
10AE  4F              MOV C,A
10AF  47              MOV B,A
10B0  2A7110          LHLD      SCRADR          ;ADRESSE DER QUELLE
10B3  EB              XCHG
10B4  1A      LH10B4  LDAX D
10B5  B6              ORA M                     ;DIESES BYTE (BEFEHL) EINPOKEN:
10B6          ;                                 ;ORA M (#B6)= + ALTES BILD BEI ZIEL
10B6          ;                                 ;NOP   (#00)= NUR NEUES BILD
10B6  77              MOV M,A
10B7  2B              DCX H
10B8  1B              DCX D
10B9  05              DCR B
10BA  C2B410          JNZ       LH10B4          ;WEITER IN DIESER ZEILE
10BD  C5              PUSH B
10BE  3A6710          LDA       LINOFF          ;BC=ZEILEN-OFFSET
10C1  4F              MOV C,A
10C2  3A6810          LDA       LINOFH
10C5  47              MOV B,A
10C6  2A6F10          LHLD      DSTADR          ;ANFANGSADRESSEN IN NACHSTER
10C9  09              DAD B                     ;ZEILE FÜR QUELLE UND ZIEL
10CA  226F10          SHLD      DSTADR          ;BERECHNEN UND MERKEN
10CD  EB              XCHG
10CE  2A7110          LHLD      SCRADR
10D1  09              DAD B
10D2  227110          SHLD      SCRADR
10D5  EB              XCHG
10D6  C1              POP B
10D7  41              MOV B,C                   ;BYTE-ZAHLER RESTAURIEREN
10D8  3A7410          LDA       R               ;ZEILENZAHL-1
10DB  3D              DCR A
10DC  327410          STA       R
10DF  C2B410          JNZ       LH10B4          ;NACHSTE ZEILE
10E2  C1              POP B
10E3  D1              POP D
10E4  E1              POP H
10E5  F1              POP PSW
10E6  C9              RET
10E7          ;
10E7          ;           BRINGE EIN SPRITE AUF DEN SCHIRM
10E7          ;  AUFRUF VOM BASIC: CALLM #100C,A$ (BEL. STRING-VAR)
10E7          ;  VORHER MUSS CALLM #1009 ERFOLGT SEIN FÜR RICHTUNGSFESTLEGUNG
10E7          ;
10E7  F5      SPRITE  PUSH PSW
10E8  D5              PUSH D
10E9  C5              PUSH B
10EA  5E              MOV E,M                   ;ZEIGER AUF STRING HOLEN
10EB  23              INX H
10EC  56              MOV D,M
10ED  2A6710          LHLD      LINOFF          ;ZEILEN-OFFSET (POS. OD. NEG.)
10F0  4D              MOV C,L
10F1  44              MOV B,H
10F2          ;
10F2          ; IN #10F3 MUSS VORHER DIE BREITE DES SPRITES EINGEPOKET
10F2          ; WERDEN. 1=8 PUNKTE NEBENEINANDER, 2=16 PUNKTE, USW.
```

```
10F2  210100          LXI H     1H
10F5  §=10F3 POKNEB   EQU       $-2H
10F5          ;
10F5  65              MOV H,L
10F6  E5              PUSH H                    ;ANZ BYTES, MERKEN
10F7  2600            MVI H     0H
10F9  29              DAD H                     ;*2+ZEILENOFFSET=
10FA  09              DAD B                     ;SPRUNG AUF ANF. NACHSTE ZEILE
10FB  D5              PUSH D
10FC  E5              PUSH H
10FD  CD1712          CALL      CLCADR          ;BERECHNE ANF.ADRESSE
1100  D1              POP D
1101  C1              POP B
1102  03      LH1102  INX B                     ;NACHSTES BYTE AUS STRING
1103  0A              LDAX B
1104          ;
1104          ; IN #1105 MUSS DIE ENDE-KENNUNG DES SPRITES EINGEPOKET WERDEN
1104          ; DIESES BYTE DARF AN DER JEWEILS ERSTEN POSITION JEDER
1104          ; ZEILE NATÜRLICH NICHT VORKOMMEN!
1104  FE0D            CPI       0DH
1106  §=1105 POKEND   EQU       $-1H
1106          ;
1106  CA1C11          JZ        LH111C          ;FERTIG
1109          ;
1109          ; IN #1109 KANN EIN BEFEHL EINGEPOKET WERDEN:
1109          ; ORA M (=#B6): DAS SPRITE WIRD AUF DAS ALTE BILD GEZEICHNET
1109          ; NOP   (=#00): DAS ALTE BILD WIRD ÜBERMALT
1109          ; XRA M (=#AE): FALLS DASSELBE SPRITE SCHON DORT IST, WIRD
1109          ;               ES GELÖSCHT
1109  B6              ORA M
110A          ;
110A  77              MOV M,A
110B  2B              DCX H
110C  2B              DCX H
110D  E3              XTHL
110E  2D              DCR L                     ;NOCH MEHR IN DIESER ZEILE?
110F  CA1611          JZ        LH1116          ;NEIN, NACHSTE
1112  E3              XTHL
1113  C30211          JMP       LH1102          ;WEITER IN DIESER ZEILE
1116  6C      LH1116  MOV L,H                   ;BYTE-ZAHLER RESTAURIEREN
1117  E3              XTHL
1118  19              DAD D                     ;+OFFSET=ANF. NACHSTE ZEILE
1119  C30211          JMP       LH1102          ;UND WEITER
111C          ;
111C  E1      LH111C  POP H                     ;ENDE DER ROUTINE
111D  C1              POP B
111E  D1              POP D
111F  F1              POP PSW
1120  C9              RET
1121          ;
1121          ;         VERGRÖSSERUNG EINES GRAFIK-BEREICHES
1121          ;
1121          ;  AUFRUF VON BASIC:
1121          ;    1. QUELLE ÜBERGEBEN:
1121          ;       X%=.., Y%=.., R%=(ANZAHL ZEILEN), CALLM #1009
1121          ;    2. ZIEL ÜBERGEBEN:
1121          ;       X%=.., Y%=.., R%=(ANZ.HOR.PUNKTE/8), CALLM #100F
1121          ;
1121  F5      DOUBLE  PUSH PSW
1122  E5              PUSH H
1123  D5              PUSH D
1124  C5              PUSH B
1125  CD1712          CALL      CLCADR          ;BERECHNE ZIELADRESSE
```

```
1128  226F10          SHLD      DSTADR
112B  EB              XCHG
112C  2A6D10          LHLD      PTR.R
112F  23              INX H
1130  46              MOV B,M                 ;R%=ANZ.BYTES HORIZONTAL
1131  48              MOV C,B
1132  2A7110          LHLD      SCRADR        ;QUELLENADRESSE
1135  EB              XCHG
1136  1A      LH1136  LDAX D                  ;ALTES BYTE LADEN
1137  D5              PUSH D
1138  C5              PUSH B
1139  E5              PUSH H
113A  CD4F12          CALL      DBITS         ;VERDOPPLE JEDES BIT
113D  EB              XCHG
113E  E1              POP H                   ;ZIELADRESSE
113F  E5              PUSH H
1140  7A              MOV A,D                 ;2 BYTE NEU EINSCHREIBEN
1141          ;
1141  B6              ORA M                   ;SIEHE AN ANDERER STELLE
1142  §=1141 POKE1   EQU       $-1H
1142          ;
1142  77              MOV M,A
1143  2B              DCX H
1144  2B              DCX H
1145  7B              MOV A,E
1146          ;
1146  B6              ORA M                   ;---------"------
1147  §=1146 POKE2   EQU       $-1H
1147          ;
1147  77              MOV M,A
1148  2B              DCX H
1149  2B              DCX H
114A  E3              XTHL                    ;ANF. ZIELZEILE
114B  3A6710          LDA       LINOFF
114E  4F              MOV C,A
114F  3A6810          LDA       LINOFH        ;BC = +- BYTES/LINE
1152  47              MOV B,A
1153  09              DAD B                   ;EINE ZEILE HÖHER ODER TIEFER
1154  7A              MOV A,D                 ;DIE GLEICHEN BYTES NOCHMAL
1155          ;
1155  B6              ORA M
1156  §=1155 POKE3   EQU       $-1H
1156          ;
1156  77              MOV M,A
1157  2B              DCX H
1158  2B              DCX H
1159  7B              MOV A,E
115A          ;
115A  B6              ORA M
115B  §=115A POKE4   EQU       $-1H
115B          ;
115B  77              MOV M,A
115C  E1              POP H
115D  C1              POP B
115E  D1              POP D
115F  1B              DCX D                   ;NÄCHSTES QUELLENBYTE
1160  1B              DCX D
1161  05              DCR B                   ;NOCH IN DIESER ZEILE?
1162  C23611          JNZ       LH1136        ;JA, WEITER IN DIESER
1165  41              MOV B,C                 ;RESTORE BYTE-ZÄHLER
1166  2A6710          LHLD      LINOFF
1169  EB              XCHG
116A  2A7110          LHLD      SCRADR
```

ROUTINEN DES INDATA-DEMOS

```
116D  19              DAD D                       ;NACHSTE ZEILE
116E  227110          SHLD      SCRADR            ;NEUER ANFANG QUELLE
1171  E5              PUSH H
1172  2A6F10          LHLD      DSTADR
1175  19              DAD D
1176  19              DAD D                       ;UBERNACHSTE ZEILE
1177  226F10          SHLD      DSTADR            ;NEUER ANFANG ZIEL
117A  D1              POP D
117B  3A7410          LDA       R
117E  3D              DCR A                       ;NOCH MEHR ZEILEN?
117F  327410          STA       R                 ;ZEILENZAHLER-1
1182  C23611          JNZ       LH1136            ;NACHSTE ZEILE
1185  C1              POP B
1186  D1              POP D
1187  E1              POP H
1188  F1              POP PSW
1189  C9              RET
118A            ;
118A            ;   STARTE INTERRUPT-GESTEUERTEN SOUND
118A            ;
118A            ;  AUFRUF VOM BASIC: ADR%=STARTADRESSE, CALLM #1015,ADR%
118A            ;  VORHER IN #1073 DIE ANZAHL DER KANALE (1..3) EINPOKEN!
118A            ;
118A            ;  FORMAT DER SOUND-DATEN: FREQ-LOWBYTE,FREQ-HIGHBYTE,DAUER..
118A            ;  DIE FREQUENZ-BYTES KÖNNEN MIT DER FUNKTION FREQ() ER-
118A            ;  MITTELT WERDEN. DAUER (0..#FF) ENTSPRICHT WAIT TIME (20ms)
118A            ;   BEIDE FREQ-BYTE=0: PAUSE
118A            ;   DAUER=#FF          ; DATEN-ENDE (SOUND OFF)
118A            ;  DER AUFBAU DER DATEN BEI MEHR ALS EINEM KANAL IST NICHT
118A            ;  EINFACH, DA EIN NEUER FREQ-WERT IMMER ERST GELADEN WIRD,
118A            ;  WENN DER TIMER FUR DEN ZUGEHÖRIGEN KANAL ABGELAUFEN IST!
118A            ;
118A  F5        SOUND     PUSH PSW
118B  D5                  PUSH D
118C  C5                  PUSH B
118D  23                  INX H
118E  23                  INX H
118F  56                  MOV D,M                   ;STARTADRESSE DER DATEN
1190  23                  INX H
1191  5E                  MOV E,M
1192  EB                  XCHG
1193  226210          SHLD      PTRSND            ;MERKEN
1196  3A7310          LDA       CHNCNT
1199  B7              ORA A                       ;ANZ-KANALE=0?
119A  CAB211          JZ        ESC3              ;NICHTS TUN
119D  FE04            CPI       4H                ;ANZ-KANALE>3?
119F  D2B211          JNC       ESC3              ;AUCH NICHT GUT
11A2  216410          LXI H     TIMERS            ;ALLE 3 TIMER STARTEN
11A5  3E01            MVI A     1H
11A7  77              MOV M,A
11A8  23              INX H
11A9  77              MOV M,A
11AA  23              INX H
11AB  77              MOV M,A
11AC  21B611          LXI H     NEXTN             ;TIMER-INTERRUPT UMLEITEN
11AF  227000          SHLD      I7USA
11B2  C1        ESC3  POP B                       ;UND FERTIG
11B3  D1              POP D
11B4  F1              POP PSW
11B5  C9              RET
11B6            ;
11B6            ; INTERRUPT 7 (20ms-TIMER)
11B6            ; FÜHRE NACHSTE NOTE AUS
```

```
11B6            ;
11B6  F3      NEXTN   DI                              ;(BEFEHL UBERFLUSSIG)
11B7  F5              PUSH PSW
11B8  E5              PUSH H
11B9  C5              PUSH B
11BA  216410          LXI H       TIMERS            ;HL AUF ERSTEN TIMER
11BD  3A7310          LDA         CHNCNT
11C0  4F              MOV C,A                         ;ANZAHL KANALE MERKEN
11C1  35      NXT8    DCR M                           ;TIMERWERT-1
11C2  CCD011          CZ          LAD                 ;=0: NEUE NOTE
11C5  23              INX H
11C6  0D              DCR C
11C7  C2C111          JNZ         NXT8                ;NACHSTER KANAL
11CA  C1              POP B
11CB  E1              POP H
11CC  F1              POP PSW
11CD  C3A9D9          JMP         RST7                ;NORMALER RST7 WEITER
11D0            ;
11D0            ; TIMER FUR KANAL (C) =0: NEUE NOTE UND TIMER-WERT LADEN
11D0            ;
11D0  D5      LAD     PUSH D
11D1  E5              PUSH H
11D2  2A6210          LHLD        PTRSND
11D5  5E              MOV E,M                         ;NACHSTE FREQ-BYTES
11D6  23              INX H
11D7  56              MOV D,M
11D8  23              INX H
11D9  79              MOV A,C                         ;KANAL 1..3
11DA  3D              DCR A                           ;SOUND-IC-KANAL 0..2
11DB  1F              RAR
11DC  1F              RAR
11DD  C636            ADI         36H                 ;BILDE KONTROLLWORT 8253
11DF  3206FC          STA         SNDC
11E2  7A              MOV A,D                         ;FREQ=0?
11E3  B3              ORA E
11E4  7E              MOV A,M                         ;NEUER TIMERWERT
11E5  23              INX H
11E6  226210          SHLD        PTRSND              ;NEUER SOUNDDATEN-ZEIGER
11E9  CAF611          JZ          SOFF                ;FREQ=0: BYTES NICHT IN CHIP,
11EC            ;                                     ;..KANAL IST AUS
11EC  0600            MVI B       0H
11EE  21FEFB          LXI H       SNDAD-2H            ;DATEN-ADRESSE 8253 BERECHNEN
11F1  09              DAD B
11F2  09              DAD B
11F3  73              MOV M,E                         ;DATEN IN CHIP
11F4  23              INX H
11F5  72              MOV M,D
11F6  E1      SOFF    POP H
11F7  FEFF            CPI         0FFH                ;TIMERWERT=#FF?
11F9  CA0012          JZ          SENDS               ;JA, PROGRAMM-ENDE
11FC  3C              INR A                           ;TIMERWERT+1
11FD  77              MOV M,A                         ;UND MERKEN
11FE  D1              POP D
11FF  C9              RET
1200            ;
1200            ; TIMER-WERT IST #FF, STELLE NORMALEN INTERRUPT WIEDER HER
1200            ;
1200  21A9D9  SENDS   LXI H       RST7                ;ALTEN V7
1203  227000          SHLD        17U5A
1206  2106FC          LXI H       SNDC                ;ALLE KANALE AUS
1209  3636            MVI M       36H
120B  3686            MVI M       86H
120D  36B6            MVI M       0B6H
```

```
                        ROUTINEN DES INDATA-DEMOS

120F  D1              POP D
1210  C1              POP B                     ;RETURN-ADRESSE VERNICHTEN
1211  C1              POP B
1212  E1              POP H
1213  F1              POP PSW
1214  C3A9D9          JMP       RST7            ;IM NORM. INTR WEITER
1217          ;
1217          ;
1217          ; BERECHNE AKTUELLE SCHIRMADRESSE AUS X% UND Y%
1217          ;  ADRESSE ANSCHLIESSEND IN HL
1217          ;
1217  2A6B10  CLCADR  LHLD      PTR.Y
121A  7E              MOV A,M                   ;A=Y%
121B  2A9800          LHLD      LOL             ;L=(BYTES/ZEILE) IM MODE
121E  2600            MVI H     0H
1220  CD8FDE          CALL      MULA            ;HL=Y%*(BYTES/ZEILE)
1223  EB              XCHG
1224  2A8800          LHLD      GRE             ;ENDE DER GRAFIK (UNTEN)
1227  19              DAD D
1228  EB              XCHG                      ;DE=ANFANG DER ZEILE
1229  2A6910          LHLD      PTR.X
122C  7E              MOV A,M                   ;HIGH-BYTE X% (MAX. 1)
122D  0F              RRC                       ;X%/4..
122E  23              INX H
122F  7E              MOV A,M
1230  1F              RAR
1231  1F              RAR
1232  E67E            ANI       7EH             ;X%-WERTE NUR (MOD 8) = 0
1234  2F              CMA                       ;WERT NEGIEREN
1235          ;
1235          ; IN #1236 DEN GEWÜNSCHTEN WERT EINPOKEN:
1235          ;  #03: HIGH ADDRESS BYTE DER 8 PUNKTE
1235          ;  #04: LOW   ----------"---------
1235          ;
1235  D603            SUI       3H              ;RAND-BYTES ÜBERSPRINGEN
1237  §=1236  POKHL   EQU       $-1H
1237          ;
1237  EB              XCHG
1238  4F              MOV C,A
1239  06FF            MVI B     0FFH
123B  09              DAD B                     ;HL=AKTUELLE ADRESSE
123C  C9              RET
123D  §=123C  POKRET  EQU       $-1H
123D          ;
123D          ; DIESER TEIL WIRD IM GESAMTEN DEMO NICHT BENUTZT
123D          ; FALLS DAS VORANGEHENDE RET GELÖSCHT WIRD, LIEFERT
123D          ; DIE ROUTINE IN DE DIE SCHIRMADRESSE UND IN C UND A
123D          ; DIE BIT-MASKE FÜR DAS GENAUE X% (ALSO NICHT MOD 8)
123D          ;  Z.B. SETZEN EINES PUNKTES BEI X%,Y%:
123D          ;  CALL 1217, LDAX D, ORA C, STAX D
123D          ;
123D  EB              XCHG                      ;DE=ADRESSE
123E  7E              MOV A,M                   ;LOW-BYTE X%
123F  E607            ANI       7H              ;3 BITS AUSBLENDEN
1241  3C              INR A
1242  47              MOV B,A
1243  3E80            MVI A     80H
1245  05      LH1245  DCR B                     ;1 BIT (X% MOD 8) MAL...
1246  CA4D12          JZ        LH124D          ;..RECHTSSCHIEBEN
1249  1F              RAR
124A  C34512          JMP       LH1245
124D  4F      LH124D  MOV C,A
124E  C9              RET
```

```
124F          ;
124F          ; VERDOPPLE JEDES BIT AUS A NACH HL
124F          ; BEISPIEL: A=10110001B ==> HL=1100111100000011B
124F          ;
124F  210000  DBITS   LXI H     0H
1252  0608            MVI B     8H              ;8 BITS
1254  07      LH1254  RLC                       ;NÄCHSTES BIT IN CARRY
1255  D25F12          JNC       LH125F          ;BIT=0
1258  29              DAD H                     ;BIT=1, ALSO 2 EINSEN
1259  23              INX H                     ;IN HL REINSCHIEBEN
125A  29              DAD H
125B  23              INX H
125C  C36112          JMP       LH1261
125F  29      LH125F  DAD H                     ;2 NULLEN REINSCHIEBEN
1260  29              DAD H
1261  05      LH1261  DCR B                     ;NOCH EIN BIT?
1262  C25412          JNZ       LH1254
1265  C9              RET
1266          ;
1266          ; LADE EIN UT-FILE VOM BASIC AUS
1266          ; AUFRUF VOM BASIC: NAME$="TEST", CALLM #101B,NAME$
1266          ; FUNKTIONIERT MIT ALLEN SPEICHERMEDIEN (FLOPPY,CAS,DCR)
1266          ;
1266  F5      LOADUT  PUSH PSW
1267  D5              PUSH D
1268  C5              PUSH B
1269  5E              MOV E,M
126A  23              INX H
126B  56              MOV D,M
126C  EB              XCHG                      ;HL ZEIGT AUF STRING
126D  01FF31          LXI B     31FFH           ;FILE-TYP '1', NAME BEIM
1270          ;                                 ;LADEN AUF SCHIRM. SONST
1270          ;                                 ;LXI B,3100H
1270  CDCE02          CALL      ROPEN
1273  213E01          LXI H     EBUF            ;2 BYTE ANFANGSADRESSE..
1276  114101          LXI D     LH0141          ;..EINLESEN
1279  CDD102          CALL      RBLK
127C  1100F9          LXI D     STACK           ;MAX. ENDWERT+1
127F  2A3E01          LHLD      EBUF            ;ANFANGSADRESSE
1282  CDD102          CALL      RBLK            ;LIES DATENBLOCK
1285  CDD402          CALL      RCLOSE
1288  C1              POP B
1289  D1              POP D
128A  F1              POP PSW
128B  C9              RET
128C          ;
128C          ;
128C  C9              RET                       ;(A28C) ??????
128D          ;
128D          ;
128D          ; SCHIEBT DAS PROGRAMM NACH #1000 UND INITIALISIERT HEAP ETC
128D          ; AUFRUF VOM BASIC: CALLM #A28D
128D          ; DAS PROGRAMM WIRD ZUERT NACH #A000... GELADEN UND DANN VON
128D          ; EINEM BASIC-PROGRAMM NACH #1000 GESCHOBEN. DAS GESCHIEHT
128D          ; HAUPTSÄCHLICH WEGEN DES DAI-DOS (FLOPPY), LÄUFT ABER AUCH
128D          ; MIT DCR UND CAS!
128D          ;
128D  3100F9  CALLM   LXI SP    STACK
1290  010010          LXI B     START           ;ZIELADRESSE
1293  1100A0          LXI D     LHA000          ;ANFANG BLOCK
1296  21E0A4          LXI H     LHA4E0          ;HINTER LETZTEM BYTE
1299  CD4FDE          CALL      MOVEBL          ;BLOCK VERSCHIEBEN
129C  C39F12          JMP       LH129F          ;JETZT VERSCHOBEN, ALSO JMP!!!
```

```
129F            ;
129F  217024 LH129F  LXI H     2470H          ;HEAP-ANFANG REICHLICH HINTER
12A2  229B02         SHLD      HEAP           ;PROG. (+PLATZ FÜR MUSIC-DATEN)
12A5  210000         LXI H     0H             ;TASTATURPUFFER LEEREN
12A8  22BA02         SHLD      KLIND
12AB  22BC02         SHLD      KLIND+2H
12AE  229702         SHLD      CMDTAB         ;BEFEHLS-ERWEITERUNG VERNICHTEN
12B1  CDB0DE         CALL      NEW256         ;CLEAR 256 + NEW (NUR BASIC V1.1 !!)
12B4            ; DIE NÄCHSTEN BEIDEN BEFEHLE BEI BEDARF LÖSCHEN,
12B4            ; DA SONST BREAK NICHT MöGLICH
12B4  217710         LXI H     INTR.6         ;NEUEN TASTATUR-INTR EIN-
12B7  226E00         SHLD      I6USA          ;SCHLEIFEN
12BA  21D714         LXI H     LH14D7         ;LADE BASIC-PROG "START"
12BD  CD4F13         CALL      LOADBS
12C0  C39EDF         JMP       RUNBAS         ;RUN
12C3            ;
12C3            ; MENÜ:
12C3            ; ANSPRUNG VON INTR6: TASTE GEDRÜCKT UND FLAG (#1076) GESETZT
12C3            ;
12C3  F3      MENUE  DI
12C4  3A06FD         LDA       PORO           ;BANK 0 ANWÄHLEN
12C7  E63F           ANI       3FH
12C9  3206FD         STA       PORO
12CC  324000         STA       POROM
12CF  AF             XRA A                    ;MENÜ VERBIETEN (BEI TASTE)
12D0  327610         STA       MENFLG
12D3  31FFF8         LXI SP    STACK1         ;?????? F900
12D6  210000         LXI H     0H             ;TAST-PUFFER LEEREN
12D9  22BA02         SHLD      KLIND
12DC  22BC02         SHLD      KLIND+2H
12DF  FB             EI
12E0  3EFF           MVI A     0FFH           ;FLAG FÜR BASIC-PROGS, DASS
12E2  32E502         STA       BASFLG         ;SIE DURCH MENÜ GEWÄHLT SIND
12E5  220001         SHLD      CURRNT         ;KEIN BASIC LÄUFT
12E8  3EFF           MVI A     0FFH           ;MODE 0
12EA  EF             RST 5
12EB  18             DB        18H
12EC  216B13         LXI H     MSG1           ;BEGRÜSSUNG AUF SCHIRM
12EF  CDD4DA         CALL      PRTMSG
12F2  21FA00         LXI H     0FAH           ;WAIT TIME 250
12F5  CD19E0         CALL      WTTIM1
12F8  0608           MVI B     8H             ;8 MENÜ-ZEILEN
12FA  211F14         LXI H     MSG2           ;ANFANG MENÜ-MELDUNGEN
12FD  CDD4DA         CALL      PRTMSG         ;1. ZEILE "MENU     :"
1300  CDD4DA LH1300 CALL      PRTMSG         ;MENÜ-TEXT
1303  E5             PUSH H
1304  219D14         LXI H     DOTS           ;PUNKTE HINTER TEXT
1307  CDD4DA         CALL      PRTMSG
130A  E1             POP H
130B  05             DCR B
130C  C20013         JNZ       LH1300         ;NÄCHSTE ZEILE
130F  AF             XRA A                    ;CURSOR=BLOCK
1310  327400         STA       CURTYP
1313  3D             DCR A
1314  327500         STA       CURIN
1317  214C04         LXI H     44CH           ;1100 = 22 SEC
131A  22BE01         SHLD      TIMER          ;TIMER STARTEN
131D  CDBED6 LH131D  CALL      GETC
1320  B7             ORA A
1321  C23213         JNZ       LH1332         ;TASTE GEDRÜCKT, AUSWERTEN
1324  3ABF01         LDA       TIMER+1H       ;TIMER ABGELAUFEN?
1327  FE01           CPI       1H
1329  D21D13         JNC       LH131D
```

```
132C  AF              XRA A                     ;JA, FLAG RÜCKSETZEN
132D  32E502          STA       BASFLG
1330  3E37            MVI A     37H             ;UND "START" LADEN
1332  FE31    LH1332  CPI       31H
1334  DA1D13          JC        LH131D          ;FALSCHE EINGABE (<'1')
1337  FE38            CPI       38H
1339  D21D13          JNC       LH131D          ;>'7'
133C  D631            SUI       31H             ;ZEIGER AUF GEWÄHLTEN PRO-
133E  87              ADD A                     ;GRAMMNAMEN BILDEN
133F  21AB14          LXI H     FILPTR          ;ANFANG ZEIGER-TABELLE
1342  CD30DE          CALL      ADDA            ;+2*A
1345  5E              MOV E,M
1346  23              INX H
1347  56              MOV D,M
1348  EB              XCHG                      ;HL ZEIGT AUF FILENAMEN
1349  CD4F13          CALL      LOADBS          ;LADE BASIC
134C  C39EDF          JMP       RUNBAS          ;UND STARTE ES
134F          ;
134F          ; LADE BASIC-PROGRAMM, HL ZEIGT AUF FILENAMEN
134F          ;
134F  010030  LOADBS  LXI B     3000H           ;TYP  0', NAME NICHT AUF SCHIRM
1352  CDCE02          CALL      ROPEN
1355  2AA502          LHLD      SCRBOT          ;START SCREEN = MAX ADR
1358  EB              XCHG
1359  2A9F02          LHLD      TXTBGN          ;START= ANF BASIC-PROG
135C  CDD102          CALL      RBLK            ;PROG-'TEXT' EINLESEN
135F  22A102          SHLD      STBBGN          ;ZEIGER START SYMBTAB SETZEN
1362  CDD102          CALL      RBLK            ;SYMBTAB EINLESEN
1365  22A302          SHLD      STBUSE          ;ZEIGER ENDE SYMBTAB SETZEN
1368  C3D402          JMP       RCLOSE
136B          ;
136B          ;
136B  0C0D0D  MSG1    DB        0CH,0DH,0DH
136E  4F6820          DB        'Oh  ! Someone touched me .....
138C  0D0D            DB        0DH,0DH
138E  492066          DB        'I feel so excited , .....
13A7  0D0D            DB        0DH,0DH
13A9  49226D          DB        'I"m even troubled'
13BA  206279          DB        ' by this soft touch '
13CF  0D0D            DB        0DH,0DH
13D1  492066          DB        'I feel this is going to be'
13EB  206C6F          DB        ' love at first sight'
1400  0D0D            DB        0DH,0DH
1402  492276          DB        'I"ve a lot to offer you ....'
141E  00              DB        0H
141F          ;
141F  0C0D    MSG2    DB        0CH,0DH
1421  4D454E          DB        'MENU     :'
142B  0D0D0D          DB        0DH,0DH,0DH,0DH
142F  00              DB        0H
1430          ;
1430  475241          DB        'GRAFICS'
1439  00              DB        0H
143A  31              DB        '1'
143B  0D0D            DB        0DH,0DH
143D          ;
143D  534F55          DB        'SOUND'
1446  00              DB        0H
1447  32              DB        '2'
1448  0D0D            DB        0DH,0DH
144A          ;
144A  415254          DB        'ART'
1453  00              DB        0H
```

```
1454  33              DB        '3'
1455  0D0D            DB        0DH,0DH
1457          ;
1457  454455          DB        'EDUCATION'
1460  00              DB        0H
1461  34              DB        '4'
1462  0D0D            DB        0DH,0DH
1464          ;
1464  564944          DB        'VIDEOTEX '
146D  00              DB        0H
146E  35              DB        '5'
146F  0D0D            DB        0DH,0DH
1471          ;
1471  494E54          DB        'INTERFACE'
147A  00              DB        0H
147B  36              DB        '6'
147C  0D0D            DB        0DH,0DH
147E          ;
147E  535441          DB        'START    '
1487  00              DB        0H
1488  37              DB        '7'
1489  0D0D            DB        0DH,0DH
148B  474956          DB        'GIVE YOUR CHOICE'
149B  20              DB
149C  00              DB        0H
149D          ;
149D  2E2E2E DOTS     DB        '.....'
14AA  00              DB        0H
14AB          ;
14AB  B914    FILPTR  DW        LH14B9
14AD  BE14            DW        LH14BE
14AF  C314            DW        LH14C3
14B1  C814            DW        LH14C8
14B3  CD14            DW        LH14CD
14B5  D214            DW        LH14D2
14B7  D714            DW        LH14D7
14B9          ;
14B9  04      LH14B9  DB        4H
14BA  4D454E          DB        'MEN1'
14BE  04      LH14BE  DB        4H
14BF  4D454E          DB        'MEN2'
14C3  04      LH14C3  DB        4H
14C4  4D454E          DB        'MEN3'
14C8  04      LH14C8  DB        4H
14C9  4D454E          DB        'MEN4'
14CD  04      LH14CD  DB        4H
14CE  4D454E          DB        'MEN5'
14D2  04      LH14D2  DB        4H
14D3  4D454E          DB        'MEN6'
14D7  05      LH14D7  DB        5H
14D8  535441          DB        'START'
14DD          ;
14DD          ;
14DD          ;
14DD          ;
14DD          ;
14DD          ; DIESER TEIL GEHÖRT NICHT MEHR ZU DIESEM INDATA-DEMO,
14DD          ; IST JEDOCH AUS EINEM ANDEREN VON INDATA !!!!!!!!!!
14DD          ; (VON MIR ETWAS MODIFIZIERT BP)
14DD          ; BENÖTIGT AUS DEM DEMO: SETX, SETY, PTR.X, PTR.Y, CLCADR
14DD          ;
14DD          ;                          SPRITE2
14DD          ;
```

```
14DD            ; ERMÖGLICHT DAS SCHNELLE BEWEGEN EINES GRAFIK-BEREICHES
14DD            ; VON UND ZU EINEM INTEGER-ARRAY.
14DD            ; MINDEST-PUNKTEZAHL HORIZONTAL IST 16 !
14DD            ; DER ARRAY MUSS MINDESTENS 2-DIMENSIONAL SEIN:
14DD            ; DIE LETZTE DIMENSION(+1) GIBT DIE ANZAHL DER ZEILEN,
14DD            ; DIE VORLETZTE(+1) DIE ANZAHL DER 16PUNKTE HORIZONTAL
14DD            ; BEISPIEL: DIM A%(9,1,31) ENTHÄLT 10 SPRITES A 32*32 PUNKTE
14DD            ;
14DD            ;
14DD            ; VARIABLEN:
14DD  00        XCNT    DB      0H              ;ANZ BYTES HOR.
14DE  00        YCNT    DB      0H              ;ANZ. ZEILEN (VERT.)
14DF  00        YCNT1   DB      0H              ;ZÄHLERVAR
14E0            ;
14E0            ;
14E0            ; DEFINE ÜBERGIBT DIE ARRAY-GRÖSSE AN DAS MLP
14E0            ; AUFRUF VOM BASIC:  CALLM DEFINE%,A%(0,0...) (ALLE IND. 0)
14E0            ; MUSS VOR DER BENUTZUNG VON TARRAY ODER TSCRN MINDESTENS
14E0            ; EINMAL AUFGERUFEN WERDEN!!!!!!!!!!
14E0            ;
14E0  F5        DEFINE  PUSH PSW
14E1  2B                DCX H                   ;LETZTE DIM
14E2  7E                MOV A,M
14E3  3C                INR A
14E4  32DE14            STA     YCNT            ;=ANZAHL ZEILEN
14E7  2B                DCX H                   ;VORLETZTE DIM
14E8  7E                MOV A,M
14E9  3C                INR A                   ;+1 = ANZ. INT-ZAHLEN
14EA  87                ADD A                   ;*4 ...
14EB  87                ADD A
14EC  32DD14            STA     XCNT            ;.. = ANZ.BYTES HORIZONTAL
14EF  F1                POP PSW
14F0  C9                RET
14F1            ;
14F1            ;
14F1            ; BRINGT BYTES AUS EINEM ARRAY AUF DEN SCHIRM
14F1            ; AUFRUF: X%=, Y%=, CALLM TSCRN%,A%(?..,0,0)
14F1            ; X%,Y% = LINKE, OBERE ECKE
14F1            ;
14F1  E5        TSCRN   PUSH H
14F2  211A77            LXI H   771AH           ;= LDAX D, MOV M,A
14F5  C3FC14            JMP     L2
14F8            ;
14F8            ;
14F8            ; BRINGT BYTES VOM SCHIRM IN EINEN ARRAY
14F8            ; AUFRUF: X%=, Y%=, CALLM TARRAY%,A%(?..,0,0)
14F8            ;
14F8  E5        TARRAY  PUSH H
14F9  217E12            LXI H   127EH           ;= MOV A,M, STAX D
14FC  221D15    L2      SHLD    LP2             ;HAUPTPROGRAMM ÄNDERN
14FF  E1                POP H
1500            ;
1500            ; HAUPTROUTINE FÜR BEIDE PROGRAMME, WIRD VORHER DEM
1500            ; JEWEILIGEN ZWECK ENTSPRECHEND MODIFIZIERT.
1500            ;
1500  F5                PUSH PSW
1501  C5                PUSH B
1502  D5                PUSH D
1503  E5                PUSH H
1504  CD1712            CALL    CLCADR          ;ECKE LINKS OBEN BERECHNEN
1507  D1                POP D                   ;POINTER 1.BYTE IN ARRAY
1508  E5                PUSH H                  ;SCREENADRESSE
1509  3ADE14            LDA     YCNT            ;ANZ ZEILEN IN ZÄHLER
```

ROUTINEN DES INDATA-DEMOS

```
150C  32DF14          STA       YCNT1
150F  3A9800          LDA       LOL             ;ZEILEN-OFFSET BERECHNEN (NEG)
1512  2F              CMA
1513  3C              INR A
1514  6F              MOV L,A
1515  26FF            MVI H     0FFH
1517  E3              XTHL                      ;LEN AUF STACK, ADR IN HL
1518  E5      LP1     PUSH H                    ;ANF DER ZEILE MERKEN
1519  3ADD14          LDA       XCNT
151C  4F              MOV C,A                   ;ANZ BYTES HOR. IN C
151D          ;
151D          ;   DIESE STELLE WIRD MODIFIZIERT:
151D  1A      LP2     LDAX D                    ;ODER: MOV A,M
151E  77              MOV M,A                   ;ODER: STAX D
151F          ;
151F  2B              DCX H                     ;AUF SCHIRM NACH RECHTS
1520  13              INX D                     ;NÄCHSTES BYTE IN ARRAY
1521  0D              DCR C                     ;NOCH MEHR IN DIESER ZEILE?
1522  C21D15          JNZ       LP2
1525  E1              POP H                     ;ZEILENANFANG SCHIRM
1526  C1              POP B                     ;OFFSET
1527  09              DAD B                     ;ANFANG NÄCHSTE ZEILE
1528  C5              PUSH B
1529  3ADF14          LDA       YCNT1           ;ZEILENZÄHLER-1
152C  3D              DCR A
152D  32DF14          STA       YCNT1
1530  C21815          JNZ       LP1             ;NÄCHSTE ZEILE
1533  C1              POP B                     ;STACK KORRIGIEREN
1534  D1              POP D
1535  C1              POP B
1536  F1              POP PSW
1537  C9              RET
1538                  END
```

```
A28C  :0000 ADDA  :DE30 BASFLG:02E5 CALLM :128D CHNCNT:1073 CLCADR:1217
CMDTAB:0297 COPY  :109D CURIN :0075 CURRNT:0100 CURTYP:0074 DBITS :124F
DEFINE:14E0 DIRFLG:1075 DOTS  :149D DOUBLE:1121 DSTADR:106F EBUF  :013E
ESC3  :11B2 FILPTR:14AB GETC  :D6BE GRE   :0088 HEAP  :029B I6USA :006E
I7USA :0070 INTR.6:1077 KBRFL :02C4 KLIND :02BA KLIOU :02C0 KNSCAN:02B9
L2    :14FC LAD   :11D0 LH0141:0141 LH1000:1000 LH1003:1003 LH1006:1006
LH1009:1009 LH100C:100C LH100F:100F LH1012:1012 LH1015:1015 LH101B:101B
LH1097:1097 LH1099:1099 LH10B4:10B4 LH1102:1102 LH1116:1116 LH111C:111C
LH1136:1136 LH1245:1245 LH124D:124D LH1254:1254 LH125F:125F LH1261:1261
LH129F:129F LH1300:1300 LH131D:131D LH1332:1332 LH14B9:14B9 LH14BE:14BE
LH14C3:14C3 LH14C8:14C8 LH14CD:14CD LH14D2:14D2 LH14D7:14D7 LHA000:A000
LHA4E0:A4E0 LINOFF:1067 LINOFH:1068 LOADBS:134F LOADUT:1266 LOL   :009B
LP1   :1518 LP2   :151D MENFLG:1076 MENUE :12C3 MOVEBL:DE4F MSG1  :136B
MSG2  :141F MULA  :DE8F NEW256:DEB0 NEXTN :11B6 NXTB  :11C1 POKE1 :1141
POKE2 :1146 POKE3 :1155 POKE4 :115A POKEND:1105 POKHL :1236 POKNEB:10F3
POKRET:123C PORO  :FD06 POROM :0040 PRTMSG:DAD4 PTR.R :106D PTR.X :1069
PTR.Y :106B PTRSND:1062 R     :1074 RBLK  :02D1 RCLOSE:02D4 ROPEN :02CE
RST6  :D578 RST7  :D9A9 RUNBAS:DF9E SCRADR:1071 SCRBOT:02A5 SENDS :1200
SETPAR:1034 SETPR1:1058 SETR  :102E SETX  :1021 SETY  :1027 SNDAD :FC00
SNDC  :FC06 SOFF  :11F6 SOUND :118A SPRITE:10E7 STACK :F900 STACK1:F8FF
START :1000 STBBGN:02A1 STBUSE:02A3 TARRAY:14F8 TIMER :01BE TIMERS:1064
TSCRN :14F1 TXTBGN:029F WTTIM1:E019 XCNT  :14DD YCNT  :14DE YCNT1 :14DF
```

Hex-Dump des gesamten Programmes:

```
1000 C3 21 10 C3 27 10 C3 2E 10 C3 34 10 C3 E7 10 C3  .!.. ....4.....
1010 21 11 C3 9D 10 C3 BA 11 C3 77 10 C3 66 12 C3 8C  !........w..f...
1020 A2 23 23 22 69 10 C9 23 23 23 22 6B 10 C9 23 23  .##"i..###"k..##
1030 22 6D 10 C9 F5 E5 D5 C5 CD 17 12 22 71 10 2A 6D  "m.........."q.*m
1040 10 23 7E 32 74 10 2A 98 00 26 00 3A 75 10 B7 CA  .#~2t.*..&.:u...
1050 58 10 7D 2F 6F 26 FF 23 22 67 10 C1 D1 E1 F1 C9  X.}/o&.#"g......
1060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
1070 00 00 00 00 00 00 00 E1 F5 AF 32 B9 02 F1 F5 CD  ........2.....
1080 99 10 AF 32 C4 02 3A 76 10 B7 CA 97 10 E5 2A C0  ...2..:v......*.
1090 02 7E B7 C2 C3 12 E1 F1 C9 E5 C3 78 D5 F5 E5 D5  .~.........x....
10A0 C5 CD 17 12 22 6F 10 EB 2A 6D 10 23 7E 87 4F 47  ...."o..*m.#~.OG
10B0 2A 71 10 EB 1A B6 77 2B 1B 05 C2 B4 10 C5 3A 67  *q....w+......:g
10C0 10 4F 3A 68 10 47 2A 6F 10 09 22 6F 10 EB 2A 71  .O:h.G*o.."o..*q
10D0 10 09 22 71 10 EB C1 41 3A 74 10 3D 32 74 10 C2  .."q...A:t.=2t..
10E0 B4 10 C1 D1 E1 F1 C9 F5 D5 C5 5E 23 56 2A 67 10  ..........^#V*g.
10F0 4D 44 21 01 00 65 E5 26 00 29 09 D5 E5 CD 17 12  MD!..e.&.)......
1100 D1 C1 03 0A FE 0D CA 1C 11 B6 77 2B 2B E3 2D CA  ..........w++.-.
1110 16 11 E3 C3 02 11 6C E3 19 C3 02 11 E1 C1 D1 F1  ......l.........
1120 C9 F5 E5 D5 C5 CD 17 12 22 6F 10 EB 2A 6D 10 23  ........"o..*m.#
1130 46 48 2A 71 10 EB 1A D5 C5 E5 CD 4F 12 EB E1 E5  FH*q.......O....
1140 7A B6 77 2B 2B 7B B6 77 2B 2B E3 3A 67 10 4F 3A  z.w++{.w++.:g.O:
1150 68 10 47 09 7A B6 77 2B 2B 7B B6 77 E1 C1 D1 1B  h.G.z.w++{.w....
1160 1B 05 C2 36 11 41 2A 67 10 EB 2A 71 10 19 22 71  ...6.A*g..*q.."q
1170 10 E5 2A 6F 10 19 19 22 6F 10 D1 3A 74 10 3D 32  ..*o..."o..:t.=2
1180 74 10 C2 36 11 C1 D1 E1 F1 C9 F5 D5 C5 23 23 56  t..6.........##V
1190 23 5E EB 22 62 10 3A 73 10 B7 CA B2 11 FE 04 D2  #^."b.:s........
11A0 B2 11 21 64 10 3E 01 77 23 77 23 77 21 B6 11 22  ..!d.>.w#w#w!.."
11B0 70 00 C1 D1 F1 C9 F3 F5 E5 C5 21 64 10 3A 73 10  p.........!d.:s.
11C0 4F 35 CC D0 11 23 0D C2 C1 11 C1 E1 F1 C3 A9 D9  O5...#..........
```

```
11D0 D5 E5 2A 62 10 5E 23 56 23 79 3D 1F 1F C6 36 32  ..*b.^#V#y=...62
11E0 06 FC 7A B3 7E 23 22 62 10 CA F6 11 06 00 21 FE  ..z.³~#"b......!.
11F0 FB 09 09 73 23 72 E1 FE FF CA 00 12 3C 77 D1 C9  ...s#r......<w..
1200 21 A9 D9 22 70 00 21 06 FC 36 36 36 86 36 B6 D1  !.."p.!..666.6..
1210 C1 C1 E1 F1 C3 A9 D9 2A 6B 10 7E 2A 98 00 26 00  .......*k.~*..&.
1220 CD 8F DE EB 2A 88 00 19 EB 2A 69 10 7E 0F 23 7E  ....*....*i.~.#~
1230 1F 1F E6 7E 2F D6 03 EB 4F 06 FF 09 C9 EB 7E E6  ...~/...O.....~.
1240 07 3C 47 3E 80 05 CA 4D 12 1F C3 45 12 4F C9 21  .<G>...M...E.O.!
1250 00 00 06 08 07 D2 5F 12 29 23 29 23 C3 61 12 29  ......_.)#)#.a.)
1260 29 05 C2 54 12 C9 F5 D5 C5 5E 23 56 EB 01 FF 31  )..T.....^#V...1
1270 CD CE 02 21 3E 01 11 41 01 CD D1 02 11 00 F9 2A  ...!>..A.......*
1280 3E 01 CD D1 02 CD D4 02 C1 D1 F1 C9 C9 31 00 F9  >.............1..
1290 01 00 10 11 00 A0 21 E0 A4 CD 4F DE C3 9F 12 21  ......!...O....!
12A0 70 24 22 9B 02 21 00 00 22 BA 02 22 BC 02 22 97  p$"..!.."..".."."
12B0 02 CD B0 DE 21 77 10 22 6E 00 21 D7 14 CD 4F 13  ....!w."n.!...O.
12C0 C3 9E DF F3 3A 06 FD E6 3F 32 06 FD 32 40 00 AF  ....:...?2..2@..
12D0 32 76 10 31 FF F8 21 00 00 22 BA 02 22 BC 02 FB  2v.1..!.."."...
12E0 3E FF 32 E5 02 22 00 01 3E FF EF 18 21 6B 13 CD  >.2.."..>...!k..
12F0 D4 DA 21 FA 00 CD 19 E0 06 08 21 1F 14 CD D4 DA  ..!.......!.....
1300 CD D4 DA E5 21 9D 14 CD D4 DA E1 05 C2 00 13 AF  ....!...........
1310 32 74 00 3D 32 75 00 21 4C 04 22 BE 01 CD BE D6  2t.=2u.!L."..... 
1320 B7 C2 32 13 3A BF 01 FE 01 D2 1D 13 AF 32 E5 02  ..2.:.........2..
1330 3E 37 FE 31 DA 1D 13 FE 38 D2 1D 13 D6 31 87 21  >7.1....8....1.!
1340 AB 14 CD 30 DE 5E 23 56 EB CD 4F 13 C3 9E DF 01  ...0.^#V..O.....
1350 00 30 CD CE 02 2A A5 02 EB 2A 9F 02 CD D1 02 22  .0...*...*.....".
1360 A1 02 CD D1 02 22 A3 02 C3 D4 02 0C 0D 0D 4F 68  ....."........Oh
1370 20 20 21 20 53 6F 6D 65 6F 6E 65 20 74 6F 75 63    ! Someone touc
1380 68 65 64 20 6D 65 20 2E 2E 2E 2E 2E 0D 0D 49 20  hed me .......I
1390 66 65 65 6C 20 73 6F 20 65 78 63 69 74 65 64 20  feel so excited
13A0 2C 20 2E 2E 2E 2E 2E 0D 0D 49 22 6D 20 65 76 65  , .......I"m eve
13B0 6E 20 74 72 6F 75 62 6C 65 64 20 62 79 20 74 68  n troubled by th
13C0 69 73 20 73 6F 66 74 20 74 6F 75 63 68 20 20 0D  is soft touch  .
13D0 0D 49 20 66 65 65 6C 20 74 68 69 73 20 69 73 20  .I feel this is
13E0 67 6F 69 6E 67 20 74 6F 20 62 65 20 6C 6F 76 65  going to be love
13F0 20 61 74 20 66 69 72 73 74 20 73 69 67 68 74 20   at first sight
1400 0D 0D 49 22 76 65 20 61 20 6C 6F 74 20 74 6F 20  ..I"ve a lot to
1410 6F 66 66 65 72 20 79 6F 75 20 2E 2E 2E 2E 00 0C  offer you ......
1420 0D 4D 45 4E 55 20 20 20 20 20 3A 0D 0D 0D 0D 00  .MENU     :.....
1430 47 52 41 46 49 43 53 20 20 00 31 0D 0D 53 4F 55  GRAFICS  .1..SOU
1440 4E 44 20 20 20 20 00 32 0D 0D 41 52 54 20 20 20  ND    .2..ART
1450 20 20 20 00 33 0D 0D 45 44 55 43 41 54 49 4F 4E     .3..EDUCATION
1460 00 34 0D 0D 56 49 44 45 4F 54 45 58 20 00 35 0D  .4..VIDEOTEX .5.
1470 0D 49 4E 54 45 52 46 41 43 45 00 36 0D 0D 53 54  .INTERFACE.6..ST
1480 41 52 54 20 20 20 20 00 37 0D 0D 47 49 56 45 20  ART     .7..GIVE
1490 59 4F 55 52 20 43 48 4F 49 43 45 20 00 2E 2E 2E  YOUR CHOICE ....
14A0 2E 2E 2E 2E 2E 2E 2E 2E 2E 20 00 B9 14 BE 14 C3  ......... ......
14B0 14 C8 14 CD 14 D2 14 D7 14 04 4D 45 4E 31 04 4D  ..........MEN1.M
14C0 45 4E 32 04 4D 45 4E 33 04 4D 45 4E 34 04 4D 45  EN2.MEN3.MEN4.ME
14D0 4E 35 04 4D 45 4E 36 05 53 54 41 52 54 00 00 00  N5.MEN6.START...
14E0 F5 2B 7E 3C 32 DE 14 2B 7E 3C 87 87 32 DD 14 F1  .+~<2..+~<..2...
14F0 C9 E5 21 1A 77 C3 FC 14 E5 21 7E 12 22 1D 15 E1  ..!.w....!~."...
1500 F5 C5 D5 E5 CD 17 12 D1 E5 3A DE 14 32 DF 14 3A  .........:..2..:
1510 98 00 2F 3C 6F 26 FF E3 E5 3A DD 14 4F 1A 77 2B  ../<o&...:..O.w+
1520 13 0D C2 1D 15 E1 C1 09 C5 3A DF 14 3D 32 DF 14  .........:..=2..
1530 C2 18 15 C1 D1 C1 F1 C9 00 00 00 00 00 00 00 00  ................
```

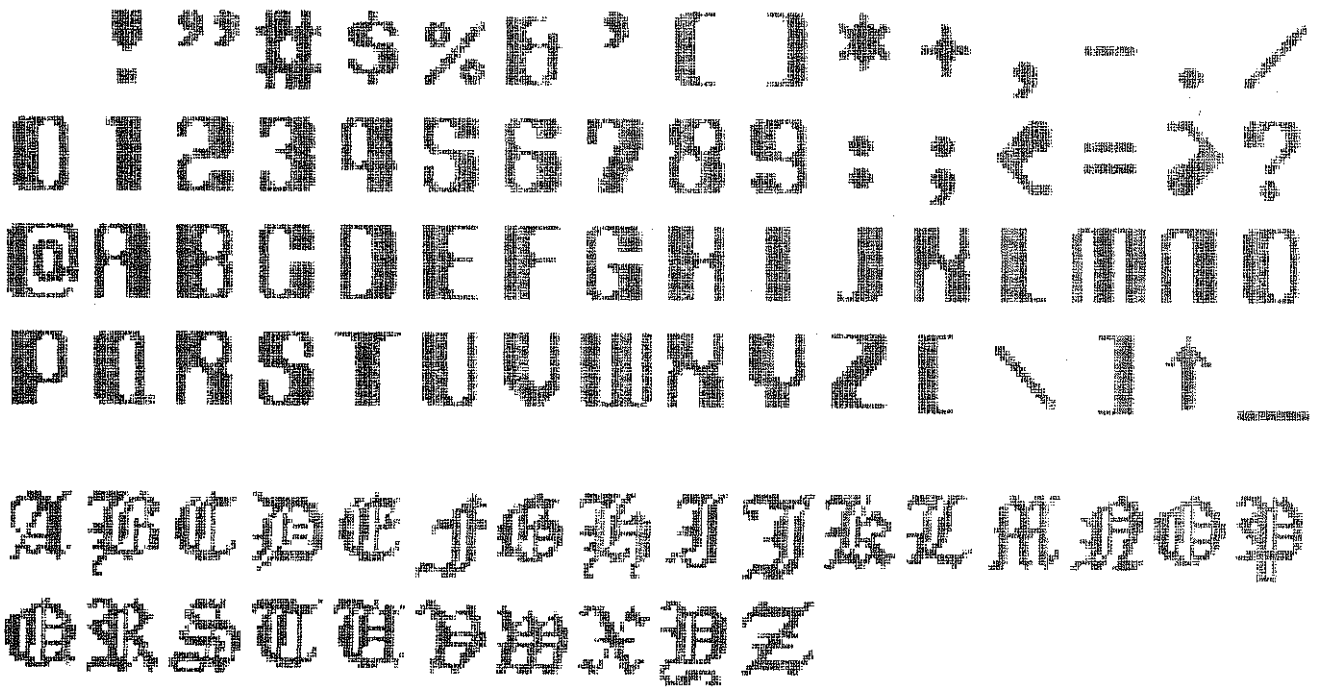
```
      Programm "START" aus INDATA-Demo

      Demonstriert die Benutzung einiger Routinen:

IMP INT

10    A$="PAINT":CALLM #101B,A$
      Lade UT-File "PAINT" (Füllen von Grafik), hier nicht benutzt!
      (Ist ein Vorläufer der von uns veröffentlichten Version)

15    COLORG 0 0 0 0:COLORT 0 0 0 0:MODE 6:MODE 0:COLORT 12 1 9 15
16    POKE #38,#C9:SOUND OFF
      Stoppe evtl. laufende Musik (intr7 gesperrt)

20    CLEAR 5000:DIM A$(64.0):LOADA A$ "EXPf"
                  lade Grafik-Schrift

21    POKE #1073,2:A$="BACH":CALLM #101B,A$:X=#14D3:CALLM #1015,X
       2-stimmig      lade+starte 2stimmige Invention von J.S.Bach

22    POKE #38,0
      öffne intr7 wieder (Musik läuft jetzt)

25    COLORG 12 3 1 4
28    POKE #1075,1:POKE #10F3,2:    POKE #1105,13
      Richtung      16 Punkt breit  Ende-Kennung=#0D

30    MODE 6:CALLM #1000,X:CALLM #1003,Y:CALLM #1009
            Variablen u. Richtungsübergabe für Sprite

50    DIM A(50),C(2),K(2):POKE #1109,0
                          =NOP (Bytes nur einschreiben)

60    FOR I=0 TO 2:READ C(I),K(I):NEXT:J=18
70    READ A:IF A<>0 THEN A(J)=A:J=J+1:GOTO 70
90    POKE #1236,4
      2. Farbsatz (LOW ADDRESS BYTE)

100   X=96:FOR I=0 TO 2:READ A:FOR Y=10 TO C(I):CALLM #100C,A$(A):NEXT:NEX
T
                                                  Sprite auf Schirm

105   WAIT TIME 20:POKE #1236,3
                   1. Farbsatz (HIGH ADDRESS BYTE)

120   FOR I=0 TO 2:Y=C(I):FOR J=0 TO K(I)
130   X=32:FOR K=J TO J+13:CALLM #100C,A$(A(K)):X=X+16:NEXT
140   NEXT:NEXT
150   POKE #BFEE-(YMAX-C(1))*PEEK(#98),#E3
160   POKE #BFEE-(YMAX-C(2))*PEEK(#98),#E1
170   DIM A$(26):LOADA A$ "OENG1":POKE #1105,2
      Alte Schrift laden            Ende-Kennung

180   FOR J=0 TO 50:A(J)=26:NEXT:FOR I=0 TO 7:READ A(I+18):NEXT
200   POKE #BFEE-(YMAX-50)*PEEK(#98),#EE
210   Y=50:FOR J=0 TO 18:X=32:FOR K=J TO J+13:CALLM #100C,A$(A(K)):X=X+16:
NEXT:NEXT
990   IF PEEK(#2E5)=0 THEN LOAD "SUNS"
      Flag, ob dies Programm durch Menü gewählt wurde

995   LOAD "MEN1"
1000  DATA 210,16,160,17,110,14,#29,#2E,#24,#21,#34,#21,0
                                 I   N   D   A   T   A
1002  DATA #24,#21,#29,15,17,4,18,4,13,19,18,0
            D   A   I   P   R   E S   E N   T   S
```

Strings aus ARRAY A$ 'EXPf' (DIM(64)) #0D=Ende-Kennung

```
0  ' ':  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0D
1  '!':  0F C0 0F C0 0F C0 0F C0 0F C0 0F C0 07 80 03 00 00 00 00 00 00 00 07 80 07 80 07 80 00 00 00 00 0D
2  '"':  38 70 7C F8 7C F8 3C 78 1C 38 38 70 70 E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0D
3  '#':  3E 60 3E 60 3E 60 FF F8 FF F8 3D E0 38 E0 38 E0 38 E0 3D E0 FF F8 FF F8 33 E0 33 E0 33 E0 00 00 0D
4  '$':  03 00 03 00 1F E0 3E 70 3C 30 3E 00 1F E0 01 F0 00 F0 30 F0 39 F0 1F E0 03 00 03 00 00 00 00 00 0D
5  '%':  00 00 00 00 38 18 7C 38 6C 70 7C E0 39 C0 03 80 07 00 0E 70 1C F8 38 D8 70 F8 60 70 00 00 00 00 0D
6  '&':  00 00 7F 80 7C 00 78 00 78 60 7C 60 7F F0 7C 60 78 60 78 60 78 60 7C E0 7F C0 00 00 00 00 0D
7  ''':  07 00 0F 80 0F 80 07 80 03 80 07 00 0E 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0D
8  '(':  1F C0 3E 00 3C 00 3C 00 3C 00 3C 00 3C 00 3C 00 3C 00 3C 00 3C 00 3C 00 3C 00 3E 00 1F C0 00 00 00 00 0D
9  ')':  0F E0 01 F0 00 F0 00 F0 00 F0 00 F0 00 F0 00 F0 00 F0 00 F0 00 F0 00 F0 00 F0 01 F0 0F E0 00 00 00 00 0D
10 '*':  03 00 03 00 37 B0 3F F0 1F E0 0F C0 1F E0 3F F0 37 B0 03 00 03 00 00 00 00 00 00 00 00 00 00 00 00 0D
11 '+':  00 00 00 00 03 00 03 00 03 00 07 80 3F F0 3F F0 07 80 03 00 03 00 03 00 00 00 00 00 00 00 00 00 0D
12 ',':  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 07 00 0F 80 0F 80 07 80 03 80 07 00 0E 00 0D
13 '-':  00 00 00 00 00 00 00 00 00 00 00 00 3F F0 3F F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0D
14 '.':  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 07 00 0F 80 0F 80 07 00 00 00 00 00 0D
15 '/':  00 00 00 00 00 18 00 38 00 70 00 E0 01 C0 03 80 07 00 0E 00 1C 00 38 00 70 00 60 00 00 00 00 00 0D
16 '0':  3F E0 7E 70 7C 30 7C 30 7C 30 7C 30 7C 30 7C 30 7C 30 7C 30 7C 30 7C 30 7E 70 3F E0 00 00 00 00 0D
17 '1':  1F C0 1F C0 07 C0 07 C0 07 C0 07 C0 07 C0 07 C0 07 C0 07 C0 07 C0 07 C0 00 00 00 00 0D
18 '2':  3F E0 73 F0 61 F0 61 F0 01 F0 01 F0 0F F0 3F E0 70 00 60 00 61 F0 61 F0 73 F0 7F F0 00 00 00 00 0D
19 '3':  7F E0 73 F0 61 F0 61 F0 01 F0 03 F0 0F E0 03 F0 01 F0 01 F0 61 F0 61 F0 73 F0 7F E0 00 00 00 00 0D
20 '4':  3F E0 77 E0 63 E0 63 E0 63 E0 63 E0 63 E0 63 E0 77 F0 7F F0 03 E0 03 E0 03 E0 03 E0 00 00 00 00 0D
21 '5':  7F F0 73 F0 61 F0 61 F0 60 00 70 00 3F E0 03 F0 01 F0 61 F0 61 F0 61 F0 73 F0 3F E0 00 00 00 00 0D
22 '6':  3F E0 73 F0 61 F0 61 E0 60 00 70 00 7F E0 73 F0 61 F0 61 F0 61 F0 61 F0 73 F0 3F E0 00 00 00 00 0D
23 '7':  7F F0 73 F0 61 F0 61 F0 61 F0 03 F0 03 E0 07 E0 07 C0 0F C0 0F 80 1F 80 1F 00 1F 00 00 00 00 00 0D
24 '8':  3F E0 7E 70 7C 30 7C 30 7C 30 3E 70 3F E0 73 E0 61 F0 61 F0 61 F0 61 F0 73 F0 3F E0 00 00 00 00 0D
25 '9':  3F E0 7E 70 7C 30 7C 30 7C 30 7C 30 7E 70 3F F0 00 70 00 30 3C 30 7C 30 7E 70 3F E0 00 00 00 00 0D
26 ':':  00 00 00 00 00 00 07 00 0F 80 0F 80 07 00 00 00 00 00 07 00 0F 80 0F 80 07 00 00 00 00 00 00 00 0D
27 ';':  00 00 00 00 00 00 07 00 0F 80 0F 80 07 00 00 00 00 00 07 00 0F 80 0F 80 07 80 03 80 07 00 0E 00 0D
28 '<':  01 E0 03 F0 07 F0 0F F0 1F F0 3F E0 7F 00 7E 00 3C 00 1C 00 0E 00 07 00 03 F0 01 F0 00 00 00 00 0D
29 '=':  00 00 00 00 00 00 00 00 3F F0 3F F0 00 00 00 00 3F F0 3F F0 00 00 00 00 00 00 00 00 00 00 00 00
30 '>':  3E 00 3F 00 03 80 01 C0 00 E0 00 F0 01 F8 03 F8 1F F0 3F E0 3F C0 3F 80 3F 00 1E 00 00 00 00 00 0D
31 '?':  3F E0 7C F0 78 70 78 70 78 70 00 E0 01 C0 03 80 07 00 06 00 00 00 06 00 0F 00 0F 00 06 00 00 00 00
32 '@':  7F F8 F8 1C F0 0C F3 CC F7 EC F6 6C F6 6C F6 6C F6 7C F7 FC F3 EC F0 00 F8 18 7F F0 00 00 00 00 0D
33 'A':  3F E0 73 F0 61 F0 61 F0 61 F0 73 F0 7F F0 73 F0 61 F0 61 F0 61 F0 61 F0 61 F0 61 F0 00 00 00 00 0D
34 'B':  7F E0 7E 70 7C 30 7C 30 7C 30 7E 70 7F E0 7E 70 7C 30 7C 30 7C 30 7C 30 7E 70 7F E0 00 00 00 00 0D
35 'C':  3F E0 73 F0 61 F0 61 F0 61 F0 60 00 60 00 61 F0 61 F0 61 F0 61 F0 61 F0 73 F0 3F E0 00 00 00 00 0D
36 'D':  7F E0 63 F0 61 F0 61 F0 61 F0 61 F0 61 F0 61 F0 61 F0 61 F0 61 F0 61 F0 63 F0 7F E0 00 00 00 00 0D
37 'E':  7F F0 7E 00 7C 00 7C 00 7C 00 7E 00 7F E0 7E 00 7C 00 7C 00 7C 00 7C 00 7E 00 7F F0 00 00 00 00 0D
38 'F':  7F F0 7E 00 7C 00 7C 00 7C 00 7E 00 7F E0 7E 00 7C 00 7C 00 7C 00 7C 00 7C 00 7C 00 00 00 00 00 0D
39 'G':  3F E0 73 F0 61 F0 61 F0 61 F0 60 00 60 00 67 F0 63 F0 61 F0 61 F0 61 F0 73 F0 3F F0 00 00 00 00 0D
40 'H':  7C 30 7C 30 7C 30 7C 30 7C 30 7E 70 7F F0 7E 70 7C 30 7C 30 7C 30 7C 30 7C 30 7C 30 00 00 00 00 0D
41 'I':  0F 80 0F 80 0F 80 0F 80 0F 80 0F 80 0F 80 0F 80 0F 80 0F 80 0F 80 0F 80 0F 80 0F 80 00 00 00 00 0D
42 'J':  03 E0 03 E0 03 E0 03 E0 03 E0 03 E0 03 E0 03 E0 03 E0 03 E0 03 E0 03 E0 27 E0 3F C0 00 00 00 00 0D
43 'K':  7C 30 7C 30 7C 30 7C 30 7C 30 7E 70 7F E0 73 E0 61 F0 61 F0 61 F0 61 F0 61 F0 61 F0 00 00 00 00 0D
44 'L':  3E 00 3E 00 3E 00 3E 00 3E 00 3E 00 3E 00 3E 00 3E 00 3E 00 3E 00 3E 00 3F 00 3F E0 00 00 00 00 0D
45 'M':  7F FC CF 3C CF 3C CF 3C CF 3C CF 3C CF 3C CF 3C CF 3C CF 3C CF 3C CF 3C CF 3C 00 00 00 00 0D
46 'N':  3F F0 73 F0 61 F0 61 F0 61 F0 61 F0 61 F0 61 F0 61 F0 61 F0 61 F0 61 F0 61 F0 00 00 00 00 0D
47 'O':  3F E0 7E 70 7C 30 7C 30 7C 30 7C 30 7C 30 7C 30 7C 30 7C 30 7C 30 7C 30 7E 70 3F E0 00 00 00 00 0D
48 'P':  7F E0 7E 70 7C 30 7C 30 7C 30 7C 30 7C 30 7C 30 7E 70 7F E0 7C 00 7C 00 7C 00 7C 00 00 00 00 00 0D
49 'Q':  3F C0 7E E0 7C 60 7C 60 7C 60 7C 60 7C 60 7C 60 7C 60 7C 60 7C 60 7C 60 7E F8 3F F8 00 00 00 00 0D
50 'R':  7F E0 7E 70 7C 30 7C 30 7C 30 7E 70 7F E0 73 E0 61 F0 61 F0 61 F0 61 F0 61 F0 61 F0 00 00 00 00 0D
51 'S':  3F E0 7E 70 7C 30 7C 30 7C 30 7E 00 3F E0 03 F0 01 F0 61 F0 61 F0 61 F0 73 F0 3F E0 00 00 00 00 0D
52 'T':  FF F8 1F C0 0F 80 0F 80 0F 80 0F 80 0F 80 0F 80 0F 80 0F 80 0F 80 0F 80 0F 80 0F 80 00 00 00 00 0D
53 'U':  7C 30 7C 30 7C 30 7C 30 7C 30 7C 30 7C 30 7C 30 7C 30 7C 30 7C 30 7C 30 7E 70 3F E0 00 00 00 00 0D
54 'V':  7C 30 7C 30 7C 30 7C 30 7C 30 7C 30 7C 30 7C 30 7C 30 7E 70 3F F0 1F E0 0F C0 07 80 00 00 00 00 0D
55 'W':  F3 CC F3 CC F3 CC F3 CC F3 CC F3 CC F3 CC F3 CC F3 CC F3 CC F3 CC F3 CC 7F F8 00 00 00 00 0D
56 'X':  7C 30 7C 30 7C 30 7C 30 7C 30 3E 70 3F E0 73 E0 61 F0 61 F0 61 F0 61 F0 61 F0 00 00 00 00 0D
57 'Y':  7C 30 7C 30 7C 30 7C 30 7C 30 7C 30 7C 30 7C 30 7E 70 3F E0 0F 80 0F 80 0F 80 0F 80 00 00 00 00 0D
58 'Z':  7F F0 03 F0 01 F0 01 F0 03 F0 07 E0 0F C0 1F 80 3F 00 7E 00 7C 00 7C 00 7E 00 7F F0 00 00 00 00 0D
59 '[':  3F 80 3C 00 3C 00 3C 00 3C 00 3C 00 3C 00 3C 00 3C 00 3C 00 3C 00 3C 00 3C 00 3C 00 3F 80 00 00 0D
60 '\':  00 00 00 00 60 00 70 00 38 00 1C 00 0E 00 07 00 03 80 01 C0 00 E0 00 70 00 38 00 1B 00 00 00 00 0D
61 ']':  07 F0 00 F0 00 F0 00 F0 00 F0 00 F0 00 F0 00 F0 00 F0 00 F0 00 F0 00 F0 00 F0 00 F0 07 F0 00 00 0D
62 '^':  02 00 02 00 07 00 0F 80 1F C0 37 60 07 00 07 00 07 00 07 00 07 00 07 00 07 00 07 00 00 00 00 00 0D
63 '_':  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF FC FF FC 0D
```

Strings aus ARRAY A$ "OENG1 (DIM(26)) #02=Ende-Kennung

```
0  'A':  0E 18 3F B0 63 E0 43 E0 46 E0 44 E0 4C E0 28 E0 1F E0 10 E0 3B EB 2E F0 60 20 00 00 00 00 00 00 02
1  'B':  7F 30 FE F8 8D 9C 19 9C 1B 18 1B 50 7B 78 F8 4C 1B 4C 3A 4C 7C 5C 1F E0 31 80 60 00 40 00 60 00 02
2  'C':  00 80 0D F8 1B F0 36 80 36 80 76 80 76 80 76 80 76 80 76 80 34 B8 3C 98 1F F0 07 C0 00 00 00 00 00 02
3  'D':  C0 00 7F E0 3F F8 0D 38 1A 1C 33 1C 32 EC 32 0C F2 EC 33 1C 34 0C 6F F8 DF F0 30 00 40 00 00 00 02
4  'E':  00 80 0D F8 1B F0 36 90 36 A0 76 B8 76 F0 76 80 76 80 34 88 3C 98 1F F0 07 C0 00 00 00 00 00 00 02
5  'F':  00 10 00 38 00 7C 01 F0 01 B0 0F BC 1F BC 11 B0 01 B0 01 B0 01 B0 01 B0 01 B0 73 60 FF C0 87 00 02
6  'G':  01 80 07 F0 1F E0 3C 40 6C C0 6D F0 EF F8 ED 38 ED F8 69 18 79 30 FD E0 0F 80 00 00 00 00 00 00 02
7  'H':  7F 60 FE C0 8D 80 19 90 1B 38 1B FC 78 1C FB FC 1B 1C 3A 1C 7C 5C 1F DC 33 98 60 30 40 20 60 30 02
8  'I':  0F D8 1F B0 11 60 03 60 03 60 03 60 0F 60 1F 60 03 60 03 60 0E 60 38 60 6C E0 07 80 00 00 00 00 02
9  'J':  3E 1C FF F8 87 C8 E0 90 61 B0 03 B0 03 B0 03 B0 0F B0 03 B0 01 B0 01 B0 03 60 3E E0 7F C0 CF 00 02
10 'K':  7E 80 FE 80 8D 00 1D 60 1A F0 1B B8 7A 18 FA 70 1B C0 1A E0 34 70 3E FC 7F B8 C3 10 00 00 00 00 02
11 'L':  7E F0 FD C0 8D 80 19 80 1B 80 1B 80 1B 80 FB 00 3B 00 36 00 3C 18 7F F0 E3 E0 00 40 00 00 00 00 02
12 'M':  0C 6C 1E F8 37 B0 33 30 33 30 33 30 77 F0 3F B0 33 30 33 30 33 34 33 3C 73 B8 E1 10 00 00 00 00 02
13 'N':  00 60 01 F8 07 38 0D 1C 3D FC 1D 1C 00 FC 0D 1B 3D 38 1D 30 0D 30 1A 60 7E 7C CC F8 00 00 00 00 02
14 'O':  01 80 07 E0 1D 70 7D 38 ED 18 ED FC ED 1C ED 1C ED F8 69 18 79 30 3D E0 0F 80 00 00 00 00 00 00 02
15 'P':  79 40 DD E0 0F 70 1D 38 7D 18 ED F8 7D 18 3D F8 1D 18 0D 18 3F F0 FF E0 0D 00 0D 00 0D 00 0F 00 02
16 'Q':  01 80 07 E0 1D 70 7D 38 ED 18 ED FC ED 1C ED 1C ED F8 69 18 79 38 3D FC 0F 86 00 00 00 00 00 00 02
17 'R':  78 40 DD E0 0F 70 1D 38 7D 30 ED 60 7D C0 3D E0 1D 70 0D 30 0D 38 7F 5C FF 9C 83 06 00 00 00 00 02
18 'S':  07 A0 0F E0 18 40 1C 40 0F F0 17 F8 30 9C 3F CC 1F EC 01 6C 7D 48 FF F8 C3 F0 7E 60 00 00 00 00 02
19 'T':  7F 84 FF F8 9B F0 36 40 36 40 76 40 76 40 76 40 76 40 34 48 3C 58 1F F0 07 C0 00 00 00 00 00 02
20 'U':  7C 80 FF 84 9B 78 36 30 36 30 77 F0 76 30 77 F0 76 30 34 30 3C 34 1F F8 07 86 00 00 00 00 00 00 02
21 'V':  30 00 78 30 DC F0 1F 38 0C 18 0E 18 0F FC 0E 1C 0F FC 0C 38 0C 70 1C 60 1F C0 11 80 00 00 00 00 02
22 'W':  00 00 40 00 E2 10 67 38 7B D8 73 9C 3F FC 31 8C FF FC 73 9C 63 18 63 18 FF F8 CE 70 00 00 00 00 02
23 'X':  00 00 78 20 DC 70 0C 7C 0E 98 06 80 07 00 1F E0 03 80 05 80 05 C0 68 D8 F8 F0 30 60 00 00 00 00 02
24 'Y':  70 00 D9 E0 1F F0 1A 38 1A 38 F8 F8 3A 38 1B F8 1A 38 1A 38 1A 70 7F C0 C0 00 87 E0 0F 30 7C 70 02
25 'Z':  18 0C 3F F8 7F F0 C0 60 80 E0 01 C0 1F E0 07 00 3F C0 1C 04 18 0C 3F F8 7F F0 C0 60 00 00 00 00 02
26 ' ':  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 02
```



NP70,21

SOUND-Daten für ein einstimmiges Musikstück
(im INDATA-Demo von #14D3 bis #16FF)

```
EE 13 1C → 5E 16 1C → C2 10 1C → EE 0E 0E
2F 0B 0E   4D 0D 1C   EE 0E 0E   2F 0B 0E
4D 0D 1C   C2 10 1C   EE 0E 1C   EE 13 1C
5E 16 1C   42 00 1C   EE 13 1C   5E 16 1C
C2 10 1C   EE 0E 0E   2F 0B 0E   4D 0D 1C
EE 0E 0E   2F 0B 0E   4D 0D 1C   C2 10 1C
EE 0E 1C   EE 13 1C   5E 16 1C   42 00 1C
5E 16 1C   EE 13 1C   9A 1A 1C   5E 16 0E
EE 13 0E   DC 1D 1C   EE 13 0E   C1 11 0E
5E 16 1C   2F 0B 1C   4D 0D 1C   EE 0E 0E
C2 10 0E   5E 16 1C   42 00 1C   5E 16 1C
EE 13 1C   9A 1A 1C   5E 16 0E   EE 13 0E
1C 19 1C   C2 10 0E   EE 0E 0E   CF 12 1C
67 09 1C   2F 0B 1C   8E 0C 0E   17 0E 0E
CF 12 1C   42 00 1C   CF 12 1C   C2 10 1C
CF 12 1C   C2 10 0E   EE 0E 0E   8E 0C 0E
C2 10 0E   CF 12 1C   17 0E 0E   8E 0C 0E
2F 0B 0E   67 09 0E   8E 0A 0E   2F 0B 0E
8E 0C 0E   8E 0A 0E   2F 0B 0E   17 0E 0E
8E 0C 1C   CF 12 1C   C2 10 1C   CF 12 1C
C2 10 0E   EE 0E 0E   8E 0C 0E   C2 10 0E
EE 0E 1C   4D 0D 1C   EE 0E 1C   4D 0D 0E
2F 0B 0E   F7 09 0E   4D 0D 0E   EE 0E 1C
2F 0B 0E   F7 09 0E   E0 08 0E   77 07 0E
61 08 0E   E0 08 0E   F7 09 0E   61 08 0E
E0 08 0E   2F 0B 0E   F7 09 1C   E0 08 0E
D9 0B 0E   2F 0B 1C   E0 08 1C   4D 0D 1C
E0 08 1C   4D 0D 1C   2F 0B 0E   EE 0E 0E
4D 0D 1C   2F 0B 1C   4D 0D 1C   2F 0B 0E
EE 0E 0E   4D 0D 1C   EE 0E 0E   EE 0E 0E
C1 11 1C   C2 10 1C   EE 0E 0E   EE 0E 0E
C1 11 1C   C2 10 0E   4D 0D 0E   EE 0E 1C
C1 11 1C   EE 0E 1C   2F 0B 1C   8E 0C 0E
4D 0D 0E   EE 0E 0E   C2 10 0E   EE 0E 1C
4D 0D 1C   2F 0B 1C   F7 09 0E   61 08 0E
2F 0B 1C   F7 09 1C   2F 0B 1C   8E 0C 0E
4D 0D 0E   EE 0E 0E   C2 10 0E   EE 0E 1C
4D 0D 1C   2F 0B 1C   F7 09 0E   61 08 0E
2F 0B 1C   F7 09 1C   2F 0B 1C   42 00 1C
42 00 1C   F7 09 0E   61 08 0E   2F 0B 1C
F7 09 1C   2F 0B 1C   EE 13 1C   5E 16 1C
C2 10 1C   EE 0E 0E   2F 0B 0E   4D 0D 1C
EE 0E 0E   2F 0B 0E   4D 0D 1C   C2 10 1C
EE 0E 1C   EE 13 1C   5E 16 1C   EE 13 1C
5E 16 1C   C2 10 1C   EE 0E 0E   2F 0B 0E
4D 0D 1C   C2 10 1C   8E 0C 1C   EE 0E 1C
C2 10 70 → FF FF FF
```

Das folgende Programm ist als Erweiterung für DAInatext gedacht.
Es gliedert sich in drei Teile:
- Definieren von Grafiksymbolen
- Benutzen des 4 Byte langen Tastatureingabebuffers
- Laden von Editorinhalten von BASIC- oder Assembler-source-Programmen

Als erstes beschreibe ich die Assemblerprogramme die obige Operationen ermölichen. Dann folgen die Änderungen im Basic-Teil von DAInatext und am Ende finden sich noch einige Tips.

1 Assemblerprogramme

1.1 Grafikzeichen

```
0005      **0358            ORG   :358  hier befindet sich die Druckerausgabe
0006 0358 F1                POP   PSW
0007 0359 EF03              RSTD  5,3
0008 035B FE1C              CPI   28    Code für Grafikzeichen
0009 035D CAEC0C            JZ    POINTER

  die Routine 'PRINT' kann man durch JMP :DD94 (RS-232) ersetzen.
0010 0360 F5       PRINT    PUSH  PSW  Ausgabe über DCE-Bus
0011 0361 3A02FE            LDA   :FE02
0012 0364 A7                ANA   A
0013 0365 F26103            JP    PRINT+1
0014 0368 F1                POP   PSW
0015 0369 3200FE            STA   :FE00
0016 036C C9                RET

0018 036D                   *
0019 036D **0CEC            NPOS  :CEC
0020 0CEC          *
0021 0CEC          * Einsprung von der Print-Routine
0022 0CEC          *
0023 0CEC E5       POINTER  PUSH  H           Einsprung nach CHR$(#1C)
0024 0CED 21F50C            LXI   H,ZEICHNE  naechstes Zeichen in spezielle
0025 0CF0 22DE02            SHLD  :2DE         Routine. Dieses definiert dann
0026 0CF3 E1                POP   H            das auzugebende Grafikzeichen.
0027 0CF4 C9                RET
0028 0CF5          *
0029 0CF5 E5       ZEICHNE  PUSH  H         wird nur nach CHR$(#1C) angesprungen
0030 0CF6 213003            LXI   H,:330
0031 0CF9 22DE02            SHLD  :2DE     alter Einsprung fuer Print
0032 0CFC E1                POP   H
0033 0CFD D641              SUI   :41      (reg A):=Zeichen-#41
0034 0CFF F8                RM             wenn Zeichen < 'A'
0035 0D00 FE00     ARR.LEN  CPI   00       hier wird die Anzahl der Punkte
                                             des Grafikzeichens hingepoked
0036 0D02 F0                RP             wenn Zeichen <= GRMAX dann Ausgabe
0037 0D03 C5                PUSH  B
0038 0D04 D5                PUSH  D
0039 0D05 E5                PUSH  H
0040 0D06 2A730D            LHLD  ARR.POS  VARPTR(PUNKT$(0))
0041 0D09 110200            LXI   D,2
0042 0D0C          *
0043 0D0C A7       COUNT    ANA   A        hochzaehlen bis zum gesuchten Array-
0044 0D0D CA150D            JZ    EX.COUNT   element
0045 0D10 3D                DCR   A
0046 0D11 19                DAD   D
0047 0D12 C30C0D            JMP   COUNT
0048 0D15          *
```

```
0049 0D15 5E      EX.COUNT  MOV   E,M      HL zeigt auf den Pointer zum String
0050 0D16 23                INX   H
0051 0D17 56                MOV   D,M
0052 0D18 EB                XCHG           HL zeigt auf Laengenbyte des Strings
0053 0D19          *
0054 0D19 7E                MOV   A,M      (reg A):=Stringlaenge
```

Der Sinn der Konvertierunsroutine ist die Umwandlung der hexadezimal codierten Stringlänge in eine BCD-codierte Zahl, die dann an den Drucker übergeben wird.

```
0055 0D1A          *
0056 0D1A          *          Konvertierung von Hex in BCD
0057 0D1A          *    Bsp.: aus (reg A):=#21 wird (reg A):=33 dezimal
0058 0D1A          *
0059 0D1A 47                MOV   B,A
0060 0D1B 0E00              MVI   C,0
0061 0D1D E6F0              ANI   :F0
0062 0D1F CA2F0D  LOOP      JZ    EXIT
0063 0D22 F5                PUSH  PSW
0064 0D23 AF                XRA   A
0065 0D24 79                MOV   A,C
0066 0D25 C606              ADI   6
0067 0D27 27                DAA
0068 0D28 4F                MOV   C,A
0069 0D29 F1                POP   PSW
0070 0D2A D610              SUI   :10
0071 0D2C C31F0D            JMP   LOOP
0072 0D2F AF      EXIT      XRA   A
0073 0D30 78                MOV   A,B
0074 0D31 27                DAA
0075 0D32 81                ADD   C
0076 0D33 27                DAA
0077 0D34 4F                MOV   C,A
0078 0D35          *
0079 0D35          *    Ende Konvertierung
0080 0D35          *
```

Die Initialisierung für den Drucker bezieht sich auf den ITOH 8510
Das Format ist hier wie folgt:

Bsp.: S0012 die nächsten 12 Bytes sind Grafikzeichen

Ich hoffe, daß mit dieser Information das Umschreiben der Routine auf einen anderen Druckertyp nicht schwerfällt.

```
0081 0D35 3E1B              MVI   A,27     Grafikinitialisierung mit
0082 0D37 CD6003            CALL  PRINT
0083 0D3A 3E53              MVI   A,"S"
0084 0D3C CD6003            CALL  PRINT
0085 0D3F 3E30              MVI   A,"0"
0086 0D41 CD6003            CALL  PRINT    die ersten beiden Ziffern seien immer
0087 0D44 CD6003            CALL  PRINT    Null, da ich davon ausgehe, daß ein
                                           Grafikzeichen maximal 99 Bytes hat.
0088 0D47 79                MOV   A,C
0089 0D48 E6F0              ANI   :F0
0090 0D4A 0F                RRC
0091 0D4B 0F                RRC
0092 0D4C 0F                RRC
0093 0D4D 0F                RRC
0094 0D4E C630              ADI   :30
0095 0D50 CD6003            CALL  PRINT    Ausgabe Ziffer 3
0096 0D53 79                MOV   A,C
0097 0D54 E60F              ANI   :0F
0098 0D56 C630              ADI   :30
0099 0D58 CD6003            CALL  PRINT    Ausgabe Ziffer 4
```

```
0100 0D5B          *
0101 0D5B 46               MOV   B,M      (reg B):=Stringlaenge
0102 0D5C 04               INR   B
0103 0D5D 23               INX   H
0104 0D5E 78               MOV   A,B
0105 0D5F 05       DOTS    DCR   B        alle Zeichen des Strings auf Drucker
0106 0D60 CA6B0D           JZ    EX.DOTS
0107 0D63 7E               MOV   A,M
0108 0D64 CD6003           CALL  PRINT
0109 0D67 23               INX   H
0110 0D68 C35F0D           JMP   DOTS     Rücksprung zum Schleifenanfang
0111 0D6B E1       EX.DOTS POP   H
0112 0D6C D1               POP   D
0113 0D6D C1               POP   B
0114 0D6E C9               RET
0115 0D6F          *
  Routine um ARR.POS zu belegen. Es zeigt hinterher auf FELD(0,0)
0116 0D6F 22730D   INIT    SHLD  ARR.POS  nach CALLM INIT wird ARR.POS belegt
0117 0D72 C9               RET
0118 0D73          *
0119 0D73 **0002   ARR.POS RES   2
0120 0D75          *
```

1.2 Tastaturbuffer

```
0131 0D75 **0AC9   CHIN    EQU   :AC9
0132 0D75 **E4BC   SOUND   EQU   :E4BC
0133 0D75 **D6BE   GETKEY  EQU   :D6BE
0134 0D75          *
0135 0D75 **0001   KEY     RES   1
0136 0D76          *
   Die Routine wird von Basic aus aufgerufen und liest Zeichen aus dem Tastatur-
   buffer. Sie kehrt zu Basic zurück, wenn irgendwelche Sonderfunktionen aufge-
   rufen werden.
0137 0D76 C5               PUSH  B
0138 0D77 23               INX   H
0139 0D78 23               INX   H
0140 0D79 23               INX   H
0141 0D7A 46               MOV   B,M
0142 0D7B C5               PUSH  B
0143 0D7C CDBED6 NEXTCHAR  CALL  GETKEY
0144 0D7F DA7C0D           JC    NEXTCHAR   ignoriere Break
0145 0D82 CA7C0D           JZ    NEXTCHAR
0146 0D85 FE07             CPI   7
0147 0D87 DACB0D           JC    EX.KEY     Sonderfunktion
0148 0D8A CDD10D           CALL  CHINENTRY  Zeichenausgabe an Textbuffer
   Es folgt die compilierte Form der Basic-Zeilen, die für die Anzeige der
   Cursorpostion und die Farbgebung des Cursors verantwortlich sind.
0149 0D8D 3E20   ENTRY2060 MVI   A,32       Zeile 2960
0150 0D8F 327DB6           STA   :B67D
0151 0D92 C1               POP   B
0152 0D93 C5               PUSH  B
0153 0D94 3AAB00           LDA   :AB
0154 0D97 B8               CMP   B
0155 0D98 DAA90D           JC    TEST2
0156 0D9B 210F0E           LXI   H,COL1
0157 0D9E EF06             RSTD  5,6
```

```
0158 0DA0 01D80D            LXI   B,SOUND1
0159 0DA3 CDBCE4            CALL  SOUND
0160 0DA6 C37C0D            JMP   NEXTCHAR
0161 0DA9 C605      TEST2   ADI   5
0162 0DAB B8                CMP   B
0163 0DAC DABD0D            JC    NOTEST
0164 0DAF 21130E            LXI   H,COL2
0165 0DB2 EF06              RSTD  5,6
0166 0DB4 01F30D            LXI   B,SOUND2
0167 0DB7 CDBCE4            CALL  SOUND
0168 0DBA C37C0D            JMP   NEXTCHAR
0169 0DBD                   *
0170 0DBD 21170E    NOTEST  LXI   H,COLNORM
0171 0DC0 EF06              RSTD  5,6
0172 0DC2 010E0E            LXI   B,SOUNDOFF
0173 0DC5 CDBCE4            CALL  SOUND
0174 0DC8 C37C0D            JMP   NEXTCHAR
0175 0DCB           *
0176 0DCB C1        EX.KEY  POP   B
0177 0DCC C1                POP   B
0178 0DCD 32750D            STA   KEY
0179 0DD0 C9                RET
0180 0DD1           *
0181 0DD1 F5      CHINENTRY PUSH  PSW
0182 0DD2 E5                PUSH  H
0183 0DD3 D5                PUSH  D
0184 0DD4 C5                PUSH  B
0185 0DD5 C3C90A            JMP   CHIN   Zeichenausgaberoutine von DAInatext
0186 0DD8           *
   Verschiedene Kontrollsequenzen zur Sound- und Farbensteuerung
0187 0DD8 14000000 SOUND1   DT    1400000001140000000014000000F14000000
0188 0DEB 00200910          DT    002009100AC80000
0189 0DF3           *
0190 0DF3 14000000 SOUND2   DT    1400000001140000000114000000914000000
0191 0E06 01200910          DT    0120091009DC0000
0192 0E0E           *
0193 0E0E FF        SOUNDOFF  DT     FF
0194 0E0F           *
0195 0E0F 08000300 COL1     DT    08000300
0196 0E13           *
0197 0E13 08000A00 COL2     DT    08000A00
0198 0E17           *
0199 0E17 08000C00 COLNORM  DT    08000C00
```

1.3 Einlesen von Editorinhalten

```
0203 0E1B C5        READ    PUSH  B
0204 0E1C 210000            LXI   H,0
0205 0E1F 0631              MVI   B,:31  Filekennung '1'
0206 0E21 0E01              MVI   C,1
0207 0E23 CDCE02            CALL  :2CE   ROPEN,PRINT NAME
0208 0E26 2AA200            LHLD  :A2    Start des Bereiches in den einzu-
                                           leasen ist
0209 0E29 E5                PUSH  H
0210 0E2A CDD102            CALL  :2D1
0211 0E2D E1                POP   H      ignoriere die ersten Bytes
0212 0E2E DCD102            CC    :2D1
0213 0E31 3600              MVI   M,0
0214 0E33 23                INX   H
0215 0E34 22A400            SHLD  :A4
```

```
0216 0E37 CDD402          CALL   :2D4
0217 0E3A D2A8D2          JNC    :D2A8
0218 0E3D C1              POP    B
0219 0E3E C9              RET
```

2 Änderungen im DAInatext Basic-Teil

2.0 Menueänderung für alle neuen Optionen

```
1000      REM ****************** Menue *********************
1010      COLORT 6 0 0 0:POKE #75,0
1020      SOUND OFF :POKE #131,1:PRINT CHR$(12)
1030      PRINT SPC(20);"Bitte waehlen Sie !!":PRINT
1040      PRINT SPC(15);"Weiter im Text . . . .   < W >"
1050      PRINT SPC(15);"Kille Text . . . . . .   < K >"
1060      PRINT SPC(15);"Drucke Text  . . . . .   < D >"
1070      PRINT SPC(15);"Speichere Text ab  . .   < S >"
1080      PRINT SPC(15);"Lade Text bei Cursor .   < L >"
1085      PRINT SPC(15);"Lade Editor-File . . .   < H >"
1090      PRINT SPC(15);"Zeige Standard-Formeln   < F >"
1095      PRINT SPC(15);"Zeige Standard-Grafik    < G >"
1100      PRINT SPC(15);"TAB-Positionen aendern   < T >"
1110      PRINT SPC(15);"Ersetze Wort . . . . .   < E >"
1120      PRINT SPC(15);"Zeilenbegrenzung . . .   < Z >"
1130      PRINT SPC(15);"Randausgleich  . . . .   < R >"
1140      PRINT SPC(15);"Blanks loeschen  . . .   < B >"
1150      PRINT SPC(15);"Verschiebe Block . . .   < V >"
1160      PRINT SPC(15);"Steuerung MDCR . . . .   < C >"
1170      PRINT SPC(15);"Programmende . . . . .   < Q >"
1200      FREI%=BUFEN%-(PEEK(#A4)+256*PEEK(#A5)):BEL%=(PEEK(#A4)+256*PEEK(#A5))
C           -BUFBEG%-1
1210      CURSOR #F,4:PRINT "frei:";FREI%;
1220      PRINT SPC(15-LEN(STR$(BEL%)));"belegt:";BEL%
1230      CALLM ZZAEHLER%:CURSOR 25,3:PRINT "Zeilen:";PEEK(#4F4)+256*PEEK(#4F5);
1240      IF PEEK(#4F3)<>13 THEN CURSOR 13,1:PRINT "Letzte Zeile endet nicht
C           mit ";CHR$(#8D);" !!!"
1250      IF FREI%=0.0 THEN FOR I%=1 TO 10:GOSUB 2350:WAIT TIME 5:NEXT
1500      INCH%=GETC:IF INCH%<ASC("B") THEN 1500
1510      ON INCH%-ASC("B")+1 GOTO 4000,9500,6000,4500,5500,8500,8700,1500
C           ,1500,9000,6500,1500,1500,8900,1500,1500,3500,7000,5000,1500
C           ,8000,2000,1500,1500,3000
```

2.2 Aufruf der Tastaturabfrageroutine

```
2010      COLORT 8 0 12 0:SOUND OFF:POKE #75,#FF:POKE #2C3,#FF
2020      CALLM EDI%
2030      CALLM #D76,ZLAENGE%:INCH%=PEEK(#D75)
```

2.1 Änderung um die definierten Grafiksymbole auflisten zu können

```
8500      REM ******* Zeige Standard-Grafiksymbole ********
8510      PRINT CHR$(12):PRINT :PRINT "      Standard-Grafiksymbole :":PRINT
8520      FOR I%=0 TO GRMAX%-1
8530      PRINT "CHR$(#1C) + ";CHR$(I%+ASC("A"));" = ":PRINT TEXT$(I%):
C           GOSUB 5540:NEXT:GOTO 1000
```

2.2 Hier werden Editorinhalte eingelesen

```
8700      REM ************ Lade Edit-File ***************
8710      CALLM #E1B:GOTO 1000
```

```
8900      REM ************ ENDE ***********************
8910      COLORT 8 0 8 0:PRINT CHR$(12)
8920      POKE #74,#FF:POKE #75,#5F
8930      END
```

2.1.1 Definition der Grafiksymbole

Die Definiton geschieht hier für einen ITOH 8510. Beim Epson-Drucker bewirken diese Grafikbytes den Ausdruck anderer Zeichen, da die Grafik anders definiert ist. Beim ITOH wird ein Punkt unten durch #80 und ein Punkt oben durch #01 dargestellt.

```
11200      STMAX%=15:REM 0 BIS 25 (=A-Z)
  ARR% ist die Anzahl der zur Zeit definierten Grafiksymbole
  CALLM #D6F initialisiert das ml-Programm mit der Startadresse des Arrays PUNKT$
  #D01 beinhaltet die Anzahl der Grafiksymbole. Fehlerabfrage im Maschinenprogramm
11210      ARR%=4:DIM PUNKT$(ARR%),TEXT$(ARR%):GRMAX%=ARR%+1:CALLM #D6F,PUNKT$(0):
C           POKE #D01,GRMAX%
11211      RESTORE:FOR J%=0 TO ARR%
11212      READ C%:IF C%=(-1) THEN 11214
11213      PUNKT$(J%)=PUNKT$(J%)+CHR$(C%):GOTO 11212
11214      NEXT
  Die Grafiksymbole weren definiert imdem man sie byteweise in DATA-Zeilen
  schreibt und als letztes Zeichen eine '-1' schreibt. In TEXT$ kommt dann
  noch eine Beschreibung des Zeichens.
11215      DATA #62,#95,#95,#93,#8B,#F7,#F3,#F7,#8B,#93,#95,#95,#62,-1
11217      TEXT$(0.0)="Telefonzeichen"
11220      DATA 0,0,#7F,#41,#41,#41,0,0,-1
11221      TEXT$(1.0)="eckige Klammer nach rechts"
11225      DATA 0,#6,#2,#8F,#FB,#9C,#FC,#8,-1
11226      TEXT$(2.0)="Kamel"
11230      DATA 0,#7C,#7C,#7C,#7C,#7C,#42,#81,#81,#81,#C1,#D5,#55,#35,#15,#D,
C            5,5,5,5,5,3,0,0,-1
11231      TEXT$(3.0)="Hand"
11240      DATA 0,0,#41,#41,#41,#7F,0,0,-1
11241      TEXT$(4.0)="eckige Klammer nach links"
19999      RETURN
```

3 Noch einige Hinweise

3.1 Grafik

Die Ausgabe eines Grafikzeichens geschieht durch:
'SHIFT' 'CHAR DEL' '1C' und einen Buchstaben
CHR$(#1C) wird also auf den Bildschirm gedruckt. Im Gegensatz zu CHR$(#1B) gelangt diese Steuerzeichen jedoch nicht auf den Drucker, sondern steuert den Computer. Der auf CHR$(#1C) folgende Buchstabe wählt wie bei den Standardformeln dann die gewünschte Grafik aus.
Zur Zeit habe folgenden Grafikzeichen definiert:

- CHR$(#1C) A
- CHR$(#1C) B
- CHR$(#1C) C
- CHR$(#1C) D
- CHR$(#1C) E

3.2 Tastaturbuffer

Diesen Teil habe ich dazugenommen, da die Zeichenverarbeitung oft hinter der Eingabegeschwindigkeit hinterherhinkt. Mit dem Buffer werden weniger Eingaben übergangen. Alles läuft etwas flüssiger. Die BREAK-Taste wird ignoriert.

3.3 Editorinhalte

Mit diesem Teil kann man Basic- oder Assemblerprogramme in den DAInatext Textbuffer übernehmen und dort dann weiterverarbeiten. Diese Beschreibung ist auf diese Weise entstanden. Dazu werden zuerst die ASC II Files auf Cassette geschrieben und danach mit dem neuen Kommando 'H' im DAInatext-Menue in den Textspeicher geladen.

3.3.1 Basic-Programme

Finde Ende der Symbol-Table >D2A3 2A4 x0 x1
Vorbereiten des Editors >SA2 ..-x0 ..-x1 ..-x0+1 ..-x1 ..-50 ..-B3
Zurück zu Basic, schreibe gewöschte Zeilen in Edit Buffer
*POKE#131,2:LIST i1-j1:LIST i2-j2:....:POKE#131,1
oder *POKE#131,2:LIST:POKE#131,1 für ganzes Programm
Speichere den erhaltenen, im Edit-Buffer stehenden, Text so ab.
>DA2 A5 x0 x1 y0 y1
>Wx1x0 y1y0 Textfile

3.3.2 Assembler source Codes

Es ist wenig sinnvoll den Inhalt des Edit Buffers abzuspeichern, da dann die Adressen und Hex-codes nicht berechnet worden sind.
Ich leite daher den RST 5 Vektor um. RST 5 DATA 3, die Routine, die Zeichen auf den Bildschirm bringt, speichert diese nun noch zusätzlich in einem Teil des RAM ab. Dieser RAM Bereich wird dann auf Cassette geschrieben und später dann von DAInatext wieder gelesen.

- Das Abspeichern geschieht wie folgt:
- Eingeben des folgenden Programms in den Assembler

```
0001      **0300   STORE    EQU   :300
0002      **0400             ORG   :400
0003 0400 F3                DI            Einsprung RST 5
0004 0401 E1                POP   H
0005 0402 E3                XTHL
0006 0403 E5                PUSH  H
0007 0404 F5                PUSH  PSW
0008 0405 7E                MOV   A,M
0009 0406 FE03              CPI   :03     Ist DATA =3
0010 0408 C21504            JNZ   RET
0011 040B F1                POP   PSW     Wenn ja, dann nehme auszugebendes
0012 040C 2A0003            LHLD  STORE    Zeichen und schreibe es in den
0013 040F 77                MOV   M,A      Speicher.
0014 0410 23                INX   H
0015 0411 220003            SHLD  STORE
0016 0414 F5                PUSH  PSW
0017 0415 F1       RET      POP   PSW
0018 0416 E1                POP   H
0019 0417 E3                XTHL
0020 0418 E5                PUSH  H
0021 0419 FB                EI
0022 041A C3FDC6            JMP   :C6FD   normale RST 5 Routine
```

- Assemblieren und den source Code löschen
- Laden des Assemblerprogramms das zu DAInatext übergeben werden soll
- Suche die höchste vom Assembler belegte Adresse (die abzuspeichernden Zeichen sollen ja nicht den Assembler oder den source Code überschreiben) Diese Endadresse ist z.B. beim Start des Basic Heap der in #29B,#29C steht, zu finden.
- >D29B 29C x0 x1
- >S300 ..-x0 ..-x1 ist die erste Adresse im Ausgabe-RAM an die ein Zeichen geschrieben wird.
- >V5 C6FD-400 umleiten von RST 5 auf die Routine
- >G1000 Startadresse Assembler (hier HT-special Assembler)
- ASSL Kommando, das ein Assemblerlisting mit berechneten Adressen ausgibt.

- UT Rücksprung zur Utility
- >Z3 RST 5 auf alten Wert
- >D300 301 x0 x1 Endadresse des RAM Bereiches in den gelistet wurde
- >D29B 29C y0 y1 Startadresse des Bereiches in den gelistet wurde
- >Wy1y0 x1x0 Textfile

In dem Textfile finden sich natürlich auch die Kommandos, die man eingegeben hat.
Eine weitere Bemerkung ist noch zu machen. Die so abgespeicherten Textfiles können recht lang werden. Beim Starten von DAInatext werden die Editbufferpointer wie folgt gesetzt: A2,A3 = #14BD Start Buffer
A4,A5 = #14BE End Buffer zur Zeit
A6,A7 = #3BCD maximale Buffergröße
Dies sind also 10000 Bytes, die beschrieben werden können. Das vorher abgespeicherte File kann aber theoretisch durchaus länger sein. Die Folge ist, daß ungewollt auch über den Textbuffer hinaus geschrieben werden würden. Man sollte also auf die Länge aufpassen.

3.4 Ändern der alten DAInatext Version

- Laden eines Assemblers
- Eingabe des source Codes wie unter Punkt 1 beschrieben
- Assemblieren
- Abspeichern auf Cassette in zwei Teilen
  >W358 36C Printer
  >WCEC E3E DAInatext Erweiterungen
- Laden des alten DAInatext-Programms
- >S29B ..-3F ..-0E
- *CLEAR 100
- >R laden von Printer
- >R laden der Erweiterungen
- Durchführen der Basic änderungen wie unter Punkt 2 beschrieben
- >D2A3 2A4 x0 x1
- >W29b x1x0 DAInatext abspeichern der erweiterten Version

Ich hoffe, daß die Anleitung möglichst fehlerlos ist und das niemand, der die Erweiterung versucht zu programmieren, Probleme haben wird.
Falls Fragen auftreten sollten wenden Sie sich bitte an:

Stefan Goller
Buschweg 14
5309 Meckenheim-Merl

September 1983

```
Ü#L.
PAGE 01    FORMATLISTING ITOH

002                 ******************************************
003                 * DGS                              30.12.83 *
004                 * Bedienung:                                *
005                 *Formatlisting ueber DBL laden              *
006                 *In Basic mit CLEAR XXXXX Editbuffer        *
007                 *dimensionieren.                            *
008                 *Basicprogramm laden,EDIT 'CR'              *
009                 *Soll ein Programmname erscheinen,jetzt     *
010                 *im Editbuffer als 1.Zeile mit space        *
011                 *beginnen,gefolgt von 'Programmname'CR.     *
012                 *  Mit RETURNTASTE abschliessen!            *
013                 *'NAME' wird auf jeder Seite eingetragen    *
014                 *Ist der Name laenger als 72 Zeichen,       *
015                 *wird er bis zu diesem Zeichen              *
016                 *eingetragen.                               *
017                 *Mit 2* BREAK Editor verlassen.             *
018                 *Printer einschalten und mit                *
019                 *CALLM #300 oder CALLM 768                  *
020                 *Das MLP starten.Es wird eine               *
021                 *fuer ITOH 8510 ausgegeben:                 *
022                 * Leftmargin 8, Schrift E, Fettschrift      *
023                 *Bei anderen Printern entsprechende         *
024                 *Sequenz bei PRI beginnend eintragen.       *
025                 *                                          *
026                 *Soll keine Steuersequenz erzeugt werden,*
027                 *so kann das MLP mit                        *
028                 *CALLM#303 oder CALLM 771 aufrufen.         *
029                 *Es wird als einziger Steuercode            *
030                 * TOF (chr$(12)) vom MLP ausgegeben.       *
031                 ******************************************
032
033                 EBUF    EQU   :A2
034                 OUTSW   EQU   :131
035                 ZL      EQU   80          ZEICHEN/ZEILE
036                 ZN      EQU   60          ZEILENZAHL/SEITE
037                 * 4 ZEILEN WERDEN FUER DEN KOPF VERBRAUCHT
038                 SCREEN  EQU   :DD60
039
040                         ORG   :300
041 0300 CD5804     INI     CALL  IPRI        ROUTINE FUER ITOH
042 0303 2AA200     STR     LHLD  EBUF
043 0306 7E                 MOV   A,M
044 0307 B7                 ORA   A
045 0308 C8                 RZ                IST BUFFER LEER,ABBRUCH
046 0309 C5                 PUSH  B           B,C RETTEN
047 030A FE20               CPI   ' '         IST ERSTES ZEICHEN ' ',
048 030C CC8403             CZ    NAME        DANN NAME IN KOPFZEILE EINTR
049 030F CDC303             CALL  LINE        LINECOUNT SETZEN
050 0312 7E         L1      MOV   A,M         ZEICHEN AUS EBUF AUSGEBEN
051 0313 23                 INX   H
052 0314 B7                 ORA   A
053 0315 CA5B03             JZ    EX          ABBRUCH BEI 0
054 0318 CD60DD             CALL  SCREEN      NUR BEI OUTSW=0 AUF RS232
055 031B FE0D               CPI   :0D
056 031D CCC303             CZ    LINE        LINECOUNT SETZEN
057 0320 3A0204             LDA   BCT         WORTZAEHLER KORR
058 0323 3D                 DCR   A
059 0324 320204             STA   BCT
060 0327 C21203             JNZ   L1
061 032A 3E0D               MVI   A,:D        NEUE ZEILE EROEFFNEN
062 032C BE                 CMP   M
063 032D CA1203             JZ    L1          WENN NOETIG
064 0330 23                 INX   H           SPACE NACH REM
065 0331 BE                 CMP   M           UND PRINT MOEGLICH
```

NP72

```
066 0332 2B              DCX   H
067 0333 CA1203          JZ    L1
068 0336 CD60DD          CALL  SCREEN
069 0339 3A0304          LDA   LCT        ZEILENZAEHLER KORR
070 033C 3D              DCR   A
071 033D 320304          STA   LCT
072 0340 C24803          JNZ   SPC1       AUFRUF FUER FORMATIERUNG
073 0343 E5              PUSH  H
074 0344 CD9A03          CALL  N2
075 0347 E1              POP   H
076                                       FORMATIERUNGSSCHLEIFE
077 0348 3E44      SPC1  MVI   A,ZL-12
078 034A 320204          STA   BCT
079 034D 060B            MVI   B,11
080 034F 3E20            MVI   A,32
081 0351 CD60DD    SPC   CALL  SCREEN
082 0354 05              DCR   B
083 0355 C25103          JNZ   SPC
084 0358 C31203          JMP   L1
085
086 035B 3E0C      EX    MVI   A,12       SCHIRM LOESCHEN
087 035D CD60DD          CALL  SCREEN     UND TOF
088 0360 3E01            MVI   A,1        RS-232 ABSCHALTEN
089 0362 323101          STA   OUTSW
090 0365 210504          LXI   H,TEXTA+1  NAME AUS KOPFZEILE
091 0368 0649            MVI   B,73       ENTFERNEN
092 036A 3E20            MVI   A,' '
093 036C 77        EX1   MOV   M,A
094 036D 23              INX   H
095 036E 05              DCR   B
096 036F C26C03          JNZ   EX1
097 0372 325304          STA   NR         UND SEITENZAHL MIT 2*SPACE
098 0375 325404          STA   NR+1       UEBERSCHREIBEN
099 0378 3E50            MVI   A,ZL       WORTCOUNT
100 037A 320204          STA   BCT
101 037D 3E01            MVI   A,1        LINECOUNT =1
102 037F 320304          STA   LCT
103 0382 C1              POP   B
104 0383 C9              RET
105
106 0384 110504    NAME  LXI   D,TEXTA+1  FUEGE NAMEN EIN
107 0387 0648            MVI   B,72       MAX. BYTES NAME
108 0389 7E        N1    MOV   A,M
109 038A B7              ORA   A
110 038B C8              RZ               ERROR,ABORT
111 038C FE0D            CPI   :D
112 038E CA9803          JZ    N4
113 0391 12              STAX  D
114 0392 23              INX   H
115 0393 13              INX   D
116 0394 05              DCR   B
117 0395 C28903          JNZ   N1
118 0398 23        N4    INX   H
119 0399 C9        FEHL  RET
120                                       SEITENZAEHLER KORRIGIEREN
121 039A 215404    N2    LXI   H,NR+1     10^0 PAGENUMBER CORR
122 039D CDBA03          CALL  NRC
123 03A0 2B              DCX   H
124 03A1 FE30            CPI   :30
125 03A3 CCBA03          CZ    NRC        10^1 PAGENUMBER CORR
126                                       TEXTAUSGABE DER KOPFZEILE
127 03A6 210404    PGN   LXI   H,TEXTA    PRINT FIXTEXT
128 03A9 0654            MVI   B,84
129 03AB 7E        N3    MOV   A,M
```

```
130 03AC CD60DD            CALL  SCREEN
131 03AF 23                INX   H
132 03B0 05                DCR   B
133 03B1 C2AB03            JNZ   N3
134 03B4 3E3C              MVI   A,ZN
135 03B6 320304            STA   LCT
136 03B9 C9                RET
137                                      ERZEUGE ZAHLEN VON 0-9
138 03BA 7E         NRC    MOV   A,M     ADJUST PAGENUMBER
139 03BB E60F              ANI   15
140 03BD 3C                INR   A
141 03BE 27                DAA
142 03BF F630              ORI   :30     IST ZAHL ='0' DANN IST
143 03C1 77                MOV   M,A     ZEROFLAG GESETZT
144 03C2 C9                RET
145                                      FORMATIERE DIE ZEILENNUMMER
146 03C3 E5         LINE   PUSH  H
147 03C4 3E50              MVI   A,ZL
148 03C6 320204            STA   BCT
149 03C9 7E                MOV   A,M
150 03CA B7                ORA   A
151 03CB CA0004            JZ    LIN1
152 03CE 3A0304            LDA   LCT
153 03D1 3D                DCR   A
154 03D2 320304            STA   LCT
155 03D5 C2E203            JNZ   LIN2
156 03D8 CD9A03            CALL  N2      PRINT NEW PAGE
157 03DB 3E3C              MVI   A,ZN
158 03DD 320304            STA   LCT
159 03E0 E1                POP   H
160 03E1 E5                PUSH  H
161                                      ANZAHL DER STELLEN BIS ' '
162 03E2 0607       LIN2   MVI   B,7
163 03E4 7E         LIN3   MOV   A,M
164 03E5 05                DCR   B
165 03E6 23                INX   H
166 03E7 FE20              CPI   ' '
167 03E9 C2E403            JNZ   LIN3
168 03EC 3E20       LIN4   MVI   A,' '    FUEHRENDE ' ' VOR ZEILENZAHL
169 03EE 05                DCR   B
170 03EF CA0004            JZ    LIN1
171 03F2 CD60DD            CALL  SCREEN
172 03F5 3A0204            LDA   BCT
173 03F8 3D                DCR   A
174 03F9 320204            STA   BCT
175 03FC 23                INX   H
176 03FD C3EC03            JMP   LIN4
177 0400 E1         LIN1   POP   H
178 0401 C9                RET
179                                      SPEICHERBEREICH VARIABLE
180 0402            BCT    RES   1,ZL
181 0403            LCT    RES   1,1
182 0404            TEXTA  RES   1,12     TOF
183 0405                   RES   73,32    TEXTBUFFER
184 044E 504147            ASC   'PAGE '
185 0453 2030       NR     ASC   ' 0'     SEITENZAEHLER 1-99
186 0455                   RES   3,:D     3* RETURN
187                                      ITOH-PROGRAMMIERUNG
188 0458 216504     IPRI   LXI   H,PRI    SETZE H,L AUF 1.ZEICHEN
189 045B 7E         ILO    MOV   A,M
190 045C B7                ORA   A        IST ZEICHEN=0 DANN ABBRUCH
191 045D C8                RZ
192 045E CD60DD            CALL  SCREEN   GIB ZEICHEN AUS UND
193 0461 23                INX   H        ERHOEHE ZEIGER
```

```
194 0462 C35B04              JMP   ILO       WEITER IN SCHLEIFE
195 0465 1B          PRI     DATA  27        INIT FUER ITOH 8510
196 0466 45                  ASC   'E'       SCHRIFT E
197 0467 1B                  DATA  27
198 0468 4C303038            ASC   'L008'    LEFTMARGIN 8
199 046C 1B                  DATA  27
200 046D 21                  ASC   '!'       BOLD CHARACTER
201 046E 00                  DATA  0         ABBRUCHBEDINGUNG FUER IPRI
202 046F                     END
```

```
****************************
* S Y M B O L   T A B L E *
****************************
```

BCT	0402	EBUF	00A2	EX	035B	EX1	036C
FEHL	0399	ILO	045B	INI	0300	IPRI	0458
L1	0312	LCT	0403	LIN1	0400	LIN2	03E2
LIN3	03E4	LIN4	03EC	LINE	03C3	N1	0389
N2	039A	N3	03AB	N4	0398	NAME	0384
NR	0453	NRC	03BA	OUTSW	0131	PGN	03A6
PRI	0465	SCREEN	DD60	SPC	0351	SPC1	0348
STR	0303	TEXTA	0404	ZL	0050	ZN	003C

```
    ü#P.
0300 CD 58 04 2A A2 00 7E B7 C8 C5 FE 20 CC 84 03 CD
0310 C3 03 7E 23 B7 CA 5B 03 CD 60 DD FE 0D CC C3 03
0320 3A 02 04 3D 32 02 04 C2 12 03 3E 0D BE CA 12 03
0330 23 BE 2B CA 12 03 CD 60 DD 3A 03 04 3D 32 03 04
0340 C2 48 03 E5 CD 9A 03 E1 3E 44 32 02 04 06 0B 3E
0350 20 CD 60 DD 05 C2 51 03 C3 12 03 3E 0C CD 60 DD
0360 3E 01 32 31 01 21 05 04 06 49 3E 20 77 23 05 C2
0370 6C 03 32 53 04 32 54 04 3E 50 32 02 04 3E 01

037F 32
0380 03 04 C1 C9 11 05 04 06 48 7E B7 C8 FE 0D CA 98
0390 03 12 23 13 05 C2 89 03 23 C9 21 54 04 CD BA 03
03A0 2B FE 30 CC BA 03 21 04 04 06 54 7E CD 60 DD 23
03B0 05 C2 AB 03 3E 3C 32 03 04 C9 7E E6 0F 3C 27 F6
03C0 30 77 C9 E5 3E 50 32 02 04 7E B7 CA 00 04 3A 03
03D0 04 3D 32 03 04 C2 E2 03 CD 9A 03 3E 3C 32 03 04
03E0 E1 E5 06 07 7E 05 23 FE 20 C2 E4 03 3E 20 05 CA
03F0 00 04 CD 60 DD 3A 02 04 3D 32 02 04 23

03FD C3 EC 03
0400 E1 C9

0402 50

0403 01

0404 0C

0405 20 20 20 20 20 20 20 20 20 20 20
0410 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0420 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0430 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0440 20 20 20 20 20 20 20 20 20 20 20 20 20 20

044E 50 41
0450 47 45 20 20 30

0455 0D 0D 0D

0458 21 65 04 7E B7 C8 CD 60
0460 DD 23 C3 5B 04 1B 45 1B 4C 30 30 3B 1B 21 00

    üH.
```

Roland Grzyb
Zieglers Weiden 7
4422 Ahaus

4422 Ahaus, 19.2.84

A S S E M B L E R S Y S T E M

Es handelt sich bei diesem Programm um einen BASIC Assembler mit in Maschinensprache geschriebenen Unterprogrammen. Der erste Start erfolgt durch *RUN 100*, dadurch werden die Mnemonic und die MLP eingepokt und die Zeiger für das Quellprogramm gesetzt. Beim Aussteigen aus dem Programm (auch bei *BREAK*) bleibt das Quellprogramm erhalten. Der erneute Start erfolgt nun mit *RUN*.

1. Folgende Befehle stehen zur Verfügung:

A	Assemblieren
B	Rückkehr zum BASIC
G	GO Maschinenprogramm
L	List
N	New
R	Read
W	Write Quellprogramm
W*	Write Maschinenprogramm
X	Autonumber
Y	Renumber
?	Peek
Zeilennr.	Anlegen des Quellprogramms

Außerdem können einige Befehle der DOS angesprochen werden. (siehe Zeilen 252-258)

2. Erklärung der Befehle:

2.1 Assemblieren
Das Quellprogramm wird in ein Maschinenprogramm umgesetzt. Dabei ist ein *ORG* anzugeben, da sonst bei Adresse 0 begonnen wird.

2.2 Basic
Wie bei der Utility. Man kann auch durch *BREAK* aus dem Programm aussteigen.

2.3 GO Maschinenprogramm
Wie bei der Utility. Es können Hex und Dez Zahlen angegeben werden (Also: G 3000 oder G #3000)

2.4 List
Wie bei BASIC. Z.B. L20-30 od. L30 od. L-1000 usw.

2.5 New
Wie bei BASIC. Man muß das *NEW* durch *RETURN* bestätigen.

2.6 Read und Write (MLP von R. Kietzmann)
Erst *R* oder *W* mit *RETURN* abschließen. Anschließend Programmname

2.7 Autonumber und Renumber
Eingabe: X(Y) erste Zeilennumer Schrittweite
Bei Autonumber wird dann beim Drücken der SPACE Taste die nächste Zeilennumer ausgegeben.

2.8 Peek
Eingabe: ?#300 oder ?400

2.9 Anlegen des Quellprogramms
Eingabe: Zeilennummer LABEL:MNEMONIC ADRESSE (bei mehrbyte Befehlen);REM

Mit freundlichen Grüßen

Roland Grzyb

NP73

```
IMP INT
10    REM 8080 Assembler Version 1.1
20    REM
30    REM (c) by Roland Grzyb  1984
40    REM
45    CLEAR 5000
50    ANF=#6500:GOTO 200:REM ** Warmstart **
100   REM ** Kaltstart **
110   CLEAR 1000:POKE #29C,#10:POKE #29B,#0:CLEAR 1000
115   COLORT 6 0 0 0:POKE #75,8
125   ANF=#6500:POKE ANF,#64:ANF=ANF-1:POKE ANF,#FC:ANF=ANF-1
130   POKE ANF,0:ANF=ANF-1:POKE ANF,0:ANF=#6500
135   B=#600:D=#D0A
140   READ A$,A
142   POKE D,B SHR 8:POKE D+1,B IAND #FF:D=D+2
143   POKE B,LEN(A$):B=B+1
145   FOR I=0 TO LEN(A$)-1
150   POKE B,ASC(MID$(A$,I,1))
155   B=B+1:NEXT
160   POKE B,A:B=B+1
165   IF A$<>"END" THEN 140
170   FOR A=#400 TO #458
175   READ B:POKE A,B
180   NEXT
185   FOR A=#300 TO #354:READ B:POKE A,B:NEXT:REM ML Search Mnemonic
190   REM ** Ende Kaltstart **
200   PRINT CHR$(12);"8080 Assembler":PRINT
205   REM Zeiger laden (steht in #6500 ff)
210   ZEIG=PEEK(ANF)*256+PEEK(ANF-1):ANF=ANF-2
215   ZNR=PEEK(ANF)*256+PEEK(ANF-1):ANF=ANF-2
220   AUTO=70000
225   REM ** Einlese Routine **
230   PRINT CHR$(9);:BEF$=""
235   A=GETC:IF A=0 GOTO 235
237   IF A=32 AND AUTO<>70000 AND BEF$="" THEN GOSUB 4200:GOTO 235:REM
       Autonumber
240   PRINT CHR$(A);
245   IF A=8 AND LEN(BEF$)>0 THEN BEF$=LEFT$(BEF$,LEN(BEF$)-1):GOTO 23
      5
250   IF A=9 THEN PRINT CHR$(12):GOTO 230
251   IF A<>13 AND A<>8 THEN BEF$=BEF$+CHR$(A):GOTO 235
252   IF BEF$="REW" THEN BEF$="":CALLM #F000:REM REW
253   IF BEF$="REW1" THEN BEF$="":CALLM #F000:REM REW1
254   IF BEF$="SKIP" THEN BEF$="":CALLM #F000:REM SKIP
255   IF BEF$="SKIP1" THEN BEF$="":CALLM #F000:REM SKIP1
256   IF BEF$="DEL" THEN BEF$="":CALLM #F000:REM DEL
257   IF BEF$="LAST" THEN BEF$="":CALLM #F000:REM LAST
258   IF BEF$="LOOK" THEN BEF$="":CALLM #F000:REM LOOK
260   IF BEF$="" THEN 230
262   POKE #2C3,#0:REM CTRL-Taste auschalten
265   A=ASC(LEFT$(BEF$,1))
270   IF (A>=48) AND (A<=57) THEN GOSUB 500:GOTO 230:REM Quellprogramm
       anlegen
275   IF A=76 THEN GOSUB 1000:GOTO 230:REM Listen
280   IF A=65 THEN GOSUB 1500:GOTO 230:REM Assemblieren
285   IF A=71 THEN GOSUB 2000:GOTO 230:REM Maschinenprogramm starten
290   IF A=78 THEN GOSUB 2500:GOTO 230:REM New
295   IF A=87 THEN GOSUB 3000:GOTO 230:REM Write
300   IF A=82 THEN GOSUB 3500:GOTO 205:REM Read
305   IF A=63 THEN GOSUB 2100:GOTO 230:REM Peek
310   IF A=89 THEN GOSUB 4000:GOTO 230:REM Renumber
315   IF A=88 THEN GOSUB 4100:GOTO 230:REM Autonumber ein- oder aussch
      alten
400   IF A=66 THEN GOTO 9500:REM Ruecksprung zum Basic
```

```
405   PRINT "*** SYNTAX ERROR ***":GOTO 230
500   REM ** Quellprogramm anlegen **
505   A=VAL(BEF$):A$=STR$(A):BEF$=RIGHT$(BEF$,LEN(BEF$)-LEN(A$)+3)
506   A$=BEF$:GOSUB 9400:BEF$=A$:IF BEF$="" THEN BYTE=0:GOTO 645
510   P1=0:P2=0
515   FOR I=0 TO LEN(BEF$)-1
520   IF MID$(BEF$,I,1)=":" THEN P1=I+1
525   IF MID$(BEF$,I,1)=";" THEN P2=I:GOTO 535
530   NEXT I
535   IF P2=0 THEN P2=LEN(BEF$)
540   A$=MID$(BEF$,P1,P2-P1):GOSUB 9400
545   I=VARPTR(A$)
550   POKE #30A,I IAND #FF
555   POKE #30B,I SHR 8
560   CALLM #300
565   I=PEEK(#2FF)
570   IF I=16 THEN PRINT "*** MNEMONIC NOT FOUND ***":RETURN
575   C=I*2+#D0A:C=PEEK(C)*256+PEEK(C+1)
577   D=PEEK(C)
580   C=C+PEEK(C)+1
590   IF PEEK(C)=0 THEN A$="":GOTO 625
595   REM ** 2 oder 3 Byte Befehl **
600   A$=RIGHT$(A$,LEN(A$)-D):GOSUB 9400
605   IF LEFT$(A$,1)="$" THEN GOTO 625
610   GOSUB 2200:REM Adresse ueberpruefen
612   IF PEEK(C)=1 AND G>255 THEN PRINT "*** NUMBER OUT OF RANGE ***":
      RETURN
615   IF G=70000 THEN RETURN
620   REM ** Einschreiben **
625   BYTE=P1-1+1+LEN(A$)+LEN(BEF$)-P2+1+3
630   IF P1=0 THEN BYTE=BYTE+1
635   REM ** Zeilennummer **
640   IF A>ZNR THEN ZNR=A:B=ZEIG:ZEIG=ZEIG-BYTE:GOTO 700
645   REM ** Programm laeuft nicht sequentiell weiter **
650   B=ANF
655   D=PEEK(B)*256+PEEK(B-1):B=B-2
660   IF D<A THEN B=B-PEEK(B)-1:GOTO 655
665   IF D>A THEN B=B+2:FOR D=ZEIG TO B:POKE D-BYTE,PEEK(D):NEXT:ZEIG=
      ZEIG-BYTE:GOTO 700:REM Einfuegen
670   REM ** Alte Zeile loeschen oder ueberschreiben **
675   IF PEEK(B)<BYTE-3 THEN E=BYTE-PEEK(B)-3:FOR D=ZEIG TO B:POKE D-E
      ,PEEK(D):NEXT:ZEIG=ZEIG-E:B=B+2:GOTO 700
680   IF PEEK(B)=BYTE-3 THEN B=B+2:GOTO 700
685   C=PEEK(B)-BYTE+3:B=B+2
690   FOR D=B TO ZEIG STEP -1:POKE D,PEEK(D-C):NEXT
695   ZEIG=ZEIG+C:IF BYTE=0 THEN 745:REM Zeig retten
700   REM ** Einpoken **
705   POKE B,A SHR 8:B=B-1:POKE B,A IAND #FF:B=B-1
706   POKE B,BYTE-3:B=B-1
710   IF P1>0 THEN FOR D=0 TO P1-2:POKE B,ASC(MID$(BEF$,D,1)):B=B-1:NE
      XT
715   POKE B,ASC(":"):B=B-1
720   POKE B,I:B=B-1
725   IF A$<>"" THEN FOR I=0 TO LEN(A$)-1:POKE B,ASC(MID$(A$,I,1)):B=B
      -1:NEXT
735   IF P2<>LEN(BEF$) THEN FOR I=P2 TO LEN(BEF$)-1:POKE B,ASC(MID$(BE
      F$,I,1)):B=B-1:NEXT
745   POKE ANF+4,ZEIG SHR 8:POKE ANF+3,ZEIG IAND #FF
755   POKE ANF+2,ZNR SHR 8:POKE ANF+1,ZNR IAND #FF
756   IF ZEIG-60<PEEK(#2A4)*256+PEEK(#2A3) THEN PRINT "*** OUT OF SPAC
      E ***":END
760   RETURN
1000  REM ** Listen **                                    NP73,2
1005  A$=RIGHT$(BEF$,LEN(BEF$)-1):GOSUB 9400
```

```
1010  IF A$="" THEN A=0:B=70000:PRINT CHR$(12):GOTO 1070
1015  IF LEFT$(A$,1)<>"-" THEN 1035
1020  A=0:A$=RIGHT$(A$,LEN(A$)-1):GOSUB 9400:B=70000
1025  IF LEN(A$)>0 THEN IF LEFT$(A$,1)>="0" AND LEFT$(A$,1)<="9" THEN
      B=VAL(A$)
1030  GOTO 1070
1035  IF LEFT$(A$,1)<"0" OR LEFT$(A$,1)>"9" THEN PRINT "*** PARAMETER
      ERROR 1 ***":RETURN
1040  A=VAL(A$):BEF$=STR$(A):A$=RIGHT$(A$,LEN(A$)-LEN(BEF$)+3):GOSUB 9
      400
1042  IF A$="" THEN B=A:GOTO 1070
1045  IF LEFT$(A$,1)<>"-" THEN PRINT "*** PARAMETER ERROR 2 ***":RETUR
      N
1050  IF LEN(A$)=1.0 THEN B=70000:GOTO 1070
1055  A$=RIGHT$(A$,LEN(A$)-1):GOSUB 9400
1060  IF LEFT$(A$,1)<"0" OR LEFT$(A$,1)>"9" THEN PRINT "*** PARAMETER
      ERROR 3 ***":RETURN
1065  B=VAL(A$)
1070  REM ** Ausdrucken **
1075  D=ANF
1080  E=PEEK(D)*256+PEEK(D-1):D=D-2.0
1085  IF D<=ZEIG THEN RETURN
1090  IF E<A THEN D=D-PEEK(D)-1:GOTO 1080
1095  IF E>B THEN RETURN
1096  I=GETC:IF I=32 THEN GOSUB 1300
1098  IF I=13 THEN RETURN
1100  PRINT E;TAB(6);
1105  E=PEEK(D):D=D-1
1110  IF CHR$(PEEK(D))=":" THEN PRINT TAB(13);":";:D=D-1:GOSUB 1200:E=
      E-1:GOTO 1125
1115  IF CHR$(PEEK(D))=";" THEN PRINT TAB(31);";";:GOTO 1125
1120  PRINT CHR$(PEEK(D));
1125  D=D-1:E=E-1:IF E=0 THEN PRINT :GOTO 1080
1130  GOTO 1110
1200  REM ** Search Mnemonic **
1205  I=PEEK(D)*2+#D0A:I=PEEK(I)*256+PEEK(I+1)
1210  P1=PEEK(I):I=I+1
1215  FOR P2=1 TO P1
1220  PRINT CHR$(PEEK(I));
1225  I=I+1:NEXT
1230  PRINT TAB(23);
1235  RETURN
1300  IF GETC=0 GOTO 1300
1305  RETURN
1500  REM ** Assemblieren **
1505  REM Sprungadressen bestimmen / First pass
1510  P1=ANF:BYTE=0:E=0:P3=ZEIG
1515  P1=P1-2:P2=P1-PEEK(P1)-1:P1=P1-1
1520  IF P1<=ZEIG+1 THEN 1705
1525  IF PEEK(P1)=#3A THEN 1565
1530  REM Label
1535  A=P3:P3=P3-1
1540  POKE P3,PEEK(P1):P1=P1-1:P3=P3-1
1545  IF PEEK(P1)<>#3A THEN 1540
1550  POKE A,A-P3-1
1555  POKE P3,(BYTE+1) SHR 8:P3=P3-1
1560  POKE P3,(BYTE+1) IAND #FF:P3=P3-1
1565  REM OP-Code
1570  P1=P1-1:A=PEEK(P1)
1575  IF A<>8 AND A<>24 AND A<>40 AND A<>32 THEN 1670
1580  REM ORG, DATA, EQU oder RESERVE
1585  A$="":P1=P1-1
1590  A$=A$+CHR$(PEEK(P1)):P1=P1-1
1595  IF PEEK(P1)<>#3B AND P1<>P2 THEN 1590
```

```
1600  IF A<>24 THEN 1625:REM Begin DATA
1605  IF LEFT$(A$,1)="$" THEN BYTE=BYTE+LEN(A$)-1:P1=P2:GOTO 1515
1610  GOSUB 2200
1615  BYTE=BYTE+1:IF G>255 THEN BYTE=BYTE+1
1620  P1=P2:GOTO 1515:REM Ende DATA
1625  IF A<>32 THEN 1660:REM Begin EQU
1630  P3=P3+2
1635  GOSUB 2200
1640  IF G=70000 THEN PRINT "*** EQU ERROR ***":RETURN
1645  POKE P3,G SHR 8:P3=P3-1
1650  POKE P3,G IAND #FF:P3=P3-1
1655  P1=P2:GOTO 1515:REM Ende EQU
1660  GOSUB 2200:BYTE=BYTE+G:IF A=8 THEN BYTE=G-1:REM ORG and RESERVE
1665  P1=P2:GOTO 1515
1670  REM Kein Assembler Pseudo-Befehl
1675  BYTE=BYTE+1:P1=P1-1
1680  IF PEEK(P1)=#3B OR P1=P2 THEN P1=P2:GOTO 1515
1685  REM 2 oder 3 Byte Befehl
1690  A=PEEK(P1+1)*2+#D0A:A=PEEK(A)*256+PEEK(A+1)
1695  A=A+PEEK(A)+1:BYTE=BYTE+PEEK(A)
1700  P1=P2:GOTO 1515
1705  REM ******** Second pass ***************
1710  BYTE=0:P1=ANF:FLAG=0
1715  P1=P1-2:P2=P1-PEEK(P1)-1:P1=P1-1
1720  IF P1<=ZEIG+1 THEN PRINT "ASSEMBLY TERMINATED, NO ERRORS FOUND":
      RETURN
1725  IF PEEK(P1)<>#3A THEN P1=P1-1:GOTO 1725:REM Label unterdruecken
1730  P1=P1-1:A=PEEK(P1)
1735  IF A<>8 AND A<>24 AND A<>40 AND A<>32 THEN 1830
1740  REM Assembler Pseudo Befehl
1745  IF A=32 THEN P1=P2:GOTO 1715:REM EQU
1750  A$="":P1=P1-1
1755  A$=A$+CHR$(PEEK(P1)):P1=P1-1
1760  IF PEEK(P1)<>#3B AND P1<>P2 THEN 1755
1775  IF A<>40 THEN 1790
1777  REM RESERVE
1780  GOSUB 2200:IF G=70000 THEN PRINT "*** RESERVE ERROR ***":RETURN
1785  FOR A=1 TO G:POKE BYTE,0:BYTE=BYTE+1:NEXT
1787  P1=P2:GOTO 1715:REM Ende Reserve
1790  IF A<>8 THEN GOTO 1795
1792  GOSUB 2200:BYTE=G
1793  IF FLAG=0 THEN FLAG=G
1794  P1=P2:GOTO 1715:REM Ende ORG
1795  REM DATA
1800  IF LEFT$(A$,1)="$" THEN FOR A=1 TO LEN(A$)-1:POKE BYTE,ASC(MID$(
      A$,A,1)):BYTE=BYTE+1:NEXT:P1=P2:GOTO 1715
1805  GOSUB 2200
1810  IF G<255 THEN POKE BYTE,G:BYTE=BYTE+1:P1=P2:GOTO 1715
1815  POKE BYTE,G IAND #FF:BYTE=BYTE+1
1820  POKE BYTE,G SHR 8:BYTE=BYTE+1
1825  P1=P2:GOTO 1715
1830  REM OP-Code
1835  POKE BYTE,A:BYTE=BYTE+1:P1=P1-1
1840  IF PEEK(P1)=#3B OR P1=P2 THEN P1=P2:GOTO 1715
1845  A$=""
1850  A$=A$+CHR$(PEEK(P1)):P1=P1-1
1855  IF PEEK(P1)<>#3B AND P1<>P2 THEN 1850
1860  A=A*2+#D0A:A=PEEK(A)*256+PEEK(A+1)
1865  A=PEEK(A+PEEK(A)+1)
1870  IF LEFT$(A$,1)="$" THEN 1900
1875  GOSUB 2200
1880  IF A=1 THEN POKE BYTE,G:BYTE=BYTE+1:P1=P2:GOTO 1715
1885  POKE BYTE,G IAND #FF:BYTE=BYTE+1
1890  POKE BYTE,G SHR 8:BYTE=BYTE+1
```

```
1895  P1=P2:GOTO 1715
1900  REM Label suchen
1905  A$=RIGHT$(A$,LEN(A$)-1)
1910  B=ZEIG:G=0:GOTO 1920
1915  B=B-PEEK(B)-3:IF B<=P3 THEN PRINT "*** ";A$;" NOT FOUND ***":RET
      URN
1920  IF PEEK(B)<>LEN(A$) THEN 1915
1925  D=B-1
1930  FOR E=0 TO PEEK(B)-1
1935  IF MID$(A$,E,1)<>CHR$(PEEK(D)) THEN 1915
1940  D=D-1:NEXT
1945  G=PEEK(D)*256+PEEK(D-1):GOTO 1880
1950  REM *** Ende Assembler ***
2000  REM ** Maschinenprogramm starten **
2005  A$=RIGHT$(BEF$,LEN(BEF$)-1):GOSUB 9400
2010  GOSUB 2200:IF G=70000 THEN RETURN
2015  CALLM G
2020  RETURN
2100  REM *** P E E K ***
2105  A$=RIGHT$(BEF$,LEN(BEF$)-1):GOSUB 2200
2110  IF G=70000 THEN RETURN
2115  PRINT HEX$(PEEK(G)):RETURN
2200  REM ** Adresse ueberpruefen **
2205  IF LEFT$(A$,1)="#" THEN 2235
2210  FOR G=0 TO LEN(A$)-1
2215  IF MID$(A$,G,1)<"0" OR MID$(A$,G,1)>"9" THEN 2270
2220  NEXT
2225  G=VAL(A$):IF G>#FFFF THEN 2275
2230  RETURN
2235  G=0:K=0
2240  FOR H=LEN(A$)-1 TO 1 STEP -1
2242  L=ASC(MID$(A$,H,1))
2245  IF L<48 OR L>70 THEN 2270
2246  IF L>57 AND L<65 THEN 2270
2250  IF L>64 THEN L=L-7
2252  G=INT(G+(L-#30)*16^K+0.5):K=K+1
2255  NEXT
2260  IF G>#FFFF THEN 2275
2265  RETURN
2270  G=70000:PRINT "*** ADRESS ERROR ***":RETURN
2275  G=70000:PRINT "*** NUMBER OUT OF RANGE ***":RETURN
2500  REM ** NEW **
2505  PRINT "Bestaetigen Sie das 'NEW' durch 'RETURN' !";
2510  A=GETC:IF A=0 THEN GOTO 2510
2515  IF A<>13 THEN PRINT :PRINT "Kein 'NEW' !!":RETURN
2520  PRINT :ZEIG=ANF:ZNR=0
2525  POKE ANF+4,#64:POKE ANF+3,#FC
2530  POKE ANF+2,0:POKE ANF+1,0
2535  RETURN
3000  REM *** Write ***
3005  IF LEN(BEF$)>1 THEN IF MID$(BEF$,1,1)="*" THEN P1=FLAG:P2=BYTE:G
      OTO 3015
3010  P1=ZEIG:P2=ANF+4
3015  INPUT "Programmname ";A$:PRINT :POKE #13F,LEN(A$)
3020  POKE #42C,P1 IAND #FF
3025  POKE #42D,P1 SHR 8
3030  POKE #42E,P2 IAND #FF
3035  POKE #42F,P2 SHR 8
3040  ANF=#6500:CALLM #400:ANF=ANF-4:RETURN
3045  REM *** Ende Write ***
3500  REM *** Read ***
3505  INPUT "Welches Programm moechten Sie einladen ";A$
3510  ANF=ANF+4:PRINT :POKE #13F,LEN(A$):CALLM #430:RETURN
3520  REM *** Ende Read ***
```

```
4000  REM ** Renumber **
4005  GOSUB 4400:P1=ANF
4010  IF A=70000 THEN PRINT "*** PARAMETER ERROR ***":RETURN
4015  POKE P1,A SHR 8:P1=P1-1
4020  POKE P1,A IAND #FF:P1=P1-1
4025  P1=P1-PEEK(P1)-1:A=A+B
4030  IF P1<=ZEIG THEN ZNR=A-B:GOTO 745:REM ZNR retten
4035  GOTO 4015
4100  REM REM *** Autonumber ein- oder ausschalten
4105  GOSUB 4400:IF A=70000 THEN AUTO=0:RETURN:REM Auto aus
4110  AUTO=A-B:X=B:RETURN:REM Auto ein
4200  REM *** Autonumber ***
4205  AUTO=AUTO+X:IF AUTO>#FFFF THEN PRINT "*** NUMBER OUT OF RANGE **
      *":RETURN
4206  BEF$=STR$(AUTO):BEF$=LEFT$(BEF$,LEN(BEF$)-2)+" "
4208  A$=BEF$:GOSUB 9400:BEF$=A$
4210  CURSOR CURX-1,CURY:PRINT AUTO;" ";
4215  P1=CURX:CURSOR 0,CURY:PRINT CHR$(9);:CURSOR P1,CURY
4400  A$=RIGHT$(BEF$,LEN(BEF$)-1):GOSUB 9400
4402  IF LEN(A$)=0 THEN A=70000:RETURN
4405  IF LEFT$(A$,1)<"0" OR LEFT$(A$,1)>"9" THEN A=70000:RETURN
4410  A=VAL(A$)
4412  IF LEN(A$)+2<=LEN(STR$(A)) THEN A=70000:RETURN
4415  A$=RIGHT$(A$,LEN(A$)+2-LEN(STR$(A))):GOSUB 9400
4417  IF LEN(A$)=0 THEN A=70000:RETURN
4420  IF LEFT$(A$,1)<"0" OR LEFT$(A$,1)>"9" THEN A=70000:RETURN
4425  B=VAL(A$):RETURN
9400  REM *** Leerzeichen unterdruecken ***
9405  IF LEN(A$)>0 THEN IF RIGHT$(A$,1)=" " THEN A$=LEFT$(A$,LEN(A$)-1
      ):GOTO 9405
9410  IF LEN(A$)>0 THEN IF LEFT$(A$,1)=" " THEN A$=RIGHT$(A$,LEN(A$)-1
      ):GOTO 9410
9415  RETURN
9500  REM Rueckkehr zum Basic
9505  PRINT CHR$(12);"BASIC V1.1"
11600 DATA NOP,0,LXI B,2,STAX B,0,INX B,0,INR B,0,DCR B,0,MVI B,1
11602 DATA RLC,0,ORG,2,DAD B,0,LDAX B,0,DCX B,0,INR C,0,DCR C,0,MVI C,
      1
11604 DATA RRC,0
11606 DATA EMP,0,LXI D,2,STAX D,0,INX D,0,INR D,0,DCR D,0,MVI D,1
11608 DATA RAL,0,DATA,2,DAD D,0,LDAX D,0,DCX D,0,INR E,0,DCR E,0,MVI E
      ,1
11610 DATA RAR,0
11612 DATA EQU,2,LXI H,2,SHLD,2,INX H,0,INR H,0,DCR H,0
11614 DATA MVI H,1,DAA,0,RES,2,DAD H,0,LHLD,2,DCX H,0,INR L,0,DCR L,0
11616 DATA MVI L,1,CMA,0
11618 DATA EMP,0,LXI SP,2,STA,2,INX SP,0,INR M,0,DCR M,0,MVI M,1,STC,0

11620 DATA EMP,0,DAD SP,0,LDA,2,DCX SP,0,INR A,0,DCR A,0,MVI A,1,CMC,0

11622 DATA MOV B.B,0,MOV B.C,0,MOV B.D,0,MOV B.E,0,MOV B.H,0
11624 DATA MOV B.L,0,MOV B.M,0,MOV B.A,0
11626 DATA MOV C.B,0,MOV C.C,0,MOV C.D,0,MOV C.E,0,MOV C.H,0
11628 DATA MOV C.L,0,MOV C.M,0,MOV C.A,0
11630 DATA MOV D.B,0,MOV D.C,0,MOV D.D,0,MOV D.E,0,MOV D.H,0
11632 DATA MOV D.L,0,MOV D.M,0,MOV D.A,0
11634 DATA MOV E.B,0,MOV E.C,0,MOV E.D,0,MOV E.E,0,MOV E.H,0
11636 DATA MOV E.L,0,MOV E.M,0,MOV E.A,0
11638 DATA MOV H.B,0,MOV H.C,0,MOV H.D,0,MOV H.E,0,MOV H.H,0
11640 DATA MOV H.L,0,MOV H.M,0,MOV H.A,0
11642 DATA MOV L.B,0,MOV L.C,0,MOV L.D,0,MOV L.E,0,MOV L.H,0
11644 DATA MOV L.L,0,MOV L.M,0,MOV L.A,0
11646 DATA MOV M.B,0,MOV M.C,0,MOV M.D,0,MOV M.E,0,MOV M.H,0
11648 DATA MOV M.L,0,HLT,0,MOV M.A,0
```

```
11650 DATA MOV A.B,0,MOV A.C,0,MOV A.D,0,MOV A.E,0,MOV A.H,0
11652 DATA MOV A.L,0,MOV A.M,0,MOV A.A,0
11654 DATA ADD B,0,ADD C,0,ADD D,0,ADD E,0,ADD H,0,ADD L,0
11656 DATA ADD M,0,ADD A,0
11658 DATA ADC B,0,ADC C,0,ADC D,0,ADC E,0,ADC H,0,ADC L,0,ADC M,0,ADC
      A,0
11660 DATA SUB B,0,SUB C,0,SUB D,0,SUB E,0,SUB H,0,SUB L,0,SUB M,0,SUB
      A,0
11662 DATA SBB B,0,SBB C,0,SBB D,0,SBB E,0,SBB H,0,SBB L,0,SBB M,0,SBB
      A,0
11664 DATA ANA B,0,ANA C,0,ANA D,0,ANA E,0,ANA H,0,ANA L,0,ANA M,0,ANA
      A,0
11666 DATA XRA B,0,XRA C,0,XRA D,0,XRA E,0,XRA H,0,XRA L,0,XRA M,0,XRA
      A,0
11668 DATA ORA B,0,ORA C,0,ORA D,0,ORA E,0,ORA H,0,ORA L,0,ORA M,0,ORA
      A,0
11670 DATA CMP B,0,CMP C,0,CMP D,0,CMP E,0,CMP H,0,CMP L,0,CMP M,0,CMP
      A,0
11672 DATA RNZ,0,POP B,0,JNZ,2,JMP,2,CNZ,2,PUSH B,0,ADI,1,RST 0,0
11674 DATA RZ,0, RET,0,JZ,2,EMP,0,CZ,2,CALL,2,ACI,1,RST 1,0
11676 DATA RNC,0,POP D,0,JNC,2,OUT,1,CNC,2,PUSH D,0,SUI,1,RST 2,0
11678 DATA RC,0,EMP,0,JC,2,IN,1,CC,2,EMP,0,SBI,1,RST 3,0
11680 DATA RPO,0,POP H,0,JPO,2,XTHL,0,CPO,2,PUSH H,0,ANI,1,RST 4,0
11682 DATA RPE,0,PCHL,0,JPE,2,XCHG,0,CPE,2,EMP,0,XRI,1,RST 5,0
11684 DATA RP,0,POP PSW,0,JP,2,DI,0,CP,2,PUSH PSW,0,ORI,1,RST 6,0
11686 DATA RM,0,SPHL,0,JM,2,EI,0,CM,2,EMP,0,CPI,1,RST 7,0,END,0
11700 REM Read & Write von R. Kietzmann (NP 59)
11705 DATA #F5,#E5,#C5,#D5,#3E,#F0,#32,#40,#00,#32,#06,#FD,#21,#1F,#04
      ,#E5
11710 DATA #2A,#2E,#04,#EB,#2A,#2C,#04,#E5,#D5,#21,#3F,#01,#C3,#F0,#EE
      ,#3E
11715 DATA #30,#32,#40,#00,#32,#06,#FD,#D1,#C1,#E1,#F1,#C9,#00,#00,#00
      ,#00
11720 DATA #F5,#E5,#C5,#D5,#3E,#F0,#32,#40,#00,#32,#06,#FD,#21,#4A,#04
      ,#E5
11725 DATA #2A,#57,#04,#E5,#21,#3F,#01,#C3,#17,#EF,#3E,#30,#32,#40,#00
      ,#32
11730 DATA #06,#FD,#D1,#C1,#E1,#F1,#C9,#00,#00
11740 REM Suchen des Mnemonic
11745 DATA #F5,#C5,#D5,#E5,#11,0,6,6,0,#2A,#58,#3F,#1A,#4F,#C,#13
11750 DATA #23,#1A,#0D,#CA,#41,#03,#BE,#CA,#0F,#03,#78,#FE,#FF,#CA,#39
      ,#03
11755 DATA #04,#1A,#FE,#00,#CA,#35,#03,#FE,#01,#CA,#35,#03,#FE,#02,#CA
      ,#35
11760 DATA #03,#13,#C3,#21,#03,#13,#C3,#09,#03,#3E,#10,#32,#FF,#02,#C3
      ,#50
11765 DATA #03,#FE,#00,#CA,#4C,#03,#7E,#FE,#20,#C2,#1A,#03,#78,#32,#FF
      ,#02
11770 DATA #E1,#D1,#C1,#F1,#C9
```

Der Assembler legt seinen Quellcode ab #64FC abwärts ab, also Richtung Basic-Programm. In den Adressen #6500 und #64FF liegt die Variable ZEIG%, die das untere Ende des Quellprogramms anzeigt. (Inhalt beim Start: #64FC). In den Adressen #64FE u. #64FD liegt die Variable ZNR%, die höchste Zeilennummer des Programms.
Beim Assemblieren werden die Label mit den daugehörigen Adressen unter dem Quellprogramm abgelegt.
Das Basic-Programm wird beim Starten durch RUN 100 auf #1000 hochgelegt. Ab Adresse #300 liegen einige ML-Programme und Daten:
#300 - #354 ML Search Mnemonic
#400 - #458 READ AND WRITE von R. Kietzmann
#600 - #D09 Mnemonics
#D0A - #D0A+2*256 Startadressen der Mnmonics

Beispiel

```
20    MNEM    :EQU      §2FF     ;Mnemonic suchen
30            :ORG      §300
32            :PUSH PSW
33            :PUSH B
34            :PUSH D
35            :PUSH H
40            :LXI D    §600     ;Anfang der DATA
50            :MVI B    0
60    BEGIN   :LHLD     §4E1B
70            :LDAX D
72            :MOV C.A
80            :INR C
90    NCHAR   :INX D
100           :INX H
110           :LDAX D
120           :DCR C
130           :JZ       $ENDE    ;Richtige Mnemonic ?
140           :CMP M             ;Richtiges Zeichen ?
150           :JZ       $NCHAR
160   LBL 1   :MOV A.B
170           :CPI      §FF      ;Mnemonic not found ?
180           :JZ       $ERR
190           :INR B
200   KONTR   :LDAX D
210           :CPI      0
220           :JZ       $NMNE
230           :CPI      1
240           :JZ       $NMNE
250           :CPI      2
260           :JZ       $NMNE
270           :INX D
290           :JMP      $KONTR
300   NMNE    :INX D
305           :JMP      $BEGIN
310   ERR     :MVI A    16
320           :STA      $MNEM
330           :JMP      $RET
340   ENDE    :CPI      00
350           :JZ       $END
360           :MOV A.M
370           :CPI      32
375           :JZ       $LBL 1
380   END     :MOV A.B
390           :STA      §2FF
400   RET     :POP H
410           :POP D
420           :POP B
430           :POP PSW
440           :RET
```

C I R C L E

von Ralf Degenhardt

CIRCLE ist ein Maschinenprogramm zum Zeichnen von Kreisen. Daß es dabei sehr schnell ist, braucht wohl kaum erwähnt zu werden.

Es funktioniert nur in MODE 6 und erkennt selbstständig die Bildschirmgrenzen (es können also am Rand z.B. auch Halbkreise gezeichnet werden). Je 4 Punkte werden gleichzeitig gesetzt.

Das Programm selbst belegt die Adressen #300-#48A, es benötigt aber zusätlich eine Cosinus-Tabelle von #48F bis #61F. Diese Tabelle wird hier nicht mit ausgedruckt, da das Abtippen weitaus aufwendiger wäre als das folgende kleine BASIC-Programm, das diese Tabelle erstellt: (vorher Heap auf #620 oder höher setzen!)

```
100 A%=#48F
110 FOR N=0 TO PI/2.0 STEP 1/255.0
120 B=255*COS(N)+0.5:B%=B
130 POKE A%,B%:A%=A%+1
140 NEXT
145 REM ersten und letzten Wert ändern für 'rundere' Kreise!
150 POKE #48F,#FE:POKE #61F,#01
```

Das Programm benötigt als Parameter:

```
Radius  R% von 0 - 255    (bei 0 Rücksprung)
X-Mitte X% von 0 - 335
Y-Mitte Y% von 0 - 255
Farbe   Q% von 0 - 3      (entsprechend COLOR6 C0 C1 C2 C3)
```

Der Aufruf des MP CIRCLE erfolgt mit:

```
VAR%=(R% SHL 24)+(Y% SHL 16)+(Q% SHL 12)+X%
CALLM #300,VAR%
```

```
0300 F5 C5 D5 E5 E5 AF 7E A7 CA 80 04 32 75 04 21 49
0310 04 23 23 23 23 BE FA 11 03 23 7E 32 76 04 23 7E
0320 32 77 04 23 7E 32 78 04 E1 23 7E 32 79 04 23 7E
0330 32 7B 04 23 7E 32 7A 04 21 8E 04 22 7C 04 21 20
0340 06 22 1E 04 2A 77 04 AF 7C A7 C2 53 03 AF 7D A7
0350 CA 81 04 2B 22 77 04 2A 7C 04 16 00 3A 76 04 5F
0360 19 22 7C 04 CD 20 04 7A 32 1D 04 2A 1E 04 16 FF
0370 7B 2F 5F 19 23 22 1E 04 CD 20 04 7A 32 7E 04 00
0380 3A 1D 04 47 AF 3A 79 04 88 47 D4 A0 03 3A 1D 04
0390 47 AF 3A 79 04 98 47 D4 A0 03 C3 44 03 A2 77 C9
03A0 2A 7A 04 CD 86 04 3A 7E 04 5F 16 00 19 11 B0 FE
03B0 E5 19 AF 7C A7 E1 C4 D0 03 2A 7A 04 CD 86 04 3A
03C0 7E 04 2F 5F 16 FF 19 AF 7C 23 E6 80 CC D0 03 C9
03D0 E5 7C E6 01 0F 4F 7D E6 F8 0F 81 0F 2F 5F 16 FF
03E0 21 46 66 19 E3 7D E6 07 21 45 04 85 6F 7E 4F 68
03F0 26 00 29 E5 29 29 E5 29 E5 29 29 D1 19 D1 19 D1
0400 19 D1 19 79 2F 57 3A 7B 04 5F E6 20 7E CA 41 04
0410 B1 77 2B 7B E6 10 7E CA 9D 03 B1 77 C9 00 00 00
0420 7E 57 01 00 00 21 00 00 3A 75 04 47 AF 78 1F 47
0430 79 1F 4F AF 7A 17 57 D2 3B 04 09 A7 C2 2C 04 54
0440 C9 A2 C3 11 04 80 40 20 10 08 04 02 01 80 01 91
0450 01 56 02 C8 00 40 03 85 00 34 04 64 00 2B 05 50
0460 00 25 06 42 00 20 07 39 00 1D 08 32 00 1A 09 2C
0470 00 00 0A 28 00 00 00 00 00 00 00 00 00 00 00 00
0480 E1 E1 D1 C1 F1 C9 7C E6 01 67 C9
```

Demoprogramm für MP CIRCLE von Ralf Degenhardt

Dies Programm enthält 4 kleine, aber sehr wirkungsvolle Demos für das Maschinenprogramm CIRCLE. Das MP muß hierbei unterhalb des Heap bei #300 im Ram liegen! (Das Abtippen lohnt sich (Red.))

```
                                                        Demo1:
10    COLORG 3 12 13 14:MODE 6:MODE 6
20    FOR N=0.0 TO 25.5 STEP 0.1
21      X%=167+100*SIN(N):Y%=127+100*COS(N)
23      R%=R%+1:Q%=1+RND(3.0):GOSUB 200
31    NEXT
                                                        Demo2:
40    COLORG 8 0 3 15:MODE 6:MODE 6
41    Q%=1
42    FOR Y%=0 TO 255 STEP 5
43      R%=Y%:X%=Y%:GOSUB 200
46    NEXT
50    Q%=3
51    FOR Y%=0 TO 255 STEP 5
52      R%=Y%:X%=335-Y%:GOSUB 200
55    NEXT
60    Q%=2
61    X%=167
62    FOR Y%=255 TO 0 STEP -5
63      R%=255-Y%:GOSUB 200
65    NEXT
                                                        Demo3: (saustark! (Red.))
70    COLORG 0 1 10 14:MODE 6:MODE 6
71    P=0.0
72    R%=15:Q%=1
73    FOR M=0.0 TO 8.0*PI STEP 8.0/40.0
74      FOR N=0.0 TO 7.0/40.0 STEP 1.0/40.0
75        X%=P+40*SIN(M+N):Y%=127+40*COS(M+N)
77        P=P+0.3:GOSUB 200
79      NEXT
80      Q%=Q%+1:IF Q%=4 THEN Q%=1
81    NEXT
                                                        Demo4:
90    COLORG 1 0 3 5:MODE 6:MODE 6
91    X%=167
92    Q%=3
93    FOR A%=80 TO 4 STEP -2
94      Y%=80+A%:R%=82-A%:GOSUB 200
97    NEXT
99    Q%=2
100   FOR A%=0 TO 20 STEP 2
101     Y%=84+A%:R%=78+A%:GOSUB 200
104   NEXT
109   Q%=3
110   FOR A%=40 TO 0 STEP -2
111     Y%=64+A%:R%=138-A%:GOSUB 200
114   NEXT
119   Q%=2
120   FOR A%=0 TO 116 STEP 2
121     Y%=64+A%:R%=138+A%:GOSUB 200
124   NEXT
180   GOTO 180
                                                        CIRCLE-Routine:
200   VAR%=(R% SHL 24)+(Y% SHL 16)+(Q% SHL 12)+X%
210   CALLM #300,VAR%
220   RETURN
```

*** DAI - SYSTEM - EXTENSION ***

Nach Starten des Basicprogramms wird ein ML-Programm von #400 - #562 geschrieben und nach 'READY !' aufgerufen. Das ML-Programm soll als Hilfsprogramm zur Programmerstellung und Programmaustestung dienen.
Wird die Funktionstaste <^> gedrückt, erscheit als Promt ein Fragezeichen '?' und eine Eingabe wird erwartet, die mit einem <RETURN> abgeschlossen werden muß.
Das Programm arbeitet auch im UTILITY. Es kann dort zum DISPLAYSTOP genutzt werden (<^> => stop <BREAK> => cont).
Nach >Z3 muß es mit CALLM #400 neu aufgerufen werden.

Folgende Funktionen sind möglich :

?<RETURN> => Der Bildschirm wird gelöscht (HOME).

?BS => Die Basic-Pointer werden gesichert. Sollte das Programm abstürzen oder wird RESET versehentlich gedrückt, kann das Progsamm nach CALLM #400 und..

?BL => Laden der Basic-Pointer, ... gerettet werden.

?Rnnnn => 'Warmstart' RUN mit Zeilennummer. Alle Variablen bleiben trotz eines Programmabruchs erhalten.

?M => MERGING: Sollen zwei Basicprogramme zusammengefügt werden, muß nach Laden des 1. Programms ?M gedrückt werden. Der Basicmonitor meldet sich nach kurzer Zeit mit *BREAK . Danach ist das 2. Programm zu laden und ?E <RETURN>,<RETURN> zu drücken. Der SYNTAX ERROR ist ohne Bedeutung. (Die Programme müssen unterschiedliche Zeilennummern aufweisen.)

?E => POKE #135,2:<RETURN>,<RETURN>

?C => Der Cursor wird auf _ zurückgesetzt.

?P => Der Start of BASIC wird auf #5EC zurücksetzt und ein CLEAR #100 durchgeführt.

?0 => MODE 0 wird gewählt. (z.B. nach Grafiken)

?V => Das Programm wird abgeschaltet.(Reset V6)

Es kann mit CALLM #400 wieder aufgerufen werden.

Viel Spaß ! Wilfried Rüsse

P.S. Die Funktionstaste kann über POKE #41D,n frei gewählt werden (Zeile 50). Für DAI-Freaks, die keine D C R angeschlossen haben, bietet sich die nicht belegte <TAB>-Taste (n=9) an.

```
5     REM : * DAI - SYSTEM - EXTENSION *
6     REM : Copyright by Wilfried Ruesse
7     REM :  (C) 3.1984, Hofheim/Taunus
10    PRINT CHR$(12);:MODE 0:COLORT 8 0 0 0:POKE #29B,#EC
20    POKE #29C,#5:CLEAR #100:I1=#400:I2=I1+#161
25    PRINT " ###   NEW  FUNKTIONS  ==>  PRESS <^>  ###":PRINT
26    PRINT " <RETURN> ==> HOME (PRINT CHR$(12)":PRINT
27    PRINT " <BS>     ==> SAVE BASIC-POINTER":PRINT
28    PRINT " <BL>     ==> LOAD BASIC-POINTER":PRINT
29    PRINT " <R>      ==> RUN nnnn (no clear)":PRINT
30    PRINT " <M>      ==> PREPARE M E R G E":PRINT
31    PRINT " <E>      ==> POKE #135,2:<RETURN>:<RETURN>":PRINT
32    PRINT " <C>      ==> RESET CURSOR  *"+CHR$(#5F):PRINT
33    PRINT " <P>      ==> BASIC-POINTER TO #5EC":PRINT
34    PRINT " <0>      ==> MODE 0":PRINT
35    PRINT " <V>      ==> RESET V6 (END FUNKTIONS)":PRINT
36    PRINT " CALLM #400 ==> START FUNKTIONS AGAIN";
40    FOR I=I1 TO I2:READ A:POKE I,A:NEXT
45    CALLM I1:PRINT "  /  READY  !"
50    POKE #41D,94:REM 94=<^> (FUNKTION-KEY):ADR(ML) #400 - #562
51    DATA #E5,#21,#9,#4,#22,#6E,#0,#E1,#C9,#F3,#21,#12
52    DATA #4,#E5,#E5,#C3,#78,#D5,#F3,#C5,#D5,#E5,#F5,#21
53    DATA #BA,#2,#16,#4,#3E,#9,#BE,#CA,#2E,#4,#23,#15
54    DATA #C2,#1E,#4,#F1,#E1,#D1,#C1,#E1,#FB,#C9,#36,#0
55    DATA #21,#0,#0,#39,#22,#71,#2,#0,#21,#78,#D5,#22
56    DATA #6E,#0,#FB,#3E,#3F,#CD,#1A,#DD,#DA,#59,#4,#CD
57    DATA #D2,#DD,#FE,#D,#C2,#7B,#4,#3E,#C,#EF,#3,#C3
58    DATA #59,#4,#CD,#5E,#DD,#F3,#3A,#40,#0,#E6,#C0,#3E
59    DATA #3E,#C2,#66,#4,#3E,#2A,#CD,#60,#DD,#21,#B9,#2
60    DATA #16,#5,#36,#0,#23,#15,#C2,#6E,#4,#CD,#0,#4
61    DATA #C3,#27,#4,#21,#35,#5,#1E,#1,#CD,#34,#CA,#D2
62    DATA #91,#4,#5E,#23,#56,#21,#56,#4,#E5,#EB,#E9,#E1
63    DATA #E9,#CD,#55,#DD,#21,#23,#DC,#CD,#D4,#DA,#C3,#56
64    DATA #4,#3E,#5F,#32,#75,#0,#C9,#3E,#D2,#32,#D2,#0
65    DATA #CD,#24,#C0,#21,#3E,#1,#E7,#F,#23,#36,#89,#21
66    DATA #0,#0,#22,#42,#1,#1,#3F,#1,#C3,#C6,#4,#9A
67    DATA #FF,#0,#0,#1,#BF,#4,#3A,#40,#0,#E6,#C0,#C2
68    DATA #91,#4,#CD,#0,#4,#C3,#92,#C8,#E1,#C3,#C,#C8
69    DATA #2A,#EC,#5,#22,#45,#0,#6,#A,#21,#9B,#2,#11
70    DATA #47,#0,#7E,#12,#23,#13,#5,#C2,#E6,#4,#C9,#2A
71    DATA #45,#0,#22,#EC,#5,#6,#A,#11,#9B,#2,#21,#47
72    DATA #0,#C3,#E6,#4,#21,#EC,#5,#22,#9B,#2,#11,#0
73    DATA #1,#CD,#BC,#E6,#C9,#11,#0,#30,#CD,#BC,#E6,#3E
74    DATA #C,#EF,#3,#CD,#65,#E2,#CD,#9F,#E1,#CD,#75,#DD
75    DATA #CD,#B8,#DE,#3E,#0,#32,#31,#1,#CD,#0,#4,#E1
76    DATA #C3,#C,#C8,#3E,#2,#32,#35,#1,#C9,#1,#43,#9D
77    DATA #4,#1,#30,#BF,#4,#1,#55,#E3,#2,#1,#52,#A3
78    DATA #4,#1,#56,#D4,#4,#2,#42,#53,#D8,#4,#2,#42
79    DATA #4C,#EF,#4,#1,#50,#0,#5,#1,#45,#2F,#5,#1
80    DATA #4D,#D,#5,#0,#0,#0,#0
```

F i l e k n a c k e r

```
1     POKE #131,1
2     REM Dieses Programm ist imstande, jedes beliebige
3     REM File auf der Diskette im LF 0 genau zu lokali-
4     REM sieren und dessen Inhalt auszudrucken.
5     REM - Ausdruck nach Wahl im HEX-Code, Diasassembliert,
6     REM im ASCII-Code oder als Charackters.
7     REM Die Sektoren sind dabei nummeriert.
8     REM Autor : GERHARD NOLL
9     REM .       FLORIAN-GEYER-STR.17
10    REM .       7100 HEILBRONN
11    REM geschrieben im April 1984.
12    REM Im Programm bitte jetzt die Leertastes druecken!
13    PRINT CHR$(12);:LIST 2-11:PRINT :PRINT :PRINT :LIST 12
14    CALLM #D6DA:REM DRUECKE LEERTASTE
16    CLEAR 12000
19    GOSUB 6000:REM DISASSEMBLER LADEN
20    PRINT CHR$(12);
25    INPUT "Gebe den Filenamen ein (ohne Kuerzel). nun:";FILE$
30    PRINT :INPUT "Gebe das Kuerzel ein (z.B. 'BAS')..... nun";KUERZ
      EL$
40    PRINT :PRINT "Das File heisst ";FILE$;".";
45    IF LEN(KUERZEL$)>0 THEN PRINT KUERZEL$;" .":GOTO 50:PRINT :REM
      ELSE
50    PRINT "Ist dieser Name korrekt...?(J/N)"
55    G=GETC
60    G=GETC:IF G<>74 AND G<>78 THEN 60
70    IF G=78 THEN 20
75    URFILE$=FILE$:URKUERZEL$=KUERZEL$:REM FUER oi
80    FILE$=FILE$+SPC(8-LEN(FILE$))
90    KUERZEL$=KUERZEL$+SPC(8-LEN(KUERZEL$))
100   VERGLEICH$=FILE$+KUERZEL$:OK=0
110   POKE #131,3:PRINT "OPENI $DKDIR"
120   PRINT "ASSIGN INPUT FROM DISK"
125   PRINT "ASSIGN OUTPUT TO PRINTER"
130   FOR I=1 TO 18
140   FOR J=1 TO 6
150   A$=""
160   FOR K=1 TO 21
170   A$=A$+CHR$(GETC)
180   NEXT K
190   IF LEFT$(A$,16)=VERGLEICH$ THEN OK=1:GOSUB 2000
200   NEXT J
210   K=GETC+GETC
215   IF OK=1 THEN 230
220   NEXT I
230   POKE #131,3:PRINT "ASSIGN INPUT FROM KEYBOARD"
235   POKE #131,1
237   IF OK=0 THEN PRINT FILE$;".";KUERZEL$;" ex. nicht!!":WAIT TIME
      100:GOTO 20
240   POKE #131,0
245   PRINT "File         : ";FILE$;".";KUERZEL$
250   PRINT "Spurnummer   : ";SPUR
260   PRINT "Sektornummer: ";SEKTOR
270   PRINT "Sektoranzahl: ";ANZAHL
275   POKE #131,1
280   PRINT :PRINT :PRINT "Soll das File ..."
290   PRINT "In HEX ............ 1"
295   PRINT "Disassembliert .... 2"
300   PRINT "In ASCII ........... 3"
310   PRINT "In CHRs ............ 4"
312   PRINT "In LESBAREN CHRs .. 5"
315   PRINT "Garnicht ........... 6"
320   PRINT "... Ausgedruckt werden?"
325   WAHL=GETC
330   WAHL=GETC:IF WAHL<49 OR WAHL>54 THEN 330
```

NP 76.1

```
350   WAHL=WAHL-48
355   DIM B(128)
357   IF URKUERZEL$="BAS" AND WAHL=2 THEN PRINT "File ist ein BASIC-P
      rogramm!":GOTO 450
360   POKE #131,3:PRINT "RESETD":PRINT "OPENI "+URFILE$;
362   IF URKUERZEL$<>"" THEN PRINT ","+URKUERZEL$;
370   PRINT :PRINT "ASSIGN INPUT FROM DISK"
371   PRINT "ASSIGN OUTPUT TO PRINTER"
372   IF WAHL=2 THEN 6100
375   IF WAHL=4 THEN GOSUB 2500
380   FOR I=1 TO ANZAHL
385   SPU=SPUR+((SEKTOR+I-2)/18)
388   SEABS=(SEKTOR+I-2) MOD 18+1
390   PRINT I;". Sektor relativ.";SPU;
395   PRINT ". Spur,";SEABS;
397   PRINT ". Sektor absolut."
400   FOR J=1 TO 128
410   ON WAHL GOSUB 2200,510,2300,2400,2600:REM 510 IST EIN DUMMY
420   NEXT J
425   PRINT
430   NEXT I
435   PRINT "*** END OF FILE":REM RUECKSPRUNG VON DISASS-ROUTINE
440   POKE #131,3:PRINT "ASSIGN INPUT FROM KEYBOARD":POKE #131,1
450   PRINT "Mit selbem File weiterarbeiten ..1"
460   PRINT "Mit neuem File weiterarbeiten ...2"
470   PRINT "Ende ..............................3"
475   WAHL=GETC
480   WAHL=GETC:IF WAHL<49 OR WAHL>51 THEN 480
490   WAHL=WAHL-48
500   ON WAHL GOTO 240,20
510   END
2000  SPUR=ASC(MID$(A$,17,1))
2010  SEKTOR=ASC(MID$(A$,18,1))
2020  ANZAHL=ASC(MID$(A$,19,1))*256
2030  ANZAHL=ASC(MID$(A$,20,1))+ANZAHL
2040  RETURN
2200  DECOD$=HEX$(GETC):REM HEX
2210  IF LEN(DECOD$)=1 THEN PRINT "0";
2220  PRINT DECOD$+" ";:IF CURX>47 THEN PRINT
2230  RETURN
2300  B(J)=GETC:REM ASCII
2310  PRINT B(J);:IF CURX>55 THEN PRINT
2320  RETURN
2400  DECOD$=CHR$(GETC):REM CHR
2410  IF DECOD$=CHR$(8) THEN DECOD$="^08^"
2420  IF DECOD$=CHR$(12) THEN DECOD$="^12^"
2430  IF DECOD$=CHR$(13) THEN DECOD$="^13^"
2440  PRINT DECOD$;:IF CURX>55 THEN PRINT
2450  RETURN
2500  PRINT "Ausdruck in CHRs"
2510  PRINT "================="
2520  PRINT "Beachte :     CH DEL <==> '^08^'"
2530  PRINT "                 FF  <==> '^12^'"
2540  PRINT "                 CR  <==> '^13^'"
2550  RETURN
2600  B(J)=GETC:REM LESBARE CHRs
2610  IF B(J)<32 OR B(J)>126 THEN PRINT "*";:GOTO 2630
2620  PRINT CHR$(B(J));
2630  IF CURX>55 THEN PRINT
2640  RETURN
6000  REM DISASSEMBLER LADEN
6010  DIM C$(2)
6020  DIM A$(255),A(255)
6030  PRINT "DISASSEMBLER LADEN"
6040  FOR T=0 TO 255
6050  READ A$(T),A(T)
```

```
6060  NEXT
6070  RETURN
6100  REM DISASSEMBLER-ROUTINE
6110  I=1:REM ZEIGER AUF DAS NAECHSTE BYTE
6120  GOSUB 7000
6130  AN=ZWI:REM ANFANGSADRESSE LADEN
6140  GOSUB 7000
6150  EN=ZWI:REM ENDADRESSE LADEN
6160  GOSUB 7000
6170  ESPR=ZWI:REM EINSPRUNGADRESSE LADEN
6180  I=7:REM SETZE DATENZEIGER
6190  IF EN-AN<=0 OR EN-AN>=ANZAHL*128 THEN PRINT "File ist kein Masc
      hinenprogramm!":GOSUB 7100:GOTO 440
6195  PRINT "am Zeilenbeginn steht die rel. Sektornummer, die Nummer
      des"
6196  PRINT "Bytes im Sektor, sowie die Position des Datenzeigers."
6200  PRINT "    Programm von HEX ";HEX$(AN);"  bis HEX ";HEX$(EN)
6210  IF ESPR>0 THEN PRINT "    Einsprung : ";HEX$(ESPR)
6220  PRINT :PRINT :PRINT
6300  REM STEUERUNG ASSEMBLERAUSDRUCK
6310  X=GETC:REM LADE BEFEHLSCODE
6320  MN$=A$(X)
6330  IF MN$="EMP" THEN PRINT :PRINT "Code #";HEX$(X);" existiert nic
      ht bei HEX ";HEX$(I+AN-7);"  (dez. : ";I+AN-7;" )":PRINT :GOTO
      6460
6340  MN$=MN$+SPC(12-LEN(MN$))
6350  AD$="000"+HEX$(I+AN-7):AD$=" "+RIGHT$(AD$,4):REM BILDE ADRESSE
6360  B$="0"+HEX$(X)+" ":B$=RIGHT$(B$,3):REM BILDE BEFEHLSCODE
6370  R=I+A(X)
6375  IF R=I THEN 6420
6380  C$(R-I)=HEX$(GETC):REM LADE DATACODE
6390  C$(R-I)="0"+C$(R-I):C$(R-I)=RIGHT$(C$(R-I),2):REM BILDE DATACOD
      E
6400  B$=B$+C$(R-I)+" "
6410  R=R-1:GOTO 6375
6420  V$=""
6430  IF A(X)=1.0 THEN V$="Data    "+C$(1.0)
6440  IF A(X)=2.0 THEN V$="Adr.    "+C$(1.0)+C$(2.0)
6445  SNR=(I-1)/128+1:REM SNR=SEKTORNUMMER
6447  ZPOS=(I-1) MOD 128+1:REM ZPOS=ZEIGERPOSITION
6450  PRINT SNR;" /";ZPOS,MN$;AD$;"    ";B$,V$
6460  I=I+A(X)+1:REM AKTUALISIERE ZEIGER
6470  IF I+AN-7<=EN THEN 6310
6480  GOSUB 7100:GOTO 435:REM ENDE ASSEMBLER
7000  ZWI=GETC:REM LADE
7010  ZWI=GETC*256+ZWI
7020  RETURN
7100  IF I MOD 128=1 THEN 7140:REM SCHLIESSE SEKTOR
7110  DUMMY=GETC
7120  I=I+1
7130  GOTO 7100
7140  RETURN
10010 DATA NOP,0,LXI B,2,STAX BC,0,INX BC,0,INR B,0,DCR B,0,MVI B,1,R
      LC,0,EMP,0,DAD BC,0,LDAX BC,0,DCX BC,0,INR C,0,DCR C,0
10020 DATA MVI C,1,RRC,0,EMP,0,LXI D,2,STAX DE,0,INX DE,0,INR D,0,DCR
       D,0,MVI D,1,RAL,0,EMP,0,DAD DE,0,LDAX DE,0,DCX DE,0,INR E,0
10030 DATA DCR E,0,MVI E,1,RAR,0,EMP,0,LXI H,2,SHLD,2,INX HL,0,INR H,
      0,DCR H,0,MVI H,1,DAA,0,EMP,00,DAD HL,0,LHLD,2,DCX HL,0
10040 DATA INR L,0,DCR L,0,MVI L,1,CMA,0,EMP,0,LXI SP,2,STA,2,INX SP,
      0,INR M,0,DCR M,0,MVI M,1,STC,0,EMP,0,DAD SP,0,LDA,2
10050 DATA DCX SP,0,INR A,0,DCR A,0,MVI A,1,CMC,0 ,MOV B-B,0,MOV B-C,
      0,MOV B-D,0,MOV B-E,0,MOV B-H,0,MOV B-L,0,MOV B-M,0
10060 DATA MOV B-A,0,MOV C-B,0,MOV C-C,0,MOV C-D,0,MOV C-E,0,MOV C-H,
      0,MOV C-L,0,MOV C-M,0,MOV C-A,0
10070 DATA MOV D-B,0,MOV D-C,0,MOV D-D,0,MOV D-E,0,MOV D-H,0,MOV D-L,
      0,MOV D-M,0,MOV D-A,0
```

NF 76,3

```
10080 DATA MOV E-B,0,MOV E-C,0,MOV E-D,0,MOV E-E,0,MOV E-H,0,MOV E-L,
      0,MOV E-M,0,MOV E-A,0
10090 DATA MOV H-B,0,MOV H-C,0,MOV H-D,0,MOV H-E,0,MOV H-H,0,MOV H-L,
      0,MOV H-M,0,MOV H-A,0
10100 DATA MOV L-B,0,MOV L-C,0,MOV L-D,0,MOV L-E,0,MOV L-H,0,MOV L-L,
      0,MOV L-M,0,MOV L-A,0
10110 DATA MOV M-B,0,MOV M-C,0,MOV M-D,0,MOV M-E,0,MOV M-H,0,MOV M-L,
      0,MOV M-M,0,MOV M-A,0
10120 DATA MOV A-B,0,MOV A-C,0,MOV A-D,0,MOV A-E,0,MOV A-H,0,MOV A-L,
      0,MOV A-M,0,MOV A-A,0
10130 DATA ADD B,0,ADD C,0,ADD D,0,ADD E,0,ADD H,0,ADD L,0,ADD M,0,AD
      D A,0
10140 DATA ADC B,0,ADC C,0,ADC D,0,ADC E,0,ADC H,0,ADC L,0,ADC M,0,AD
      C A,0
10150 DATA SUB B,0,SUB C,0,SUB D,0,SUB E,0,SUB H,0,SUB L,0,SUB M,0,SU
      B A,0
10160 DATA SBB B,0,SBB C,0,SBB D,0,SBB E,0,SBB H,0,SBB L,0,SBB M,0,SB
      B A,0
10170 DATA ANA B,0,ANA C,0,ANA D,0,ANA E,0,ANA H,0,ANA L,0,ANA M,0,AN
      A A,0
10180 DATA XRA B,0,XRA C,0,XRA D,0,XRA E,0,XRA H,0,XRA L,0,XRA M,0,XR
      A A,0
10190 DATA ORA B,0,ORA C,0,ORA D,0,ORA E,0,ORA H,0,ORA L,0,ORA M,0,OR
      A A,0
10200 DATA CMP B,0,CMP C,0,CMP D,0,CMP E,0,CMP H,0,CMP L,0,CMP M,0,CM
      P A,0
10210 DATA RNZ,0,POP BC,0,JNZ,2,JMP,2,CNZ,2,PUSH BC,0,ADI,1,RST 0,0,R
      Z,0,RET,0,JZ,2,EMP,0,CZ,2,CALL,2,ACI,1,RST 1,1,RNC,0
10215 DATA POP DE,0,JNC,2
10220 DATA OUT,1,CNC,2,PUSH DE,0,SUI,1,RST 2,0,RC,0,EMP,0,JC,2,IN,1,C
      C,2,EMP,0,SBI,1,RST 3,0
10230 DATA RPO,0,POP HL,0,JPO,2,XTHL,0,CPO,2,PUSH HL,0,ANI,1,RST 4,1,
      RPE,0
10240 DATA PCHL,0,JPE,2,XCHG,0,CPE,2,EMP,0,XRI,1,RST 5,1,RP,0,POP PSW
      ,0
10250 DATA JP,2,DI,0,CP,2,PUSH PSW,0,ORI,1,RST 6,0,RM,0,SPHL,0,JM,2,E
      I,0,CM,2,EMP,0,CPI,1,RST 7,0
20000 POKE #131,0
20010 FF$=CHR$(12)
20020 LIST 1-320:PRINT FF$;
20030 LIST 325-510:PRINT FF$;
20040 LIST 2000-2640:PRINT FF$;
20050 LIST 6000-7140:PRINT FF$;
20060 LIST 10000-10250:PRINT FF$;
20070 POKE #131,1
```

INTERAKTIVROUTINE

Dieses Programm ersetzt alle Label-Basic-Hilfen, ist in der Lage, Funktionen und 'PRINT'- oder 'DATA'-Zeilen in das Programm aufzunehmen, ohne es abbrechen zu müssen, Kann (fast) beliebige Zeilen einfügen und löschen uvam.

Das Programm schreibt eine Maschinenroutine in den Heap und stellt die Variable INTAKT% als Entrypoint zur Verfügung. (Probleme mit Vektoren kann es beim Verschieben im Speicher nicht geben.) Die Benutzung geschieht etwa folgendermassen:

```
100     A$="1000 LIST 111-222"+CHR$(13)
110     CALLM INTAKT%,A$
```

Nach Abarbeitung der beiden Zeilen enthielte das Programm die zusätzliche '1000 LIST 111-222'. Eine evtl. vorhandene Zeile 1000 währe dabei gelöscht worden.

- An diesen beiden Zeilen sieht man bereits die Hauptmerkmale der Routine.
- Der Text, den der String enthält, wird in das Programm genommen, sofern er gültige Anweisungen enthält. (Dazu gehören eine korrekte Zeilennummer ohne Dezimalteil (STR$(VAR) erzeugt einen solchen!), korrekte Befehle und ein <CR> (CHR$(13))) Ist in dem String ein Fehler, so bricht das Programm mit der entsprechenden Fehlermeldung ab.

Die einzigen Beschränkungen sind erstens: Der CALLM-Aufruf darf nicht in einer FOR-NEXT-Schleife erfolgen (???). (Diese sind gegebenenfalls durch IF-THEN-Bedingungen zu umschreiben.) Und zweitens darf in der Regel keine Zeilennummer vor dem Aufruf verändert werden. (Die gleich darauffolgende darf sich ändern !)

```
20000   REM ××× INTERAKTIVROUTINE BY VOLKER RAAB ×××
20010   DIM INTAKT(11.0)
20020   INTAKT=VARPTR(INTAKT(0.0))
20030   FOR N=0 TO 45
20040   READ P
20050   POKE INTAKT+N,P
20060   NEXT N
20100   DATA #C5,#D5,#E5,#F5,#5E,#23,#56,#2A,#32,#1
20110   DATA #E5,#2A,#34,#1,#E5,#EB,#7E,#23,#22,#32
20120   DATA #1,#21,#34,#1,#77,#23,#36,#1,#E,#0
20130   DATA #CD,#18,#C9,#E1,#22,#34,#1,#E1,#22,#32
20140   DATA #1,#F1,#E1,#D1,#C1,#C9
```

Viel Spaß beim interaktiven Programmieren.

Volker Raab

NP77

```
100   REM  RUN Zeilennummer ohne Löschen der Variablen auch bei V1.0
110   REM            entnommen aus: DAINAMIC-B 83/15/138
120   REM                  Anwendungsbeispiel von:
130   REM         EMIL ZAHNER CH 8910 AFFOLTERN 8.9.84
140   REM
150   REM  Das Programm benutzt den ENVELOPE Speicher #235...#253
160   REM  um das Maschinenprogramm abzulegen
170   REM
180   PRINT CHR$(12);
190   REM    Das Maschinenprogramm wird eingelesen
200   FOR ADRESSE=#235 TO #253:READ BEFEHL:POKE ADRESSE,BEFEHL:NEXT
210   REM
220   SCROLLZ=13:SCROLLZ=#B3E5+SCROLLZ*#86:POKE #8A,SCROLLZ MOD 256
230   POKE #8B,SCROLLZ SHR 8
240   REM  Den SCROLL Bereich des Bildschirms auf den Rest ab
250   REM  Zeile 13 einschränken
260   FARPOS2=10:POKE #BFEE-FARPOS2*#86,203
270   REM  Für diesen Bereich eine andere Hintergrundfarbe wählen
280   PRINT SPC(28);"CHECK":PRINT " während Ablauf eines Programms"
290   PRINT " Befehl zum Weiterfahren nach CHECK :"
300   PRINT :PRINT SPC(20);"CALLM RUN,ZEILE":RUN=#235:ZEILE=350
310   REM  Die Variable Zeile beinhaltet die Zeilennummer ab der
320   REM  der 'RUN' beginnen soll
330   STOP  : REM   CHECK ausprobieren anschließend  CALLM RUN,ZEILE
340   REM
350   REM   Ab hier kann dann weiter gearbeitet werden, wobei
360   REM   alle Variablen ihre alten Werte behalten !!!

1000  REM   -------      Daten des Maschinenprogramms      -------
1010  DATA #23,#23,#56,#23,#5E,#EB,#CD,#F6,#CA,#44,#4D,#CD,#01,#E4
1020  DATA #21,#00,#00,#22,#15,#01,#AF,#32,#26,#01,#31,#00,#F9,#B7
1030  DATA #C3,#8F,#C8
```

Generelles Verfahren beim Einschränken des SCROLL Bereiches:
Es werden zwei Werte benötigt: OBEN und UNTEN
Diese beiden Variablen geben den Zeilenbereich an, in dem der Text nach oben geschoben werden soll. Es ist somit möglich, oben und unten am Bildschirm z.B. Statusmeldungen auszugeben und alle Eingaben in der Mitte machen zu lassen. Dieses Verfahren hat den Vorteil, daß beim 'nach-oben-rollen' des Bildschirms die Statusmeldungen nicht zerstört werden.
OBEN sind die Zeilen, die von oben durch den SCROLL nicht angetastet werden dürfen; UNTEN entsprechend die unteren Zeilen.

Nun sind noch zwei Zuweisungen und vier POKEs durchzuführen:

```
TOP=#BFEF-134*OBEN:BOT=#BFEF-134*(24-UNTEN)
POKE #8A,TOP MOD 256:POKE #8B,TOP / 256 : Hiermit wird der obere und
POKE #8C,BOT MOD 256:POKE #8D,BOT / 256 : hiermit der untere Bereich
                                          eingeschränkt
```

Beispiel: OBEN=4:UNTEN=1; d.h. es bleiben oben vier und unten eine Zeile unberücksichtigt; Das SCROLLING bezieht sich also auf die Zeilen 1-19

Anmerkungen:
Die Adressen (#8A,#8B) und (#8C,#8D) sollten nicht mit den gleichen Werten beschrieben werden.
Die Befehle ?CHR$(12), MODE 0, EDIT und LIST stellen wieder die alten Zustände her; der ganze Bildschirminhalt wird ab sofort wieder bewegt.

Tilgungsplan - Problembeschreibung

Bei der Aufnahme eines Kredits ist es wichtig zu wissen, wie hoch die Belastung pro Monat/Jahr ist, wie lange man abzahlt und wieviel man zahlt. Mit diesen Daten kann man verschiedene Darlehen miteinander vergleichen.

Vorgehensweise:
Zum Vergleich verschiedener Darlehen dient häufig ein sogenannter Effektivzins. In diesem Programm wird nun nicht dieser Effektivzins berechnet, sondern einige andere Vergleichswerte. Dazu muß man zunächst die benötigte Darlehenshöhe (DA), das Disagio (DI), sowie die Bearbeitungsgebühr in % (BE) eingeben. Daraus wird die aufzunehmende Darlehenshöhe nach der Formel:

```
SA=INT(DA*100/(100-DI-BE)
```

berechnet. Danach verlangt das Programm die Eingabe des Zinssatzes. Die pro Zinsperiode zu bezahlende Prämie ergibt sich aus:

```
PRS=SA*(ZS+TS)/100
```

Die Zinsen ergeben sich aus

```
ZINS=SA*ZS/100
```

und die Tilgung aus

```
TILG=PRS-ZINS
```

Die Darlehenshöhe SN nach einer Periode ist dann:

```
SN=SA-TILG
```

Diese Werte werden für jede Zinsperiode berechnet und ausgegeben. Zusätzlich werden sie summiert und am Ende angezeigt. In der letzten Zinsperiode muß eine etwas abgewandelte Formel für die Tilgung verwendet werden.

```
100   REM  --------- Tilgungsplan ----------------
102   REM  Autor: R.Meinert
104   REM
110   COLORT 8 1 0 0
120   REM
130   REM VARIABLEN
140   REM DA     Benoetigtes Darlehen
150   REM DI     Disagio
160   REM BE     Bearbeitungsgebuehr
170   REM ZS     Zinssatz
180   REM TS     Tilgungssatz
190   REM SA     Saldo - alt
200   REM SN     Saldo - neu
210   REM GS     Benoetigtes Gesamtdarlehn
220   REM GP     Gesamtpraemien
230   REM GZ     Gesamtzinsen
240   REM J      Zinsperiode
250   REM PRS    pro Zinsperiode zu zahlende Praemie
260   REM ZINS   Zinsen einer Zinsperiode
270   REM TILG   Tilgung einer Zinsperiode
280   REM
290   MODE 0:PRINT CHR$(12);
300   PRINT "Tilgungsplan"
305   PRINT "-------------"
310   INPUT "Benoetigtes Darlehen in DM eingeben";DA!:PRINT
320   INPUT "Disagio in % eingeben";DI!:PRINT
330   INPUT "Bearbeitungsgebuehr in % eingeben";BE!:PRINT
340   SA!=INT(DA!*100.0/(100.0-DI!-BE!))
350   GS!=SA!
```

Tilgungsplan - Problembeschreibung

Bei der Aufnahme eines Kredits ist es wichtig zu wissen, wie hoch die Belastung pro Monat/Jahr ist, wie lange man abzahlt und wieviel man zahlt. Mit diesen Daten kann man verschiedene Darlehen miteinander vergleichen.

Vorgehensweise:
Zum Vergleich verschiedener Darlehen dient häufig ein sogenannter Effektivzins. In diesem Programm wird nun nicht dieser Effektivzins berechnet, sondern einige andere Vergleichswerte. Dazu muß man zunächst die benötigte Darlehenshöhe (DA), das Disagio (DI), sowie die Bearbeitungsgebühr in % (BE) eingeben. Daraus wird die aufzunehmende Darlehenshöhe nach der Formel:

SA=INT(DA*100/(100-DI-BE)

berechnet. Danach verlangt das Programm die Eingabe des Zinssatzes. Die pro Zinsperiode zu bezahlende Prämie ergibt sich aus:

PRS=SA*(ZS+TS)/100

Die Zinsen ergeben sich aus

ZINS=SA*ZS/100

und die Tilgung aus

TILG=PRS-ZINS

Die Darlehenshöhe SN nach einer Periode ist dann:

SN=SA-TILG

Diese Werte werden für jede Zinsperiode berechnet und ausgegeben. Zusätzlich werden sie summiert und am Ende angezeigt. In der letzten Zinsperiode muß eine etwas abgewandelte Formel für die Tilgung verwendet werden.

```
100   REM  --------- Tilgungsplan -----------------
102   REM  Autor: R.Meinert
104   REM
110   COLORT 0 1 0 0
120   REM
130   REM VARIABLEN
140   REM DA     Benoetigtes Darlehen
150   REM DI     Disagio
160   REM BE     Bearbeitungsgebuehr
170   REM ZS     Zinssatz
180   REM TS     Tilgungssatz
190   REM SA     Saldo - alt
200   REM SN     Saldo - neu
210   REM GS     Benoetigtes Gesamtdarlehn
220   REM GP     Gesamtpraemien
230   REM GZ     Gesamtzinsen
240   REM J      Zinsperiode
250   REM PRS    pro Zinsperiode zu zahlende Praemie
260   REM ZINS   Zinsen einer Zinsperiode
270   REM TILG   Tilgung einer Zinsperiode
280   REM
290   MODE 0:PRINT CHR$(12);
300   PRINT "Tilgungsplan"
305   PRINT "-------------"
310   INPUT "Benoetigtes Darlehen in DM eingeben";DA!:PRINT
320   INPUT "Disagio in % eingeben";DI!:PRINT
330   INPUT "Bearbeitungsgebuehr in % eingeben";BE!:PRINT
340   SA!=INT(DA!*100.0/(100.0-DI!-BE!))
350   GS!=SA!
```

```
001                     * D I S K - C O M M A N D - H A N D L E R
002                     * A.Pfeifle  02.04.1983
003                     * verbessert 28.05.1984 /**
004                     * dieses Programm laedt und startet Maschinen-
005                     * programme als
006                     * Kommandos, deren Code auf Disk steht.
007                     * Aufruf der Kommandos erfolgt durch
008                     *   #<Name der COM-Datei, die Code enthaelt>
009                     * Der Code muss ab STBS beginnen
010                     * und dort auch gestartet werden koennen.
011                     *
012                     * Definitionen:
013                     *
014                     POPRET  EQU   :C14D
015                     CRLF    EQU   :DD5E
016                     ENVST   EQU   :01F5     Envelope Storage
017                     HEAP    EQU   :029B
018                     TBLNK   EQU   :297      DOS Command Table Link
019                     DOSEND  EQU   :1B00     hinter Druckertreiber
020                     ADDA    EQU   :DE30     HL:=HL+A
021                     NEW     EQU   :DEB5     Fuehrt BASIC-NEW durch
022                     OTSW    EQU   :0131
023                     OUTCPR  EQU   :D695     Output Accu
024                     ENCINP  EQU   :DDE0     Input from Command Line
025                     PMSGR   EQU   :DAFF
026                     ENDBS   EQU   :27FF     Ende res.Raum fuer Betr.sys
027                     STBS    EQU   :2000     Start fuer Betriebssystem
028                     ERRADR  EQU   :9B7      notierte DOS-Fehler
029                     FAULT   EQU   :5CE      DOS-Error Report
030                     RESETD  EQU   :477      Disk-Reset
031                     POPRT2  EQU   :75C      POP B,D,H,PSW;RET
032                     MOVE    EQU   :DE4F
033                     DRIVE   EQU   :9A7      Laufwerk-Nr.
034                     FILNAM  EQU   :992      Filename-Buffer
035                     FILEXT  EQU   :99A      File-Extension Buffer
036                     *
037                     * Initialisierung des Handlers:
038                     *  dies steht im Binaerfile IDKCOM.BIN
039                     *
040                             ORG   ENVST
041 01F5 F5         INIT       PUSH  PSW
042 01F6 C5                    PUSH  B
043 01F7 D5                    PUSH  D
044 01F8 E5                    PUSH  H
045 01F9 113002                LXI   D,FILE    setze Filename in Buffer
046 01FC 213B02                LXI   H,EFILE
047 01FF 019209                LXI   B,FILNAM
048 0202 CD4FDE                CALL  MOVE
049 0205 3E30                  MVI   A,'0'     Laufwerk 0 setzen
050 0207 32A709                STA   DRIVE
051 020A CDA910                CALL  :10A9     in DLOAD-Routine weiter
052 020D C4CE05                CNZ   FAULT     falls Fehler
053 0210 3E01                  MVI   A,1       keine Druckerausg.
054 0212 323101                STA   OTSW
055 0215 C22D02                JNZ   BYE       bei DOS-Fehler keine
056                                            Handler-Installation
057 0218 2A9702                LHLD  TBLNK     Verbinde neues Kommando
058 021B 22051B                SHLD  CONTA
059 021E 21001B                LXI   H,COMTB
060 0221 229702                SHLD  TBLNK
061 0224 21002B                LXI   H,ENDBS+1 neuer Heapbeginn
062 0227 229B02                SHLD  HEAP
063 022A CDB5DE                CALL  NEW
064 022D C34DC1         BYE    JMP   POPRET    *** Zum Aufrufer
065 0230 444B43         FILE   ASC   'DKCOM   BIN'                    NP 80,1
066 023B 00             EFILE  ***
```

```
0000          ;    ********************
0000          ;    *                  *
0000          ;    *   R E L E A S E  *
0000          ;    *                  *
0000          ;    ********************
0000          ;
0000          ;
0000          ; Mit dem DOS-Kommando  #RELEASE filename.ext
0000          ; wird das angegebene File in der der Directory als
0000          ; Directory-Ende markiert, und damit das angegebene File mit
0000          ; allen physikalisch dahinterliegenden Files geloescht.
0000          ; Dadurch koennen diese Sektoren wieder neu belegt werden.
0000          ;
0000          ; Das Programm setzt in der vorliegenden Form voraus, dass
0000          ; es vom Disk Command Handler DKCOM aufgerufen wird.
0000          ;
0000          ; Stand: 8.6.84  A.Pfeifle
0000          ;
0000          ; Referenzen zum DA1-pc DOS 1.0
0000          ;
0000  §=11E6 SRCFN   EQU       11E6H          ;liest Filename von Input und sucht
0000  §=0DA4 ERRES   EQU       0DA4H          ;evtl. Error-Report; RESETD und RET
0000  §=1298 REWRIT  EQU       1298H          ;setzt (A) als neues Attribut und
0000          ;                               ;schreibt ihn in die Directory zurueck.
0000          ;
0000          ;
0000                 ORG       2000H
2000  F5      START   PUSH PSW
2001  E5              PUSH H
2002  D5              PUSH D
2003  C5              PUSH B
2004  CD0A20          CALL      DORELS
2007  C3A40D          JMP       ERRES          ;fertig
200A          ;
200A  CDE611 DORELS   CALL      SRCFN
200D  C0              RNZ                      ;Fehler
200E  3E09            MVI A     9H
2010  3D              DCR A                    ;Z=0
2011  D8              RC                       ;File not found
2012  3EFF            MVI A     0FFH           ;End of Direct. - Attribut
2014  C39812          JMP       REWRIT
2017          ;
2017                 END

DORELS:200A  ERRES :0DA4  REWRIT:1298  SRCFN :11E6  START :2000
```

```
0000            ;
0000            ;
0000                    ORG     1B00H           ;HINTER DOS
1B00  C5        LINPUT  PUSH B                  ;SAVE BASIC-IP
1B01  E5                PUSH H                  ;SAVE STRING-VARPTR
1B02  AF                XRA A
1B03  32B709            STA     ERRADR          ;ERRORFLAG LOESCHEN
1B06  013E01            LXI B   EBUF
1B09  CD121B            CALL    INCHRS          ;STRING EINLESEN
1B0C  E1                POP H
1B0D  CDB8E4            CALL    ASSTR2          ;UND ZUWEISEN
1B10  C1                POP B
1B11  C9                RET                     ;ZUM BASIC-AUFRUFER
1B12            ;
1B12            ; HIER WIRD DER STRING EINGELESEN:
1B12            ;
1B12            ; ABBRUCH DES EINLESENS BEIM AUTRETEN FOLGENDER ZEICHEN:
1B12  §=0000 NUL        EQU     0H              ;/ DA DAS DOS NUR BEIM PHYSIKALISCHEN
1B12  §=0003 ETX        EQU     3H              ;/ END OF FILE ABBRICHT.
1B12  §=000D CR         EQU     0DH             ;REGULAERES ZEILENENDE
1B12            ;
1B12  C5        INCHRS  PUSH B                  ;EINGELESENER STRING WIRD ALS
1B13  3E18              MVI A   18H             ;/ STRING KONSTANTE ABGELEGT
1B15  02                STAX B                  ;/ WEGEN ASSTR2-ROUTINE
1B16  03                INX B
1B17  C5                PUSH B
1B18  1E00              MVI E   0H              ;ZAEHLER STRINGLAENGE
1B1A  167D              MVI D   MAXCH           ;VERHINDERT BUFFERUEBERLAUF
1B1C  3AB709    LOOP    LDA     ERRADR
1B1F  B7                ORA A                   ;BEENDE EINLESEN,
1B20  C2431B            JNZ     EOF2            ;FALLS FEHLER AUFGETRETEN
1B23  CDBED6            CALL    GETC            ;ZEICHEN HOLEN
1B26  B7                ORA A                   ;UND UEBERPRUEFEN:
1B27  CA411B            JZ      EOF             ;NUL EINGELESEN
1B2A  FE03              CPI     ETX
1B2C  CA411B            JZ      EOF
1B2F  FE0D              CPI     CR              ;ENDE ZEILE?
1B31  CA461B            JZ      OK2
1B34  03        OK1     INX B
1B35  02                STAX B                  ;ZEICHEN IN BUFFER
1B36  1C                INR E                   ;LAENGE
1B37  15                DCR D
1B38  C21C1B            JNZ     LOOP            ;FALLS LAENGE NICHT UEBERSCHR.
1B3B  0B                DCX B                   ;SONST ZEILE ZU ENDE LESEN
1B3C  1D                DCR E
1B3D  14                INR D
1B3E  C31C1B            JMP     LOOP
1B41            ;
1B41  3E0E      EOF     MVI A   0EH             ;END OF FILE ERROR MELDEN
1B43  CD5617    EOF2    CALL    INPERR
1B46  C1        OK2     POP B                   ;EINGABE BEENDEN
1B47  7B                MOV A,E
1B48  02                STAX B                  ;STRINGLAENGE SETZEN
1B49  C1                POP B
1B4A  C9                RET
1B4B            ;
1B4B                    END

ASSTR2:E4B8  CR    :000D  EBUF  :013E  EOF   :1B41  EOF2  :1B43  ERRADR:09B7
ETX   :0003  GETC  :D6BE  INCHRS:1B12  INPERR:1756  LINPUT:1B00  LOOP  :1B1C
MAXCH :007D  NUL   :0000  OK1   :1B34  OK2   :1B46
```

STRAGIC AUTOLADER

```
1       REM H.STROBEL/NEUSELSBRUNN 51/8500 NUERNBERG 50
2       REM CODENAME: >STRAGIC
3       REM DATUM: 24.10.1984
4       REM PROGRAM: AUTOLOADER GAME
5       POKE #5F,#84:REM TASTATUR AUS
10      MODE 0:PRINT CHR$(12):COLORT 0 6 0 14:POKE #74,0:
        POKE #75,#FF:ENVELOPE 0 15
11      DIM STRAGIC(2):STRAGIC(0)=#CDBBD632:
        STRAGIC(1)=#E602FEFF:STRING(2)=#CA00C0C9:
        STRAGICX=VARPTR(STRAGIC(0)):REM MLP ZUR TASTENABFRAGE
15      C$=" DAInamic Game: ":CY=22
20      FOR I=1 TO 20:READ A$:PRINT CHR$(137);C$;A$:NEXT
25      POKE #B4F0,#D1:POKE #B46A,#D3
30      FOR I=0 TO 59:PRINT CHR$(1);:NEXT:PRINT
40      PRINT CHR$(137);" STATUS: ","","WAEHLEN=[";CHR$(94);
        "≈";CHR$(140);"] START= [SPACE]"
45      SOUND 0 0 15 2 FREQ(3000):WAIT TIME 25:SOUND OFF
50      CURSOR R0,CY:POKE #2B2,0:REM TASTATURBUFFER 1 BYTE
        LEEREN
60      GOSUB 9999:G=PUSH:IF G=0 GOTO 60:REM TASTATURAUFRUF
70      IF G=17 AND CY>2 THEN CY=CY-1:IF CY=2 THEN CY=22
80      IF G=16 AND CY<23 THEN CY=CY+1:IF CY=23 THEN CY=3
90      IF G=32 GOTO 100
99      GOTO 50
100     RESTORE:CCY=22-CY:FOR I=0 TO CCY:READ A$:NEXT
105     IF LEFT$(A$,1)<>"." GOTO 110:REM ABFRAGE OB PROGRAMM
        AN DIESER STELLE VORHANDEN IST
106     SOUND 0 0 15 0 FREQ(1000):WAIT TIME 25:SOUND OFF:GOTO 50
110     GOSUB 9998:POKE CB,#DE:POKE CB-134,#D6:
        AN$="SUCHE+LADE+STARTE PROGRAMM   ":GOSUB 9997:GOTO 900
900     CALLM #F003,ERR:IF CCY=0 THEN LOAD :REM
        DCR-FEHLERMELDUNG AUS
910     FOR H=1 TO CCY:CALLM #F000:REM SKIP1
920     NEXT:LOAD :END
1000    DATA "Dainapac.....................CONCAT.PROT."
1001    DATA "Gomuku.......................BAS.PROT...."
1002    DATA "Vier in einer Reihe..........BAS.NONPROT."
1003    DATA "3D-Labyrinth.................BAS.NONPROT."
1004    DATA "Bridge Building..............BAS.NONPROT."
1005    DATA "Surround.....................BAS.PROT...."
1006    DATA "Zickzack.....................BAS.PROT...."
1007    DATA "............................................"
1008    DATA "............................................"
1009    DATA "...........o.ae.............................."
1010    DATA "............................................"
1011    DATA "............................................"
1012    DATA "............................................"
1013    DATA "............................................"
1014    DATA "............................................"
1015    DATA "............................................"
1016    DATA "............................................"
1017    DATA "............................................"
1018    DATA "............................................"
1019    DATA "............................................"
9997    CURSOR 24,1:PRINT AN$:CURSOR 0,1:RETURN
9998    CB=PEEK(#79)*256+PEEK(#78)-1:RETURN:REM SETZT DIE
        AUSGEWAEHLTE ZEILE AUF EINE ANDERE FARBE
9999    POKE #5F,#C4:CALLM STRAGICX:POKE #5F,#84:
        PUSH=PEEK(#2E6):RETURN:REM TASTATUR
        EIN/TASTATURABFRAGE/TASTATURWERT IN 'PUSH'/TASTATUR AUS
10000   REM DAS MLLP LEGT DIE TASTATURDATEN AN DIE STELLE #2E6
        (NORMALERWEISE IST DAS EINE STELLE DES
        CASSETTENLEADERS)
10001   REM DER CASSETTENREKORDER SOLLTE DAHER NICHT MIT DIESEM
        PROGRAMM BEDIENT WERDEN !
```

DISK DUMP und EDIT

Autor: Uwe Wienkop

Das nachstehende Programm ermöglicht es 2K Byte von der Diskette zu lesen und diesen Bereich zu editieren. Nachdem alle Änderungen vorgenommen worden sind können diese dann wieder abgespeichert werden.

Diese 2K Byte sind normalerweise je nach Sektorgröße 2, 8 oder 16 Sektoren, die mit RSEC oder bei der neueren Version mit RDSK gelesen werden können. Diese zwei KB werden ab Adresse #A000 abgelegt. Da es nicht möglich ist einen derartig großen Ausschnitt auf einmal auf dem Bildschirm darzustellen, wurde hier folgendes Verfahren angewandt: Mit dem Maschinenprogramm ab #B200 wird ein 1/16 also 128 Bytes dieses Bereiches nach #B000 kopiert und nur dieser Bereich wird dargestellt, bzw. auf dieser Kopie wird gearbeitet.

Es ist somit auch leicht möglich, ohne weiteren Diskettenzugriff, den Inhalt zu restaurieren, wenn man bemerkt, die Änderungen waren doch nicht so gut. Alle Änderungen werden sowieso nur auf ausdrücklichen Wunsch abgespeichert!

Das Programm kann mit einigen Änderungen jedoch auch als "RAM-Monitor" verwendet werden, z.B. um sich den Inhalt des Speichers anzusehen, bzw. zu verändern.

Nach dem Starten des Programms wird man nach einer kurzen Weile gefragt, welcher Bereich editiert werden soll. Bei meinem Diskettensystem (INDATA DOS 3.0) gebe ich dies in Blöcken zu je 2 K Byte an und zwar gleich mit einem Offset von 4 K Byte, da bei mir die Spur 0 fast uninteressant ist. Die Directory ist bei mir auf der Spur 1! Als Eingabewert wird hier also die Blocknummer erfragt. Dies ist bei kleineren Blöcken ggf. entsprechend abzuändern.
Daraufhin wird der erste 128 Byte große Ausschnitt dieses 2K Bereichs dargestellt und mit 'V' für Vorwärts und 'Z' für Zurück kann man in diesem Bereich blättern. Werden die 2K Grenzen überschritten, so wird automatisch der nachfolgende oder vorhergehende Block geladen

Mit 'A' für Abspeichern können die Änderungen wieder zurückgeschrieben werden.

Durch 'S' springt man zu einem angegebenen Block. Dies wird benötigt, wenn zwischen zwei interessierenden Blöcken viel Platz liegt!

Durch 'E' ruft man einen Mini Editor auf. Der Disketteninhalt wurde ja schon bisher sowohl in ASCII als auch in HEX dargestellt. Nun landet man zunächst in der obersten ASCII Zeile. Mit den Cursortasten kann der Cursor beliebig auf diesem Ausschnitt herumbewegt werden. Wird eine andere Taste gedrückt, so wird das entsprechende Zeichen an dieser Stelle in ASCII und HEX dargestellt und mit der Tab-Taste kann der Editor verlassen werden.
Um außerdem noch HEX-Zahlen darstellen zu können kann man mit Shift-Cursor oben/ unten zwischen der ASCII und der HEX Spalte hin und herschalten. Im HEX Modus ist nur noch die Eingabe von HEX Zahlen erlaubt. Jedoch wird nach Eingabe einer HEX Zahl gleichzeitig in der oberen Reihe das entsprechende ASCII Zeichen dargestellt - falls möglich.

```
2120  IF A=21 THEN INC=1:COLORT 8 13 8 1:POKE #BE5C,64+PEEK(#7D):G
      OTO 2010
2130  IF A=8 GOTO 2080
2140  IF INC=1 GOTO 2170
2150  PRINT " ";CHR$(A);:POKE #B000+16*LINE+COL,A
2160  CURSOR CURX-2,CURY-1:PRINT RIGHT$("0"+HEX$(A),2);:GOTO 2100
2170  IF A<#30 OR (A>#39 AND A<#41) OR A>#46 GOTO 2030:A=A-#30:IF
      A>9 THEN A=A-7
2180  X=16*A
2190  A=GETC
2200  A=GETC:IF A=0 GOTO 2200
2210  IF A<#30 OR (A>#39 AND A<#41) OR A>#46 GOTO 2200:A=A-#30:IF
      A>9 THEN A=A-7
2220  X=X+A
2230  PRINT RIGHT$("0"+HEX$(X),2);:CURSOR CURX-1,CURY+1
2240  POKE #B000+16*LINE+COL,X:X=X IAND 127:IF X<32 THEN X=95
2250  PRINT CHR$(X);:GOTO 2100
5000  DATA #C5,#23,#23,#23,#7E,#21,#80,#00,#CD,#8F,#DE,#11,#00,#A0
      ,#19,#EB
5010  DATA #21,#81,#00,#19,#01,#00,#B0,#CD,#4F,#DE,#21
5020  DATA #C7,#BD,#11,#00,#B0,#06,#08,#0E,#10,#D5,#2B,#2B,#1A,#E6
      ,#7F,#FE
5030  DATA #20,#D2,#31,#B2,#3E,#5F,#77,#2B,#2B,#2B,#36,#FF,#2B,#13
      ,#0D,#C2
5040  DATA #25,#B2,#11,#DA,#FF,#19,#D1,#0E,#10,#1A,#D5,#CD,#63,#B2
      ,#73,#2B
5050  DATA #2B,#77,#D1,#2B,#2B,#2B,#2B,#13,#0D,#C2,#44,#B2,#D5,#11
      ,#DA,#FF
5060  DATA #19,#D1,#05,#C2,#22,#B2,#C1,#C9,#F5,#1F,#1F,#1F,#1F,#CD
      ,#6D,#B2
5070  DATA #5F,#F1,#E6,#0F,#C6,#30,#FE,#3A,#F8,#C6,#07,#C9
5080  DATA #C5,#23,#23,#23,#7E,#21,#80,#00,#CD,#8F,#DE,#11
5090  DATA #00,#A0,#19,#44,#4D,#11,#00,#B0,#21,#80,#B0,#CD
5100  DATA #4F,#DE,#C1,#C9,#FFFF
```

```
TOP     EQU   :BE4F-136
        /
        ORG   :B200
        PUSH  B
        INX   H
        INX   H
        INX   H
        MOV   A,M
        LXI   H,128
        CALL  :DE8F
        LXI   D,:A000
        DAD   D
        XCHG
        LXI   H,129
        DAD   D
        LXI   B,:B000
        CALL  :DE4F
        LXI   H,TOP
        LXI   D,:B000
        MVI   B,8
LOOP    MVI   C,16
        PUSH  D
LOOP2   DCX   H
        DCX   H
        LDAX  D
        ANI   :7F
        CPI   32
        JNC   $+5
        MVI   A,95
        MOV   M,A
        DCX   H
        DCX   H
        DCX   H
        MVI   M,:FF
        DCX   H
        INX   D
        DCR   C
        JNZ   LOOP2
        LXI   D,-38
        DAD   D
```

```
21000 REM ---  Bandende finden  -----
21005 REM von DAInamic Belgien  &  E.Zahner
21010 CLEAR 1000:CURS=20:LF$=SPC(60):FF$=CHR$(12):REM CURS=20
21020 MODE 0:POKE #131,1:AD$="":PRINT FF$
21030 FOR I=1 TO 18:READ B:AD$=AD$+CHR$(B):NEXT
21040 AD=PEEK(VARPTR(AD$)) IOR (PEEK(VARPTR(AD$)+1) SHL 8)+1
21050 CURSOR 0,22:PRINT LF$;:CURSOR 0,22:CALLM AD:CURS=CURS-2
21060 IF CURS=0 THEN CURS=2
21070 CURSOR 1,CURS
21080 POKE #40,#30:REM Cassettenrecorder abschalten
21090 SCL$=""
21100 FOR A=#BF61-2 TO #BF61-40 STEP -2
21110 SCL$=SCL$+CHR$(PEEK(A))
21120 NEXT
21130 PRINT SCL$:PRINT LEN(SCL$):COMPARE$=".FINITO        "
21140 IF LEFT$(SCL$,14)=COMPARE$ THEN PRINT "finito":
      GOTO 21180
21160 GOTO 21040
21170 DATA #F5,#C5,#D5,#E5,#01,#FF,#00,#21,#00,#00,#CD,#CE,
      #02,#E1,#D1,#C1,#F1,#C9
21180 PRINT "HIER GEHT ES WEITER"
```

Das obenstehende Programm dient dazu ein logisch gekennzeichnetes Bandende zu finden.
Als letzte Datei soll sich auf dem Band ein Eintrag mit den Namen ".FINITO   " befinden. Es wird nun solange gesucht, bis dieser Programmname gefunden wurde.

Das Programm benutzt dabei eine kleine Maschinenroutine, die sich an einer beliebigen Stelle im Speicher befinden kann (In diesem Programm wird diese Routine in den String AD$ geschrieben). Dieses kleine Maschienprogramm führt nun den ersten Teil der CHECK-Routine durch - es wird einfach Read Open aufgerufen. Dadurch werden dann auch die Datei-Namen auf den Bildschirm geschrieben. Nun wird aber der Check nicht mehr normal durchgeführt, sondern es wird durch den POKE #40,#30 der Cassettenrecorder abgeschaltet.
Anschließend wird der Filename aus dem Bildschirm gelesen und auf das obige ".FINITO   " verglichen. Falls es nicht Finito gewesen sein sollte, so wird das Verfahren wiederholt, ansonsten wird das Programm bei Zeile 21180 weiterbearbeitet.

SAVEA-Trick

Wer einmal ein Programm geschrieben hat, bei dem Daten unterschiedlicher Menge durch SAVEA abgespeichert werden, wird das Problem kennen : wenn die abzuspeichernde Datenmenge sehr groß sein kann, muß ein grosses Array (dann meistens 2 Dimensionen) dimensioniert werden. Wenn dann aber dieses Array manchmal nur gering ausgenutzt wird, muß man etliche Kilobyte umsonst abspeichern, so daß unnötig lange Ladezeiten auftreten bzw. Magnetband „verheizt" wird. Bei dem „Universal ORG" hatte ich ebenfalls das Problem (kleine Maschinenprogramme - kurzer Datensatz, lange MP's - grosser Datensatz ⇒ Datensatzlängen von 100 bis 6000 Byte). Das Problem wurde dermassen gelöst, daß das Array vor dem Abspeichern auf die passende Grösse, auf 128 Byte genau, verkleinert wurde. Wenn man ein Array „betriebssystemrichtig" verkleinern kann, so braucht man nur noch maximal zu dimensionieren und „passend" abspeichern, so daß man die wahrscheinlich benötigte Menge nicht vorher angeben muß.
Vorgehen bei einem 2-dimensionalen Feld :
Am Programmanfang :

```
SDIM=MAXBYTES/128
DIM ARR(31,SDIM)
STARR=VARPTR(ARR(0,0))
```

Vor dem SAVEA (hier als GOSUB) :

```
NDIM=(BENUTZTEBYTES-1)/128
IF NDIM>=SDIM THEN RETURN
P=(NDIM+1)*128+3:
POKE STARR-5,P SHR 8:POKE STARR-4,P IAND 255:
POKE STARR-1,NDIM
P2=STARR-3+P:P=(SDIM-NDIM)*128-2
POKE P2,P SHR 8 IOR #80:POKE P2+1,P IAND 255
RETURN
```

Dabei sind MAXBYTES die im Maximalfall, BENUTZTEBYTES die tatsächlich benötigten Bytes. Statt MAXBYTES kann SDIM natürlich auch direkt angegeben werden. Will man BENUTZTEBYTES berechnen, so kann man von BenutzteFeldelemente * 4 bei INT/FPT-Arrays oder von der maximal benötigten Dimensionierung NDIM ausgehen. Die Teilung auf 128 Bytes wurde gewählt, da sie zum einen sehr klein ist und zum anderen mit SDIM=255 das größtmögliche Array (256*128=32 kByte) zuläßt ! Ist für den Anwendungsfall etwas anderes (N Bytes) trotzdem günstiger, so ist die 31 durch N/4-1 bei INT/FPT-Arrays zu ersetzen. N muß also immer durch 4 teilbar sein !
Bei der Benutzung des ARRays sollte die zweite Dimension, die am Schluß NDIM ausmacht, möglichst niedrig gehalten werden, um dann das Array optimal auszunutzen.
Bei eindimensionalen und String-Arrays tritt das Problem seltener auf, da der „überschuß" gering ist. Bei einer Dimension ist

STARR-5	durch	STARR-4	zu ersetzen
STARR-4		STARR-3	
STARR-3		STARR-2	
P=..+3		P=..+2	

Bei Stringfeldern ist nur die Dimensionsformel zu ersetzen. Statt N/4-1 nun N/2-1 (128 ⇒ 31/63).

Der letzte Vorteil dieser Methode ist, daß das Array zwar kleiner, aber noch voll „funktionsfähig" ist (alle Feldelemente zugreif- und veränderbar).

Heinrich Tegethoff

Wieder einmal P A S A C L

Liebe Mitglieder

Ich habe wieder einmal das DAI Tiny-Pascal aus der Schublade genommen und habe einige kleine Pascalprogramme geschrieben.
Obwohl es nur ein Tiny-Pascal ist,sind seine Möglichkeiten durchaus nicht so beschränkt,wie ich erst angenommen habe.Ich habe diese kleinen Programme geschrieben,um vielleicht einige unter euch zu motivieren,in nächster Zeit selbst einige 'Prográmmlein' zu veröffentlichen,zumal die Sprache Pascal um einiges elegantere Problemlösungen zulässt,als beispielsweise BASIC.So sind vielleicht auch bei dem 'Nimmspiel' schönere Lösungen zu finden??? (Graphik,etc) Ich wäre sehr daran interessiert.

Das wars also.Viele Grüsse Stephan

Meine Adresse:
STEPHAN FISCHLI
SAEGENSTR. 104
CH-7000 CHUR

P.S Erst Pascal-System laden,dann Runtime Modules

```
! Das NIMM-SPIEL [IN PACAL]          !
! ~~~~~~~~~~~~~~~~~~~~~~~~~          !
! PROGRAM NIMM-SPIEL by S.Fischli    !
! SAEGENSTR. 104 CH-7000 CHUR        !
! C/ FEB. '85                        !

VAR I,MZ,INP,FLAG,ANZAHL,BE,A : INTEGER;
 PROC EINGABE1;
 VAR BE: INTEGER;
   BEGIN
   FLAG:=0;
     WRITE (12,13,13,'Das  N I M M  -  S P I E L',13);
     WRITE (          '============================');
     WRITE (13,13,
             'Mit wieviel Staebchen willst Du spielen?');
     REPEAT
     READ (ANZAHL%);
     UNTIL ANZAHL>1;
      WRITE (13,'Willst Du beginnen [J/N => RETURN]',13);
      READ (BE);
       CASE BE OF
            'J' : FLAG:= 1;
            'N' : FLAG:= 0
       ELSE WRITE (13,'Schummler, also beginne ich!',13)
     END;
 END;

! HAUPTPROGRAMM !

BEGIN
  BE:= 0;
  REPEAT
  EINGABE1;
    REPEAT
      IF FLAG=0
         THEN BEGIN
              MZ:=(ANZAHL+3) MOD 4;
              IF MZ=0 THEN MZ:=1;
              WRITE (13,'Ich nehme ',MZ%,' Staebchen');
              ANZAHL:=ANZAHL-MZ;
              WRITE ('=>> Es bleiben ',ANZAHL%,
                     ' Staebchen',13);
              FLAG:=1;
              END
         ELSE BEGIN
              REPEAT    ! Kontrollschleife 1 !
                REPEAT  ! Kontrollschleife 2 !
                 WRITE  (13,'Wieviele Staebchen',
                            'nimmst Du ? [<=3 Stk.',13);
                 READ (I%)
                UNTIL I<4;
              UNTIL    I>0;
              ANZAHL:=ANZAHL-I;
              WRITE (' =>> Es bleiben ',ANZAHL%,
                     ' Staebchen',13);
            FLAG:=0;
            END;
    UNTIL ANZAHL<=1;
  IF FLAG=0
     THEN WRITE (13,13,13,'Du hast gewonnen,',
                          ' reines Glueck !!!',13)
     ELSE WRITE (13,'HA, ich habe als Koenner gewonnen !!!'
                 ,13);

  WRITE (13,13,'Noch ein Spiel [J/N] ?');
  READ (A);
  CASE A OF
```

```
        'J' : BE:=1;
        'N' : BE:=0
    ELSE WRITE (13,13,'Falsche Taste, dann also ...')
    END;
    UNTIL BE=0;

WRITE (13,    ' G O O D    B Y E');
WRITE (13,13,' Bis zum naechsten Mal');
END.
```

```
! PROGRAM FAKULTAET (BIS max. 7) !

VAR  N,H,P,A : INTEGER;

FUNC FAK(N);
VAR I,F : INTEGER;
BEGIN
 F:=1;
 I:=1;
 REPEAT
    F:=F*I;
    I:=I+1
 UNTIL I=N+1;
FAK:=F;
END;

! Hauptprogramm !
BEGIN
 REPEAT
   WRITE (12,13,'Fakultaet von [<8]:');
   READ  (N%);
   WRITE (13,' ist ',FAK(N)%,13);
   WRITE (13,13,'Noch eine Berechnung? [J/N]',13,13);
   READ (A);
     CASE A OF
          'J' : P:=1;
          'N' : BEGIN
                  WRITE (13,'OK, GOOD BYE');
                  P:=0
                END
          ELSE  BEGIN
                  WRITE (13,'Falsche Taste');
                  P:=1;
                  READ (A)    ! Warte und druecke Taste !
                END
     END
  UNTIL P=0
END.
```

```
0002                    *
0003                    * WILFRIED RUESSE, 6239 KRIFTEL. (C) FEB.1985
0004                    * ============================================
c                                       =====
0005                    * *$FILENAME -) laed und startet ein BASICpro
c                                       gramm
0006                    * ! Im Basicprogramm muss ein POKE #1935,0 st
c                                       ehen !
0007                    * *#FILENAME -) laed und startet ein  ML-Prog
c                                       ramm,
0008                    *  das mit eienr Startadresse abgesichert wur
c                                       de.
0009                    * *DSAVE FILENAME.BIN ANFANG-ADR. END-ADR. ST
c                                       ART-ADR.
0010                    *  Wird keine START-ADR. angegeben, muss  FFF
c                                       F
0011                    *  anstelle der START-ADR. eingegeben werden.
0012                    * --------------------------------------------
c                                       -------
0013                    *
0014      **19CC        EDOS    EQU   :19CC     ;END OF $MSTRDOS
0015      **C14D        POP     EQU   :C14D     ;POP & RET
0016      **029B        HEAP    EQU   :29B      ;START OF HEAP
0017      **0297        TBLNK   EQU   :297      ;CONT-ADR.
0018      **DEB5        NEW     EQU   :DEB5     ;BASIC NEW
0019      **0477        RESETD  EQU   :477      ;RESET DISK
0020      **0992        FILNAM  EQU   :992      ;FILENAME-BUFFER
0021      **0EFD        READ    EQU   :EFD      ;READ FILENAME
0022                    ******************************************
0023      **19CC                ORG   EDOS
0025 19CC               ******** NEW TABLE **********************
0026 19CC 0123          COMTB   DATA  1,"#"
0027 19CE EB19                  DBL   BIN
0028 19D0 0124                  DATA  1,"$"
0029 19D2 D719                  DBL   BAS
0030 19D4 00                    DT    0
0031 19D5 3718          CONT    DBL   :1837
0032 19D7               ******** LOAD BAS ***********************
0033 19D7 0B            BAS     DCX   B
0034 19D8 CDFD0E                CALL  READ
0035 19DB 3E08                  MVI   A,8
0036 19DD 329209        S       STA   FILNAM    ;LENGTH OF FN$
0037 19E0 CD7704                CALL  RESETD
0038 19E3 210003                LXI   H,:300    ;CONT ADR.
0039 19E6 E5                    PUSH  H         ;ON STACK
0040 19E7 21DE19                LXI   H,S+1     ;HL: ADR OF EBUF
0041 19EA C9                    RET
0042 19EB               ******** LOAD BIN ***********************
0043 19EB CDFD0E        BIN     CALL  READ
0044 19EE 3E08                  MVI   A,8
0045 19F0 329109                STA   FILNAM-1
0046 19F3 010031                LXI   B,:3100
0047 19F6 C5                    PUSH  B
0048 19F7 3E01                  MVI   A,1
0049 19F9 219109                LXI   H,FILNAM-1
0050 19FC CDCE02                CALL  :2CE
0051 19FF 1100F9                LXI   D,:F900
0052 1A02 D5                    PUSH  D
0053 1A03 213E01                LXI   H,:13E
0054 1A06 114101                LXI   D,:141
0055 1A09 CDD102                CALL  :2D1
0056 1A0C D1                    POP   D
0057 1A0D 2A3E01                LHLD  :13E
0058 1A10 010000                LXI   B,0000
0059 1A13 37                    STC
0060 1A14 CDD102                CALL  :2D1
0061 1A17 C1                    POP   B
0062 1A18 3E00                  MVI   A,0
0063 1A1A 323519                STA   :1935
0064 1A1D C3D402                JMP   :2D4      ;GO TO START-AD
0065 1A20 00            END     DT    00        ;OR TO BASIC
0066 1A21               ******** INITIALISIERUNG **************
0067 1A21 F5                    PUSH  PSW
0068 1A22 C5                    PUSH  B
0069 1A23 D5                    PUSH  D
0070 1A24 E5                    PUSH  H
0071 1A25 2A9702                LHLD  TBLNK
0072 1A28 22D519                SHLD  CONT
0073 1A2B 21CC19                LXI   H,COMTB
0074 1A2E 229702                SHLD  TBLNK
0075 1A31 21211A                LXI   H,END+1
0076 1A34 229B02                SHLD  :29B      ;NEW END OF DOS
0077 1A37 CDB5DE                CALL  NEW
0078 1A3A C34DC1                JMP   POP       ;==) POP & RET
0079 1A3D               ****************************************
0080 1A3D               * INITIALISIERUNG :
0081 1A3D               * I.    ASSEMBLIEREN
0082 1A3D               * II.  *DSAVE $USER.BIN 19CC 1A3D 1A21
0083 1A3D               * III.  WIRD BEI RESET INITIALISIERT
0084 1A3D               ****************************************


**************************
* S Y M B O L   T A B L E *
**************************

BAS     19D7    BIN     19EB    COMTB   19CC    CONT    19D5
EDOS    19CC    END     1A20    FILNAM  0992    HEAP    029B
NEW     DEB5    POP     C14D    READ    0EFD    RESETD  0477
S       19DD    TBLNK   0297
```

Schnelle Sortierroutinen

Autor: Uwe Wienkop

Nachdem in den letzten Zeitungsausgaben immer wieder einmal Routinen in Tinypascal aufgetaucht sind, die jedesmal auf die Eleganz und Strukturiertheit von PASCAL hinwiesen, ist es nun an der Zeit auch mal etwas positives zu BASIC, genauer DBASIC zu sagen. Hiermit ist es, wie in PASCAL, möglich Prozeduren mit lokalen Variablen aufzubauen. Somit benötigt man zum Programmieren von rekursiven Prozeduren keine Tricks mehr. Da bei DBASIC weiterhin alle Variablentypen (zusätzlich Arrays mit bel. Indices (auch größer 255)) existieren bildet dieses Programm eine echte Alternative zu PASCAL, besonders, wenn man die vielen zusätzlichen Befehle betrachtet.

Hier sollen nun zwei unter DBASIC erstellte Sortierprogramme vorgestellt werden: SHELLSORT und QUICKSORT.
Beide Routinen sortieren jeweils den Inhalt eines Arrays in aufsteigender Reihenfolge, jedoch zeigt sich beim Vergleich, daß QUICKSORT immer wesentlich besser abschneidet. (Man muß jedoch auch zur Rettung von Shellsort sagen, es sortiert immer noch schneller als ein "Allerweltssort"). Quicksort arbeitet allerdings stark rekursiv, d.h. das Programm ruft sich selbst auf. Diese Rekursivität bereitet im normalen DAI BASIC jedoch Schwierigkeiten, da dann alle lokalen Variablen in einem künstlich definierten Stack abgelegt werden müssen und der Programmierer muß sich auch um die Stackverwaltung kümmern. Viel leichter ist dies jedoch in DBASIC, welches entsprechende Funktionen wie lokale Variablen und rekursive Aufrufe bereits kennt.

Geschwindigkeitsvergleich: Int-Array (N-Elemente)

	N=50	N=100	N=200	N=500	N=1000
Quicksort	5.44	11.36	24.08	70.82	150.96
Shellsort	10.40	26.18	73.20	204.60	529.00

```
10      N=1000:DIM A(N)
20      FOR I=0 TO N:A(I)=RND(100):NEXT
30      PRINT "Beginn":POKE #1BE,255:POKE #1BF,255
40      Sortierprogrammaufruf
45      T!=(#FFFF-PEEK(#1BE)-256*PEEK(#1BF)/50.0
50      PRINT "Benoetigte Zeit:";T!:CALLM#D6DA
60      FOR I=0 TO N:PRINT A(I),:NEXT
100     END

1000    REM ------------  QUICKSORT  ----------------
1005    REM -- Aufruf:  QUICKSORT 0,N
1010    PROCEDURE QUICKSORT L,R
1020    LOCAL I,J,X,W
1030    I=L:J=R
1040    X=A((L+R)/2)
1050    REPEAT WHILE A(I)<X DO I=I+1:WEND
1060    WHILE X<A(J) DO J=J-1:WEND
1070    IF I<=J THEN W=A(I):A(I)=A(J):A(J)=W
1080    I=I+1:J=J-1:END IF
1090    UNTIL I>J
1100    IF L<J THEN QUICKSORT L,J:END IF
1110    IF I<R THEN QUICKSORT I,R:END IF
1120    END PROC

3000    REM ------------  SHELLSORT  -------------
3002    REM -- Aufruf:  SHELLSORT A()
3005    PROCEDURE SHELLSORT  ARR A
3010    LOCAL L,I,DNE,DM
3015    DM=DIM(A,1):L=DM
3020    WHILE L>1 DO L=L/2
3030    REPEAT DNE=1
3035    FOR I=0 TO DM-L
3040    IF A(I+L)<A(I) THEN H=A(I):A(I)=A(I+L):A(I+L)=H:DNE=0:END IF
3050    NEXT:UNTIL DNE=1:WEND:END PROC
```

Nachfolgend ein Farb-Screencopy-Programm für den Epson JX-80. Bei Mode 5+6 wird ein Ausdruck erstellt der eine DIN A4 Seite, fast vollständig ausfüllt. Es entsteht eine 1:1 Bildschirm Copy mit bis zu 16 Farben.
Mode 1 bis 4 sind auch möglich , aber die Ausdrucke sind dementsprechend kleiner.

Bei Mode 7/8 haut das Ganze nicht mehr hin, da man mit dem Befehl RST 5 DATA 27 nicht mehr so arbeiten kann. Wie man dort verfahren muß, weiß ich leider nicht, da meine bescheidenen Assemblerkenntnisse dazu nicht ausreichen . Über Informationen, des Bildschirmaufbau's bei Mode 7/8, wäre ich dankbar.

Heiner Wahnelt
Putbusser Str.7
1000 Berlin 65

NP 90,1

```
04C8  3E13    MVI A:13
04CA  CD2D05  CALL :052D
04CD  E1      POP H
04CE  D1      POP D
04CF  C1      POP B
04D0  F1      POP PSW
04D1  C9      RET
04D2  3E11    MVI A:11
04D4  CD2D05  CALL :052D
04D7  3E1B    MVI A:1B
04D9  CD2D05  CALL :052D
04DC  3E4F    MVI A:4F
04DE  CD2D05  CALL :052D
04E1  3E0D    MVI A:0D
04E3  CD2D05  CALL :052D
04E6  3E1B    MVI A:1B
04E8  CD2D05  CALL :052D
04EB  3E72    MVI A:72
04ED  CD2D05  CALL :052D
04F0  3AFB02  LDA  :02FB
04F3  CD2D05  CALL :052D
04F6  FE04    CPI  :04
04F8  C20005  JNZ  :0500
04FB  3E0A    MVI A:0A
04FD  CD2D05  CALL :052D
0500  3E1B    MVI A:1B
0502  CD2D05  CALL :052D
0505  3E41    MVI A:41
0507  CD2D05  CALL :052D
050A  3E08    MVI A:08
050C  CD2D05  CALL :052D
050F  3E1B    MVI A:1B
0511  CD2D05  CALL :052D
0514  3E2A    MVI A:2A
0516  CD2D05  CALL :052D
0519  3E05    MVI A:05
051B  CD2D05  CALL :052D
051E  E5      PUSH H
051F  2AF902  LHLD :02F9
0522  29      DAD H
0523  7D      MOV A,L
0524  CD2D05  CALL :052D
0527  7C      MOV A,H
0528  CD2D05  CALL :052D
052B  E1      POP H
052C  C9      RET
052D  F5      PUSH PSW
052E  3E40    MVI A:40
0530  3D      DCR A
0531  C23005  JNZ  :0530
0534  F3      DI sp
0535  3A00FD  LDA  :FD00
0538  E608    ANI  :08
053A  CA3505  JZ   :0535
053D  3A03FF  LDA  :FF03
0540  E610    ANI  :10
0542  CA3D05  JZ   :053D
0545  F1      POP PSW
0546  3206FF  STA  :FF06
0549  FB      EI fr
054A  C9      RET
```

```
0300  F5 C5 D5 E5 3E 04 32 FB 02 21 00 00 01 00 00 EF  5F6
0310  27 7B FE FF DA 19 03 3E FF 32 F9 02 EB CD D2 04  80A
0320  3A F9 02 4F E5 21 00 00 E3 06 04 C5 0D EF 27 0C  568
0330  C1 F5 3A FB 02 FE 04 CA 1F 04 FE 02 CA C0 03 FE  867
0340  01 CA 61 03 F1 FE 00 C2 50 03 11 03 03 C3 7B 04  58C
0350  FE 00 C2 5B 03 11 01 02 C3 7B 04 11 00 00 C3 7B  4CB
0360  04 F1 FE 02 C2 6D 03 11 03 03 C3 7B 04 FE 03 C2  643
0370  7B 03 11 03 03 C3 7B 04 FE 04 C2 83 03 11 03 03  435
0380  C3 7B 04 FE 06 C2 8E 03 11 01 02 C3 7B 04 FE 07  5F4
0390  C2 99 03 11 01 02 C3 7B 04 FE 09 C2 A4 03 11 01  536
03A0  02 C3 7B 04 FE 0A C2 AF 03 11 01 02 C3 7B 04 FE  614
03B0  0B C2 BA 03 11 01 02 C3 7B 04 11 00 00 C3 7B 04  433
03C0  F1 FE 01 C2 CC 03 11 03 03 C3 7B 04 FE 04 C2 D7  775
03D0  03 11 03 03 C3 7B 04 FE 05 C2 E2 03 11 03 03 C3  4E0
03E0  7B 04 FE 06 C2 ED 03 11 01 02 C3 7B 04 FE 07 C2  652
03F0  F8 03 11 01 02 C3 7B 04 FE 09 C2 03 04 11 01 02  435
0400  C3 7B 04 FE 0C C2 0E 04 11 01 02 C3 7B 04 FE 0D  581
0410  C2 19 04 11 01 02 C3 7B 04 11 00 00 C3 7B 04 F1  479
0420  FE 03 C2 2B 04 11 03 03 C3 7B 04 FE 04 C2 36 04  549
0430  11 03 03 C3 7B 04 FE 05 C2 41 04 11 03 03 C3 7B  4B8
0440  04 FE 06 C2 4C 04 11 03 03 C3 7B 04 FE 07 C2 57  591
0450  04 11 01 02 C3 7B 04 FE 0A C2 62 04 11 01 02 C3  461
0460  7B 04 FE 0D C2 6D 04 11 01 02 C3 7B 04 FE 0E C2  5E1
0470  7B 04 11 03 03 C3 7B 04 11 00 00 E3 29 29 19 E3  417
0480  2B 05 C2 2B 03 E3 7C CD 2D 05 7D CD 2D 05 E1 11  5EC
0490  04 00 19 0D C2 24 03 3A FB 02 FE 00 CA A8 04 37  4F5
04A0  3F 1F 32 FB 02 C3 1D 03 3E 04 32 FB 02 11 FC FF  5ED
04B0  19 DA 1D 03 3E 0A CD 2D 05 3E 0D CD 2D 05 3E 1B  3FD
04C0  CD 2D 05 3E 4E CD 2D 05 3E 13 CD 2D 05 E1 D1 C1  64D
04D0  F1 C9 3E 11 CD 2D 05 3E 1B CD 2D 05 3E 4F CD 2D  5E7
04E0  05 3E 0D CD 2D 05 3E 1B CD 2D 05 3E 72 CD 2D 05  456
04F0  3A FB 02 CD 2D 05 FE 04 C2 00 05 3E 0A CD 2D 05  546
0500  3E 1B CD 2D 05 3E 41 CD 2D 05 3E 08 CD 2D 05 3E  459
0510  1B CD 2D 05 3E 2A CD 2D 05 3E 05 CD 2D 05 E5 2A  4D2
0520  F9 02 29 7D CD 2D 05 7C CD 2D 05 E1 C9 F5 3E 40  738
0530  3D C2 30 05 F3 3A 00 FD E6 08 CA 35 05 3A 03 FF  68C
0540  E6 10 CA 3D 05 F1 32 06 FF FB C9 -- -- -- -- --  5EE
```

VC-1541

Disk-Dump-Edit (von A. Koldehoff)

Das nachstehende Programm ermöglicht es einen beliebigen Block von der Diskette zu lesen und diese 256 Bytes zu editieren. Nachdem alle Änderungen vorgenommen worden sind, können diese dann wieder abgespeichert werden. Weiterhin ist es mit diesem Programm möglich, in einer beliebigen Datei auf der Diskette zu blättern, um z.B. einen bestimmten Bereich zu suchen bzw. zu ändern. Ferner können in der Directory beliebige Änderungen vorgenommen werden, um z.B. den Diskettennamen, die ID, Dateinamen zu ändern oder um Files gegen Überschreiben zu schützen u.s.w.

Nach dem Start des Programms wird man nach einer Track- und einer Sektornummer gefragt. Wie im Handbuch der Floppy beschrieben kommen dafür folgende Werte in Frage :

Track	Sektor	Anzahl der Sektoren
1 bis 17	0 bis 20	21
18 bis 24	0 bis 18	19
25 bis 30	0 bis 17	18
31 bis 35	0 bis 16	17

Gibt man nun eine Tr/Sk-Nummer ein, so wird dieser Block nach #A000 gelesen. Anschließend wird dieser Block nach #B000 kopiert, wo dann evtl. Änderungen vorgenommen werden können. Überhaupt wird nur der Bereich ab #B000 angezeigt. Es ist somit leicht möglich, ohne weiteren Diskettenzugriff, den Inhalt zu restaurieren, wenn man bemerkt, daß die Änderungen doch nicht so gut waren. Alle Änderungen werden sowieso nur auf ausdrücklichen Wunsch abgespeichert!

Mit 'V' für Vorwärts wird der nachfolgende Block, welcher durch die ersten beiden Bytes im angezeigten Block erkenntlich ist, eingelesen. Die Tr/Sk-Nummer des vorher angezeigten Blocks wird in einem Stack zwischengespeichert.

Mit 'Z' für Zurück wird wieder der vorhergehende Block angezeigt.

Mit 'L' für Laden kann ein neuer Block eingelesen werden.

Mit 'A' für Abspeichern können die Änderungen wieder zurückgeschrieben werden.

☞ Eine besondere Stelle nimmt die Tr/Sk-Nummer '18,0' ein, da diese den Anfang der Directory darstellt. Will man nun in einer bestimmten Datei blättern, so muß man zunächst die Anfangs-Tr/Sk-Nummer finden. Dazu gibt man ganz einfach die Tr/Sk-Nummer '18,0' ein und drückt wiederholt solange auf die Taste 'V', bis die Directoryeintragung der betreffenden Datei angezeigt wird. Nun schaut man sich die beiden Bytes (in HEX), die vor dem Dateinamen stehen an, da diese den Track und Sektor es ersten Blocks der Datei angeben (dazu kann man auch in den Editor gehen damit man die beiden Bytes leichter identifizieren kann).

```
1500  CLS:PRINT "Zurueck :"
1510  IF SP=0 THEN PRINT "Stack underflow";:CALLM #D6DA:GOTO 180
1520  SP=SP-1:TR=T(SP):SEC=S(SP):GOTO 330
2000  CLS:GOSUB 3010
2010  LINE=0:COL=0:INC=0:REM INC=0 ==> ASCII
2020  PRINT "Edit : ^,↓,←,→  Sh←,Sh→  Sh^  TAB":CURSOR 51,23:PRINT
       "Byte:"
2030  GOSUB 6000
2040  CURSOR 56,23:A$=STR$(16*LINE+COL)
2050  A$=MID$(A$,1,LEN(A$)-3):PRINT RIGHT$("  "+A$,3)
2060  IF INC=0 THEN CURSOR 40+COL,16-LINE:GOTO 2080
2070  CURSOR 7+2*COL,16-LINE
2080  A=GETC
2090  A=GETC:IF A=0 GOTO 2090
2100  IF A=9 THEN GOSUB 5000:GOTO 180
2110  IF A=16 THEN GOSUB 5000:LINE=LINE-1:IF LINE>(-1) GOTO 2030:L
      INE=15:GOTO 2030
2120  IF A=17 THEN GOSUB 5000:LINE=LINE+1:IF LINE<16 GOTO 2030:LIN
      E=0:GOTO 2030
2130  IF A<>18 GOTO 2150
2140  GOSUB 5000:COL=COL-1:IF COL>(-1) GOTO 2030:COL=15:LINE=LINE-
      1:IF LINE>(-1) GOTO 2030:LINE=15:GOTO 2030
2150  IF A<>19 GOTO 2170
2160  GOSUB 5000:COL=(COL+1) MOD 16:IF COL<>0 GOTO 2030:LINE=(LINE
      +1) MOD 16:GOTO 2030
2170  IF A=20 THEN X=CURX:CURSOR 0,CURY:CALLM MOVE:CALLM MP:CURSOR
       X,CURY:GOTO 2030
2180  IF A=22 THEN INC=1:GOTO 2030
2190  IF A=23 THEN INC=0:GOTO 2030
2200  IF A=8 GOTO 2140
2210  IF INC=1 GOTO 2250
2220  X=A:X=X IAND 127:IF X<32 THEN X=95
2230  PRINT CHR$(X):POKE #B000+16*LINE+COL,A
2240  CURSOR 7+2*COL,16-LINE:PRINT RIGHT$("0"+HEX$(A),2):GOTO 2160
2250  IF A<#30 OR (A>#39 AND A<#41) OR A>#46 GOTO 2090:A=A-#30:IF
      A>9 THEN A=A-7
2260  PRINT HEX$(A);:X=16*A
2270  A=GETC
2280  A=GETC:IF A=0 GOTO 2280
2290  IF A<#30 OR (A>#39 AND A<#41) OR A>#46 GOTO 2280:A=A-#30:IF
      A>9 THEN A=A-7
2300  PRINT HEX$(A):X=X+A
2310  POKE #B000+16*LINE+COL,X:X=X IAND 127:IF X<32 THEN X=95
2320  CURSOR 40+COL,16-LINE:PRINT CHR$(X):GOTO 2160
3000  POKE #8C,#45:POKE #8D,#BC:RETURN:REM UM
3010  POKE #8C,#5F:POKE #8D,#B3:RETURN:REM NORMAL
4000  OK=0
4010  FOR I=0 TO 3
4020  IF TR>=TM1(I) AND TR<=TM2(I) AND SEC>=0 AND SEC<=SM(I) THEN
      OK=1
4030  NEXT:RETURN
5000  POKE #76,0
5010  POKE #BC2C-4*COL-134*LINE,0
5020  POKE #BC2A-4*COL-134*LINE,0
5030  POKE #BBEA-2*COL-134*LINE,0:RETURN
6000  POKE #BC2C-4*COL-134*LINE,#FF
6010  POKE #BC2A-4*COL-134*LINE,#FF
6020  POKE #BBEA-2*COL-134*LINE,#FF:RETURN
10000 DATA #C5,#21,#39,#BC,#11,#00,#B0,#06,#10,#AF,#CD,#42,#B2,#7B
      ,#CD,#42
10010 DATA #B2,#2B,#2B,#0E,#10,#D5,#1A,#CD,#42,#B2,#13,#0D,#C2,#16
      ,#B2,#D1
10020 DATA #2B,#2B,#0E,#10,#1A,#E6,#7F,#FE,#20,#D2,#2E,#B2,#3E,#5F
```

ABLEITUNG

(von A. Koldehoff)

Das folgende BASIC-Programm erzeugt eine Ableitungsfunktion f'(x) einer zuvor eingegebenen Funktion f(x). Das Programm stammt im Wesentlichen aus der Zeitschrift mc 2/1984, wurde etwas erweitert und an den DAI angepasst.
Ableitungsvariable ist x; andere Variable werden als Konstanten interpretiert. Zulässige Eingaben sind darüber hinaus Zahlen, Klammern und die in der folgenden Tabelle aufgelisteten neunzehn Funktionen:

sin(	cos(	sinh(	cosh(	exp(
sqr(	tan(	cot(	tanh(	coth(
arcsin(	arccos(	arctan(	arccot(	arsinh(
arcosh(	artanh(	arcoth(	ln(	

Die Funktionen können in beliebiger Verschachtelung eingegeben werden. Es gilt dabei die normale BASIC-Syntax. (Ausnahme: Das Minuszeichen wird vom Programm nicht als Vorzeichen erkannt, die Funktion sinh(-x) muß daher z.B. als sinh(0-x) eingegeben werden.) Die Ableitung der eingegebenen Funktion wird rekursiv berechnet, d.h., der Funktionsterm wird in der Reihenfolge aufsteigender Rechenzeichenpriorität in einfachere Terme zerlegt aus deren Ableitungen dann die Ableitungsfunktion gemäß der Summen-, Produkt-, Quotienten- bzw. Kettenregel zusammengesetzt wird.

☞ Das Programm kann nicht kürzen, d.h. daß z.B. die Funktion f(x)=x/x nach der Quotientenregel abgeleitet wird. Auch im Ergebnis f'(x) kann so etwas auftreten.

```
*LIST (return)
100   REM ***ABLEITUNG*** 09.07.1985
110   CLEAR 6000:M=16:REM hoechste Verschachtelungstiefe
120   DIM F$(M),FS$(M),FU$(M),FV$(M),F1$(M),O(M):Z$="+-*/^"
130   READ FZ:DIM X$(FZ),XS$(FZ),L(FZ)
140   FOR I=1 TO FZ:READ X$(I),XS$(I):L(I)=LEN(X$(I)):NEXT
150   PRINT CHR$(12);"ABLEITUNG":PRINT "=========="
160   RG=0:POKE #2C3,0:INPUT "f(x)";F$(0):PRINT
170   Z=0:RG=RG+1
180   GOSUB 290
190   PRINT "f";
200   FOR I=1 TO RG:PRINT "'";:NEXT
210   PRINT "(x)=";FS$(0)
220   PRINT "Weiter ableiten (J/N) ?";
230   G=GETC
240   G=GETC IAND #DF:G$=CHR$(G)
250   IF G$="N" THEN PRINT G$:PRINT :GOTO 160
260   IF G$<>"J" GOTO 240
270   PRINT G$;F$(0)=FS$(0):GOTO 170
280   REM ABLEITUNGS-UNTERPROGRAMM
290   F$=F$(Z)
300   S$=F$:GOSUB 1230
310   IF P=1 OR P=L THEN 1170
320   IF P=0 GOTO 900
330   REM F=U # V, BERECHNE U' UND V'
340   O(Z)=O:FU$(Z)=LEFT$(F$,P-1):FV$(Z)=RIGHT$(F$,LEN(F$)-P)
350   Z=Z+1:F$(Z)=FU$(Z-1):GOSUB 290:F1$(Z-1)=FS$(Z)
360   F$(Z)=FV$(Z-1):GOSUB 290:Z=Z-1
370   FU$=FU$(Z):FV$=FV$(Z):F1$=F1$(Z):F2$=FS$(Z+1):O=O(Z)
380   ON O GOTO 400,400,450,450,670
390   REM F=U±V
400   IF F1$<>"0" THEN IF F2$<>"0" THEN FS$(Z)=F1$+MID$(Z$,O-1,1)+
      F2$:RETURN
410   IF F1$<>"0" THEN FS$(Z)=F1$:RETURN
420   IF F2$<>"0" THEN FS$(Z)=MID$(Z$,1,O-1)+F2$:RETURN
430   FS$(Z)="0":RETURN
440   REM F=U*/V
450   IF F1$="0" THEN T=0:FS$=F1$:GOTO 510
460   T=1
470   IF F1$="1" THEN FS$=FV$:GOTO 510
```

```
1160  FS$(Z)="0":RETURN
1170  PRINT "??? SYNTAX ERROR IN FORMULA":PRINT F$:PRINT :GOTO 160
1180  REM Suche Rechenzeichen mit niedrigster Prioritaet
1190  REM Eingabe : S$ Suchstring
1200  REM Ausgabe : P  Index des Rechenzeichens in S$
1210  REM .         O  Art des Rechenzeichens (1 +,2 -,3 *,4 /,5 ^
      ,6 keins)
1220  REM .         L  Laenge des Strings
1230  L=LEN(S$):P=0:O=6:KL=0
1240  FOR I=1 TO L
1250  K$=MID$(S$,I-1,1)
1260  IF K$=")" THEN KL=KL-1
1270  IF K$="(" THEN KL=KL+1
1280  IF KL<>0 GOTO 1370:REM Klammerausdruecke ueberlesen
1290  IF K$<>"+" AND K$<>"-" GOTO 1320
1300  P=I:O=1
1310  IF K$="-" THEN O=O+1
1320  IF K$<>"*" AND K$<>"/" GOTO 1360
1330  IF O<=2 GOTO 1360
1340  P=I:O=3
1350  IF K$="/" THEN O=O+1
1360  IF K$="^" AND O>4 THEN P=I:O=5
1370  NEXT
1380  IF KL<>0 GOTO 1170:REM Zahl der "(" <> Zahl der ")"
1390  RETURN
1400  REM Teste, ob Term eine Zahl ist
1410  REM Eingabe : S$ Term
1420  REM Ausgabe : N 1 fuer Zahlen, 0 fuer zusammengesetzte Ausdr
      uecke
1430  FOR I=1 TO LEN(S$)
1440  K$=MID$(S$,I-1,1)
1450  IF K$>"/" AND K$<":" THEN NEXT:N=1:RETURN
1460  IF K$="-" THEN NEXT:N=1:RETURN
1470  N=0:RETURN
1480  REM ERLAUBTE FUNKTIONEN
1490  DATA 19
1500  DATA sin(,cos(
1510  DATA cos(,(-1)*sin(
1520  DATA sinh(,cosh(
1530  DATA cosh(,sinh(
1540  DATA exp(,exp(
1550  DATA sqr(,.5/sqr(
1560  DATA tan(,(1+tan(
1570  DATA cot(,(-1)*(1+cot(
1580  DATA tanh(,(1-tanh(
1590  DATA coth(,(-1)*(1-coth(
1600  DATA arcsin(,sqr(1-(
1610  DATA arccos(,(-1)/sqr(1-(
1620  DATA arctan(,(1+(
1630  DATA arccot(,(-1)/(1+(
1640  DATA arsinh(,sqr(1+(
1650  DATA arcosh(,(-1)/(sqr(-1+(
1660  DATA artanh(,(1-(
1670  DATA arcoth(,(1-(
1680  DATA ln(,(
```

BEISPIEL:

```
*RUN (return)
ABLEITUNG
=========
f(x)=sinh(cos(2*x)) (return)
f'(x)=2*(-1)*sin(2*x)*cosh(cos(2*x))
Weiter ableiten (J/N) ?J
f''(x)=2*(-1)*2*cos(2*x)*cosh(cos(2*x))+2*(-1)*sin(2*x)*2*(-1)*sin(
       2*x)*sinh(cos(2*x))
```

```
! ***** Magisches Quadrat ***** !
VAR ZEILE,SPALTE,A,X,GROESSE,SUM      : INTEGER;
    M1,M2,M3,M4,M5,M6,M7 : ARRAY[7] OF INTEGER;

PROC INMAT(ZEILE,SPALTE,ZAHL);
 BEGIN
  CASE ZEILE OF
   1 : M1[SPALTE]:=ZAHL;
   2 : M2[SPALTE]:=ZAHL;
   3 : M3[SPALTE]:=ZAHL;
   4 : M4[SPALTE]:=ZAHL;
   5 : M5[SPALTE]:=ZAHL;
   6 : M6[SPALTE]:=ZAHL;
   7 : M7[SPALTE]:=ZAHL
  END
 END;

FUNC MATINHALT(ZEILE,SPALTE);
 VAR ZAHL : INTEGER;
 BEGIN
  CASE ZEILE OF
   1 : ZAHL:=M1[SPALTE];
   2 : ZAHL:=M2[SPALTE];
   3 : ZAHL:=M3[SPALTE];
   4 : ZAHL:=M4[SPALTE];
   5 : ZAHL:=M5[SPALTE];
   6 : ZAHL:=M6[SPALTE];
   7 : ZAHL:=M7[SPALTE]
  END;
  MATINHALT:=ZAHL
 END;

PROC DRUCKZAHL(ZAHL);
 VAR HILF:INTEGER;
  BEGIN
   HILF:=ZAHL;
   WHILE HILF<1000 DO
    BEGIN
     WRITE (' ');
     HILF:=HILF*10
    END;
   WRITE (' ',ZAHL%)
  END;

BEGIN !HAUPTPROGRAMM!
 WRITE (12,'Geben sie eine ungerade Zahl von 3...7 ein ');
 READ (GROESSE%);
 IF ((GROESSE MOD 2=1) AND (GROESSE>=3) AND (GROESSE<=7))
 THEN
  BEGIN
   WRITE (13,13,'Magisches Quadrat ');
   WRITE (GROESSE%,' * ',GROESSE%,13);
   WRITE ('**************************',13);
   X:=0;
   FOR ZEILE:=1 TO GROESSE DO
    FOR SPALTE:=1 TO GROESSE DO INMAT(ZEILE,SPALTE,X);
   ZEILE:=1;
   A:=GROESSE * GROESSE;
   SPALTE:=(GROESSE DIV 2)+1;
   X:=1;INMAT(ZEILE,SPALTE,X);
```

```
! Primfaktoren-Analyse !

VAR   FLAGS : ARRAY[1800] OF INTEGER;
      ZAHL,I,YN                : INTEGER;
PROC PRIMGEN(ZAHL);
 VAR HILF,I : INTEGER;
 BEGIN
  FOR I:=2 TO ZAHL DO FLAGS[I] := I;
  FOR I:=2 TO ZAHL DO
   IF FLAGS[I] <> 0
    THEN
     BEGIN
      HILF:=2*I;
      WHILE HILF <= ZAHL DO
       BEGIN
        FLAGS[HILF]:=0;
        HILF:=HILF+I;
       END;
     END;
 END;

BEGIN ! HAUPTPROGRAMM !
 YN:='Y';
 WHILE YN='Y' DO
  BEGIN
   WRITE(13,13,'ZU ANALYSIERENDE ZAHL ');
   READ(ZAHL%);
   IF ZAHL <= 1800
    THEN
     BEGIN
      WRITE(13,ZAHL%,'= ');
      PRIMGEN(ZAHL);
      I:=2;
      WHILE ZAHL<>1 DO
       BEGIN
        IF FLAGS[I] <> 0
         THEN
          IF ZAHL MOD FLAGS[I] = 0
           THEN
            BEGIN
             WRITE(FLAGS[I]%);
             ZAHL:=ZAHL DIV FLAGS[I];
             IF ZAHL <> 1 THEN WRITE(' * ');
             I:=1;
            END;
        I:=I+1
       END;
      WRITE(13,13,'NOCH EINMAL (Y/N) ');
      READ(YN);
     END
    ELSE WRITE(13,'ZAHL IST ZU GROSS !!');
  END;
END.
```

Circle und Ellipse (Heinrich Tegethoff)

Es sind bisher schon viele Programme zum Zeichnen von Kreisen in dieser Clubzeitung veröffentlicht worden. Aber alle hatten irgendeinen Schönheitsfehler : waren die Programme in Basic, mit Sinus- und Cosinus- oder auch mit Quadratwurzelberechnungen, so werden die Kreise einfach zu langsam gemalt. Maschinenprogramme gab es nur wenige, schnell und schön gleichzeitig konnte bisher nur NP 74 von Herrn Degenhardt zeichnen. NP 74 hat ausserdem noch den Vorteil, daß Kreispunkte, die über den Bildschirmrand hinausgehen, einfach abgeschnitten werden, also insbesondere kein „OUT OF SCREEN" erzeugen. Allerdings hat das Programm auch einen gravierenden Nachteil : es läuft nur in MODE 6. Ausserdem läßt das Arbeitsprinzip auch noch weitere Möglichkeiten zu : neben Kreisen können (mit diesem Programm) auch Ellipsen gezeichnet werden, und insbesondere : die Kreise/Ellipsen können ausser aus einer Umfangslinie auch aus einem Vollkreis (ausgemalt) bestehen. Bei nicht allzu grossen Kreisen (Ellipsen) ist ein Vollkreis erstaunlicherweise sogar schneller gemalt als seine reine Umfangslinie, so daß man nicht erwarten muß, beim Vollkreismalen einzuschlafen. Auch beim Ausmalen werden Kreise am Rand abgeschnitten. Dieses Kreiszeichnungsprogramm kann ausserdem vom Benutzer viel einfacher aufgerufen werden. CIRCLE und ELLIPSE sind kostenloser Bestandteil von XBASIC, bei dem sie direkt als neue (NCU-) Basicbefehle benutzt werden können. Der Aufruf dort sieht so aus :

```
CIRCLE   x-Koor , y-Koor   radius  farbe
ELLIPSE  x-Koor , y-Koor   x-radius , y-radius  farbe
FILL = ON
FILL = OFF
```

(FILL=ON ≙ Kreis/Ellipse ausmalen, FILL=OFF ≙ nur Umfangslinie). Da nicht jeder XBASIC oder zumindestens ein NCU-System installiert hat, werden bei diesem Programm hier die Befehle folgendermassen aufgerufen :

```
CE=#300:REM siehe Sourcelisting-Zeile 0017

CALLM CE:COLORG  x-Koor   y-Koor  radius  farbe
CALLM CE:FILL    x-Koor , y-Koor  x-radius , y-radius  farbe
CALLM CE:TRON
CALLM CE:TROFF
```

Durch den CALLM CE werden die dahinterstehenden Basicbefehle anders interpretiert, sofern sie zu den obigen 4 gehören. Statt COLORG kann auch COLORT, statt FILL auch DRAW benutzt werden. Man kann beliebg viele uminterpretierte Befehle hintereinander setzen, bis entweder die Basiczeile zu Ende ist oder ein anderer Basicbefehl auftritt. Beispiel :

```
10    CE=#300:MODE 5A
100   FOR R=0 TO 127 STEP 4:CALLM CE:TROFF:COLORG 128 128 R 21:NEXT
```

Die Parameter bei CIRCLE und ELLIPSE (≙COLORG/FILL) können folgende Werte annehmen :

```
x-Koor,y-Koor :  muß auf dem Bildschirm liegen (0..XMAX,0..YMAX)
(x/y-) radius :  0..255, bei 0 bzw. 0,0 Kein Kreis/Ellipse
farbe         :  0..23, Bedeutung wie bei DOT etc.
```

Weitere Beispiele :

```
200   CALLM CE:TRON:FILL 0,0 50,30 21:FILL XMAX,0 70,30 22
      → Mit Randabschneidung ≙ 2 ausgemalte Viertelellipsen in den Ecken

1000  CALLM CE:TROFF:MODE 6
1010  FOR R=3 TO 255 STEP 4:CALLM CE:COLORG 128 128 R 0:NEXT
1020  REM siehe Hardcopy 1 am Schluß des Sourcelistings

2000  MODE 8:REM oder MODE 6
2010  FOR N=1 TO 50:CALLM CE:TROFF
      FILL RND(XMAX),RND(YMAX) RND(XMAX/3),RND(YMAX/3) 0:NEXT
2020  REM siehe Hardcopy 2 am Schluß des Sourcelistings
```

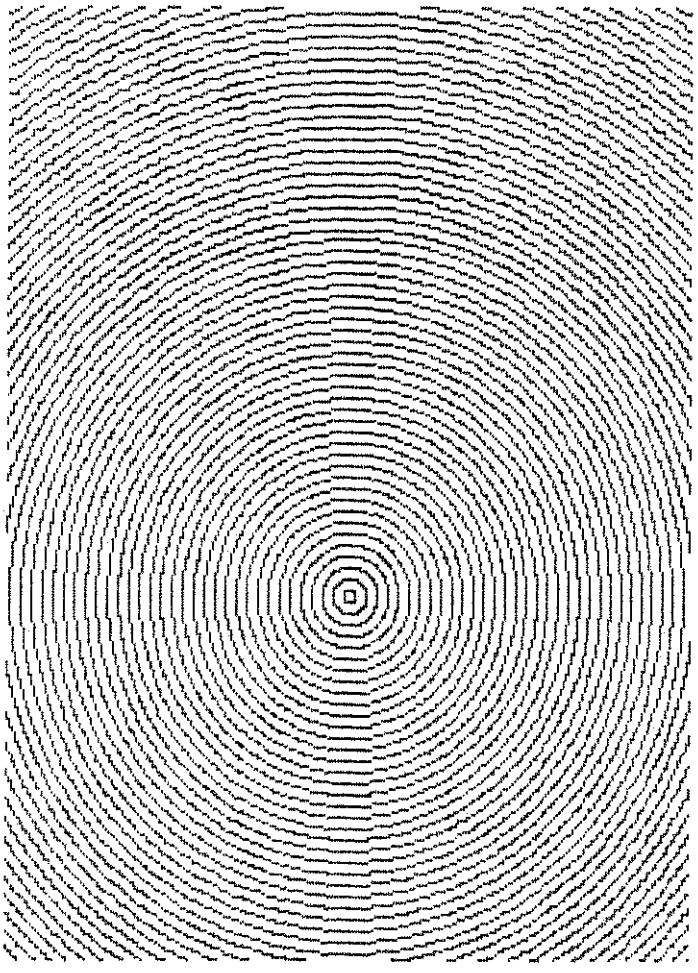
```
0053 0343 1D              DCR   E         E=0 ?  => CIRCLE
0054 0344 5F              MOV   E,A       E:=X-Radius
0055 0345 C41DE7          CNZ   GA8       hole Y-Radius bei ELLIPSE
0056 0348 323106          STA   RY
0057 034B B8              CMP   E
0058 034C DA5003          JC    $+4
0059 034F 5F              MOV   E,A       E:=Max(RX,RY)
0060 0350 CDFDE5          CALL  GACOL
0061 0353 F5              PUSH  PSW       A:=0..23, Farbe
0062 0354 3EB0            MVI   A,:B0     E-Bank 2
0063 0356 CD08D8          CALL  SETFD06
0064 0359 F1              POP   PSW
0065 035A CDC3E9          CALL  SETVCOL   Farbe definieren
0066 035D 3E10            MVI   A,:10
0067 035F DAF5D9          JC    ERROR     "COLOR NOT AVAILABLE"
0068 0362 C5              PUSH  B         BC-Register sichern (*BASIC)
0069 0363 2A2D06          LHLD  X
0070 0366 3A2F06          LDA   Y
0071 0369 4F              MOV   C,A       HL,C:=Mittelpunktkoordinaten
0072 036A CD7AEB          CALL  TSTKOOR   Mittelpunkt auf Bildschirm ?
0073 036D 3E11            MVI   A,:11
0074 036F DAF5D9          JC    ERROR     Nein, "OUT OF SCREEN"
0075 0372 7B              MOV   A,E       A:=Max(RX,RY)=:R'
0076 0373 B7              ORA   A
0077 0374 CA1804          JZ    END1      Fertig, falls RX=RY=0
0078 0377 323706          STA   LDY
0079 037A 3EFF            MVI   A,255
0080 037C 57              MOV   D,A       D:=-1
0081 037D 42              MOV   B,D       B:=-1
0082 037E 14              INR   D
0083 037F 93              SUB   E
0084 0380 D27E03          JNC   $-2
0085 0383 7A              MOV   A,D       A:=255/R', Tabellenschrittweite
0086 0384 FE0A            CPI   10
0087 0386 DA8B03          JC    $+5
0088 0389 3E0A            MVI   A,10
0089 038B 323406          STA   A1        A1:=Min(255/R',10)
0090 038E 2F              CMA
0091 038F 3C              INR   A
0092 0390 4F              MOV   C,A       BC:=-A1
0093 0391 219101          LXI   H,401
0094 0394 3C              INR   A         A1=1 ?
0095 0395 CAA003          JZ    ANZ401    dann Anzahl_Schritte:=401
0096 0398 50              MOV   D,B       DE:=-1
0097 0399 58              MOV   E,B
0098 039A 13              INX   D
0099 039B 09              DAD   B
0100 039C DA9A03          JC    $-2
0101 039F EB              XCHG            HL:=401/A1
0102 03A0 110000  ANZ401  LXI   D,0       DE:=Tabellenlaeufer
0103 03A3 E5      LOOP    PUSH  H         HL:=Anzahl der Schritte fuer
0104 03A4 3A3406          LDA   A1        Viertelkreis
0105 03A7 83              ADD   E
0106 03A8 5F              MOV   E,A       DE:=DE+(Tabellenschrittweite)
0107 03A9 D2AD03          JNC   $+4
0108 03AC 14              INR   D
0109 03AD D5              PUSH  D
0110 03AE 212D06          LXI   H,TAB.E+1
0111 03B1 7D              MOV   A,L
0112 03B2 93              SUB   E
0113 03B3 6F              MOV   L,A
0114 03B4 7C              MOV   A,H
0115 03B5 9A              SBB   D
0116 03B6 67              MOV   H,A       HL:=(TAB.E)+1-(Tab.Zeiger)
0117 03B7                                 HL:= ↑ (Sinus)
```

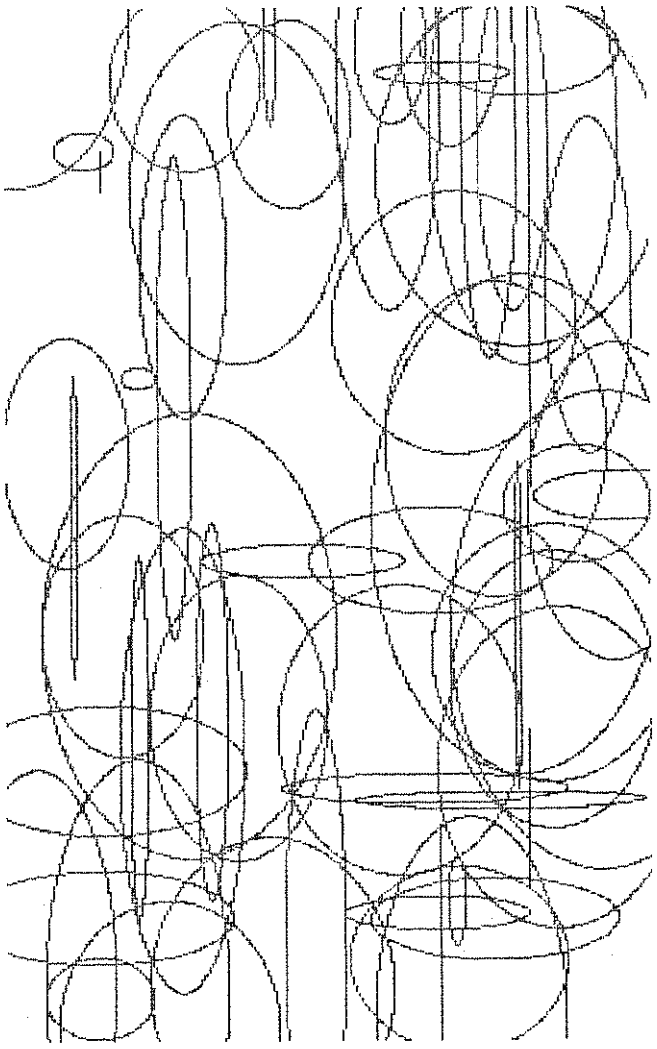
```
0183 0434 EB              XCHG
0184 0435 D44704          CNC   PLOT      rechten Punkt malen
0185 0438 2A2D06          LHLD  X
0186 043B 3A3506          LDA   DX
0187 043E 2F              CMA
0188 043F 5F              MOV   E,A
0189 0440 16FF            MVI   D,:FF
0190 0442 13              INX   D
0191 0443 19              DAD   D         HL:=(Mittelpunkt-X)-DX
0192 0444 7C              MOV   A,H
0193 0445 B4              ORA   H         HL<0 ?
0194 0446 F8              RM              sonst: linken Punkt malen
0195 0447
0196 0447 78       PLOT   MOV   A,B       A:=Y
0197 0448 C5D5E5          PUSH  B,D,H
0198 044B 44              MOV   B,H       BC:=X
0199 044C 4D              MOV   C,L
0200 044D CDB9EB          CALL  MASKADR   HL:= Screen_Adr(BC,A)
0201 0450 79              MOV   A,C
0202 0451 E607            ANI   7         Registerinitialisierung fuer
0203 0453 4F              MOV   C,A       'l Punkt'
0204 0454 1600            MVI   D,0       Hoehe-1
0205 0456 0601            MVI   B,1       Breite
0206 0458 CDE1EB          CALL  :EBE1     Mask-Bit berechnen
0207 045B C35AEA          JMP   :EA5A     Dot !  (Stack beachten)
0208 045E
0209 045E 19       FILLING DAD   D        HL>XMAX ?
0210 045F EB              XCHG
0211 0460 D26704          JNC   $+7
0212 0463 2A9400          LHLD  :94
0213 0466 2B              DCX   H         Ja: => HL:=XMAX
0214 0467 E5              PUSH  H         X-rechts
0215 0468 2A2D06          LHLD  X
0216 046B 3A3506          LDA   DX
0217 046E 2F              CMA
0218 046F 5F              MOV   E,A
0219 0470 16FF            MVI   D,:FF
0220 0472 13              INX   D
0221 0473 19              DAD   D         HL:=X-links
0222 0474 78              MOV   A,B       A:=Y
0223 0475 D1              POP   D         DE:=X-rechts
0224 0476 F5C5D5E5        PUSH  PSW,B,D,H
0225 047A 24              INR   H
0226 047B 25              DCR   H
0227 047C F28204          JP    $+6       HL<0 ?
0228 047F 210000          LXI   H,0       Ja: => HL:=0
0229 0482 44              MOV   B,H       BC:=X-links
0230 0483 4D              MOV   C,L
0231 0484 EB              XCHG
0232 0485 CD1ADE          CALL  SUB
0233 0488 EB              XCHG            DE:=delta-X  (Breite-1)
0234 0489 CDB9EB          CALL  MASKADR   HL:=S_Adr(BC,A)
0235 048C 79              MOV   A,C
0236 048D E607            ANI   7
0237 048F 4F              MOV   C,A
0238 0490 0600            MVI   B,0       Hoehe:=1
0239 0492 C3FBEA          JMP   FLLSTRP   Linie zeichnen
0240 0495
0241 0495 6E       R.D.256 MOV   L,M
0242 0496 2600            MVI   H,0       HL:=Tabellenwert (Co-/Sinus)
0243 0498 C38FDE          JMP   MUL       HL:=HL*A, H wichtig
0244 049B
0245 049B
0246 049B 00       FILLFLG DATA  0         anfangs FILL=OFF
0247 049C
```

```
0019 0300 0A 03 CD 08 03 C3 00 03 FE BE CA 21 03 FE BD CA
0027 0310 21 03 E6 FE FE 9C CA 27 03 FE 9E CA 28 03 E1 0B
0035 0320 C9 D6 BE 32 9B 04 C9 11 1E 00 CD F3 E5 22 2D 06
0046 0330 32 2F 06 2A 94 00 CD 26 DE 22 32 06 CD 1D E7 32
0051 0340 30 06 1C 1D 5F C4 1D E7 32 31 06 BB DA 50 03 5F
0060 0350 CD FD E5 F5 3E B0 CD 08 D8 F1 CD C3 E9 3E 10 DA
0067 0360 F5 D9 C5 2A 2D 06 3A 2F 06 4F CD 7A EB 3E 11 DA
0074 0370 F5 D9 7B B7 CA 18 04 32 37 06 3E FF 57 42 14 93
0084 0380 D2 7E 03 7A FE 0A DA 8B 03 3E 0A 32 34 06 2F 3C
0092 0390 4F 21 91 01 3C CA A0 03 50 58 13 09 DA 9A 03 EB
0102 03A0 11 00 00 E5 3A 34 06 83 5F D2 AD 03 14 D5 21 2D
0110 03B0 06 7D 93 6F 7C 9A 67 3A 31 06 CD 95 04 7C 32 36
0121 03C0 06 2A 37 06 32 37 06 BD C2 D2 03 3A 98 04 B7 C2
0128 03D0 10 04 F5 21 98 04 19 3A 30 06 CD 95 04 7C 32 35
0135 03E0 06 F1 C2 EC 03 3A 38 06 BC CA 10 04 7C 32 38 06
0144 03F0 3A 36 06 47 3A 2F 06 80 47 DA 04 04 3A 96 00 3D
0152 0400 B8 D4 1E 04 3A 36 06 47 3A 2F 06 90 47 D4 1E 04
0161 0410 D1 E1 2B 7C B5 C2 A3 03 C1 3E 30 C3 08 D8 2A 2D
0172 0420 06 3A 35 06 5F 16 00 19 EB 2A 32 06 3A 98 04 B7
0181 0430 C2 5E 04 19 EB D4 47 04 2A 2D 06 3A 35 06 2F 5F
0189 0440 16 FF 13 19 7C B4 F8 78 C5 D5 E5 44 4D CD B9 EB
0201 0450 79 E6 07 4F 16 00 06 01 CD E1 EB C3 5A EA 19 EB
0211 0460 D2 67 04 2A 94 00 2B E5 2A 2D 06 3A 35 06 2F 5F
0219 0470 16 FF 13 19 78 D1 F5 C5 D5 E5 24 25 F2 82 04 21
0228 0480 00 00 44 4D EB CD 1A DE EB CD B9 EB 79 E6 07 4F
0238 0490 06 00 C3 FB EA 6E 26 00 C3 8F DE 00
```

Die Sinus/Cosinus-Tabelle muß man nicht eintippen, sondern sie kann wie bei NP 74 eingepokt werden (IMPFPT, IMPINT A-A) :

```
A=#49C
FOR X=0 TO PI/2 STEP 1/255 : POKE A,255.0*COS(X)+0.5:A=A+1 : NEXT
POKE #49C,#FE
POKE #62C,1
```





Uhr

Autor M.Rahlff

Dieses Programm erstellt eine Uhr, die auf eine einfache Art ausgelesen und zurückgesetzt werden kann. Hierfür wird eine Stringvariable der Länge 6 [std,std,min,min,sec,sec] verwendet.

Nach dem Start des Programms wird das untenstehende Maschinenprogramm in die Adressen #300-#3E8 gepokt (Der HEAP des BASIC Programms sollte daher mindestens bei #3E9 liegen!). Dem Progamm wird dann der Zeiger auf die gewünschte String Variable übergeben (Zeile 50). Der Name der Variablen ist unerheblich! Mit einem CALLM #310 wird dann das Programm gestartet. Sie können Ihr BASIC Programm ganz normal weiterlaufen lassen und z.B. mit PRINT T$ ausgeben lassen, wieviel Zeit seit dem Start vergangen ist. Durch T$="000000" können Sie die Uhr auch wieder zurücksetzen lassen.

```
10     REM ** UHR **   Copyright by M.Rahlff
20     REM
30     CLEAR 300:T$="000000"
40     FOR ADR=#300 TO #3E8:READ BYTE:POKE ADR,BYTE:NEXT
50     POKE 964,VARPTR(T$)/256:POKE 963,VARPTR(T$) IAND 255
60     CALLM #310:T$="000000":END
1000   REM
1010   REM -----  ML Programm  --------
1020   DATA 6,50,52,50,48,53,52,52,3,73
1030   DATA 84,84,69,32,87,65,245,62,6,50
1040   DATA 0,3,243,62,37,50,112,0,62,3
1050   DATA 50,113,0,251,195,155,3,245,229,195
1060   DATA 177,3,0,0,33,243,2,126,60,119
1070   DATA 254,58,218,150,3,62,48,119,43,126
1080   DATA 60,119,254,54,218,52,3,62,48,119
1090   DATA 43,126,60,119,254,58,218,52,3,62
1100   DATA 48,119,43,126,60,119,254,54,218,52
1110   DATA 3,62,48,119,43,126,60,50,7,3
1120   DATA 119,254,58,202,114,3,254,52,202,124
1130   DATA 3,195,150,3,62,48,119,43,126,60
1140   DATA 119,195,150,3,43,126,254,50,218,52
1150   DATA 3,0,0,0,42,45,3,62,48,119
1160   DATA 43,119,43,119,43,119,43,119,43,119
1170   DATA 225,241,195,169,217,62,3,50,156,2
1180   DATA 62,0,50,158,2,62,128,50,236,3
1190   DATA 62,40,50,237,3,241,201,58,8,3
1200   DATA 60,50,8,3,254,50,218,52,3,62
1210   DATA 0,50,8,3,42,28,4,35,35,35
1220   DATA 35,35,35,34,45,3,195,42,3,3
1230   DATA 49,77,86,73,32,68,44,48,82,65
1240   DATA 76,44,45,44,48,68,65,68,32,68
1250   DATA 44,48,76,3,127,1,151,108,127,255


   32 Lines            3 Symbols
    0 References       8 References
 1041 Bytes Text      24 Bytes Symbols

   39 Commands
```

DATA WRITER:

Autor: M.Rahlff
Überarbeitet durch: Uwe Wienkop

Dieses Programm ermöglich auf eine besonders interessante Art die Überführung von HEX-Daten (z.B. eines MP Programmes) in ein BASIC Programm. Die hier verwendete Technik ist dabei so einfach und zugleich universell, daß auch weitergehende Aufgaben, z.B. ein Maskengenerator ua, damit möglich sind.

Das zu erstellende Programm wird hinter dem momentanen BASIC Programm im ASCII Format in den Speicher gepokt (Zeilen 4000-). Mit ASCII Format ist folgendes gemeint: Text und als Zeilenende ein 0D. Somit wird nun das komplette neue Programm im Speicher hinter dem eigentlichen BASIC Programm erstellt. Als Endekennung des neuen Programms wird zum Schluß noch das Zeichen #FF angehängt.

Nun werden der Eingabevektor des DAIs (DINC=#2E0) und das Eingabeflag (#296) auf ein kleines Maschinenprogramm umgesetzt. Dieses Programm liest dann den Speicher (mit dem neu erstellten Programm) wieder aus und gibt dabei jeweils ein Zeichen als neues Eingabezeichen an den DAI ab, quasi so, als würden Sie diese Daten jetzt eingeben. Beim Auffinden des letzten Zeichens (das #FF) wird das Eingabeflag wieder zurückgesetzt, so daß Sie nun normal weitermachen können.

Mit dieser Technik sind übrigens auch Direkt Kommandos, wie z.B. NEW (siehe Zeile 1050) möglich!

Noch eine Anmerkung zur Eingabe des Programms: Die BASIC Version V1.2 ermöglicht, den Unterstreichungsstrich in Variablen zu verwenden. Sollten Sie nur eine BASIC Version V1.0/V1.1 besitzen, so lassen Sie den Strich bei der Eingabe einfach weg!

```
10    REM Daten - Writer
20    REM Copyright by M.Rahlff
30    REM
40    CLEAR 1000:POKE #29C,3:CLEAR 1000
50    FOR CHAR=750 TO 790:READ BYTE:POKE CHAR,BYTE:NEXT CHAR
60    PRINT CHR$(12);"          *** Data - Writer ***"
70    PRINT :INPUT "In welchem Bereich steht Ihr Programm (von,
      bis)";VON,BIS
80    PRINT :INPUT "Mit welcher Zeilennummer sollen die Daten
      beginnen";LINE$:PRINT
90    PRINT :PRINT :PRINT "** Ich schreibe die Daten **"
1000  REM
1010  REM --------- INIT ----------
1020  REM
1030  BUF_BEG=PEEK(#2A3)+PEEK(#2A4)*256:BUF_BEG=BUF_BEG+100:
      POKE 750,BUF_BEG MOD 256:POKE 751,INT(BUF_BEG/256)
1040  BUF_END=PEEK(#2A5)+PEEK(#2A6)*256-2
1050  A$="NEW"+CHR$(13):GOSUB 4030
1060  A$=LINE$+" FOR ADR= #"+HEX$(VON)+" TO #"+HEX$(BIS)+":READ
      BYTE:POKE ADR,BYTE:NEXT:END"+CHR$(13)
1070  GOSUB 4030
1080  NXT_LINE=VAL(LINE$)
2000  REM
2010  REM ------- SCHLEIFE ---------
2020  REM
2030  NXT_LINE=NXT_LINE+10:N$=STR$(NXT_LINE):LINE$=MID$(N$,1,
      LEN(N$)-3)
2040  A$=LINE$+" DATA  #"+HEX$(PEEK(VON)):GOSUB 4030:VON=VON+1
2050  FOR CHAR=1 TO 10:A$=",#"+HEX$(PEEK(VON)):GOSUB 4030:
      VON=VON+1
```

Birthday song

```
100   REM +++ BIRTHDAY SONG - Barry Hunt - 4/3/82 +++
110   REM +++ The name of the person whose birthday +++
120   REM +++ is being celebrated should be substituted +++
130   REM +++ for "CLAUDIA" (the name of my wife) +++
140   REM +++ in the third line of the song +++
150   REM +++ and the third DATA  statement should be +++
160   REM +++ changed accordingly +++
170   PRINT CHR$(12):POKE #75,32
180   CURSOR 20,20:PRINT "BIRTHDAY  SONG"
190   CURSOR 20,19:FOR I=1 TO 14:PRINT CHR$(11);:NEXT I
200   CURSOR 16,16:PRINT "Press any key to start."
210   IF GETC=0.0 THEN 210
220   CURSOR 16,16:PRINT "                          "
230   FOR I=3 TO 1 STEP -1
240   CURSOR 25,16:PRINT I:WAIT TIME 30
250   NEXT I
260   MODE 0:COLORT 8 0 8 12:PRINT CHR$(12)
270   ENVELOPE 0 2,4;10,6;15,20;12,30;8,40;4,50;0
280   REM ++ change text letter size
290   FOR ADDR=#BFEF TO #BCCB STEP -#86
300   POKE ADDR,#6A
310   NEXT ADDR
320   PRINT
330   PRINT "    HAPPY  BIRTHDAY  TO  YOU"
340   PRINT "    HAPPY  BIRTHDAY  TO  YOU"
350   PRINT "    HAPPY  BIRTHDAY  DEAR  CLAUDIE"
360   PRINT "    HAPPY  BIRTHDAY  TO  YOU"
370   REM ++ set start of line and current position
380   START=#BED2:POSN=START
390   FOR I=1 TO 28
400   REM ++ read note frequncy, note length and (length of syllab
      le - 1)
410   READ NOTE,NL,CL
420   REM ++ Note of zero means new line
430   IF NOTE=0 THEN 610
440   SOUND 0 0 15 0 FREQ(NOTE)
450   SOUND 0 0 8 0 FREQ(NOTE*2.0)
460   SOUND 2 0 4 0 FREQ(NOTE*4.0)
470   REM ++ change colour of syllable
480   FOR ADDR=POSN TO (POSN-2*CL) STEP -2
490   POKE ADDR,#FF
500   NEXT ADDR
510   WAIT TIME 8*NL
520   SOUND OFF
530   REM ++ restore colour of syllable
540   FOR ADDR=POSN TO (POSN-2*CL) STEP -2
550   POKE ADDR,#0
560   NEXT ADDR
570   REM ++ update current position
580   POSN=POSN-2*CL-2
590   GOTO 630
600   REM ++ new line
610   WAIT TIME NL*8
620   START=START-#86:POSN=START
630   NEXT I
640   DATA 264,1,2,264,1,3,297,2,4,264,2,3,352,2,3,330,4,3,0,2,0
650   DATA 264,1,2,264,1,3,297,2,4,264,2,3,396,2,3,352,4,3,0,2,0
660   DATA 264,1,2,264,1,3,528,2,4,440,2,3,352,2,5,330,2,4,297,2,2
      ,0,0,0
670   DATA 467,1,2,467,1,3,440,2,4,352,2,3,396,2,3,352,4,3
680   POKE #75,95
690   RESTORE:GOTO 260
```

```
1       REM ** RAKETENSPIEL , copyright by DAInamic **
2       REM ********* 81, hb, rh  **********************
3       MODE Ø:POKE #75,32
5       GOSUB 2ØØØ
6       SOUND OFF
8       DIM A(6.Ø)
1Ø      PRINT CHR$(12):PRINT
2Ø      PRINT TAB(15);"****** RAKETENSPIEL ******":PRINT
4Ø      PRINT "  Sie haben 6 Raketen auf der Basis stehen,mit
        denen"
45      PRINT "  das am Himmel vorbeifliegende Flugobjekt 'UFO
        ' ge-"
5Ø      PRINT "  troffen werden muss."
51      PRINT "  Die Geschwindigkeit (MACH) der Rakete und des
         UFO aen-"
52      PRINT "  dert sich nach jedem Schuss bzw. Vorbeiflug !
        !"
53      PRINT "  Sie zuenden die Raketen mit den Tasten ' 1 -
        6 ',"
54      PRINT "  entsprechend der Nummer wird die Rakete abges
        chossen"
55      PRINT
57      PRINT CHR$(13):PRINT TAB(4);"Wollen Sie beginnen ?  ,
        druecken Sie eine Taste "
58      C=GETC:IF C=Ø THEN 58
59      PRINT CHR$(12):COLORT 14 Ø Ø Ø:CURSOR 12,12:PRINT "***
        ** ACHTUNG  wir starten *****":WAIT TIME 15Ø
6Ø      COLORG Ø Ø Ø Ø:E=INT(RND(1.Ø)*3.Ø)+1.Ø
65      X=Ø:FOR Q=Ø TO 6:A(Q)=Ø:NEXT
7Ø      COLORT 8 Ø Ø Ø
8Ø      PRINT CHR$(12):MODE 1A
9Ø      D=Ø:X=Ø:F=Ø
1ØØ     FOR A=1Ø TO 6Ø STEP 1Ø
11Ø     DRAW A,2 A,5 15:DRAW A-1,1 A+1,1 1Ø
115     NEXT A
12Ø     PRINT "          1       2       3       4       5
         6"
13Ø     DRAW 1,Ø 7Ø,Ø 11
14Ø     PRINT "      ANZAHL abgefeuerte RAKETEN:";F
15Ø     PRINT "      ANZAHL getroffene  U F O's:";D
155     CURSOR 42,2:PRINT "Geschw.RAKETE"
16Ø     CURSOR 42,1:PRINT "MACH. ";E
165     FA=12
17Ø     GOSUB 1ØØØ
2ØØ     C=GETC:IF C=Ø THEN GOSUB 1ØØØ
21Ø     IF C=Ø THEN 2ØØ
211     IF C>54 OR C<49 THEN 2ØØ
22Ø     C1=1Ø*(C-48):Q=C1/1Ø
222     IF A(Q)<>Ø.Ø THEN 2ØØ
224     A(Q)=1
23Ø     F=F+1
24Ø     B=2
3ØØ     FOR I=1 TO E*2:B=B+1
3Ø5     SOUND 1 Ø 15 3 FREQ(15ØØ.Ø):SOUND 1 Ø 15 2 FREQ(8ØØ.Ø)

31Ø     DRAW C1,B-2 C1,B+3 15:DRAW C1-1,B-1 C1+1,B-1 1Ø
```

```
40    ENVELOPE 1 15,10;0,10;
50    MODE 0:PRINT CHR$(12);
55    COLORT 12 2 12 12:PRINT " 6 0 M 0 K 0"
60    POKE #BFEF,#5C
70    PRINT :PRINT :PRINT " Verhindern Sie dass der Computer fuenf
       Spielsteine in"
71    PRINT :PRINT "                          eine Reihe  setzt ."
75    PRINT :PRINT :PRINT "      Horizontal , vertikal oder diagona
      l gewinnt."
80    PRINT :PRINT :PRINT "  Geben Sie zuerst die horizontale Zahl
       und dann durch"
85    PRINT :PRINT "            Komma getrennt die vertikale Zahl e
      in ."
90    PRINT :PRINT :PRINT "                    Sie spielen mit  ''Schw
      arz'', "
91    PRINT :PRINT "                und muessen den Computer besiegen
      ."
140   FOR J%=#5915 TO #5ACA:READ K%:POKE J%,K%:NEXT J%
160   FOR J=#5D00 TO #5E90:POKE J,0:NEXT J
170   BOARD=#5E2C
171   WAIT TIME 200:PRINT CHR$(12):PRINT :PRINT " SPIELBEREIT ?
      DRUECKE TASTE !"
175   KEY=GETC:IF KEY=0.0 THEN 175
180   PRINT CHR$(12);:GOSUB 1000
190   PRINT CHR$(12);:INPUT "Wollen Sie anfangen ";A$:IF A$="" THE
      N 190
195   PRINT :PRINT
200   IF LEFT$(A$,1)="N" THEN 270
210   INPUT "Gebe Zug ein ? (horizontal,vertikal)  ";R,C:PRINT
220   R=R+1.0:C=C+1.0
230   T=(R-1.0)*10.0+C
235   IF R<1.0 OR R>10.0 THEN 245
240   IF C<1.0 OR C>10.0 THEN 245
244   GOTO 250
245   PRINT "Der eingegebene Zug ist nicht moeglich !!":GOSUB 3000
      :GOTO 210
250   IF PEEK(BOARD+T)<>0 THEN PRINT "Dieser Platz ist besetzt ":G
      OSUB 3000:GOTO 210
260   POKE (BOARD+T),1
265   X=R:Y=C:C=0.0:GOSUB 3100
270   CALLM #5915
280   IF PEEK(#5DF3)=2 THEN PRINT "GEWONNEN!!!";:HW%=HW%+1:GOTO 56
      0
290   IF PEEK(#5DCB)=3 THEN PRINT "UNENTSCHIEDEN ";:DD%=DD%+1:GOTO
       560
300   REM PRINT CHR$(12)
310   CM=PEEK(#5DC9)
320   POKE (BOARD+CM),2
330   CR%=INT((CM-1.0)/10.0)
340   CC%=(CM-10.0*(CR%))-1
350   PRINT "Mein Zug ";CR%;",";CC%
355   SOUND 2 1 10 0 FREQ(800.0):WAIT TIME 20:SOUND OFF
360   X=CR%+1.0:Y=CC%+1.0:FOR I=0.0 TO 3.0
362   C=15.0:GOSUB 3100:WAIT TIME 10:C=9.0:GOSUB 3100:WAIT TIME 10

364   NEXT I:C=15.0:GOSUB 3100
370   IF PEEK(#5DCB)=1 THEN PRINT :PRINT "ICH HABE GEWONNEN !!!";:
      CW%=CW%+1:GOTO 560
380   POKE (#5DF4),1
390   GOTO 210
560   PRINT TAB(38);HW%
562   PRINT TAB(38);"VERLOREN  :";CW%
564   PRINT TAB(38);"UNENDSCH. :";DD%
570   INPUT "Wollen Sie noch einmal spielen";A$:A$=LEFT$(A$,1):PRI
```

SP 2

```
      NT
580   IF A$="Y" OR A$="O" OR A$="J" THEN 160
590   END
1000  C=9.0:COLORG 0 15 9 4:MODE 4A:Y=-1.0:A=0.0
1005  FOR X=31.0 TO 121.0 STEP 9.0
1010  A=A+1.0:ON A GOSUB 2000,1100,1200,1300,1400,1500,1600,1700,1
      800,1900
1020  NEXT X
1030  X=20.0:A=11.0:FOR Y=91.0 TO 10.0 STEP -9.0
1040  A=A-1.0
1060  ON A GOSUB 2000,1100,1200,1300,1400,1500,1600,1700,1800,1900

1070  NEXT Y
1080  FILL 30,10 120,100 9
1085  FOR A=19.0 TO 91.0 STEP 9.0:DRAW 30,A 120,A 4:NEXT A
1090  FOR A=39.0 TO 111.0 STEP 9.0:DRAW A,10 A,100 4:NEXT A:RETURN

1100  DRAW X+3,Y+2 X+5,Y+2 C
1101  DRAW X+4,Y+3 X+4,Y+8 C
1102  DRAW X+3,Y+7 X+2,Y+6 C:RETURN
1200  DRAW X+2,Y+6 X+2,Y+7 C
1201  DRAW X+3,Y+8 X+4,Y+8 C
1202  DRAW X+5,Y+7 X+5,Y+6 C
1203  DRAW X+4,Y+5 X+2,Y+3 C
1204  DRAW X+2,Y+2 X+5,Y+2 C:RETURN
1300  DRAW X+2,Y+7 X+3,Y+8 C
1301  DRAW X+4,Y+8 X+5,Y+7 C
1303  DRAW X+5,Y+6 X+4,Y+5 C
1304  DRAW X+5,Y+4 X+5,Y+3 C
1305  DRAW X+4,Y+2 X+3,Y+2 C
1306  DOT X+2,Y+3 C:RETURN
1400  DRAW X+1,Y+5 X+4,Y+8 C
1401  DRAW X+4,Y+8 X+4,Y+2 C
1402  DRAW X+1,Y+4 X+5,Y+4 C:RETURN
1500  DRAW X+2,Y+8 X+5,Y+8 C
1501  DRAW X+2,Y+7 X+2,Y+6 C
1502  DRAW X+3,Y+5 X+4,Y+5 C
1503  DRAW X+5,Y+4 X+5,Y+3 C
1504  DRAW X+4,Y+2 X+3,Y+2 C
1505  DOT X+2,Y+3 C:RETURN
1600  DRAW X+3,Y+8 X+5,Y+8 C
1601  DRAW X+2,Y+7 X+2,Y+6 C
1602  DRAW X+3,Y+5 X+4,Y+5 C
1603  DRAW X+5,Y+4 X+5,Y+3 C
1604  DRAW X+4,Y+2 X+3,Y+2 C
1605  DRAW X+2,Y+3 X+2,Y+4 C:RETURN
1700  DRAW X+2,Y+8 X+5,Y+8 C
1701  DRAW X+5,Y+7 X+3,Y+5 C
1702  DRAW X+3,Y+4 X+2,Y+3 C
1703  DOT X+2,Y+2 C:RETURN
1800  DRAW X+3,Y+8 X+4,Y+8 C
1801  DRAW X+5,Y+7 X+5,Y+6 C
1802  DRAW X+4,Y+5 X+3,Y+5 C
1803  DRAW X+2,Y+4 X+2,Y+3 C
1804  DRAW X+3,Y+2 X+4,Y+2 C
1805  DRAW X+5,Y+3 X+5,Y+4 C
1806  DRAW X+2,Y+6 X+2,Y+7 C:RETURN
1900  DRAW X+2,Y+7 X+3,Y+8 C
1901  DRAW X+4,Y+8 X+5,Y+7 C
1902  DRAW X+5,Y+6 X+4,Y+5 C
1903  DRAW X+3,Y+5 X+2,Y+6 C
1904  DRAW X+5,Y+4 X+5,Y+3 C
1905  DRAW X+4,Y+2 X+2,Y+2 C:RETURN
2000  DRAW X+2,Y+8 X+4,Y+8 C
```

```
2001  DRAW X+5,Y+7 X+5,Y+3 C
2002  DRAW X+4,Y+2 X+2,Y+2 C
2003  DRAW X+1,Y+3 X+1,Y+7 C
2004  DRAW X+2,Y+4 X+4,Y+6 C:RETURN
2999  END
3000  FOR A=0.0 TO 10.0:SOUND 1 0 10 0 FREQ(1000.0)
3010  SOUND 1 0 12 2 FREQ(500.0):WAIT TIME 10:NEXT
3020  SOUND OFF :RETURN
3100  FILL 22+(X*9),2+(Y*9) 29+(X*9),9+(Y*9) C:RETURN
31000 DATA 245,197,213,229
31001 DATA 22,0,33,1,93,6,100,114,35,5,194,32,89,58,244,93,254,0,1
      94,52,89,33,55,93,22,139,114
31002 DATA 6,0,4,14,0,12,62,0,50,202,93,120,214
31041 DATA 1,22,9,95,131,21,194,69,89,129,50,245,93,121,254,3,250,
      100,89,254,9,242,100,89,62
31042 DATA 1,50,202,93,50,201,93,205,165,89,120,254,3,250,126
31081 DATA 89,254,9,242,126,89,33,202,93,62,1,134,119,62,10,50,201
      ,93,205,165,89,58,202,93
31082 DATA 254,2,250,150,89,62,11,50,201,93,205,165,89,62,9,50
31121 DATA 201,93,205,165,89,121,254,10,250,57,89,120,254,10,250,5
      4,89,195,133,90,245,197,62,0,50
31122 DATA 205,93,50,206,93,50,207,93,6,253,4,58,201,93,79
31161 DATA 62,0,128,13,194,187,89,79,58,245,93,129,79,33,44,94,35,
      13,194,201,89,126,198,1,79
31162 DATA 33,204,93,35,13,194,213,89,126,198,1,119,120,254,2
31201 DATA 194,180,89,58,206,93,254,0,202,247,89,58,207,93,254,0,2
      02,247,89,193,241,201,58,206
31202 DATA 93,254,5,250,4,90,62,2,50,243,93,58,207,93,254,4
31241 DATA 194,17,90,62,1,50,203,93,58,206,93,254,0,194,48,90,58,2
      07,93,79,62,1,129,79,33,197
31242 DATA 90,35,13,194,36,90,126,50,236,93,195,68,90,58
31281 DATA 206,93,79,62,1,129,79,33,192,90,35,13,194,59,90,126,50,
      236,93,6,253,4,58,201,93,79
31282 DATA 62,0,128,13,194,77,90,79,58,245,93,129,79,50
31321 DATA 238,93,33,44,94,35,13,194,94,90,126,254,0,194,124,90,58
      ,238,93,79,33,0,93,35,13
31322 DATA 194,112,90,126,79,58,236,93,129,119,120,254,2,194,70
31361 DATA 90,193,241,201,62,1,50,201,93,58,1,93,50,240,93,33,1,93
      ,6,1,4,35,58,240,93,78,145
31362 DATA 242,167,90,121,50,240,93,120,50,201,93,62,100
31401 DATA 184,194,149,90,58,240,93,254,0,194,186,90,62,3,50,203,9
      3,225,209,193,241,201,0
31420 DATA 0,1,1,3,9,36,1,1,3,12,40
31430 REM 0 ON SCREEN =10.
```

```
1       REM
2       REM
3       REM Harald Koegler
4       REM
5       REM
10      CLEAR 1000
20      DIM WERT(3.0,31.0):DIM M(12.0)
30      HI=8.0:REM Hintergrund = grau
40      GE=5.0:REM geistig     = gruen
50      PH=0.0:REM physisch    = schwarz
60      SE=3.0:REM seelisch    = rot
70      COLORT HI PH GE SE
80      COLORG HI GE PH SE
100     GOSUB 1000:REM Geburtsdatum einholen
110     GOSUB 2000:A=S:REM Anzahl der Tage berechnen
120     GOSUB 3000:REM gewuenschtes Jahr und Monat einholen
130     GOSUB 4000:REM Werte berechnen
140     GOSUB 5000:REM Werte graphisch ausgeben
150     GOSUB 6000:REM Werte alphanumerisch ausgeben
160     GOTO 120
1000    MODE 0
1010    PRINT CHR$(12):PRINT "Bitte Geburtsdatum eingeben.":PR
        INT
1020    INPUT "Tag     :";TAG:PRINT
1030    INPUT "Monat   :";MONAT:PRINT
1040    INPUT "Jahr    :";JAHR:PRINT
1050    RETURN
2000    RESTORE:Z=0.0
2010    IF JAHR/4.0<>INT(JAHR/4.0) THEN 2040
2020    IF MONAT<=2.0 THEN 2040
2030    Z=1.0
2040    Z=Z+(JAHR-1.0)*365.0+INT((JAHR-1.0)/4.0)
2050    FOR K=1.0 TO 12.0:READ M(K):NEXT K
2060    S=Z+M(MONAT)+TAG
2070    R=S-INT(S/7.0)*7.0
2080    DATA 0,31,59,90,120,151,181,212,243,273,304,334,5,0,3
2090    RETURN
3000    PRINT :PRINT "Monat und Jahr fuer die Prognose eingebe
        n.":PRINT
3010    INPUT "Monat   :";MONAT:PRINT
3020    INPUT "Jahr    :";JAHR:PRINT
3030    RETURN
4000    REM Werte berechnen
4010    IF MONAT=1.0 OR MONAT=3.0 OR MONAT=5.0 OR MONAT=7.0 OR
         MONAT=8.0 OR MONAT=10.0 OR MONAT=12.0 THEN TAGE=31.0:
        GOTO 4100
4020    IF MONAT=2.0 THEN TAGE=28.0:GOTO 4100
4030    TAGE=30.0
4100    FOR TAG=1.0 TO TAGE
4110    GOSUB 2000:B=S
4120    L=33.0:REM berechnen des geistigen Zustandes
4130    GOSUB 4900
4140    WERT(1.0,TAG)=N
4200    L=23.0:REM physisch
4210    GOSUB 4900
```

```
4220  WERT(2,0,TAG)=N
4300  L=28.0:REM seelisch
4310  GOSUB 4900
4320  WERT(3,0,TAG)=N
4400  NEXT TAG
4410  RETURN
4899  REM Wert des einzelnen Tages bestimmen
4900  N=100.0*SIN(2.0*PI*(B-A)/L):RETURN
5000  MODE 6:MODE 2A
5010  DRAW 0,28 XMAX,28 0
5020  DRAW 0,2 2,2 0:DRAW 0,51 2,51 0:DRAW 1,52 1,50 0
5030  FOR I=1.0 TO 3.0
5040  READ ZYKLUS
5050  FOR J=1.0 TO TAGE
5060  DOT J*2+5,28+(WERT(I,J))/4.0 ZYKLUS
5070  NEXT J
5080  NEXT I
5090  RETURN
6000  GOSUB 6800
6010  I%=1:GOSUB 6700
6020  GOSUB 6900
6030  IF EIN$="U" THEN GOSUB 6100
6040  IF EIN$="D" THEN GOSUB 6300
6050  IF EIN$="M" THEN GOSUB 6800
6060  IF EIN$="N" THEN RETURN
6070  GOTO 6020
6100  I%=I%-1
6110  IF I%<1 THEN I%=1
6120  GOSUB 6700
6130  RETURN
6300  I%=I%+1
6310  IF I%>TAGE THEN I%=TAGE
6320  GOSUB 6700
6330  RETURN
6700  DRAW 0,0 XMAX,0 8
6710  DOT I%*2+5,0 0
6720  G%=WERT(1.0,I%):P%=WERT(2.0,I%):S%=WERT(3.0,I%):MONAT%
      =MONAT:JAHR%=JAHR
6730  PRINT I%;".";MONAT%;".";JAHR%;TAB(15);"geist.=";G%;TAB
      (30);"phys.=";P%;TAB(45);"seel.=";S%
6790  RETURN
6800  PRINT "U = UP        D = DOWN     M = MENUE    N = NEU"
6820  RETURN
6900  EIN$=CHR$(GETC)
6910  IF EIN$="" THEN 6900
6920  RETURN
```

Ein kleines Spiel von J.M.Fischer

```
20    PRINT CHR$(12):MODE 0:COLORT 8 0 0 0:ENVELOPE 1 4,4;15
      ,5;1:ENVELOPE 0 15,10;0,10;15
30    PRINT "                    Spielregel !":PRINT " Bei dies
      en Spiel lenkst Du mit der CURSOR-STEUERUNG eine"
40    PRINT " bewegte Linie . Diese Linie wird im laufe des
      Spieles":PRINT " immer schneller . Druecks Du die SPAC
      E-TASTE,dann ver="
50    PRINT " setzt sich Deine Linie zufaellig .  Du darfst
      dich mit"
60    PRINT " dieser Linie nur auf der dunklen Flaechen bewe
      gen ,kommst":PRINT " Du gegen eine farbige Linie ist d
      as Spiel beendet."
70    PRINT "                  W A R N U N G !":PRINT " In der
      naehe von farbigen Linien wird Deine Linie"
80    PRINT " gelegendlich unsichtbar . Es tauchen ploetzlic
      h neue Linien"
90    PRINT " auf und verschwinden wieder."
100   PRINT " Anzahl der farbigen Linien eingeben und die RE
      TURN-Taste "
110   INPUT " druecken.";A$:PRINT " Ein neues Spiel kann mit
       der TAB-Taste":PRINT " gestartet werden.Viel Spass !
      ; Bitte TAB-Taste druecken ."
120   A=GETC:IF A<>9 GOTO 120
130   GOSUB 350:A=19:X=10:Y=64:C=30:D=0:FILL 5,59 15,69 0:DO
      T X,Y 15:WAIT TIME 100
140   FOR T=0 TO 15:WAIT TIME C:B=GETC:SOUND OFF :IF (B>15)
      AND (B<20) THEN A=B
150   IF A=16 THEN Y=Y+1
160   IF A=17 THEN Y=Y-1
170   IF A=18 THEN X=X-1
180   IF A=19 THEN X=X+1
190   IF B=32 THEN X=RND(XMAX):Y=RND(YMAX):GOTO 250
200   IF X>XMAX THEN X=0:GOTO 220
210   IF X<0 THEN X=XMAX
220   IF Y>YMAX THEN Y=0:GOTO 240
230   IF Y<0 THEN Y=YMAX
240   IF SCRN(X,Y)<>0 GOTO 290
250   DOT X,Y 0:SOUND 1 0 9 0 FREQ(200):DOT X,Y 15
260   NEXT:DRAW X1,Y1 X2,Y2 0:X1=INT(RND(XMAX)):Y1=INT(RND(Y
      MAX)):X2=INT(RND(XMAX))
270   Y2=INT(RND(YMAX)):X4=RND(15):DRAW X1,Y1 X2,Y2 X4:D=D+1
      :IF C>0 THEN C=C-5:GOTO 140
280   C=0:GOTO 140
290   FOR A=0 TO 10:SOUND 0 0 15 0 FREQ(800):SOUND 0 0 12 2
      FREQ(400)
300   D1=D*(1+A1):PRINT "Du hast ";D1;" Punkte erreicht !!!"
      :NEXT:SOUND OFF :IF D1>H THEN H=D1:GOTO 320
310   PRINT "Hoechste Punktzahl ";H;" hat ";B$;" erreicht !!
      !"
320   IF D1=H THEN INPUT "Du kannst Deinen Namen eintragen !
      !";B$:PRINT :PRINT "Weiter mit TAB."
330   B=GETC:IF B=9 THEN MODE 0:PRINT CHR$(12):GOTO 100
340   GOTO 330
350   MODE 3:IF (48.0>ASC(A$)) OR (57<ASC(A$)) THEN RETURN
360   A1=VAL(A$):FOR I=1 TO A1:A=RND(XMAX):B=RND(YMAX):C=RND
      (XMAX):D=RND(YMAX)
370   E=RND(15):IF E=8 GOTO 370
380   DRAW A,B C,D E:NEXT:RETURN
```

SP 4

```
1     REM KUGELLABYRINTH VON NILS KAY 14.11.82
2     REM DAS SPIEL LÄUFT MIT EIGENEN PADDLES:
3     REM PDL(0)=X PDL(1)=Y (KREUZPOTI)
4     REM DAS PROGRAMM IST GGF. AN ANDERE PADDLES ANZUPASSEN.
5     REM ZIEL IST ES DIE KUGEL BIS ZUM AUSGANG RECHTS UNTEN ZU STEUERN
6     REM DIE NEIGUNG DER SPIELEBENE IN X ZEIGT DER OBERE BALKEN,
7     REM IN Y DER UNTERE.
8     REM ÜBER LEICHT,MITTEL,SCHWER WERDEN REIBUNG UND DICHTE DER KUGEL BE
      EINFLUSST.
9     PDLX%=0:PDLY%=1
10    MODE 0:PRINT CHR$(12):PRINT :PRINT "DIES IST DAS KUGELSPIEL."
14    PRINT :PRINT :PRINT :PRINT "SXCHWIERIGKEITSGRAD:-LEICHT"
16    PRINT "                    -MITTEL"
17    PRINT "                    -SCHWER"
18    PRINT :PRINT :INPUT "WELCHEN SCHWIERIGKEITSGRAD WUENSCHEN SIE ";A$
19    IF A$="" THEN 24
20    A$=LEFT$(A$,1)
21    IF A$="L" THEN SCH=0.6:JH=60.0:GOTO 30
22    IF A$="M" THEN SCH=0.8:JH=50.0:GOTO 30
23    IF A$="S" THEN SCH=0.95:JH=40.0:GOTO 30
24    PRINT :PRINT "FALSCHE EINGABE. NOCH EINMAL!":GOTO 18
30    MODE 1:MODE 4:COLORG 0 9 5 14
32    X%=33:Y%=4:A1=2.0:B1=2.0:KUG%=8:DX=4.0:FL%=0:DY=4.0:FY%=90:FR%=FY%:F
      G%=40:FH%=40:GOSUB 1000:GOSUB 1050
35    DRAW 29,1 XMAX,1 23:DRAW XMAX,1 XMAX,YMAX 23
36    DRAW XMAX,YMAX 29,YMAX 23:DRAW 29,YMAX 29,1 23
37    RESTORE:GOSUB 1100
40    A1=A:B1=B
42    A=(127.0-PDL(PDLX%))/JH:B=(127.0-PDL(PDLY%))/JH:CB%=(PEEK(#FD00) IAN
      D #20) SHR 5
43    IF CB%=1 THEN A=A*5.0:B=B*5.0
44    IF ABS(A-A1)<0.1 THEN A=A1
46    IF ABS(B-B1)<0.1 THEN B=B1
47    X1%=X%:Y1%=Y%:DX1=DX:DY1=DY
48    IF A<>A1 THEN GOSUB 1000
49    IF B<>B1 THEN GOSUB 1050
51    REM BEGINN VON KUGELBEWEGUNG
60    DX=DX1+A:DY=DY1+B
70    DX=DX*SCH:DY=DY*SCH
75    GOSUB 3000:IF FL%=1.0 THEN 300
77    X%=XX%:Y%=YY%:FILL X1%,Y1% X1%+1,Y1%+1 20:FILL X%,Y% X%+1,Y%+1 21
80    IF X%>148.0 AND X%<XMAX AND Y%<9.0 THEN 500
100   GOTO 40
110   REM KUGEL IST AUF MAUER ODER LOCH GESTOSSEN
300   REM KUGEL IN LOCH
310   FILL XX1%,YY1% XX1%+1,YY1%+1 21:FILL X%,Y% X%+1,Y%+1 20
312   COLORG 0 10 14 5
315   T1%=2:T2%=1:FOR O%=1 TO 15:GOSUB 600
317   IF INT(RND(10.0))=1.0 THEN COLORG RND(16.0) RND(16.0) RND(16.0) RND(
      16.0)
318   NEXT
319   COLORG 0 9 5 14
320   SOUND OFF :FL%=0:PRINT "SIE SIND IN EIN LOCH GEFALLEN!"
330   KUG%=KUG%-1:PRINT "SIE HABEN NOCH";KUG%;" KUGELN UEBER."
340   IF KUG%<=0.0 THEN 380
350   FILL XX1%,YY1% XX1%+1,YY1%+1 20
360   WAIT TIME 100:X%=34:Y%=4:DX=4.0:DY=4.0:A=2.0:B=2.0:MODE 4:GOTO 40
380   WAIT TIME 100:MODE 0:PRINT :PRINT :PRINT
383   PRINT :PRINT "DER SCHWIERIGKEITSGRAD WAR ";A$;" ."
385   PRINT :PRINT " WOLLEN SIE NOCH EINMAL SPIELEN?"
390   A7%=GETC:IF A7%=0 THEN 390
400   IF A7%<>78 THEN 10
410   END
500   T1%=0:T2%=0
510   FOR O%=0 TO 80:GOSUB 600:NEXT:SOUND OFF
520   MODE 0:PRINT :PRINT :PRINT "SIE HABEN GEWONNEN!!"
```

```
530    PRINT :PRINT "ES SIND NOCH";KUG%;" KUGELN UEBER.":GOTO 380
600    SOUND 1 0 15 T1% FREQ(PEEK(#FC02)+31+PEEK(#FC03)*256)
610    SOUND 2 0 15 T2% FREQ(PEEK(#FC04)+31+PEEK(#FC05)*256):RETURN
1000   REM ANZEIGE VON EBENE
1010   FY1%=FY%:FR1%=FR%:FY%=89+A*2:FR%=91-A*2
1020   DRAW 1,FY1% 27,FR1% 20:DRAW 1,FY% 27,FR% 22:RETURN
1050   FG1%=FG%:FH1%=FH%:FG%=39+B*2:FH%=41-B*2
1060   DRAW 1,FG1% 27,FH1% 20:DRAW 1,FG% 27,FH% 21:RETURN
1100   READ K%:IF K%=999 THEN 1120
1110   READ L%,M%,N%:FILL K%,L% M%+1,N%+1 22:GOTO 1100
1120   READ K%,L%:IF K%<>888 THEN FILL K%,L% K%+4,L%+4 14:GOTO 1120
1130   RETURN
2000   DATA 38,2,38,40,38,40,43,40,43,40,43,75,43,75,38,75,38,75,38,118
2010   DATA 46,120,46,95,46,95,65,95,55,95,55,80,55,80,68,80
2020   DATA 52,75,52,65,55,60,55,10,65,2,65,11,65,11,80,11,68,18,68,70,63,7
       0,77,70
2030   DATA 77,70,77,110,60,113,60,118,60,110,130,110,88,15,110,15,100,15,1
       00,100
2040   DATA 115,127,115,115,115,110,115,97,115,85,115,65,115,55,144,55,120,
       55,120,2
2050   DATA 135,10,135,42,130,110,130,96,130,105,140,105,140,105,140,112,14
       0,112,152,112
2060   DATA 152,112,152,95,152,95,142,95,100,86,158,86,150,75,150,55,155,45
       ,137,45
2070   DATA 148,2,148,36,999
2100   DATA 30,30,35,42,33,64,33,97,30,118,42,10,50,18,43,26,42,51,53,60
2110   DATA 50,75,38,89,48,97,54,107,45,122,62,85,66,12,60,3,65,103,82,4,80
       ,25
2120   DATA 90,33,78,50,71,40,95,55,87,68,80,88,90,93,92,112,108,118,104,10
       5,112,3
2130   DATA 103,28,114,37,105,61,120,56,113,85,120,98,126,120,133,108,132,8
       8,123,80
2140   DATA 127,67,133,77,133,42,125,2,139,123,147,115,153,84,138,66,144,54
       ,153,45,130,25,143,28
2150   DATA 153,26,147,10,888,888
3000   REM LINIE ZWISCHEN DEN STEP'S VERGLEICHEN KOMMT MIT X,Y DX! UND DY!
3010   C=SQR(DX*DX+DY*DY):IF C<1.0 THEN RETURN
3013   CO=DX/C:SI=DY/C
3015   FOR O=1.1 TO C:XX1%=XX%:YY1%=YY%:XX%=X%+O*CO:YY%=Y%+O*SI
3030   IF XX%<30.0 OR XX%>156.0 THEN XX%=XX1%:DX=-DX/2.0
3040   IF YY%<3.0 OR YY%>126.0 THEN YY%=YY1%:DY=-DY/2.0
3045   S1%=SGN(DX):S2%=SGN(DY)
3046   F%=SCRN(XX%,YY%):F1%=SCRN(XX%+1,YY%):IF S2%<0 AND F%=5 OR F1%=5 THEN
        3100
3047   F%=SCRN(XX%,YY%+1):F1%=SCRN(XX%+1,YY%+1):IF S2%>0 AND F%=5 OR F1%=5
       THEN 3110
3048   F%=SCRN(XX%,YY%):F1%=SCRN(XX%,YY%+1):IF S1%<0 AND F%=5 OR F1%=5 THEN
        3120
3049   F%=SCRN(XX%+1,YY%):F1%=SCRN(XX%+1,YY%+1):IF S1%>0 AND F%=5 OR F1%=5
       THEN 3130
3060   IF F%=14 OR F1%=14 THEN FL%=1:GOTO 3080
3070   FLAG%=0:NEXT
3080   RETURN
3100   IF SCRN(XX1%+S1%,YY1%+S2%)=5 OR SCRN(XX1%+S1%+1,YY1%+S2%)=5 THEN XX%
       =XX1%:YY%=YY1%:DY=-DY/3.0:GOTO 3150
3110   IF SCRN(XX1%+S1%,YY1%+1+S2%)=5 OR SCRN(XX1%+1+S1%,YY1%+S2%+1)=5 THEN
        XX%=XX1%:YY%=YY1%:DY=-DY/3.0:GOTO 3150
3120   IF SCRN(XX1%+S1%,YY1%+S2%)=5 OR SCRN(XX1%+S1%,YY1%+1+S2%)=5 THEN YY%
       =YY1%:XX%=XX1%:DX=-DX/3.0:GOTO 3160
3130   IF SCRN(XX1%+1+S1%,YY1%+S2%)=5 OR SCRN(XX1%+1+S1%,YY1%+1+S2%)=5 THEN
        YY%=YY1%:XX%=XX1%:DX=-DX/3.0:GOTO 3160
3140   XX%=XX1%:YY%=YY1%:DY=-DY/3.0:DX=-DX/3.0
3150   IF FLAG%=1 THEN DX=0.0
3155   FLAG%=1:GOTO 3080
3160   IF FLAG%=1 THEN DY=0.0
3165   FLAG%=1:GOTO 3080
```

Labyrinth:

Wenn die Umrandung und eine zufällige Verteilung von grauen "Wand"-Zeichen erschienen sind, muß mittels der Cursor-Tasten, der TAB-Taste (zum Setzen) und der CHAR DEL-Taste (zum Löschen von Wand-Zeichen) sichergestellt werden, daß es für den Läufer (Stern) mindestens einen Weg von der linken, unteren zur rechten, oberen Ecke gibt. Dabei empfiehlt es sich, keine zusammenhängenden Flächen frei zu lassen, weil dann der Suchlauf lange dauern kann. Die Cursor-Tasten bewegen ein Wand-Zeichen (das zunächst in der Mitte des Bildes steht) über ein Feld, ohne ohne die bestehende Verteilung zu verändern. TAB setzt ein bleibendes Wand-Zeichen an die Stelle des bewegten Zeichens, und CHAR DEL löscht seinen Platz.

Nun kann der Suchlauf mit S gestartet werden.

Hat der Läufer die rechte, obere Ecke erreicht, verschwindet er.

Dann kann der Ziellauf gestartet werden (mit Z). Hierbei beginnt der Läufer wieder in der linken, unteren Ecke. Er vermeidet jetzt aber die Wege, die beim Suchlauf als Sackgassen erkannt wurden.

Mit N kann man ein neues Labyrinth bekommen.

Prinzip des Programms:

Der Läufer sucht bei seinem Lauf zur rechten, oberen Ecke an Verzweigungsstellen jeweils die andere Richtung einzuschlagen, verglichen mit seiner Herkunftsrichtung. Wenn er nach oben ging, wird er, wenn möglich, nach rechts abbiegen; nach oben, wenn er nach rechts ging. Wenn es nur nach links oder nach unten weitergeht, nimmt er auch diese Richtung. Am Ende einer Sackgasse kehrt er um.

Bei jeder Verzweigungsstelle werden die Platznummer P% (Bildschirm-Speicheradresse), die Herkunftsrichtung HR% und die neue Richtung R% in einer "Ankunft"-Zeichenkette A$(I%) registriert; I% ist dabei die fortlaufend gezählte Anzahl der passierten Verzweigungsstellen. Bei neuen Verzweigungsstellen wird der Läufer in die bevorzugte Richtung geschickt, die von seinem Platz aus zur Verfügung steht. Kommt der Läufer an eine Verzweigungsstelle, an der er schon einmal war, wird er in die nächstmögliche Richtung geschickt.

Bei jeder Verzweigungstelle wird also geprüft, ob sie schon bekannt oder neu ist.

Nach dem Suchlauf sind alle passierten Verzweigungsstellen mit der jeweiligen Herkunftsrichtung und er erfolgreichen Fortsetzungsrichtung bekannt; der Läufer findet daher nun seinen Weg, indem er bei jeder Verzweigungsstelle "Auskunft" darüber bekommt, in welcher Richtung er weiterlaufen muß.

```
1       REM Labyrinth; Suchlauf und Ziellauf
2       REM Peter Buettner
3       REM Offenbacher Str. 6
4       REM 1000 Berlin 33
5       REM Tel.: 8228108
11      PRINT CHR$(12):CLEAR 6000:DIM R%(254.0),A$(254.0)
12      COLORT 8 0 0 0:MODE 0
14      FOR I%=46061.0 TO 48875.0 STEP 134.0
15      POKE I%,30
16      NEXT
17      FOR I%=49009.0 TO 49127.0 STEP 2.0
18      POKE I%,30
19      NEXT
20      FOR I%=1 TO 22
21      CURSOR 0,I%:PRINT CHR$(30)
22      NEXT
23      FOR I%=46063.0 TO 46177.0 STEP 2.0
24      POKE I%,30
25      NEXT
27      POKE 46311,42
30      P%=47597.0:POKE P%,30
35      CURSOR 0,0:I%=1:A$(I%)="000000000"
37      GOSUB 3000
40      A%=GETC
50      IF A%=16.0 THEN 150
60      IF A%=18.0 THEN 180
70      IF A%=17.0 THEN 210
80      IF A%=19.0 THEN 240
90      IF A%=9.0 THEN T%=1.0:TA%=1.0
100     IF A%=8.0 THEN T%=0.0:TA%=1.0
110     IF A%=78 GOTO 1·
130     IF A%=83.0 GOTO 1070
135     IF A%=90 THEN Z%=1:I%=1:GOTO 1070
140     GOTO 40
150     GOSUB 270
160     IF P%>49000 THEN 175
170     P%=P%+134.0
175     GOSUB 276
177     GOTO 40
180     GOSUB 270
190     IF ABS(FRAC((P%-46047.0)/134.0))<5E-2 THEN 205
200     P%=P%+2.0
205     GOSUB 276
207     GOTO 40
210     GOSUB 270
220     IF P%<46047.0 THEN 235
230     P%=P%-134.0
235     GOSUB 276
237     GOTO 40
240     GOSUB 270
250     IF ABS(FRAC((P%-45927.0)/134.0))<1E-2 THEN 265
260     P%=P%-2.0
265     GOSUB 276
267     GOTO 40
270     IF TA%=1.0 THEN 273
271     IF VO%=1.0 THEN POKE P%,30:GOTO 275
272     GOTO 274
273     IF T%=1.0 THEN POKE P%,30:GOTO 275
274     POKE P%,32
275     TA%=0.0:T%=0.0:RETURN
276     IF PEEK(P%)=30 THEN VO%=1.0:GOTO 280
278     VO%=0.0
280     POKE P%,30:RETURN
340     REM HAUPTPROGRAMM ----------
```

SP 6,1

```
1000  POKE P%,32:REM Stern loeschen
1010  RETURN
1060  REM Startermittlung ------------------------------------
1070  P%=46311.0
1080  IF PEEK(P%-2)=32 GOTO 1100
1090  IF PEEK(P%+134)=32 GOTO 1200
1098  REM
1099  REM Nach rechts ----------------------------------------
1100  GOSUB 1000:REM Stern loeschen
1105  P%=P%-2.0:W%=0.0:REM Schritt, Weg
1110  GOSUB 1500:REM Ende?
1115  IF PEEK(P%+134)=32 THEN W%=W%+1.0:R%=2.0:REM Oben frei
1120  IF PEEK(P%-2)=32 THEN W%=W%+1.0:R%=1.0:REM Rechts frei
1125  IF PEEK(P%-134)=32 THEN W%=W%+1.0:R%=4.0:REM Unten frei
1130  IF W%>1.0 THEN HR%=1:GOTO 1600:REM Verzweigung
1135  IF W%=1.0 GOTO 1155:REM Keine Verzweigung
1140  GOTO 1300:REM Sackgasse
1155  ON R% GOTO 1100,1200,1300,1400
1199  REM Nach oben ------------------------------------------
1200  GOSUB 1000
1205  P%=P%+134.0:W%=0.0
1210  GOSUB 1500
1215  IF PEEK(P%-2)=32 THEN W%=W%+1.0:R%=1.0
1220  IF PEEK(P%+134)=32 THEN W%=W%+1.0:R%=2.0
1225  IF PEEK(P%+2)=32 THEN W%=W%+1.0:R%=3.0
1230  IF W%>1.0 THEN HR%=2:GOTO 1600
1235  IF W%=1.0 GOTO 1255
1240  GOTO 1400
1255  ON R% GOTO 1100,1200,1300,1400
1299  REM Nach links -----------------------------------------
1300  GOSUB 1000
1305  P%=P%+2.0:W%=0.0
1310  GOSUB 1500
1315  IF PEEK(P%+134)=32 THEN W%=W%+1.0:R%=2.0
1320  IF PEEK(P%-134)=32 THEN W%=W%+1.0:R%=4.0
1325  IF PEEK(P%+2)=32 THEN W%=W%+1.0:R%=3.0
1330  IF W%>1.0 THEN HR%=3:GOTO 1600
1335  IF W%=1.0 GOTO 1355
1340  GOTO 1100
1355  ON R% GOTO 1100,1200,1300,1400
1399  REM Nach unten -----------------------------------------
1400  GOSUB 1000
1405  P%=P%-134.0:W%=0.0
1410  GOSUB 1500
1415  IF PEEK(P%-2)=32 THEN W%=W%+1.0:R%=1.0
1420  IF PEEK(P%+2)=32 THEN W%=W%+1.0:R%=3.0
1425  IF PEEK(P%-134)=32 THEN W%=W%+1.0:R%=4.0
1430  IF W%>1.0 THEN HR%=4:GOTO 1600
1435  IF W%=1.0 GOTO 1455
1440  GOTO 1200
1455  ON R% GOTO 1100,1200,1300,1400
1499  REM Ende? Stern setzen ---------------------------------
1500  IF P%=48877.0 GOTO 40
1510  POKE P%,42
1520  RETURN
1599  REM Auskunft -------------------------------------------
1600  IF Z%=1 GOTO 1980
1605  IF I%=0.0 GOTO 1650
1610  IF P%=VAL(MID$(A$(I%),1,7)) GOTO 1660
1618  I%=L%
1623  IF I%=0 THEN I%=L%:GOTO 1650
1630  IF P%<>VAL(MID$(A$(I%),1,7)) THEN I%=I%-1:GOTO 1623
1640  GOTO 1660
1650  I%=I%+1:P$=STR$(P%):A$(I%)=P$+STR$(HR%)+" 0.0":R%=0:L%=I%
```

SP6,2

```
1660  HR%=VAL(MID$(A$(I%),9,3)):R%=VAL(RIGHT$(A$(I%),3))
1670  ON HR% GOTO 1680,1760,1820,1900
1680  IF R%=0 GOTO 1720
1690  IF R%=2 GOTO 1730
1700  IF R%=1 GOTO 1740
1710  IF R%=4 GOTO 1750
1720  IF PEEK(P%+134)=32 THEN A$(I%)=LEFT$(A$(I%),12)+" 2.0":GOTO 1200

1730  IF PEEK(P%-2)=32 THEN A$(I%)=LEFT$(A$(I%),12)+" 1.0":GOTO 1100
1740  IF PEEK(P%-134)=32 THEN A$(I%)=LEFT$(A$(I%),12)+" 4.0":GOTO 1400

1750  A$(I%)=LEFT$(A$(I%),12)+" 3.0":GOTO 1300
1760  IF R%=0 GOTO 1780
1770  ON R% GOTO 1790,1800,1810
1780  IF PEEK(P%-2)=32 THEN A$(I%)=LEFT$(A$(I%),12)+" 1.0":GOTO 1100
1790  IF PEEK(P%+134)=32 THEN A$(I%)=LEFT$(A$(I%),12)+" 2.0":GOTO 1200

1800  IF PEEK(P%+2)=32 THEN A$(I%)=LEFT$(A$(I%),12)+" 3.0":GOTO 1300
1810  A$(I%)=LEFT$(A$(I%),12)+" 4.0":GOTO 1400
1820  IF R%=0 GOTO 1860
1830  IF R%=2 GOTO 1870
1840  IF R%=4 GOTO 1880
1850  IF R%=3 GOTO 1890
1860  IF PEEK(P%+134)=32 THEN A$(I%)=LEFT$(A$(I%),12)+" 2.0":GOTO 1200

1870  IF PEEK(P%-134)=32 THEN A$(I%)=LEFT$(A$(I%),12)+" 4.0":GOTO 1400

1880  IF PEEK(P%+2)=32 THEN A$(I%)=LEFT$(A$(I%),12)+" 3.0":GOTO 1300
1890  A$(I%)=LEFT$(A$(I%),12)+" 1.0":GOTO 1100
1900  IF R%=0 GOTO 1940
1910  IF R%=1 GOTO 1950
1920  IF R%=3 GOTO 1960
1930  IF R%=4 GOTO 1970
1940  IF PEEK(P%-2)=32 THEN A$(I%)=LEFT$(A$(I%),12)+" 1.0":GOTO 1100
1950  IF PEEK(P%+2)=32 THEN A$(I%)=LEFT$(A$(I%),12)+" 3.0":GOTO 1300
1960  IF PEEK(P%-134)=32 THEN A$(I%)=LEFT$(A$(I%),12)+" 4.0":GOTO 1400

1970  A$(I%)=LEFT$(A$(I%),12)+" 2.0":GOTO 1200
1979  REM Verzw.-Richtung Bei Ziellauf ----------------
1980  IF P%=VAL(MID$(A$(I%),1,7)) THEN R%=VAL(RIGHT$(A$(I%),3)):GOTO 2
      000
1990  IF I%<L% THEN I%=I%+1:GOTO 1980
1995  I%=1:GOTO 1980
2000  ON R% GOTO 1100,1200,1300,1400
2010  END
3000  REM Zufalls-Labyrinth
3005  FOR M%=2 TO 22:CURSOR 2,M%
3010  FOR N%=1 TO 57
3020  X=RND(2.0):IF X>1.2 THEN PRINT CHR$(30);:GOTO 3030
3025  PRINT CHR$(32);
3030  NEXT
3040  PRINT
3050  NEXT
3060  RETURN
```

S I M P L E B A L L

Simple Ball, ein kleines Spiel in MODE 2 (=BREAKOUT (Red.))
In Zeile 400 werden PDL 1 und 2 angesprochen, das Spielen ist
allerdings auch mittels der Cursortasten möglich.

Unter dem Spielfeld werden 3 Zahlen angezeigt:
Die erste ist ein Zähler für die Ballbewegung (damit Spieldauer),
sie sollte 1000-2000 betragen,
die zweite zeigt den Punktestand: bei Spielbeginn 10 Punkte Bonus
je getroffenes Geld gibt es 1 Punkt (70 Felder+10 Bonus = 80 Punkte
max.), je verlorenem Ball werden 2 Punkte abgezogen,
die dritte bildet den Quotienten aus 2.Zahl*1000/1.Zahl,
sie sollte 35-70 betragen (über 70 ausgezeichnet).

IMP INT

```
1     REM ******************************************************
2     REM *            S I M P L E   B A L L                   *
3     REM *                     von                            *
4     REM *                Ralf Degenhardt              3/84 *
5     REM ******************************************************
10    CLEAR 250
20    A=5+RND(61.0):B=31
21    K=1+RND(2.0):L=1
22    I=29:J=2
23    V=1:W=10
30    COLORT 1 14 1 1:COLORG 1 5 11 14:MODE 0:MODE 2A
31    PRINT CHR$(12):POKE #75,32
40    DRAW 0,0 70,0 21
41    DRAW 0,0 0,52 21
42    DRAW 70,0 70,52 21
43    DRAW 0,52 70,52 21
50    FOR M=1 TO 2
51    FOR N=1.0 TO 66.0 STEP 5.0
52    FILL N,45 N+3,46 21+RND(3.0)
53    FILL N,42 N+3,43 21+RND(3.0)
54    FILL N,39 N+3,40 21+RND(3.0)
55    FILL N,36 N+3,37 21+RND(3.0)
56    FILL N,33 N+3,34 21+RND(3.0)
57    NEXT N
58    NEXT M
60    PRINT CHR$(12)
61    PRINT "     Steuerung mittels Cursortasten : <T>"
62    PRINT "     ......... ....... Kreuzregler  : <R>"
63    PRINT "     ......... ....... Autofunktion : <A>"
70    Q1=GETC
71    IF Q1=84 THEN Q=1:GOTO 78
72    IF Q1=82 THEN Q=2:GOTO 78
73    IF Q1=65 THEN Q=3:GOTO 78
74    GOTO 70
78    PRINT CHR$(12):POKE #BAE7,#57:CURSOR 15,3:PRINT CHR$(Q1)
80    GOSUB 307:GOSUB 208:WAIT TIME 100
81    ON Q GOSUB 300,400,500
82    IF SCRN(A,B-1)<>1 AND B>1 THEN 92
83    IF B=1 THEN W=W-2:GOSUB 208:GOSUB 140:A=5+RND(61.0):B=31:GOTO 81
84    GOSUB 140:B=B-1:GOSUB 150:GOTO 81
90    IF B=1 THEN 83
91    ON Q GOSUB 300,400,500
92    ON K GOSUB 100,110,120,130
93    ON L GOTO 90,230
```

```
100   IF SCRN(A+1,B)<>1 THEN C=A+1:D=B:GOSUB 200:K=2:RETURN
101   IF SCRN(A,B+1)<>1 THEN C=A:D=B+1:GOSUB 200:K=4:RETURN
102   IF SCRN(A+1,B+1)<>1 THEN C=A+1:D=B+1:GOSUB 200:K=3:RETURN
103   GOSUB 140:A=A+1:B=B+1:GOSUB 150:RETURN
110   IF SCRN(A-1,B)<>1 THEN C=A-1:D=B:GOSUB 200:K=1:RETURN
111   IF SCRN(A,B+1)<>1 THEN C=A:D=B+1:GOSUB 200:K=3:RETURN
112   IF SCRN(A-1,B+1)<>1 THEN C=A-1:D=B+1:GOSUB 200:K=4:RETURN
113   GOSUB 140:A=A-1:B=B+1:GOSUB 150:RETURN
120   IF SCRN(A-1,B)<>1 THEN C=A-1:D=B:GOSUB 200:K=4:RETURN
121   IF SCRN(A,B-1)<>1 THEN C=A:D=B-1:GOSUB 200:K=2:RETURN
122   IF SCRN(A-1,B-1)<>1 THEN C=A-1:D=B-1:GOSUB 200:K=1:RETURN
123   GOSUB 140:A=A-1:B=B-1:GOSUB 150:RETURN
130   IF SCRN(A+1,B)<>1 THEN C=A+1:D=B:GOSUB 200:K=3:RETURN
131   IF SCRN(A,B-1)<>1 THEN C=A:D=B-1:GOSUB 200:K=1:RETURN
132   IF SCRN(A+1,B-1)<>1 THEN C=A+1:D=B-1:GOSUB 200:K=2:RETURN
133   GOSUB 140:A=A+1:B=B-1:GOSUB 150:RETURN
140   DOT A,B 20:RETURN
150   DOT A,B 23:V=V+1:RETURN
200   IF C<1.0 OR C>69.0 OR D<33.0 OR D>46.0 THEN FILL I,J I+3,J+1 21:RET
URN
201   E=C/5*5+1:G=E
202   IF E=A THEN G=G+1
203   IF SCRN(G,D+1)<>1 THEN F=D+1:GOTO 205
204   IF SCRN(G,D-1)<>1 THEN F=D-1
205   FILL E,D E+3,F 20
206   W=W+1
207   U=U+1:IF U=70 THEN L=2
208   CURSOR 8,3:PRINT "        "
209   R=1000*W/V
210   CURSOR 0,3:PRINT V;" ";W;" ";R
220   RETURN
230   IF GETC=0.0 THEN 230
231   GOTO 10
300   Z=GETC
301   IF Z=0 THEN RETURN
302   FILL I,J I+3,J+1 20
303   IF Z=16 THEN J=J+2:IF J>29 THEN J=29:GOTO 307
304   IF Z=17 THEN J=J-2:IF J<1 THEN J=1:GOTO 307
305   IF Z=19 THEN I=I+3:IF I>66 THEN I=66:GOTO 307
306   IF Z=18 THEN I=I-3:IF I<1 THEN I=1
307   FILL I,J I+3,J+1 21
308   RETURN
400   M1=PDL(1):N1=PDL(2)
401   IF ABS(M1-M2)<12.0 AND ABS(N1-N2)<18.0 THEN RETURN
402   FILL I,J I+3,J+1 20
403   I=2+3*M1/12:J=1+2*N1/18
404   FILL I,J I+3,J+1 21
405   M2=M1:N2=N1
410   RETURN
500   IF B>30 THEN RETURN
501   FILL I,J I+3,J+1 20
502   I=A-1.5+RND(3.0):J=28
503   IF I>66.0 THEN I=66
504   IF I<1 THEN I=1
505   FILL I,J I+3,J+1 21
506   RETURN
```

```
10    REM ..................................................
20    REM ............. Target by R.Schall 1982 .............
30    REM ..................................................
40    REM ... Mit den Cursortasten sind die beiden Linien ..
50    REM ... so zu lenken, dass sie sich ueber dem Recht-..
60    REM ... eck kreuzen. In dem Moment SPACE-Taste be- ...
70    REM ... taetigen. Das Ziel wird nach jedem Treffer ...
80    REM ... kleiner. .....................................
90    REM ..................................................
100   MODE 2:COLORG 14 2 14 14
110   FOR SIZEX=20 TO 2 STEP -2:MODE 2:DX=1:DY=1:SIZEY=SIZEX*3/4
120   X=RND(XMAX-SIZEX):Y=RND(YMAX-SIZEY)
130   H=(XMAX+1)/2:V=(YMAX+1)/2:GOSUB 500
140   IF RND(SIZEX)<1 THEN DX=-DX
145   IF RND(SIZEX)<1 THEN DY=-DY
150   IF X+DX<0 OR X+DX>XMAX-SIZEX THEN DX=-DX
160   IF Y+DY<0 OR Y+DY>YMAX-SIZEY THEN DY=-DY
170   G=GETC:IF G<16 THEN 180:IF G<18 THEN DV=(16.5-G)*2:GOTO 180
175   IF G<20 THEN DH=(G-18.5)*2
177   IF G=32 AND H-X<SIZEX AND V-Y<SIZEY THEN NEXT SIZEX:GOTO 100
180   H0=H:H=H+DH:IF H<0 OR H>XMAX THEN H=H0
190   V0=V:V=V+DV:IF V<0 OR V>YMAX THEN V=V0
250   X0=X:Y0=Y:X=X+DX:Y=Y+DY:N=1-N
260   GOSUB 500:GOTO 140
500   REM ..... ZEICHEN-SBR
510   COL1=17+2*N:COL2=18-2*N
520   FILL X,Y X+SIZEX,Y+SIZEY COL1:FILL X+1,Y+1 X+SIZEX-1,Y+SIZEY-1 16+2*N
530   DRAW H,0 H,YMAX COL1:DRAW 0,V XMAX,V COL1
540   COLORG 14 2+12*N 2+12*(1-N) 2
550   DRAW H0,0 H0,YMAX COL2:DRAW 0,V0 XMAX,V0 COL2
560   FILL X0,Y0 X0+SIZEX,Y0+SIZEY COL2
570   RETURN
```

<<IMP INT>>

```
10    REM ..................................................
20    REM ..................... WEGWEISER ..................
30    REM ..................................................
40    REM ..................... (c) 1984 by Rolf Schall ....
50    REM ..................................................
100   REM ............... Erklaerung ........................
110   COLORT 8 0 0 0:MODE 0:PRINT CHR$(12)
120   PRINT TAB(16);"Wegweiserspiel V2.0":PRINT
130   PRINT "Aufgabe: Der gelbe Fleck ist durch ein Labyrinth
      von Raeumen"
140   PRINT "unterschiedlicher Farbe zu lenken."
150   PRINT "Folgen Sie mit Hilfe der  Cursortasten den Pfeile
      n  bis  zum"
160   PRINT "durch ein  Kreuz  gekennzeichneten Zielpunkt.  Ve
      rsuchen Sie"
170   PRINT "danach, zum Ausgangsort mit moeglichst wenigen Sc
      hritten zu-"
180   PRINT "rueckzufinden.  Wenn Sie einen Raum verlassen hab
      en,  werden"
190   PRINT "die Wegweiser weggewischt."
200   PRINT "Wenn Sie aufgeben wollen,  druecken Sie SPACE.  E
      s erscheint"
210   PRINT "dann der Grundriss des Labyrinth mit allen Durchg
      aengen."
220   PRINT :PRINT "Darin bedeuten:"
230   PRINT "  Schwarzer Fleck: Anfangs- und Zielpunkt"
240   PRINT "       Roter Fleck: Endpunkt"
250   PRINT "    Gruener Fleck: Ihr Standort"
260   PRINT "  Schwarze Punkte: Hinweg"
270   PRINT "    Weisse Punkte: Ihr Weg"
280   PRINT :PRINT TAB(16);"Bitte einen Moment warten..."
300   REM ........... Initialisierung ........................
310   CLEAR 2000:DIFFICULTY=12:WIDTH=10:HEIGHT=8:DIM STATUS(WI
      DTH-1,HEIGHT-1),PASSAGE(3)
320   DIM XDIR(3),YDIR(3),XBEG(3),YBEG(3),OPPOSITE(3),XMOV(3),
      YMOV(3)
330   YDIR(0)=1:YDIR(1)=-1:XDIR(2)=-1:XDIR(3)=1
340   OPPOSITE(0)=1:OPPOSITE(1)=0:OPPOSITE(2)=3:OPPOSITE(3)=2
350   FOR I=0 TO 3:XMOV(I)=XDIR(I)*8:YMOV(I)=YDIR(I)*8:NEXT
360   GOSUB 1500:REM ........ Grundriss und Wegberechnung ..
370   PRINT TAB(16);"Spiel startet mit SPACE":CALLM #D6DA
380   REM ............. Init. der Bildkonstanten ............
390   MODE 6:COLORG 8 1 2 14
400   XMID=XMAX/2+1:YMID=YMAX/2+1
410   XS=XMID+8:YS=YMID+8
420   XBEG(0)=XMID-16:XBEG(1)=XMID+8:XBEG(2)=XMAX-31:XBEG(3)=3
      2
430   YBEG(0)=32:YBEG(1)=YMAX-31:YBEG(2)=YMID-16:YBEG(3)=YMID+
      8
440   FILL 0,0 XMAX,YMAX 1:FILL 8,8 XMAX-8,YMAX-8 8
450   FILL XMID-16,YMID-16 XMID+16,YMID+16 8
460   REM ............. Spielbeginn ........................
470   RX=STARTX:RY=STARTY:STATUS=STATUS(RX,RY) IOR #8000
480   STATUS(RX,RY)=STATUS
490   GOSUB 1010:GOSUB 1200
600   REM ............... INKEY-Routine ....................
610   KEY=GETC:IF KEY=32 THEN 2000:IF KEY<16 OR KEY>19 THEN 61
      0
620   KEY=KEY-16
630   FILL XS,YS XS+7,YS+7 8
640   XS=XS+XMOV(KEY):YS=YS+YMOV(KEY)
```

```
650   IF SCRN(XS+4,YS+4)<>8 THEN XS=XS-XMOV(KEY):YS=YS-YMOV(KE
      Y)
660   FILL XS,YS XS+7,YS+7 14
670   IF XS>7 AND XS<XMAX-7 AND YS>7 AND YS<YMAX-7 THEN 610
800   REM ............... Wechsel des Raumes .................
810   DIRECTION=KEY:IF DESTINATION=1 THEN STPS=STPS+1
820   RX=(WIDTH+RX+XDIR(DIRECTION)) MOD WIDTH:RY=(HEIGHT+RY+YD
      IR(DIRECTION)) MOD HEIGHT
830   STATUS=STATUS(RX,RY):GOSUB 1000
840   IF STATUS IAND #2400=#400 THEN GOSUB 1200
850   IF STATUS IAND #100>0 THEN GOSUB 1300
860   IF STATUS IAND #200>0 THEN GOSUB 1400
870   XS=XBEG(DIRECTION):YS=YBEG(DIRECTION)
880   FILL XS,YS XS+7,YS+7 14
890   STATUS(RX,RY)=STATUS IOR #2000:REM Raum betreten
900   GOTO 610
910   REM ................... Zurueckgefunden .................
920   PRINT "Schritte hin:";ROOMS,"Zurueck";STPS-1
930   END
1000  REM ................. Neuen Raum zeichnen .............
1010  OPPOSITE=OPPOSITE(DIRECTION):COLORG 8 STATUS IAND #F 2 1
      4
1020  FOR I=0 TO 3:PASSAGE(I)=(STATUS SHR I+4) IAND 1:NEXT
1030  FILL XMID-16,YMAX XMID+16,YMAX-7 21-PASSAGE(0)
1040  FILL XMID-16,0 XMID+16,7 21-PASSAGE(1)
1050  FILL 0,YMID-16 7,YMID+16 21-PASSAGE(2)
1060  FILL XMAX,YMID-16 XMAX-7,YMID+16 21-PASSAGE(3)
1070  FILL XMID-8,YMID-8 XMID+8,YMID+8 20
1080  RETURN
1200  REM ................ Wegweiser zeichnen ..............
1210  WAY=(STATUS(RX,RY) SHR 11) IAND 3
1220  WX=XMOV(WAY):WY=YMOV(WAY)
1230  DRAW XMID-WX,YMID-WY XMID+WX,YMID+WY 23
1240  DRAW XMID-WY/2,YMID-WX/2 XMID+WX,YMID+WY 23
1250  DRAW XMID+WY/2,YMID+WX/2 XMID+WX,YMID+WY 23
1260  RETURN
1300  REM ................ Start erreicht ..................
1310  FILL XMID-4,YMID-4 XMID+4,YMID+4 22:FILL XMID-3,YMID-3 X
      MID+3,YMID+3 20
1320  IF DESTINATION=1 THEN 910
1330  RETURN
1400  REM ................ Ende erreicht ...................
1410  DRAW XMID-4,YMID-4 XMID+4,YMID+4 22:DRAW XMID+4,YMID-4 X
      MID-4,YMID+4 22
1420  DESTINATION=1
1430  RETURN
1500  REM ................ Hauptdurchgang ..................
1510  STARTX=RND(WIDTH):STARTY=RND(HEIGHT)
1520  RX=STARTX:RY=STARTY:OPPOSITE=5:ROOMS=RND(DIFFICULTY)
1530  STATUS(RX,RY)=#100
1540  FOR I=0 TO ROOMS
1550  DIRECTION=RND(4):IF DIRECTION=OPPOSITE THEN 1550:OPPOSIT
      E=OPPOSITE(DIRECTION)
1560  STATUS(RX,RY)=STATUS(RX,RY) IOR #400 IOR (DIRECTION SHL
      11) IOR (16 SHL DIRECTION)
1570  RX=(WIDTH+RX+XDIR(DIRECTION)) MOD WIDTH:RY=(HEIGHT+RY+YD
      IR(DIRECTION)) MOD HEIGHT
1580  STATUS(RX,RY)=STATUS(RX,RY) IOR (16 SHL OPPOSITE)
1590  NEXT I:STATUS(RX,RY)=STATUS(RX,RY) IOR #200
1600  XEND=RX:YEND=RY
1800  REM ............. Durchgaenge & Farben .............
```

```
1810  FOR I=0 TO WIDTH-1:FOR J=0 TO HEIGHT-1
1820  COL=RND(16):IF COL=8 THEN 1820
1830  DIRECTION=RND(4):STATUS(I,J)=STATUS(I,J) IOR COL IOR (16
       SHL DIRECTION)
1840  RX=(WIDTH+I+XDIR(DIRECTION)) MOD WIDTH:RY=(HEIGHT+J+YDIR
      (DIRECTION)) MOD HEIGHT
1850  STATUS(RX,RY)=STATUS(RX,RY) IOR (16 SHL OPPOSITE(DIRECTI
      ON))
1860  NEXT J:NEXT I
1870  RETURN
2000  REM ................. Plan ..........................
2010  MODE 3
2020  FOR I=0 TO WIDTH-1:FOR J=0 TO HEIGHT-1:XI=I*16:XJ=J*16
2030  FILL XI,XJ XI+15,XJ+15 STATUS(I,J) IAND #F
2040  FILL XI+2,XJ+2 XI+13,XJ+13 8
2050  NEXT J:NEXT I
2060  FOR I=0 TO WIDTH-1:FOR J=0 TO HEIGHT-1:XI=I*16:XJ=J*16:S
      TATUS=STATUS(I,J)
2070  IF STATUS IAND 16>0 THEN FILL XI+6,XJ+14 XI+10,XJ+15 8
2080  IF STATUS IAND 32>0 THEN FILL XI+6,XJ XI+10,XJ+1 8
2090  IF STATUS IAND 64>0 THEN FILL XI,XJ+6 XI+1,XJ+10 8
2100  IF STATUS IAND 128>0 THEN FILL XI+14,XJ+6 XI+15,XJ+10 8
2110  IF STATUS IAND #400>0 THEN DOT XI+8,XJ+6 0
2120  IF STATUS IAND #2000>0 THEN DOT XI+8,XJ+10 15
2130  NEXT J:NEXT I
2140  XI=STARTX*16:XJ=STARTY*16
2150  FILL XI+2,XJ+2 XI+12,XJ+12 0:REM ............... START
2160  XI=XEND*16:XJ=YEND*16
2170  FILL XI+2,XJ+2 XI+12,XJ+12 2:REM ................ ENDE
2180  XI=RX*16:XJ=RY*16
2190  FILL XI+2,XJ+2 XI+12,XJ+12 5:REM ............ STANDORT
2200  GOTO 2200
3000  REM ..................................................
3010  REM ............. BIT-Belegung STAT ..................
3020  REM .....  0- 3 : Farben ...........  4- 7 : Tueren ...
3030  REM .....     8 : Start ............     9 : Ende .....
3040  REM .....    10 : Weg ............. 11-12 : Richtung .
3050  REM .....    13 : Vom Spieler betretenes Feld ........
3060  REM ..................................................
```

Datum : 25.3. Programm : Wegweiser

COL %
1820,1830,

DESTINATION % Flag, ob Kreuz erreicht
810,1320,1420,

DIFFICULTY % Schwierigkeitsgrad = Maximalzahl Schritte
310,1520,

DIRECTION % Richtungsvariable für Cursor, Turm...
810,820,870,1010,1550,1560,1570,1830,1840,1850,

HEIGHT % Höhe des Labyrinths
310,820,1510,1570,1810,1840,2020,2060,

I %
350,1020,1540,1590,1810,1830,1840,1860,2020,2030,
2050,2060,2130,

J %
1810,1830,1840,1860,2020,2030,2050,2060,2130,

KEY %
610,620,640,650,810,

OPPOSITE % gibt Richtungsindex der Gegenrichtung von DIR an
1010,1520,1550,1580,

OPPOSITE % ()
320,340,1010,1550,1850,

PASSAGE % () Nummer des Index steht für Turmnummer: Art/Zu-Flag
310,1020,1030,1040,1050,1060,

ROOMS % Anzahl der Schritte vom Start zum Ziel
920,1520,1540,

RX % Aktuelle Spielerkoordinaten
470,480,820,830,890,1210,1520,1530,1560,1570,
1580,1590,1600,1840,1850,2180,

RY %
470,480,820,830,890,1210,1520,1530,1560,1570,
1580,1590,1600,1840,1850,2180,

STARTX % Startkoordinaten
470,1510,1520,2140,

STARTY %
470,1510,1520,2140,

STATUS % Enthält alle Angaben über Raum: Farbe, Türen...
470,480,830,840,850,860,890,1010,1020,2060,2070,
2080,2090,2100,2110,2120,

STATUS % () s. Zeilen 3000 –
310,470,480,830,890,1210,1530,1560,1580,1590,
1830,1850,2030,2060,

STPS % Schritte des Spielers
810,920,

WAY %
1210,1220,

WIDTH % Breite d. Lab.
310,820,1510,1570,1810,1840,2020,2060,

WX %
1220,1230,1240,1250,

WY %
1220,1230,1240,1250,

XBEG % () Startpunkt Cursor bei Raumwechsel
320,420,870,

XDIR % () Richtungsindex bestimmt XDIR;
320,330,350,820,1570,1840,

XEND % Endkoordinaten
1600,2160,

XI %
2020,2030,2040,2060,2070,2080,2090,2100,2110,
2120,2140,2150,2160,2170,2180,2190,

```
XJ %
        2020,2030,2040,2060,2070,2080,2090,2100,2110,
        2120,2140,2150,2160,2170,2180,2190,
XMID %                                  Bildschirmmitte
        400,410,420,450,1030,1040,1070,1230,1240,1250,
        1310,1410,
XMOV % ()                   XDIR #8; Für Cursorbewegung
        320,350,640,650,1220,
XS %
        410,630,640,650,660,670,870,880,
YBEG % ()
        320,430,870,
YDIR % ()
        320,330,350,820,1570,1840,
YEND %
        1600,2160,
YMID %
        400,410,430,450,1050,1060,1070,1230,1240,1250,
        1310,1410,
YMOV % ()
        320,350,640,650,1220,
YS %
        410,630,640,650,660,670,870,880,
*
```

Liebe Freunde,

Hier ein kleines Orientierungsspielchen in BASIC. Die Aufgabe besteht darin, den Weg zu einem Ausgangsort zurueckzufinden. Das Spiel nimmt sich in der vorliegenden Sparversion allerdings mit der Zeit etwas öd au, überzeugt aber durch seine enormen Erweiterungsmoeglichkeiten. Wie wäre es z. Bsp. mit:

- einigen Monstern, die dem Spieler das Leben schwer machen (Türen versperren, Angriffe)
- Aufgaben (einen verborgenen Schatz bergen, ein Unterlabyrinth durchqueren, einen Wärter überrumpeln)
- Hilfsutensilien (Schlüssel, Kompaß, Zauberring)
- Diversen Pforten mit unterschiedlichen Eigenschaften (abwechselnd offen/geschlossen; Geheimtueren; Tueren, die nur mit Schluessel zu öffnen sind; solchen, die nur in einer Richtung durchlässig)
- ... aber seht selbst

Dadurch könnte es sich zu einem echten Abenteuerspiel mausern. Mit Hilfe eines FGT ließe sich auch die Graphik arcadefähig machen. Wenn die Geschwindigkeit nachläßt, ist auch der Übergang zu einer anderen Programmiersprache (Assembler, TP) sinnvoll, wobei meine Vorliebe TP gilt.
Wem gefällt die Spielidee, wer hat Lust, das Spiel so aufzumöbeln, daß daraus ein spannendes Abenteuerspiel wird? Ich jedenfalls erwarte die nächsten Ausgaben mit Spannung.

RG Schad

Super-Wurm von Heinrich Tegethoff

Bei diesem kleinen Spiel muß ein zunächst kleiner, jedoch später immer größer werdender Wurm gelenkt werden und zwar so, daß er möglichst viele der Nahrungshappen fressen kann. Bei jedem gefressenen Nahrungshappen wächst der Wurm und damit wird die Kontrolle schwieriger. Jedoch sollte man hier schnell reagieren, denn die Nahrung ist leicht verderblich: Sie verfault.
Außer der schlechten Nahrung, die er nicht fressen darf, muß er sich auch vor den Wänden in Acht nehmen. In sich selbst hineinlaufen und dann einen Knoten erzeugen ist ebenfalls nicht erlaubt.
Der einstellbare Schwierigkeitsgrad bestimmt, ab welcher Wurmlänge ein Bonus Wurm gewährt wird.

```
100   REM ==========================================
110   REM +     SUPER-WURM
120   REM +
130   REM + H. Tegethoff, 22.3.1981
140   REM ==========================================
150   REM
160   CLEAR 3000
162   MODE 0:PRINT CHR$(12):PRINT :PRINT :PRINT
164   PRINT "Spielstaerke (1-5) ";:POKE #77,63
166   A255=GETC-45:IF A255<4 OR A255>8 GOTO 166:A255=(1 SHL A255)-1
170   DIM DX(3),DY(3)
180   FOR R=0 TO 3:READ DX(R),DY(R):NEXT
190   DATA 0,1, 0,-1, -1,0, 1,0
200   COLORG 4 0 7 5:MODE 2:PRINT :ANZW=5
210   CURSOR 10,3:PRINT "Wuermer :";ANZW;"              Laenge :"
220   CURSOR 15,1:PRINT "     Punkte :";PUNKTE;SPC(20)
230   POKE #75,32:IF S<>K THEN COLORG 4 0 14 5:WAIT TIME 150
240   IF ANZW>0 GOTO 300
250   CURSOR 0,0:PRINT "Noch ein Spiel ";:POKE #75,95:POKE #77,63:POKE
      #2C3,0
260   G=GETC:IF G<32 GOTO 260
270   IF G=78 THEN PRINT "? Nein !":END
280   PUNKTE=0:S=K:GOTO 200
300   DIM X(A255),Y(A255):COLORG 4 0 7 5
310   KX=35:KY=29:S=0:K=0:R=RND(4)
320   FILL 1,1 XMAX-1,51 21:DOT KX,KY 22
330   X(0)=KX:Y(0)=KY:COUNT=RND(2)+1:FAULEN=160
350   X=RND(70.0)+1.0:Y=RND(51.0)+1.0:IF SCRN(X,Y)>0 GOTO 350:DOT X,Y
       23
360   G=GETC:IF G>15 THEN IF G<20 THEN R=G-16
370   KX=KX+DX(R):KY=KY+DY(R)
380   IF SCRN(KX,KY)=0 GOTO 400:IF SCRN(KX,KY)<>5 THEN PUNKTE=PUNKTE+
      (K-S IAND A255):ANZW=ANZW-1:GOTO 210
390   DOT KX,KY 21:KX=KX-DX(R):KY=KY-DY(R):COUNT=0:FAULEN=160+(K-S IA
      ND A255)/1.5:GOTO 350
400   DOT KX,KY 22:K=K+1 IAND A255:IF K=S GOTO 500:X(K)=KX:Y(K)=KY
410   CALLM #C0BB,COUNT:IF COUNT<6 THEN CURSOR 41,3:PRINT K-S IAND A2
      55;"  ";:GOTO 360
420   DOT X(S),Y(S) 21:S=S+1 IAND A255:IF COUNT<FAULEN GOTO 360
430   COUNT=5:FAULEN=160+RND(7)+(K-S IAND A255):DOT X,Y 20:GOTO 350
500   PUNKTE=PUNKTE+A255*1.2:ANZW=ANZW+1
510   CURSOR 15,1:PRINT "Sie haben den SUPER-WURM ";:POKE #77,33
520   WAIT TIME 200:GOTO 210
```

Autorennen von Heinrich Tegethoff

Bei diesem Spiel muß ein kleiner Punkt - das Auto - möglichst schnell eine Runde um einen Ring gefahren werden. Sie können vorher die 'Geschwindigkeit' einstellen. Dies bestimmt Ihr Beschleunigungs- und Abbremsverhalten.
Ein Rennen wird jeweils mit 'S' gestartet und eine neue Geschwindigkeit können Sie mit 'G' einstellen. 'B' korrigiert das 'Bestergebnis'

```
100   CLEAR 200:DIM WX!(3),WY!(3),DX!(3),DY!(3)
110   FOR N=0 TO 3:READ WX!(N),WY!(N),DX!(N),DY!(N):NEXT
120   BEST=1200+RND(500.0):L=BEST:R=1
130   DATA 0,2,0,1,  0,-2,0,1,  -2,0,1,0,  2,0,1,0
140   DATA 40,14,23,70,28,110,50,145,64,187,67,220,60,255,34,280,11
150   DATA 297,12,310,22,315,37,320,70,320,130,315,168,305,178
160   DATA 280,193,190,200,100,197,44,174,18,148,8,120,9,28,15,13,40,
      14
170   DATA 62,78,21,82,94,110,110,137,115,164,118,178,115,217,105
180   DATA 243,90,262,87,268,93,265,113,260,135,248,155,215,165
190   DATA 177,167,137,165,100,160,75,148,65,126,60,100,60,90,62,78
200   MODE 6A:COLORG 0 5 12 1
210   COLORT 0 5 0 0:PRINT CHR$(12)
220   FOR N=1 TO 2:READ A,B,C:FOR M=1 TO C:READ A1,B1
230   DRAW A,B A1,B1 5:DRAW A+1,B A1+1,B1 5:DRAW A,B+1 A1,B1+1 5
240   A=A1:B=B1:NEXT:NEXT
250   FILL 190,180 210,181 5
260   FILL 265,60 266,75 5
270   FILL 50,40 51,60 5
280   FILL 155,67 160,116 1
290   CURSOR 5,1:PRINT "LAST :";
300   CURSOR 48,1:PRINT "BEST :";
310   CURSOR 17,1:PRINT "SCHNITT :";
320   CURSOR 34,1:PRINT "GESAMT :";
330   CURSOR 20,3:INPUT "Geschwindigkeit ";D!
340   RESTORE
350   FOR N=0 TO 3:READ WX!(N),WY!(N),DX!(N),DY!(N):DX!(N)=DX!(N)*1.5
      E-2*D!:DY!(N)=DY!(N)*1.5E-2*D!:NEXT
360   CURSOR 20,3:PRINT SPC(27);
370   CURSOR 48,3:PRINT R;". RUNDE "
380   CURSOR 4,0:PRINT L;"    ";
390   IF R>1 THEN SN=S*1.0/(R-1)+0.5:CURSOR 18,0:PRINT SN:"    ";
400   CURSOR 47,0:PRINT BEST;"    ";
410   CURSOR 34,0:PRINT S;"    ";
420   CURSOR 29,3:POKE #77,12:POKE #75,95
430   A=GETC IAND #DF:IF A<65 AND A<>8 GOTO 430
440   IF A=71 GOTO 330:IF A=69 THEN CURSOR 0,3:COLORT 8 0 0 0:END
450   IF A=66 THEN BEST=L:CURSOR 44,0:PRINT "    ";:GOTO 360
460   P=0:ZX!=161.0:ZY!=90.0:VX!=0.0:VY!=0.0:POKE #75,32
470   CURSOR 44,0:PRINT "    ";
480   FILL 155,67 160,116 1
490   FX=158:FY=65
500   A=GETC-16:IF A>=0 AND A<4 THEN VX!=VX!+(WX!(A)-SGN(VX!))*DX!(A)
      :VY!=VY!+(WY!(A)-SGN(VY!))*DY!(A)
505   POKE #2B1,0:POKE #2B2,0:POKE #2B3,0:POKE #2B4,0
510   ZX!=ZX!+VX!:ZY!=ZY!+VY!
520   IF ZX!<0.0 OR ZX!>335.0 OR ZY!<0.0 OR ZY!>211.0 THEN DOT ZX1!,Z
      Y1! 0:P=P+100+SQR(VX!*VX!+VY!*VY!)*40:ZX!=161.0:ZY!=90.0:VX!=0.
      0:VY!=0.0:GOTO 500
530   CALLM #C0BB,P:CURSOR 27,3:PRINT P;
540   IF SCRN(ZX!,ZY!)<>5 GOTO 560:DOT ZX1!,ZY1! 0:DOT FX,FY 5:DOT ZX
      !,ZY! 12:FX=ZX!:FY=ZY!:ZX1!=ZX!:ZY1!=ZY!
550   VX!=0.0:VY!=0.0:P=P+80+SQR(VX!*VX!+VY!*VY!)*60:GOTO 500
560   IF SCRN(ZX!,ZY!)<>1 GOTO 570:IF VX!>0.0 GOTO 580:P=P+5+SQR(VX!*
      VX!+VY!*VY!)*60:VX!=0.0:VY!=0.0:ZX!=161.0:ZY!=90.0
570   DOT ZX1!,ZY1! 0:DOT ZX!,ZY! 12:ZX1!=ZX!:ZY1!=ZY!:GOTO 500
580   S=S+P:IF P<BEST THEN CURSOR 44,0:PRINT "***";
590   L=P:DOT FX,FY 5
600   R=R+1:GOTO 360
```

NP 89

```
100   REM Copyright by M.Rahlff Feb.1982
110   GOSUB 1070:CLEAR 1000:COLORT 8 0 5 0
120   INPUT "Wievielstelliger Code";ZA!:PRINT
130   ZA!=INT(ABS(ZA!)):IF ZA!<=0.0 OR ZA!>26.0 THEN 120
140   DIM EI!(ZA!+1.0),ES!(ZA!+1.0),ER!(ZA!+1.0),EI$(ZA!+1.0)
150   PRINT CHR$(12);:PR$=CHR$(29)
160   PRINT TAB(23);:FOR I!=1.0 TO 15.0:PRINT PR$;:NEXT:PRINT
170   PO!=#BFEF-192.0:FOR I!=1.0 TO 26.0 STEP 2.0:POKE PO!-I!,25
      5:NEXT
180   FOR I!=1.0 TO 24.0:PRINT PR$;:NEXT:PRINT "   Anzeige   ";:
      FOR I!=1.0 TO 23.0:PRINT PR$;:NEXT:CURSOR 0,CURY-1
190   PRINT TAB(23);:FOR I!=1.0 TO 15.0:PRINT PR$;:NEXT:CURSOR 0
      ,CURY:PRINT PR$;:CURSOR 59,CURY:PRINT PR$
200   FOR I!=1.0 TO 20.0:PRINT PR$;SPC(58);PR$:NEXT
210   FOR I!=1.0 TO 60.0:PRINT PR$;:NEXT:CURSOR 2,18
220   FOR I!=1.0 TO ZA!:ES!(I!)=INT(RND(0.0)*10.0):NEXT I!:LO!=0
      .0
230   LO!=LO!+1.0:HOE!=HOE!+1.0:IF HOE!=15.0 THEN HOE!=0.0:CURSO
      R 2,19:GOTO 530
240   FOR I!=1.0 TO ZA!:ER!(I!)=ES!(I!):NEXT I!
250   L!=0.0
260   L!=L!+1.0:IF L!=ZA!+1.0 THEN 280
270   GOTO 320
280   R$=CHR$(GETC):IF R$=CHR$(0) THEN 280
290   IF R$=CHR$(8) THEN 380
300   IF ASC(R$)<>13 THEN 280
310   GOTO 420
320   EI$(L!)=CHR$(GETC):IF EI$(L!)=CHR$(0) THEN 320
330   IF EI$(L!)=CHR$(8) THEN 380
340   ENVELOPE 0 15,70;0:SOUND 0 0 15 0 FREQ(L!*20.0+31.0)
350   IF ASC(EI$(L!))<48.0 OR ASC(EI$(L!))>57.0 THEN 320
360   PRINT EI$(L!);:IF L!<ZA!+1.0 THEN 260
370   GOTO 420
380   SOUND 0 0 15 0 FREQ(500.0)
390   IF L!=1.0 THEN L!=L!-1.0:GOTO 410
400   PRINT "  ";:CURSOR CURX-3,CURY:PRINT " ";:CURSOR CURX-1,CU
      RY:L!=L!-2.0
410   SOUND OFF :IF L!<ZA!+1.0 THEN 260
420   SOUND OFF :PRINT "   ";:FOR X!=1.0 TO ZA!:EI!(X!)=VAL(EI$(
      X!)):NEXT X!
430   FOR V!=1.0 TO ZA!:IF ER!(V!)=EI!(V!) THEN PRINT "*";:ER!(V
      !)=10.0:EI!(V!)=10.0:GE!=GE!+1.0
440   NEXT V!:PRINT " ";
450   IF GE!=ZA! THEN GE!=0.0:GOTO 550
460   FOR N!=1.0 TO ZA!
470   FOR M!=1.0 TO ZA!
480   IF EI!(N!)=10.0 THEN 510
490   IF ER!(M!)=10.0 THEN 510
500   IF EI!(N!)=ER!(M!) THEN ER!(M!)=10.0:EI!(N!)=10.0:PRINT "+
      ";
510   NEXT M!
520   NEXT N!
530   GE!=0.0:PRINT :CURSOR 2,CURY:PRINT TAB(45);
540   GE!=0.0:CURSOR 2,CURY:GOTO 230
550   WAIT TIME 150:PRINT CHR$(12);TAB(25);:FOR I!=1.0 TO 11.0:P
      RINT PR$;:NEXT:PRINT
560   PRINT TAB(25);PR$;" geloest ";PR$:PRINT TAB(25);:FOR I!=1.
      0 TO 11.0:PRINT PR$;:NEXT:PRINT
570   PO!=#BFEF-61.0:COLORT 8 0 5 14:FOR T!=1.0 TO 2.0
580   FOR I!=0.0 TO 21.0 STEP 2.0:POKE PO!-I!,255:POKE PO!-I!-13
      4,255:POKE PO!-I!-268,255:WAIT TIME 1:POKE PO!-I!,0:POKE P
      O!-I!-134,0
590   POKE PO!-I!-268,0:SOUND OFF :WAIT TIME 1:SOUND 0 0 10 3 FR
      EQ((264.0)):SOUND 1 0 10 3 FREQ(330.0):NEXT I!:NEXT T!
```

SP 12,1

```
1050  IF R$<>"N" THEN 1030
1060  PRINT CHR$(12);"   AUF WIEDERSEHEN!":PRINT :PRINT :PRINT :
      END
1070  MODE 0:PRINT CHR$(12);"                              Code Brecher"
1080  COLORT 8 0 10 0
1090  PO!=#BFEF-52.0:FOR I!=1.0 TO 24.0 STEP 2.0:POKE PO!-I!,255
      :NEXT
1100  PRINT :PRINT :PRINT "Sie sind Geheimagent 0 0 8.Sie haben
      die Aufgabe,":PRINT "Geheimplaene zu photographieren.Diese
       Plaene";
1110  PRINT " liegen in einem"
1120  PRINT "Safe,dessen Code sie brechen muessen."
1130  PRINT "Dabei hilft ihnen ein Geraet,dass ihnen folgendes a
      nzeigt:"
1140  PRINT TAB(10);"'*'= Richtige Zahl an richtiger Stelle"
1150  PRINT TAB(10.0);"'+'= Richtige Zahl an falscher Stelle"
1160  PRINT :PRINT "VIEL GLUECK!"
1170  PRINT :PRINT :PRINT :PRINT "(Taste druecken!)";:CURSOR CUR
      X-2,CURY
1180  IF GETC=0.0 THEN 1180:PRINT CHR$(12):RETURN


  109 Lines           27 Symbols
   30 References     259 References
 4736 Bytes Text     200 Bytes Symbols

  284 Commands
```

```
100   REM ------------ SPACE - INVADERS ---------------------
110   REM ----  Autor:  Uwe Wienkop
120   REM
130   ENVELOPE 0 16
140   COLORT 0 5 6 7:PRINT CHR$(12):POKE #75,32:P=0:L=3:N=0
150   MODE 6A:COLORG 0 5 3 9:L=L+1:N=N+1:Q=135-15*N:IF Q<50 THEN Q=5
      0
160   FOR I=0 TO 100:DOT RND(XMAX),RND(180)+30 9:NEXT
170   FOR A=Q TO Q+60 STEP 15:FOR I=35 TO 285 STEP 25
180   FILL I,A+9 I+8,A+4 21:DRAW I,A+4 I+6,A 21:DRAW I+8,A+4 I+2,A 2
      1
190   DOT I+2,A+6 0:DOT I+6,A+6 0
200   DRAW I,A+8 I+2,A+9 0:DRAW I+6,A+9 I+8,A+8 0:DOT I,A+9 0:DOT I+
      8,A+9 0:NEXT:NEXT
210   FOR I=60 TO 240 STEP 60:FILL I,50 I+20,35 21:FILL I+5,40 I+15,
      35 0:NEXT
220   F=55
230   CURSOR 0,2:PRINT "Laser:";L;TAB(18);"Punkte:";P;TAB(35);"Highs
      core:";BEST
240   H=30:X=10
250   IF PEEK(#FD00) IAND 32=0 GOTO 270:DRAW S,H S,H-6 0:SOUND 0 0 1
      5 1 FREQ(500):SOUND 1 0 15 1 FREQ(1500)
260   S=X+5:FLAG=1:H=32
270   X=PDL(0)*1.21+5:IF X>LX-4 AND X<LX+4 GOTO 300:X=LX+3*SGN(X-LX)
280   FILL LX,30 LX+11,20 0:LX=X
290   FILL X,25 X+11,20 5:FILL X+5,30 X+6,25 5
300   IF FLAG=0 GOTO 370
310   SOUND OFF
320   H=H+6:IF SCRN(S,H)<>5 GOTO 350:IF H<52 THEN FILL S-2,H+3 S+2,H
      -1 0:FLAG=0:GOTO 370
330   NOISE 0 15:FILL (S/25)*25,(H/15+1)*15-2 (S/25+1)*25,(H/15)*15
      0:NOISE OFF
340   P=P+50:CURSOR 25,2:PRINT P;"  ":F=F-1:IF F=0 GOTO 150:GOTO 360
350   IF H<211 THEN DRAW S,H S,H-3 9:DRAW S,H-6 S,H-9 0:GOTO 370
360   FLAG=0:DRAW S,H-6 S,H-9 0
370   IF FC=1 GOTO 390
380   XC=X-5+RND(25):FC=1:HC=RND(0-50)+60
390   HC=HC-6:IF HC<6 THEN DRAW XC,HC+6 XC,HC 0:GOTO 380
400   IF SCRN(XC,HC-6)<>5 GOTO 450
410   IF HC>35 THEN DRAW XC,HC+6 XC,HC+3 0:FILL XC-2,HC+1 XC+2,HC-3
      0:FC=0:GOTO 250
420   NOISE 0 8:FOR I=0 TO 6:FILL XC-2,HC+1 XC+2,HC-3 3:WAIT TIME 10
430   FILL XC-2,HC+1 XC+2,HC-3 0:WAIT TIME 10:NEXT:NOISE OFF
440   DRAW XC,HC+6 XC,HC+3 0:FC=0:L=L-1:FILL X-1,30 X+12,20 0:IF L=0
       GOTO 480:GOTO 230
450   DRAW XC,HC XC,HC-3 9:DRAW XC,HC+1 XC,HC+6 0
460   IF XC=S AND HC=H THEN FLAG=0:FC=0
470   GOTO 250
480   IF BEST<P THEN BEST=P:CURSOR 45,2:PRINT BEST
490   CURSOR 20,1:PRINT "Noch ein Spiel (J/N)?"
500   A=GETC:IF A=0 GOTO 500:IF A<>78 GOTO 140
510   COLORT 8 0 0 0:POKE #75,95:END
```

```
   42 Lines            16 Symbols
  417 References      181 References
 2131 Bytes Text      122 Bytes Symbols

  125 Commands
```

L a s s D i r Z e i t

```
100    GOTO 150
110    POKE #1BE,255:POKE #1BF,255
120    A=PDL(PDLNR)*X!:P=COUNTUP(A)+1:IF P<Y THEN DOT A,P 21:COUNTU
       P(A)=P:GOTO 120
130    ZEIT=65535-PEEK(#1BE)-PEEK(#1BF)*256:RETURN
140    REM
150    CLEAR 2000:PDLNR=0
160    PRINT CHR$(12),"L a s s   D i r   Z e i t":PRINT :PRINT
170    PRINT "An der Stelle, auf die Sie mit dem Paddle zusteuern,
       werden"
180    PRINT "Steinchen uebereinandergestapelt. Sie muessen nun ver
       suchen,"
190    PRINT "moeglichst lange die entstehenden Tuerme breit zu str
       euen,"
200    PRINT "bis ein Turm die obere Kante erreicht.":PRINT
210    PRINT "Der Titel 'Lass Dir Zeit' kommt uebrigens daher, dass
        die"
220    PRINT "Spielkameraden im allgemeinen ein gehaessiges 'Lass D
       ir doch"
230    PRINT "Zeit' rufen, je wilder man von Sekunde zu Sekunde an
       dem"
240    PRINT "Paddle herumzuckt...":PRINT :PRINT
250    PRINT :PRINT :PRINT "Druecken Sie den Paddle-Knopf !"
260    IF PEEK(#FD00) IAND #30=0 GOTO 260
270    DIM COUNTUP(255)
280    COLORG 1 15 5 10
290    MODE 2:DRAW 0,YMAX XMAX,YMAX 22
300    Y=YMAX:X!=XMAX/256.0
310    GOSUB 110
320    DRAW 0,YMAX XMAX,YMAX 23
330    IF PEEK(#FD00) IAND #30=0 GOTO 330
340    MODE 0
350    IF ZEIT>REKORD_ZEIT THEN REKORD_ZEIT=ZEIT
360    PRINT :PRINT :PRINT :PRINT "Zeit : ";ZEIT/50.0;" s      ReKor
       d : ";REKORD_ZEIT/50.0;" s"
370    WAIT TIME 100
380    GOTO 250


   29 Lines             8 Symbols
    6 References       24 References
  985 Bytes Text       77 Bytes Symbols

   50 Commands

A %           120
COUNTUP % ()  120,270
P %           120
PDLNR %       120,150
REKORD_ZEIT %  350,360
X !           120,300
Y %           120,300
ZEIT %        130,350,360
```