FLASHING IN 4 COLOR

Here is a convenient way of making items flash on the DAI display. The method uses a small machine code routine which is triggered by the RST7 interrupt. At intervals controlled by a parameter the routine switches the value in the colour control byte at location £BFF2. This is the one which is used to set up colour register 3 in a 4-colour mode. As a result anything which is drawn on the screen using colour '3' will flash (change colour regularly).

Here is the machine code routine that I use.-

```
FLASH:   PUSH H
         LXI  H,SWITCH
         SHLD £70                ; INTERCEPT RST7
         POP  H
         RET
SPEED:   DC   10                 ; FLASH RATE PARAMETER
REMAIN   DC   0                  ; WORKING REMAINDER VALUE
DIFFER   DC   0                  ; COLOUR CHANGE PARAMETER
SWITCH:  PUSH PSW
         LXI  H,SPEED            ; EACH TIME SWITCH IS EXECUTED
         MOV  A,M                ; SPEED IS ADDED TO REMAIN
         INX  H                  ; IF A CARRY OCCURS IT IS TIME
         ADD  M                  ; TO SWAP COLOURS IN £BFF2
         MOV  M,A
         JNC  NOCARRY
         INX  H
         LDA  £BFF2              ; 'DIFFER' IS TAKEN AS THE
         XRA  M                  ; LOGICAL DIFFERENCE BETWEEN
         STA  £BFF2              ; THE TWO COLOUR CODES
NOCARRY: POP  PSW
         LXI  H,£D9A9            ; NORMAL RST7 CONTINUATION
         PCHL
```

Here is a simple example program using the technique.

```
10 CLEAR 1000
20 POKE £29B,3: POKE £29D,PEEK(£29D)-1: REM make space
30 FOR I=£2EC TO £30F: READ J: POKE I,J: NEXT
40 MODE 6A: COLORG 15 5 1 5
50 CALLM £2EC: REM Activate FLASH routine
60 POKE £2F7,10: REM Colour register will alternate 15 - 5
60 FILL 100,30 110,60 21: REM Ordinary green blob
70 FILL 150,50 160,60 23: REM Flashing green blob
80 DATA £E5,£21,£F5,£02,£22,£70,£00,£E1,£C9
90 DATA £0A,£00,£00,£F5,£21,£F5,£02,£7E,£23
100 DATA £86,£77,£D2,£0B,£03,£23,£3A,£F2,£BF
110 DATA £AE,£32,£F2,£BF,£F1,£21,£A9,£D9,£E9
```

Now you can try different speeds and colours by POKING values into £2F5 and £2F7.- (£2F5 is the speed control and is preset at 10 giving a flash time in milliseconds of 20*(256/10) or about half a second in each colour; - £2F7 is the logical difference between the colour codes). The difference should not exceed 15 or it will destroy the screen format.

IMPORTANT NOTE: flashing at a rate of between 7 and 10 cycles per second can be unpleasant and even dangerous! so avoid values for SPEED which are greater than 50 or so.

Harry Harte
28. 4. 82

SOFTWARE PRINTER-SPOOLER

```
002                 *
003                 * Peter Lelie, 19-Oct-82
004                 * D-4047 Dormagen-Zons
005                 * Wilhelm Busch Strasse 36
006                 * Tel.: (02106) 46852
007                 *
008                 *
009                 * Program Description
010                 *
011                 * The printer spooler consists of three parts:
012                 *
013                 * 1. Entry point ENTER
014                 * Called via DOUTC (:02DD). If you want text to
015                 * be printed, POKE #131,3 . From now on all the
016                 * PRINTed stuff will be entered into the printer
017                 * buffer (inside RAM). If the buffer is filled,
018                 * the program waits for buffer to become free.
019                 * To switch off the entering of output into the
020                 * buffer, just POKE #131,1 .
021                 *
022                 * 2. Entry point SPOOL
023                 * Called via RST7 every 20 msec. As the printer
024                 * is ready and there are characters inside the
025                 * buffer area, they will be printed, one after
026                 * another. After a character has been printed,
027                 * the used bufferspace will be freed for further
028                 * use by ENTER. SPOOL takes attention on the
029                 * 'DATA TERMINAL READY'-signal from the printer.
030                 *
031                 * 3. Entry point INIT
032                 * To be called from BASIC (only one time) to
033                 * install the spooler. BEFORE loading a BASIC
034                 * program, do a CALLM #300 to init the spooler
035                 * and to adjust the heap pointer and a NEW is
036                 * generated. After the first INIT, nothing bad
037                 * will happen if you do another CALLM #300,
038                 * as this will only execute a RETurn instruction.
039                 *
040                 *
041                 * DECLARATIONS
042                 *
043 C0      BAUD    EQU   :C0       ;9600 Bd
044                 *
045 0800    BUFSIZ  EQU   :800      ;buffer size (2k)
046 02DD    DOUTC   EQU   :02DD     ;output vector
047 029B    HEAP    EQU   :029B     ;heap ptr loc.
048 C3      JUMP    EQU   :C3       ;code for JMP
049 0131    PRISW   EQU   :0131     ;printer switch
050 C9      RETUR   EQU   :C9       ;code for RET
051 0070    RST7    EQU   :0070     ;vector addr.
052                 *
053                 * DAI hardware related declarations
054 FF05    BRREG   EQU   :FF05     ;Baudrate reg.
055 FD00    DTR     EQU   :FD00     ;data term rdy
056 FFF6    V24DA   EQU   :FFF6     ;ser. data
057 FFF3    V24ST   EQU   :FFF3     ;ser. status
058                 *
059                 * DAI ROM entrypoints:
060 DE14    CMPDH   EQU   :DE14     ;compare DE, HL
061 D9A9    OPSYS   EQU   :D9A9     ;rest of RST7
062 DAD4    PMSG    EQU   :DAD4     ;print message
063 DEB8    RNEW    EQU   :DEB8     ;Basic NEW
064                 *
065                 * Literals
```

```
066                     CR      EQU   :0D       ;car ret
067                     FF      EQU   :0C       ;formfeed
068                     LF      EQU   :0A       ;linefeed
069                     *
070                     *
071                             ORG   :0300
072 0300 C37A03         INIT    JMP   SETUP     ;RET after init
073                     *
074                     * ENTRY POINT VIA RST7
075                     *
076 0303 D5             SPOOL   PUSH  D         ;SAVE REG'S
077 0304 E5                     PUSH  H
078 0305 F5                     PUSH  PSW
079                     *
080                     * avoid output to screen and printer:
081                     *
082 0306 213101                 LXI   H,PRISW   ;if = 0
083 0309 7E                     MOV   A,M       ;then
084 030A FE00                   CPI   :00       ;make
085 030C C21003                 JNZ   OK        ;PRISW=1
086 030F 34                     INR   M         ;
087                     *
088                     * is RS232 ready for output?
089                     *
090 0310 3A00FD         OK      LDA   DTR       ;term ready?
091 0313 E608                   ANI   :08       ;
092 0315 CA3E03                 JZ    RETRN     ;
093 0318 3AF3FF                 LDA   V24ST     ;UART ready?
094 031B E610                   ANI   :10       ;
095 031D CA3E03                 JZ    RETRN     ;
096                     *
097                     * ready for output;
098                     * do we have any output?
099                     *
100 0320 2A7803                 LHLD  PTR2      ;
101 0323 EB                     XCHG            ;
102 0324 CD6003                 CALL  COMPAR    ;is PTR1<>PTR2?
103 0327 CA3E03                 JZ    RETRN     ;RET if not
104 032A 7E                     MOV   A,M       ;get char
105 032B 32F6FF                 STA   V24DA     ;print it
106 032E FE0D                   CPI   CR        ;carriage return ?
107 0330 C23803                 JNZ   NOCR      ;no
108 0333 360A                   MVI   M,LF      ;replace CR into LF
109 0335 C33E03                 JMP   RETRN     ;no change of pionters
110 0338 CD6703         NOCR    CALL  INCPTR    ;move pointer
111 033B 227603                 SHLD  PTR1      ;and restore
112 033E F1             RETRN   POP   PSW       ;restore reg's
113 033F E1                     POP   H         ;
114 0340 D1                     POP   D         ;
115 0341 C3A9D9                 JMP   OPSYS     ;back to opsys
116                     *
117                     * The following routine is called via
118                     * 'DOUTC'
119                     * Purpose: enter the received character
120                     * into printer buffer
121 0344 D5             ENTER   PUSH  D         ;save reg's
122 0345 E5                     PUSH  H         ;
123 0346 F5                     PUSH  PSW       ;
124                     *
125 0347 2A7803                 LHLD  PTR2      ;
126 034A CD6703                 CALL  INCPTR    ;
127 034D EB                     XCHG            ;
128 034E CD6003         LOOP    CALL  COMPAR    ;try to find
129 0351 CA4E03                 JZ    LOOP      ;buffer space
```

```
130                     *
131                     * buffer space available:
132                     * now put char into buffer:
133 0354 F1                     POP   PSW       ;
134 0355 2A7803                 LHLD  PTR2      ;
135 0358 77                     MOV   M,A       ;stor in buff
136 0359 EB                     XCHG            ;
137 035A 227803                 SHLD  PTR2      ;stor new PTR2
138 035D E1                     POP   H         ;restore reg's
139 035E D1                     POP   D         ;
140 035F C9                     RET             ;to sender!
141                     *
142                     * SUBROUTINES
143                     *
144                     * compare values in HL and DE
145                     *
146 0360 2A7603         COMPAR  LHLD  PTR1      ;get 1st ptr
147 0363 CD14DE                 CALL  CMPDH     ;comp DE, HL
148 0366 C9                     RET             ;
149                     *
150                     * move PTR in HL to next position
151                     *
152 0367 11790B         INCPTR  LXI   D,BUFEND  ;get end of buff
153 036A CD14DE                 CALL  CMPDH     ;compare
154 036D C27403                 JNZ   NXT       ;OK if not end
155                     * reached end of buffer:
156                     * goto start of buffer
157 0370 217A03                 LXI   H,BUFFER  ;PTR = BUFFER
158 0373 C9                     RET             ;
159 0374 23             NXT     INX   H         ;PTR = next pos.
160 0375 C9                     RET             ;
161                     *
162                     * Buffer area following:
163                     *
164 0376 0000           PTR1    DBL   :0000
165 0378 0000           PTR2    DBL   :0000
166                     *
167 037A                BUFFER  RES   BUFSIZ-1
168 0B79 00             BUFEND  DATA  :00
169 0B7A 00             FINI    NOP
170                     *
171                     * the following code is located inside
172                     * the printer buffer. After INIT it
173                     * will be overwritten!
174                     *
175                             ORG   BUFFER
176                     *
177 037A E5             SETUP   PUSH  H         ;save reg's
178 037B F5                     PUSH  PSW       ;
179 037C 3EC3                   MVI   A,JUMP    ;set up DOUTC
180 037E 32DD02                 STA   DOUTC     ;
181 0381 214403                 LXI   H,ENTER   ;
182 0384 22DE02                 SHLD  DOUTC+1   ;
183                     *
184 0387 3EC9                   MVI   A,RETUR   ;kill entrypoint
185 0389 320003                 STA   INIT      ;
186                     *
187 038C 217A03                 LXI   H,BUFFER  ;ini PTR1 & PTR2
188 038F 227603                 SHLD  PTR1      ;
189 0392 227803                 SHLD  PTR2      ;
190                     *
191 0395 3EC0                   MVI   A,BAUD    ;init baudrate
192 0397 3205FF                 STA   BRREG     ;
193                     *
```

```
194 039A 3E01              MVI   A,1       ;printer off
195 039C 323101            STA   PRISW     ;
196                 *
197 039F 217A0B            LXI   H,FINI    ;set heap ptr
198 03A2 229B02            SHLD  HEAP      ;
199 03A5 CD88DE            CALL  RNEW      ;run NEW
200                 *
201 03A8 21B703            LXI   H,TXT$    ;print msg
202 03AB CDD4DA            CALL  PMSG      ;
203                 *
204 03AE 210303            LXI   H,SPOOL   ;init RST7
205 03B1 227000            SHLD  RST7      ;
206                 *
207 03B4 F1                POP   PSW       ;
208 03B5 E1                POP   H         ;
209 03B6 C9                RET             ;
210                 *
211                 *
212 03B7 0C        TXT$    DATA  FF
213 03B8 507269            ASC   'Printer spooler '
214 03C8 696E69            ASC   'initialized.'
215 03D4 0D0D              DATA  CR,CR
216 03D6 507269            ASC   'Printer ON : POKE #131,3'
217 03EE 0D                DATA  CR
218 03EF 202020            ASC   '        OFF: POKE #131,1'
219 0407 0D0D00            DATA  CR,CR,0
220                 *
221                 LENGTH  EQU   FINI-SPOOL
222 040A                    END
```

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

BAUD	00C0	BRREG	FF05	BUFEND	0B79	BUFFER	037A
BUFSIZ	0800	CMPDH	DE14	COMPAR	0360	CR	000D
DOUTC	02DD	DTR	FD00	ENTER	0344	FF	000C
FINI	0B7A	HEAP	029B	INCPTR	0367	INIT	0300
JUMP	00C3	LENGTH	0877	LF	000A	LOOP	034E
NOCR	0338	NXT	0374	OK	0310	OPSYS	D9A9
PMSG	DAD4	PRISW	0131	PTR1	0376	PTR2	0378
RETRN	033E	RETUR	00C9	RNEW	DE88	RST7	0070
SETUP	037A	SPOOL	0303	TXT$	03B7	V24DA	FFF6
V24ST	FFF3						

READ/WRITE IN UTILITY

HOW TO USE UTILITY 'WRITE' AND 'READ' IN M.L.PROGRAMS

The DAI utility package doesnot have a simple entry to enable writing to and reading from tape in combination with machine language programs. The oniest possibility to write and read M.L.programs is via the direct Utility commands 'W' and 'R'.
Annexed routines give an entry to the 'W'- and 'R'-routines in combination with M.L.programs.

When you are working with the Utility package, ROM-bank 3 of the firmware is used. Here also the write and the read routines for file type 1 (hex-files) can be found, resp. on 3EEE4 and 3EF0F. These are used in the routines below.

The WRITE routine (to be used via CALL WRITE), prints a 'W' on the screen. Then the startaddress and the endaddress of the file to be saved can be typed in, as well as the name of the file. Then the file is written to tape in exactly the same way as always.

The READ routine (CALL READ) works the same way. It prints a 'R', and then an eventual offset and the file name can be typed in. Then the file is read from tape.

```
002                     *
003                     *
004                             ORG    :300
005                     *
006                     *
007 0300 E5             WRITE   PUSH   H
008 0301 D5                     PUSH   D
009 0302 C5                     PUSH   B
010 0303 F5                     PUSH   PSW
011 0304 CD5EDD                 CALL   :DD5E       CAR.RET
012 0307 3E57                   MVI    A,:57
013 0309 CD60DD                 CALL   :DD60       PRINT 'W'
014 030C CDE4EE                 CALL   :EEE4       'WRITE'
015 030F F1                     POP    PSW
016 0310 C1                     POP    B
017 0311 D1                     POP    D
018 0312 E1                     POP    H
019 0313 C9                     RET
020                     *
021                     *
022                     *
023 0314 E5             READ    PUSH   H
024 0315 D5                     PUSH   D
025 0316 C5                     PUSH   B
026 0317 F5                     PUSH   PSW
027 0318 CD5EDD                 CALL   :DD5E       CAR.RET
028 031B 3E52                   MVI    A,:52
029 031D CD60DD                 CALL   :DD60       PRINT'R'
030 0320 CD0FEF                 CALL   :EF0F       'READ'
031 0323 F1                     POP    PSW
032 0324 C1                     POP    B
033 0325 D1                     POP    D
034 0326 E1                     POP    H
035 0327 C9                     RET
036                     *
037                     *
038                     *
039 0328                        END
```

The routine WRITE1 is a second possibility to write files on tape under control of a m.l.program.
Before calling this routine, the low-address of the file to be saved must be in the HL-register and the high-address in the DE-register. This is useful when the boundaries of the file depend on the action performed by your m.l.program.
The routine asks for the filename to be typed in before the file is written to tape.

```
001                             ORG    :300
002                     *
003 0300 E5             WRITE1  PUSH   H
004 0301 D5                     PUSH   D
005 0302 C5                     PUSH   B
006 0303 F5                     PUSH   PSW
007 0304 CD0C03                 CALL   WRITE2
008 0307 F1                     POP    PSW
009 0308 C1                     POP    B
010 0309 D1                     POP    D
011 030A E1                     POP    H
012 030B C9                     RET
013                     *
014 030C E5             WRITE2  PUSH   H
015 030D D5                     PUSH   D
016 030E 211703                 LXI    H,NAME
017 0311 CD2FED                 CALL   :ED2F       PRINT NAME:
018 0314 C3EDEE                 JMP    :EEED       'WRITE'
019                     *
020 0317 0D             NAME    DATA   :0D
021 0318 4E                     DATA   :4E         N
022 0319 41                     DATA   :41         A
023 031A 4D                     DATA   :4D         M
024 031B 45                     DATA   :45         E
025 031C 20                     DATA   :20
026 031D 3A                     DATA   :3A         :
027 031E 20                     DATA   :20
028 031F 00                     DATA   :00
029                     *
030 0320                        END
```

The use of the CHECK-routine in m.l.programs is difficult. Not only because it is a routine which can only be aborted by a BREAK, but also because the program- or filenames are given via a 'fast-print' routine. This routine 'prints' the filenames on the screen by poking the name directly into the screen RAM area.
Maybe someone wants to try to write a routine for this purpose. The article 'Cassette list' in DAInamic 82/1-p.48 can be helpful.
Please let me know when you succeeded.

Jan Boerrigter

program identification

title : LARGE TEXT (PRINTER-POSTER)

author : T. GROENEVELD

purpose : generates jumbo print-out

comment : control-characters are for EPSON-printers

```
1     REM CHANGE LINE SPACING ON EPSON : ?CHR$(27);"A";CHR$(9)
2     MODE 0:CLEAR 5000:COLORT 15 0 0 0:POKE #131,1
5     PRINT CHR$(12):PRINT
6     PRINT "              ############################"
7     PRINT "              #      LARGE TEXT          #"
8     PRINT "              #      ==========          #"
9     PRINT "              ############################":PRINT
10    INPUT "   Hoeveel karakters vertikaal........ ";X%:PRINT :PRINT
20    INPUT "   Hoeveel karakters horizontaal...... ";Y%:PRINT :PRINT
21    INPUT "   Wilt u tabulator gebruiken (J/N)... ";L$:PRINT :PRINT
22    G1%=0.0:IF L$="J" THEN G1%=1.0
23    PRINT "   Met welke tekst dienen karakters opgebouwd"
24    INPUT "   te worden.......................... ";M$:PRINT :PRINT
29    INPUT "   Wat zijn de gewenste karakters..... ";A$:PRINT :PRINT
30    IF M$="" OR A$="" THEN 5
35    INPUT "   STEL PAPIER IN EN GEEF RETURN";O$:PRINT :POKE #131,0
40    A%=ASC(LEFT$(A$,1))
65    DIM S%(10.0),J%(10.0),F%(10.0)
70    FOR T%=0.0 TO LEN(A$)-1.0
80    P$=MID$(A$,T%,1)
90    FOR Q%=1.0 TO 50.0
94    READ S$,S%(1.0),S%(2.0),S%(3.0),S%(4.0),S%(5.0),S%(6.0),S%(7.0)
98    IF P$=" " THEN 812
100   IF P$=S$ THEN 200
120   NEXT Q%
200   RESTORE
'201  X$=M$
205   FOR U%=1.0 TO 7.0
210   FOR K%=8.0 TO 0.0 STEP -1.0
230   IF INT(2.0^K%+0.5)<INT(S%(U%)) THEN 270
235   REM PRINT INT(2.0^K),INT(S(U)),U
240   J%(9.0-K%)=0.0
250   GOTO 280
270   J%(9.0-K%)=1.0:S%(U%)=S%(U%)-INT(2.0^K%+0.5)
272   IF INT(S%(U%))=1.0 THEN 815
280   NEXT K%
445   FOR T1%=1.0 TO X%
447   PRINT TAB((63.0-4.5*Y%)*G1%/(LEN(X$))+1.0);
450   FOR B%=1.0 TO F%(U%)
460   IF INT(J%(B%))=0.0 THEN 500
465   FOR I%=1.0 TO Y%:PRINT X$;:NEXT I%
470   GOTO 600
500   FOR I%=1.0 TO Y%
510   FOR I1%=1.0 TO LEN(X$)
520   PRINT " ";:NEXT I1%
530   NEXT I%
600   NEXT B%
```

```
620   PRINT
630   NEXT T1%
700   NEXT U%
750   FOR H%=1.0 TO 2.0*X%:PRINT :NEXT H%
800   NEXT T%
806   REM FOR N=1.0 TO 5.0:PRINT :NEXT N
810   POKE #131,1:GOTO 5
812   FOR N%=1.0 TO 7.0*X%:PRINT :NEXT N%
813   GOTO 800
815   F%(U%)=9.0-K%:GOTO 445
899   DATA " ",0,0,0,0,0,0,0
900   DATA "A",505,37,35,34,35,37,505
901   DATA "G",125,131,258,258,290,163,101
902   DATA "E",512,274,274,274,274,258,258
903   DATA "T",2,2,2,512,2,2,2
904   DATA "W",256,257,129,65,129,257,256
905   DATA "L",512,257,257,257,257,257,257
906   DATA "S",69,139,274,274,274,163,69
907   DATA "O",125,131,258,258,258,131,125
908   DATA "N",512,7,9,17,33,193,512
909   DATA "F",512,18,18,18,18,2,2
910   DATA "K",512,17,17,41,69,131,258
911   DATA "B",512,274,274,274,274,274,239
912   DATA "D",512,258,258,258,258,131,125
913   DATA "H",512,17,17,17,17,17,512
914   DATA "M",512,7,13,25,13,7,512
915   DATA "?",5,3,2,354,18,11,5
916   DATA "U",128,129,257,257,257,129,128
917   DATA "R",512,18,18,50,82,146,271
918   DATA "P",512,18,18,18,18,18,15
919   DATA "Q",125,131,258,258,322,131,381
920   DATA "Y",8,9,17,481,17,9,8
921   DATA "V",64,65,129,257,129,65,64
922   DATA "X",388,69,41,17,41,69,388
923   DATA "Z",386,322,290,274,266,262,260
924   DATA "I",258,258,258,512,258,258,258
925   DATA "C",125,131,258,258,258,131,69
926   DATA "J",65,129,257,257,257,129,128
927   DATA "1",0,0,261,259,512,257,257
928   DATA "2",261,387,322,290,274,267,261
929   DATA "*",69,41,17,512,17,41,69
930   DATA "3",66,130,258,274,266,150,100
931   DATA "4",33,49,41,37,35,512,33
932   DATA "5",160,274,274,274,274,274,226
933   DATA "6",194,291,293,297,305,289,193
934   DATA "7",258,130,66,34,18,10,8
935   DATA "8",69,171,274,274,274,171,69
936   DATA "9",263,138,74,42,26,10,7
937   DATA "=",41,41,41,41,41,41,41
938   DATA "!",1,1,1,384,1,1,1
939   DATA "0",57,69,131,258,131,69,52
940   DATA ".",1,1,129,449,129,1,1
1000  GOTO 5
1010  REM MET DEZE LARGE TEXT KUNT U TEKSTEN NAAR GEWENSTE GROOTTE MAKEN.
1020  REM VOORBEELD 1 IS: hor=4  vert=5 TAB=N  tekst=X   kar=DAI
1030  REM VOORBEELD 2 IS: hor=2  vert=4 TAB=N  tekst=DAI kar=HALLO
1040  REM m.b.v TAB BEGINT DE PRINTER TAB(40) VERDER
1050  END
```

BASICODE II

HET BASICODE MACHINETAAL PROGRAMMA

MET DIT PROGRAMMA IS HET MOGELIJK PROGRAMMA'S MET ANDERE COMPUTERS UIT TE WISSELEN IN BASICODE FORMAAT.
HET PROGRAMMA WORDT GELADEN DOOR ACHTEREENVOLGENS IN TE TOETSEN :'UT','R' EN RETURN. NA HET LADEN TOETST U 'B' DAN IS DE COMPUTER TERUG IN DE BASIC MODE.
HET PROGRAMMA WORDT GESTART DOOR CALLM #300.
DAN IS ER KEUS UIT 7 MOGELIJKHEDEN:

1 HET LADEN VAN EEN BASICODE PROGRAMMA: HIERBIJ WORDT AUTOMATISCH HET MID$-STATEMENT GECORRIGEERD (DEZE IS BIJ DE DAI AFWIJKEND VAN ANDERE COMPUTERS).
2 WEGSCHRIJVEN VANAF REGEL 1000 VAN HET AANWEZIGE PROGRAMMA. OOK HIER WORDT HET MID$-STATEMENT GECORRIGEERD;BOVENDIEN WORDEN DE SPATIES TUSSEN REGELNUMMER EN REGEL ONDERDRUKT.
3 HET LADEN VAN EEN BASICODE PROGRAMMA (ZONDER CORRECTIE).
4 WEGSCHRIJVEN VAN HET BASIC PROGRAMMA (ZONDER CORRECTIE).
5 WISSEN VAN HET AANWEZIGE BASIC PROGRAMMA EN HET (OPNIEUW) INVOEREN VAN DE BASICODE SUBROUTINES.
6 HET MERGEN VAN DE BASICODE SUBROUTINES MET HET AANWEZIGE PROGRAMMA.
7 HET STARTEN VAN HET (NIEUWE) BASICODE PROGRAMMA.

VOOR HET LADEN VAN EEN BASICODE PROGRAMMA,WORDT HET AANWEZIGE PROGRAMMA NAAR DE EDITBUFFER GESTUURD (DIT DUURT ENIGE TIJD). ALS DIT KLAAR IS VERSCHIJNT DE MEDEDELING 'TYPE SPACE'. DAN START U DE RECORDER EN DRUKT U OP DE SPATIEBALK ALS U DE FLUITTOON (= LEADER) HOORT.
DE REST GEBEURT AUTOMATISCH.
ALS HET PROGRAMMA GELADEN IS VERSCHIJNT EEN VAN DE VOLGENDE MEDEDELINGEN:
1 DATA DROPOUT ERROR (BIJ EEN DROPOUT OP DE BAND).
2 CHECKSUM ERROR (STORING OP DE BAND).
3 NO TAPE ERROR (GEEN BANDFOUT).
OP DIT MOMENT WORDT DE EDITBUFFER GELEEGD,WAARNA HET PROGRAMMA GEREED VOOR GEBRUIK IS. MOGELIJK ZIJN ER ENKELE KLEINE FOUTJES BV SYNTAX ERROR,OVERFLOW ETC.DEZE ZIJN OP DE GEBRUIKELIJKE MANIER IN DE EDITMODE TE VERBETEREN.

VOOR HET WEGSCHRIJVEN VAN HET PROGRAMMA,WORDT DIT VERPLAATST NAAR EEN BUFFER. DIT DUURT ENIGE TIJD (AFHANKELIJK VAN DE LENGTE VAN HET PROGRAMMA),WAARNA DE MEDEDELING 'SET RECORD,START TAPE TYPE SPACE'VERSCHIJNT.
NU DRUKT U OP DE SPATIEBALK,WAARNA DE BASICODE UIT DE COMPUTER (CASSETTE INTERFACE) KOMT.

BASICODE II

THE BASICODE MACHINE LANGUAGE PROGRAM.

WITH THIS PROGRAM IT IS POSSIBLE TO EXCHANGE PROGRAMS WITH OTHER COMPUTERS.
THIS PROGRAM CAN BE LOADED BY KEYING (FROM BASIC):
'UT','R' AND RETURN . WHEN THIS IS DONE TYPE 'B'
THE COMPUTER IS THEN BACK IN BASIC MODE.
THIS TRANSLATION PROGRAM IS STARTED BY CALLM #300.
THERE IS THEN A CHOICE OF 7 POSSIBILITIES:

1 LOADING A BASICODE PROGRAM FROM TAPE:HERE,THE MID$-
  STATEMENT WILL BE AUTOMATICALLY CORRECTED (THIS
  STATEMENT IS DIFFERENT FROM OTHER COMPUTERS).
2 SAVING FROM LINE 1000 OF THE PRESENT BASICODE PROGRAM:
  ALSO HERE THE MID$-STATEMENT WILL BE CORRECTED;BESIDES,THE
  SPACES BETWEEN LINE NUMBER AND LINE WILL BE SUPPRESSED.
3 LOADING OF THE BASICODE PROGRAM (WITHOUT ANY CORRECTION).
4 SAVING OF THE TOTAL PROGRAM (WITHOUT ANY CORRECTION).
5 CANCELLING THE PRESENT PROGRAM AND (RE)STORING THE
  BASICODE SUBROUTINES.
6 MERGING THE BASICODE SUBROUTINES WITH THE PRESENT
  PROGRAM.
7 STARTING THE (NEW) BASICODE PROGRAM.

BEFORE LOADING A BASICODE PROGRAM, THE PRESENT PROGRAM
WILL BE SENT TO THE EDIT BUFFER (THIS TAKES SOME TIME).
WHEN THIS IS READY THE ANNOUNCEMENT 'TYPE SPACE' APPEARS.
THEN YOU CAN START THE RECORDER AND TYPE SPACE WHEN THE
LEADER IS HEARD.THE REST TAKES PLACE AUTOMATICALLY.
WHEN THIS ALL IS DONE AND LOADED THE FOLLOWING ANNOUNCE-
MENTS CAN APPEAR:
1 DATA DROPOUT ERROR (IF THERE IS A DROPOUT ON TAPE).
2 CHECKSUM ERROR (FAILURE ON TAPE).
3 NO TAPE ERROR (IF EVERYTHING ON TAPE IS OK).
AT THIS MOMENT THE EDITBUFFER WILL BE EMPTIED AND AFTER
THIS THE PROGRAM IS READY FOR USE.THERE ARE POSSIBLY
A FEW SMALL ERRORS SUCH AS E.G. SYNTAX ERROR ,OVERFLOW ETC.
THESE WILL BE CORRECTABLE AS USUAL IN THE EDITMODE.

BEFORE SAVING THE PROGRAM, IT WILL BE MOVED INTO A BUFFER.
THIS ALSO TAKES SOME TIME (DEPENDING ON THE LENGTH OF
PROGRAM), WHEREAFTER THE ANNOUNCEMENT 'SET RECORD ,START
TAPE AND TYPE SPACE'APPEARS. THE SPACE KEY SHOULD NOW BE
PRESSED IN. THE BASICODE PROGRAM THEN COMES OUT OF THE
COMPUTER (CASSETTE INTERFACE).

```
002                     POPALL  EQU   :C14D     POP ALL REGISTERS AND RETURN
003                     CASST   EQU   :D42E     START CASSETTE MOTORS
004                     CASSP   EQU   :D445     STOP CASSETTE MOTORS
005                     TIMER   EQU   :D54B     PRECISION TIMER TIME IN L
006                     WSPACE  EQU   :D6DA
007                     SRSTTS  EQU   :DAFF     STRT TAPE,SET REC,TYPE SPACE
008                     PSTR    EQU   :DB32     PRINT A STRING
009                     FILLBU  EQU   :DD75     FILL EDIT BUFFER
010                     ALFA    EQU   :DE02     TEST A-Z
011                     COMPAR  EQU   :DE14     COMPARE  DE - HL
012                     NEW     EQU   :DEB8
013                     LIEDBU  EQU   :E19F     LIST IN EDIT BUFFER
014                     DELEDI  EQU   :E228     MOVE EDIT BUFFER TO BASIC
015                     INITED  EQU   :E26C     INIT EDITMODE
016                     CURSOR  EQU   :0075
017                     EDBBGN  EQU   :00A2     STARTADDRESS EDITBUFFER
018                     EDBPTR  EQU   :00A4     (EDIT) INPUT POINTER
019                     EDBMAX  EQU   :00A6     END OF EDITBUFFER
020                     INSW    EQU   :296
021                     INPMEM  EQU   :2EC      CASS.INP.MEMORY
022                     CHKSUM  EQU   :2ED      CHECKSUM
023                     ENDFLG  EQU   :2EE
024                     BASTAB  EQU   :2EF
025                     BTPTR   EQU   :2F1
026                     MID$FG  EQU   :2F3
027                     PTRIN   EQU   :2F4
028                     PTROUT  EQU   :2F6
029                     OLDPTR  EQU   :2F8
030                     STRPTR  EQU   :2FA
031                     PORMEM  EQU   :2FC
032                     TMPPTR  EQU   :2FD
033                     SPCFLG  EQU   :2FF
034                     INPORT  EQU   :FD00     CASSETTE INPUT PORT
035                     OUTPRT  EQU   :FD06     CASSETTE OUTPUT PORT
036                             ORG   :300
037 0300 C32C07     BASTRT  JMP   SLOT
038 0303 C36703     READ    JMP   R         READ BASICODE
039 0306 C37203     READCO  JMP   RCO       READ W MID$ CORR
040 0309 C35103     WRITE   JMP   W         WRITE BASICODE
041 030C C35C03     WRITCO  JMP   WCO       WRITE W MID$&SPACE CORR
042 030F C35706     WFINAL  JMP   BUFEND
043 0312 C3F005     HORSEW  JMP   PRESUB    HORSEW+1=POINTER
044 0315 C36604     HORSER  JMP   FIN       HORSER+1=POINTER
045 0318 CDB8DE     NE&BAS  CALL  NEW       CANCEL WHOLE PROGRAM
046 031B C5         PREBAS  PUSH  B         RESTORE BASIC PART
047 031C D5                 PUSH  D
048 031D E5                 PUSH  H
049 031E F3                 DI
050 031F 213303             LXI   H,BASIC
051 0322 22E102             SHLD  :2E1
052 0325 AF                 XRA   A
053 0326 3C                 INR   A
054 0327 329602             STA   INSW
055 032A 2AEF02             LHLD  BASTAB    START OF BASICODE PROGR
056 032D 22F102             SHLD  BTPTR
057 0330 C33603             JMP   Y
058 0333 C5         BASIC   PUSH  B
059 0334 D5                 PUSH  D
060 0335 E5                 PUSH  H
061 0336 2AF102     Y       LHLD  BTPTR
062 0339 7E                 MOV   A,M
```

```
063 033A 23                 INX   H

PAGE 02    NEW BASICODE  TH.V.LIESHOUT,WOGNUM,NL

064 033B 22F102             SHLD  BTPTR
065 033E FE00               CPI   :00
066 0340 C24D03             JNZ   POPEXP
067 0343 329602             STA   INSW
068 0346 21B4DD             LXI   H,:DDB4
069 0349 22E102             SHLD  :2E1
070 034C FB                 EI
071 034D E1          POPEXP POP   H
072 034E D1                 POP   D
073 034F C1                 POP   B
074 0350 C9                 RET
075 0351 E5          W      PUSH  H
076 0352 21F005             LXI   H,PRESUB
077 0355 221303             SHLD  HORSEW+1
078 0358 E1                 POP   H
079 0359 C38405             JMP   CSTWRT
080 035C E5          WCO    PUSH  H
081 035D 21B905             LXI   H,SPCABS
082 0360 221303             SHLD  HORSEW+1
083 0363 E1                 POP   H
084 0364 C38405             JMP   CSTWRT
085 0367 E5          R      PUSH  H
086 0368 216604             LXI   H,FIN      WITHOUT MID$-
087 036B 221603             SHLD  HORSER+1   CORRECTION
088 036E E1                 POP   H
089 036F C37A03             JMP   INIT
090 0372 E5          RCO    PUSH  H
091 0373 214904             LXI   H,COR      WITH MID$-
092 0376 221603             SHLD  HORSER+1   CORRECTION
093 0379 E1                 POP   H
094 037A F5          INIT   PUSH  PSW
095 037B C5                 PUSH  B
096 037C D5                 PUSH  D
097 037D E5                 PUSH  H
098 037E 3EFF               MVI   A,:FF      *PLACE BLACK SQUARE
099 0380 327500             STA   CURSOR     *ON CURSOR
100 0383 CD6CE2             CALL  INITED     INIT EDITBUFFER
101 0386 CD9FE1             CALL  LIEDBU     LIST IN EDIT BUFFER
102 0389 AF                 XRA   A
103 038A 323101             STA   :131       RESET OUTPUT POINTER
104 038D 32EE02             STA   ENDFLG     RESET ENDFLG
105 0390 CD2ED4             CALL  CASST      CASS.START ROUTINE IN DAI
106 0393 215305             LXI   H,MESTYS   MESSAGE 'TYPE SPACE'
107 0396 CD32DB             CALL  PSTR
108 0399 CDDAD6             CALL  WSPACE     WAIT FOR SPACEBAR PRESSED
109 039C 216005             LXI   H,MESLDB   MESSAGE 'LOADING BASICODE'
110 039F CD32DB             CALL  PSTR
111 03A2 CD2ED4             CALL  CASST      CASS.START ROUTINE IN DAI
112 03A5 2100FD      START  LXI   H,INPORT   ON FIRST TIME (ON START OF
113                  *                       PROGRAM)WAIT FOR CHANGING
114                  *                       INPUT.NO CHECK FOR TIME.
115 03A8 4E                 MOV   C,M
116 03A9 1E00        STARTS MVI   E,:00
117 03AB 7E                 MOV   A,M
118 03AC A9                 XRA   C
119 03AD F2AB03             JP    STARTS+2
120 03B0 F3                 DI
121 03B1 4E                 MOV   C,M
122 03B2 1C          WAITFL INR   E           WAIT FOR AN EDGE
123                  *                       ON THE INPUT PORT
124 03B3 CA1404             JZ    ERROR      TOO LONG ERROR
```

```
125 03B6 7E                     MOV   A,M

PAGE 03    NEW BASICODE  TH.V.LIESHOUT,WOGNUM,NL

126 03B7 A9                     XRA   C
127 03B8 F2B203                 JP    WAITFL     WAIT FOR CHANGING INPUT
128 03BB 4E                     MOV   C,M
129 03BC 1E78         STRTBT    MVI   E,:78
130 03BE 2E0D                   MVI   L,:0D      ******************************
131 03C0 CD4BD5                 CALL  TIMER
132 03C3 1C           COUNT1    INR   E
133 03C4 CA1404                 JZ    ERROR      TOO LONG ERROR
134 03C7 7E                     MOV   A,M
135 03C8 A9                     XRA   C
136 03C9 F2C303                 JP    COUNT1
137 03CC 4E                     MOV   C,M
138 03CD 1C                     INR   E
139 03CE F2BC03                 JP    STRTBT     TOO SHORT FOR STARTBIT
140 03D1 1E8A                   MVI   E,:8A
141 03D3 06FF                   MVI   B,:FF
142 03D5 C30905                 JMP   SB
143 03D8 7B           BYTE      MOV   A,E
144 03D9 C620                   ADI   :20        CARRY SET :BIT='1'
145 03DB 78                     MOV   A,B
146 03DC 1F                     RAR              SHIFT
147 03DD 47                     MOV   B,A
148 03DE DAF804                 JC    BIT        CARRY SET = NO STARTBIT
149                   *                          SHIFTED IN CARRY
150 03E1 FB           CSUM      EI
151 03E2 3AEE02                 LDA   ENDFLG
152 03E5 B7                     ORA   A
153 03E6 C21E04                 JNZ   STOTEX     IF ENDFLAG SET
154 03E9 3AED02                 LDA   CHKSUM
155 03EC A8                     XRA   B          MODIFY CHECKSUM
156 03ED 32ED02                 STA   CHKSUM
157 03F0 78                     MOV   A,B
158 03F1 E67F                   ANI   :7F
159 03F3 FE03                   CPI   :03        ASCII 3 = END OF TEXT
160 03F5 CA0804                 JZ    SETFLG
161 03F8 FE02                   CPI   :02        CLEAR CHECCKSUM
162 03FA CA0E04                 JZ    CLCHSM
163 03FD CD75DD       RETURN    CALL  FILLBU     STORE CHARACTER IN
164                   *                          (EDIT)BUFFER AND MODIFY
165                   *                          POINTER
166 0400 3E20                   MVI   A,:20
167 0402 327500                 STA   CURSOR
168 0405 C3A503                 JMP   START
169 0408 32EE02       SETFLG    STA   ENDFLG     SET ENDFLAG
170 040B C3A503                 JMP   START
171 040E 32ED02       CLCHSM    STA   CHKSUM
172 0411 C3A503                 JMP   START
173 0414 FB           ERROR     EI
174 0415 214005                 LXI   H,MESERR   MESSAGE 'DATA DROPOUT ERROR'
175 0418 CD32DB                 CALL  PSTR       PRINT MESSAGE
176 041B C33604                 JMP   LAST
177 041E 3AED02       STOTEX    LDA   CHKSUM
178 0421 90                     SUB   B
179 0422 E67F                   ANI   :7F
180 0424 CA3004                 JZ    GOOD
181 0427 212205       FALSE     LXI   H,MESFAL   MESSAGE 'CHECKSUM ERROR'
182 042A CD32DB                 CALL  PSTR       PRINT MESSAGE
183 042D C33604                 JMP   LAST
184 0430 213105       GOOD      LXI   H,MESGD    MESSAGE ' OK '
185 0433 CD32DB                 CALL  PSTR       PRINT MESSAGE
186 0436 3E0D         LAST      MVI   A,:0D
```

```
187 0438 EF                RST   5

PAGE 04    NEW BASICODE  TH.V.LIESHOUT,WOGNUM,NL

188 0439 03                DATA  :3
189 043A 3E0D              MVI   A,:0D
190 043C EF                RST   5
191 043D 03                DATA  3
192 043E 3E5F              MVI   A,:5F
193 0440 327500            STA   CURSOR     ' - 'IN CURSOR
194 0443 CD45D4            CALL  CASSP      STOP CASSETTE MOTORS
195 0446 C31503            JMP   HORSER
196 0449 0600        COR   MVI   B,:00
197 044B 217505            LXI   H,MID$
198 044E 22FA02            SHLD  STRPTR
199 0451 2AA400            LHLD  EDBPTR     EDIT INPUT POINTER
200 0454 22F802            SHLD  OLDPTR     STORE IN POINTER
201 0457 EB                XCHG
202 0458 2AA200            LHLD  EDBBGN     START OF EDITBUFFER
203 045B 7E          CHKWRD MOV   A,M
204 045C 23                INX   H
205 045D CD7004            CALL  WDRCZR     CALL WORDRECOGNIZER
206 0460 CD14DE            CALL  COMPAR     END OF EDITBUFFER?
207 0463 C25B04            JNZ   CHKWRD
208 0466 AF          FIN   XRA   A          A=00
209 0467 CD75DD            CALL  FILLBU     LAST BYTE IN BUFFER
210                  *                      (MUST BE A '00')
211 046A CD28E2            CALL  DELEDI     DELETE EDIT BUFFER
212 046D C34DC1            JMP   POPALL     POP REGISTERS AND RETURN
213                  *
214 0470 E5          WDRCZR PUSH  H         SUBROUTINE WORDRECOGNIZER
215                  *                      TO RECOGNIZE THE MID$-
216                  *                      STATEMENT
217 0471 FE0D              CPI   :0D        END OF ROW
218 0473 CA9B04            JZ    CKM$FG     CHECK MID$FLAG (IN ROW)
219 0476 FE20              CPI   :20        SPACE
220 0478 CA9604            JZ    STORE      NO FURTHER ACTION
221 047B 4F                MOV   C,A        C=CHARACTER
222 047C 78                MOV   A,B        B=COUNTER
223 047D FE05              CPI   :5
224 047F CAAD04            JZ    COMMA      5 COUNTS AND A COMMA MEANS
225                  *                      FILL WITH  '- 1+'
226                  *                      E.G. MID$("ABC",-1+N,2)
227 0482 79          NCOMMA MOV   A,C
228 0483 2AFA02            LHLD  STRPTR     NO COMMA - STRINGPOINTER
229 0486 BE                CMP   M
230 0487 CAD004            JZ    INRB       INCREMENT STRINGPOINTER
231 048A 78                MOV   A,B
232 048B FE05              CPI   :5
233 048D FADC04            JM    CLRB
234 0490 79                MOV   A,C
235 0491 FE29              CPI   :29        ')' EG MID$(A$(0,5),N,2)
236 0493 CAD804            JZ    DCRB
237 0496 CD75DD      STORE CALL  FILLBU
238 0499 E1                POP   H
239 049A C9                RET
240                  *
241 049B 3AF302      CKM$FG LDA  MID$FG     MID$FLAG MEANS THIS ROW
242                  *                      CONTAINS A STATEMENT 'MID$'
243 049E B7                ORA   A
244 049F C2E804            JNZ   ENCODE
245 04A2 2AF802            LHLD  OLDPTR     NO MID$ STATEMENT  OLD POINT
246 04A5 22A400            SHLD  EDBPTR     IN EDITBUFFERPOINTER
247 04A8 0600              MVI   B,:00
248 04AA C39904            JMP   STORE+3
```

```
249 04AD 79          COMMA   MOV   A,C

PAGE 05    NEW BASICODE  TH.V.LIESHOUT,WOGNUM,NL

250 04AE FE2C                CPI   :2C       CHECK','
251 04B0 C28204              JNZ   NCOMMA
252 04B3 32F302              STA   MID$FG    SET MID$FLAG
253 04B6 217505              LXI   H,MID$
254 04B9 22FA02              SHLD  STRPTR
255 04BC 0600                MVI   B,:00
256 04BE CD75DD              CALL  FILLBU
257 04C1 3E2D                MVI   A,:2D
258 04C3 CD75DD              CALL  FILLBU
259 04C6 3E31                MVI   A,:31
260 04C8 CD75DD              CALL  FILLBU
261 04CB 3E2B                MVI   A,:2B
262 04CD C39604              JMP   STORE
263                  *
264 04D0 23          INRB    INX   H
265 04D1 04                  INR   B
266 04D2 22FA02              SHLD  STRPTR
267 04D5 C39604              JMP   STORE
268 04D8 05          DCRB    DCR   B
269 04D9 C39604              JMP   STORE
270 04DC 217505      CLRB    LXI   H,MID$    DEFAULT VALUE $POINTER
271 04DF 22FA02              SHLD  STRPTR
272 04E2 0600                MVI   B,:00
273 04E4 79                  MOV   A,C
274 04E5 C39604              JMP   STORE
275 04E8 AF          ENCODE  XRA   A
276 04E9 32F302              STA   MID$FG    CLEAR MID$FLAG
277 04EC 2AA400              LHLD  EDBPTR
278 04EF 23                  INX   H
279 04F0 22F802              SHLD  OLDPTR    POINTER INCL CARR RET
280 04F3 3E0D                MVI   A,:0D
281 04F5 C39604              JMP   STORE
282                  *
283 04F8 1E7A        BIT     MVI   E,:7A
284 04FA 4E          BITSEC  MOV   C,M       SECOND TIME (=SECOND CYCLE
285                  *                       OF 2400 HZ)
286 04FB 2E0D                MVI   L,:0D     ****************************
287 04FD CD4BD5              CALL  TIMER
288 0500 1C          COUNT2  INR   E
289 0501 CA1404              JZ    ERROR
290 0504 7E                  MOV   A,M
291 0505 A9                  XRA   C
292 0506 F20005              JP    COUNT2
293 0509 4E          SB      MOV   C,M
294 050A 2E0D                MVI   L,:0D     ****************************
295 050C CD4BD5              CALL  TIMER
296 050F 53                  MOV   D,E
297 0510 1D          COUNT3  DCR   E
298 0511 CA1404              JZ    ERROR
299 0514 7E                  MOV   A,M
300 0515 A9                  XRA   C
301 0516 F21005              JP    COUNT3
302 0519 14                  INR   D
303 051A FAD803              JM    BYTE
304 051D 1EEF                MVI   E,:EF
305 051F C3FA04              JMP   BITSEC
306 0522 0E          MESFAL  DATA  :0E       LENGTH OF STRING
307 0523 434845              ASC   'CHECKSUM ERROR'
308 0531 0E          MESGD   DATA  :0E
309 0532 4E4F20              ASC   'NO TAPE ERRORS'
310 0540 12          MESERR  DATA  :12
```

```
311 0541 444154              ASC   'DATA DROPOUT ERROR'


PAGE 06    NEW BASICODE  TH.V.LIESHOUT,WOGNUM,NL

312 0553 0C          MESTYS  DATA  :0C
313 0554 0C0D                DATA  :0C,:0D
314 0556 545950              ASC   'TYPE SPACE'
315 0560 14          MESLDB  DATA  :14
316 0561 0C0D                DATA  :0C,:0D
317 0563 4C4F41              ASC   'LOADING BASICODE'
318 0573 0D0D                DATA  :0D,:0D
319 0575 4D4944      MID$    ASC   'MID$((((((((((('
320                  *       **********************
321                  *       * BASICODE WRITE PART *
322                  *       **********************
323 0584 F5          CSTWRT  PUSH  PSW        COLD START OF BASICODE
324 0585 C5                  PUSH  B          WRITE ROUTINE
325 0586 D5                  PUSH  D
326 0587 E5                  PUSH  H
327 0588 F3                  DI
328 0589 2AA302              LHLD  :2A3       END OF SYMBOL TABLE
329 058C 24                  INR   H
330 058D 22F402              SHLD  PTRIN      SET INPUT &
331 0590 22F602              SHLD  PTROUT     OUTPUT POINTER
332 0593 AF                  XRA   A
333 0594 32FF02              STA   SPCFLG     RESET SPACEFLAG
334 0597 3EC3                MVI   A,:C3      *SET POINTER TO
335 0599 32DD02              STA   :2DD       *(WARM)START OF
336 059C 21B205              LXI   H,SWRITE   *WRITEROUTINE
337 059F 22DE02              SHLD  :2DE       *
338 05A2 3E03                MVI   A,:03      OUTPUT SWITCH
339 05A4 323101              STA   :131
340 05A7 FB                  EI
341 05A8 21ED02              LXI   H,CHKSUM   ALSO USED BY READROUTINE
342 05AB 3E82                MVI   A,:82      START OF TEXT (02)
343                  *                        8TH BIT MUST BE A '1'
344 05AD 3683                MVI   M,:83      CHECKSUM INCLUSIVE
345                  *                        END OF TEXT (03)
346 05AF C3B605              JMP   SWRITE+4   FIRST TIME NO PUSHING
347 05B2 F5          SWRITE  PUSH  PSW
348 05B3 C5                  PUSH  B
349 05B4 D5                  PUSH  D
350 05B5 E5                  PUSH  H
351 05B6 C31203              JMP   HORSEW
352                  * SUBROUTINE TO ABSORB THE SPACES
353                  * BETWEEN LINENUMBER AND LINE
354 05B9 FE0D        SPCABS  CPI   :0D        RETURN
355 05BB CC3A06              CZ    RSFLAG     RESET FLAG
356 05BE CD02DE              CALL  ALFA       TEST A-Z
357 05C1 DC4706              CC    STFLAG     SET FLAG
358 05C4 FE20                CPI   :20        SPACE?
359 05C6 CA4B06              JZ    ABSORB
360 05C9 2AFD02              LHLD  TMPPTR
361 05CC 6F                  MOV   L,A        L =CHARACTER
362 05CD 7C                  MOV   A,H        H=STRINGCOUNTER
363 05CE FE05                CPI   :5
364 05D0 CA0806              JZ    WCOMMA
365 05D3 EB          WNCOMA  XCHG
366 05D4 2AFA02              LHLD  STRPTR
367 05D7 7B                  MOV   A,E        E=CHARACTER
368 05D8 BE                  CMP   M
369 05D9 CA2906              JZ    INCREG     INCREMENT COUNTER
370 05DC 7A                  MOV   A,D        D=STRINGCOUNTERC
371 05DD FE05                CPI   :5
372 05DF FA3206              JM    CLRREG     CLEAR COUNTER
```

```
373 05E2 7B                     MOV   A,E

PAGE 07    NEW BASICODE  TH.V.LIESHOUT,WOGNUM,NL

374 05E3 FE29                   CPI   :29
375 05E5 CA2E06                 JZ    DECREG    DECREMENT COUNTER
376 05E8 22FA02     WSTORE      SHLD  STRPTR
377 05EB EB                     XCHG
378 05EC 22FD02                 SHLD  TMPPTR
379 05EF 7D                     MOV   A,L
380 05F0 CDF605     PRESUB      CALL  SUB
381 05F3 C34DC1                 JMP   POPALL    RETURN
382 05F6 F680       SUB         ORI   :80
383 05F8 47                     MOV   B,A       SAVE CHARACTER IN B
384 05F9 21ED02                 LXI   H,CHKSUM  OLD CHECKSUM
385 05FC AE                     XRA   M         MODIFY
386 05FD 77                     MOV   M,A       STORE CHECKSUM
387 05FE 2AF402                 LHLD  PTRIN     INPUT POINTER
388 0601 70                     MOV   M,B       STORE CHARACTER
389 0602 23                     INX   H         INCREMENT /
390 0603 22F402                 SHLD  PTRIN     MODIFY POINTER
391 0606 78                     MOV   A,B       RESTORE CHARACTER IN ACCU
392 0607 C9                     RET
393 0608 7D         WCOMMA      MOV   A,L
394 0609 FE2C                   CPI   :2C       CHECK','
395 060B C2D305                 JNZ   WNCOMA    NO COMMA
396 060E 217505                 LXI   H,MID$
397 0611 22FA02                 SHLD  STRPTR
398 0614 3E2C                   MVI   A,:2C     (=',')
399 0616 CDF605                 CALL  SUB
400 0619 3E20                   MVI   A,:20     (='-')
401 061B CDF605                 CALL  SUB
402 061E 3E31                   MVI   A,:31     (='1')
403 0620 CDF605                 CALL  SUB
404 0623 212B00                 LXI   H,:002B   H=STRINGCOUNTER(=00)
405 0626 C3D305                 JMP   WNCOMA    L=CHARACTER(2B='+')
406 0629 14         INCREG      INR   D
407 062A 23                     INX   H
408 062B C3E805                 JMP   WSTORE
409 062E 15         DECREG      DCR   D
410 062F C3E805                 JMP   WSTORE
411 0632 217505     CLRREG      LXI   H,MID$
412 0635 1600                   MVI   D,:00
413 0637 C3E805                 JMP   WSTORE
414 063A F5         RSFLAG      PUSH  PSW
415 063B AF                     XRA   A
416 063C 32FF02                 STA   SPCFLG
417 063F F1                     POP   PSW
418 0640 217505                 LXI   H,MID$
419 0643 22FA02                 SHLD  STRPTR
420 0646 C9                     RET
421 0647 32FF02     STFLAG      STA   SPCFLG
422 064A C9                     RET
423 064B 3AFF02     ABSORB      LDA   SPCFLG
424 064E B7                     ORA   A
425 064F 3E20                   MVI   A,:20
426 0651 CA4DC1                 JZ    POPALL
427 0654 C3F005                 JMP   PRESUB
428 0657 F5         BUFEND      PUSH  PSW       BUFFER END
429 0658 C5                     PUSH  B
430 0659 D5                     PUSH  D
431 065A E5                     PUSH  H
432 065B 213101                 LXI   H,:131    OUTPUT SWITCH
433 065E 3600                   MVI   M,:00     BACK
434 0660 21ED02                 LXI   H,CHKSUM
```

```
435 0663 46                    MOV   B,M        CHECKSUM IN B

PAGE 08    NEW BASICODE  TH.V.LIESHOUT,WOGNUM,NL

436 0664 2AF402                LHLD  PTRIN
437 0667 3683                  MVI   M,:83      END OF TEXT IN BUFFER
438 0669 23                    INX   H          INCREMENT A POINTER
439 066A 70                    MOV   M,B        CHECKSUM IN BUFFER
440 066B 22F402                SHLD  PTRIN
441 066E CDFFDA                CALL  SRSTTS     PRINT SET RECORD ,START
442                    *                        TAPE TYPE SPACE
443 0671 9CDB                  DBL   :DB9C
444 0673 CDDAD6                CALL  WSPACE     WAIT FOR SPACEBAR/CR
445 0676 CD2ED4                CALL  CASST      START CASSETTE MOTORS
446 0679 F3                    DI
447                   * THE TIMING IS CALCULABLE :
448                   * FOR PROGRAM IN RAM : EVERY STATE ~1uS
449                   * FOR PROGRAM IN ROM : EVERY STATE 0.5uS
450                   * EXPLANATION :THE VIDEO PROCESSOR USES ALSO
451                   * THE RAM IN TURN:THAT MEANS A FACTOR 2 SLOWER
452                   * EXEPT WHEN SCREENSPOT GOES BACK FROM UNDERSIDE
453                   * TO UPPERSIDE OF THE SCREEN ,THE VIDEOPROCESSOR
454                   * IS NOT USING THE RAM.
455                   *
456                   *                        STATES   TIME AFTER LAST FALLING
457                   *                                 EDGE    ( uS )
458                   *                          *       *       *      *
459 067A 3A4000                LDA   :0040
460 067D F601                  ORI   :01
461 067F 32FC02                STA   PORMEM
462 0682 1106FD                LXI   D,OUTPRT   10
463 0685 010018                LXI   B,:1800    10
464                   *
465                   * TIMER   L=(208-T-42)/7.5
466                   *
467 0688 2E0C          LEADER  MVI   L,:0C      7              >58
468 068A CDF506                CALL  BITONE     17     >27      75>
469 068D 0B                    DCX   B          5       32
470 068E 78                    MOV   A,B        5       37
471 068F B7                    ORA   A          4       41
472 0690 C28806                JNZ   LEADER     10      51>
473 0693 2E0E                  MVI   L,:0E      7       46
474 0695 CDF506                CALL  BITONE     17      63>   >27
475 0698 2AF602        DATA    LHLD  PTROUT     16             43
476 069B D5                    PUSH  D          11             54
477 069C 0609                  MVI   B,:09      7              61
478 069E 7E                    MOV   A,M        7              68
479 069F 2E07                  MVI   L,:07      7              75
480 06A1 F5            BITW    PUSH  PSW        11             86     73
481 06A2 D41507                CNC   BITZER     11/17  >15     103>   >90
482 06A5 DCF506                CC    BITONE     11/17   26            >101
483 06A8 F1                    POP   PSW        10      36
484 06A9 1F                    RAR              4       40
485 06AA 05                    DCR   B          5       45
486 06AB 2E09                  MVI   L,:09      7       52
487 06AD C2A106                JNZ   BITW       10      62>
488 06B0 2AF602                LHLD  PTROUT     16      78
489 06B3 23                    INX   H          5       83
490 06B4 22F602                SHLD  PTROUT     16      99
491 06B7 2B                    DCX   H          5      104
492 06B8 E5                    PUSH  H          11     115
493 06B9 2E03                  MVI   L,:03      7      122
494 06BB 00                    NOP              4      126
495 06BC CDF506                CALL  BITONE     17     143>
496 06BF E1                    POP   H          10      >37
```

```
497 06C0 EB                     XCHG            4       41


PAGE 09    NEW BASICODE  TH.V.LIESHOUT,WOGNUM,NL

498 06C1 2AF402                 LHLD    PTRIN   16      58
499 06C4 7C                     MOV     A,H     5       63
500 06C5 BA                     CMP     D       7       70
501 06C6 7D                     MOV     A,L     5       75
502 06C7 C2CB06                 JNZ     POPD    10      85>
503 06CA BB                     CMP     E       7       92
504 06CB D1             POPD    POP     D       10      102
505 06CC F5                     PUSH    PSW     11      113
506 06CD 2E04                   MVI     L,:04   7       120
507 06CF CDF506                 CALL    BITONE  17      137>
508 06D2 F1                     POP     PSW     10      >37
509 06D3 C29806                 JNZ     DATA    10       47
510 06D6 01000F                 LXI     B,:0F00 10       57
511 06D9 2E0B                   MVI     L,:0B   7        64
512 06DB CDF506         TRAILR  CALL    BITONE  17       81>   >75>
513 06DE 0B                     DCX     B       5    >32
514 06DF 78                     MOV     A,B     5     37
515 06E0 B7                     ORA     A       4     41
516 06E1 2E0C                   MVI     L,:0C   7     48
517 06E3 C2DB06                 JNZ     TRAILR  10    58>
518 06E6 3EC9                   MVI     A,:C9
519 06E8 32DD02                 STA     :2DD
520 06EB 210000                 LXI     H,:0000
521 06EE 22DE02                 SHLD    :2DE
522 06F1 FB                     EI
523 06F2 C34DC1                 JMP     POPALL
524 06F5 CD4BD5         BITONE  CALL    TIMER
525 06F8 3AFC02                 LDA     PORMEM
526 06FB 12                     STAX    D
527 06FC 3D                     DCR     A
528 06FD 2E16                   MVI     L,:16
529 06FF CD4BD5                 CALL    TIMER
530 0702 12                     STAX    D
531 0703 3C                     INR     A
532 0704 2E16                   MVI     L,:16
533 0706 CD4BD5                 CALL    TIMER
534 0709 12                     STAX    D
535 070A 3D                     DCR     A
536 070B 2E16                   MVI     L,:16
537 070D CD4BD5                 CALL    TIMER
538'0710 12                     STAX    D
539 0711 00                     NOP
540 0712 00                     NOP
541 0713 00                     NOP
542 0714 C9                     RET
543 0715 00             BITZER  NOP
544 0716 00                     NOP
545 0717 00                     NOP
546 0718 CD4BD5                 CALL    TIMER
547 071B 2E1A                   MVI     L,:1A
548 071D CD4BD5                 CALL    TIMER
549 0720 3AFC02                 LDA     PORMEM
550 0723 12                     STAX    D
551 0724 3D                     DCR     A
552 0725 2E32                   MVI     L,:32
553 0727 CD4BD5                 CALL    TIMER
554 072A 12                     STAX    D
555 072B C9                     RET
556 072C E5             SLOT    PUSH    H
557 072D 21800E                 LXI     H,:0E80
558 0730 229B02                 SHLD    :29B
```

```
559 0733 E1                 POP   H


PAGE 10    NEW BASICODE  TH.V.LIESHOUT,WOGNUM,NL

560 0734 C31803             JMP   NE&BAS
561 0737            END     END             END

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

ABSORB 064B   ALFA   DE02   BASIC  0333   BASTAB 02EF
BASTRT 0300   BIT    04F8   BITONE 06F5   BITSEC 04FA
BITW   06A1   BITZER 0715   BTPTR  02F1   BUFEND 0657
BYTE   03D8   CASSP  D445   CASST  D42E   CHKSUM 02ED
CHKWRD 045B   CKM$FG 049B   CLCHSM 040E   CLRB   04DC
CLRREG 0632   COMMA  04AD   COMPAR DE14   COR    0449
COUNT1 03C3   COUNT2 0500   COUNT3 0510   CSTWRT 0584
CSUM   03E1   CURSOR 0075   DATA   0698   DCRB   04D8
DECREG 062E   DELEDI E228   EDBBGN 00A2   EDBMAX 00A6
EDBPTR 00A4   ENCODE 04E8   END    0737   ENDFLG 02EE
ERROR  0414   FALSE  0427   FILLBU DD75   FIN    0466
GOOD   0430   HORSER 0315   HORSEW 0312   INCREG 0629
INIT   037A   INITED E26C   INPMEM 02EC   INPORT FD00
INRB   04D0   INSW   0296   LAST   0436   LEADER 0688
LIEDBU E19F   MESERR 0540   MESFAL 0522   MESGD  0531
MESLDB 0560   MESTYS 0553   MID$   0575   MID$FG 02F3
NCOMMA 0482   NE&BAS 0318   NEW    DEB8   OLDPTR 02F8
OUTPRT FD06   POPALL C14D   POPD   06CB   POPEXP 034D
PORMEM 02FC   PREBAS 031B   PRESUB 05F0   PSTR   DB32
PTRIN  02F4   PTROUT 02F6   R      0367   RCO    0372
READ   0303   READCO 0306   RETURN 03FD   RSFLAG 063A
SB     0509   SETFLG 0408   SLOT   072C   SPCABS 05B9
SPCFLG 02FF   SRSTTS DAFF   START  03A5   STARTS 03A9
STFLAG 0647   STORE  0496   STOTEX 041E   STRPTR 02FA
STRTBT 03BC   SUB    05F6   SWRITE 05B2   TIMER  D54B
TMPPTR 02FD   TRAILR 06DB   W      0351   WAITFL 03B2
WCO    035C   WCOMMA 0608   WDRCZR 0470   WFINAL 030F
WNCOMA 05D3   WRITCO 030C   WRITE  0309   WSPACE D6DA
WSTORE 05E8   Y      0336
```

```
10    REM
20    REM
30    REM ~~~~~~~~ NOTTE DI FUOCO IN MODE 1 ~~~~~~~~~~~~
40    REM
50    MODE 1:DIM A%(40.0)
52    FOR I%=0 TO 40:READ A%:POKE (#2F1+I%),A%:NEXT
57    DOT RND(XMAX),1 RND(16.0)
60    CALLM #2F1
70    GOTO 57
100   DATA #C5,#D5,#E5,#F5,#01,#D7,#BF,#11,#EF,#BF,#00
110   DATA #00,#00,#00,#00,#00,#00,#00,#00,#00,#26,#18
120   DATA #2E,#40,#0A,#12,#0B,#1B,#2D,#C2,#09,#03,#25
130   DATA #C2,#07,#03,#F1,#E1,#D1,#C1,#C9
```

COMMANDES DANS UN PROGRAMME

```
>tSPL V1.0  PAGE 1

0000          ;                  ****************************
0000          ;                  Commandes dans un programme
0000          ;                  Instructions  en  commandes
0000          ;                  ****************************
0000          ;       Grace a ce petit programme, vous pourrez
0000          ;       utiliser des Commandes dans un programme
0000          ;       et vice-versa.
0000          ;       Par exemple :   *100 RUN 10
0000          ;Mise en route :
0000          ;  Changer les vecteurs et taper le programme(#AE0-#AFA)
0000          ;  SI ce n'est deja fait
0000          ;  (*POKE #29C,11      *NEW)
0000          ;  Taper    *UT
0000          ;           >V1 C7E0-0AE0 (space+curseur gauche)
0000          ;           >B
0000          ;Mise hors service :
0000          ;  Taper    *UT
0000          ;           >Z3
0000          ;           >B
0000          ;
0000                      ORG       0AE0H          ; Compatible S.P.U
0AE0 E1                   POP H                    ; *****************
0AE1 F3                   DI                       ; * Recopie de    *
0AE2 224300               SHLD      43H            ; *   C70E a C714 *
0AE5 F5                   PUSH PSW                 ; *               *
0AE6 3EC0                 MVI A     0C0H           ; *****************
0AE8 E1                   POP H                    ; * Recopie de    *
0AE9 224100               SHLD      41H            ; *   C705 a C70A *
0AEC 67                   MOV H,A                  ; *               *
0AED AF                   XRA A                    ; *****************
0AEE E3                   XTHL                     ; Data apres RST 1
0AEF B6                   ORA M                    ; Si = 0 : Codage
0AF0 CCF80A               CZ        DATA0          ;
0AF3 E3                   XTHL                     ; Restore pile
0AF4 AF                   XRA A                    ; A=0
0AF5 C3CFC6               JMP       0C6CFH         ; Continue RST
0AF8 16C0     DATA0       MVI D     0C0H           ; Masque de test
0AFA C9                   RET                      ; pour
0AFB          ;                                    COMMAND INVALID
0AFB          ;
0AFB          ; AE0  E1 F3 22 43 00 F5 3E C0 E1 22 41 00 67 AF
0AFB          ; AEE  E3 B6 CC F8 0A E3 AF C3 CF C6 16 C0 C9
0AFB          ;
0AFB          ;Le test qui affiche le message d'erreur
0AFB          ;'COMMAND INVALID' se trouve aux adresses
0AFB          ; E035 ... de la banque 3 (Apres un RST1 DATA 0)
0AFB          ;Le test est fait comme suit :
0AFB                      ORG       0E035H
E035 7E                   MOV A,M                  ;Dans A code de l'instruction
E036 E6C0                 ANI       0C0H           ;On garde les deux derniers bits
E038 A2                   ANA D                    ;Masque de test
E039 3E18                 MVI A     18H
E03B CAF5D9               JZ        0D9F5H         ;message d'erreur
E03E          ;En imposant a D la valeur #C0 avant d'executer
E03E          ;ces instructions (En deviant le vecteur du RST1)
E03E          ;le jump au message d'erreur ne pourra plus se faire.
E03E          ;(Le resultat du ET n'etant jamais nul)
E03E                      END
```

Cédric
-DUFOUR-

CALLM

La fonction CALLM du basic peut etre appelée de deux façon :

1) CALLM nn
2) CALLM nn,var

Avec nn qui représente une adresse (éventuellement dans une variable) et var qui est le nom d'une variable
Dans les deux cas cértains renseignements sont placés dans les registres avant d'exécuter le sous-programme en langage machine. Ces renseignements sont donnés dans les deux cas çi-desous:

1) CALLM nn

* A contient la valeur #FF
* BC contient l'adresse de l'instruction qui suit le CALLM
* HL contient l'adresse nn

2) CALLM nn,var

* A contient une indication sur la nature de 'var'
  soit : #04 = FPT (Virgule flottante)
  #14 = INT (Entier)
  #22 = STR (Chaine)
* BC Idem a 1)
* HL contient l'adresse de 'var' soit:
  - Adresse de la variable si var est numérique
  - Adresse du pointeur de la variable (Vecteur) si var est une chaine

Il est à remarquer que seul le contenu de BC est important pour le BASIC. Vos sous programmes peuvent alors s'ecrire plus simplement.

```
Ex:   C5  -  PUSH B
      xx  -  ...Programme...
      C1  -  POP B
      C9  -  RET
```

Cédric DUFOUR

```
IMPINT
10    REM MONTAGNES
12    REM F.DEBROUWERE & C.DUFOUR
14    MODE 6:COLORG 15 9 12 1:Y1=101:Z1=130
18    FOR J=1 TO 55
20    A!=RND(6)-3:B!=RND(6)-3
22    FOR I=1 TO 6
24    X=X+1
26    Y=A!*I+Y1:IF Y<1 THEN Y=1
28    Z=B!*I+Z1:IF Z<1 THEN Z=1
30    DRAW X,Z X,Y 23:DRAW X,Z+2 X,Z 22
32    DRAW X,Y X,0 21:DRAW X,Y+2 X,Y 22
34    NEXT:Y1=Y:Z1=Z:NEXT
36    I=GETC:WAIT TIME 3:IF I=0 THEN 36
38    END
```

INTEGERS

```
*DNA
    ]#L.
    PAGE 01

    001                 ************************
    002                 *    I N T E G E R S   *
    003                 ************************
    004                 *
    005                 *
    006                 *    Dit  machinetaalprogramma zorgt  ervoor dat
    007                 *    INTEGER-variabelen  naar  rechts  geschikt
    008                 *    worden.  Dit in tegenstelling tot wat  DAI
    009                 *    personal computer doet.
    010                 *    Het kan aangeroepen worden vanuit BASIC met
    011                 *        CALLM#300,VARIABLE%.
    012                 *    De integervariabele wordt uitgeprint in een
    013                 *    formaat  van  6  cijfers  met  ervoor  een
    014                 *    spatie  voor  de  positieve  en een minteken
    015                 *    voor de negatieve integers.
    016                 *    Dit kan gewijzigd worden vanuit  BASIC  met
    017                 *        POKE #371,AANT%.
    018                 *    Hierbij  is  AANT%  minstens  even groot als
    019                 *    het aantal cijfers van het  langste  uit te
    020                 *    printen getal.
    021                 *    Als AANT% kleiner is dan het aantal cijfers
    022                 *    van het langste getal dan wordt het  BASIC-
    023                 *    programma onderbroken met vermelding :
    024                 *        "FORMAT ERROR".
    025                 *    Als  VARIABLE% = 0  dan  worden  er  alleen
    026                 *    spaties  ( aantal = AANT% + 1 )  geprint.
    027                 *    Dit kan  eveneens  gewijzigd  worden vanuit
    028                 *    BASIC met
    029                 *        POKE #370,#30   er  worden  een  aantal
    030                 *                        spaties en een  "0" ge-
    031                 *                        geprint.
    032                 *        POKE #370,#20   er worden enkel spaties
    033                 *                        geprint.
    034                 *
    035                 *    With   this   machinelanguageprogramm   the
    036                 *    INTEGERvariables  will  be  ordered  to the
    037                 *    right.
    038                 *    From BASIC called with
    039                 *        CALLM#300,VARIABLE%.
    040                 *    The maximum lenght  of the  integervariable
    041                 *    is  6.  This lenght  can  be  changed  from
    042                 *    BASIC with
    043                 *        POKE #371,NUMB%.
    044                 *        NUMB%  >=  lenght  of  longest  integer-
    045                 *                  variable
    046                 *    If  NUMB% < the lenght of the variable then
    047                 *    there will be  a BREAK  of the BASICprogram
    048                 *    and an ERROR report : "FORMAT ERROR".
    049                 *    If  VARIABLE% = 0  then only spaces will be
    050                 *    printed.
    051                 *    This can be changed by
    052                 *        POKE #370,#30 : print a "0"
    053                 *        POKE #370,#20 : print spaces
```

```
055                     *
056                     NUL     EQU   :370      *  #20=SPACE, #30=0
057                     AANT    EQU   :371      *  Number of figures to
058                     *                      *  printout
059                     XIBC    EQU   :C027     *  Convert integer for
060                     *                      *  OUTPUT
061                     PMSG    EQU   :DAD4     *  Print a message
062                     OUTCPR  EQU   :D695     *  OUTPUT character
063                     KBRFL   EQU   :2C4      *  BREAK flag
064                             ORG   :300
065 0300 C5             POUT    PUSH  B
066 0301 D5                     PUSH  D
067 0302 F5                     PUSH  PSW
068 0303 E7                     RST   :4        *  Convert integer variable
069 0304 0C                     DATA  :0C       *  for OUTPUT
070 0305 CD27C0                 CALL  XIBC      *
071 0308 217103                 LXI   H,AANT
072 030B 56                     MOV   D,M       *  Number of digits to print
073                     *                      *  in D
074 030C 14                     INR   D         *  +1
075 030D 3AF100                 LDA   :F1
076 0310 BA                     CMP   D
077 0311 DA2303                 JC    NEW
078 0314 218003                 LXI   H,MSG$
079 0317 CDD4DA                 CALL  PMSG      *  If D<F1 then print
080 031A 21C402                 LXI   H,KBRFL   *  message 'FORMAT ERROR'
081 031D 36FF                   MVI   M,:FF     *  and BREAK BASICprogram
082 031F F1                     POP   PSW
083 0320 D1                     POP   D
084 0321 C1                     POP   B
085 0322 C9                     RET
086 0323 3AF100         NEW     LDA   :F1       *  Lenght of the integer in
087                     *                      *  in #F1
088 0326 15                     DCR   D
089 0327 BA                     CMP   D
090 0328 CA3303                 JZ    PSTART
091 032B 3E20                   MVI   A,:20
092 032D CD95D6                 CALL  OUTCPR    *  OUTPUT character
093 0330 C32303                 JMP   NEW
094 0333 3AE400         PSTART  LDA   :E4       *  Sign of the integer
095 0336 FE2D                   CPI   :2D       *  if #E4="-" then CONT 2
096 0338 C23E03                 JNZ   CONT1     *  if #E4=space then CONT 1
097 033B C34003                 JMP   CONT2
098 033E 3E20           CONT1   MVI   A,:20
099 0340 CD95D6         CONT2   CALL  OUTCPR    *  OUTPUT character
100 0343 3AF100                 LDA   :F1
101 0346 21E600                 LXI   H,:E6     *  ASCII-string from #E6 to
102 0349 47                     MOV   B,A       *  #E6 + (#F1)
103 034A FE01                   CPI   :1
104 034C CA5C03                 JZ    ZERO
105 034F 7E             PRLOOP  MOV   A,M       *  PRINT-Routine
106 0350 CD95D6         RETURN  CALL  OUTCPR    *  OUTPUT character
107 0353 23                     INX   H         *
108 0354 05                     DCR   B         *
109 0355 C24F03                 JNZ   PRLOOP    *
110 0358 F1                     POP   PSW
111 0359 D1                     POP   D
112 035A C1                     POP   B
113 035B C9                     RET
114 035C 7E             ZERO    MOV   A,M       *  If #F1=1 and Integer=0
115 035D FE30                   CPI   :30       *  then print "0" or space
116 035F C24F03                 JNZ   PRLOOP    *  NUL=#30 or NUL=#20
```

```
PAGE 03

117 0362 3A7003             LDA   NUL
118 0365 C35003             JMP   RETURN
119                         ORG   :380
120 0380 0D         MSG$    DATA  :0D       *  MESSAGE
121 0381 464F52             ASC   'FORMAT ERROR'
122 038D 00                 DATA  0
123 038E                    END

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

AANT   0371   CONT1  033E   CONT2  0340   KBRFL  02C4
MSG$   0380   NEW    0323   NUL    0370   OUTCPR D695
PMSG   DAD4   POUT   0300   PRLOOP 034F   PSTART 0333
RETURN 0350   XIBC   C027   ZERO   035C
```

```
PC UTILITY V3.3
>D300 38F

0300 C5 D5 F5 E7 0C CD 27 C0 21 71 03 56 14 3A F1 00
0310 BA DA 23 03 21 80 03 CD D4 DA 21 C4 02 36 FF F1
0320 D1 C1 C9 3A F1 00 15 BA CA 33 03 3E 20 CD 95 D6
0330 C3 23 03 3A E4 00 FE 2D C2 3E 03 C3 40 03 3E 20
0340 CD 95 D6 3A F1 00 21 E6 00 47 FE 01 CA 5C 03 7E
0350 CD 95 D6 23 05 C2 4F 03 F1 D1 C1 C9 7E FE 30 C2
0360 4F 03 3A 70 03 C3 50 03 FF FF FF FF FF FF FF FF
0370 FF FF FF FF FF FF FF FF FF FF FF FF BF BF FF FF
0380 0D 46 4F 52 4D 41 54 20 45 52 52 4F 52 00 FF FF
```

```
10    PRINT CHR$(12):CLEAR 100:DIM A(10.0):B=1
15    PRINT CHR$(14);"PRINT OUT OF INTEGERVARIABLES"
16    PRINT CHR$(14);"-----------------------------"
20    PRINT :PRINT " FORMAT = 6 figures"
30    PRINT " If A%=0 then print spaces":PRINT :GOTO 100
40    PRINT " DAI          DAInamic":PRINT " PRINT A%;    CALLM#300,A%":RETURN
50    FOR P=0 TO 5
55    PRINT
60    PRINT A(P),:REM DAI PRINT
65    CALLM #300,A(P):REM PRINT WITH MLP
70    NEXT P
75    RETURN
80    POKE #131,1:PRINT TAB(30);"Push any key to continue.";:POKE #131,0:RETURN
90    GC=GETC:IF GC=0 THEN 90:PRINT :PRINT CHR$(12)
92    RETURN
100   FOR P=0 TO 5
102   A(P)=INT(RND(10.0)*(10.0^P))
104   IF A(P)=0.0 THEN 102
106   NEXT P
110   FOR Q=0 TO 1
120   FOR P=0 TO 5:A(P)=INT(A(P)*(100.0^Q)+0.5):NEXT P
130   GOSUB 40
140   GOSUB 50
160   PRINT :PRINT "+--------   +--------"
170   A(10.0)=0:FOR P=0 TO 5:A(10.0)=A(10.0)+A(P):NEXT P
180   PRINT " ????????",:CALLM #300,A(10.0)
185   GOSUB 80
186   POKE #371,8:REM CHANGE FORMATLENGHT : 8 figures
187   GOSUB 90
188   IF Q=0.0 THEN PRINT :LIST 186:PRINT
189   NEXT Q
190   POKE #371,6:REM CHANGE FORMATLENGHT : 6 figures
210   LIST 190
215   PRINT :GOSUB 40
220   FOR P=0 TO 5:A(P)=B*P:B=B*10:NEXT P
230   GOSUB 50:PRINT :PRINT
240   LIST 250
245   PRINT
250   POKE #370,#30:REM IF VARIABLE% IS ZERO THEN PRINT: 0
260   GOSUB 40:FOR P=0 TO 5:PRINT :PRINT A(P),:CALLM #300,A(P):NEXT P
270   POKE #370,#20:REM IF VARIABLE% ( A%(P%) ) IS ZERO THEN PRINT SPACES
275   GOSUB 80
280   GOSUB 90
290   LIST 190:PRINT
300   A=12345678
305   LIST 310:PRINT
310   PRINT A,:REM DAI PRINT
311   PRINT TAB(15);"( Lenght of A% = 8 figures )"
312   PRINT
315   LIST 320
320   CALLM #300,A:REM PRINT WITH MLP
```

PRINT OUT OF INTEGERVARIABLES

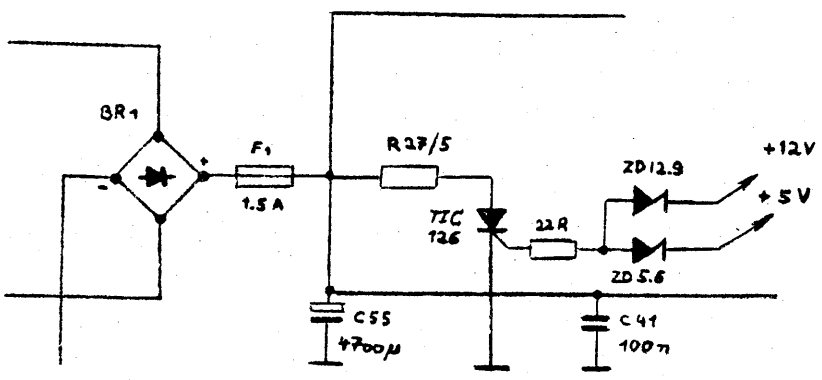
```
 FORMAT = 6 figures
 If A%=0 then print spaces

 DAI          DAInamic
 PRINT A%;    CALLM#300,A%

 5                 5
 65               65
 451             451
 5311           5311
 69630         69630
 395732       395732
+--------    +--------
 ????????     471194

186   POKE #371,8:REM CHANGE FORMATLENGHT : 8 figures

 DAI          DAInamic
 PRINT A%;    CALLM#300,A%

 500               500
 6500             6500
 45100           45100
 531100         531100
 6963004       6963004
 39573224     39573224
+--------    +--------
 ????????     47119428
190   POKE #371,6:REM CHANGE FORMATLENGHT : 6 figures

 DAI          DAInamic
 PRINT A%;    CALLM#300,A%

 0
 10               10
 200             200
 3000           3000
 40000         40000
 500000       500000

250   POKE #370,#30:REM IF VARIABLE% IS ZERO THEN PRINT: 0

 DAI          DAInamic
 PRINT A%;    CALLM#300,A%

 0                 0
 10               10
 200             200
 3000           3000
 40000         40000
 500000       500000
190   POKE #371,6:REM CHANGE FORMATLENGHT : 6 figures

310   PRINT A,:REM DAI PRINT

 12345678       ( Lenght of A% = 8 figures )

320   CALLM #300,A:REM PRINT WITH MLP

FORMAT ERROR
BREAK IN LINE 320
```

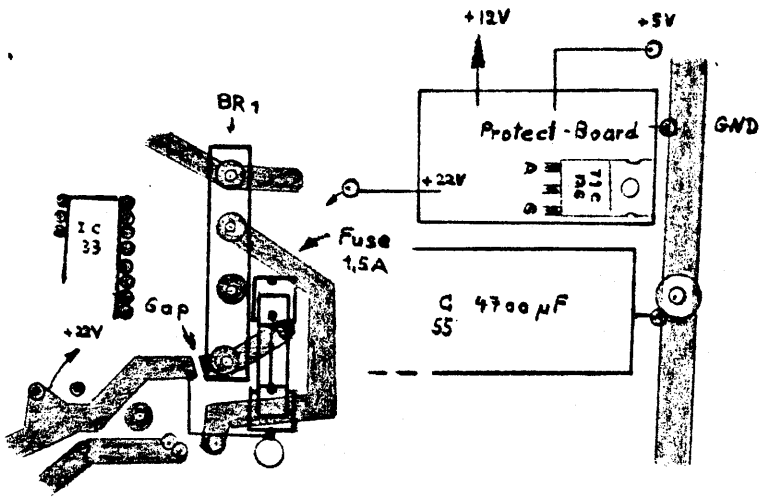
Netzteil u. Überspannungsschutz

Powersupply and Overvoltage-Protection.

Diese Schaltung und die Sicherung F1 verhindern im Falle eines Netzteildefekts große Schäden im Rechner, die durch die ungeregelte Gleichspannung von 22V verursacht werden könnte. Ebenso wird das bislang ungesicherte Netzteil vor Überlastung geschützt.
Überschreitet die Versorgunsspannung mehr als 0,9V ihres Normalwertes, zündet der Thyristor und die Sicherung wird ausgelöst.
Der Aufbau kann auf einer kleiner Veroplatine erfolgen. Diese kann vor dem Elko C55 im Netzteilkäfig eingebaut werden. Durch zwei kleine Bohrungen werden die Zuleiungen für Masse und die Eingangsspannung 22V mit der Grundplatine verbunden. Auf der Bestückungsseite umläuft eine breite Leiterbahn die Grundplatine. Von dieser Leiterbahn kann +5V zur Schutzschaltung geführt werden. Durch diese Anschlüsse wird die Schutzplatine gleichzeitig an ihrem Platz fixiert.
Die Sicherung wird neben dem Gleichrichter BR1 eingebaut. Dazu werden 4 kleine Löcher für die Halterung gebohrt. Die U+ Leiterbahn vom Gleichrichter BR1 wird unterbrochen und die Sicherung (1.5 Amp.) dazwischen geschaltet.

02.1983
r.corswandt





SIMULATION OF GOTO X / RUN X ONLY 7 BYTE

With the following program you can simulate GOTO X and it's also possible to do RUN (LINE NUMBER) in BASIC V1.0 without emptying the symbol table.

To get the program into the computer type in the following basicline:

```
*10 DIM JMP(1):JMP(0)=#232344:JMP(1)=#4DC363DF:JUMP=VARPTR(JMP(0))+1
```

Now JUMP contains the address and JMP(0),JMP(1) the program.

!!! Be careful if you have a CLEAR after executing the basicline JMP(0), JMP(1) and JUMP are zero !!!

You can use my program in the following ways:

1) as RUN X then type e.g. `*X=100:CALLM JUMP,X`
2) as GOTO X then type e.g.
```
*10 X=100
*20 CALLM JUMP,X:REM GOTO X
```

!!!!! Note: X, JUMP, JMP(0) and JMP(1) have to be integers. !!!!!

Just van Dunne'

Here follows the machine language program and a basic example.

```
23          INX H
23          INX H
44          MOV B,H
4D          MOV C,L
C3 63 DF    JMP DF63  (basiccmd. GOTO)
            END
```

```
5     MODE 4
10    DIM JMP(1):JMP(0)=#232344:JMP(1)=#4DC363DF:JUMP=VARPTR(JMP(0))+1
20    H=GETC:IF H<16 OR H>23 GOTO 20:X=(H-13)*10:CALLM JUMP,X
30    IF Q<YMAX THEN Q=Q+1
35    GOTO 110
40    IF Q>0 THEN Q=Q-1
45    GOTO 110
50    IF P>0 THEN P=P-1
55    GOTO 110
60    IF P<XMAX THEN P=P+1
65    GOTO 110
70    IF Q<YMAX-15 THEN Q=Q+15
75    GOTO 110
80    IF Q>15 THEN Q=Q-15
85    GOTO 110
90    IF P>15 THEN P=P-15
95    GOTO 110
100   IF P<XMAX-15 THEN P=P+15
110   DOT A,B 0:DOT P,Q 15:A=P:B=Q:GOTO 20
```

GOTO X / RUN X

```
30    REM TEST DE REGLAGE DE LECTURE DE CASSETTE
40    REM ALAIN MARIATTE
50    REM MAGNETOPHONES RADIOLA,BELL&HOWELL,SANYO
60    REM
70    REM
80    REM
90    REM
100   DIM A(0.0)
120   FOR I=1 TO 20
130   I$=STR$(I):L!=LEN(I$):N$=LEFT$(I$,L!-2):IF I<10 THEN N$=" "+N$
140   A$="   "+CHR$(137)+" "+N$+" "+"CASSETTE ADJUSTMENT TEST "+CHR$(136)
150   SAVEA A A$
160   NEXT
```

PROGRAMMING TECHNIQUES

(from DAInamic 11, page 173)

I am planning to provide a chapter in this journal about the problems and techniques of programming. The methods to be given will always be valid for the DAI and sometimes also for many other computers. The subjects offered arise from your suggestions and/or from weak spots in submitted programs. In this connexion I am thinking of such things as drawing circles quickly, avoiding conflicting colours, working with arrays, the speed and readability of programs, etc. For this first occasion my subject will be the construction of a list of finite but randomly dispersed numbers. The idea came from T. Groeneveld's BINGO program (DAInamic 10, page 136). I hope that Mr. Groeneveld, after reading this, will not imagine that people think he is a poor programmer. I certainly do not, particularly as he himself has found a solution. In his program an array is needed with the numbers from 1 to 75 inclusive, but the numbers must be stored in a random order. If you cannot make the program work properly try inputting it after an IMP INT command; that was not done originally as can be seen from the .0s From the original program :

```
10 CLEAR 1000: DIM G(76.0)
25 R=RND(75.0)+1.0
26 ENVELOPE 1 15,9;0: SOUND 1 1 15 0 FREQ(RND(269.0)+31.0)
27 NOISE 1 15
30 FOR A=1.0 TO 75.0: IF G(A)<>R THEN NEXT A
40 IF A<76.0 AND G(A)=R THEN 25
50 IF G(I)=0.0 THEN G(I)=R
65 I=I+1.0: IF I=76.0 THEN 100
70 GOTO 25
```

Let us analyse these lines:

In 25 we choose a suitable number
26 and 27 the waiting time is made pleasant
30 checks that the number chosen has not previously been selected
40 if so, we choose another
50 if the number is not already in the table we put it there
65 allocate the next number to the next location and check to see if we have them all yet
70 if not go back and choose the following one

As you can see the program works all right, but very slowly. On my machine (with the maths chip) I recorded an average of 72 seconds. What can be done about it ? Firstly change the program a little: I started by removing the .0s, took the NOISE and ENVELOPE out of the loop and coloured the sound with R*4 which is much faster than another RND. I improved the time a little, the average now being 66 seconds.

```
10 CLEAR 1000: DIM G(76.0)
16 ENVELOPE 1 15,9;0
17 NOISE 1 15
25 R=RND(75)+1
28 SOUND 1 1 15 0 FREQ(R*4+31)
30 FOR A=1 TO 75: IF G(A)<>R THEN NEXT A
40 IF A<76 AND G(A)=R THEN 25
50 IF G(I)=0 THEN G(I)=R
65 I=I+1: IF I=76 THEN 100
70 GOTO 25
```

Back again to the rebuilding! Multiple statements can be combined on the same line. The test of A<76 seems to me superfluous. The end test is the thing that is giving BASIC a bad name (spaghetti code) with its inverted jump combined with a GOTO address. The average time is now 58 seconds.

```
10 CLEAR 1000: DIM G(76)
16 ENVELOPE 1 15,9;0: NOISE 1 15
25 R=RND(75)+1: SOUND 1 1 15 0 FREQ(R*4+31)
30 FOR A=1 TO 75: IF G(A)<>R THEN NEXT A
40 IF G(A)=R THEN 25: IF G(I)=0 THEN G(I)=R
65 I=I+1: IF I<76 GOTO 25
```

Now only running the test, with the sound removed, brings the average to 48 seconds, but I find it better to wait 58 seconds with sound than 48 without. Later I saw that the loop only had to go to I. We still have too long to wait and so must try another tack.

The original program is sluggish because a random number has to be chosen and then checked to see if it has appeared before. It is better firstly to get the numbers we want and then to put them in a random place in the array. I have not worked this method out because it would only be turning our problem around. In the first method there is the difficulty of obtaining a suitable number while in the suggested alternative there is the difficulty of obtaining a suitable location. The new method certainly would be quicker because of the simpler checking (only looking to see if the place is empty instead of checking all the numbers). If I were to make the array much larger, say 256, then it should be easier to find an empty place in it. Afterwards, when I have all the numbers 1 to 75 in this auxilliary array I only have to transfer them neatly one after the other into array G(). The time gained is vast, resulting in a period of 3.4 seconds.

```
10 CLEAR 10000: DIM G(75),S(255)
20 FOR I=1 TO 75
30 R=RND(256): IF S(R)<>0 GOTO 30: S(R)=I: NEXT
40 FOR I=0 TO 255: IF S(I)<>0 THEN J=J+1: G(J)=S(I)
50 NEXT
```

But we have not finished yet. The method is best for an array with 75 elements, but less suitable as the quantity increases because it then becomes more difficult to find an empty place. Sometimes too the penalty of all that extra memory space is significant. If we do not mind the list not being truly random, then the following method may do: Place the numbers 1 to 75 randomly into the array of 75 locations. If the location is already occupied, do not choose a new random place but take the next empty one. I have not worked out these possibilities because of the non-random features. Now the fastest method which still has a short program is:

```
10 CLEAR 10000: DIM G(75)
20 FOR I=1 TO 75: G(I)=I: NEXT
30 FOR I=1 TO 75: G(0)=G(I): R=RND(75)+1: G(I)=G(R): G(R)=G(0): NEXT
```

We put the numbers 1 to 75 inclusive successively into the array and then exchange each number with another random element. We get a nice random list in a nice quick time: 2.4 seconds if we use G(0) as a parking place and even better, 2.36 seconds, when we use A. This latter is because the DAI has more difficulty in finding array variables. If anyone knows of a shorter or quicker method I would like to hear of it and will then certainly come back to the subject.

Frank H Druijff

0 zwart
1 blauw
2 d.rood
3 rood
4 paars
5 groen
6 d.bruin
7 l.bruin
8 grijs
9 blauw
10 oranje
11 rose
12 l.blauw
13 l.groen
14 geel
15 wit

alle adressen in HEXvorm!

29B-29C	start heap
29D-29E	size heap
29F-2A0	start text buffer
2A1-2A2	start symbol table
2A3-2A4	end of symbol table
2A5-2A6	bottom screen ram

75	cursor symbol
74	cursor mode
72-73	cursor position

40,28 cass motor 1 ON
40,18 cass motor 2 ON
40,30 1 and 2 OFF

131,0	output scrn+ RS232
131,1	screen only
131,2	edit buffer
135,2	read from edit buffer

MODE	XMAX	YMAX
1/2	71	64
3/4	159	129
5/6	335	255

MERGE
°CLEAR XXX
°LOAD"A"
°EDIT BREAK/BREAK
°LOAD"B"
°POKE 135,2

IMP INT *** IMP FPT
°IMP FPT
°CLEAR XXXX
°EDIT BREAK/BREAK
°IMP INT
°POKE 135,2

COLORG R1 R2 R3 R4
20 21 22 23
16 :R2*R1 R4*R3
17 :R1*R2 R3*R4
18 :R3*R1 R4*R2
19 :R1*R3 R2*R4

32K 7XXX
12K 2XXX
8K 1XXX

LIJN	CTRL	COLOR	LIJN	CTRL	COLOR
23	BFEF	BFEE	11	B9A7	B9A6
22	BF69	BF68	10	B921	B920
21	BEE3	BEE2	9	B89B	B89A
20	BE5D	BE5C	8	B815	B814
19	BDD7	BDD6	7	B78F	B78E
18	BD51	BD50	6	B709	B708
17	BCCB	BCCA	5	B683	B682
16	BC45	BC44	4	B5FD	B5FC
15	BBBF	BBBE	3	B577	B576
14	BB39	BB38	2	B4F1	B4F0
13	BAB3	BAB2	1	B46B	B46A
12	BA2D	BA2C	0	B3E5	B3E4

CTRL&COLOR BYTES IN A-MODE

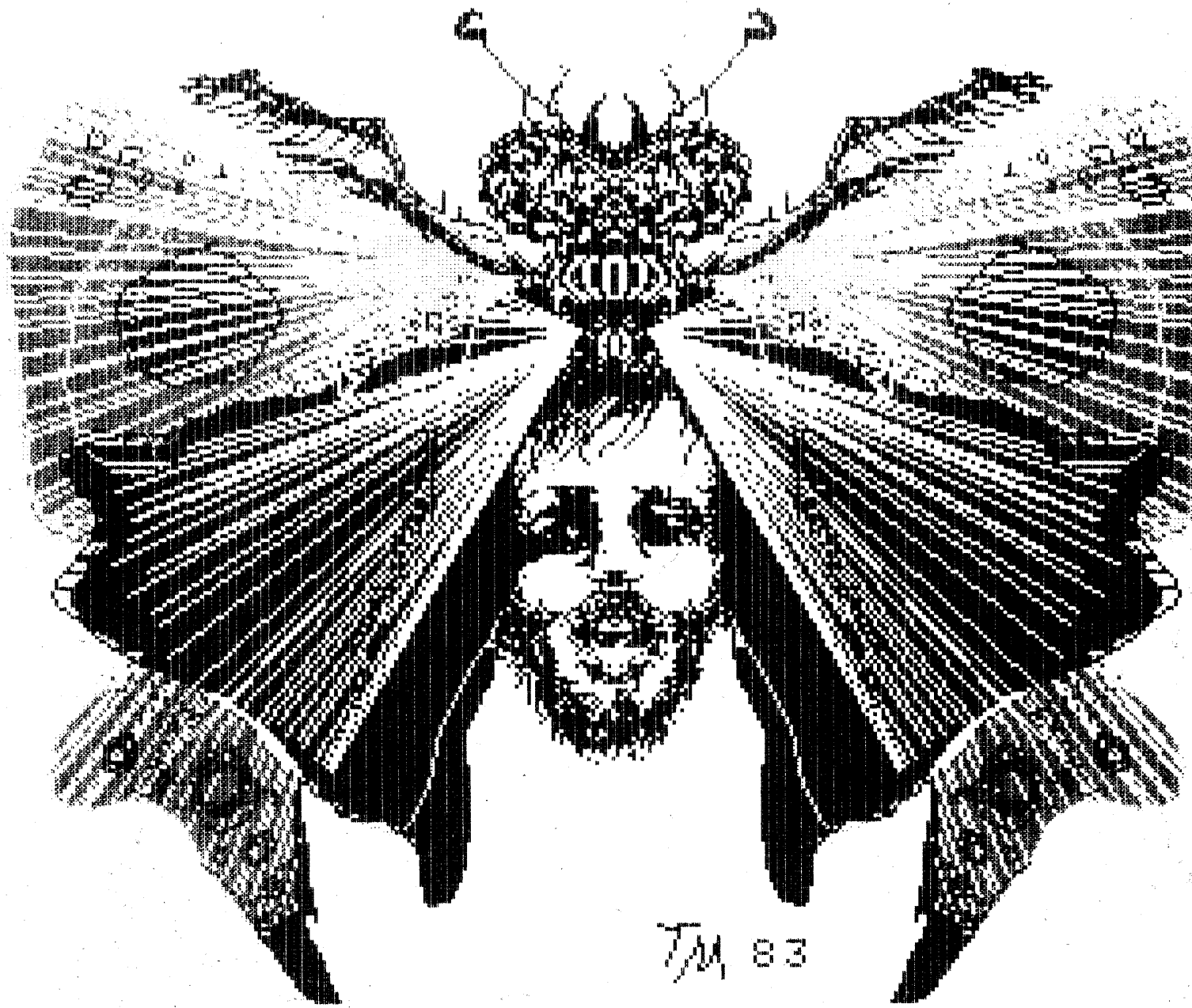
MODE	CTRL	COLOR	LIJN
1A/2A	BAE7	BAE6	3
	BA61	BA60	2
	B9DB	B9DA	1
	B955	B954	0
3A/4A	ACD3	ACD2	3
	AC4D	AC4C	2
	ABC7	ABC6	1
	AB41	AB40	0
5A/6A	7557	7556	3
	74D1	74D0	2
	744B	744A	1
	73C5	73C4	0

FD00 b2 page signal
b3 serial out rdy
b4 right paddle
b5 left paddle
b6 random data
b7 cass. input
FD01 Trigger paddle
FD04 0-3 volume ch.1(0)
4-7 volume ch.2(1)
FD05 0-3 volume ch.3(2)
4-7 volume noise
FD06 b0 cass.out
b1/2 paddle select
b3 paddle enable
b4 cass motor 1
b5 cass motor 2
b6/7 ROM BANK SWITCH

FF00 ser.inp.buff
FF01 b0-6 keyb.inp.
b7 in7 DCE
FF02 Interr.reg.
FF03 b1 frame error
b2 overrun error
b3 rec.buf.loaded
b4 trans.buf.empty
FF04 COMMAND REGISTER
FF05 BAUD RATE REGISTER
FF06 ser.out buf.
FF07 keyb.output
FF08 interr.mask reg.

TEST EVENT
PEEK(éFD00) IAND 32
PEEK(éFD00) IAND 16
PEEK(éFD00) IAND 48

FF09 TIMER 0
FF0A TIMER 1
FF0B TIMER 2
FF0C TIMER 3
FF0D TIMER 4
8253
CH 0 FC00/FC01
CH 1 FC02/FC03
CH 2 FC04/FC05
STATUS FC06/FC07


TM 83